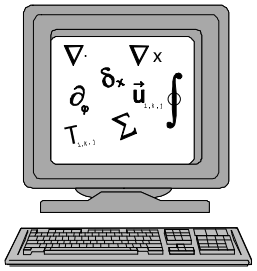
—Draft— Feb 2000 —

MOM 3.0 Manual

by

Ronald C. Pacanowski

Stephen M. Griffies



Contents

I Introduction to MOM and its use 1

1 Introduction 3
- 1.1 What is MOM? 3
- 1.2 Accessing the manual, code, and database 3
- 1.3 Minimum computational requirements 4
- 1.4 How this manual is organized 4
- 1.5 Special acknowledgments and disclaimers 5
  - 1.5.1 Acknowledgments 5
  - 1.5.2 Disclaimer 5
  - 1.5.3 Software license 5

2 A brief history of ocean model development at GFDL 7
- 2.1 Bryan-Cox-Semtner: 1965-1989 7
- 2.2 The GFDL Modular Ocean Models: MOM 1 and MOM 2: 1990-1995 7
- 2.3 MOM 3: 1996-1999 8
- 2.4 Documentation 8
  - 2.4.1 Main differences between MOM 2 and MOM 3 9
  - 2.4.2 Parallelization and Fortran 90 9
  - 2.4.3 Model physics and numerics 9
- 2.5 Main differences between MOM 1 and MOM 2 11
  - 2.5.1 Architecture 11
  - 2.5.2 Physics and analysis tools 12

3 Getting Started 15
- 3.1 How to find things in MOM 15
- 3.2 Directory Structure 15
- 3.3 The MOM Test Cases 19
  - 3.3.1 The run_mom script 20
- 3.4 Sample printout files 21
- 3.5 How to set up a model 22
- 3.6 Executing the model 24
- 3.7 Analyzing solutions 24
- 3.8 Executing on 32 bit workstations 24
- 3.9 NetCDF and time averaged data 25
- 3.10 Using Ferret 25
- 3.11 Upgrading from MOM 1 27
- 3.12 Upgrading to the latest version of MOM 27

3.12.1 The recommended method to incorporate personal changes 28
3.12.2 An alternative recommended method 28
3.13 Finding all differences between two versions of MOM 29
3.14 Applying bug fixes 29

II Basic formulation **33**

4 Fundamental equations **35**
4.1 Assumptions 35
4.2 The primitive equations 36
4.2.1 Basic constants and parameters 37
4.2.2 Hydrostatic pressure and the equation of state 38
4.2.3 Horizontal momentum equations 38
4.2.3.1 Coriolis force 38
4.2.3.2 Horizontal pressure gradient 38
4.2.3.3 Advection 38
4.2.3.4 Nonlinear advective "metric" term 39
4.2.3.5 Vertical friction 39
4.2.3.6 Horizontal friction 39
4.2.4 Tracer equations 39
4.3 Boundary and initial conditions 40
4.3.1 Bottom kinematic boundary condition 41
4.3.2 Surface kinematic boundary condition 41
4.3.3 Dynamic boundary conditions 42
4.3.4 Tracer fluxes through the model boundaries 43
4.3.5 Open boundaries and sponges 44
4.3.6 Initial conditions 44
4.4 Comments on volume versus mass conservation 44
4.4.1 Volume conservation 44
4.4.2 Mass conservation 45
4.4.3 Surface kinematic boundary conditions revisited 46
4.5 Flux form and finite volumes 47
4.6 Some basic formulae and notation 47
4.6.1 Differential operators 48
4.6.2 Leibnitz's Rule 49
4.6.3 Cross-products and the Levi-Civita symbol 49
4.6.4 Area element and volume element on a sphere 49
4.6.5 Vertical grid levels 49

5 Momentum equation methods **51**
5.1 Separation into vertical modes 51
5.1.1 Vertical modes in MOM and their relation to eigenmodes 52
5.1.2 Motivation for separating the modes 53
5.2 Methods for solving the separated equations 53
5.2.1 The fixed surface / rigid lid method in brief 54
5.2.1.1 Fixed surface height 54
5.2.1.2 Vanishing velocity at the ocean surface 55

5.2.1.3 Fresh water forcing in the rigid lid 55
5.2.1.4 Two rigid lid methods in MOM 55
5.2.2 The free surface / non-rigid lid method in brief 56
5.2.2.1 The barotropic equation and its two solution methods 56
5.2.2.2 The non-rigid lid approximation 56

6 Rigid lid streamfunction method 59
6.1 The barotropic streamfunction 59
6.2 Streamfunction and volume transport 60
6.3 Hydrostatic pressure with the rigid lid 60
6.4 The barotropic vorticity equation 61
6.4.1 Tendencies for the vertically averaged velocities 61
6.4.2 The barotropic vorticity equation 63
6.4.3 Caveat: inversions with steep topography 64
6.5 Boundary conditions and island integrals 64
6.5.1 Dirichlet boundary condition on the streamfunction 64
6.5.2 Separating the streamfunction's boundary value problem 65
6.5.3 Island integrals for the volume transport 66
6.6 The baroclinic mode 66
6.7 Summary of the rigid lid streamfunction method 67
6.8 Rigid lid surface pressure method 68

7 Free surface method 69
7.1 Hydrostatic pressure with the free surface 69
7.2 The barotropic system 71
7.2.1 Vertically integrated transport 71
7.2.2 Bottom and surface kinematic boundary conditions 72
7.2.3 Free surface height equation 72
7.2.4 Vertically integrated momentum equations 73
7.2.5 Global water budget 73
7.3 A linearized barotropic system 74
7.3.1 The barotropic system 74
7.3.2 The shallow water limit 75
7.3.3 The linearized free surface height equation 75
7.3.4 Summary of the linear barotropic system 76
7.4 Stresses at the ocean surface and bottom 77
7.4.1 Bottom stress 78
7.4.2 Surface stress 79
7.4.3 Revisiting the surface stress 80
7.5 A comment about atmospheric pressure 81
7.6 Vertically integrated transport 81
7.6.1 General considerations 81
7.6.2 An approximate streamfunction 81

8 The tracer budget 85
8.1 The continuum tracer concentration budget 85
8.2 Finite volume budget for the total tracer 85
8.3 Surface tracer flux 86

8.4 Comments on the surface tracer fluxes . . . 87
8.4.1 Fresh water flux into the free surface model . . . 88
8.4.2 Heat flux into the free surface model . . . 89

9 Momentum friction **91**
9.1 History of friction in MOM . . . 91
9.2 Basic properties of the stress tensor . . . 92
9.2.1 The deformation or rate of strain tensor . . . 92
9.2.2 Relating strain to stress . . . 94
9.2.3 Angular momentum and symmetry of the stress tensor . . . 94
9.3 The stress tensor in Cartesian coordinates . . . 95
9.3.1 Generalized Hooke's law form . . . 96
9.3.2 Angular momentum . . . 96
9.3.3 Dissipation of total kinetic energy . . . 96
9.3.4 Transverse isotropy . . . 97
9.3.5 Trace-free frictional stress . . . 98
9.3.6 Summary of the frictional stress tensor . . . 98
9.3.7 Quasi-hydrostatic assumption . . . 99
9.3.8 Cartesian form of the friction vector . . . 99
9.3.9 The case of nonconstant viscosity . . . 100
9.4 Orthogonal curvilinear coordinates . . . 100
9.4.1 Some rules of tensor analysis on manifolds . . . 101
9.4.2 Orthogonal coordinates . . . 104
9.4.3 Physical components of tensors . . . 104
9.4.4 General form of the frictional stress tensor . . . 105
9.4.5 Horizontal tension and shearing rate of strain . . . 106
9.4.6 The friction vector . . . 107
9.4.7 Effects on kinetic energy . . . 108
9.4.8 Summary of second order friction . . . 109
9.5 Biharmonic friction . . . 110
9.5.1 General formulation . . . 111
9.5.2 Effects on kinetic energy . . . 112
9.6 Comments on frictional and advective metric terms . . . 112
9.6.1 Motion on an infinite plane . . . 113
9.6.2 Conservation of angular momentum about the north pole . . . 114
9.6.3 The advective and frictional metric terms . . . 115
9.7 Functional formalism . . . 116
9.7.1 Continuum formulation . . . 116
9.7.2 Discrete formulation . . . 118
9.8 Old friction implementation . . . 118
9.8.1 Spherical form of second order friction . . . 118
9.8.2 Zonal friction . . . 119
9.8.3 Meridional friction . . . 120
9.8.4 Old biharmonic algorithm . . . 121

III Code design 123

10 Design Philosophy 125
10.1 Objective . . . 125
10.1.1 Speed . . . 125
10.1.2 Flexibility . . . 126
10.1.3 Modularity . . . 126
10.1.4 Documentation . . . 126
10.1.5 Coding efficiency. . . . 127
10.1.6 Ability to upgrade. . . . 127

11 Uni-tasking 129
11.1 Why memory management is important . . . 129
11.2 Minimizing the memory requirement . . . 130
11.2.1 Slicing through the 3-D prognostic data . . . 131
11.3 The Memory Window . . . 131
11.3.1 Detailed anatomy . . . 132
11.3.2 Solving prognostic equations within the MW. . . . 133
11.3.3 Moving the memory window . . . 135
11.3.4 Questions and Answers . . . 136

12 Multi-tasking 147
12.1 Scalability . . . 147
12.2 When to multi-task . . . 147
12.3 Approaches to multi-tasking . . . 148
12.4 The distributed memory paradigm . . . 148
12.5 Domain Decomposition . . . 149
12.5.1 Calculating row boundaries on processors . . . 150
12.5.2 Communications . . . 151
12.5.3 The barotropic solution . . . 152

13 Database 157
13.1 Data files . . . 157

14 Variables 159
14.1 Naming convention for variables . . . 159
14.2 The main variables . . . 160
14.2.1 Relating indices j and jrow . . . 160
14.2.2 Cell faces . . . 160
14.2.3 Model size parameters . . . 161
14.2.4 T cells . . . 161
14.2.5 U cells . . . 162
14.2.6 Vertical spacing . . . 162
14.2.7 Time level indices . . . 162
14.2.8 3-D Prognostic variables . . . 163
14.2.9 2-D Prognostic variables . . . 163
14.2.10 3-D Workspace variables . . . 164
14.2.11 3-D Masks . . . 165

14.2.12 Surface Boundary Condition variables . . . 165
14.2.13 2-D Workspace variables . . . 165
14.3 Operators . . . 166
14.3.1 Tracer Operators . . . 166
14.3.2 Momentum Operators . . . 167
14.4 Input Namelist variables . . . 168
14.4.1 Time and date . . . 168
14.4.2 Integration control . . . 169
14.4.3 Surface boundary conditions . . . 170
14.4.4 Time steps . . . 170
14.4.5 External mode . . . 171
14.4.6 Mixing . . . 172
14.4.7 Diagnostic intervals . . . 173
14.4.8 Directing output . . . 175
14.4.9 Isoneutral diffusion . . . 176
14.4.10 Nonconstant isoneutral diffusivities . . . 176
14.4.11 Pacanowski/Philander mixing . . . 177
14.4.12 Smagorinsky mixing . . . 177
14.4.13 Bryan/Lewis mixing . . . 177

15 Modules and Modularity 179
15.1 List of Modules . . . 179
15.1.1 convect.F . . . 179
15.1.2 denscoef.F and MOM's density . . . 180
15.1.2.1 Bryan and Cox 1972 . . . 180
15.1.2.2 Computing density within MOM . . . 180
15.1.2.3 *in situ* density and potential density . . . 182
15.1.2.4 Linearized density and option linearized_density . . . 182
15.1.3 grids.F . . . 183
15.1.4 iomngr.F . . . 183
15.1.5 poisson.F . . . 184
15.1.6 vmix1d.F . . . 185
15.1.7 timeinterp.F . . . 185
15.1.8 timer.F . . . 185
15.1.9 Time manager . . . 186
15.1.9.1 Introduction . . . 186
15.1.9.2 Overview of interfaces . . . 186
15.1.9.3 Time interfaces . . . 186
15.1.9.4 Calendar Interfaces . . . 188
15.1.9.5 Sample test program . . . 190
15.1.9.6 Logical Switches . . . 192
15.1.10 topog.F . . . 193
15.1.11 util.F . . . 194
15.1.11.1 indp . . . 194
15.1.11.2 ftc . . . 194
15.1.11.3 ctf . . . 194
15.1.11.4 extrap . . . 194
15.1.11.5 setbcx . . . 194

15.1.11.6 iplot . . . 194
15.1.11.7 imatrix . . . 194
15.1.11.8 matrix . . . 194
15.1.11.9 scope . . . 195
15.1.11.10 sum1st . . . 195
15.1.11.11 plot . . . 195
15.1.11.12 checksum . . . 195
15.1.11.13 print_checksum . . . 195
15.1.11.14 wrufio . . . 195
15.1.11.15 rrufio . . . 195
15.1.11.16 tranlon . . . 195

IV Grids, Geometry, and Topography **197**

16 Grids **199**
16.1 Domain and Resolution . . . 199
16.1.1 Regions . . . 200
16.1.2 Resolution . . . 200
16.1.3 Describing a domain and resolution . . . 200
16.1.3.1 Example 1: One resolution domain . . . 201
16.1.3.2 Example 2: Two resolution domains . . . 202
16.1.3.3 Example 3: Horizontally isotropic grid . . . 202
16.2 Grid cell arrangement . . . 203
16.2.1 Relation between T and U cells . . . 203
16.2.2 Regional and domain boundaries . . . 203
16.2.3 Non-uniform resolution . . . 204
16.2.3.1 Accuracy of numerics . . . 205
16.3 Constructing a grid . . . 205
16.3.1 Grids in two dimensions . . . 206
16.4 Summary of options . . . 207

17 Grid Rotation **213**
17.1 Defining the rotation . . . 213
17.2 Rotating Scalars and Vectors . . . 215
17.3 Considerations . . . 215

18 Topography and geometry **217**
18.1 Designing topography and geometry . . . 217
18.2 Options for constructing the KMT field . . . 218
18.3 Meta land masses . . . 220
18.4 Modifications to KMT . . . 220
18.4.1 Altering the code . . . 220
18.4.2 Directly editing the KMT field . . . 221
18.5 Topographic instability . . . 221
18.6 Viewing results . . . 222
18.7 Summary of options for topography . . . 222

V Boundary Conditions 227

19 Generalized Surface Boundary Condition Interface 229
19.1 Coupling to atmospheric models . . . 229
19.1.1 GASBC . . . 230
19.1.1.1 SST outside Ocean domain . . . 231
19.1.1.2 Interpolations to atmos grid . . . 231
19.1.2 GOSBC . . . 232
19.1.2.1 Interpolations to ocean grid . . . 232
19.2 Coupling to datasets . . . 233
19.2.1 Bulk parameterizations . . . 235
19.3 Surface boundary conditions . . . 235
19.3.1 Default Surface boundary conditions . . . 236
19.3.2 Adding or removing surface boundary conditions . . . 237

20 Stevens Open Boundary Conditions 243
20.1 Boundary specifications . . . 244
20.2 Options . . . 245
20.3 New Files . . . 245
20.4 Important changes to existing subroutines . . . 246
20.5 Data Preparation Routines . . . 247
20.6 TO-DO List (How to set up open boundaries) . . . 247

VI Finite Difference Equations 251

21 The Discrete Equations 253
21.1 Time and Space discretizations . . . 253
21.1.1 Averaging operators . . . 253
21.1.2 Derivative operators . . . 253
21.2 Key to understanding finite difference equations . . . 254
21.2.1 Rules for manipulating operators . . . 255
21.2.2 Rules involving summations . . . 256
21.2.3 Other rules . . . 256
21.3 Primitive finite difference equations . . . 257
21.3.1 Momentum equations . . . 257
21.3.2 Tracer equations . . . 259
21.4 Time Stepping Schemes . . . 260
21.4.1 Leapfrog . . . 260
21.4.2 Forward . . . 261
21.4.3 Euler Backward . . . 261
21.4.4 Robert time filter . . . 262

22 Solving the Discrete equations 265
22.1 Start of computation within Memory Window . . . 265
22.2 loadmw (load the memory window) . . . 266
22.2.1 Land/Sea masks . . . 266
22.2.2 Reading latitude rows into the Memory window . . . 266

22.2.3 Constructing the total velocity . . . 266
22.2.4 Computing quantities within the memory window . . . 267
22.2.4.1 Example 1: density . . . 268
22.2.4.2 Example 2: Advective velocity on the eastern face of T-cells . . 269
22.2.4.3 Example 3: Advective velocity on the bottom face of U-cells . . 269
22.3 adv_vel (computes advective velocities) . . . 269
22.3.1 Advective velocities for T cells . . . 269
22.3.2 Advective velocities for U cells . . . 271
22.3.3 Vertical velocity on the ocean bottom . . . 272
22.3.3.1 Summary of the continuum results . . . 272
22.3.3.2 Discrete vertical velocity at the ocean bottom . . . 273
22.4 isopyc (computes isoneutral mixing tensor components) . . . 275
22.5 vmixc (computes vertical mixing coefficients) . . . 275
22.6 hmixc (computes horizontal mixing coefficients) . . . 276
22.7 setvbc (set vertical boundary conditions) . . . 276
22.8 tracer (computes tracers) . . . 276
22.8.1 Tracer components . . . 277
22.8.2 Advective and Diffusive fluxes . . . 277
22.8.3 Isoneutral fluxes . . . 278
22.8.4 Source terms . . . 278
22.8.5 Sponge boundaries . . . 279
22.8.6 Shortwave solar penetration . . . 279
22.8.7 Tracer operators . . . 279
22.8.7.1 Implicit vertical diffusion . . . 280
22.8.7.2 Isoneutral mixing . . . 280
22.8.7.3 Gent-McWilliams advection velocities . . . 280
22.8.8 Solving for the tracer . . . 281
22.8.8.1 Explicit vertical diffusion . . . 281
22.8.8.2 Implicit vertical diffusion . . . 282
22.8.9 Diagnostics . . . 282
22.8.10 End of tracer components . . . 282
22.8.11 Explicit Convection . . . 282
22.8.12 Filtering . . . 283
22.8.13 Accumulating $sbcocn_{i,jrow,m}$ . . . 283
22.9 baroclinic (computes internal mode velocities) . . . 283
22.9.1 Hydrostatic pressure gradient terms . . . 283
22.9.2 Momentum components . . . 284
22.9.3 Advective and Diffusive fluxes . . . 284
22.9.4 Source terms . . . 285
22.9.5 Momentum operators . . . 286
22.9.5.1 Coriolis treatment . . . 287
22.9.6 Solving for the time derivative of velocity . . . 287
22.9.6.1 Explicit vertical diffusion . . . 287
22.9.6.2 Implicit vertical diffusion . . . 287
22.9.7 Diagnostics . . . 288
22.9.8 Vertically averaged time derivatives of velocity . . . 288
22.9.9 End of momentum components . . . 289
22.9.10 Computing the internal modes of velocity . . . 289

22.9.10.1 Explicit Coriolis treatment . . . 289
22.9.10.2 Semi-implicit Coriolis treatment . . . 289
22.9.11 Filtering . . . 290
22.9.12 Accumulating $sbcocn_{i,jrow,m}$ . . . 290
22.10 End of computation within Memory Window . . . 290
22.11 barotropic (computes external mode velocities) . . . 290
22.12 diago . . . 291

VII General model options **297**

23 Options for testing modules **301**
23.1 test_convect . . . 301
23.2 drive_denscoef . . . 301
23.3 drive_grids . . . 301
23.4 test_iomngr . . . 301
23.5 test_poisson . . . 302
23.6 test_vmix . . . 302
23.7 test_rotation . . . 302
23.8 test_timeinterp . . . 302
23.9 test_timer . . . 302
23.10 test_tmngr . . . 302
23.11 drive_topog . . . 302
23.12 test_util . . . 302

24 Options for the computational environment **303**
24.1 Computer platform . . . 303
24.1.1 cray_ymp . . . 303
24.1.2 cray_c90 . . . 303
24.1.3 cray_t90 . . . 303
24.1.4 cray_t3e . . . 303
24.1.5 sgi . . . 303
24.2 Compilers . . . 303
24.3 Dataflow I/O Options . . . 304
24.3.1 ramdrive . . . 304
24.3.2 crayio . . . 305
24.3.3 ssread_sswrite . . . 305
24.3.4 fio . . . 305
24.4 Parallelization . . . 305
24.4.1 parallel_1d . . . 305

25 Options for grid, geometry and topography **307**
25.1 Grid generation . . . 307
25.1.1 drive_grids . . . 307
25.1.2 generate_a_grid . . . 307
25.1.3 read_my_grid . . . 307
25.1.4 write_my_grid . . . 307
25.1.5 centered_t . . . 307

25.2 Grid Transformations . . . 308
25.2.1 rot_grid . . . 308
25.3 Topography and geometry generation . . . 308
25.3.1 rectangular_box . . . 308
25.3.2 idealized_kmt . . . 308
25.3.3 gaussian_kmt . . . 308
25.3.4 scripps_kmt . . . 308
25.3.5 etopo_kmt . . . 309
25.3.6 read_my_kmt . . . 309
25.3.7 write_my_kmt . . . 309
25.3.8 flat_bottom . . . 309
25.3.9 fill_isolated_cells . . . 309
25.3.10 fill_shallow . . . 309
25.3.11 deepen_shallow . . . 309
25.3.12 round_shallow . . . 309
25.3.13 fill_perimeter_violations . . . 309
25.3.14 widen_perimeter_violations . . . 310

26 Partial Bottom Cells 311
26.1 Motivation . . . 311
26.2 Discrete Equations . . . 312
26.2.1 Momentum equations . . . 312
26.2.2 Pressure gradient . . . 314
26.2.2.1 Example where density varies linearly with depth . . . 315
26.2.2.2 Computing density in partial bottom cells . . . 316
26.2.3 Tracer equations . . . 316
26.3 Conservation of energy . . . 317
26.3.1 Changes in Kinetic energy due to partial bottom cells . . . 317
26.3.2 Additional kinetic energy change due to boundary effects . . . 318
26.3.3 Changes in Potential energy due to partial bottom cells . . . 319

27 Filtering 325
27.1 Convergence of meridians . . . 325
27.1.1 fourfil . . . 325
27.1.2 firfil . . . 327
27.1.3 An analysis of polar filtering . . . 329
27.1.4 Recommendation for tuning the polar filter . . . 330
27.2 Inertial period . . . 331
27.2.1 damp_inertial_oscillation . . . 331

28 Initial and boundary conditions 333
28.1 Initial Conditions . . . 333
28.1.1 ideal_thermocline . . . 333
28.1.2 ideal_pycnocline . . . 333
28.1.3 idealized_ic . . . 334
28.1.4 levitus_ic . . . 334
28.2 Surface Boundary Conditions . . . 335
28.2.1 simple_sbc . . . 335

28.2.2 constant_taux . . . 335
28.2.3 constant_tauy . . . 335
28.2.4 analytic_zonal_winds . . . 335
28.2.5 linear_tstar . . . 336
28.2.6 time_mean_sbc_data . . . 337
28.2.7 time_varying_sbc_data . . . 337
28.2.8 coupled . . . 337
28.2.9 restorst . . . 337
28.2.10 shortwave . . . 338
28.2.11 minimize_sbc_memory . . . 339
28.3 Lateral Boundary Conditions . . . 339
28.3.1 cyclic . . . 339
28.3.2 solid_walls . . . 339
28.3.3 symmetry . . . 340
28.3.4 sponges . . . 340
28.3.5 obc . . . 340

29 Old options for the external mode 345
29.1 Concerning which external mode option to use . . . 345
29.1.1 Wave processes . . . 345
29.1.2 Surface tracer fluxes . . . 346
29.1.3 Killworth topographic instability . . . 346
29.1.4 Wave speed considerations . . . 347
29.1.5 Polar filtering . . . 347
29.1.6 Parallelization . . . 348
29.2 stream_function . . . 348
29.2.1 The equation . . . 349
29.2.2 The coefficient matrices . . . 351
29.2.3 Solving the equation . . . 352
29.2.4 Island equations . . . 352
29.2.4.1 Another approach . . . 353
29.2.5 Symmetry in the stream function equation . . . 354
29.2.5.1 Symmetry of the explicit equations . . . 355
29.2.5.2 Anti-symmetry of the implicit Coriolis terms . . . 355
29.2.5.3 Island equations and symmetry . . . 355
29.2.5.4 Asymmetry of the barotropic equations in MOM 1 . . . 356
29.2.6 zero_island_flow . . . 356
29.3 rigid_lid_surface_pressure . . . 357
29.3.1 The equations . . . 357
29.3.2 Remarks . . . 358
29.3.2.1 Boundary conditions . . . 358
29.3.2.2 Conditioning of the elliptic operator . . . 358
29.3.2.3 Non-divergent barotropic velocities . . . 358
29.3.2.4 Polar filtering . . . 359
29.3.2.5 Checkerboarding in surface pressure . . . 359
29.4 implicit_free_surface . . . 359
29.4.1 The equations . . . 359
29.4.1.1 Modifications for various kinds of time steps . . . 361

29.4.2 Remarks . . . 362
29.4.2.1 Boundary conditions . . . 362
29.4.2.2 Conditioning with topography . . . 362
29.4.2.3 Barotropic velocities . . . 362
29.4.2.4 Polar filtering . . . 362
29.4.2.5 Checkerboarding in surface pressure . . . 362
29.5 The Killworth *et al* explicit_free_surface . . . 363
29.5.1 The numerical implementation . . . 363
29.5.1.1 Time stepping . . . 363
29.5.1.2 The delplus - delcross filter . . . 364
29.5.1.3 Interaction with subroutine *baroclinic* . . . 366
29.5.2 Energy analysis . . . 366
29.5.3 Options . . . 366
29.5.4 Compatibility with other model options . . . 367
29.5.5 Test cases . . . 367
29.5.6 Open boundary conditions and river inflow . . . 367

30 Explicit free surface and fresh water 369
30.1 Free surface options . . . 369
30.2 Momentum equations . . . 370
30.3 Time stepping algorithm . . . 373
30.4 Vertical velocities . . . 378
30.5 Comments on the algorithm . . . 379
30.6 Discrete tracer budgets . . . 379
30.7 Time discretization of the tracer budgets . . . 380
30.8 Further comments on surface fluxes and the case of salt . . . 380
30.9 Discrete conservation properties . . . 382
30.9.1 Volume conservation . . . 382
30.9.2 Energetic consistency . . . 383
30.9.3 Tracer quasi-conservation . . . 383
30.10 Detailed treatment of surface tracer budgets . . . 384
30.10.1 Summary of the surface tracer fluxes . . . 385
30.10.1.1 Flux between the ocean model and other model components . 385
30.10.1.2 Surface flux in the ocean model . . . 385
30.10.2 Advection and diffusion on different time slices . . . 386
30.10.3 Multiple sources and sinks of fresh water . . . 387
30.10.4 The special case of salt . . . 388
30.10.5 Neutral tracer fluxes . . . 388
30.11 Implementation of fresh water fluxes and rivers in MOM . . . 389
30.11.1 How river fluxes are input to MOM . . . 389
30.11.2 Approximations for the surface boundary conditions . . . 389
30.11.3 New files and changed subroutines . . . 390
30.11.4 Changed and new variables for the surface boundary conditions . . . 391
30.11.5 Data flow between the model components . . . 391
30.11.6 New model options . . . 392
30.11.7 The river code . . . 392
30.11.7.1 Files . . . 393
30.11.7.2 Setup of the river geometry . . . 393

30.11.7.3 The river - ocean interface . . . 394
30.11.7.4 Time dependent fresh water and tracer data management . . . 394
30.11.7.5 Initializing the river procedures . . . 396
30.11.8 The time interpolation . . . 397
30.11.9 Limitations of the river code . . . 397
30.12 Checkerboard null mode . . . 400
30.12.1 Experiences with the checkerboard null mode . . . 400
30.12.2 A caveat concerning filtering the surface height . . . 401
30.12.3 Suggestions . . . 402
30.13 Polar filtering . . . 402

31 Options for solving elliptic equations 403
31.1 conjugate gradient . . . 403
31.2 sf_9_point . . . 405
31.3 sf_5_point . . . 407

32 Options for advecting tracers 409
32.1 Considerations of accuracy in one-dimension . . . 409
32.1.1 Lattice and continuum operators . . . 410
32.1.2 Leap frog in time and centered in space . . . 411
32.1.3 A critique of upwind advection . . . 412
32.2 second_order_tracer_advection . . . 414
32.3 linearized_advection . . . 415
32.4 fourth_order_tracer_advection . . . 415
32.5 quicker . . . 416
32.6 fct . . . 418
32.6.1 Sub-options fct_dlm1 and fct_dlm2 . . . 421
32.6.2 Sub-option fct_3d . . . 423
32.7 bottom_upwind . . . 424

33 Vertical SGS options 425
33.1 Vertical convection . . . 425
33.1.1 Summary of the vertical convection options . . . 426
33.1.2 Explicit convection . . . 427
33.1.2.1 The standard Cox 1984 scheme: *oldconvect* . . . 428
33.1.2.2 Marotzke's scheme . . . 428
33.1.2.3 The fast way: MOM default explicit convection . . . 428
33.1.2.4 Discussion . . . 429
33.2 Vertical SGS mixing schemes . . . 431
33.2.1 constvmix . . . 431
33.2.2 bryan_lewis_vertical . . . 431
33.2.3 kppvmix . . . 432
33.2.3.1 Vertical discretization . . . 432
33.2.3.2 Semi-implicit time integration . . . 434
33.2.3.3 Diagnostic output . . . 436
33.2.4 ppvmix . . . 437
33.2.4.1 Richardson number . . . 437
33.2.4.2 Vertical mixing coefficients . . . 438

33.2.4.3 Adjustable parameters . . . 438
33.2.5 tcvmix . . . 439

34 Horizontal SGS options 441
34.1 Summary of the options . . . 441
34.1.1 Horizontal tracer mixing options . . . 441
34.1.2 Horizontal velocity mixing options . . . 443
34.2 Some numerical constraints . . . 444
34.2.1 Balance between advection and diffusion . . . 444
34.2.2 Linear stability of the diffusion equation . . . 446
34.2.2.1 Laplacian mixing . . . 446
34.2.2.2 Biharmonic mixing . . . 447
34.2.3 Western boundary currents . . . 448
34.2.4 Summary: viscosity on the sphere . . . 448
34.3 A comment on mixing and finite impulse filtering . . . 450
34.4 Comparing Laplacian and biharmonic mixing . . . 451
34.5 bryan_lewis_horizontal . . . 452
34.6 Variable horizontal mixing coefficients . . . 453
34.6.1 Discretization of the new metric terms . . . 453
34.6.2 am_cosine . . . 453
34.6.3 am_taper_highlats . . . 453
34.7 The Smagorinsky scheme . . . 454
34.7.1 General ideas . . . 454
34.7.2 Choosing the scaling coefficient . . . 456
34.7.3 Scaling coefficient conventions . . . 457
34.7.4 Smagorinsky and isoneutral mixing together . . . 457
34.7.5 Biharmonic Smagorinsky . . . 458
34.7.6 Discretization of the Smagorinsky viscosity coefficient . . . 458
34.7.7 Diffusive terms for the tracer equation . . . 460
34.8 tracer_horz_laplacian . . . 460
34.9 tracer_horz_biharmonic . . . 461
34.10 velocity_horz_laplacian . . . 462
34.11 velocity_horz_biharmonic . . . 463
34.12 velocity_horz_friction_operator . . . 464

35 Isoneutral SGS options 465
35.1 Basic isoneutral schemes . . . 465
35.1.1 A note about MOM3 updates . . . 465
35.1.2 Summary of the isoneutral mixing schemes . . . 465
35.1.3 Summary of the options and namelist parameters . . . 467
35.1.4 Some caveats and comments . . . 469
35.1.5 redi_diffusion . . . 470
35.1.5.1 Zonal isoneutral diffusion flux . . . 470
35.1.5.2 Meridional isoneutral diffusion flux . . . 471
35.1.5.3 Vertical isoneutral diffusion flux . . . 472
35.1.6 gent_mcwilliams . . . 472
35.1.6.1 gm_skew . . . 473
35.1.6.2 gm_advect . . . 473

35.1.7 Linear numerical stability for Redi and GM . . . 475
35.1.8 biharmonic_rm . . . 475
35.1.8.1 The RM98 operator . . . 476
35.1.8.2 RM98 for a special vertical profile . . . 477
35.1.8.3 Effects on potential energy of the RM98 operator . . . 478
35.1.8.4 Effects on potential energy of an operator suggested by Gent . 479
35.1.8.5 A note about spherical coordinates and extra metric terms . . . 479
35.1.8.6 Linear numerical stability for the RM98 operator . . . 482
35.1.8.7 Choosing the biharmonic coefficient . . . 483
35.1.8.8 Discretization details for the RM98 operator . . . 483
35.1.9 Isoneutral mixing and steep sloped regions . . . 484
35.1.9.1 dm_taper . . . 485
35.1.9.2 gkw_taper . . . 486
35.1.9.3 isotropic_mixed . . . 486
35.2 Schemes with nonconstant diffusivities . . . 487
35.2.1 hl_diffusivity . . . 488
35.2.1.1 The thermal wind Richardson number and the depth range . . 488
35.2.1.2 The effective β parameter . . . 489
35.2.1.3 Smoothing and temporal frequency of computation . . . 490
35.2.1.4 Summary of namelist parameters . . . 491
35.2.2 vmhs_diffusivity . . . 492
35.2.2.1 Time scale same as Held and Larichev . . . 492
35.2.2.2 Length scale based on baroclinic zone width . . . 492
35.2.2.3 Diffusivity and the basic tunable parameter . . . 493
35.2.2.4 Smoothing and temporal frequency of computation . . . 494
35.2.2.5 Summary of namelist parameters . . . 494
35.2.3 Held and Larichev combined with Visbeck *et al.* . . . 494
35.2.4 Netcdf information for nonconstant diffusivities . . . 495

36 Miscellaneous SGS options 497
36.1 Eddy-topography interactions and neptune . . . 497
36.2 xlandmix . . . 498
36.2.1 Formulation . . . 498
36.2.2 Considerations . . . 498
36.2.3 xlandmix_eta . . . 499

37 Bottom Boundary Layer 501

38 Miscellaneous options 503
38.1 max_window . . . 503
38.2 knudsen . . . 503
38.3 pressure_gradient_average . . . 503
38.4 fourth_order_memory_window . . . 504
38.5 implicitvmix . . . 505
38.6 beta_plane . . . 507
38.7 f_plane . . . 507
38.8 source_term . . . 507
38.9 readrmsk . . . 507

38.10 show_details . . . 507
38.11 timing . . . 507
38.12 equivalence_mw . . . 507

VIII Diagnostic options **509**

39 Design of diagnostic options **511**
39.1 Ferret . . . 511
39.2 Naming Diagnostic files . . . 512
39.3 Format of diagnostic data files . . . 512
39.4 Sampling data . . . 512
39.5 Regional masks . . . 513
39.6 A note about areas on the sphere . . . 514

40 Diagnostics for physical analysis **515**
40.1 cross_flow_netcdf . . . 515
40.1.1 Continuous formulation . . . 515
40.1.2 Discretization . . . 516
40.2 density_netcdf . . . 517
40.3 diagnostic_surf_height . . . 518
40.4 energy_analysis . . . 519
40.5 fct_netcdf . . . 521
40.6 gyre_components . . . 521
40.7 local_potential_density_terms . . . 523
40.7.1 Locally referenced potential density equation . . . 523
40.7.1.1 Cabbeling, thermobaricity, and halobaricity . . . 524
40.7.1.2 Summary of the terms forcing locally referenced potential density 527
40.7.2 Discretization . . . 527
40.7.2.1 Equation of state considerations . . . 528
40.7.2.2 Advection . . . 529
40.7.2.3 Vertical diffusion . . . 529
40.7.2.4 Laplacian horizontal diffusion . . . 530
40.7.2.5 Laplacian skew-diffusion . . . 530
40.7.2.6 Biharmonic skew-diffusion . . . 531
40.7.2.7 Cabbeling, thermobaricity, halobaricity, and partial cells . . . 531
40.7.2.8 Cabbeling . . . 531
40.7.2.9 Thermobaricity and halobaricity . . . 532
40.7.3 Output . . . 533
40.8 matrix_sections . . . 533
40.9 meridional_overturning . . . 533
40.9.1 Thickness equation . . . 534
40.9.2 Zonally integrated circulation and its streamfunction . . . 534
40.9.3 Overturning streamfunction . . . 536
40.9.4 Comments on the free surface overturning streamfunction . . . 537
40.9.5 Overturning streamfunction in the (ϕ, z) plane . . . 538
40.9.6 Overturning streamfunction in the (ϕ, θ) plane . . . 539
40.9.7 Overturning streamfunction in the $(\phi, \rho^{(0)})$ plane . . . 540

40.9.8 Overturning streamfunction in the $(\phi, \rho^{(p)})$ plane . . . 540
40.9.9 Overturning streamfunction in the $(\phi, \rho^{neutral})$ plane . . . 540
40.9.10 Discrete vertical-meridional streamfunction . . . 540
40.9.11 Discrete density-meridional streamfunction . . . 541
40.9.12 Option *merid_by_basin* . . . 541
40.9.13 Output . . . 541
40.10 meridional_tracer_budget . . . 542
40.11 monthly_averages . . . 544
40.12 save_convection . . . 544
40.13 save_mixing_coeff . . . 544
40.14 show_external_mode . . . 545
40.15 show_zonal_mean_of_sbc . . . 546
40.16 snapshots . . . 546
40.17 term_balances . . . 547
40.17.1 Momentum Equations . . . 549
40.17.2 Tracer Equations . . . 550
40.18 time_averages . . . 552
40.19 time_step_monitor . . . 553
40.20 topog_diagnostic . . . 554
40.21 tracer_averages . . . 555
40.22 tracer_yz . . . 556
40.23 trajectories . . . 556
40.24 save_xbts . . . 557
40.24.1 Momentum Equations . . . 558
40.24.2 Tracer Equations . . . 560

41 Diagnostics for numerical analysis **563**
41.1 General debug options . . . 563
41.2 stability_tests . . . 563
41.3 trace_coupled_fluxes . . . 565
41.4 trace_indices . . . 565

IX Appendices and references **567**

A Kinetic energy budget **569**
A.1 Continuum version of the kinetic energy budget . . . 569
A.1.1 The kinetic energy density . . . 569
A.1.2 External and internal mode kinetic energies . . . 570
A.1.3 Budget for the local kinetic energy . . . 571
A.1.4 Budget for the volume averaged kinetic energy and kinetic energy density 572
A.1.4.1 Budget for the kinetic energy within a vertical column . . . 572
A.1.4.2 Interpreting the terms in the kinetic energy budget . . . 573
A.1.4.3 Budget for the averaged kinetic energy density within a column 574
A.1.4.4 Budget for the globally averaged kinetic energy density . . . 575
A.1.5 External mode kinetic energy budget . . . 575
A.1.5.1 Partitioning the budget into physical processes . . . 575
A.1.5.2 Basic interpretation of the terms in the budget . . . 577

A.1.5.3 Budget for the global volume averaged external mode energy density . . . 577
A.1.6 Internal mode global kinetic energy density budget . . . 578
A.1.6.1 Comparing the external mode and full energy density budgets 578
A.1.6.2 Budget for the internal mode's global averaged kinetic energy density . . . 579
A.1.7 Concerning the diagnostic option *energy_analysis* . . . 579
A.1.7.1 Splitting of the energy density . . . 579
A.1.7.2 A useful result . . . 580
A.1.7.3 Algorithm for the internal mode . . . 581
A.1.7.4 Algorithm for the external mode . . . 581
A.1.7.5 Special case of a flat bottom and rigid lid . . . 582
A.2 Energetics on the discrete grid . . . 582
A.2.1 Conservative advection: part I . . . 582
A.2.2 Conservative advection: part II . . . 583
A.2.3 Zero work by the Coriolis force . . . 585
A.2.4 Work done by pressure terms . . . 585
A.2.5 Work done by Buoyancy . . . 587

B Tracer mixing kinematics 589
B.1 Basic properties . . . 589
B.1.1 Kinematics of an anti-symmetric tensor . . . 590
B.1.1.1 Effective advection velocity . . . 590
B.1.1.2 Skew or anti-symmetric flux . . . 591
B.1.2 Tracer moments . . . 591
B.2 Horizontal-vertical diffusion . . . 592
B.3 Isopycnal diffusion . . . 592
B.3.0.1 Basis vectors . . . 593
B.3.0.2 Orthonormal isopycnal frame . . . 594
B.3.0.3 z-level frame . . . 594
B.3.0.4 Small angle approximation . . . 595
B.3.0.5 Errors with z-level mixing . . . 596
B.4 Symmetric and anti-symmetric tensors . . . 597
B.5 Summary . . . 598

C Isoneutral diffusion discretization 599
C.0.1 Summary and Caveats . . . 599
C.0.2 Functional formalism . . . 600
C.0.3 Neutral directions . . . 601
C.0.4 Full isoneutral diffusion tensor . . . 601
C.0.5 Active tracers versus passive tracers . . . 601
C.1 Functional for isoneutral diffusion . . . 602
C.2 Discretization of the diffusion operator . . . 603
C.2.1 A one-dimensional warm-up . . . 604
C.2.2 Grid partitioning . . . 605
C.2.3 Partial cells . . . 606
C.2.4 Quarter cell volumes . . . 609
C.2.4.1 x-y plane . . . 609

C.2.4.2 x-z plane . . . 610
C.2.4.3 y-z plane . . . 611
C.2.5 Tracer gradients within the 36 quarter cells . . . 612
C.2.5.1 x-y plane . . . 612
C.2.5.2 x-z plane . . . 613
C.2.5.3 y-z plane . . . 614
C.2.6 Schematic form of the discretized functional . . . 615
C.2.7 Reference points for computing the density gradients . . . 616
C.2.8 Piecing together the discretized functional . . . 617
C.2.8.1 Functional in the x-y plane . . . 617
C.2.8.2 Functional in the x-z plane . . . 618
C.2.8.3 Functional in the y-z plane . . . 618
C.2.9 Slope constraint . . . 619
C.2.10 Derivative of the functional . . . 619
C.2.10.1 x-y plane . . . 620
C.2.10.2 x-z plane . . . 622
C.2.10.3 y-z plane . . . 625
C.2.10.4 Recombination of terms in the x-y plane . . . 628
C.2.10.5 Recombination of terms in the x-z plane . . . 634
C.2.10.6 Recombination of terms in the y-z plane . . . 638
C.3 Isoneutral diffusive flux . . . 638
C.3.1 Zonal component to the isoneutral diffusive flux . . . 639
C.3.2 Meridional component to the isoneutral diffusive flux . . . 641
C.3.3 Vertical component to the isoneutral diffusive flux . . . 642
C.3.4 Stencils for small angle flux components . . . 644
C.4 General comments . . . 645
C.4.1 Isoneutral diffusion operator . . . 645
C.4.2 Vertical diffusion equation . . . 646
C.4.3 Dianeutral piece . . . 646
C.4.3.1 Full tensor . . . 646
C.4.3.2 Small tensor . . . 646
C.4.4 Highlighting the different average operations . . . 646

D Horizontal friction discretization 655
D.1 Motivation and summary . . . 655
D.2 Review of the continuum results . . . 656
D.3 Discretization of the functional . . . 658
D.3.1 General form of the discrete functional . . . 658
D.3.2 Subcell volumes . . . 660
D.3.3 Derivative operators . . . 660
D.3.4 Tension for the subcells . . . 660
D.3.5 Strain for the subcells . . . 660
D.4 Discrete friction . . . 661
D.4.1 Functional derivative of the physical velocity components . . . 661
D.4.2 Functional derivative of D_T . . . 662
D.4.3 Functional derivative of D_S . . . 662
D.4.4 Rearrangement of terms for the zonal friction . . . 663
D.4.5 Rearrangement of terms for the meridional friction . . . 666

D.5 Discretization of tension and strain for the quadrants 667
D.6 Comments 668

E A note about computational modes 669

F References 671

List of Figures

3.1 Directory structure for MOM 31

4.1 Ocean free surface 42

11.1 Various ways to slice a 3-D volume of data on disk and the array shapes needed in memory. 139
11.2 Anatomy of a Memory Window for 2nd order numerics. 140
11.3 Anatomy of a Memory Window for 3rd or 4th order numerics 141
11.4 Using a 2nd order memory window to integrate 3-D prognostic equations for one timestep 142
11.5 Using a 4th order memory window to integrate 3-D prognostic equations for one timestep 143
11.6 Copying data within a memory window 144
11.7 Copying data within a memory window for pressure gradient averaging 145
11.8 Various ways of configuring MOM 146

12.1 1d vs 2d Domain Decomposition 153
12.2 Basic Dataflow for Multi-tasking with 2nd order numerics 154
12.3 Basic Dataflow for Multi-tasking with 4th order numerics 155

16.1 Specifying a region of T cells. 208
16.2 Grid cells in λ and ϕ 209
16.3 Grid cells in λ and z 210
16.4 Grid cells in ϕ and z 211
16.5 Comparing grid construction methods 212

17.1 Euler rotation angles for grid rotations 216

18.1 A sample $kmt_{i,jrow}$ field 225

19.1 Coupling ATMOS and OCEAN models 230
19.2 Flowchart for program *driver* 239
19.3 Flowchart for subroutine *gasbc* 240
19.4 Flowchart for subroutine *gosbc* 241

20.1 Topography at the open boundary 249
20.2 Example of specified throughflow 250

21.1 Dataflow for various types of timesteps 263

22.1 Flowchart for subroutine *mom* 292
22.2 Grid cells within the memory window indicating where quantities are defined 293
22.3 Advective velocities 294
22.4 Vertical velocity at ocean bottom 295

26.1 Comparing realistic and discretized bottom topography 321
26.2 Comparing realistic and discretized bottom topography 322
26.3 Two vertical columns of T-cells with one partial bottom cell 323
26.4 Partial Relation between partial bottom T-cells and U-cells 324

28.1 Analytic zonal wind stress and its curl. 342
28.2 Linear surface temperature profile. 343

29.1 Relationship between baroclinic and barotropic timesteps. 368

30.1 Schematic of surface model cells 376
30.2 Schematic of the split-explicit time stepping scheme 377
30.3 Sketch of the cell arrangement for a northern river 398
30.4 Sketch of the cell arrangement for a river flowing eastward 399

35.1 Baroclinic zone used for the Visbeck *et al.* scheme 496

40.1 Fresh water input and net northward transport 539

C.1 One dimensional warm-up 605
C.2 Full cell partitioning of tracer cells in x-y plane 648
C.3 Full cell partitioning of tracer cells in x-z plane 649
C.4 Partial cell partitioning of tracer cells in x-z plane 650
C.5 Partial cell partitioning of tracer cells in y-z plane 651
C.6 Grid stencil for zonal diffusion flux 652
C.7 Grid stencil for vertical diffusion flux 653

D.1 Stencil for the discrete frictional functional 659
D.2 Notation for the quadrants surrounding a velocity point 667

Part I

Introduction to MOM and its use

Chapter 1

Introduction

1.1 What is MOM?

MOM is an acronym for Modular Ocean Model. The model was designed and developed by researchers at the Geophysical Fluid Dynamics Laboratory (GFDL/NOAA Department of Commerce) as a numerical ocean modelling tool for use in studying ocean circulation over a wide range of space and time scales. Institutionally, MOM is supported by GFDL. The focus of development work is to maximize scientific productivity within the computational environment at GFDL. However, the model is sufficiently general to be of use elsewhere. Therefore, MOM is being made freely available to the general oceanographic and climate research community as public domain software. Unless otherwise noted, MOM refers to MOM 3 version 0 (MOM 3.0) which represents the state of the art in ocean modelling at GFDL near the end of 1999.

This manual is included as part of MOM. Its purpose is to provide documentation as well as guidance to aid in the educated use of MOM by exposing details of the salient theoretical and numerical ideas upon which MOM is based. Without it, details inappropriate for published papers would certainly be lost or at best remain obscure to all but a very few. Although the bulk of this document has been written by two main authors, many researchers from around the world have contributed as well and their work is acknowleged in their respective sections. If questions arise, authors may be contacted for help. However, do not expect them to solve your coding problems.

1.2 Accessing the manual, code, and database

The manual and FORTRAN code in their entirety may be obtained by anonymous ftp from GFDL using:

ftp ftp.gfdl.gov	use *ftp* as your login name and your *e-mail* address as password
cd pub/GFDL_MOM3	Change to the pub/GFDL_MOM2 directory
get manual3.0.ps.Z	Copy the manual to your directory
get mom3.0.tar.Z	Copy the model to your directory
quit	disconnect from the ftp
uncompress manual3.0.ps.Z	Expand to manual3.0.ps
uncompress mom3.0.tar.Z	Expand to mom3.0.tar
tar xvf mom3.0.tar	Extract MOM_3 from the tar file

A database is also included as part of MOM and is the same as the database that was included with MOM_2. While in the pub/GFDL_MOM3 directory, do a *cd* DATABASE to get into the DATABASE directory. This DATABASE directory contains approximately 160MB of IEEE 32bit data files which are described in Chapter 13. Files can be retrieved with the *get* command as in the anonymous ftp example given above. It is best to copy the files one at a time since file sizes range from about 4MB to about 8MB and there are 30 of them. So be prepared to go to lunch. The dataset is not available via exabyte tape or any other way.

1.3 Minimum computational requirements

MOM requires a Fortran 90 compiler, UNIX, and a C-preprocessor.

MOM was designed to execute most efficiently on vector processors, although it will run reasonably well on scalar processors. It was also designed with a single processor in mind. However, it was extended for use on multiple processors. On the CRAY T90 or CRAY T3E at GFDL, compiler version 3.1.0.0 (or later) and message passing toolkit version 1.2.1.0 (or later) are required. Both SHMEM and MPI message passing protocols are supported through the GFDL message passing interface (http://www.gfdl.gov/ vb). Parallel I/O is supported through the GFDL parallel I/O interface (http://www.gfdl.gov/ vb).

1.4 How this manual is organized

The table of contents serves as a detailed outline of what is available within MOM. This manual consists the following parts.

- A brief history of ocean modelling at GFDL.
- The nuts and bolts of using MOM
- The basic formulation
- The code design
- Grids, Geometry, and Topography
- Boundary conditions
- Finite difference equations.
- Physics and numerics options
- Diagnostic options
- Appendices and references

The best way to digest this manual is in piecemeal fashion by bouncing back and fourth between the table of contents and reading sections of interest.

1.5 Special acknowledgments and disclaimers

1.5.1 Acknowledgments

To a large part, MOM owes its existence to Kirk Bryan and to Jerry Mahlman (the director of GFDL) for creating an environment in which this work could take place. Continued strong support comes from Robbie Toggweiler who is the current head of the ocean group at GFDL. Their generosity is gratefully appreciated. Also appreciated is the time and efforts of countless researchers who have tested beta versions, pointed out problems, and continue to suggest improvements along with offering parameterizations. Clipart is from Corel Gallery.

1.5.2 Disclaimer

As with any research tool of this magnitude and complexity, bugs are inevitable and some have undoubtedly survived the testing phase. Researchers are encouraged to bring them to our attention.

Although the model will catch many oversights of the kind typically made by novices, it is ultimately the responsibility of the researcher to insure that the combination of options being used is relevant to the problem being studied. It is also stressed that the researcher accepts full responsibility for verifying that their particular configuration is working correctly.

Anyone may use MOM freely on a "use as is" basis. The authors of MOM assume no responsibility (zero) for any problems, incorrect usage, or bugs.

1.5.3 Software license

U.S. Department of Commerce (DOC) Software License for MOM 3

1. Scope of License. Subject to all the terms and conditions of this license, DOC grants USER the royalty-free, nonexclusive, non transferable, and worldwide rights to reproduce, modify, and distribute MOM, herein referred to as the Product.

2. Conditions and Limitations of Use Warranties. Neither the U.S. Government, nor any agency or employee thereof, makes any warranties, expressed or implied, with respect to the Product provided under this License, including but not limited to the implied warranties or merchantability and fitness for any particular purpose. Liability. In no event shall the U.S. Government, nor any agency or employee thereof, be liable for any direct, indirect, or consequential damages flowing from the use of the Product provided under this License. Non-Assignment. Neither this License nor any rights granted hereunder are transferable or assignable without the explicit prior written consent of DOC. Names and Logos. USER shall not substitute its name or logo for the name or logo of DOC, or any of its agencies, in identification of the Product. Export of technology. USER shall comply with all U.S. laws and regulations restricting the export of the Product to other countries. Governing Law. This License shall be governed by the laws of United States as interpreted and applied by the Federal courts in the District of Columbia.

3. Term of License. This License shall remain in effect as long as USER uses the Product in accordance with Paragraphs 1 and 2.

Chapter 2

A brief history of ocean model development at GFDL

2.1 Bryan-Cox-Semtner: 1965-1989

The GFDL Ocean Model started as a three dimensional primitive equation model based on the pioneering work of Kirk Bryan (1969). Early Fortran implementations of Bryan's ideas were carried out chiefly by Mike Cox in Washington, D.C. during the late 1960's on an IBM 70301/stretch and then a CDC 6600 computer. After GFDL moved to Forrestal Campus of Princeton University, Cox continued developments by constructing a global model in 1968 on a UNIVAC 1108. Bert Semtner[1] converted that model to execute on Princeton University's IBM 360/91 in 1970 and both codes were in use through 1973 with Semtner's version surviving for use on an interim IBM 360/195 in 1974. While at GFDL and UCLA, Semtner (1974) rewrote the model to take advantage of the instruction stack on the IBM 360/195 and also with future vector architectures in mind. The first vector machine arrived at GFDL in 1975. It was a four pipeline Texas Instrument ASC (acronym for Advanced Scientific Computer) and the model of Semtner (1974) was used as the starting point for further conversion efforts by Cox and Pacanowski. After the ASC, Cox abandoned the ASC version of the model in favor of Semtner's latest version and optimized it for the CDC Cyber 205 which required very long vector lengths for efficiency. Actually, the Cyber 205 experience involved two Cyber 205's and a Cyber 170 front end delivered to GFDL in stages between 1982 and 1983. It was the resulting "Cyberized" version of the model that was distributed as the Cox (1984) ocean model code. Over the lifetime of the Cyber system, Cox installed other features such as variable horizontal resolution, multiple tracers, and isopycnal mixing until his untimely death in 1989. The Cox code entailed about 5000 lines of fortran code.

2.2 The GFDL Modular Ocean Models: MOM 1 and MOM 2: 1990-1995

In anticipation of a Cray YMP with 8 processors, 32 Mwords of central memory (eventually upgraded to 64 Mwords), and 256 Mwords of Solid State Disk arriving at GFDL in 1990, the Cox code was abandoned. The reason was that many of the coding features specific to the

[1] He was stationed at GFDL in the early 1970's as a commissioned officer in the NOAA CORPS. He later did his Ph.D work with Kirk Bryan at Princeton.

CYBER 205 design were not needed to take advantage of the Cray class of supercomputers. The model was rewritten again, this time by Pacanowski, Dixon, and Rosati (1991) using ideas of modular programming to allow for more options and increased model flexibility. This development work, which became known as MOM 1 (the first Modular Ocean Model) entailed about 17000 lines of fortran code. It could not have happened without reliance on new ideas for model design, workstations, and the acceptance of UNIX[2]. With the realization of the importance of workstations for productivity within GFDL, SUN workstations were replaced by a suite of SGI 4D/25, INDIGO, and INDIGO2's totaling 115 within the early 1990's. With the aid of these faster workstations, further design work was carried out primarily by Pacanowski and Rosati but with numerous contributions from others both inside and outside of GFDL. This led to the incarnation known as MOM 2 Version 1 (1995). MOM 2 entailed about 60,000 lines of fortran 77 code.

2.3 MOM 3: 1996-1999

Early in 1996, a Cray C90 was installed at GFDL with 16 processors, 256 Mwords of central memory, 1 Gword of solid state disk, and 370 Gbytes of rotating disk. Later that year, the system was replaced by a Cray T90 having 20 processors, 512 Mword central memory and a 2 Gword solid state disk. The Cray T90 was later upgraded to 26 processors in 1997 and a Cray T3E with 40 processors and 640 Mwords of memory also arrived. In 1998, a Cray T90 with 4 processors was added. In 1996, a beta version of MOM 2 (version 2) was made available as a stepping stone to MOM 3. In anticipation that parallelization will be needed to keep overall system efficiency high in the future, attention has been and continues to be placed in this direction. Throughout these developments, the intent has been to construct a flexible research tool useful for ocean and coupled air-sea modeling applications over a wide range of space and time scales. As outlined below, progress with parallelization, and the implementation of fundamentally new physics and numerics options, motivated the release of MOM 3 which is described within this manual.

2.4 Documentation

Early use of the GFDL ocean model in the 1960's and early 1970's was limited to researchers within GFDL. In the early 1980's, as the number of researchers increased, Kirk Bryan convinced Mike Cox of the need for documentation and Cox proceeded to document the numerical discretizations used at that time. As a result, the Cox manual was made available in 1984 along with his code. By entraining outside researchers, the GFDL model opened itself up to a larger variety of uses. In turn, the code and manual were exposed to intense scrutiny, much of which has led to the development of numerous improvements.

With the release of MOM 1 in 1990, there were many requests for updated documentation. To satisfy this need, the core of this present document was written for the release of MOM 2 in 1995 (Pacanowski 1995). Subsequent additions and revisions by Pacanowski and Griffies have resulted in the present document associated with the release of MOM 3.

References to MOM in the literature can be given as

[2]In the latter half of the 1980's, SUN 3/50 workstations were introduced which ushered in a new era of model development. Before this, code development was done without the aid of editors or utilities like UNIX *grep*.

- Pacanowski, R. C., and S. M. Griffies, 2000: **MOM 3.0 Manual**, NOAA/Geophysical Fluid Dynamics Laboratory, Princeton, USA 08542. 680 pages.

2.4.1 Main differences between MOM 2 and MOM 3

This section highlights the main differences between MOM 2 and MOM 3. Section 2.5 provides a discussion of the differences between MOM 1 and MOM 2. In general, model variables and indices are the same as in MOM 2 except that two dimensional arrays have been removed from common blocks and placed into modules. The three dimensional arrays associated with the baroclinic and tracer portions of the model remain as common blocks. The intent is to eventually have all arrays placed within modules but this has not yet been implemented because doing so currently results in a significant speed penalty (about 30%) on the CRAY T90. At present common blocks can be replaced by modules under control of an *ifdef* for testing purposes. When the Fortran 90 environment matures, the intent is to remove common blocks and install modules. For upgrading local modifications from MOM 2 to MOM 3, refer to Section 3.12.

2.4.2 Parallelization and Fortran 90

In MOM 2 and previous versions, Fortran 77 was required. The minimum requirement for MOM 3 is Fortran 90. Much has been learned from experimenting with parallelization in MOM 2. From this experience, a distributed memory paradigm has been adopted with communication calls to exchange data between processors. The domain decomposition is limited to one dimensional (latitude only) which means that there must be more latitude rows than processors. Both SHMEM and MPI message passing protocols are supported through the GFDL message passing interface (http://www.gfdl.gov/ vb). Parallel I/O is supported through the GFDL parallel I/O interface (http://www.gfdl.gov/ vb).

2.4.3 Model physics and numerics

The main advances in MOM 3 relative to MOM 2 are in the model's physics, numerics, and parallelization. The following is a brief outline of the additions to physics and numerics.

1. Implementation of KPP vertical mixing scheme of Large, McWilliams, and Doney (1994) (Section 33.2.3).

2. Implementation of partial bottom cell topography of Pacanowski and Gnanadesikan (1998) (Chapter 26).

3. Implementation of bottom boundary layer of Gnanadesikan, Winton, and Hallberg (1998) (Chapter 37. Work on this option is ongoing.).

4. Implementation of the Gent-McWilliams skew-flux of Griffies (1998) (Section 35.1.6).

5. Generalization of the isoneutral diffusion scheme of Griffies *et al.* (1998) to allow for partial bottom cells (derivation in Appendix C).

6. Streamlining of the isoneutral mixing schemes which results in a reduction in model run time relative to the MOM 2 implementation.

7. Implementation of the Held and Larichev (1996) and Visbeck, Marshall, Haine, and Spall (1997) closures for the Redi and GM tracer diffusivities (Section 35.2).

8. Implementation of the Roberts and Marshall (1998) biharmonic mixing scheme (Section 35.1.8).

9. Implementation of an explicit free surface (Chapter 7 and Section 29.5).

10. Implementation of fresh water fluxes into the explicit free surface, rather than virtual salt fluxes. Formulation is given in Chapter 7.

11. Implementation of a specified spatially variable horizontal viscosity which includes the proper kinematic terms proportional to the spatial derivatives of the viscosity (Chapter 9 and Section 34.6).

12. The meridional streamfunction diagnostic has been expanded so that the streamfunction can be computed using potential density as a vertical coordinate (Section 40.9).

13. A diagnostic has been implemented which will map all the terms affecting the evolution of locally referenced potential density (Section 40.7).

14. The old time manager has been replaced by a Fortran 90 time manager which defines time structures and overloads the standard numerical operations of plus, minus, times, and divide to work with structures. All manipulations involving time are now much simpler than before.

15. An exchange module will be added to conserve quantities being passed between different latitude-longitude grids. The intent is for coupled air-sea applications. (planned but currently not implemented)

16. Common blocks are being replaced by Fortran 90 modules. (For the barotropic portion only. There is a 30% slow down in speed when common blocks are removed from the baroclinic and tracer portions of the model. As Fortran 90 matures, the remaining ones will be replaced.)

17. In addition to the Euler Backward and the forward mixing time steps every $nmix$ time steps (usually $nmix = 17$), an option has been added for a Robert filter applied every time step (Section 21.4.4).

18. The model topography can now be changed by editing the file *kmt.dta*with a text editor.

19. There is an option for an isotropic grid (one where Δ_y compensates for the convergence of meridians to keep the grid cells square).

20. The test case resolution has been changed from a 4° x 3° grid to a 3° x 2.765° grid to facilitate parallel processing tests with up to 64 processors.

21. The mean radius of the earth has been changed from 6370 km to 6371 km.

22. A parameterization for mixing tracers between unconnected regions of ocean has been added as a way to handle the tracer exchange between the Mediterranean and Atlantic as well as other regions where resolution is insufficient to allow realistic exchanges (Section 36.2).

23. The older relaxation methods for solving elliptic equations have been removed in favor of the method of conjugate gradients.

24. As an ongoing research topic, ways to speed up communication between processors are being explored. When improvements are implemented, changes are confined to a small communication package.

2.5 Main differences between MOM 1 and MOM 2

This section highlights the main differences between MOM 1 and MOM 2. As mentioned above, there are more fundamental architectural similarities between MOM 2 and MOM 3 than between MOM 1 and MOM 2.

2.5.1 Architecture

There are major architectural differences between MOM 1 and MOM 2. As a result, there is no simple utility which will provide a meaningful upgrade path from MOM 1 to MOM 2. One of the first differences to notice is a change in naming variables. To remove lack of uniformity and to provide guidance in choosing variable names for future parameterizations, a naming convention has been adopted as described in Section 14.1. Not only variable names but details of subscripts and numerics within this documentation consistently match what is found in the model code. Therefore, understanding this documentation will allow the researcher to take a big step towards gaining a working knowledge of MOM 2.

Apart from renaming of variables, the next thing to notice is that a latitude "j" index has been added to expose all indices of arrays in MOM 2. Although the organization of the code bears similarity to MOM 1, this added "j" index results in fewer variable names being required and triply nested "do loops" replacing the doubly nested loop structure in MOM 1. It also allows the slab architecture of MOM 1 to be extended to a more general memory window structure which permits solving equations on one or more latitude rows at a time. This has implications for parallelization and simplifies incorporating parameterizations (such as fourth order accurate schemes, flux corrected transport schemes, etc.) which require referencing data from more than one grid point away. For such parameterizations, the memory window is simply opened up to contain four latitude rows as opposed to the usual three. In the limit when enough central memory is available, the memory window can be opened all the way to contain all latitude rows, in which case all data is entirely within central memory, and therefore no movement of data between central memory and disk is needed. Also, in contrast to a partially opened memory window, there are no redundant computations necessary. The main point is that all arrays and equations look the same regardless of the size of the memory window and whether one, a few, or all latitude rows are being solved at once. The details are given in Chapter 10.

The memory window also allows flexibility in parallelization as described in Chapter 12. When executing on multiple processors, MOM 2 can make use of fine grained parallelism ("autotasking") or the coarse grained parallelism ("microtasking"). Each method has its advantages and disadvantages. Fine grained parallelism makes efficient use of available memory and offers a robust coding environment which is easy to use thereby keeping the researchers efforts focused on science as opposed to debugging. It suffers from relatively low parallel ef-

ficiency[3] which limits its use to multi-tasking with a small number of processors. However, the highest parallel efficiency may not be important when multi-tasking on systems with tens of processors and when the number of jobs in the system exceeds the number of processors. Higher parallel efficiency, which is necessary when executing in a dedicated system, can be achieved through coarse grained parallelism. The down side is that this approach uses significantly more memory than fine grained parallelism and is more prone to introducing errors. The researcher who is intent on developing new parameterizations with coarse grained parallelism in mind may find the focus shifts from science to debugging.

2.5.2 Physics and analysis tools

Other features which are new to MOM 2 relative to MOM 1 include the following.

1. Various modules can be exercised alone or as part of a fully configured model, as discussed in Chapter 15.

2. An integrated DATABASE is described in Chapter 13 along with run_scripts in Section 3.2 which will automatically prepare this data for any of the configurations and arbitrary resolutions of MOM 2.

3. Chapter 19 details a generalized surface boundary condition interface which handles all surface boundary conditions as if they come from a hierarchy of atmospheric models. This includes simple datasets which are fixed in time through complicated atmospheric GCM's. Mismatches in geometry and resolution between atmospheric GCM's and MOM 2 are automatically taken care of. However, since linear interpolation is used to apply atmospheric fluxes to MOM 2, the fluxes are not stricly conserved.

4. Elliptic equation solvers for the external mode have been re-worked to be more accurate and give speedier convergence as discussed in Section 22.11.

5. The vertical velocity fields have been reformulated to prevent numerical separation in the presence of sharp topographic gradients as described in Section 22.3. The grid is constructed by a module which allows for a MOM 1 type construction with grid points always in the center of tracer cells on non-uniform grids <u>or</u> a new way with grid points always in the center of velocity cells on a non-uniform grids. Both are second order accurate if the stretching function is analytic and are described in Chapter 16.

6. All diagnostics have been re-written to be more modular, old ones have been improved, many new ones added (such as reconstructing the surface pressure from the stream function, calculating particle trajectories, time averaged fields, xbt's etc.), and all are described in Chapter 39.

7. The prognostic surface pressure and implicit free surface methods of Dukowicz and Smith (1993,1994) have been implemented.

8. The isoneutral diffusion scheme of Griffies *et al.* (1998) has been implemented.

9. The eddy advection of Gent and McWilliams (1990) has been implemented, with numerics made consistent with the new isoneutral diffusion formulation.

[3]The efficiency is limited by how smart the parallelizing compiler is.

10. Options for tracer advection include the traditional centered differences, a fourth order advection scheme taken from Mahlman's stratospheric code (SKYHI), the FCT scheme of Gerdes, Köberle and Willebrand (1991), a third order advection scheme (by Holland) very similar to the Quick scheme of Leonard (1979).

11. The pressure gradient averaging technique of Brown and Campana (1978).

12. Neptune effect of Holloway (1992).

13. Rigid grid rotation; i.e., rotate poles, while keeping the identical lat/lon structure.

14. Open boundaries of Stevens (1990).

15. The discretization of vertical mixing of Pacanowski/Philander (1981) has been changed to yield more accurate and stable solutions as indicated in Section 33.2.4.

16. Some restructuring of the memory window logic to allow for a more robust implementation of parallelism and fourth order schemes.

17. There are also more options for configuring MOM as described in Part VII and many other little features and code improvements too numerous to summarize here but covered in this manual.

Some of the differences between MOM 2 version 1 and version 2 are as follows.

1. All diagnostics have been given an interface to generate NetCDF formatted output as described in Chapter 39. The NetCDF format allows easy access to results without writing intermediate analysis code. The recommended way to visualize results is to use Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL
email: ferret@pmel.noaa.gov
web: http://www.pmel.noaa.gov/ferret/home.html

2. Uni-tasking is discussed in Chapter 11 and multi-tasking is discussed in Chapter 12.

3. Since the arrival of a CRAY T90 and Unicos 9 operating system in August 1996, there is no longer a CRAY Fortran 77 compiler. It has been replaced by a Fortran 90 compiler which for the most part is compatible with "cf77". The minimum requirement for MOM is now Fortran 90.

Chapter 3

Getting Started

This chapter describes what is needed to start using the code by executing the supplied test cases. These test cases are only intended as examples of how to start using MOM. Once the concepts are clear, researchers are expected to devise their own run scripts and configurations for archiving data. Since most researchers wish to start running MOM as soon as possible without knowing what they are doing, this "nuts and bolts" chapter is presented at the beginning of the manual rather than at the end. Accessing MOM are given in 1.2

3.1 How to find things in MOM

Assuming nothing about MOM is know, finding things presents a problem. The solution is to use UNIX utilities such as *grep*. For example, suppose all areas within the model having anything to do with isoneutral mixing are to be located. Searching for option *isoneutralmix* with the following command

```
grep -i isoneutralmix *.[Fh]
```

will find all such sections. The "-i" option is useful because it ignores upper/lower case distinctions. Searching for names of variables can likewise show every place where they are used. Definitions for variables can be seen by searching all ".h" files. Another very useful UNIX utility is "diff" as described in Section 3.13.

3.2 Directory Structure

First, refer to Figure 3.1 for a schematic view of how the directory structure of MOM_3 is organized at GFDL. The structure is divided between two file systems: the CRAY file system contains the data part and the workstation file system contains all code and *run_scripts*. This structure is arbitrary but not without reason; especially the flat file structure used for the code which is described below. The recommendation is that this structure be retained as much as possible. Doing so will make things easy.

On the CRAY file system, there is an ARCHIVE/MOM_3/DATABASE directory. The DATABASE contains Hellerman and Rosenstein (1983) monthly climatological wind stress on a 2° grid, Oort (1983) Monthly Surface air temperature on a 5° grid, Levitus (1982) monthly temperature

and salinity on a 1° grid, and Scripps topography on a 1° grid[1]. There is also an ARCHIVE/MOM_3/EXP directory where interpolated data from the DATABASE and results[2] from various experiments are stored, each under their own sub-directory[3]. The only sub-directory included is ..EXP/TEST_CASE which (after executing run scripts described below) will contain an interpolated version[4] of the DATABASE appropriate for the domain and resolution of the test case which is described below.

On the workstation file system, there is also a MOM_3 directory containing code, *run_scripts*, and four sub-directories: MOM_3/PREP_DATA for preparing surface boundary conditions and initial conditions, MOM_3/SBC for handling various types of surface boundary conditions, MOM_3/NETCDF[5] containing routines to interface to the netcdf library, and MOM_3/EXP which in general contains a sub-directory for each experiment.

Note that as far as the actual fortran code, the file structure is basically flat with all code relating to the model proper being lumped into one place (in the MOM_3 directory). An alternative is to impose some structure by dividing the code up and placing related routines into sub-directories under MOM_3. For instance, vertical diffusion routines could be placed under sub-directory MOM_3/VERT_DIFFUSION, etc. With such a segmented file structure, finding and editing source code becomes a chore. However, with the aid of UNIX, any file structure can be easily sifted out of the flat file structure. For instance, suppose, it is necessary to look at all routines having anything to do with biharmonic mixing. The following UNIX call

```
grep -l biharmonic *
```

will list the subset of filenames. The files are all in one place and immediately available for editing. For the future, this method can be made even more effective by embedding keywords in the comments of routines. For instance, placing a comment with the phrase "SGS parameterization" in each routine that is a sub-grid scale parameterization will allow all such routines to be easily listed.

Details of the sub-directories under MOM_3 are given below:

- PREP_DATA contains subroutines and CRAY T90 *run_scripts* for extracting data[6] from the DATABASE and interpolating it to arbitrary resolution for use as surface boundary conditions and initial conditions within MOM. Before this can be done, the domain and resolution must first be specified in module *grids* as discussed in Chapter 16. The run scripts are:

  1. *run_sbc* reads unformatted climatological monthly (also annual means) Hellerman stress (1983) and Oort (1983) surface air temperature and interpolates to the grid

[1] In principle, this DATABASE could be expanded to include other datasets but this has not been done as of this writing

[2] Model output may be composed of a printout file, diagnostic files and restart data.

[3] For example, EXP/ATLANTIC, EXP/PACIFIC, EXP/GLOBE.

[4] Note that these interpolated datasets are only needed for test cases #1 and #2. Test cases #0 and #3 rely on internally generated data.

[5] This directory has been superseded by the parallel I/O interface described in http://www.gfdl.gov/ vb.

[6] All DATABASE data consists of a header record preceding each data record. Included in each header is a time stamp. It contains the time corresponding to the instantaneous time at the end of the averaging period. It also contains a period which refers to the length of the time average. As an example, a time stamp of: m/d/y= 2/1/1900,h:m:s= 0: 0: 0. points to the beginning of the 1st day of Feb on year 1900. A period of 31 days for this record means that the data is average over the preceding 31 days; i.e, it is an average for January.

defined by module *grids*. Look for the USER INPUT section to choose the type of interpolation appropriate for the grid resolution. The run script uses file *sbc.F* which is included in the directory. If option *netcdf* is enabled in *run_sbc* then a NetCDF version of the interpolated dataset *sbc.dta.nc* will also be produced. Land values are not flagged. Refer to Section 3.10 for how to mask out land values in plots.

2. *run_ic* reads unformatted monthly Levitus (1982) temperature[7] and salinity data and generates monthly (and annual mean) climatological initial conditions along with surface temperature and salinity for the grid defined by module *grids*. Look for the USER INPUT section to choose the type of interpolation appropriate for the grid resolution. This script uses file *ic.F* which is included in the directory. If option *netcdf* is enabled in *run_ic* then a NetCDF version of the interpolated dataset *ic.dta.nc* will also be produced. Land values are not flagged. Refer to Section 3.10 for how to mask out land values in plots.
3. *run_sponge* reads output files produced by *run_ic* to construct sponge rows for damping model predicted temperature and salinity back to these data near northern and southern artificial walls. This is only appropriate for use in limited domain models and is the poor mans open boundary condition. This script uses file *sponge.F* which is included. The width of the sponge layers and the variation of Newtonian damping time scale within the sponge layer may be set within file *sponge.F*.
4. *run_read_levitus* is a simple workstation script showing how to read the Levitus (1982) data (with Levitus land/sea masks) on a workstation. It assumes the Levitus (1982) data has been copied to the workstation's local disk. If option *netcdf* is enabled in *run_read_levitus* then a NetCDF dataset *levitus.dta.nc* will be produced. Land values are flagged.
5. *run_obc* is a run script which uses file *obc.F* for constructing data needed for open boundary conditions. This was done by Arne Biastoch (abiastoch@ifm.uni-kiel.de) but has not been converted to the CRAY T90 at GFDL at this point.
6. *run_obcpsi* is a run script which uses file *obcpsi.F* for constructing data needed for open boundary conditions. This was done by Arne Biastoch (abiastoch@ifm.uni-kiel.de) but has not been converted to the CRAY T90 at GFDL at this point.

- SBC contains three sub-directories for supplying various types of surface boundary conditions to MOM. Each is located in a separate sub-directory:

  1. TIME_MEAN contains subroutines which supply the time mean Hellerman and Rosenstein climatological winds (1983) along with the time mean Levitus (1982) SST and sea surface salinity climatologies which are used by the test case to compute effective heat and salt fluxes given a damping time scale and thickness which can be input from a MOM namelist. Refer to Section 14.4 for information on namelist variables. Note that the time scale can be different for restoring temperature and salinity. These time means are assumed to have been created using scripts from PREP_DATA so they are appropriately defined as functions of latitude and longitude on the domain and resolution specified by module *grids*. The option used to configure this type of surface boundary condition for MOM is *time_mean_sbc_data* which is described further in Chapter 19.

[7] These are potential temperatures.

2. MONTHLY contains subroutines which supply monthly mean Hellerman and Rosenstein climatological winds along with monthly mean Levitus (1982) climatological SST and sea surface salinity which are used by the test case to compute effective monthly mean heat and salt fluxes given a damping time scale which can be input from a MOM namelist. Refer to Section 14.4 for information on namelist variables. Note that the time scale can be different for restoring temperature and salinity. All are assumed to have been created by scripts from PREP_DATA so they are monthly averages appropriately defined as functions of latitude and longitude on the domain and resolution specified by module *grids*. Each dataset is defined by an averaging period and time stamp which marks the end of the averaging period. As the model integrates, the datasets are used to interpolate to the time corresponding to each model time step. It should be noted that there is enough generality to accommodate datasets with other periods (daily, hourly, etc) and treat them as climatologies (periodic) or real data (non periodic). Also datasets with differing periods may be mixed (example: climatological monthly SST may be used with hourly winds from other datasets). The option used to configure this type of surface boundary condition for MOM is *time_varying_sbc_data* which is described further in Chapter 19. There are four methods for interpolating these datasets to the time step level required by MOM as described in Section 19.2 .
3. ATMOS contains subroutines that prototype what must be done to couple MOM to an atmosphere model for the general case of two way coupling when resolution and land/sea areas do not match. The atmosphere model is unrealistic. It is intended only to show that essentially two things must be done: a boundary condition grid must be defined to match the atmospheric grid (which is assumed to be different from the MOM grid resolution) and boundary conditions such as winds and heat flux must be accumulated in arrays as indicated. The option used to configure this type of surface boundary condition for MOM is *coupled* which is explained further in Section 19.1.

- NETCDF contains[8] fortran routines written by John Sheldon at GFDL for interfacing to lower level *netcdf* routines. These lower level routines are resolved by linking to the appropriate NetCDF libraries which will be site specific. The proper linking to these libraries at GFDL is given in script *run_mom*. For other sites, the appropriate links will have to be made by the researcher. The NetCDF section of any diagnostic can be used as a template to add NetCDF capability to new diagnostics.

- EXP contains one sub-directory for each experimental design but only EXP/TEST_CASE is indicated. If there were others, they would have the same structure. EXP may also contains printout files from the four test cases described later. They were produced on the CRAY T90 at GFDL and are named *printout.0.gfld*, *printout.1.gfld*, *printout.2.gfld*, and *printout.3.gfld*. These files can be used for comparison with results generated elsewhere and are described further in Section 3.4. Under the EXP/TEST_CASE are two sub-directories:

  1. MOM_UPDATES contains only code and run_scripts from the MOM_3 directory which need to be altered to define an experiment (e.g. the test case on another platform). Actually, no fortran code is included here because the basic MOM_3 files are already configured for the test case at GFDL. Typically though, the following

[8] This directory has been superseded by the parallel I/O interface described in http://www.gfdl.gov/ vb.

would be a minimum set of useful ones: module *grids* and *run_grids* which are used to design the grid, *size.h* which is used to implement the grid size, and module *topog* and *run_topog* which are used to design the topography and geometry. Also, any other subroutine requiring changes must be placed in this directory because Cray script *run_mom* looks to this MOM_UPDATES directory for all updated code.

2. PREP_UPDATES contains only code and CRAY T90 run_scripts from the PREP_DATA directory which would have be altered to define the test case. Actually, none are here since the ones in PREP_DATA are already setup to do the test case. Typically though, only run_scripts need be copied into this directory to alter pathnames (near the beginning of the scripts) which point to where interpolated initial conditions and surface boundary conditions are to be written. The scripts are then executed from this directory on the CRAY T90 to build the interpolated DATABASE appropriate for the resolution specified by module *grids*.

3.3 The MOM Test Cases

MOM is executed by a CRAY T90 script *run_mom* which is in directory MOM_3 on the workstation side of the file structure. The script executes a test case global domain with a horizontal resolution of 3° in longitude by about 2.8° in latitude with 15 vertical levels. This yields 122 points in longitude (120 + 2 for cyclic conditions) and 66 latitude rows (64 + 2 for boundary rows which is a useful size for parallel processing tests with up to 64 processors). For simplicity and portability, idealized internally generated geometry (not very accurate) and topography (absolutetly bogus) are used. More realistic data can be easily included by enabling the option for Scripps topography in the run script. Many diagnostics are enabled (to demonstrate that they work) and output is in 32 bit IEEE format. As an alternative, an option for NetCDF formatted output can be enabled within the run script.

Only a very few options are enabled to keep physics simple for the test cases. Basically, an option is enabled for constant vertical mixing. In the horizontal, a variable horizontal mixing parameterization is enabled which weights the constant horizontal viscosity coefficient by the cosine of latitude to compensate for the convergence of meridians. This aids in resolving the Munk boundary layer at each latitude yet keeps the Killworth time step restriction from limiting the time step at high latitudes. When realistic topography is used, a light smoothing of topography is also needed and enabled northward of 85N to reduce topographic slopes so the Killworth condition remains satisfied. Latitudes northward of 75N are filtered with a fourier filter to compensate for time step restrictions due to convergence of meridians.

The barotropic equation is solved by the method of rigid lid stream function although options exist for an implicit and explicit free surface as well. The time steps are asynchronous with 1 day for density and 1 hour for internal and external modes.

Test cases #0, #1, #2, and #3 use various types of surface boundary conditions with the above configuration. They are selected by setting the CASE variable within script *run_mom* as follows:

- CASE=0 uses idealized surface boundary conditions which are a function of latitude only and independent of time: zonally averaged annual mean Hellerman and Rosenstein (1983) wind stress with surface temperature and salinity damped back to initial conditions on a time scale of 50 days using a thickness of about 25 meters. Initial conditions are no motion and an idealized temperature (function of latitude and depth) and

salinity (constant) structure[9]. All required data is generated internally and therefore the DATABASE is not needed. This is similar to the test case for MOM 1. The results are in file *EXP/TEST_CASE/printout.0.gfdl*.

- CASE=1 is similar to CASE=0 except uses time mean surface boundary conditions from SBC/TIME_MEAN which are assumed to have been prepared using scripts *run_sbc* and *run_ic* in PREP_DATA. These surface boundary conditions are a function of longitude and latitude but independent of time. The results are in file *EXP/TEST_CASE/printout.1.gfdl*.

- CASE=2 is similar to CASE=0 except uses time varying surface boundary conditions from SBC/MONTHLY as described in Section 3.2 which are assumed to have been prepared using scripts *run_sbc* and *run_ic* in PREP_DATA. The surface boundary conditions are linearly interpolated to each time step as the integration proceeds. The results are in file *EXP/TEST_CASE/printout.2.gfdl*.

- CASE=3 is similar to CASE=0 except uses surface boundary conditions supplied by an idealized atmospheric model as described in Section 3.2. This illustrates coupling MOM to an atmospheric GCM. The results are in file *EXP/TEST_CASE/printout.3.gfdl*.

3.3.1 The run_mom script

As mentioned previously, script *run_mom* is a UNIX C shell script which executes the MOM four test cases (#0, #1, #2, and #3) on the CRAY T90 at GFDL. Questions regarding the extension of this script or developing scripts for other platform architectures cannot be answered by GFDL. All extensions or alterations are left to the researcher. The following is a description of how script *run_mom* works: Near the beginning of script *run_mom*, pathnames point to where all required directories are located at GFDL. They will have to be changed at each installation. Control for which test case executes is given by C shell variable CASE. CASE=0 is for test case 0 and so forth.

When *run_mom* executes, it copies all Fortran code from directory MOM_3 into a working directory followed by all code from either MOM_3/SBC/TIME_MEAN (if CASE = 1), MOM_3/SBC/MONTHLY (if CASE = 2), or MOM_3/SBC/ATMOS (if CASE = 3). If any NetCDF option is on, all files from MOM_3/NETCDF[10] are also copied. Lastly, it copies all Fortran code from the EXP/TEST_CASE/MOM_UPDATES directory thereby installing all changes necessary (if any) to build the particular model.

Various ways of configuring MOM are controlled by options in Part VII. Diagnostics options are enabled as described in Chapter 39 . Options are set within the script using *cpp* preprocessor commands of the form *-Doption1*, *-Doption2* and so forth. These options eliminate or include various portions of code to construct a model having the desired components. They are also used to enable diagnostics and whether output is in NetCDF format or not. Note also, that the computer platform is specified within *run_mom*. Currently, the list includes -Dcray_ymp, -Dcray_c90, -Dcray_t90, and -Dsgi. Based on which setting is chosen, appropriate platform options are added for routines in MOM_3/NETCDF[11].

There is no makefile supplied for compiling MOM. If compile time is an issue, then one can be constructed by the researcher. In the compiling section of the script, there is provision for

[9] If Levitus (1982) SST and sea surface salinity are to be used as initial conditions, option *levitus_ic* must be enabled as discussed in Section 28.1.

[10] This directory has been superseded by the parallel I/O interface described in http://www.gfdl.gov/ vb.

[11] This directory has been superseded by the parallel I/O interface described in http://www.gfdl.gov/ vb.

enabling a bounds checker. This is strongly recommended as standard operating practice for verifying that subscripts do not exceed array bounds in newly developed code. Afterwards, the bounds checker should not be used since it significantly slows execution.

The compiling and linking to an executable is done in one step under Fortran 90. After compiling, separate namelist files are constructed which contain specifications to reset various defaulted quantities. Refer to Section 14.4 for a list. These namelist files are read by subroutine *setocn* and other initialization subroutines specific to individual parameterizations which have been enabled.

At this point, the executable is executed and output is redirected to file *results* which is later copied to the appropriate printout file. Except for the *printout* file which is ASCII, all diagnostic data is written to separate files as either *IEEE 32 bit* data (having a ".yyyyyy.mm.dd.dta" suffix) or NetCDF formatted data (having a ".nc" suffix) as described in Chapter 39. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2.

There are some additional files. The files *document.dta* and *restart.yyyyyy.mm.dd.dta* also have a ".dta" suffix but are not diagnostic files. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. File *document.dta* is a formatted file containing all namelist settings plus some additional information. File *restart.yyyyyy.mm.dd.dta* contains data needed to restart the integration from the point where it last ended. To write a restart, variable *restrt* must be set to *true* in the namelist. Within an actual integration, pathnames would be changed so that these files would be copied to the experiment directory (e.g. *cp* $*$*.nc* EXP/ATLANTIC) on the supercomputer archive. Refer to Section 39.3 for post processing the results.

The script *run_mom* is set to execute CASE=0. All test cases have a heavy load of diagnostics enabled for demonstration purposes. Look at the timing estimates at the end of the printout to see what they cost. Turn off the ones not needed by removing them from the option list in script *run_mom*.

3.4 Sample printout files

As mentioned previously, there are four printout files corresponding to four test cases which were executed at GFDL on a CRAY T90. Results from CASE=0 are in file *EXP/TEST_CASE/printout.0.gfdl*, from CASE=1 are in file *EXP/TEST_CASE/printout.1.gfdl*, and so forth. These cases are not intended to be scientifically meaningful. Rather they are included as examples of how to use various types of surface boundary conditions and provide a means of checking that MOM is behaving as intended. The following is a brief tour of file *printout.0.gfdl*.

File *EXP/TEST_CASE/printout.0.gfdl* begins by listing various surface boundary condition names and units followed by the version number of MOM and the values of namelist variables. If any variable was not included in the *run_mom* script namelist section, then it retains its initialized value. Otherwise, the new value is given. Immediately below, there is the file sizes in megawords needed for two dimensional fields (kflds) and the three dimensional fields (latdisk1 and latdisk2) where the latitude rows reside. If option *ramdisk* is enabled, these files are actually stored in memory but behave as if stored on disk.

Next, there is output from the *grids* module detailing everything relevant to grid cells. All this is summarized by a checksum which is essentially a sum over all grid cell information. Following this is a summary of temperature and salinity ranges used to compute density co-

efficients and a checksum of the coefficients[12]. Afterwards, output from the topography module *topog* supplies information about the $kmt_{i,jrow}$ field. Changes to the $kmt_{i,jrow}$ field are best done in a stand alone mode using script *run_topog*. Checking for violations is done iteratively so this section rambles on for awhile. If everything is as it should be then some basic statistics relating to geometry and topography are given with a map of land masses and island perimeters followed by a map of $kmt_{i,jrow}$. Along with these maps is information on how to suppress their printing since they can take up lots of space. The section finishes with a checksum for the topography and geometry.

After constructing a checksum over initial conditions, various initializations are indicated. The time manager module *tmngr* gives details on the calendar and time at initial conditions as well as information on the reference time for diagnostic switches. This is followed by information on damping surface tracers back to data which is given when option *restorst* is enabled.

Since the test case is of global extent, filtering of latitude rows is enabled (by option *fourfil* or *firfil*) and information is given as to which latitudes are filtered and by how much. Refer to Chapter 27 for a discussion of filtering.

Next comes some statistics on regions which have been arbitrarily set for diagnostic purposes along with a map detailing where the regions are. A detailed listing of filtering indices is suppressed but may be switched on as indicated in the printout. The time step multipliers are all set to unity indicating that there is no timestep acceleration with depth.

If option *time_averages* has been enabled, information on the grid over which data will be time averaged is given and if option *save_xbts* is enabled, the XBT station locations are given. Depending on enabled options, other initializations may give output here after which a general consistency check is done involving all enabled options. Two levels of messages are given: error and warning checks. After all messages are listed, if one or more error messages is present the model will stop. Warning messages will not stop the model, but the researcher should be aware of them and convinced they are harmless before continuing.

The preliminary setup finishes with a summary of enabled options after which a breakdown of the number of time steps per ocean segment and number of segments per integration is given. This is of interest only when option *coupled* is enabled.

A heavy battery of diagnostics are enabled but only for illustrative purposes. They are fully described in Chapter 39 and the I/O control variables are set to save data to unformatted files with suffix .dta as well as formatted to the printout file. At the end of the integration, all files are listed and a timing analysis is given detailing times taken by various sections of MOM. This is useful but once the information is digested, the timing should be turned off by not enabling option *timing*.

The other printout files are very similar except for case #3 which has more output because option *trace_coupled_fluxes* has been enabled to show details of surface boundary conditions as they are being interpolated from ocean to atmosphere and visa versa.

3.5 How to set up a model

As an example, assume an Atlantic model is to be set up. Once familiar with the directory structure as outlined in Section 3.2 and illustrated in Fig 3.1, the following steps[13] may be used:

[12]Density coefficients are computed by a call to module *denscoef.F* from within the model.

[13]Note that it is no longer necessary to construct density coefficients prior to executing the model.

1. Add a sub-directory under EXP with two additional sub-directories for containing updates or changes which when applied to the base code in MOM_3 will defined the new model. For example, the new experiment might be named EXP/ATLANTIC and the two additional sub-directories: EXP/ATLANTIC/MOM_UPDATES and EXP/ATLANTIC/PREP_UPDA

2. Copy file *grids.F* and script *run_grids* from MOM into EXP/ATLANTIC/MOM_UPDATES. If not executing on a CRAY T90, add option *sgi* to script *run_grids*. Specify a domain and grid resolution by entering changes in the USER INPUT of module *grids* as described in Chapter 16. Execute script *run_grids*. Examine the output and when satisfied, copy *size.h* from MOM into the EXP/ATLANTIC/MOM_UPDATES directory and make the indicated parameter changes. This domain and grid resolution will now be available to other modules and MOM.

3. Copy file *topog.F* and script *run_topog* from MOM_3 into EXP/ATLANTIC/MOM_UPDATES. The model geometry and topography will be generated by executing script *run_topog* with *options* outlined in Chapter 18. If not executing on a CRAY T90, add option *sgi* to script *run_topog*. The domain and resolution will be defined from module *grids*. Note that the $kmt_{i,jrow}$ field is printed out. Decide which if any changes are needed and enter them in the USER INPUT section of module *topog*. The $kmt_{i,jrow}$ field can also be viewed with option *topog_netcdf* which may be more convenient.

   Recommendation: When setting up a model for the first time, use options *idealized_kmt* with *flat_bottom* to generate a flat bottomed idealized geometry for the region of interest. After becoming comfortable with the way the process works, enable a more appropriate option (e.g., option *scripps_kmt*).

4. Select options from a list of available ones described in Part VII. Enable selected options by including them on the compile statement in script *run_mom* to configure the model. Diagnostics are the analysis tools used to help understand the model solution. Select appropriate diagnostics from the ones described in Chapter 39 and enable them by including them on the compile statement in script *run_mom*.

   Recommendation: When setting up a new model, use options *idealized_ic*, *simple_sbc*, and *restorst* which will simplify initial conditions and boundary conditions. Also, choose the simplest mixing schemes using options *consthmix* and *constvmix*. Only after verifying that results are as expected should consideration be given to moving on to more appropriate options. In general, progress by enabling and verifying one option at a time until the desired configuration is reached. If problems occur, simlify the configuration to help pinpoint the cause.

5. Some options require input variables to be set. All are set to default values but these values may not be appropriate for the researcher's particular configuration. Their values may be changed to more appropriate ones through namelist. For setting input variables, read through Section 14.4 for a description of the namelist variables. Also, guidance is often given within the description of the option and this information should be read.

The following steps are optional. They apply if the DATABASE is to be used or option *time_averages* is enabled.

1. If it is desirable to use data from the DATABASE, copy the scripts from PREP_DATA into the EXP/ATLANTIC/PREP_UPDATES directory, change pathnames to point appropriately, and execute *run_sbc* followed by scripts *run_ic* and *run_sponge*. These will

build a copy of the DATABASE appropriate to the domain and resolution specified in module *grids*.

2. If it is desirable to produce time averages during the integration, copy script *run_timeavgs* and file *timeavgs.F* to EXP/ATLANTIC/MOM_UPDATES. The grid used for producing time averages must be defined by modifying the USER INPUT section of file *timeavgs.F*. The entire model grid or a subset of grid points may be chosen. If after executing this script, a change is made to module *grids*, then this script must be executed again to re-establish the averaging grid. If examining the *results file* indicates that everything is as intended, copy file *timeavgs.h* from MOM_3 to EXP/ATLANTIC/MOM_UPDATES and make the indicated parameter changes.

3.6 Executing the model

Once the steps in Section 3.5 have been taken, make a copy of script *run_mom*, change pathnames to point appropriately and add the desired options from Part VII. Any options used in scripts *run_grids* and *run_topog* must also be included in the *run_mom* script. If not executing on a CRAY T90, try using option *sgi*, option *cray_ymp*, or option *cray_c90* in script *run_mom*. To keep things simple, make a short test run with options *time_step_monitor,snapshots* and *netcdf* to produce a snapshot of the data. Have a look at the data using Ferret (Section 39). After a successful test run, enable the desired diagnostic options and disable option *timing*.

3.7 Analyzing solutions

MOM is instrumented with a large number of diagnostic options for analyzing model solutions. Some diagnostics simply output prognostic quantities while others perform involved computations to generate derived quantities. Diagnostic datasets can be saved in NetCDF format and the recommended way of visualizing and manipulating this data is with Ferret. Ferret is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (email: ferret@pmel.noaa.gov, web: http://www.pmel.noaa.gov/ferret/home.html).

Production scripts

Production scripts are left to the researcher although they can be modeled after *run_mom*. At the minimum, commands must be added to allow for automatic reloading, archiving of results, and handling problems associated with long running experiments which may be site specific.

3.8 Executing on 32 bit workstations

The platform option to use with workstations is "-Dsgi" which works for the SGI workstations at GFDL. Other workstation platforms may require some changes. For instance, It has been reported that 32 bit IEEE formatted data can be read on a DEC Alpha workstation by altering the open statement and using the option "convert" as in

```
open ( ... ,convert='big_endian')
```

although since GFDL has no DEC Alpha's, this has not been verified.

If executing on any workstation with a typical 32 bit word length, it is recommended that double precision (usually a compiler option) be used otherwise numerical truncation may significantly limit accuracy. On an SGI Indigo 4000 workstation, the options to do this are "-O2 -mips2 -r8 -align64 -Olimit 2160".

For test cases #1 and #2, recall that data in the DATABASE is in 32 bit IEEE format. Routines for reading this data (e.g. *ic.F* and *sbc.F* in PREP_DATA) and preparing it for the model can be compiled with 32 bit word length "-O2" and the write statements changed to output real*8 data.

Also note that when a direct access record length is being specified while executing in double precision (as is done in MOM_3/SBC/MONTHLY/setatm.F), the number of words needs to be doubled to account for writing double precision data.

3.9 NetCDF and time averaged data

All datasets with NetCDF format end with a ".nc" suffix. If these datasets contain time averaged data, the time at which the data is defined is at the middle of the averaging period (not at the end). For example, a monthly mean windstress for September would be defined at Sept 15th. This is done to prevent confusion on plots. Otherwise, if the convention of placing the time stamp at the end of the averaging period were followed, the same plot would show a date of October 1st at zero hours.

3.10 Using Ferret

Here are some useful things to keep in mind when using Ferret to analyze diagnostic output in Netcdf format.

- Ferret recognized files as being Netcdf format by the "use" command. For example, to analyze the diagnostic file "snapshots.dat.nc", try

  ```
  use  snapshots.dat.nc
  ```

  If a message about negative values at the start of the time axis appears, it just means that the time stamp in the file is before year 1900.

  If a message appears complaining that "evenly spaced axis has edges definition: xt_i - ignored", it just means that the grid has constant resolution and edges specifications which have been added to the NetCDF file to account for non-uniform grid resolution is being ignored. Nothing to worry about.

- All data on land points are currently set to a flag value of $-1.E34$ in MOM_3. In early versions, the flag value was set to zero. Ferret can use either of these values to mask out land points for plots. For example, if a flag value of zero was used, the temperature (variable "temp") at level $k = 1$ from a snapshot file "snapshots.yyyyyy.mm.dd.dta.nc" can be plotted using

```
shade if temp ne 0 then temp
```

If the flag value is $-1.E34$, the following command will produce the same plot

```
shade temp
```

because Ferret interprets $-1.E34$ as missing data.

- When analyzing data from global models where option *cyclic* has been enabled, it is sometimes useful to move the Greenwich Meridian to the middle of a plot. Otherwise, the Atlantic ocean will be split at the Greenwich Meridian between the eastern and western sides of the plot. As an example, consider the file "snapshots.yyyyyy.mm.dd.dta.nc" from the test case. Moving the Greenwich Meridian can be done in Ferret by defining a new longitude axis "xnew" by cloning a portion of the original x-axis without the extra longitudes (i=1 and i=92), and using option "modulo". The Ferret command is

```
define axis/from_data/name=xnew/x/units=degrees x=[g=temp,i=2:91]
set axis/modulo xnew
```

The following Ferret commands will contour the stream function from file "snapshots.yyyyyy.mm.dd.dta.nc" with the Greenwich with a longitudinal region specified from $20°W$ to $20°E$.

```
set reg/x=20w:20e
fill psi[gx=xnew]
```

- Some NetCDF datasets such as Hellerman windstress which has been interpolated to model resolution by script *run_sbc* or Levitus data which has been interpolated to model resolution by script *run_ic* do not have land values flagged. The interpolated data from which NetCDF formatted data is constructed contains linearly extrapolated values over land (there is no information on which cells are land and which are ocean). The reason for this is so that if changes are made to topography the datasets don't need to be re-generated. However, the un-interpolated Levitus NetCDF dataset produced by script *run_levitus_netcdf*) has land values flagged. When comparing interpolated datasets (without flagged land values) with model generated data, the land flags can easily be generated for plotting purposes. For instance, suppose model generated temperature (variable "temp" from "snapshots.yyyyyy.mm.dd.dta.nc" which is assumed to be the first dataset [d=1]) is to be compared with interpolated Levitus temperature (variable "t_lev" from "levitus.dta.nc") for March at level $k = 2$ (the second dataset [d=2]). The following commands will show the interpolated Levitus data with land masked out

```
shade if temp[d=1] ne -1.E34 then t_lev[k=2,l=3,d=2,j=2:60]
```

If the flag value of "-1.E34" does not work then use a "0" instead. The $l = 3$ signifies the month of March and the range on "j" is to make the latitude range match the range from the file *snapshots.yyyyyy.mm.dd.dta.nc* for the test case resolution.

3.11 Upgrading from MOM 1

MOM 1 included an upgrade script for incorporating changes. Since there is a major architectural difference between MOM 1 and MOM 2, there are too many changes to offer a meaningful upgrade approach from MOM 1. The only recourse is to bite the bullet and switch to the latest release. It should be obvious that the appropriate time for switching is at the beginning of an experiment but not in the middle of one.

3.12 Upgrading to the latest version of MOM

MOM 2 version 1.0 included a script "run_upgrade_sgi" for incorporating local changes into newer versions of MOM 2. This approach has since been discarded in favor of a much better one: reliance on the directory structure in MOM and a new utility ... the graphical difference analyzer "gdiff" which exists on Silicon Graphics workstations and makes easy, painless work out of what was once a difficult, time consuming, and complicated task. Even better is "xdiff" which is an X windows based graphical difference analyzer. If similar tools are not available, an alternative method outlined below will still work reasonable well, only not as easily as using "gdiff" of "xdiff". These utilities have become indispensable for development work at GFDL.

Standard operating practice

First and foremost, as a matter of operating practice, NEVER change a routine within the parent MOM_3 directory. Copy the routine first into an UPDATES sub-directory and make changes there. A different UPDATES sub-directory should be maintained for each experiment. Variants of routines within each UPDATES sub-directory can be kept in further sub-directories; with each sub-directory inheriting routines to be changed from its parent directory and adding local modifications; for example,

- PACIFIC/MOM_UPDATES/HLFX
- PACIFIC/MOM_UPDATES/HLFX/TEST1

In this way, a hierarchy of changes can be built. If this hierarchy is carefully designed, selecting sub-directories to copy in decending order down the branches of the hierarchy will allow any combination of updates to be applied. This can convieniently be done within the run script. Look at script *run_mom* to see how all routines are first copied from the parent directory MOM_3 into a temporary working directory followed by all routines from MOM_3/EXP/TEST_CASE/MO If there were a sub-directory hanging off of MOM_UPDATES, all routines from this last sub-directory would be copied next and so forth. After routines from all sub-directories have been copied, the desired model will have been built in the working directory.

Before starting to upgrade to a newer version of MOM, move the whole existing MOM_3 directory structure to MOM_3_OLD using

```
mv  MOM_3  MOM_3_OLD
```

Then install the newer MOM_3 directory by uncompressing and extracting the tar file after retrieving it from the GFDL anonymous ftp. Now, for illustrative purposes, assume all local updates are kept in

```
MOM_3_OLD/EXP/BOX/MOM_UPDATES
```

This sub-directory will be referred to as OLD_UPDATES. It is important to realize that although many of the routines in the newer version of MOM may have changed, only routines in OLD_UPDATES will have to be examined. Make a similar sub-directory within the new MOM_3 directory using

```
mkdir  MOM_3/EXP/BOX/MOM_UPDATES
```

Let this new sub-directory be referred to as NEW_UPDATES. Note the list of files in OLD_UPDATES. Copy the corresponding files from the new MOM_3 directory into NEW_UPDATES. If additional personal files have been added to OLD_UPDATES, then copy them as well into NEW_UPDATES.

3.12.1 The recommended method to incorporate personal changes

Go to the NEW_UPDATES sub-directory and use "gdiff" or "xdiff" to compare each file (one at a time) in OLD_UPDATES with the corresponding one in NEW_UPDATES. As an example, consider the file "grids.F" and use the command

```
gdiff OLD_UPDATES/grids.F grids.F
```

Within "gdiff", click the right mouse button to bring up the option menu. Select PICK RIGHT to mark all changes from "grids.F" in NEW_UPDATES. Then scroll through the code and use the left mouse button to mark each local change to be taken from "grids.F" in OLD_UPDATES. In the event that part of a local change from OLD_UPDATES overlaps a change from NEW_UPDATES, an editor can be used afterwards to make the resulting code as intended. With "xdiff", individual lines from overlapping changes can be selected from each file which makes the use of an editor unnecessary. When done, use the right mouse button to select WRITE FILE. The correct filename and path "NEW_UPDATES/grids.F" should appear as the default. After clicking on the OK button, this new file containing all marked changes merged together will replace the existing file in NEW_UPDATES.
When finished, all local changes will have been transferred to the files in NEW_UPDATES. As a check, use "gdiff" to compare routines in NEW_UPDATES to the ones in the new MOM_3 directory. Only local changes should show up. As newer releases of MOM become available, the above strategy for upgrading is strongly recommended. This method has been in use at GFDL over the past year to incorporate new changes into the development version of MOM as well as to upgrade researchers from older versions to the development version.

3.12.2 An alternative recommended method

If there is no access to a "gdiff" or "xdiff" utility, the alternative method will work well. Compare each routine (one at a time) in OLD_UPDATES to the corresponding one in MOM_3_OLD to find local changes. Do this using "diff" with the "-e" option. As an example, take the file "grids.F" and use the command

```
diff -e  MOM_3_OLD/grids.F  OLD_UPDATES/grids.F > mods
```

Inspect the file "mods" to view local modifications which have been made. While viewing file 'mods" in an editor, open another editor and look for each modification in file OLD_UPDATES/grids.F. Find the corresponding location in file NEW_UPDATES/grids.F and use the "cut and paste" method to transfer the local modifications.

3.13 Finding all differences between two versions of MOM

As changes, bug fixes, and new parameterizations are incorporated into MOM, newer versions of the code and manual will be placed along older versions on the GFDL anonymous ftp server. To upgrade existing code to the latest version, refer to Section 3.12. To inspect what changes have been made since a previous version, create a NEW directory, then from within this NEW directory, uncompress and extract the tar file (e.g. *MOM3.xtar.Z*)to build the latest MOM_3 directory structure. Differences between old and new versions can be generated using

```
diff -r  MOM_3_OLD MOM_3_NEW > x
```

where option "r" generates differences between sub-directories recursively and places the differences in file "x" which may be a huge file. To find which routines have changes, use

```
 grep \^diff   x
```

Files that appear in one directory and not in the other can be found with

```
 grep \^Only   x
```

3.14 Applying bug fixes

In between releases of MOM, it is necessary to be able to correct bugs. If the number of lines of code needed to correct a bug is significantly smaller than the number of code lines in the file being corrected, then it is more efficient to supply the changes rather than the new file. For example, If file "changes" was constructed as

```
diff -e oldfile newfile > changes
```

then file "newfile" can be constructed from file "oldfile" using the following C shell script:

```
#! /bin/csh -f
# update script to build newfile from oldfile using changes which
# were generated by diff -e oldfile newfile > changes
if ($3 == "") then
  echo " "
  echo 'script "update" builds "newfile" from "oldfile" using "changes"'
  echo "which were generated by diff -e oldfile newfile > changes"
```

```
  echo "->usage: update oldfile changes newfile"
  exit
endif
set oldfile = $1
set chgs = $2
set newfile = $3
set work = .temp
cat $chgs > $work
echo "w $newfile" >> $work
echo "q $newfile" >> $work
ed $oldfile < $work
/bin/rm $work
echo "->Done building $newfile from $oldfile + $changes"
```

If the above script is saved as file "update", then the following one-liner will built the "newfile" from the "oldfile":

```
update oldfile changes newfile
```

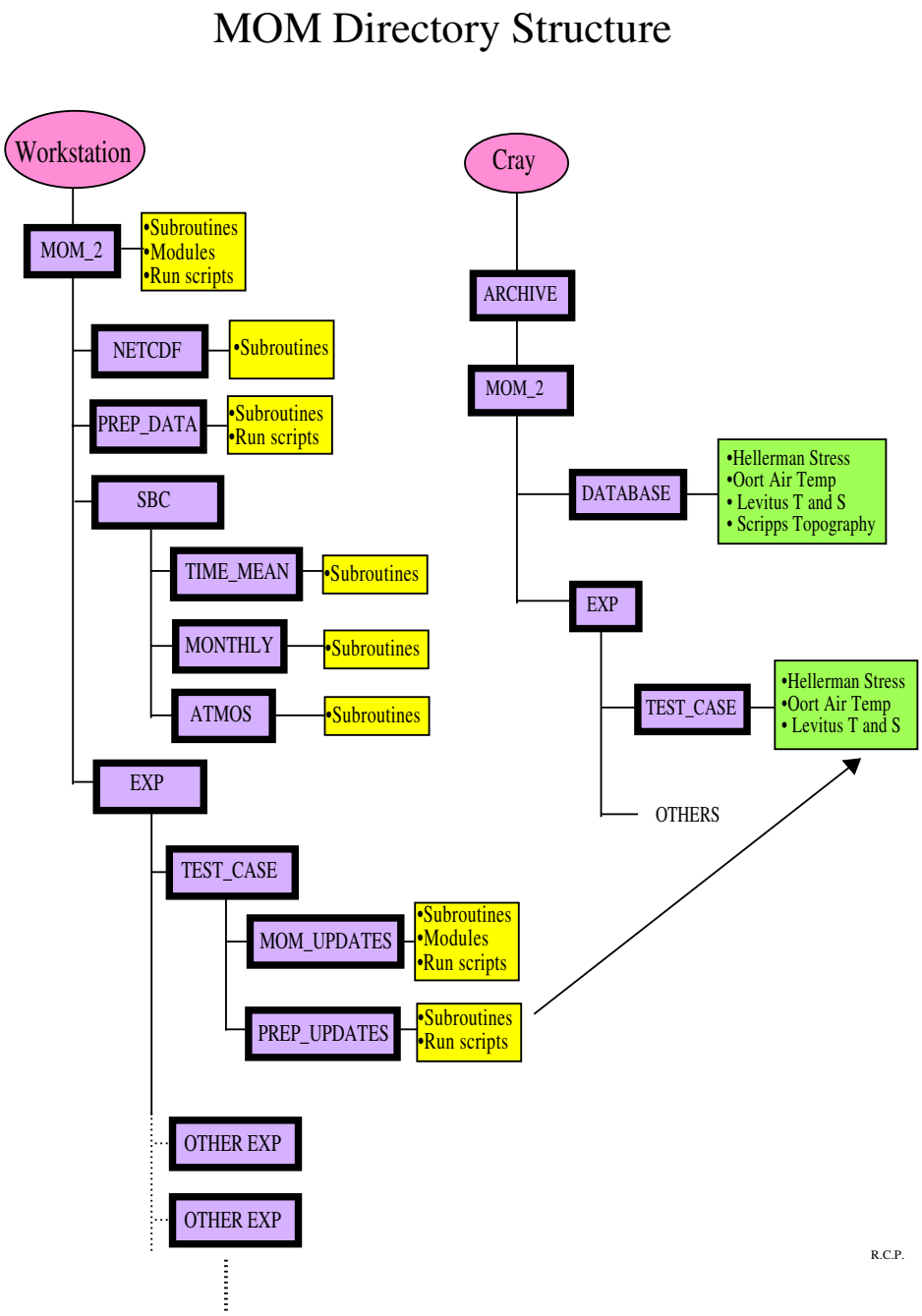


Figure 3.1: Directory structure for MOM at GFDL

Part II

Basic formulation

Chapter 4

Fundamental equations

The continuum equations discretized by MOM are introduced in this chapter. Subsequent chapters in this part of the manual will elaborate on the equations and the numerical methods used to realize solutions. Much of this discussion was written with the help of Martin Schmidt (martin.schmidt@io-warnemuende.de).

4.1 Assumptions

MOM is a finite difference version of the *ocean primitive equations*, which govern much of the large scale ocean circulation. As described by Bryan (1969), the equations consist of the Navier-Stokes equations subject to the Boussinesq and hydrostatic approximations. The equation of state relating density to temperature, salinity, and pressure can generally be nonlinear, thus representing important aspects of the ocean's thermodynamics. Prognostic variables are the two active tracers potential temperature and salinity, the two horizontal velocity components, any number of passive tracer fields, and optionally the height of the free ocean surface. The discretization consists of spatial coordinates fixed in time (fully Eulerian), with surfaces of constant depth determining the vertical discretization and a spherical (latitude/longitude) grid for the horizontal.

As discussed by many authors, including the original paper by Boussinesq (1903), as well as Spiegel and Veronis (1960), Chandrasekhar (1961), Gill (1982), and Müller (1995), the Boussinesq approximation is justified for large-scale ocean modeling on the basis of the relatively small variations in density within the ocean. The mean ocean density profile $\rho_\circ(z)$ typically varies no more than 2% from its depth averaged value $\rho_\circ = 1.035\,\mathrm{g\,cm^{-3}}$ (page 47 of Gill 1982). The Boussinesq approximation consists of replacing $\rho_\circ(z)$ by its vertically averaged value $\rho_\circ$.[1] In order to account for density variations affecting buoyancy, the Boussinesq approximation retains the full prognostic density $\rho = \rho(\lambda, \phi, z, t)$ when multiplying the constant gravitational acceleration. Equivalently, the vertical scale for variations in the vertical velocity is much less than the vertical scale for variations in $\rho_\circ(z)$, and fluctuating density changes due to local pressure variations are negligible. The latter implies that the fluid can be treated as incompressible, which excludes sound and shock waves.

In addition to the Boussinesq approximation, Bryan imposed the hydrostatic approximation, which implies that vertical pressure gradients are due only to density. When horizontal

[1] In MOM 1 and the Cox versions of the model, $\rho_\circ$ was set to $1.0\,\mathrm{g\,cm^{-3}}$ (a difference of 3.5% relative to the accepted value of $1.035\,\mathrm{g\,cm^{-3}}$) to eliminate a few multiplies in the momentum equations for reasons of computational speed. MOM 2 and subsequent versions use $\rho_\circ = 1.035\,\mathrm{g\,cm^{-3}}$.

scales are much greater than vertical scales, the hydrostatic approximation is justified and, in fact, is identical to the long-wave approximation for continuously stratified fluids. According to Gill (1982), the ocean can be thought of as being composed of thin sheets of fluid in the sense that the horizontal extent is very much larger than the vertical extent[2]. Therefore, kinetic energy is largely dominated by horizontal motions.

Consistent with the above approximations, Bryan also made the thin shell approximation because the depth of the ocean is much less than the earth's radius, which itself is assumed to be a constant (i.e., a sphere rather than an oblate spheroid). The thin shell approximation amounts to replacing the radial coordinate of a fluid parcel by the mean radius of the earth, unless this cordinate is differentiated. Correspondingly, the Coriolis component and viscous terms involving vertical velocity in the horizontal momentum equations are ignored on the basis of scale analysis. These assumptions form the basis of the Traditional Approximation. For a review and critique of the Traditional Approximation, as well as for a review of the typical approximations made in ocean modeling, see Marshall *et al.* (1997). Additionally, the thesis by Adcroft (1995) provides added details regarding the different dynamical processes omitted upon making the various approximations.

For the handling of subgrid scale (SGS) processes, Bryan made an eddy viscosity/diffusivity hypothesis. This hypothesis says that the affect of sub-grid scale motion on larger scale motions can be accounted for in terms of eddy mixing coefficients, whose size is many orders of magnitude larger than the molecular values. This hypothesis is controversial, and likely will remain so as long as turbulence remains a fundamentally unsolved problem. Pragmatically, however, some form of this approximation appears necessary in order to maintain numerical stability. Much of the research since Bryan revolves around SGS parameterizations. The hope is that such work will yield more physically based and consistent SGS assumptions.

Bryan made the rigid lid approximation to filter out external gravity waves. The speed of these waves places a severe limitation on economically solving the equations numerically. As noted above, displacements of the ocean surface are relatively small. Their affect on the solution is represented as a pressure against the rigid lid at the ocean surface. Two options in MOM relax the rigid lid approximation by allowing a free surface.

4.2 The primitive equations

The continuous equations solved by MOM are given by

$$u_t = -\nabla \cdot (u\,\mathbf{u}) + v\left(f + \frac{u \tan\phi}{a}\right) - \left(\frac{1}{a\,\rho_\circ \cdot \cos\phi}\right) p_\lambda + (\kappa_m u_z)_z + F^u \tag{4.1}$$

$$v_t = -\nabla \cdot (v\,\mathbf{u}) - u\left(f + \frac{u \tan\phi}{a}\right) - \left(\frac{1}{a\,\rho_\circ}\right) p_\phi + (\kappa_m v_z)_z + F^v \tag{4.2}$$

$$w_z = -\nabla_h \cdot \mathbf{u}_h \tag{4.3}$$

$$p_z = -\rho\, g \tag{4.4}$$

[2]Note that this is not valid if the purpose is to accurately model convection where horizontal and vertical scales may be comparable. See Marshall *et al.* (1997) for more details.

$$\theta_t = -\nabla \cdot [\mathbf{u}\,\theta + \mathbf{F}(\theta)] \tag{4.5}$$

$$s_t = -\nabla \cdot [\mathbf{u}\,s + \mathbf{F}(s)] \tag{4.6}$$

$$\rho = \rho(\theta, s, z). \tag{4.7}$$

The coordinate ϕ is latitude, which increases northward and is zero at the equator. λ is longitude, which increases eastward with zero defined at an arbitrary longitude (e.g., Greenwich, England). z is the vertical coordinate, which is positive upwards and zero at the surface of a resting ocean. Boldface characters represent vector quantities.

4.2.1 Basic constants and parameters

All units in MOM are *cgs*.

- The Boussinesq density is given by (page 47, Gill 1982)

$$\rho_o = 1.035\,\mathrm{g\ cm^{-3}}. \tag{4.8}$$

- The mean acceleration from gravity is given by

$$g = 980.6\,\mathrm{cm\ s^{-2}}. \tag{4.9}$$

- The mean radius of the earth is given by

$$a = 6371 \times 10^5\,\mathrm{cm}. \tag{4.10}$$

  This is the radius of a sphere having the same volume as the earth (page 597, Gill 1982). For earth, the equatorial radius is about 6378 km and the polar radius is about 6357 km. Neglect of such non-spheroidal effects is ubiquitous in ocean modeling. For a discussion of the differences between spheroidal and the more exact oblate-spheroidal, refer to the discussion in Veronis (1973).

- The Coriolis parameter is given by

$$f = 2\Omega \sin\phi. \tag{4.11}$$

  The earth's angular velocity Ω is comprised of two main contributions: the spin of the earth about its axis, and the orbit of the earth about the sun. Other heavenly motions can be neglected. Therefore, in the course of a single period of 24 * 3600 = 86400 s, the earth experiences an angular rotation of $(2\pi + 2\pi/365.24)$ radians. As such, the angular velocity of the earth is given by

$$\begin{aligned} \Omega &= \left(\frac{2\pi + 2\pi/365.24}{86400\mathrm{s}}\right) \\ &= \left(\frac{\pi}{43082}\right)\mathrm{s}^{-1} \\ &= 7.292 \times 10^{-5}\mathrm{s}^{-1}. \end{aligned} \tag{4.12}$$

4.2.2 Hydrostatic pressure and the equation of state

The pressure p is diagnosed through the hydrostatic equation (4.4). In this equation, the *in situ* density ρ is employed, where $\rho = \rho(\theta, s, p)$ is a diagnostic expression for the equation of state. Note that traditionally, $\rho = \rho(\theta, s, z)$ is used to evaluate the equation of state, rather than $\rho = \rho(\theta, s, p)$. Due to the strong hydrostatic nature of the ocean, the horizontal pressure variations are neglected for the purpose of evaluating the equation of state. Dewar *et al.* (1997) discuss the potential inaccuracies associated with this approach. Recent MOM implementations include the ability to diagnose density using the actual pressure from the previous model time step.

4.2.3 Horizontal momentum equations

Equations (4.1) and (4.2) are the horizontal momentum equations. The velocity field

$$\mathbf{u} = (u, v, w) = (\mathbf{u}_h, w) \tag{4.13}$$

is written in terms of the zonal, meridional and vertical components, respectively. The vertical velocity is diagnosed through the continuity equation (4.3). Horizontal velocities are driven by the following terms:

4.2.3.1 Coriolis force

The Coriolis force

$$\mathbf{F}_C = \mathbf{u} \wedge f\,\hat{z} = f(v, -u, 0) \tag{4.14}$$

arises from writing the equations in the rotating reference frame of the earth. This force does zero work on the fluid, since $\mathbf{u} \cdot \mathbf{F}_C = 0$. In the northern hemisphere, this force acts to the right of the fluid velocity vector, and in the south, it acts to the left.

4.2.3.2 Horizontal pressure gradient

The horizontal pressure gradient $-\nabla_h p$ acts to drive the fluid towards regions of low pressure. This gradient arises from spatial gradients in the pressure at the ocean surface (the *barotropic pressure gradient*) and pressure interior to the ocean (the *baroclinic pressure gradient*). Baroclinic pressure gradients arise from density gradients as determined through the hydrostatic relation. The steady state balance of the Coriolis force and the total pressure gradient force forms the *geostrophic balance*. This balance is relevant for large scale ocean circulation. All of the other terms in the velocity equation act to make the flow deviate from geostrophy.

4.2.3.3 Advection

The convergence of the advective fluxes $-\nabla_h \cdot (u\,\mathbf{u}_h)$ and $-\nabla_h \cdot (v\,\mathbf{u}_h)$ provide fundamental sources of nonlinearity to the equations of motion. They arise from the use of an Eulerian, rather than a Lagrangian, reference frame for describing the fluid motion. Their presence adds substantially to both the richness and complexity of fluid dynamics.

4.2.3.4 Nonlinear advective "metric" term

The term

$$a^{-1}\, u\, (\mathbf{u} \wedge \hat{z})\, \tan\phi \tag{4.15}$$

arises from the curvature of the earth. Since the longitudinal and latitudinal unit vectors $(\hat{\lambda}, \hat{\phi})$ are not material constants, they contribute to the material time derivative of the velocity vector

$$\begin{aligned} \frac{D\mathbf{u}_h}{Dt} &= \frac{D(\hat{\lambda}u + \hat{\phi}v)}{Dt} \\ &= \hat{\lambda}\,\frac{Du}{Dt} + \hat{\phi}\frac{Dv}{Dt} + u\,\frac{D\hat{\lambda}}{Dt} + v\,\frac{D\hat{\phi}}{Dt}. \end{aligned} \tag{4.16}$$

For a derivation of the material derivatives of the unit vectors, see Section 2.3 of Holton (1992). These these extra "metric" terms vanish when working on a plane, such as the f-plane or β-plane, since for this case the unit vectors $\hat{x}$ and $\hat{y}$ are constant. Additionally, this term, just as the Coriolis force, does zero work on the fluid.

4.2.3.5 Vertical friction

The vertical friction $(\kappa_m\, \mathbf{u}_h)_z$ parameterizes the vertical exchange of horizontal momentum due to subgrid scale processes. An eddy-viscosity hypothesis is the basis for the form of the friction. No "metric terms" are needed, regardless of whether the vertical viscosity is constant or spatially dependent.

4.2.3.6 Horizontal friction

Horizontal momentum friction parameterizes the exchange of horizontal momentum from the SGS processes to the grid scale. An eddy-viscosity hypothesis is the basis for the form of the friction. In general, this friction acts to dissipate kinetic energy without introducing spurious sources of angular momentum. MOM provides options in which the friction can be second order (Laplacian) or a fourth order (biharmonic). Details concerning the derivation of these operators are provided in Chapter 9.

4.2.4 Tracer equations

Equations (4.5) and (4.6) are the equations for the active tracers potential temperature θ and salinity s. Potential temperature is used rather than *in situ* temperature because it more closely approximates a conservative variable. However, the paper by McDougall and Jackett (1998) provide motivation to employ a modified version of potential temperature which even more closely approximates a conservative variable than the traditional definition of potential temperature. Note that in an adiabatic ocean for which nonlinear equation of state effects are ignored, both salinity and potential temperature are materially conserved active tracers.

The flux vector $\mathbf{F}$ takes on one of a variety of forms depending on the choice of subgrid scale parameterizations. For example, an older choice is to use horizontal and vertical diffusion

$$\mathbf{F}_h(T) = -A_h\, \nabla_h T \tag{4.17}$$

$$F^z(T) = -\kappa_h T_z, \tag{4.18}$$

where A_h is the horizontal diffusivity ($\mathrm{cm}^2\,\mathrm{s}^{-1}$) and κ_h ($\mathrm{cm}^2\,\mathrm{s}^{-1}$) is the vertical diffusivity. The "h" subscript on the diffusivities is historical, and it stands for "heat." As discussed in Section B.3.0.5 of Appendix B, there is no problem with the use of vertically aligned tracer diffusion as a framework for parameterizing dianeutral processes. Yet there is a fundamental problem with horizontally aligned diffusive fluxes. The problem is that the ocean has a strong tendency to diffuse along, rather than across, directions of constant locally referenced potential density (the *neutral directions* as described by McDougall 1987), rather than constant depth. The differences can be nontrivial for certain regions of the ocean, especially in western boundary currents (Veronis 1975). For this reason, preference is given to use of the isoneutral diffusion tensor (see Section 35.1) rather than horizontal diffusion. Another increasingly common choice is to add the Gent-McWilliams eddy induced transport, which can be formulated as a skew-diffusion (see Section 35.1.6). Given these two choices, along with vertical diffusion, the flux for an arbitrary tracer T is then given by

$$\mathbf{F}_h(T) = -A_I \nabla_h T - (A_I - \kappa)\mathbf{S}\, T_z \tag{4.19}$$

$$F^z(T) = -(A_I + \kappa)\,\mathbf{S}\cdot\nabla_h T - (\kappa_h + A_I S^2)\, T_z, \tag{4.20}$$

where

$$\mathbf{S} = -\left(\frac{\nabla_h \rho}{\rho_z}\right) \tag{4.21}$$

is the isoneutral slope vector with magnitude S, A_I is the isoneutral diffusivity, and κ is the "thickness diffusivity." Note that κ parameterizes the mixing of thickness only in the case when κ is constant. The distinction is discussed by McDougall (1998). Taking $A_I = \kappa$ is common, and it simplifies the horizontal tracer flux tremendously.

The hydrostatic approximation necessitates the use of a parameterization of vertical overturning associated with gravitationally unstable water columns. This parameterization is often represented in the model by a convective adjustment algorithm, as described in Section 33.1. Other means to gravitationally stabilize the column are through the use of a very large value of the vertical diffusivity, thus enhancing the vertical flux. More on vertical convection is discussed in Section 33.1.

Finally, the model allows for the input of various tracer source terms, which may represent, for example, a radioactive source for a passive tracer. These terms are not explicitly represented in the equations written above for purposes of brevity.

4.3 Boundary and initial conditions

The system of model equations (4.1)-(4.7) is completed by a set of boundary conditions. The kinematic boundary conditions define the ocean domain and describe its volume budget. The sea floor is defined by specifying a surface with no normal flow. The sea surface is defined by means of an equation of motion of the sea surface height.

The dynamic boundary conditions prescribe the flux of various quantities such as momentum, heat, and passive tracers through the ocean boundaries. The sea floor employs insulation conditions for heat, salt, and passive tracers; i.e., no-normal tracer flux through the sea floor. Geothermal heating can be set through the model's tracer source field. At the ocean surface, boundary conditions are supplied for heat, passive tracers, and momentum. If fresh water flux is not considered in the volume budget, an additional *virtual* salinity flux is needed.

4.3.1 Bottom kinematic boundary condition

The bottom kinematic boundary condition is the no-normal flow condition

$$\hat{n}_{bottom} \cdot \mathbf{u} = 0. \tag{4.22}$$

Namely, with the ocean bottom defined by the algebraic expression $f(\lambda, \phi, z) = z + H(\lambda, \phi) = 0$, the unit vector normal to the ocean bottom is given by

$$\begin{aligned} \hat{n}_{bottom} &= \frac{\nabla f}{|\nabla f|} \\ &= \frac{(\nabla_h H, 1)}{\sqrt{1 + |\nabla_h H|^2}}. \end{aligned} \tag{4.23}$$

As such, the no-normal flow condition implies

$$w = -\mathbf{u}_h \cdot \nabla_h H \qquad z = -H(\lambda, \phi). \tag{4.24}$$

For the degenerate case of a steep sidewall, $\hat{n}_{bottom}$ orients horizontally and the usual lateral boundary condition

$$\hat{n}_{wall} \cdot \mathbf{u}_h = 0 \tag{4.25}$$

is retained, where $\hat{n}_{wall}$ is a horizontal unit vector normal to the sidewall.

A second derivation uses the fact that the bottom is a material surface at $z = -H(\lambda, \phi)$. This fact means that a particle initially on the bottom will remain so, and hence

$$\frac{D(z + H)}{Dt} = 0. \tag{4.26}$$

This result implies again equation (4.24).

For the rigid lid, a third derivation allows for a more ready implementation in MOM of the bottom condition. The bottom kinematic condition can be generated by integrating Equation (4.3) from the surface to the ocean bottom using Equations (4.1), (4.2), (6.4), and (6.5). The finite difference equivalent of this method is used to generate vertical velocities in the interior as well as at the bottom. A more complete discussion of the discrete vertical velocity at the ocean bottom is given in Section 22.3.3.

4.3.2 Surface kinematic boundary condition

In order to construct the boundary condition to be placed at the free surface $z = \eta$ (see Figure 4.1), consider the algebraic equation which defines the position of the free surface

$$\eta - z = 0. \tag{4.27}$$

In the special case when there is zero water penetrating the free surface, $D(\eta - z)/Dt = 0$ since the free surface in this case is a material boundary for which a particle initially on the boundary will remain on the boundary. The same reasoning was used previously to derive the bottom kinematic boundary condition. The advent of a fresh water flux means that the free surface is generally not an impermeable material boundary. Rather, the material time derivative of $(\eta - z)$ has a source term determined by the fresh water flux

$$\frac{D(\eta - z)}{Dt} = q_w \qquad z = \eta, \tag{4.28}$$

where q_w is the volume per unit time per unit area (dimensions of a velocity) of fresh water entering the ocean through the free surface ($q_w > 0$ for water entering the ocean across the free surface). This result leads to the surface kinematic boundary condition

$$(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta = w + q_w \qquad\qquad z = \eta. \tag{4.29}$$

As such, the surface height $z = \eta$ has a time tendency $\partial_t \eta$ determined by an advective flux of height $-\mathbf{u}_h \cdot \nabla_h \eta$, the Eulerian vertical velocity $w(\eta)$, and the fresh water velocity q_w.

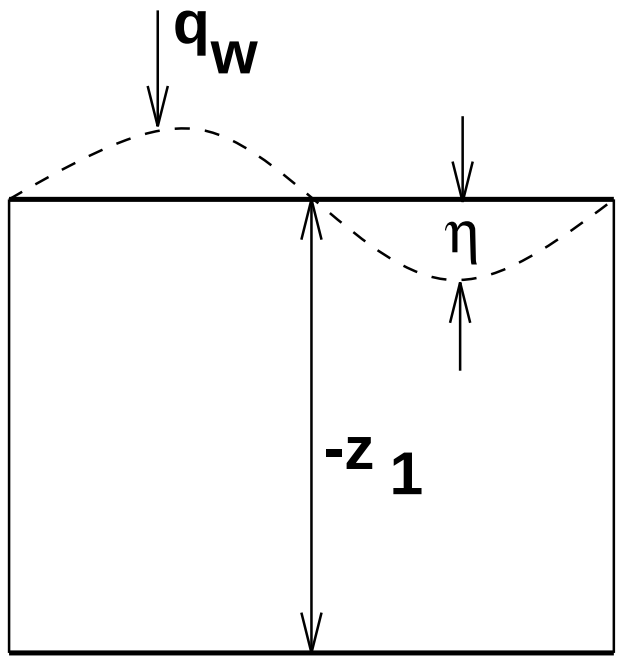


Figure 4.1: Sketch of a shallow layer of fluid with a free surface and whose lower interface is flat. Both the top and bottom boundaries are generally open to fluid. The height η is the deviation from a resting ocean state, and $z = z_1 < 0$ is the vertical position of the layer bottom.

4.3.3 Dynamic boundary conditions

The purpose of this section is to discuss the dynamic boundary conditions, which are conditions that prescribe the momentum flux through the model's side, bottom, and top boundaries.

As discussed in Section 7.4.1, bottom stress arises from both resolved topography, as well as unresolved or sub-grid scale (SGS) topography and bottom boundary layer effects. In MOM, it is possible to parameterize the SGS bottom stress either as a free-slip bottom drag,

$$\tau_{bottom-sgs} \;=\; 0, \tag{4.30}$$

or in terms of the flow near the bottom

$$\tau_{bottom-sgs} \;=\; \rho_o C_D |\mathbf{u}_h| \mathbf{u}_h. \tag{4.31}$$

Issues related to the stress arising from resolved topography with the full and partial cells are discussed in Chapter 26, and the bottom boundary layer issues are discussed in Chapter 37.

The momentum flux through the sea surface τ_{surf} (dyn cm^{-2}) comes from two sources:

$$\tau_{surf} = \tau_{winds} + \tau_{fresh}, \tag{4.32}$$

which are the wind stress τ_{winds} and momentum transfer in connection with a fresh water flux, τ_{fresh}. The dominating mechanism is the wind stress which comes from the interaction of the wind field with the ocean surface waves. Since the atmosphere-ocean boundary layer is not resolved by the model, it is parametrized, e.g., as function of the wind speed in some reference height in the boundary layer. A simple example is

$$\tau_{winds} = \rho_a C_D^{wind} |\mathbf{u}^{wind}| \mathbf{u}^{wind}, \tag{4.33}$$

where ρ_a is the density of the air, $\mathbf{u}^{wind}$ is the wind speed and C_D^{wind} a drag coefficient which depends on the wind speed, but also on the stability of the atmospheric boundary layer and the wave height. Generally, the physically correct calculation of the wind stress is not well known. Such uncertainty has prompted some climate modelers to consider coupling their ocean model to a surface wave model. The wave model then directly feels the winds from the atmosphere and is able to more accurately compute the surface stress field for use in the ocean model.

The other mechanism for the vertical momentum transfer is fresh water flux. The fresh water volume flux through the air-sea interface carries a momentum which is approximately

$$\tau_{fresh} = q_w \rho_f \mathbf{u}^{wind}. \tag{4.34}$$

As discussed in Section 7.4.2, MOM identifies this flux with $q_w \rho_o \mathbf{u}(z = \eta)$, where $\mathbf{u}(z = \eta)$ is the horizontal current at the ocean surface. With a resolved boundary layer model, such as a wave model, this identification would not necessarily be exact.

Momentum flux through lateral boundaries is given by no-normal flow as well as no-slip boundary conditions. Therefore, all velocity components next the side walls are set to zero. The means for doing so are through the model's land-sea mask. Although the model employs no-slip next to the side boundaries, all that is necessary for formulating the solution methods for the tracer and momentum equations is the no-normal flow condition. This is an important point since the distinction made in MOM between "side" and "bottom" is possible only through its use of artificial stepped topography. In the real ocean, there clearly is no distinction. In principle, therefore, the methods employed in MOM can be used for a free-slip model with a smooth representation of the bottom.

The details on how the prescribed momentum flux through the model boundaries is linked with the model variables are described in Sections 6.4.1.

4.3.4 Tracer fluxes through the model boundaries

The tracer equations as given in Section 4.2.4 require the specification of tracer fluxes through the model boundaries. Tracer fluxes through the lateral boundaries and the sea-floor are generally set to zero. The fluxes through the sea surface as heat flux, fresh water flux or radiation are maintained by turbulent processes in the atmosphere-ocean boundary layer which is not resolved by MOM. Parameterizations require input from both, such as the sea surface temperature, surface salinity, surface air temperature, humidity, and/or wind speed. MOM provides a coupling module which collects the required variables and permits the calculation of the tracer surface fluxes. This includes information which is not calculated by MOM but must be provided from a database or an atmosphere model. The relation of the surface tracer fluxes to the scheme for the calculation of the ocean model variables is given in Section 8. Simple parameterizations for the heat flux and the fresh water flux are discussed in Section 8.4.

4.3.5 Open boundaries and sponges

Limited domain models must consider the manner for which open boundaries are handled. In general, the mathematical consistency of open boundaries is not clear, so much care should be exercised. In MOM, the methodology of Stevens (1990) has been implemented (Section 28.3.5).

An alternative to open boundaries is to restore fields along the boundary to some prescribed values. This method goes by the name of "sponge" boundary conditions. Details of the MOM implementation of sponges is described in Section 28.3.4.

4.3.6 Initial conditions

Initial conditions for model experiments on a global scale typically consist of specifying a density structure through potential temperature and salinity, with the ocean at rest. Limited area models often start with prescribed nonzero velocity fields.

4.4 Comments on volume versus mass conservation

MOM assumes the volume of a fluid parcel is conserved, unless the parcel is affected by external sources such as surface fresh water fluxes. This assumption is part of the standard Boussinesq approximation. Mass conservation is more fundamental than volume conservation. One place where the limitations of volume conservation are most apparent is when formulating the equations for the free surface. This section briefly discusses this point.

4.4.1 Volume conservation

Consider a shallow layer of fluid with a free surface as shown in Figure 4.1. For definiteness, such a fluid layer can be considered the fluid which occupies the surface layer of the ocean model. The position $z = \eta(\lambda, \phi, t)$, which could be negative, is the vertical deviation from a resting ocean state $z = 0$. The position $z = z_1 < 0$ is the fixed position of the bottom of the layer. The volume of an infinitesimally thin column of fluid extending over the finite vertical extent of this layer is given by

$$\delta V = h\,\delta A, \tag{4.35}$$

where

$$h = -z_1 + \eta \tag{4.36}$$

is the height of the fluid column, and δA is its infinitesimal horizontal cross-sectional area. Volume conservation implies that the change of the box volume with time equals the sum of all volume fluxes through the box surface,

$$\partial_t(\delta V) \quad = \quad \left(q_w + w_1 - \nabla_h \cdot \int_{z_1}^{\eta} dz\, \mathbf{u}_h \right) \delta A. \tag{4.37}$$

The convergence of the horizontal flux stems from the infinitesimal horizontal extension of the box. $w_1\,\delta A$ is the volume per unit time crossing through the bottom of the layer, where $w_1 = w(z = z_1)$ is positive for water moving upward into the surface layer. $q_w\,\delta A$ is the volume per unit time of ocean water appearing in the surface box. In the spirit of a volume conserving model, it is equivalent to the volume of fresh water per unit time crossing the free surface, with

$q_w > 0$ indicating water entering the ocean. The accuracy of this equivalence is determined by the deviation of the ratio of the fresh water density ρ_f and the ocean density from unity

$$\frac{\rho_f}{\rho(z=\eta)} - 1. \tag{4.38}$$

For the most applications this deviation should be smaller than the accuracy of the fresh water flux data.

The identity

$$\partial_t(\delta V) \quad = \quad \delta A\, \partial_t h \tag{4.39}$$

leads to the balance for the layer thickness

$$\partial_t h + \nabla_h \cdot \int_{z_1}^{\eta} \mathbf{u}_h \quad = \quad q_w + w_1. \tag{4.40}$$

With a uniform velocity in the surface layer, this result takes the more familiar form

$$\partial_t h + \nabla_h \cdot (h\, \mathbf{u}_h) \quad = \quad q_w + w_1. \tag{4.41}$$

4.4.2 Mass conservation

Now consider the mass of the infinitesimal column of water

$$\delta m \quad = \quad \int_{z_1}^{\eta} dz\, \rho\, \delta A. \tag{4.42}$$

In this expression, ρ is the mass density. The column mass changes when either the volume or the density is changed,

$$\partial_t(\delta m) \quad = \quad \rho(\eta)\, \delta A\, \partial_t \eta + \int_{z_1}^{\eta} dz\, \partial_t \rho\, \delta A. \tag{4.43}$$

Mass conservation implies that this change is due to mass flux through the box surface, i.e., from the convergence of the horizontal mass flux and from the water coming through the bottom and through the free surface

$$\partial_t(\delta m) \quad = \quad \left(Q_w + w_1\, \rho_1 - \nabla_h \cdot \int_{z_1}^{\eta} dz\, \rho \mathbf{u}_h \right) \delta A, \tag{4.44}$$

where ρ_1 is the density of water entering from the bottom of the layer, and Q_w is the mass flux density of water entering through the free surface. This result leads to the mass balance equation for the surface layer

$$\partial_t \int_{z_1}^{\eta} dz\, \rho + \nabla_h \cdot \int_{z_1}^{\eta} \rho \mathbf{u}_h \quad = \quad Q_w + w_1\, \rho_1. \tag{4.45}$$

A more transparent form emerges from the assumption of a vertically uniform density in the surface layer, and $\delta m \approx \rho_s\, h\, \delta A$, which leads to

$$\partial_t(\rho_s\, h) + \nabla_h \cdot (h\, \rho_s\, \mathbf{u}_h) = Q_w + w_1\, \rho_1, \tag{4.46}$$

or in the alternate form

$$\partial_t h + \nabla_h \cdot (h\,\mathbf{u}_h) \quad = \quad \frac{w_1\rho_1 + Q_w - h\,D_h\rho_s/Dt}{\rho_s}. \tag{4.47}$$

Comparison with the volume conservation equation (4.41) reveals three differences. The first is the presence of the density ratio weighting the vertical velocitiy w_1. To leading order, this ratio is close to unity. The second is the occurence of the fresh water mass flux instead of the volume flux. The third difference is the fundamentally new term

$$-\frac{h}{\rho_s}\frac{D_h\rho_s}{Dt} \quad = \quad -\frac{h}{\rho_s}(\partial_t\rho_s + \mathbf{u}_h \cdot \nabla_h\rho_s) \tag{4.48}$$

This term acts to increase the surface height whenever the density of the surface layer is reduced, such as occurs when the layer is heated. It is this effect which is absent in the current formulation of MOM. A general way to incorporate this effect is to reformulate the model's equations in their non-Boussinesq form.

Differences between a volume conserving and mass conserving ocean model are discussed more thoroughly in the papers by Greatbatch (1994) and Mellor and Ezer (1995). Both papers argue that the difference in sea level height is a spatially independent, time dependent height.

4.4.3 Surface kinematic boundary conditions revisited

The discussion in Section 4.3.2 provided one method to derive the surface kinematic boundary condition. This section provides another, which is based on the volume or mass conserving balance equations for the layer thickness.

For a volume conserving fluid, the thickness equation (4.40) can be written

$$(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta \quad = \quad q_w + w_1 - \int_{z_1}^{\eta} dz\,\nabla_h \cdot \mathbf{u}_h. \tag{4.49}$$

To recover the surface boundary condition (4.29), vertically integrate the incompressibility condition (4.3), $\nabla \cdot \mathbf{u} = 0$, to yield an expression for the vertical velocity at the bottom of the surface layer

$$\begin{aligned} w_1 &= w(\eta) + \int_{\eta}^{z_1} dz\,w_z \\ &= w(\eta) + \int_{z_1}^{\eta} dz\,\nabla_h \cdot \mathbf{u}_h. \end{aligned} \tag{4.50}$$

This expression in equation (4.49) then yields the surface kinematic boundary condition

$$(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta \quad = \quad q_w + w(\eta). \tag{4.51}$$

For a mass conserving fluid, the thickness equation (4.45) can be written

$$\rho(\eta)(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta \quad = \quad w_1\,\rho_1 + Q_w - \int_{z_1}^{\eta} dz\,(D_h\rho/Dt + \rho\,\nabla_h \cdot \mathbf{u}_h) \tag{4.52}$$

With the identity

$$w_1\rho_1 \quad = \quad w(\eta)\rho(\eta) - \int_{z_1}^{\eta} dz\,\frac{\partial(w\rho)}{\partial z} \tag{4.53}$$

the horizontal Langrangian derivative $D_h\rho/Dt$ can be completed to the full Lagrangian derivative

$$\rho(\eta)(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta \;=\; w(\eta)\,\rho(\eta) + Q_w - \int_{z_1}^{\eta} dz\,(D\rho/Dt + \rho\,\nabla\cdot\mathbf{u}). \tag{4.54}$$

The term on the right hand side under the integral vanishes due to mass conservation

$$\frac{D\rho}{Dt} + \rho\,\nabla\cdot\mathbf{u} \;=\; 0. \tag{4.55}$$

As such, one recovers the surface kinematic boundary condition appropriate for a mass conserving fluid

$$(\partial_t + \mathbf{u}_h \cdot \nabla_h)\,\eta \;=\; w + \frac{Q_w}{\rho} \qquad z = \eta. \tag{4.56}$$

The quantity

$$q_w = \frac{Q_w}{\rho(\eta)} \tag{4.57}$$

is the volume flux in a mass conserving model. Thus, the only apparent difference from the volume conserving kinematic condition (4.29) is the approximation used for the calculation of the volume flux through the sea surface. However, other differences as thermal expansion are hidden in the vertical velocity $w(\eta)$.

4.5 Flux form and finite volumes

In general, MOM implments tracer and momentum advection as the divergence of a flux, rather than the advective form $\mathbf{u}\cdot\nabla\Psi$. In the continuum with an incompressible fluid, the advective form $\mathbf{u}\cdot\nabla\Psi$ and flux form $\nabla\cdot(\Psi\,\mathbf{u})$ are equivalent. In a numerical model, the flux formulation provides a straightforward way to ensure conservation properties of scalar quantities, and it allows a clear finite volume interpretation of the discrete equations.

As discussed by Adcroft *et al* (1996), a finite volume approach aims to formulate the discrete equations as self-consistent approximations of the volume integrated continuum equations, where the volume integration is taken over the a grid cell control volume. Such an approach is natural on a C-grid. With the B-grid in MOM, there are difficulties. Most notably, the bottom for a tracer cell does admit a finite volume interpretation. However, the bottom velocity cells do not rest on the ocean bottom (see Section 22.3.3). This is a notable instance where MOM does not respect the traditional finite volume approach.

4.6 Some basic formulae and notation

Before closing this chapter, it is useful to summarize some of the formulae, definitions, and notation which will at times be useful in this manual.

4.6.1 Differential operators

In MOM, the radial coordinate is taken as

$$r = a + z \tag{4.58}$$

where a is the earth's radius. $z = 0$ is assumed to be the position of the resting ocean, which is defined parallel to the geoid. $z = -H(\lambda, \phi)$ is the position of the ocean bottom. As mentioned earlier, although the geoid is not a perfect sphere, the relatively mild deviations from a sphere are ignored in MOM, which allows for spherical coordinates. Finally, since $z = r - a$, the unit vector $\hat{z}$ points in the radial direction $\hat{r}$

$$\hat{z} = \hat{r}. \tag{4.59}$$

Consistent with the Traditional Approximation (see Marshall *et al.* 1997 for a review), the differential operators used in the model take on the following form (see Appendix A of Washington and Parkinson (1986) for derivations). The gradient operator is given by

$$\begin{aligned} \nabla\Psi &= \hat{\lambda}\left(\frac{\Psi_\lambda}{a\cos\phi}\right) + \hat{\phi}\left(\frac{\Psi_\phi}{a}\right) + \hat{z}\,\Psi_z \\ &= \nabla_h\Psi + \hat{z}\,\Psi_z. \end{aligned} \tag{4.60}$$

The three-dimensional divergence operator acting on a vector $\mathbf{u} = (\mathbf{u}_h, w)$ is given by

$$\begin{aligned} \nabla\cdot\mathbf{u} &= \left(\frac{1}{a\cos\phi}\right)[u_\lambda + (v\cos\phi)_\phi] + w_z \\ &= \nabla_h\cdot\mathbf{u}_h + w_z. \end{aligned} \tag{4.61}$$

If $\mathbf{u}$ is the velocity field, then its three dimensional divergence vanishes since the fluid is always assumed incompressible in MOM. The three-dimensional curl operator acting on the velocity is given by

$$\omega = \hat{\lambda}\left(\frac{1}{a}\frac{\partial w}{\partial\phi} - \frac{\partial v}{\partial z}\right) + \hat{\phi}\left(\frac{\partial u}{\partial z} - \frac{1}{a\cos\phi}\frac{\partial w}{\partial\lambda}\right) + \hat{z}\left(\frac{1}{a\cos\phi}\frac{\partial v}{\partial\lambda} - \frac{1}{a\cos\phi}\frac{\partial(u\cos\phi)}{\partial\phi}\right), \tag{4.62}$$

where $\omega = \nabla\wedge\mathbf{u}$ is the three dimensional vorticity vector. Often, the vertical component of the vorticity will be written

$$\zeta = \hat{z}\cdot\omega = \left(\frac{1}{a\cos\phi}\right)\left[v_\lambda - (u\cos\phi)_\phi\right]. \tag{4.63}$$

The three-dimensional Laplacian is given by

$$\begin{aligned} \nabla\cdot(\nabla\Psi) &= \left(\frac{1}{a^2\cos\phi}\right)\left(\frac{1}{\cos\phi}\Psi_{\lambda\lambda} + (\cos\phi\cdot\Psi_\phi)_\phi\right) + \Psi_{zz} \\ &= \nabla_h\cdot(\nabla_h\Psi) + \Psi_{zz}. \end{aligned} \tag{4.64}$$

4.6.2 Leibnitz's Rule

Leibnitz's Rule for differentiation of integrals

$$\begin{aligned}\frac{\partial}{\partial x}\int_{f(x)}^{g(x)} dx'\, F(x,x') &= F(x,g(x))\frac{\partial g(x)}{\partial x} - F(x,f(x))\frac{\partial f(x)}{\partial x} \\ &+ \int_{f(x)}^{g(x)} dx'\, \frac{\partial}{\partial x}F(x,x') \end{aligned} \tag{4.65}$$

is employed especially when dealing with vertical integrals where the bottom topography $z = -H$ and free surface height $z = \eta$ are integration limits.

4.6.3 Cross-products and the Levi-Civita symbol

In this manual, cross products are sometimes written with the notation

$$\mathbf{A} \times \mathbf{B} = \mathbf{A} \wedge \mathbf{B}. \tag{4.66}$$

This notation is consistent with many math and physics texts. Its use is helpful for those situations when the usual $\times$ symbol can be mistaken for the spatial variable x.

When writing the components of a vector cross-product, it is often useful to employ the Levi-Civita symbol ϵ_{ijk}

$$(\mathbf{A} \wedge \mathbf{B})_k = \epsilon_{ijk}\, A^i\, B^j, \tag{4.67}$$

where repeated indices are summed over the spatial directions. The Levi-Civita symbol ϵ_{ijk} is defined by

$$\epsilon_{ijk} = \begin{cases} 0, & \text{if any two labels are the same} \\ 1, & \text{if } i,j,k \text{ is an even permutation of } 1,2,3 \\ -1, & \text{if } i,j,k \text{ is an odd permutation of } 1,2,3. \end{cases} \tag{4.68}$$

This symbol is anti-symmetric on each pair of indices.

4.6.4 Area element and volume element on a sphere

When considering budgets over finite domains, integration over an element of the sphere is common. A useful bit of notation is the area of an infinitesimal element of the sphere

$$d\Omega = a^2 \cos\phi\, d\phi\, d\lambda = a^2\, d(\sin\phi)\, d\lambda. \tag{4.69}$$

With this notation, the volume element on the sphere takes the form

$$d\mathbf{x} = d\Omega\, dz. \tag{4.70}$$

4.6.5 Vertical grid levels

For the interior part of the model ocean, discrete cells have time independent depths $z_k < 0$. In this case, the interior "layers" are most often called "levels" to make the distinction with models for which the vertical coordinate evolves in time (e.g., isopycnal-layer models such as that of Bleck *et al.*, 1992, or Hallberg 1995), or models where the vertical coordinate is contoured

according to the bottom topography (i.e., sigma-layer models such as that of Blumberg and Mellor 1987 or Haidvogel *et al.* 1991). The top ocean cell, however, generally has a time dependent upper surface height

$$z_0 = \eta, \tag{4.71}$$

where η, which can be positive or negative, represents the vertical distance from the sea surface to the height $z = 0$ of a resting sea. In the rigid lid approximation, $\eta = 0$, and so all model cells have fixed volumes, whereas for the free surface, $\eta \neq 0$.

Chapter 5

Momentum equation methods

This chapter describes the methods available in MOM for solving the momentum equations. There are two basic approaches: the rigid lid and the free surface. Each approach itself has two different methods: the rigid lid streamfunction, the rigid lid surface pressure, the explicit free surface, and the implicit free surface. Currently, development work is focused on the explicit formulation of the free surface. The other methods essentially remain in their MOM 2 form. Plans are to only support development of the explicit free surface in the future (post Summer 1999). The reasons for this focus can be summarized by the following:

- The rigid lid is less physically complete than the free surface. Most notably, it does not allow for a direct treatment of fresh water fluxes.
- The rigid lid in MOM has not been parallelized, nor are there plans to do so.
- Both the explicit and implicit free surfaces have been parallelized in MOM. The explicit free surface shows enhanced scaling properties over the implicit scheme.
- The explicit free surface has been formulated so that it can be of use for either global climate integrations or limited area models.
- The implicit free surface algorithm is more complex than the explicit free surface.
- As of Summer 1999, the explicit free surface has incorporated the effects of the undulating top cell thickness into the depth dependent equations. Hence, the model is conservative of tracers, including total salt in the presence of fresh water fluxes, as well as momentum.

Each of these points will become more clear in this chapter as well as Chapters 8 and 29.

5.1 Separation into vertical modes

Numerical solutions are computed within MOM by dividing the ocean volume into a three dimensional lattice, discretizing the equations within each lattice cell, and solving these equations by finite difference techniques. The solution method could be formulated in a straightforward manner, but the result would be a numerically inefficient algorithm. Instead, Bryan (1969) introduced a fundamental technique to ocean modeling in which the ocean velocity field is separated into its depth averaged part and the deviation from the depth average. The following discussion introduces the motivations and ideas behind this approach.

5.1.1 Vertical modes in MOM and their relation to eigenmodes

As discussed in Section 6.11 of Gill (1982), the linearized primitive equations for a stratified fluid can be partitioned into a countably infinite (i.e., discrete) number of orthogonal eigenmodes, each with a different vertical structure. Gill denotes the zeroth vertical eigenmode the barotropic mode, and the infinity of higher modes are called baroclinic modes. Because of the weak compressibility of the ocean, wave motions associated with the barotropic mode are weakly depth dependent, and so correspond to elevations of the sea surface (see Hidgon and Bennett 1996 for a proof of the weak depth dependence in a flat bottomed ocean). Consequently, the barotropic field goes also by the name *external mode*. Barotropic or external waves constitute the fast dynamics of the ocean primitive equations. Baroclinic waves are associated with undulations of internal density surfaces, which motivates the name *internal mode*. Baroclinic waves, along with advection and planetary waves, constitute the slow dynamics of the ocean primitive equations.

For a flat bottom ocean, the vertical eigenmode problem is straightforward to solve, and many important ideas can be garnered from its analysis. For a free surface with a flat bottom, Gill shows that the barotropic mode has a vertical velocity which is approximately a linear function of depth, with the maximum vertical velocity at the free ocean surface and zero velocity at the flat bottom. In contrast, for a rigid lid and flat bottom ocean, the barotropic mode is depth independent and the vertical velocity identically vanishes. The baroclinic modes, as they are associated with movements of the internal interfaces, are little affected by the surface boundary condition. Therefore, the baroclinic modes in the free surface are quite similar to those in the rigid lid. Note that nonlinearities and nontrivial bottom topography generally couple the barotropic and baroclinic modes.

By construction, the depth averaged momentum equations only have solutions which depend on the horizontal directions. Consequently, the depth averaged mode of a rigid lid ocean model corresponds directly to the barotropic mode of the linearized rigid lid primitive equations. Additionally, the rigid lid model's depth dependent modes correspond to the baroclinic modes of the linearized rigid lid primitive equations. Therefore, depth averaging in the rigid lid model provides a clean separation between the linear vertical modes.

Just as for the rigid lid, the baroclinic modes are well approximated by the depth dependent modes of the free surface ocean model, since the baroclinic modes do not care so much about the upper surface boundary condition. In contrast, the ocean model's depth averaged mode cannot fully describe the free surface primitive equation's barotropic mode, which is weakly depth dependent. Therefore, some of the true barotropic mode spills over into the model's depth dependent modes. In other words, a linearized free surface ocean model's depth averaged mode is only approximately orthogonal to the model's depth dependent modes. It turns out that the ensuing weak coupling between the ocean model's fast and slow linear modes can be quite important for free surface ocean models, as described by Killworth *et al.* (1991) and Higdon and Bennett (1996). The coupling, in addition to the usual nonlinear interactionas associated with advection, topography, etc., can introduce pernicious linear instabilities whose form is dependent on details of the time stepping schemes.

Regardless of the above distinction between vertically averaged and barotropic mode for free surface models, it is common parlance in ocean modeling to refer to the vertically integrated mode as the barotropic or external mode. This terminology is largely based on the common use of the rigid lid approximation, for which there is no distinction. With the above discussion kept in mind, there should be no confusion, and so the terminology will be used in this manual for both the rigid lid and free surface formulations. Since there is little difference

between the rigid lid and free surface baroclinic modes, it is quite sensible to use this term to refer to the ocean model's depth dependent modes.

5.1.2 Motivation for separating the modes

Although there are several technical problems associated with the separation into the vertically averaged and vertically dependent modes, it is essential to build large scale ocean models using some version of this separation for the following reasons:

1. Without a separation, the full momentum field will be subject to the CFL constraints of the external mode gravity wave speed, which is roughly $\sqrt{gH} = 200 - 250\text{m}\,\text{s}^{-1}$ for ocean depths $H = 4000\text{m} - 6000\text{m}$. When splitting, the internal modes, which are roughly 100 times slower than the external mode, can be integrated with approximatly 100 times longer time steps, thus enhancing the utility of the model for climate simulations.

2. As vertical resolution is improved, the computation requirements for the barotropic mode will remain the same. However, for a non-separated model, adding vertical resolution adds more equations which are subject to the barotropic mode time step. Modern ocean simulations are tending towards increasing the vertical resolution in order to improve the representation of vertical physical processes such as boundary layers. Therefore, the low efficiency of the non-separated model is a greater burden for these high vertical resolution models.

There are two fundamental methods in MOM for solving the momentum equations. The traditional rigid lid method completely filters out the very fast waves associated with the external mode by fixing the ocean surface to be flat. This filtering transforms the generally hyperbolic external mode problem to an elliptic problem. The free surface, in contrast, admits the fast external waves and so care must be exercised in order to maintain numerical stability, and additional care must be exercised due to the possible linear interaction between the depth independent and depth dependent modes. It turns out that each method, and certain variants thereof, imply far reaching consequences for the numerical methods and physical content of the whole model. Much of the discussion in the remainder of this chapter elaborates on these consequences. The remainder of this section provides a general overview of these two methods, and later sections and chapters provide the full details.

5.2 Methods for solving the separated equations

In symbols, the horizontal velocity $\mathbf{u}_h$ is separated into two parts. The vertically averaged velocity representing the *approximate* barotropic or external part is given by

$$\overline{\mathbf{u}}_h = \frac{1}{H+\eta} \int_{-H}^{\eta} dz\, \mathbf{u}_h, \tag{5.1}$$

where $H(\lambda, \phi)$ is the distance from the resting ocean surface $z = 0$ to the bottom, and $\eta(\lambda, \phi, t)$ is the departure of the ocean surface height from $z = 0$. Typically, $|\eta| \leq 200\text{cm}$, but may be much larger, if tides are taken into consideration. In general, fields which are averaged over the vertical coordinate will be denoted with the overbar. The residual

$$\widehat{\mathbf{u}}_h = \mathbf{u}_h - \overline{\mathbf{u}}_h \tag{5.2}$$

is a depth dependent velocity, which embodies the *approximate* baroclinic or internal mode flow. Often, it will be convenient to introduce the vertically integrated horizontal velocity field

$$\mathbf{U} = (H + \eta)\,\overline{\mathbf{u}}_h = \int_{-H}^{\eta} dz\, \mathbf{u}_h. \tag{5.3}$$

Additionally, the following vertically integrated velocity

$$\mathbf{U}_0 = \int_{-H}^{0} dz\, \mathbf{u}_h \tag{5.4}$$

will prove useful. In the fixed surface / rigid lid method (see below), there is no distinction between $\mathbf{U}$ and $\mathbf{U}_0$ in the baroclinic model part, since $\eta = 0$ is assumed. Additionally, with $\eta = 0$ and $w(z = 0) = 0$, then $\nabla_h \cdot \mathbf{U} = 0$. This result is exploited in the rigid lid formulation, as seen in Section 5.2.1.

The dynamical equations for the vertically averaged velocity are generally more complicated than the unaveraged equations. Two means for handling these equations are implemented in MOM:

- The *fixed surface / rigid lid method*. This method fixes the upper surface to $\eta = 0$, and closes the upper boundary with $w(z = 0) = 0$. There are two flavors of this method: the streamfunction and the surface pressure methods.

- The *free surface / non-rigid lid method*. This method allows for a freely evolving surface $\eta \neq 0$, and it uses open boundary conditions at $z = 0$ with $w(0) \neq 0$ for the baroclinic and tracer equations. There are two flavors of this method: the explicit and implicit free surface methods.

In short, these two methods differ fundamentally in how they handle the upper ocean boundary conditions.

5.2.1 The fixed surface / rigid lid method in brief

The basics of the *rigid lid method* are summarized in this section. More complete details for the rigid lid streamfunction method are given in Chapter 6.

5.2.1.1 Fixed surface height

The key assumption with the rigid lid is that the ocean surface height is fixed

$$\eta \equiv 0. \tag{5.5}$$

This assumption eliminates the wave modes associated with vertical displacements of the full water column. These are the modes associated with the fast external mode gravity waves. Therefore, fixing the surface height eliminates the fast waves and consequently allows for relatively large momentum time steps. On its own, the removal of the fast modes is thought to be of minimal consequence for global ocean simulations. It does preclude the study of phenomena associated with barotropic waves, such as the barotropic tides.

5.2.1.2 Vanishing velocity at the ocean surface

A related assumption made by Bryan (1969) is to set the vertical velocity at the surface to zero

$$w(z = 0) = 0. \tag{5.6}$$

Setting w to zero at the ocean surface is *not* necessary for eliminating the external mode gravity waves. Again, $\eta = 0$ is sufficient. As discussed in Section 6.1, Bryan's choice to set $w(z = 0) = 0$ is based on the consequent ability of a single streamfunction to specify the two components of the barotropic velocity. If w is allowed to fluctuate at the surface, and $\eta = 0$ is still imposed, then a velocity potential must be used in addition to the streamfunction (see Section 7.6). Both the streamfunction and velocity potential have separate elliptical problems which must be solved. In large scale ocean models, especially those with realistic geography and topography, the elliptic problems are quite expensive computationally. Therefore, Bryan's choice to set $w(z = 0) = 0$ removed a large amount of computational burden from the model.

5.2.1.3 Fresh water forcing in the rigid lid

A constant surface height, $\eta = 0$, does not imply a vanishing vertical velocity component at the ocean surface. The basic reason is that the surface height is a Lagrangian coordinate and the vertical velocity at the surface is Eulerian. Relatedly, as seen by the equation (4.29) for the surface kinematic boundary condition, $\eta = 0$ does not imply $w(z = 0) = 0$ *if* there is a surface fresh water flux. This is the fundamental point raised in the work of Huang (1993). As can be seen through the kinematic boundary condition, allowing w to fluctuate at the surface, while still keeping $\eta = 0$, will enable a more physical means to force a rigid lid model with fresh water $q_w \neq 0$ while at the same time filtering out the external mode gravity waves. Setting both $\eta = 0$ and $w(z = 0) = 0$ precludes a direct use of fresh water forcing. As such, the upper boundary is effectively *closed* to fresh water in the traditional rigid lid method. Instead, the effects of fresh water must be introduced through a *virtual salt flux* added to the salinity equation. The implementation of other surface boundary conditions likewise may involve certain unphysical assumptions.

In summary, the tradeoff that Bryan (1969) made was to eliminate the need for computing a velocity potential while sacrificing a physically based fresh water forcing. Huang (1993) argues that this choice is not satisfactory for large scale modeling since it eliminates some fundamental modes of the ocean circulation. Additionally, it possibly affects the sensitivity of the oceanic variability which might be realized in a coupled ocean-atmosphere model. Currently, the choice made in the development of MOM is to not implement the method from Huang. Rather, it is to focus on the free surface method for the reasons explained below.

5.2.1.4 Two rigid lid methods in MOM

Since the method of Huang (1993) has not been implemented, in this manual the rigid lid will imply the two assumptions $\eta = 0$ and $w(z = 0) = 0$. In MOM, there are two methods which employ the rigid lid approximation. The first is the original Bryan (1969) streamfunction method (option *stream_function*). The second method is that from Smith, Dukowicz, and Malone (1992) and Dukowicz, Smith, and Malone (1993). Instead of solving for a streamfunction, they solved for the surface pressure. Hence, the option is called *rigid_lid_surface_pressure* (Section 6.8). Both of these rigid lid methods have been unchanged since MOM 2.

5.2.2 The free surface / non-rigid lid method in brief

In the last few years, modelers have been motivated for various reasons to jettison the fixed surface / rigid lid approximation. Instead, they have decided to formulate the free surface problem in various manners. Briefly, these reasons are the following:

1. A free surface method treats the sea level elevation as a prognostic variable and allows for a representation of the spreading of surface gravity waves and/or Kelvin waves.

2. A free surface allows for a more accurate representation of surface tracer fluxes, such as providing for a real fresh water flux (see Section 5.2.1.3).

3. A free surface eliminates the Killworth (1987) topographic instability.

4. A free surface allows for more ready code parallelization, whereas the island integrals in the rigid lid streamfunction approach are more cumbersome on shared memory architectures.

Section 29.1 provides a more a detailed discussion of these points.

5.2.2.1 The barotropic equation and its two solution methods

Integrating the continuity equation vertically yields a prognostic equation for the sea surface elevation η, together with the vertically integrated momentum equations. These equations describe the fast barotropic gravity waves. In general, the numerical solution scheme for these equations requires a very small time step if these waves are resolved. There are two methods used in MOM: the option *implicit_free_surface* of Dukowicz and Smith (1994), and the option *explicit_free_surface*. The explicit free surface can be time stepped either by the methods of Killworth, Stainforth, Webb and Paterson (1991), or by the methods discussed in Section 29.5 and in Griffies, Pacanowski, Schmidt, and Balaji (2000). The implicit method as coded in MOM has not been found to be unconditionally stable, although in principle it should be so. In general, if the time step is too large, the barotropic gravity waves are damped out with the implicit scheme. If this damping is not desired, then the explicit method can be used. In this approach, the barotropic mode is integrated with a small time step. While integrating the barotropic equations, the baroclinic flow as well as tracer fields are kept constant. This approximation is justified since the time scale of baroclinic and tracer processes is much larger than the barotropic process. Note that if the time step is smaller than required by the CFL-criterion for barotropic gravity waves, both the implicit and explicit methods have been found to yield similar results.

5.2.2.2 The non-rigid lid approximation

An ocean model with a free surface strictly has a time dependent domain with a top model grid box possessing a variable upper surface. This added degree of freedom increases the complexity of the numerical scheme for the tracers and the baroclinic mode due to the need to take into account the variable top model grid box. Instead of introducing this complexity, those using a free surface in z-level models have instead considered a model with fixed vertical levels. The upper kinematic, dynamic, and tracer boundary conditions are formulated as open boundary conditions applied at $z = 0$ rather than at $z = \eta$. The application of the upper boundary at $z = 0$ is similar to the rigid lid approach. However, for the non-rigid lid free

surface, the upper boundary is *open* or permeable, whereas the upper boundary in the rigid lid method is closed or impermeable. Most notably, the vertical velocity $w(z = 0)$ does not vanish with the free surface. This comparison motivates the name *non-rigid lid method* for the handling of surface boundary conditions in the free surface method.

An approach such as this was employed by Killworth *et al.* (1989) and is also used by Dukowicz and Smith (1994). In general, the non-rigid lid approximation is well justified so long as the top model grid box between $z_1 \leq z \leq 0$ is much thicker than the maximum free surface height η. In shallow seas or in models with very fine vertical grid spacings, this assumption is not valid. More crucially, such models do not conserve total tracer or momentum. It is for this reason that MOM has recently (Summer 1999) implemented a full free surface method in which the effects of the undulating surface height has been incorporated into the depth dependent fields. This method is fully documented in Griffies, Pacanowski, Schmidt, and Balaji (2000).

Chapter 6

Rigid lid streamfunction method

The rigid lid streamfunction method was developed by Bryan (1969). This chapter presents the formulation of the rigid lid in a fashion to closely parallel the derivation given for the free surface in Chapter 7.

6.1 The barotropic streamfunction

With a rigid lid at the ocean surface, the surface height $\eta = 0$. Setting η to zero eliminates the very fast (order $200\,\mathrm{m\,s^{-1}}$ in water with depth $4000 - 5000m$) external mode gravity waves. As a result of making the rigid lid assumption, the vertically integrated horizontal velocity satisfies

$$\begin{aligned}
\nabla_h \cdot \mathbf{U} &= \mathbf{u}_h(-H) \cdot \nabla_h H + \int_{-H}^{0} dz\, \nabla_h \cdot \mathbf{u}_h \\
&= \mathbf{u}_h(-H) \cdot \nabla_h H - \int_{-H}^{0} dz\, w_z \\
&= \mathbf{u}_h(-H) \cdot \nabla_h H + w(-H) - w(0) \\
&= -w(0). \qquad (6.1)
\end{aligned}$$

To reach this result, Leibnitz's Rule (4.65), the continuity equation (4.3), and the bottom kinematic boundary condition (4.24) were used. Taking the additional assumption

$$w(z = 0) = 0 \qquad (6.2)$$

renders

$$\nabla_h \cdot \mathbf{U} = \nabla_h \cdot H\,(\overline{u}, \overline{v}) = 0. \qquad (6.3)$$

As a result, the external mode velocity can be expressed in terms of the *external mode streamfunction*

$$\overline{u} = -\left(\frac{1}{Ha}\right) \psi_\phi \qquad (6.4)$$

$$\overline{v} = \left(\frac{1}{Ha \cdot \cos\phi}\right) \psi_\lambda. \qquad (6.5)$$

As a vector equation, this relation takes the form

$$\mathbf{U} = H\,(\overline{u}, \overline{v}) = \hat{z} \wedge \nabla_h \psi. \qquad (6.6)$$

6.2 Streamfunction and volume transport

The barotropic streamfunction is specified only to within a constant. As such, only differences are physically relevant. In particular, consider the vertically integrated advective transport between two points

$$T_{ab} = \int_a^b dl\ \hat{n} \cdot \int_{-H}^0 dz\ \mathbf{u}_h, \tag{6.7}$$

where dl is the line element along any path connecting the points a and b, and $\hat{n}$ is a unit vector pointing perpendicular to the path in a rightward direction when facing the direction of integration. As written, T_{ab} has units of $volume/time$, and so it represents a volume transport. The definition of the barotropic streamfunction and Stokes' Theorem renders

$$\begin{aligned} T_{ab} &= \int_a^b dl\ \hat{n} \cdot \mathbf{U} \\ &= \int_a^b dl\ \hat{n} \cdot \hat{z} \wedge \nabla_h \psi \\ &= -\int_a^b dl\ \nabla_h \psi \cdot (\hat{z} \wedge \hat{n}) \\ &= -\int_a^b dl\ \nabla_h \psi \cdot \hat{t} \\ &= \psi_a - \psi_b, \end{aligned} \tag{6.8}$$

where

$$\hat{t} = \hat{z} \wedge \hat{n} \tag{6.9}$$

is a unit vector tangent to the integration path, pointing in the direction of integration from point a to point b. Therefore, the difference between the barotropic streamfunction at two points represents the vertically integrated volume transport between the two points. It is for this reason that the barotropic streamfunction is sometimes called the *volume transport streamfunction*. Note that Bryan (1969) defined the barotropic streamfunction with an extra factor of the Boussinesq density ρ_o, such than $\psi_{bryan} = \rho_o\ \psi_{mom}$. Hence, the barotropic streamfunction of Bryan has the dimensions of $mass/time$ rather than volume per time, and so it represents a *mass transport streamfunction*. Since MOM assumes a Boussinesq fluid, the difference is trivial.

6.3 Hydrostatic pressure with the rigid lid

The hydrostatic equation $p_z = -\rho g$ can be integrated from the surface $z = 0$ to some position $z < 0$ to yield

$$\begin{aligned} p(\lambda, \phi, z, t) &= p_a(\lambda, \phi, t) + p_l(\lambda, \phi, t) + g \int_z^0 dz\ \rho \\ &= p_a(\lambda, \phi, t) + p_l(\lambda, \phi, t) + p_b(\lambda, \phi, z, t), \end{aligned} \tag{6.10}$$

where p_a is the atmospheric pressure, p_l is the surface *lid* pressure, and p_b is the hydrostatic pressure arising from the ocean's density field. The surface lid pressure is the pressure which

would be exerted by an imaginary rigid lid placed on top of the ocean. An alternative interpretation is that it is the pressure exerted by undulations of a free surface. The latter interpretation is further disscussed in Section 7.1. In summary, the horizontal pressure gradient is given by

$$\nabla_h p = \nabla_h (p_a + p_l) + g \int_z^0 dz \, \nabla_h \rho. \tag{6.11}$$

The horizontal pressure gradient at some depth z has a contribution from gradients in the atmospheric and lid pressure, gradients which act the same for all depths, and the vertical integral of horizontal gradients in the interior density. These latter gradients are due to baroclinicity in the density field, which prompts the often used name *baroclinic pressure gradient*, and which motivates the "b" subscript. The vertically integrated horizontal pressure gradient is needed for the development of the barotropic vorticity equation. This gradient is given by

$$\int_{-H}^0 dz \, \nabla_h p \;=\; \int_{-H}^0 dz \, \nabla_h (p_a + p_l) + \int_{-H}^0 dz \, \left(g \int_z^0 dz' \, \nabla_h \rho \right). \tag{6.12}$$

Since the horizontal gradient of the atmospheric and lid pressures are independent of depth,

$$\int_{-H}^0 dz \, \nabla_h (p_a + p_l) = H \, \nabla_h (p_a + p_l), \tag{6.13}$$

which renders

$$\int_{-H}^0 dz \, \nabla_h p \;=\; H \left(\nabla_h (p_a + p_l) + \overline{\nabla_h p_b} \right). \tag{6.14}$$

6.4 The barotropic vorticity equation

For cases where the ocean model is driven by a realistic atmospheric model, the atmospheric pressure p_a may be determined. But since the lid pressure p_l is actually an artifact of making the rigid lid approximation, it is generally not available. To address this fact, Bryan formed the barotropic vorticity equation in which case the lid and atmospheric pressures are eliminated. Through inverting the vorticity equation, the streamfunction is obtained. As a result, the barotropic velocities can then be determined through the relation (6.6). As will be seen in Section 6.6, the lid and atmospheric pressures will likewise not be needed for determining the baroclinic velocity. This algorithm has been used for solving the rigid lid in MOM up until the formulation of the surface pressure approach of Smith, Dukowicz, and Malone (1992) and Dukowicz, Smith, and Malone (1993) (Section 6.8). In the present section, a derivation of the barotropic vorticity equation is provided.

6.4.1 Tendencies for the vertically averaged velocities

The first step is to integrate the horizontal velocity equations (4.1) and (4.2) vertically over the full depth of the rigid lid ocean

$$\partial_t U - f V \;=\; - \left(\frac{H}{a \, \rho_o \cos \phi} \right) (p_a + p_l)_\lambda + X_0 \tag{6.15}$$

$$\partial_t V + f U \;=\; - \left(\frac{H}{a \, \rho_o} \right) (p_a + p_l)_\phi + Y_0. \tag{6.16}$$

The vertical integral of depth dependent quantities (mod the Coriolis force) has been lumped into the terms

$$X_0 = \int_{-H}^{0} dz \left(-\nabla \cdot (u\mathbf{u}) + \frac{uv \tan\phi}{a} + (\kappa_m u_z)_z + F^u - \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} \right) \tag{6.17}$$

$$Y_0 = \int_{-H}^{0} dz \left(-\nabla \cdot (v\mathbf{u}) - \frac{u^2 \tan\phi}{a} + (\kappa_m v_z)_z + F^v - \frac{(p_b)_\phi}{a\rho_\circ} \right). \tag{6.18}$$

These equations are implemented in MOM through a straightforward vertical integration of the forcing terms.

To facilitate physical interpretation of the surface terms forcing the vertically integrated momentum, it is useful to perform some manipulations on X_0 and Y_0. First, use the bottom kinematic boundary condition (4.24) and the surface kinematic condition $w(0) = 0$ in order to bring the vertical integral of the convergence of the advective flux to the form

$$\begin{aligned} -\int_{-H}^{0} dz \left(\nabla_h \cdot (\alpha\, \mathbf{u}_h) + (\alpha\, w)_z \right) &= -\nabla_h \cdot \int_{-H}^{0} dz\, \alpha\, \mathbf{u}_h \\ &+ \alpha(-H) \Big[\nabla_h H \cdot \mathbf{u}_h(-H) + w(-H) \Big] - w(0)\, \alpha(0) \\ &= -\nabla_h \cdot \int_{-H}^{0} dz\, \alpha\, \mathbf{u}_h, \end{aligned} \tag{6.19}$$

where $\alpha = u, v$. Second, recall from equations (9.187) and (9.193) that the horizontal momentum friction can be written as a Laplacian acting on the horizontal velocity, plus an extra "metric" term

$$\mathbf{F} = \nabla_h \cdot (A_m \nabla_h \mathbf{u}) + \mathbf{F}_{metric}. \tag{6.20}$$

As such, the depth integral of momentum friction can be written

$$\begin{aligned} &\int_{-H}^{0} dz \Big((\kappa_m\, \mathbf{u}_z)_z + \mathbf{F} \Big) \\ &= \left(\frac{\tau_{surf} - \tau_{bottom}}{\rho_o} \right) + \int_{-H}^{\eta} dz\, \mathbf{F}_{metric} + \nabla_h \cdot \int_{-H}^{0} dz\, A_m \nabla_h \mathbf{u}, \end{aligned} \tag{6.21}$$

where

$$\tau_{surf} = \rho_o\, \kappa_m\, \mathbf{u}_z|_{z=0} \tag{6.22}$$

$$\tau_{bottom} = \rho_o \left(\kappa_m\, \mathbf{u}_z + A_m\, (\nabla_h H) \cdot (\nabla_h \mathbf{u}) \right)_{z=-H} \tag{6.23}$$

are the surface and bottom stresses (dyn cm^{-2}) due to the winds at the ocean surface and friction and topography at the bottom. The above results render

$$\begin{aligned} X_0 &= \frac{\Delta(\tau^\lambda)}{\rho_o} - \int_{-H}^{0} dz\, \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} \\ &+ \int_{-H}^{0} dz \left(\frac{uv \tan\phi}{a} + F^u_{metric} \right) - \nabla_h \cdot \left(\int_{-H}^{0} dz\, \left(u\, \mathbf{u}_h - A_m \nabla_h u \right) \right) \end{aligned} \tag{6.24}$$

$$\begin{aligned} X_0 &= \frac{\Delta(\tau^\phi)}{\rho_o} - \int_{-H}^{0} dz\, \frac{(p_b)_\phi}{a\rho_\circ} \\ &+ \int_{-H}^{0} dz \left(-\frac{u^2 \tan\phi}{a} + F^v_{metric} \right) - \nabla_h \cdot \left(\int_{-H}^{0} dz\, \left(v\, \mathbf{u}_h - A_m \nabla_h v \right) \right). \end{aligned} \tag{6.25}$$

Using an overline to denote a vertical column average, and using the expression (6.14) for the horizontal pressure gradient, yields the vertically *averaged* velocity equations

$$\partial_t \overline{u} = f\,\overline{v} - \left(\frac{1}{a\,\rho_o\,\cos\phi}\right)\left[(p_a + p_l)_\lambda + \overline{(p_b)_\lambda}\right] + (\Delta\tau^\lambda/\rho_o\,H) + \Gamma^u, \tag{6.26}$$

$$\partial_t \overline{v} = -f\,\overline{u} - \left(\frac{1}{a\,\rho_o}\right)\left[(p_a + p_l)_\phi + \overline{(p_b)_\phi}\right] + (\Delta\tau^\phi/\rho_o\,H) + \Gamma^v, \tag{6.27}$$

where

$$\Gamma^u = \overline{F^u_{metric}} + \overline{u\,v}\,(\tan\phi/a) + H^{-1}\,\nabla_h \cdot \left[H\,\overline{(A_m\,\nabla_h - \mathbf{u}_h)\,u}\right] \tag{6.28}$$

$$\Gamma^v = \overline{F^v_{metric}} - \overline{u\,u}\,(\tan\phi/a) + H^{-1}\,\nabla_h \cdot \left[H\,\overline{(A_m\,\nabla_h - \mathbf{u}_h)\,v}\right]. \tag{6.29}$$

represent the vertical average of the horizontal friction metric terms, advection metric terms, divergence of the horizontal viscous fluxes, and convergence of the horizontal advective fluxes.

6.4.2 The barotropic vorticity equation

In order to eliminate the lid and atmospheric pressures, it is sufficient to form the time tendency of the barotropic vorticity

$$\begin{aligned}
\overline{\zeta} &= \hat{z}\cdot\nabla\wedge\overline{\omega} \\
&= \left(\frac{1}{a\,\cos\phi}\right)[\overline{v}_\lambda - (\overline{u}\,\cos\phi)_\phi] \\
&= \frac{1}{a\,\cos\phi}\left[\frac{1}{\cos\phi}\left(\frac{\psi_\lambda}{H}\right)_\lambda + \left(\frac{\psi_\phi\,\cos\phi}{H}\right)_\phi\right] \\
&= \nabla_h\cdot\left(\frac{1}{H}\,\nabla_h\psi\right).
\end{aligned} \tag{6.30}$$

A few lines of manipulations renders

$$\begin{aligned}
&\overline{\zeta}_t + \beta\,\overline{v} = \\
&\quad - f\,\nabla_h\cdot\overline{\mathbf{u}_h} - \left(\frac{1}{a^2\,\rho_o\,\cos\phi}\right)\left[\partial_\lambda(\overline{\partial_\phi p_b}) - \partial_\phi(\overline{\partial_\lambda p_b})\right] + \hat{z}\cdot\nabla\wedge\left(\frac{\Delta\tau}{\rho_o\,H} + \Gamma^{\mathbf{u}}\right),
\end{aligned} \tag{6.31}$$

where

$$\begin{aligned}
\beta &= \frac{1}{a}\frac{\partial f}{\partial\phi} \\
&= (2\,\Omega/a)\,\cos\phi
\end{aligned} \tag{6.32}$$

is the planetary vorticity gradient. The forcing for tendencies in $\overline{\zeta}$ consists of the following terms:

1. Meridional advection of planetary vorticity: $-\beta\,\overline{v}$.

2. Curl of the Coriolis force, which takes the form of the convergence of the barotropic velocity weighted by the Coriolis parameter: $-f\,\nabla_h \cdot \overline{\mathbf{u}_h}$. For a flat bottom, this term vanishes with the rigid lid. Combined with the $\beta\,\overline{v}$ term, these provide the forcing $-\nabla_h \cdot (f\,\mathbf{u}_h)$ to the barotropic vorticity.

3. The antisymmetric term proportional to $\partial_\lambda(\overline{\partial_\phi p_b}) - \partial_\phi(\overline{\partial_\lambda p_b})$. This term vanishes in a barotropic model, or in a baroclinic model with a flat bottom.

4. Curl of the depth weighted surface minus bottom stresses: $\hat{z} \cdot \nabla \wedge (\Delta\tau/\rho_o\,H)$.

5. Curl of the nonlinear lateral friction terms and advection terms embodied by Γ.

To touch bases with familiar textbook dynamics, note that a steady state ocean with $\Gamma = 0$ and a flat bottom will result in the familiar barotropic Sverdrup balance

$$\beta\,\overline{v} = \hat{z} \cdot \nabla \wedge \left(\frac{\Delta\tau}{\rho_o\,H}\right). \tag{6.33}$$

6.4.3 Caveat: inversions with steep topography

In order to solve for the barotropic velocity $\overline{\mathbf{u}}$, it is necessary to invert the elliptical operator appearing in the relation (6.30) relating the barotropic vorticity to the barotropic streamfunction. The presence of the inverse depth in this operator implies that near regions where H changes rapidly, the elliptic operator also changes rapidly. As emphasized by Smith, Dukowicz, and Malone (1992) and Dukowicz, Smith, and Malone (1993), numerical elliptic inversions will potentially have problems converging in the presence of such regions. This form for the elliptic operator is also the central reason for the Killworth topographic instability (Killworth 1987). This instability can result in significant time step constraints in rigid lid models with non-smoothed topography, and these time step constraints can be more stringent than the typical CFL constraints. More problematic from the perspective of very long-term climate modeling, the Killworth instability can sometimes be slowly growing, and may become problematical only after some few hundreds of years. A slow growing instability is arguably more pernicious than rapidly growing instabilities, since a significant amount of computation time can be used before encountering the problem. Experience at GFDL indicates that the Killworth instability is a nontrivial problem with realistic models, and so it provides motivation to avoid the rigid lid streamfunction method in such models.

6.5 Boundary conditions and island integrals

6.5.1 Dirichlet boundary condition on the streamfunction

The no-normal flow boundary condition implies that next to lateral boundaries,

$$\begin{aligned} \hat{n} \cdot H(\overline{u}, \overline{v}) &= \hat{n} \cdot \hat{z} \wedge \nabla_h \psi \\ &= -\hat{t} \cdot \nabla_h \psi \\ &= 0, \end{aligned} \tag{6.34}$$

where $\hat{n}$ is a unit vector pointing outwards from the boundary, with the interior of the closed domain to the left, and $\hat{t} = \hat{z} \wedge \hat{n}$ is a unit vector parallel to the path traversing the boundary. This constraint says that the streamfunction is a constant along the boundaries. In the parlance

of applied mathematics, such boundary conditions are known as *Dirichlet* conditions. In general, the streamfunction can be a different time dependent constant along the different closed boundaries

$$\psi = \mu_r(t), \tag{6.35}$$

where $\mu_r(t)$ is a time dependent number, and $r = 1, 2, \ldots R$ labels the particular boundary (an island label), with R the total number of islands. The interpretation afforded by Stokes' Theorem, discussed in Section 6.1, indicates that $\mu_r(t)$ represents the time dependent volume transport circulating around the island with label r.

The presence of Dirichlet boundary conditions on the streamfunction indicates that the streamfunction at one point along the boundary is identical to the streamfunction at another point, which can generally be thousands of kilometres away. Such non-local forms of information can be quite problematical in the solution of the streamfunction on parallel machines. The discussions in Smith, Dukowicz, and Malone (1992), Dukowicz, Smith, and Malone (1993), and Dukowicz and Smith (1994) highlight this point.

6.5.2 Separating the streamfunction's boundary value problem

The purpose of this section is to develop an algorithm for solving the streamfunction's boundary value problem (BVP). To do so, reconsider the horizontal momentum equations written in the form

$$\mathbf{u}_t = -\nabla_h(p/\rho_o) + \mathbf{G}, \tag{6.36}$$

where the vector $\mathbf{G}$ represents all the remaining terms given in equations (4.1) and (4.2). Taking the vertical average of this equation, substituting the definition (6.6) for the barotropic streamfunction, and using the expression (6.14) for the hydrostatic pressure gradient, yields

$$\frac{1}{H}\hat{z} \wedge \nabla_h \psi_t = -\frac{1}{\rho_o}\left(\nabla_h(p_a + p_l) + \overline{\nabla_h p_b}\right) + \overline{\mathbf{G}}. \tag{6.37}$$

Taking the curl of this equation eliminates the atmospheric and lid pressures

$$\nabla_h \wedge \left(\frac{1}{H}\hat{z} \wedge \nabla_h \psi_t\right) = -\frac{1}{\rho_o}\nabla_h \wedge \left(\overline{\nabla_h p_b} - \overline{\mathbf{G}}\right). \tag{6.38}$$

The elliptic boundary value problem

$$\nabla_h \wedge \left(\frac{1}{H}\hat{z} \wedge \nabla_h \psi_t\right) = -\frac{1}{\rho_o}\nabla_h \wedge \left(\overline{\nabla_h p_b} - \overline{\mathbf{G}}\right) \qquad \text{interior points} \tag{6.39}$$

$$\psi = \mu_r(t) \qquad \text{on island } r, \tag{6.40}$$

can be separated into two simpler BVPs. The first one is a forced elliptical problem with homogeneous boundary conditions

$$\nabla_h \wedge \left(\frac{1}{H}\hat{z} \wedge \nabla_h (\psi_o)_t\right) = -\frac{1}{\rho_o}\nabla_h \wedge \left(\overline{\nabla_h p_b} - \overline{\mathbf{G}}\right) \qquad \text{interior points} \tag{6.41}$$

$$\psi_o = 0 \qquad \text{on islands.} \tag{6.42}$$

This system can be solved for ψ_o using some time-stepping scheme, such as leap-frog. The second BVP is a time-independent unforced elliptical problem with constant boundary conditions on the islands

$$\nabla_h \wedge \left(\frac{1}{H} \hat{z} \wedge \nabla_h \psi_r \right) = 0 \quad \text{everywhere, except island boundaries} \tag{6.43}$$

$$\psi_r = 1 \quad \text{on islands.} \tag{6.44}$$

This BVP can be solved for ψ_r using some type of an elliptical solver, such as congugate gradient. The full streamfunction is built from the sum

$$\psi(\lambda, \phi, t) = \psi_o(\lambda, \phi, t) + \sum_{r=1}^{R} \mu_r(t)\, \psi_r(\lambda, \phi), \tag{6.45}$$

which can be shown to satisfy the original boundary value problem.

6.5.3 Island integrals for the volume transport

As a final step in the streamfunction solution, it is necessary to formulate a prognostic equation for the volume transports $\mu_r(t)$. To do so, reconsider the elliptical problem for the streamfunction

$$\nabla_h \wedge \left(\frac{1}{H} \hat{z} \wedge \nabla_h \psi_t \right) = -\frac{1}{\rho_o} \nabla_h \wedge \left(\overline{\nabla_h p_b} - \overline{\mathbf{G}} \right). \tag{6.46}$$

Now integrate both sides over the area bounding a particular island with label r, and employ Stokes' Theorem. The direction normal to the island's surface is $\hat{z}$, and the area element on the island is $d\Omega_r$. The left hand side becomes

$$\begin{aligned} \int\int \hat{z}\, d\Omega_r\, \nabla_h \wedge \left(\frac{1}{H} \hat{z} \wedge \nabla_h \psi_t \right) &= \int\int \hat{z}\, d\Omega_r\, \nabla_h \wedge \left(\frac{1}{H} \hat{z} \wedge \nabla_h \psi_r \right) \dot{\mu}_r \\ &= \dot{\mu}_r \oint dl\, \hat{t} \cdot \frac{1}{H} \hat{z} \wedge \nabla_h \psi_r \\ &= -\dot{\mu}_r \oint dl\, \hat{n} \cdot \frac{1}{H} \nabla_h \psi_r, \end{aligned} \tag{6.47}$$

where $\hat{t} \wedge \hat{z} = -\hat{n}$ was used, and $\hat{n}$ represents the outward normal to the island boundary. The right hand side becomes

$$-\int\int \hat{z}\, d\Omega_r\, \nabla_h \wedge \frac{1}{\rho_o} \nabla_h \wedge \left(\overline{\nabla_h p_b} - \overline{\mathbf{G}} \right) = -\frac{1}{\rho_o} \oint dl\, \hat{t} \cdot (\overline{\nabla_h p_b} - \overline{\mathbf{G}}). \tag{6.48}$$

Equating yields the prognostic equations for the volume transports around an island

$$\dot{\mu}_r = \frac{1}{\rho_o} \frac{\oint dl\, \hat{t} \cdot (\overline{\nabla_h p_b} - \overline{\mathbf{G}})}{\oint dl\, \hat{n} \cdot \frac{1}{H} \nabla_h \psi_r}. \tag{6.49}$$

6.6 The baroclinic mode

The time tendency for the zonal baroclinic velocity is given by

$$\begin{aligned} \partial_t \widehat{u} &= \partial_t (u - \overline{u}) \\ &= \left(1 - \frac{1}{H} \int_{-H}^{0} dz \right) \left(G^u - \left(\frac{1}{a\, \rho_\circ \cos\phi} \right) (p_b + p_l + p_a)_\lambda \right) \end{aligned} \tag{6.50}$$

where the vector **G** was introduced in equation (6.36). Since the atmospheric and lid pressures are depth independent, these pressures have zero deviation from their depth average. Hence, the tendency for the baroclinic velocity is independent of the atmospheric and lid pressures. In determining the baroclinic velocity, it is therefore sufficient to consider

$$\partial_t \widehat{u} \quad = \quad \partial_t (u' - \overline{u}') \tag{6.51}$$

where u' and $\overline{u}'$ satisfy the full zonal velocity equation and vertically averaged zonal velocity equation, respectively, but without any atmospheric or lid pressure contributing to the forcing. Without the atmospheric and lid pressure contributions, all forcing terms in the two "primed" equations are known. Consequently, the primed velocities can be time stepped in a straightforward manner without using any tricks such as those needed to eliminate the atmospheric and lid pressures from the streamfunction equation. Time stepping the primed velocities then yields the updated baroclinic velocity through equation (6.51). After obtaining the updated baroclinic velocity, the full horizontal velocity field

$$(u, v) = (\overline{u}, \overline{v}) + (\widehat{u}, \widehat{v}) \tag{6.52}$$

is known at the new time step. Formulation of the rigid lid streamfunction method is now complete.

6.7 Summary of the rigid lid streamfunction method

In summary, the rigid lid streamfunction method allows for the computation of the total velocity field using the following algorithm:

- Baroclinic mode
    1. Compute the forcing terms for the "primed" momentum equation, which include all terms except the lid pressure and atmospheric pressure (Section 6.6).
    2. Solve the primed momentum equation at each model level to get the primed velocity at the next time step.
    3. Subtract the vertically averaged primed momentum from the primed momentum in order to get the baroclinic velocity field at the next time step.
- Barotropic mode
    1. Compute the forcing terms in the barotropic vorticity equation (Section 6.4).
    2. Solve the elliptical boundary value problem for the time tendency of the barotropic streamfunction.
    3. Time step the streamfunction forward and compute the barotropic velocity field at the new time step.
- Add the baroclinic velocity to the barotropic velocity to then have the full velocity field.

6.8 Rigid lid surface pressure method

The rigid lid surface pressure method in MOM was developed by Smith, Dukowicz, and Malone (1992) and Dukowicz, Smith, and Malone (1993). A useful discussion of the analytical issues related to the surface pressure formulation can be found in the paper by Pinardi, Rosati, and Pacanowski (1995). The basic advantages over the streamfunction approach are the following:

1. The elliptical operator has an H factor, rather than the H^{-1} found in the streamfunction approach. This property makes the elliptical operator more readily inverted than with the streamfunction approach (see Section 6.4.3). In particular, steep topography is much less of an issue with the surface pressure approach.

2. The boundary conditions for the elliptical problem are Neumann rather than Dirichlet. As discussed by Smith, Dukowicz, and Malone (1992) and Dukowicz, Smith, and Malone (1993), Neumann boundary conditions are much easier to handle on parallel machines than Dirichlet conditions.

As the formulation of the rigid lid surface pressure method follows quite closely the logic of the streamfunction approach, the details are not reproduced here. Rather, reference should be made to the Smith, Dukowicz, and Malone (1992), Dukowicz, Smith, and Malone (1993) and Pinardi, Rosati, and Pacanowski (1995) papers.

Chapter 7

Free surface method

This chapter presents the continuum formulation of the free surface momentum equations. Issues related to the tracer equations are described in Chapter 8. The discrete formulation is presented in Chapter 29.

Much of the foundations for the free surface approach in MOM were established by Blumberg and Mellor (1987), Killworth *et al.* (1991) and Dukowicz and Smith (1994). From the perspective of global climate modeling, it is arguable that the most significant change from MOM's rigid lid is the natural ability to directly incorporate fresh water into the free surface ocean model.

Recent MOM development has focused on the explicit free surface. In particular, a new time stepping scheme for the explicit free surface has been introduced. This new scheme is more suitable for large scale modeling than that of Killworth *et al.* (1991) due to its enhanced stability. Additionally, the fresh water flux terms have been incorporated into the equation for the surface height *only* for the explicit free surface. As of Summer 1999, to provide a conservative model, the effects of an undulating surface height have been incorporated into the baroclinic and tracer equations. MOM's free surface is completely documented in the paper Griffies, Pacanowski, Schmidt, and Balaji (2000).

7.1 Hydrostatic pressure with the free surface

To separate the fast barotropic gravity waves from the much slower baroclinic fluctuations, it is necessary to split the hydrostatic pressure into three terms. To start, integrate the hydrostatic equation (4.4) from the atmosphere-ocean interface at $z = \eta(\lambda, \phi, t)$ to some ocean depth z

$$p(\lambda, \phi, z, t) = p_a(\lambda, \phi, t) + g \int_z^{\eta(\lambda,\phi,t)} \rho(\lambda, \phi, z', t)\, dz' \tag{7.1}$$

In this expression, η is the free surface displacement with respect to a resting ocean (η can be positive or negative), p_a is the atmospheric pressure at the surface of the ocean, and ρ is the *in situ* ocean density. Note that p_a and η are functions only of the horizontal position (λ, ϕ) and time. The use of Leibnitz's Rule (equation (4.65)) leads to the expression for the horizontal pressure gradient

$$\nabla_h p \;=\; \nabla_h p_a + g\, \rho(z = \eta)\, \nabla_h \eta + g \int_z^{\eta} \nabla_h \rho\, dz'. \tag{7.2}$$

This gradient is needed in the horizontal momentum equations. In this expression, the first term arises from horizontal gradients of the atmospheric pressure, the second term from horizontal gradients of the free surface height, and the third term from the vertically integrated baroclinicity in the ocean column above the depth z. It is the gradient of the free surface height which is responsible for fast gravity waves, and so it must be separated from the baroclinic model part.

Now consider the integral

$$\int_z^\eta \nabla_h \rho \, dz' = \int_0^\eta \nabla_h \rho \, dz' + \int_z^0 \nabla_h \rho \, dz'. \tag{7.3}$$

The first integral is over the very small vertical distance from the resting ocean height $z = 0$ to the free surface height $z = \eta$, with the free surface height $|\eta| \approx 100 - 200\,\text{cm}$ at the most. In this region, the ocean has very small vertical density gradients due to the strong mixing effects from interactions with the atmosphere. Therefore, it is quite accurate to assume that

$$\rho(z = \eta) \approx \rho(z = 0). \tag{7.4}$$

This approximation leads to the expression

$$\begin{aligned} g \int_z^\eta \nabla_h \rho \, dz' &\approx g \, \eta \, \nabla_h \rho(z = 0) + g \int_z^0 \nabla_h \rho \, dz' \\ &= g \, \eta \, \nabla_h \rho(0) + \nabla_h p_b, \end{aligned} \tag{7.5}$$

where

$$p_b(\lambda, \phi, z, t) = g \int_z^0 \rho(\lambda, \phi, z', t) \, dz' \tag{7.6}$$

defines the hydrostatic pressure field associated with density in the vertical column between z and a resting ocean surface $z = 0$. In turn, the horizontal gradient of this field, $\nabla_h p_b = g \int_z^0 \nabla_h \rho \, dz'$, arises from baroclinic effects in that part of the ocean between the resting ocean surface and the depth z. It is for this reason that p_b is often termed the *baroclinic pressure field*. Note that the full depth integral of p_b does not vanish, as may mistakenly be construed by the adjective "baroclinic."

As a result of the well mixed assumption, the horizontal pressure gradient can be written

$$\begin{aligned} \nabla_h p &= \nabla_h (p_a + p_b) + g \, \nabla_h [\eta \, \rho(z = 0)] \\ &= \nabla_h (p_a + p_b + p_s). \end{aligned} \tag{7.7}$$

In this expression, the surface pressure

$$p_s(\lambda, \phi, t) = g \, \rho(z = 0) \, \eta, \tag{7.8}$$

was introduced. This is the hydrostatic pressure head associated with the surface height, where again is it assumed that the density field is well mixed between $z = 0$ and $z = \eta$. The total hydrostatic pressure field has been written

$$p = p_a + p_b + p_s. \tag{7.9}$$

For $z < 0$ the depth dependent baroclinic pressure is separated from the depth independent atmospheric and surface pressures. Such a separation is important for the methods described

below for integrating the momentum equations. Note that for $z > 0$ the baroclinic pressure p_b cancels partly the surface pressure p_s. Thus, for $z = \eta$ the boundary condition $p(z = \eta) = p_a$ is correctly recovered.

It is interesting to make a connection between the prognostic surface pressure used in the free surface method to the diagnostic lid pressure associated with the rigid lid method. To do so, recall that the pressure at an arbitrary depth in the rigid lid method (Section 6.3) can be written

$$p_{rl} = p_a + p_l + p_b, \tag{7.10}$$

where the atmospheric and baroclinic pressures are identical to those used in the free surface. Hence, the lid pressure p_l can be identified with the surface pressure. Indeed, if the rigid lid is allowed to freely deform, the ocean surface would then take the shape given by η.

In practice, the gradient of the surface pressure $\nabla_h [\eta \, \rho(z = 0)]$ is often approximated (e.g., Killworth *et al.*, Dukowicz and Smith) by

$$\nabla_h [\eta \, \rho(z = 0)] \approx \rho_o \, \nabla_h \, \eta, \tag{7.11}$$

where the space-time dependent surface density is approximated as a constant

$$\rho(\lambda, \phi, z = 0, t) \approx \rho_o = 1.035 \, \mathrm{g \, cm^{-3}}. \tag{7.12}$$

For convenience, this constant is also the constant value for the Boussinesq density used in MOM. The approximation (7.12) is valid so long as $\eta \, \Delta\rho / (\rho \, \Delta\eta) << 1$, where $\Delta\rho$ and $\Delta\eta$ represent typical horizontal variations. This assumption is true in the ocean over the horizontal scales of a model grid box (order 100km). Note that the global averaged annual mean surface density from Levitus (1982) is $1.024 \, \mathrm{g \, cm^{-3}}$, which is close to the chosen ρ_o. However, setting $\rho(z = 0) = 1.035 \, \mathrm{g \, cm^{-3}}$ may yield an unacceptably large systematic error for regions where the average density differs drastically from $1.035 \, \mathrm{g \, cm^{-3}}$. In such cases, it may be more appropriate to use a different value than ρ_o, or it might be best to avoid the approximation (7.12). Since Summer 1999, the standard explicit free surface in MOM does *not* use this approximation.

Now that the pressure field has been suitably split into the surface and atmospheric pressure $p_s + p_a$, whose fluctuations are associated with fast barotropic processes, and the baroclinic pressure p_b, whose fluctuations are associated with the slower baroclinic processes, it is appropriate to consider the formulation of the barotropic and baroclinic systems. This formulation provides the focus for the next few sections.

7.2 The barotropic system

The purpose of this section is to derive the equations describing the barotropic motion. These equations result from vertically integrating the momentum and continuity equations.

7.2.1 Vertically integrated transport

The vertically integrated horizontal velocity field, also called the vertically integrated transport, is defined by

$$\mathbf{U} \;=\; \int_{-H}^{\eta} dz \, \mathbf{u}_h. \tag{7.13}$$

In the following, prognostic equations for **U** will be derived. From the updated value of **U**, the updated value for the vertically averaged velocity field

$$\overline{\mathbf{u}}_h = \frac{1}{H+\eta} \int_{-H}^{\eta} dz\, \mathbf{u}_h \tag{7.14}$$

can be diagnosed for use in constructing the full velocity field in combination with the baroclinic field.

7.2.2 Bottom and surface kinematic boundary conditions

The bottom of the ocean is a material surface. The corresponding kinematic boundary condition, as derived in Section 4.3.1, is given by

$$w = -\mathbf{u}_h \cdot \nabla_h H \qquad z = -H(\lambda, \phi). \tag{7.15}$$

The details of how MOM realizes this boundary condition on the discrete grid are given in Section 22.3.3.

The ocean surface is generally permeable to fresh water fluxes. The surface kinematic boundary condition, as derived in Section 4.3.2, is given by

$$(\partial_t + \mathbf{u}_h \cdot \nabla_h)\, \eta = w + q_w \qquad z = \eta(\lambda, \phi, t). \tag{7.16}$$

where q_w is the flux of fresh water, in units of velocity (volume per unit time per unit area), crossing the ocean surface. The presence of horizontal advection of the free surface height makes this equation nonlinear.

7.2.3 Free surface height equation

Knowledge of the surface currents and fresh water flux q_w allow one to time step the free surface height through use of the surface kinematic boundary condition (7.16). However, because the motion of the free surface height is associated with fast barotropic motions, it is more useful algorithmically to determine η within the barotropic system. Additionally, a direct discretization of the surface kinematic boundary condition (7.16) would require a discretization of the advective term, which is inconvenient at best.

Instead of directly discretizing the kinematic boundary condition, perform a vertical integral of the continuity equation over the full depth of the ocean to find

$$\begin{aligned} w(\eta) - w(-H) &= -\int_{-H}^{\eta} dz\, \nabla_h \cdot \mathbf{u}_h \\ &= -\nabla_h \cdot \mathbf{U} + \mathbf{u}(\eta) \cdot \nabla_h \eta + \mathbf{u}(-H) \cdot \nabla_h H. \end{aligned} \tag{7.17}$$

Use of the bottom and surface kinematic boundary conditions (7.15) and (7.16) yields

$$\eta_t = -\nabla_h \cdot \mathbf{U} + q_w. \tag{7.18}$$

This is a fundamental balance with the free surface. In words, the time tendency for the free surface height is determined by the convergence of the vertically integrated transport plus the fresh water flux through the sea surface. Note that no extra boundary conditions for η are needed to solve this equation. Namely, the surface and bottom kinematic boundary conditions are implicitly fulfilled, and the lateral boundary conditions come from the boundary condition for the vertically integrated transport, which vanishes at the side walls. Note that in the rigid lid approximation, *each* of the three terms in this balance is *individually* set to zero.

7.2.4 Vertically integrated momentum equations

From the definition of $\mathbf{U}$ and Leibnitz's Rule (4.65), the time tendency $\mathbf{U}_t$ takes the form

$$\mathbf{U}_t = \eta_t \, \mathbf{u}_h(\eta) + \int_{-H}^{\eta} dz \, \partial_t \mathbf{u}_h. \tag{7.19}$$

In this expression, $\mathbf{u}_h(\eta)$ is the horizontal current on the ocean side of the free surface $z = \eta$, and the term $\eta_t \, \mathbf{u}_h(\eta)$ arises from changes in the free surface height. The following discussion amounts to an insertion of the terms from the momentum equations into $\int_{-H}^{\eta} dz \, \partial_t \mathbf{u}_h$.

Recall the considerations of Section 7.1, which rendered the decomposition (7.9) of hydrostatic pressure into a depth dependent "baroclinic pressure" p_b and a depth independent "barotropic pressure" $p_a + p_s$. As a result, the horizontal momentum equations take the form

$$u_t = fv - \frac{(p_a + p_s + p_b)_\lambda}{a \, \rho_\circ \cos\phi} - \nabla \cdot (u\mathbf{u}) + \frac{uv \tan\phi}{a} + (\kappa_m u_z)_z + F^u \tag{7.20}$$

$$v_t = -fu - \frac{(p_a + p_s + p_b)_\phi}{a \, \rho_\circ} - \nabla \cdot (v\mathbf{u}) - \frac{u^2 \tan\phi}{a} + (\kappa_m v_z)_z + F^v. \tag{7.21}$$

A vertical integral of these equations between $-H \le z \le \eta$ leads to the equations for the vertically integrated velocity

$$U_t - fV = \eta_t \, u(\eta) - \left(\frac{H + \eta}{a \, \rho_\circ \cos\phi} \right) (p_a + p_s)_\lambda + X \tag{7.22}$$

$$V_t + fU = \eta_t \, v(\eta) - \left(\frac{H + \eta}{a \, \rho_\circ} \right) (p_a + p_s)_\phi + Y. \tag{7.23}$$

The forcing terms

$$X = \int_{-H}^{\eta} dz \left(-\nabla \cdot (u\mathbf{u}) + \frac{uv \tan\phi}{a} + (\kappa_m u_z)_z - \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} + F^u \right) \tag{7.24}$$

$$Y = \int_{-H}^{\eta} dz \left(-\nabla \cdot (v\mathbf{u}) - \frac{u^2 \tan\phi}{a} + (\kappa_m v_z)_z - \frac{(p_b)_\phi}{a\rho_\circ} + F^v \right) \tag{7.25}$$

contain vertically integrated baroclinic-baroclinic, barotropic-barotropic, and baroclinic-barotropic interactions as well as the vertically integrated baroclinic pressure gradient and friction. Note that the nonlinear term $\eta_t \, \mathbf{u}_h(\eta)$ follows from equation (7.19); it arises from the time dependence of the free surface η. Since the surface and atmospheric pressures are independent of depth, their horizontal gradients can be trivially integrated vertically, hence the $H + \eta$ factors multiplying these terms in equations (7.22) and (7.23).

The equation (7.18) for the free surface height η, and the equations (7.22) and (7.23) for $\mathbf{U}$, form the dynamical equations for the barotropic system. Again, this is the barotropic system defined by vertically integrating the continuity and momentum equations between $-H \le z \le \eta$.

7.2.5 Global water budget

For the computation of global budgets, it is useful to consider the global integral of the free surface height tendency given in equation (7.18). Due to the no-normal flow condition, the global integral of the convergence $-\nabla_h \cdot \mathbf{U}$ vanishes. Additionally, for a closed fresh water

budget, the global integral of q_w vanishes. Hence, the global integral of η_t vanishes. This result means that the volume of water in the global ocean is constant, assuming no net water flux into or out of the ocean. Constant total ocean volume results for such a closed system when assuming an incompressible fluid. Assuming the total surface area of the ocean is constant, the global integral of η_t vanishing implies that the global integral of η is a constant. This constant can conveniently be set to zero. Conservation of volume is an important constraint to satisfy with a discretization of the free surface method.

For regional models, the total model volume may vary due to fresh water flux or due to fluxes through open model boundaries. For these cases, the global integral of the free surface height need not be constant in time. Also, for coupled models, there can be source and sink terms on land and in the atmosphere which may render the ocean volume non-constant.

7.3 A linearized barotropic system

The previous discussion made no approximation other than to employ a hydrostatic and Bousinnesq fluid, as well as assumptions in Section 7.1 leading to the expression for the surface pressure. The purpose of this section is to discuss a version of the barotropic system which has been is implemented by Killworth *et al* (1991) and Dukowicz and Smith (1994). This method is also available in MOM, but is not recommended due to its inability to conserve (see Griffies, Pacanowski, Schmidt, and Balaji (2000) for details).

7.3.1 The barotropic system

For the barotropic system, split the vertically integrated transport into two terms

$$\mathbf{U} = \mathbf{U}_0 + \int_0^\eta dz\, \mathbf{u}_h, \tag{7.26}$$

where

$$\mathbf{U}_0 = \int_{-H}^0 dz\, \mathbf{u}_h \tag{7.27}$$

is the vertically integrated transport contained within $-H \leq z \leq 0$, and $\int_0^\eta dz\, \mathbf{u}_h$ is that transport maintained within the free surface layer. Consequently,

$$\partial_t \mathbf{U} = \partial_t \mathbf{U}_0 + \eta_t\, \mathbf{u}_h(\eta) + \int_0^\eta dz\, \partial_t \mathbf{u}_h. \tag{7.28}$$

In a similar manner, split the vertically integrated forcing into two terms

$$\begin{aligned} X &= X_0 + \int_0^\eta dz \left(-\nabla \cdot (u\mathbf{u}) + \frac{uv \tan\phi}{a} + (\kappa_m u_z)_z - \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} + F^u \right) \\ Y &= Y_0 + \int_0^\eta dz \left(-\nabla \cdot (v\mathbf{u}) - \frac{u^2 \tan\phi}{a} + (\kappa_m v_z)_z - \frac{(p_b)_\phi}{a\rho_\circ} + F^v \right), \end{aligned} \tag{7.29}$$

where X_0 and Y_0 are the contributions from $-H \leq z \leq 0$

$$\begin{aligned} X_0 &= \int_{-H}^0 dz \left(-\nabla \cdot (u\mathbf{u}) + \frac{uv \tan\phi}{a} + (\kappa_m u_z)_z - \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} + F^u \right) \\ Y_0 &= \int_{-H}^0 dz \left(-\nabla \cdot (v\mathbf{u}) - \frac{u^2 \tan\phi}{a} + (\kappa_m v_z)_z - \frac{(p_b)_\phi}{a\rho_\circ} + F^v \right). \end{aligned} \tag{7.30}$$

Using these expressions in equation (7.22) for the zonal transport yields

$$\partial_t U_0 - f V_0 + \frac{H\,(p_a + p_s)_\lambda}{a\,\rho_o\,\cos\phi} - X_0 = -\frac{\eta\,(p_a + p_s)_\lambda}{a\,\rho_o\,\cos\phi}$$
$$+ \int_0^\eta dz \left(-u_t + fv - \nabla\cdot(u\mathbf{u}) + \frac{uv\tan\phi}{a} + (\kappa_m u_z)_z - \frac{(p_b)_\lambda}{a\rho_\circ\cos\phi} + F^u \right). \tag{7.31}$$

Note the cancelation of the $\eta_t\, u(\eta)$ term. Use of the zonal momentum equation (7.20) provides for a convenient cancellation of the remaining terms, hence revealing that the vertically integrated transport $\mathbf{U}_0$ satisfies the equation

$$\partial_t U_0 - f V_0 = -\frac{H\,(p_a + p_s)_\lambda}{a\,\rho_o\,\cos\phi} + X_0. \tag{7.32}$$

Similarly, the meridional transport V_0 satisfies

$$\partial_t V_0 + f U_0 = -\frac{H\,(p_a + p_s)_\phi}{a\,\rho_o} + Y_0. \tag{7.33}$$

7.3.2 The shallow water limit

The equations satisfied by the two fields $\mathbf{U}$ and $\mathbf{U}_0$ are rewritten here for purposes of comparison

$$\partial_t \mathbf{U} + f\hat{z}\wedge\mathbf{U} = -(H+\eta)\,\nabla_h(p_a + p_s)/\rho_o + \eta_t\,\mathbf{u}_h(\eta) + \mathbf{X} \tag{7.34}$$
$$\partial_t \mathbf{U}_0 + f\hat{z}\wedge\mathbf{U}_0 = -H\,\nabla_h\,(p_a + p_s)/\rho_o + \mathbf{X}_0. \tag{7.35}$$

Notice that the equation satisfied by $\mathbf{U}_0$ contains nonlinearities *only* within the vertically integrated forcing term $\mathbf{X}_0$. Hence, the "shallow water version" of the $\mathbf{U}_0$ equation, defined by dropping $\mathbf{X}_0$, is linear. In contrast, the shallow water version of the $\mathbf{U}$ equation contains additional nonlinearities arising from the terms

$$\eta_t\,\mathbf{u}_h(\eta) - \eta\,\nabla_h\,p_s/\rho_o = \eta_t\,\mathbf{u}_h(\eta) - (g/2)\,\nabla_h\eta^2. \tag{7.36}$$

Each of these terms is very small in comparison to the other terms, *except* in those regions of the ocean where the depth H is on the order of the free surface height undulations η. It is in these regions that nonlinear wave steepening and breaking become important processes which the nonlinear shallow water equations admit.

7.3.3 The linearized free surface height equation

Recall the two equivalent equations for the free surface height, repeated here for convenience

$$\eta_t = -\nabla_h\cdot\mathbf{U} + q_w$$
$$= w(\eta) - \mathbf{u}_h(\eta)\cdot\nabla_h\eta + q_w. \tag{7.37}$$

Given knowledge of $\mathbf{U}_0$ rather than $\mathbf{U}$, it is possible to determine a linearized free surface height

$$\eta_t^0 = -\nabla_h\cdot\mathbf{U}_0 + q_w. \tag{7.38}$$

As a brief aside, it is useful to note that if one vertically integrates the continuity equation between $-H$ and 0, then uses the bottom kinematic boundary condition (7.15), one finds that

$$w(z=0) = -\nabla_h \cdot \mathbf{U}_0. \tag{7.39}$$

Hence, equation (7.38) can be equivalently written

$$\eta_t^0 = w(0) + q_w. \tag{7.40}$$

Considering this result as a diagnostic for $w(0)$, one sees that nonzero $w(0)$ arises from the difference between the free surface height tendency η_t^0 and the fresh water flux q_w. For example, when there is zero fresh water flux, $w(0) \approx \eta_t^0$, whereas in a steady state, $w(0)$ arises just from the input of fresh water. Note that a positive input of fresh water into the ocean model, with $q_w > 0$, drives a negative vertical velocity $w(0) < 0$.

Of note is the linear nature of equation (7.40) for η^0. This property should be contrasted with the nonlinear kinematic boundary condition satisfied by η. To pursue this point a bit further, start from the surface kinematic boundary condition and write

$$\begin{aligned}(\partial_t + \mathbf{u}_h(\eta)\cdot\nabla_h)\eta &= w(\eta) + q_w \\ &= \left(w(0) + \int_0^\eta dz\, w_z\right) + q_w \\ &= \eta_t^0 + \int_0^\eta dz\, w_z. \end{aligned} \tag{7.41}$$

Hence, neglect of the advection of free surface height, as well as the vertical shear of the vertical velocity within the layer between η and 0, brings $\eta \to \eta^0$. Otherwise, the two heights differ. That is, a linearized version of the equation satisfied by η yields the equation satisfied by η^0.

7.3.4 Summary of the linear barotropic system

The linearized barotropic system consists of the vertically integrated momentum and continuity equations, where the vertical integral extends between $-H \le z \le 0$. The results are the dynamical equations

$$\partial_t \mathbf{U}_0 = -f\hat{z}\wedge\mathbf{U}_0 - H\nabla_h(p_a + p_s)/\rho_o + \mathbf{X}_0 \tag{7.42}$$

$$\partial_t \eta^0 = -\nabla_h\cdot\mathbf{U}_0 + q_w, \tag{7.43}$$

where the vertically integrated forcing is given by

$$\mathbf{X}_0 = \int_{-H}^0 dz\left[-\nabla\cdot(\mathbf{u}_h\,\mathbf{u}) + (\mathbf{u}_h\wedge\hat{z})\,u\tan\phi/a - \nabla_h(p_b/\rho_o) + (\kappa_m\mathbf{u}_z)_z + F^{\mathbf{u}}\right]. \tag{7.44}$$

The shallow water limit consists of setting $\mathbf{X}_0 = 0$. The result is the linear shallow water equations. In general, the barotropic system for $\mathbf{U}_0$ and η^0 is identical to the linearized version of the barotropic equations for $\mathbf{U}$ and η, where the $\mathbf{U}, \eta$ system results from vertically integrating the momentum and continuity equations between $-H \le z \le \eta$.

Note that the barotropic system has been formulated for the vertically integrated transport. After solving for this transport, the vertically averaged velocity can simply be diagnosed and then the full barotropic plus baroclinic velocity field can be constructed. An alternative approach is to directly time step the prognostic equations for the vertically averaged velocity and hence save some computational time by avoiding the extra diagnostic step. However,

undocumented experience with shallow water models (Pacanowski) has shown that time stepping the equations for $\overline{\mathbf{u}}_h$ provides for less numerical stability and introduces more noise. The formulation of MOM's free surface is motivated by this experience.

In the shallow water system, linear waves with speeds $\sqrt{gH}$ are admitted. In the deep ocean, these waves can reach over $200 m/sec$, and they are generally much faster than the first baroclinic wave motions. An explicit time stepping algorithm is therefore constrained by the need to resolve these fast shallow water waves. When $\mathbf{X}_0$ is not neglected, there are additional nonlinear couplings between the shallow water system and the depth dependent baroclinic system. Due to the complexity of this nonlinear coupling, and due to the large contribution from the slower baroclinic modes to the temporal changes of $\mathbf{X}_0$, it is reasonable to hold $\mathbf{X}_0$ constant when integrating over the small barotropic time steps, and to only update them at the long baroclinic time steps. This is the approach used by Blumberg and Mellor (1987), Killworth *et al.* (1991), Hallberg (1995), and in the MOM implementation of the explicit free surface (Section 29.5).

It is useful to compare the linearized barotropic system with the rigid lid method. Indeed, the comparison is trivial: the only difference is that the rigid lid equations set η to zero and introduce a streamfunction to describe the divergence-free transport $\mathbf{U}_0$. The vertically integrated forcing $\mathbf{X}_0$ is formally the same in both the rigid lid and free surface. The essential difference is that $\mathbf{X}_0$ in the free surface contains contributions from a generally nonzero vertical velocity $w(0)$. But note that in the steady state with fresh water input set to zero, the free surface $w(0)$ vanishes. Hence, the steady state solutions realized with the linearized free surface reduce to the rigid lid solutions. Such connection between the two solution methods provides an essential point of departure for developing the free surface.

7.4 Stresses at the ocean surface and bottom

In Section 4.3.3, a discussion was given of the dynamic boundary conditions. These conditions determine how the momentum field is forced at the ocean surface and bottom. That discussion is now extended by performing a few manipulations on the expressions for the vertically integrated forcing discussed in Section 7.2.4. The ideas in this section are also relevant for considering how the baroclinic system is forced at the ocean surface and bottom, and so they will be revisited in that context later in this chapter. The formulation here employs the barotropic equations as defined by vertically integrating between $-H$ and η. The limit of these results for $\eta \to 0$ recovers the expressions relevant for a linearized free surface method.

First, use the kinematic boundary conditions (7.15) and (7.16) to establish the identity

$$\begin{aligned}
\alpha(\eta)\,\eta_t - \int_{-H}^{\eta} dz\,\nabla\cdot(\alpha\,\mathbf{u}) &= \alpha(\eta)\,[\eta_t - w(\eta)] + w(-H)\,\alpha(-H) - \int_{-H}^{\eta} dz\,\nabla_h\cdot(\alpha\,\mathbf{u}_h) \\
&= \alpha(\eta)\left(q_w - \mathbf{u}_h(\eta)\cdot\nabla\eta\right) - \alpha(-H)\,\mathbf{u}_h(H)\cdot\nabla H \\
&\quad - \int_{-H}^{\eta} dz\,\nabla_h\cdot(\alpha\,\mathbf{u}_h) \\
&= \alpha(\eta)\,q_w - \nabla_h\cdot\int_{-H}^{\eta} dz\,\alpha\,\mathbf{u}_h. \qquad (7.45)
\end{aligned}$$

Note that Killworth *et al.* (1991) performed similar manipulations, yet their equations (19) and (20) are missing a $\cos\phi$ factor in the $\partial/\partial\phi$ terms, which precludes them from exposing the convergence as done in equation (7.45). Additionally, they omit the freshwater forcing term.

Second, recall from the discussion of momentum friction in Chapter 9 (i.e., equations (9.187) and (9.193)) that the second order friction operator can be written as a Laplacian acting on the horizontal velocity, plus an extra metric term

$$\mathbf{F} = \nabla_h \cdot (A\,\nabla_h \mathbf{u}_h) + \mathbf{F}_{metric}. \tag{7.46}$$

The biharmonic friction operator described in Section 9.5 has a similar form, and so no loss of generality arises from assuming such. Consequently, the vertical integral of momentum friction can be rewritten as

$$\begin{aligned}\int_{-H}^{\eta} dz\mathbf{F} &= -A\,(\nabla_h \eta)\cdot(\nabla_h \mathbf{u}_h)|_{z=\eta} - A\,(\nabla_h H)\cdot(\nabla_h \mathbf{u}_h)|_{z=-H} \\ &\quad + \int_{-H}^{\eta} dz\mathbf{F}_{metric} + \nabla_h \cdot \int_{-H}^{\eta} dz A\,\nabla_h \mathbf{u}_h. \end{aligned} \tag{7.47}$$

Combining these two results brings the vertically integrated forcing to the form

$$\begin{aligned} X &= \frac{\tau^{\lambda}_{surf}}{\rho_o} - \frac{\tau^{\lambda}_{bottom}}{\rho_o} - \nabla_h \cdot \int_{-H}^{\eta} dz\ (u\,\mathbf{u}_h - A\,\nabla_h u) \\ &\quad + \int_{-H}^{\eta} dz \left(\frac{uv\tan\phi}{a} - \frac{(p_b)_\lambda}{a\rho_\circ \cos\phi} + F^u_{metric} \right), \end{aligned} \tag{7.48}$$

$$\begin{aligned} Y &= \frac{\tau^{\phi}_{surf}}{\rho_o} - \frac{\tau^{\phi}_{bottom}}{\rho_o} - \nabla_h \cdot \int_{-H}^{\eta} dz\ (v\,\mathbf{u}_h - A\,\nabla_h v) \\ &\quad + \int_{-H}^{\eta} dz \left(-\frac{u^2\tan\phi}{a} - \frac{(p_b)_\phi}{a\rho_\circ} + F^v_{metric} \right). \end{aligned} \tag{7.49}$$

In these expressions, the surface and bottom stresses have been introduced

$$\frac{\tau_{surf}}{\rho_o} = \kappa_m \mathbf{u}_z - A\,(\nabla_h \eta)\cdot(\nabla_h \mathbf{u}) + q_w\,\mathbf{u}_h \qquad z = \eta(\lambda,\phi,t) \tag{7.50}$$

$$\frac{\tau_{bottom}}{\rho_o} = \kappa_m \mathbf{u}_z + A\,(\nabla_h H)\cdot(\nabla_h \mathbf{u}) \qquad z = -H(\lambda,\phi). \tag{7.51}$$

Each term is now interpreted physically and their appearance in the numerical model indicated.

7.4.1 Bottom stress

At the ocean bottom, there is a stress arising from two terms. The first is a stress associated with variations in the model's *resolved* bottom topography

$$\tau_{bottom-resolved} = \rho_o\,A\,(\nabla_h H)\cdot(\nabla_h \mathbf{u}). \tag{7.52}$$

For a model with full bottom cells, this term appears as a direct result of discretizing horizontal momentum friction and bottom topography on the velocity cells. For the partial bottom cells discussed in Chapter 26, there are added terms which occur due to variations in the bottom cell thickness.

In addition to bottom stress from resolved topography, there is the possibility of adding a parameterized stress which can be used to account for subgrid scale (SGS) effects. MOM assumes this stress to take the aerodynamic form (see equation (4.31))

$$\tau_{bottom-sgs} = \rho_o\,C_D\,|\mathbf{u}_h|\,\mathbf{u}_h. \tag{7.53}$$

This stress is formally associated with the vertical friction term evaluated at the ocean bottom

$$\rho_o C_D |\mathbf{u}_h| \mathbf{u}_h = \kappa_m \mathbf{u}_z \qquad z = -H(\lambda, \phi). \tag{7.54}$$

The stress $\tau_{bottom-sgs}$ is what the model calls the "bottom momentum flux." The default in MOM is to set C_D to zero, which means that all bottom stress arises from the resolved topography. The introduction of a bottom boundary layer (Chapter 37) in MOM also affects the presence or absence of SGS bottom stress.

Note that the treatment of the ocean bottom is identical in both the rigid lid and free surface methods.

7.4.2 Surface stress

The stress τ_{surf} at the ocean surface arises from two terms

$$\tau_{surf} = \tau_{fresh} + \tau_{winds}. \tag{7.55}$$

The first term is the usual wind stress contribution τ_{wind}. Unless MOM is coupled to a wave model which resolves the interactions between the sea surface and atmospheric winds, the surface wind stress is unresolved and so must be parameterized. As discussed in Section 4.3.3, MOM assumes that this parameterization takes the same aerodynamic form used for the bottom

$$\tau_{wind} = \rho_a C_D^{wind} |\mathbf{u}^{wind}| \mathbf{u}^{wind} \tag{7.56}$$

where C_D is a dimensionless drag coefficient and ρ_a is the atmospheric density. This stress is formally identified with the friction terms evaluated at the ocean surface

$$\rho_a C_D^{wind} |\mathbf{u}^{wind}| \mathbf{u}^{wind} = \kappa_m \mathbf{u}_z - A (\nabla_h \eta) \cdot (\nabla_h \mathbf{u}). \tag{7.57}$$

In the linearized free surface as well as the rigid lid, the contribution from $A (\nabla_h \eta) \cdot (\nabla_h \mathbf{u})$, which arises from the curvature of the free surface, is absent. However, since the identification (7.57) is formal in the sense that no rigorous microscopic treatment of these friction terms is enabled in MOM, the differences are not important in practice. The stress τ_{winds} is what MOM calls the "surface momentum flux."

In addition to the turbulent stress from the winds, fresh water entrained in atmospheric winds introduces momentum into the free surface ocean in the form given by equation (4.34)

$$\tau_{fresh} = q_w \rho_f \mathbf{u}^{wind}, \tag{7.58}$$

where ρ_f is the density of the fresh water. Equivalently, this stress can be thought of as the vertical advection of horizontal momentum across the ocean surface. In this case it should be proportional to the momentum of the fresh water outside the ocean. In general, it might be appropriate to consider the presence of an atmospheric model, a river model, or both, coupled to MOM. One may then wish to evaluate this stress from the zonal wind velocity and the zonal river current. Such detail, however, is not currently incorporated into MOM (although see Section 30.11 for a preliminary river runoff model). As a default, the simplest case

$$\tau_{fresh} = \rho_o u(\eta) q_w \tag{7.59}$$

is assumed. In actuality, there is always some difference in the wind speed and ocean current, and so some effective velocity may be more appropriate. Some consideration to this fact is given in the paper by Pacanowski (1987).

It is useful to further mention the importance of the stress from the fresh water term, especially in seas with a high fresh water flux. The implication of this term is that fresh water entrains also momentum if it enters the ocean with a horizontal velocity, which should be approximately the wind velocity. Thus, the above more complete boundary condition is necessary for an overall momentum conservation. An ocean with neither horizontal shear nor horizontal pressure gradients can be used as a simple test case to see the effect of the modified momentum flux boundary condition. It can be checked easily that the total momentum is constant if water is added at zero wind speed. If the wind speed is equal to the ocean surface velocity, there acts the usual wind stress but additionally the total momentum grows proportionally to the increasing surface height due to the input of volume through the fresh water.

In summary, the complete free surface dynamic boundary condition takes the form

$$q_w\,\mathbf{u}_h + \kappa_m \mathbf{u}_z - A\,(\nabla_h \eta)\cdot(\nabla_h \mathbf{u}) \quad = \quad \frac{\rho_a}{\rho}\,C_D^{wind}\,|\mathbf{u}^{wind}|\,\mathbf{u}^{wind} + q_w\,\mathbf{u}^{wind} \qquad z = \eta(\lambda,\phi,t). \tag{7.60}$$

The left hand side is the vertical momentum flux in the ocean, and the right hand side describes the vertical turbulent momentum flux in the atmosphere-ocean boundary layer. Note that for the fresh water volume flux the mass conserving form (see Section 4.4.3) has been used, although MOM strictly conserves only volume.

7.4.3 Revisiting the surface stress

The surface stress applied at $z = 0$ for the linearized free surface

$$\tau_{surf}/\rho_o \quad \approx \quad q_w\,\mathbf{u}(z=0) + \kappa_m \mathbf{u}_z|_{z=0} \tag{7.61}$$

was found through taking the $\eta \to 0$ limit of the $\eta \neq 0$ results. The following discussion attempts to further illustrate this limit.

For this purpose, subdivide the surface box in the $\eta \neq 0$ case into two boxes. The first box, with $z_1 < z < 0$, is explicitly resolved by the ocean model. It represents the surface model grid cell, and it has a fixed volume. The second layer, between $z = 0$ and $z = \eta$, is considered a *virtual* model cell. It is not explicitly part of the ocean model, and it has a variable volume. When $\eta < 0$, the virtual box overlaps with the $z_1 < z < 0$ box, whereas for $\eta > 0$ it is distinct from the $z_1 < z < 0$ box. Without an extra prognostic equations integrated by the ocean model for the virtual box, the velocity and the tracer concentration in both boxes must be taken to be same. This assumption should be valid since the small virtual box is typically well mixed with the model box due to interactions with the atmosphere. The momentum flux entering the model domain from the virtual box through the level $z = 0$ is given by

$$\mathbf{F}_{z_o} \quad = \quad (\kappa_m \mathbf{u}_z)_{z=0} - (w(z)\mathbf{u}(z))_{z=0}. \tag{7.62}$$

Again, the first term is the turbulent momentum flux, and the second term is an advective flux representing the vertical advection of horizontal momentum. The goal is to incorporate the effects of the virtual box on the resolved box through an appropriate form of the flux $(\kappa_m \mathbf{u}_z)_{z=0}$.

The momentum balance of the virtual upper box can be found from the volume averaged velocity equations by replacing z_1 by $z_0 = 0$. Resolving the momentum equations for the top model box for $\mathbf{F}_{z_o}$, taking the limit as $\eta \to 0$, and neglecting gradients of η, an approximation for the open vertical boundary condition at $z = 0$ is found to be

$$\mathbf{F}_{z_o} \approx \tau_{surf}/\rho_o - \mathbf{u}(z=0)\left(w(z=0) + q_w\right). \tag{7.63}$$

Combined with equation (7.62), this result can be rearranged to the form

$$(\kappa_m \mathbf{u}_z)_{z=0} \approx \tau_{surf}/\rho_o - \mathbf{u}(z=0)\, q_w. \tag{7.64}$$

This result is identical to the expression (7.61) obtained through the formal $\eta \to 0$ limit.

7.5 A comment about atmospheric pressure

Short of providing a realistic atmospheric model for the upper ocean surface conditions, the atmospheric pressure $p_a(\lambda, \phi, t)$ is typically set to zero. Nevertheless, in some cases the atmospheric pressure gradients should be included, especially if open boundary conditions apply with sea level values taken from gauges. So the atmospheric pressure is included in the model. It turns out to be more convenient to do so within the baroclinic equations rather than the barotropic equations. One is allowed such freedom since $p_a(\lambda, \phi, t)$ is depth independent.

7.6 Vertically integrated transport

7.6.1 General considerations

As shown in Section 6.1, without making the assumption that $w(z = 0) = 0$, the barotropic streamfunction provides an incomplete description of the external mode velocity since the vertically integrated velocity is no longer divergence-free. Instead, both a streamfunction and a velocity potential are necessary

$$\mathbf{U} = \hat{z} \wedge \nabla_h \psi + \nabla_h \chi, \tag{7.65}$$

where χ is the velocity potential and $\mathbf{U} = (H + \eta)(\overline{u}, \overline{v})$ is the vertically integrated horizontal velocity. Taking the horizontal divergence of both sides of this equation, and using the free surface height equation (7.18) renders

$$\nabla_h \cdot \nabla_h \chi = -\eta_t + q_w. \tag{7.66}$$

Only in the case of a steady state ocean with zero surface water fluxes can the velocity potential be ignored.

7.6.2 An approximate streamfunction

In general, to evaluate the vertically integrated transport passing between two points, a direct evaluation of the integral (see Section 6.2)

$$T_{ab} = \int_a^b dl\, \hat{n} \cdot \mathbf{U}, \tag{7.67}$$

can be given. Although accurate and complete, this integral does not readily provide a map of transport, and so it looses much of the appeal associated with the barotropic streamfunction used with a rigid lid. Instead, for many practical situations, maps of the function

$$\Psi(\lambda, \phi) = -a \int_{\phi_o}^{\phi} d\phi' \, U(\lambda, \phi') \tag{7.68}$$

may prove useful, where the lower limit ϕ_o is taken at the southern boundary of the domain (either a wall or $-\pi/2$). The meridional derivative of Ψ yields the exact zonal transport

$$-\frac{1}{a}\Psi_{,\phi} = U. \tag{7.69}$$

The longitudinal derivative, however, is given by

$$\frac{1}{a\cos\phi}\Psi_{,\lambda} = V - \frac{a}{\cos\phi}\int_{\phi_o}^{\phi} d\phi' \, \cos\phi' \, \nabla_h \cdot \mathbf{U}, \tag{7.70}$$

where $\cos\phi_o \, V(\lambda, \phi_o) = 0$ was used. The free surface height equation (Section 7.2.3)

$$\nabla_h \cdot \mathbf{U} = -\eta_t + q_w \tag{7.71}$$

indicates that $\nabla_h \cdot \mathbf{U}$ vanishes only when there is zero time tendency of the free surface height and zero fresh water flux through the surface. Hence, the longitudinal derivative leads to the meridional transport plus an error term, where the error term vanishes in the case of a steady state ($\eta_t = 0$) and a zero fresh water flux. It is useful to see how large the error term might be. For this purpose, zonally integrate $\Psi_{,\lambda}$ to find

$$\begin{aligned} \Psi(\lambda, \phi) - \Psi(\lambda_o, \phi) &= a\cos\phi \int_{\lambda_o}^{\lambda} d\lambda' \, V(\lambda', \phi) \\ &- a^2 \int_{\lambda_o}^{\lambda} d\lambda' \int_{\phi_o}^{\phi} d\phi' \, \cos\phi' \, (-\eta_t + q_w). \end{aligned} \tag{7.72}$$

For example, with $-\eta_t + q_w = 1m/year$ applied over a $1000km \times 1000km$ area, the error term contributes much less than a Sv to the streamfunction. Cases where the differences are larger certainly can be constructed. But for many diagnostic purposes, the differences are negligible.

By construction, Ψ reduces to the barotropic streamfunction in the case of a rigid lid model where $\nabla_h \cdot \mathbf{U} = 0$. However, this is not a unique choice and alternatives do exist. For example,

$$\Psi^*(\lambda, \phi) = \Psi(\lambda_o, \phi) + a\cos\phi \int_{\lambda_o}^{\lambda} d\lambda' \, V(\lambda', \phi), \tag{7.73}$$

gives

$$U(\lambda, \phi) = -\frac{1}{a}\Psi^*_{,\phi} + a\cos\phi \int_{\lambda_o}^{\lambda} d\lambda' \nabla_h \cdot \mathbf{U} \tag{7.74}$$

$$V = \frac{1}{a\cos\phi}\Psi^*_{,\lambda}. \tag{7.75}$$

Ψ^* has the advantage that zonal derivatives give the exact meridional transport. In the end, it might be useful to plot Ψ and Ψ^* and compare. These two streamfunctions are indeed plotted in MOM's snapshots when the free surface is enabled. In snapshots, Ψ is called *psiU*, and Ψ^*

is called *psiV*. The reference points are ϕ_o is the southern-most latitude and λ_o is the western-most longitude.

Additionally, as each streamfunction is defined only up to an arbitrary constant, it is useful to specify this constant in a manner to correspond to that resulting from the rigid lid approximation. The option *explicit_psi_normalize* normalizes each streamfunction by the value at $\lambda = 300°$ and $\phi = -20°$, which corresponds to a point over South America. This convention corresponds to taking the Americas as the zeroth island in the rigid lid method.

A final example distributes the $(-\eta_t + q_w)$ piece evenly:

$$\Psi^{**} = -\frac{a}{2}\int_{\phi_o}^{\phi} d\phi' \, [U(\lambda,\phi') - U(\lambda_o,\phi')] + \frac{a\cos\phi}{2}\int_{\lambda_o}^{\lambda} d\lambda' \, V(\lambda',\phi) \tag{7.76}$$

yields

$$U = -\frac{1}{a}\Psi^{**}_{,\phi} + \frac{a\cos\phi}{2}\int_{\lambda_o}^{\lambda} d\lambda' \, (-\eta_t + q_w) \tag{7.77}$$

$$V = \frac{1}{a\cos\phi}\Psi^{**}_{,\lambda} + \frac{a}{2\cos\phi}\int_{\phi_o}^{\phi} d\phi' \, \cos\phi' \, (-\eta_t + q_w). \tag{7.78}$$

This streamfunction is not computed in MOM.

Chapter 8

The tracer budget

The purpose of this chapter is to discuss the tracer budget in the continuum as well as finite volumes. Discrete budgets are described in Section 30.6.

8.1 The continuum tracer concentration budget

Before starting, it is useful to summarize the terms appearing in a local budget for the tracer concentration, as determined by the tracer equation

$$T_t = -\nabla \cdot (\mathbf{u}\, T + \mathbf{F}). \tag{8.1}$$

In this equation, T represents the amount of a substance (generically called a *tracer*) per unit volume; i.e., it is a *concentration*. Equation (8.1) is the conservation equation for this tracer concentration (e.g., Gill 1982, chapter 4 or Apel 1987, chapters 3,4). Within the Boussinesq approximation employed by MOM, $T = \rho_o\, s$, representing the mass of salt per volume of seawater, satisfies this equation. Note that salinity s, which is a prognostic variable carried in MOM, is dimensionless since it represents the grams of salt per kilogram of seawater (*ppt*). Additionally, with the same Boussinesq approximation, the amount of potential heat per volume $T = \rho_o\, c_p\, \theta$, satisfies this equation (in this context, *heat* is considered a "substance"). Note that c_p, the specific heat at constant pressure, is held fixed within the Boussinesq approximation (e.g., Chandrasekhar 1961). Passive tracers, where T represents the amount of tracer per volume, also satisfy this equation.

The terms on the right hand side were discussed in Section 4.2.4. In particular, $\mathbf{F} = \left(\mathbf{F}^h, F^z\right)$ are the horizontal and vertical tracer flux components distinct from the advective flux $\mathbf{u}T$ due to the resolved velocity field. Typically in the mixed layer, which includes the region between the free surface $z = \eta$ and the bottom of the upper ocean model box $z = z_1$, the fluxes take the form $\mathbf{F}_h(T) = -A_h \nabla T$, and $F^z(T) = -\kappa_h T_z$. More general closures can be considered. In the following, these fluxes are not specified explicitly and they are generally dependent on the space-time resolution used in the model. For purposes of terminology, they will be referred to as *diffusive* fluxes.

8.2 Finite volume budget for the total tracer

The time tendency for the total amount of tracer in a cell is given by

$$\partial_t [[T]_k] = [[T_t]_k] + [T(\eta)\, \eta_t]\, \delta_{k1}, \tag{8.2}$$

where the square brackets denote volume integration. For example, within the Boussinesq approximation, the volume integral of $T = \rho_o s$ represents the total mass of salt within this volume. Likewise, the volume integral of $T = \rho_o c_p \theta$ represents the total heat (in units of energy) within this volume. For T representing the mass/volume of a passive tracer, the volume integral of T is the total mass of this tracer in the volume.

The first term on the right hand side of equation (8.2) arises from the time tendency of the tracer concentration inside the box. The second term occurs only for a surface cell and comes from the time tendency of the surface height multiplied by the surface tracer concentration. This term alters the tracer budget through changing the volume of the box, and it is absent in the rigid lid approximation for which $\eta = 0$. Note that the value of the tracer in this expression is that at the ocean side of the free surface $T(\eta)$.

Use of the surface kinematic boundary condition (7.16) brings the tracer budget equation (8.2) to the form

$$\partial_t[[T]_k] \quad = \quad [[-\nabla \cdot (\mathbf{u}T + \mathbf{F})]_k] + \left[T \left(w + q_w - \mathbf{u}_h \cdot \nabla_h \eta \right) \right] \delta_{k1}, \tag{8.3}$$

where all the terms multiplying δ_{k1} are evaluated at $z = \eta$. Integration by parts leads to

$$[[\nabla_h \cdot (\mathbf{u}_h T + \mathbf{F}_h)]_k] \quad = \quad [\nabla_h \cdot [\mathbf{u}_h T + \mathbf{F}_h]_k] - [\nabla_h \eta \cdot (\mathbf{u}_h T + \mathbf{F}_h)] \delta_{k1}, \tag{8.4}$$

where again the terms multiplying δ_{k1} are evaluated at $z = \eta$. Using this result in equation (8.3) brings about a cancellation of the $\nabla_h \eta \cdot \mathbf{u}_h T$ term appearing in the δ_{k1} part of the budget. The vertical divergence integrates to

$$[[\partial_z (w T + F^z)]_k] \quad = \quad [(w T + F^z)_{z=z_{k-1}} - (w T + F^z)_{z=z_k}], \tag{8.5}$$

which is the divergence of the area integrated flux across the horizontal cell faces. These results render

$$\begin{aligned} \partial_t[[T]_k] \quad &= \quad -[\nabla_h \cdot [\mathbf{u}_h T + \mathbf{F}_h]_k] - [(w T + F^z)_{z=z_{k-1}} - (w T + F^z)_{z=z_k}] \\ &+ \quad [T (w + q_w) + \nabla_h \eta \cdot \mathbf{F}_h] \delta_{k1} \end{aligned} \tag{8.6}$$

For a surface cell with $k = 1$, there is a cancellation of the $T(\eta) w(\eta)$ term to yield the budget

$$\partial_t[[T]_{k=1}] \quad = \quad -[\nabla_h \cdot [\mathbf{u}_h T + \mathbf{F}_h]_k] + [(w T + F^z)_{z=z_1} + (q_w T - F^z + \nabla_h \eta \cdot \mathbf{F}_h)_{z=\eta}] \tag{8.7}$$

8.3 Surface tracer flux

From the budget equation (8.7), it is possible to identify the total concentration flux (dimensions *tracer/volume* $\times$ *velocity*) across the air-sea interface as

$$Q_{wT} = -F^z(T) + \mathbf{F}_h(T) \cdot \nabla_h \eta + T q_w \qquad z = \eta. \tag{8.8}$$

The area integral $[Q_{wT}]$ represents the area integrated tracer concentration flux entering or leaving the ocean across the free surface interface. Equivalently, it is the total amount of tracer substance crossing the ocean surface per unit time (dimensions *tracer/time*). The subscript w signifies that the flux is measured at the ocean or "water" side of the interface. The sign convention is that Q_{wT} is counted as positive if it is directed *into* the ocean.

It is useful to note that the total surface tracer flux (8.8) has a direct analogue in the equation (7.50) for the surface momentum stress, which is rewritten here for convenience

$$\frac{\tau_{surf}}{\rho_o} = \kappa_m \mathbf{u}_z - A\,(\nabla_h \eta)\cdot(\nabla_h \mathbf{u}) + q_w\,\mathbf{u}_h \qquad z=\eta. \tag{8.9}$$

For momentum, the friction terms $\kappa_m \mathbf{u}_z - A\,(\nabla_h \eta)\cdot(\nabla_h \mathbf{u})$ are formally associated with parameterized turbulent momentum fluxes across the air-sea interface. Likewise, diffusive terms $-F^z + \mathbf{F}_h \cdot \nabla_h \eta$ are formally associated with parameterized turbulent tracer fluxes across the air-sea interface. These fluxes can be derived from a boundary layer model and are discussed in Section 8.4.

8.4 Comments on the surface tracer fluxes

The diffusive or turbulent part of the surface tracer flux

$$Q_{wT}^{diff} = \mathbf{F}_h(T)\cdot\nabla_h \eta - F^z(T) \tag{8.10}$$

must be specified by a boundary condition. This flux enters the ocean after passing a sequence of boundary layers between atmosphere and ocean. The tracer transport through these layers is governed by a complex superposition of several processes, as molecular and turbulent diffusion, wave breaking, Langmuir circulation, radiation processes, chemical reactions and biological processes. Strong local gradients as in a thermal skin layer may be built up. Thus, the calculation of Q_{wT}^{diff} is a problem in nonequilibrium thermodynamics, turbulence theory, physical chemistry, and/or biophysics and is a rather complex problem by itself. For an introduction see the book by de Groot and Mazur (1962), or the articles by Forland *et al.* (1988) or Doney (1995).

The set of basic equations of MOM does not provide information on the turbulent tracer transport, and the complicated vertical structure of the air-sea boundary layer is not resolved. However, especially for long time integrations, the surface boundary conditions are cruical for the accuracy of MOM integrations. Therefore it is useful to make a few general statements here in hopes of exposing the basic issues.

First, recall that the dimensions of the flux Q_{wT} are tracer concentration times velocity (see the definition of Q_{wT} in equation (8.8)). As such, Q_{wT}^{diff} represents the total mass (or energy for the case of heat) of a tracer passing through the sea surface per unit area and per unit time. The amount of the tracer substance crossing the air-sea interface fulfills certain conservation laws. For example, if the tracer is a substance i, the mass of the tracer passing the air-sea interface must be conserved, i.e., the flux at the ocean and the air side of the interface is the same,

$$Q_{wi}^{diff} = Q_{ai}^{diff} \tag{8.11}$$

If chemical reactions at the sea surface are possible, more general expressions can be found from the conservation of mass for the involved chemical elements. If the tracer is an energy, the reaction heats Q_R from phase transitions or chemical reactions must be included into the total energy balance at the air-sea interface,

$$Q_{wi}^{diff} = Q_{ai}^{diff} + Q_R. \tag{8.12}$$

At the ocean side of the boundary layer, the flux (equation (8.8))

$$Q_{wT} = T\,q_w + \mathbf{F}_h(T)\cdot\nabla_h \eta - F^z(T) \qquad z=\eta \tag{8.13}$$

appears at the top of the uppermost ocean box. As mentioned earlier, the first contribution, $T(\eta)\,q_w$, brings about a change in tracer concentration due to changes in ocean volume upon introducing fresh water. The air-sea interface acts on a tracer like a filter with a transparency depending on the difference of the chemical potentials of the tracers in the air and in water. For example, ionic tracers such as dissolved salt have a total air-sea flux Q_{wT} which is zero due to the large hydration energy. On the other hand, many weakly dissolved trace gases leave the ocean together with evaporating water. For this reason, the remaining terms $Q_{wT}^{diff} = \mathbf{F}_h(T) \cdot \nabla_h \eta - F^z(T)$ in equation (8.13) are *not* independent of the fresh water flux term $T(\eta)\,q_w$. Rather, they describe the turbulent diffusive tracer flux at the top of the surface box, and this flux is established by both the tracer concentration gradients across the air-sea interface, and tracer gradients within the ocean boundary layer coming from the chemical tracer kinetics in connection with the fresh water flux.

The next issue concerns how approximations for the tracer fluxes can be found. The complexity of the boundary layer prevents a direct coupling of an ocean circulation model with a boundary layer model which resolves the genuine dynamics of the boundary layer. Alternately, the tracer fluxes are often calculated from empirical approximations. The difference of the bulk tracers in the atmosphere and the ocean, $T_a - T_s$, is taken as the thermodynamic force for the diffusive tracer flux. Then the tracer flux has the general form

$$Q_{wT}^{diff} \sim C_T u^{wind} (T_a - T_s) \tag{8.14}$$

The empirical tubulent kinetic coefficient C_T summarizes the complex dynamics within the boundary layer. It depends mainly on the tracer properties, on the wind velocity and on the stability of the boundary layer.

In the following, the boundary conditions for the fresh water flux, heat flux, and salt flux are specified in more detail.

8.4.1 Fresh water flux into the free surface model

The calculation of the fresh water flux requires a boundary layer model to be coupled with MOM. Here, a very simple version is described which permits a first order guess for the fresh water flux. Fresh water flux may have two different components, rain Q_w^R and vapour from condensation or evaporation at the sea surface, Q_w^V,

$$Q_w = Q_w^R + Q_w^V, \tag{8.15}$$

The fresh water velocity, q_w, which is needed in the boundary condition is

$$q_w = \frac{Q_w}{\rho} \tag{8.16}$$

The amount of rain can be provided by an atmosphere model or field data might be used.

If the boundary layer is in a turbulent steady state the water vapour flux through the boundary layer can be parameterized as

$$Q_w^V = \rho_a C_w u^{wind} (h_a - h_s)\,. \tag{8.17}$$

Here the thermodynamic forcing is the difference of the specific humidity h_a in some reference height (usually 10 m) and at the sea surface, h_s. The specific humidity is defined as the mass ratio

$$h = \frac{m_w}{m} = \frac{\rho_w}{\rho_a} \tag{8.18}$$

where m is the total mass of the air in a volume element and m_w the mass of water vapour in the same volume. Alternatively the difference of the water vapour pressure or the partial density of water vapour can be used. The kinetic coefficient $C_w u^{wind}$ describes the vertical turbulent diffusion in the boundary layer and is a function of the wind speed u^{wind} in the reference height and of the stability of the atmospheric boundary layer. There is a considerable literature on empirical parameterizations of C_w from experimental data sets. Details can be found for example in Large and Pond (1982), Smith and Dobson (1984), Rosati and Miyakoda (1988) and in the literature cited there.

As a result for the calculation of the fresh water flux the specific humidity h_a and the wind velocity u^{wind} must be known in some reference height. This information may come from an atmosphere model. For the calculation of the drag coefficient C_w additional information on the stability of the boundary layer, i.e, on the atmosphere temperature is necessary. The specific humidity at the sea surface, h_s, can be calculated from the assumption of saturated water vapour immediately over the sea surface.

8.4.2 Heat flux into the free surface model

For temperature, the heat balance in the upper box has to be considered. The heat flux enters the ocean through a boundary layer which has an atmospheric and an oceanic component. There are four major contributions to the heat flux,

- the insolation,
- the infrared radiation balance between ocean and atmosphere,
- the sensible heat flux, which is basically a turbulent diffusion of heat,
- the heat transfer in connection with a fresh water flux. This effect involves a direct energy flux in connection with the flux of matter and a latent heat due to the liquid-vapour phase transition at the sea surface.

The insolation and the infrared radiation are not discussed here. For simple parameterization see e.g. Smith and Dobson (1984) or Rosati and Miyakoda (1988).

The enthalpy flux Q_{ae} through the top of the atmosphere-ocean boundary layer is,

$$Q_{ae} = Q_{ae}^{fresh} + Q_{ae}^{rad} + Q_{ae}^{sens}. \tag{8.19}$$

The radiative component Q_{ae}^{rad} includes insolation and the infrared radiation from the ocean and the atmosphere. The thermal radiation is emitted or absorbed in a thin skin layer at the sea surface and the approximation of a surface flux is justified. For the structure of the thermocline it may be important to resolve the vertial absorption profile of short wave radiation. To do this, the short wave radiation must be removed from the surface flux and the vertical divergence of the short wave radiation must be included in the source term. Q_{ae}^{sens} describes the turbulent diffusion of heat, Q_{ae}^{fresh} is the heat flux in connection with the heat capacity of the fresh water advected relative to the sea surface. Under the assumption that the heat flux in the boundary layer has no vertical divergence, the enthalpy flux from the bottom of the boundary layer into the ocean, Q_{we}, is

$$Q_{we} = Q_{ae} + Q_e^{lat}, \tag{8.20}$$

where Q_e^{lat} is the latent heat from that amount of fresh water which undergoes a phase transition at the air-sea interface and can be calculated from the water vapour flux q_w^V,

$$Q_e^{lat} = LQ_w^V. \tag{8.21}$$

L is the evaporation heat of fresh water. Q_e^{lat} is positive if the ocean gains heat by condensation and negative if heat is used for evaporation. It is a common approximation that the latent heat flux goes directly into the ocean and leaves the atmosphere temperature unaffected.

For a simple parameterization of the sensible heat flux the difference of the bulk virtual potential temperature of the atmosphere, θ_{va} and the ocean, θ_{vs}, is assumed as the thermodynamic forcing function,

$$Q_{ae}^{sens} = \rho_a c_{ap} C_T u^{wind} (\theta_{va} - \theta_{vs}). \tag{8.22}$$

As for the fresh water flux the kinetic coefficient $C_T u^{wind}$ describes the turbulent vertical diffusion of heat and can be parameterized in terms on the wind speed and the stability of the atmosphere. c_{ap} is the specific heat of air at constant pressure, ρ_a the density of air. The sign convention is to count a heat flux directed into the ocean as positive.

The heat flux between atmosphere and air-sea boundary layer due to the heat capacity of the fresh water is

$$Q_{ae}^{fresh} = \rho_w c_p T_R Q_w^R + \rho_a c_{ap} \theta_a Q_w^V. \tag{8.23}$$

T_R is the temperature of the liquid fresh water flux, i.e. of rain, ρ_w the fresh water density, T_a the temperature of vapour, which should be the atmosphere temperature. Usually, T_R is not known and simpler approximations are necessary.

Finally, the boundary condition for the potential temperature θ is

$$\begin{aligned} Q_{w\theta}^{diff} &= \theta(\eta) Q_w + \nabla_h \eta \cdot \mathbf{F}_h(\theta) - F^z(\theta), \\ &= (c_p \rho)^{-1} Q_{we} \\ &= (c_p \rho)^{-1} \left(Q_{ae}^{rad} + Q_{ae}^{sens} + LQ_w^V + c_p T_r Q_w^R + c_{ap} \theta_a Q_w^V \right). \end{aligned} \tag{8.24}$$

Chapter 9

Momentum friction

The purpose of this chapter[1] is to discuss the formulation of momentum friction used in MOM. Maintaining certain symmetry properties of the frictional stress tensor guarantees that the resulting friction vector, which is constructed as the covariant divergence of the stress tensor, dissipates total kinetic energy without introducing internal sources of angular momentum. Providing a representation of these properties using curvilinear coordinates is facilitated with some of the tools of tensor analysis.

The central reference for this chapter is the review article by Smagorinsky (1993) as well as some unpublished notes of his from 1963. The related papers by Williams (1972) and Wajsowicz (1993) are also of use. These papers are generally complete, yet the derivations lack some of the tools of modern tensor analysis. Consequently, it has been found useful to rederive the results in this chapter using such tools in hopes of adding some clarity and generality to arbitrary orthogonal coordinates. Smagorinsky based much of his ideas on works from elasticity theory (e.g., Love 1944, Landau and Lifshitz 1986, Synge and Schild 1949, and Segel 1987). The fluid mechanics books by Aris (1962) and Landau and Lifshitz (1987) also consider many of the matters dealt with in the following. The mathematical formalism of Aris, which is consistent with the tensor analysis used by many mathematical physicists today, is closely followed in this chapter. Those more familiar with general relativity will also find such books as Weinberg (1972) useful.

9.1 History of friction in MOM

A brief history of discretizing friction in MOM is the following:

- Bryan (1969): Scalar Laplacian plus metric term for constant viscosity.
- Cox (1984): Omitted metric terms for constant viscosity.
- Rosati and Miyakoda (1988): Smagorinsky scheme in which friction, with a non-constant viscosity, was determined from derivatives of the stress tensor components.
- MOM1: Reintroduced metric terms for constant viscosity.
- MOM2/MOM3: Scalar Laplacian plus constant viscosity metric terms of Bryan (1969) and non-constant viscosity terms of Wajsowicz (1993).

[1]This chapter benefited greatly by comments from Bob Hallberg.

- MOM3/MOM4 (after Summer 1999): Functional formalism in which discretization of the derivatives of the stress tensor are provided. Approach valid for constant and non-constant viscosity and generalizes easily to arbitrary orthogonal horizontal curvilinear coordinates.

In the remainder of this chapter as well as Appendix D, the details of these approaches will be described.

9.2 Basic properties of the stress tensor

The forces acting on an element of a continuous media are of two kinds. *External or body* forces, such as gravitation, Coriolis, or electromagnetic forces, act throughout the media. *Internal or contact* forces, such as pressure forces, act on an element of volume through its bounding surface. The balance between these forces and acceleration leads, through Newton's second law, to the equations of motion. Furthermore, if all torques acting on the fluid arise from macroscopic forces, which is the case in typical Newtonian fluids, then fluid elements respect an angular momentum conservation law. As MOM assumes the fluid it simulates to be Newtonian, ideally its solutions conserve angular momentum for closed systems.

The stresses acting within a continuous media can be organized into a second order stress tensor with generally 3×3 independent elements. The divergence of these stresses gives rise to the internal forces acting in the media. As seen in the following discussion, a proper account of the angular momentum budget implies that the stress tensor is symmetric, which brings the number of independent stress elements down to six. It is useful to note that symmetry of the stress tensor is equivalent to *Cauchy's reciprocal theorem* (section 5.13 of Aris), which says that each of two stresses at a point has an equal projection on the normal to the surface on which the other acts.

For simplicity, the analysis in this section assumes Cartesian coordinates. Generalizations are straightforward (e.g., Chapters 7 and 8 of Aris).

9.2.1 The deformation or rate of strain tensor

Consider two infinitesimally close fluid parcels with material coordinates ζ^s and $\zeta^s + d\zeta^s$, where $s = 1, 2, 3$. The components (u^1, u^2, u^3) of the velocity for these two parcels differ by the increment

$$\begin{aligned} du^m &= \frac{\partial u^m}{\partial \zeta^s} d\zeta^s \\ &= \frac{\partial u^m}{\partial \zeta^s} \frac{\partial \zeta^s}{\partial x^n} dx^n \\ &= \frac{\partial u^m}{\partial x^n} dx^n. \end{aligned} \tag{9.1}$$

The velocity derivatives $\partial u^m / \partial x^n$ form the components to a second order tensor. In order to attach physical significance to this tensor, it is useful to separately consider its symmetric and anti-symmetric components, which are written

$$u_{m,n} = \Omega_{mn} + e_{mn}, \tag{9.2}$$

where

$$2\,\Omega_{mn} = u_{m,n} - u_{n,m} \tag{9.3}$$

$$2\,e_{mn} = u_{m,n} + u_{n,m}, \tag{9.4}$$

and $u_m = g_{mn}\,u^n$ are the covariant components to the velocity vector. Note that in curvilinear coordinates, the partial derivatives appearing in Ω_{mn} and e_{mn} generalize to covariant derivatives.

The anti-symmetric piece of the velocity derivative tensor is related to the vorticity through

$$\begin{aligned} 2\,\omega^i &= -\epsilon^{ijk}\,\Omega_{jk} \\ &= \epsilon^{ijk}\,u_{k,j}, \end{aligned} \tag{9.5}$$

where ϵ^{ijk} is the Levi-Civita symbol defined in Section 4.6.3. In conventional Cartesian vector notation, this result takes the form

$$\omega = \frac{1}{2}\,\nabla \wedge \mathbf{u}. \tag{9.6}$$

Standard results from fluid mechanics establish the connection between vorticity and rigid body rotation of a fluid parcel. If the motion is completely rigid, which means that it consists of a translation plus a rotation, then the symmetric part of the velocity derivative tensor vanishes. Consequently, the symmetric tensor e_{mn} is called the *deformation* or *rate of strain* tensor since it represents deviations from rigid body motion.

To provide a further interpretation of the strain tensor, consider the squared distance between two infinitesimally close material parcels of fluid

$$\begin{aligned} ds^2 &= g_{mn}\,dx^m\,dx^n \\ &= g_{mn}\,\frac{\partial x^m}{\partial \zeta^p}\,\frac{\partial x^n}{\partial \zeta^q}\,d\zeta^p\,d\zeta^q, \end{aligned} \tag{9.7}$$

where $g_{mn} = \delta_{mn}$ since we are working with Cartesian coordinates. The material time derivative of this distance is given by

$$\frac{D(ds^2)}{Dt} = g_{mn}\left(\frac{\partial u^m}{\partial \zeta^p}\,\frac{\partial x^n}{\partial \zeta^q} + \frac{\partial x^m}{\partial \zeta^p}\,\frac{\partial u^n}{\partial \zeta^q}\right)d\zeta^p\,d\zeta^q \tag{9.8}$$

where $D\zeta^p/Dt = 0$ since ζ^p are material coordinates. Use of the chain rule in the forms

$$\frac{\partial u^m}{\partial \zeta^p}\,d\zeta^p = u^m_{\ ,n}\,dx^n \tag{9.9}$$

$$\frac{\partial x^m}{\partial \zeta^p}\,d\zeta^p = dx^m \tag{9.10}$$

renders

$$\frac{D(ds^2)}{Dt} = 2\,e_{mn}\,dx^m\,dx^n, \tag{9.11}$$

or equivalently

$$\frac{1}{ds}\,\frac{D(ds)}{Dt} = e_{mn}\,\frac{dx^m}{ds}\,\frac{dx^n}{ds}. \tag{9.12}$$

Now dx^m/ds is a component of the unit vector which points from one fluid parcel to the other. Hence, equation (9.12) says that the rate of change of the infinitesimal distance separating the two parcels, as a fraction of the distance, is related to the relative position of the parcels through the strain tensor. Section 4.42 of Aris (1962), amongst other places, provides further elaboration.

9.2.2 Relating strain to stress

Newton's second law of motion provides a relation between forces on a fluid parcel and the parcel's acceleration. The forces are given by the sum of the external and internal forces. Relating the stress, whose divergence yields the internal forces, to the strain, which arises from the kinematics of parcel deformations, forms a fundamental problem in continuum mechanics.

In elasticity theory, the relation between stress and strain is typically assumed to follow some form of Hooke's law. In its simplest form, this "law" linearly relates the stress to the strain. In fluid dynamics, it is common to also assume a stress-strain relation in the form of Hooke's law. The details of this relation often depend quite strongly on the properties of the fluid as well as the flow state. Such dependencies can generally make the fluid's stress-strain relation nonlinear.

Under hydrostatic balance, the only form of stress on a fluid parcel is due to the pressure. Hence, the stress tensor for such a state takes the form

$$T^{ij} = -p\,\delta^{ij} \tag{9.13}$$

where p is the pressure and δ^{ij} is the Kronecker delta. Chapter 1 of Salmon (1998) provides some comments on the implicit identification of this pressure with thermodynamic pressure.

When the fluid undergoes deformations, there will be further stresses which bring the stress tensor to the more general form

$$T^{ij} = -p\,\delta^{ij} + \tau^{ij}. \tag{9.14}$$

The divergence of τ^{ij} is typically associated with dissipative stresses in the fluid, which motivates the name *frictional stress tensor*. For a Newtonian fluid, the frictional stress tensor can be written

$$\tau^{ij} = \rho\, C^{ijmn}\, e_{mn}. \tag{9.15}$$

In general, this relation between stress and strain is of the form of Hooke's law, where the components C^{ijmn} of the fourth-order kinematic *viscosity tensor* can depend on the state of the fluid. Assuming this form for the internal stresses, the essential problem with subgrid scale parameterization of momentum fluxes reduces to determining appropriate forms for C^{ijmn}.

9.2.3 Angular momentum and symmetry of the stress tensor

As mentioned previously, the symmetric deformation or strain tensor vanishes for motion consisting of rigid rotation plus uniform translation. In such cases, the generalized Hooke's law (9.15) says that the stress tensor T^{ij} reduces to $-p\,\delta^{ij}$ since τ^{ij} vanishes. The purpose of this section is to provide some further details regarding these ideas and their connection to conservation of angular momentum.

The continuum form of Newton's law is given by

$$\rho \frac{Du^i}{Dt} = \rho f^i + T^{ij}_{,j} \tag{9.16}$$

where ρ is the mass density, and f^i are components to external or body forces such as those arising from gravity and the Coriolis force. $T^{ij}_{,j}$ is the divergence of the stress tensor, where T^{ij} is written in the form (9.14) which incorporates the pressure. A component of the angular momentum for a fluid parcel is given by

$$L_i = \epsilon_{ijk} x^j u^k \rho \, dV, \tag{9.17}$$

where $\rho \, dV$ is the mass of the infinitesimal parcel. The material time derivative of this angular momentum is given by

$$\frac{DL_i}{Dt} = \epsilon_{ijk} x^j \frac{Du^k}{Dt} \rho \, dV, \tag{9.18}$$

where $D(\rho \, dV)/Dt = 0$ through conservation of mass. For a Boussinesq fluid, ρ appears as the constant ρ_o, and $D(dV)/Dt = 0$ then follows from volume conservation. Substituting Newton's law into this expression leads to

$$\frac{DL_i}{Dt} = \epsilon_{ijk} (x^j \rho f^k + x^j T^{km}_{,m}) \, dV. \tag{9.19}$$

The first term accounts for torques placed on the parcel from external forces. The second term arises from torques on the fluid from internal stresses. To further interpret the second term, consider the budget for total angular momentum of the fluid, which is obtained by integrating over the fluid volume

$$\frac{DL_i^T}{Dt} = \int \epsilon_{ijk} (x^j \rho f^k + x^j T^{km}_{,m}) \, dV. \tag{9.20}$$

Now integrate by parts on the stress tensor term to find

$$\int \epsilon_{ijk} x^j T^{km}_{,m} \, dV = \int \epsilon_{ijk} [\partial_m (x^j T^{km}) - T^{kj}] \, dV. \tag{9.21}$$

The first term integrates to a boundary contribution, which is non-vanishing for cases in which there are torques arising from boundary stresses. The second term is a volume contribution and it picks out the term $\epsilon_{ijk} \tau^{kj}$, since $\epsilon_{ijk} \delta^{kj} = 0$. For most fluids, such as ocean water, the internal torques are balanced and so there will be no net contribution to angular momentum from internal stresses. This case can be ensured if the frictional stress tensor is symmetric

$$\tau^{mn} = \tau^{nm}, \tag{9.22}$$

which renders $\epsilon_{ijk} \tau^{kj} = 0$.

9.3 The stress tensor in Cartesian coordinates

In general, the viscosity tensor C^{ijmn} contains 81 degrees of freedom ($81 = 3 \times 3 \times 3 \times 3$). However, the properties just described, and others to follow, greatly reduce this number. The purpose of this section is to provide such a reduction. Thereafter, the form of the stress tensor and friction vector will be written. For simplicity, the coordinates will again be assumed Cartesian in this section.

9.3.1 Generalized Hooke's law form

By assuming the form $\tau^{mn} = \rho\, C^{mnij}\, e_{ij}$, one immediately assumes that the only relevant forms of the viscosity tensor are those satisfying

$$C^{mnij} \;=\; C^{mnji} \tag{9.23}$$

since the strain or deformation tensor e_{ij} is symmetric. This constraint reduces the degrees of freedom to $3 \times 3 \times 6 = 54$. The 6 arises from the $3 + 2 + 1 = 6$ degrees of freedom in a symmetric 3×3 matrix.

9.3.2 Angular momentum

Assuming a symmetric stress tensor brings about the following symmetry on the viscosity tensor

$$C^{mnij} \;=\; C^{nmij}. \tag{9.24}$$

As such, the total degrees of freedom become $6 \times 6 = 36$, which are the same degrees of freedom in a 6×6 matrix.

9.3.3 Dissipation of total kinetic energy

The budget for kinetic energy of a fluid parcel is given by

$$\begin{aligned}
\frac{\rho}{2}\frac{D(\delta_{ij}\, u^i\, u^j)}{Dt} &= \rho\, \delta_{ij}\, u^i\, f^j + \delta_{ij}\, u^i\, T^{jk}_{\;\;,k} \\
&= \rho\, u_i\, f^i + \partial_k\,(u_j\, T^{jk}) - u_{j,k}\, T^{jk} \\
&= \rho\, u_i\, f^i + \partial_k\,(u_j\, T^{jk}) - e_{jk}\, T^{jk} \\
&= \rho\, u_i\, f^i + \partial_k\,(u_j\, T^{jk}) + p\, u^j_{\;\;,j} - e_{jk}\, \tau^{jk}.
\end{aligned} \tag{9.25}$$

The first term on the right hand side arises from work done by external forces. The second term, when integrated over the fluid domain, accounts for work done at boundaries by the stresses. The third term arises from pressure work against changes in the parcel's volume. This term vanishes for a volume conserving fluid. The fourth term is present throughout the fluid domain, and it can be written

$$e_{ij}\, \tau^{ij} = \rho\, e_{ij}\, C^{ijmn}\, e_{mn}. \tag{9.26}$$

In general, this term is sign-indefinite. However, for a frictional stress tensor which manifests dissipative friction at each point in the fluid, one requires

$$e_{ij}\, C^{ijmn}\, e_{mn} \geq 0. \tag{9.27}$$

Since the strain tensor e_{ij} is symmetric, this constraint can be satisfied if

$$C^{ijmn} \;=\; C^{mnij}. \tag{9.28}$$

This constraint brings the number of degrees of freedom in the viscosity tensor down to $21 = 6 + 5 + 4 + 3 + 2 + 1$, which is the number of degrees of freedom in a symmetric 6×6 matrix.

9.3.4 Transverse isotropy

The presence of gravity provides a symmetry breaking from three-dimensional isotropy down to transverse isotropy about the local vertical direction $\hat{z}$. It is important that the stress tensor also respect this symmetry, which in turn provides constraints on the form of the viscosity tensor.

In order to understand what transverse, or axial, isotropy imposes on the viscosity tensor, it is necessary to recall that a fourth order tensor transforms under a change of coordinates in the following manner

$$C^{\bar{a}\bar{b}\bar{c}\bar{d}} = \Lambda^{\bar{a}}_{a}\Lambda^{\bar{b}}_{b}\Lambda^{\bar{c}}_{c}\Lambda^{\bar{d}}_{d}C^{abcd}. \tag{9.29}$$

Transverse isotropy means two things. First, the physical system remains invariant under arbitrary rotations about the $\hat{e}_3$ direction, where $\hat{e}_3 = \hat{z}$ is the vertical direction. Second, the physical system remains invariant under the transformation $z \to -z$, and $x \to y, y \to x$, which is a transformation between two right handed coordinate systems, with the vertical pointing up and down, respectively. The transformation matrix for the rotational symmetry takes the form of a rotation about the vertical

$$\Lambda^{\bar{a}}_{a} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{9.30}$$

and the transformation matrix between right handed systems takes the form

$$\Lambda^{\bar{a}}_{a} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}. \tag{9.31}$$

Imposing the constraint that

$$\begin{aligned} C^{\bar{a}\bar{b}\bar{c}\bar{d}} &\equiv C^{abcd} \\ &= \Lambda^{\bar{a}}_{a}\Lambda^{\bar{b}}_{b}\Lambda^{\bar{c}}_{c}\Lambda^{\bar{d}}_{d}C^{abcd}, \end{aligned} \tag{9.32}$$

where $\Lambda^{\bar{a}}_{a}$ is one of the given transformation matrices, provides for relations between the 21 remaining elements of C^{abcd}.

To determine the relations between the elements of C^{abcd} requires no more than careful enumeration of the possibilities. For example, with a rotation angle of $\pi/2$ about $\hat{z}$, rotational symmetry implies

$$C^{\bar{1}\bar{2}\bar{2}\bar{2}} \equiv C^{1222} = -C^{2111}. \tag{9.33}$$

However, the transformation between the two right handed coordinate systems implies

$$C^{1222} = C^{2111}. \tag{9.34}$$

These two results are satisfied only if

$$C^{1222} = C^{2111} = 0. \tag{9.35}$$

For a rotation of $\pi/4$, isotropy implies

$$C^{1111} = (C^{1111} + C^{1122} + 2\,C^{1212})/2, \tag{9.36}$$

or

$$C^{1212} = (C^{1111} - C^{1122})/2. \tag{9.37}$$

Continuing in this manner implies that the only nonzero elements of C^{ijmn} are C^{ijij}, where $i \neq j$, and C^{iijj}. The relation between these nonzero elements can be written

$$C^{iijj} \equiv c^{ij} = \begin{pmatrix} c^{11} & c^{12} & c^{13} \\ c^{12} & c^{11} & c^{13} \\ c^{13} & c^{13} & c^{33} \end{pmatrix}, \tag{9.38}$$

and

$$C^{1212} = (c^{11} - c^{12})/2 \tag{9.39}$$

$$C^{2323} = C^{1313} = c^{44}/2, \tag{9.40}$$

which brings to five the total number of independent degrees of freedom.

9.3.5 Trace-free frictional stress

The frictional stress tensor under consideration here is a deviatoric stress tensor (e.g., Smagorinsky 1993, Salmon 1998), which is defined to have a zero trace

$$\delta_{ik}\,\tau^{ik} = \tau^{ii} = 0. \tag{9.41}$$

Consequently,

$$C^{iimn}\,e_{mn} = (C^{1111} + C^{1122} + C^{3311})\,(e_{11} + e_{22}) + C^{3333}\,e_{33}. \tag{9.42}$$

Since MOM assumes an incompressible fluid, the trace of the strain or deformation tensor also vanishes

$$e_{mm} = u_{m,m} = 0. \tag{9.43}$$

As such, a trace-free frictional stress tensor implies the following relation between the viscosity tensor elements

$$C^{3333} = C^{1111} + C^{1122} - C^{3311}, \tag{9.44}$$

or

$$c^{33} = c^{11} + c^{12} - c^{13}. \tag{9.45}$$

9.3.6 Summary of the frictional stress tensor

In summary, the viscous stress tensor is given by

$$\tau^{mn} = \rho \begin{pmatrix} e_{11}\,(c^{11} - c^{13}) + e_{22}\,(c^{12} - c^{13}) & e_{12}\,(c^{11} - c^{12}) & e_{13}\,c^{44}/2 \\ e_{12}\,(c^{11} - c^{12}) & e_{11}\,(c^{12} - c^{23}) + e_{22}\,(c^{11} - c^{13}) & e_{23}\,c^{44} \\ e_{13}\,c^{44} & e_{23}\,c^{44} & e_{33}\,(c^{33} - c^{13}) \end{pmatrix}. \tag{9.46}$$

Motivated by Wajsowicz (1993) and Smagorinsky (1993), define the kinematic viscosity coefficients

$$\nu = 3\alpha = (c^{11} + c^{12})/2 - c^{13}, \tag{9.47}$$

$$A = \beta = (c^{11} - c^{12})/2 \tag{9.48}$$

$$\kappa = \gamma = c^{44}/2, \tag{9.49}$$

where ν, A, κ is the notation used in Wajsowicz (1993), and α, β, γ is the notation used in Smagorinsky (1993). The stress tensor components now take the form

$$\tau^{mn} = \rho \begin{pmatrix} (A+\nu)\, e_{11} + (\nu - A)\, e_{22} & 2\, A\, e_{12} & 2\, \kappa\, e_{13} \\ 2\, A\, e_{12} & (\nu - A)\, e_{11} + (\nu + A)\, e_{22} & 2\, \kappa\, e_{23} \\ 2\, \kappa\, e_{13} & 2\, \kappa\, e_{23} & 2\, \nu\, e_{33} \end{pmatrix} \tag{9.50}$$

which exposes a total of three viscous degrees of freedom.

9.3.7 Quasi-hydrostatic assumption

As MOM is designed for large-scale ocean modeling, it is a good approximation to assume motions maintain the hydrostatic balance. So far as the stress tensor is concerned, this assumption boils down to setting the viscosity coefficient ν to zero (Smagorinsky 1993),

$$\nu = 0. \tag{9.51}$$

It also amounts to approximating the following off-diagonal strains as

$$2\, e_{13} \approx u_{1\,,3} \tag{9.52}$$

$$2\, e_{23} \approx u_{2\,,3}. \tag{9.53}$$

The resulting stress tensor is given by

$$\tau^{mn} = \rho \begin{pmatrix} A\,(e_{11} - e_{22}) & 2\, A\, e_{12} & 2\, \kappa\, e_{13} \\ 2\, A\, e_{12} & A\,(e_{22} - e_{11}) & 2\, \kappa\, e_{23} \\ 2\, \kappa\, e_{13} & 2\, \kappa\, e_{23} & 0 \end{pmatrix}, \tag{9.54}$$

$$= \rho \begin{pmatrix} A\,(u_{1,1} - u_{2,2}) & A\,(u_{1,2} + u_{2,1}) & \kappa\, u_{1,3} \\ A\,(u_{1,2} + u_{2,1}) & A\,(u_{2,2} - u_{1,1}) & \kappa\, u_{2,3} \\ \kappa\, u_{1,3} & \kappa\, u_{2,3} & 0 \end{pmatrix}, \tag{9.55}$$

which exposes the familiar two viscous degrees of freedom. The scales

$$A >> \kappa \geq 0 \tag{9.56}$$

are relevant for large-scale stratified GFD flows. Generalizations for non-hydrostatic applications are given in Williams (1972).

9.3.8 Cartesian form of the friction vector

The friction vector in Cartesian coordinates is given by the divergence of the frictional stress tensor

$$\rho\, F^m = \tau^{mn}_{\,,n}. \tag{9.57}$$

Performing the divergence leads to the components

$$\rho\, F^1 = \nabla_h \cdot (\rho\, A\, \nabla_h\, u_1) + \hat{z} \cdot \nabla_h\, u_2 \wedge \nabla_h\, \rho\, A + [\rho\, \kappa\, (u_{1,3})]_{,z} \tag{9.58}$$

$$\rho\, F^2 = \nabla_h \cdot (\rho\, A\, \nabla_h\, u_2) - \hat{z} \cdot \nabla_h\, u_1 \wedge \nabla_h\, \rho\, A + [\rho\, \kappa\, (u_{2,3})]_{,z} \tag{9.59}$$

$$\rho\, F^3 = 0. \tag{9.60}$$

In these expressions, the horizontal divergence operator $\nabla_h = (\partial_1, \partial_2, 0)$ was introduced, and $z = \xi^3$ is the vertical coordinate. The factors of density cancel out trivially upon making the Boussinesq approximation. Note that when making the quasi-hydrostatic approximation, the vertical friction F^3 is set to zero so that the vertical momentum equation reduces to the inviscid hydrostatic equation. The extra cross-product terms appearing in the transverse friction vanish when using a constant viscosity. Their importance when using a spatially nonconstant viscosity is briefly highlighted in the next section.

9.3.9 The case of nonconstant viscosity

It is not uncommon for ocean modelers to employ a nonconstant viscosity for various numerical reasons. As emphasized by Wajsowicz (1993), some implementations of the corresponding friction vector often ignore the importance of formulating friction as the divergence of a symmetric stress tensor. Namely, what is sometimes done is to simply take the friction appropriate for a constant viscosity for a Boussinesq fluid

$$F^1_{const} = \nabla_h \cdot (A \, \nabla_h \, u_1) + [\kappa \, (u_{1,3})]_{,z} \tag{9.61}$$

$$F^2_{const} = \nabla_h \cdot (A \, \nabla_h \, u_2), + [\kappa \, (u_{2,3})]_{,z} \tag{9.62}$$

and then letting A be nonconstant. That is, the cross-product terms derived above are dropped. Focusing on the two-dimensional transverse sub-space, doing so amounts to employing the non-symmetric stress tensor

$$\tau^{mn}_{NS} = \rho_o \, A \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix}. \tag{9.63}$$

It is easy to show that the chosen friction dissipates kinetic energy since it is written as a Laplacian. However, for a fluid in uniform rotation

$$\mathbf{u} = \blacksquare \wedge \mathbf{x}, \tag{9.64}$$

where $\mathbf{u} = (u_1, u_2, 0)$, $\mathbf{x} = (x_1, x_2, 0)$, and $\blacksquare$ is spatially constant, the horizontal friction vector takes the form

$$\mathbf{F}_h = -\nabla \wedge (A \, \blacksquare), \tag{9.65}$$

and it vanishes only when A is a constant. As such, by using friction derived from a non-symmetric stress tensor and with a non-constant viscosity, a uniformly rotating fluid will feel a nonzero stress. Conversely, such a stress tensor can introduce uniform rotation; i.e., it can act as an internal source or sink of angular momentum. Unless one has a physical reason for doing so, such viscosity dependent sources of angular momentum should be avoided.

9.4 Orthogonal curvilinear coordinates

The purpose of this section is to derive the stress tensor and the corresponding friction vector in orthogonal curvilinear coordinates. The derivation requires a fair amount of calculations using curvilinear tensors. Since there are ocean models running with general curvilinear coordinates, the following derivation for general coordinates may be of use for understanding the form of friction used in those models.

9.4.1 Some rules of tensor analysis on manifolds

For many pragmatic situations, the rules of tensor analysis can be thought of as a systematic way to apply the chain rule on curved manifolds. More fundamentally, tensor analysis provides a general formalism which efficiently exploits the linkage between analysis and geometry. In turn, it can render a deeper and more concise description of physical laws without being diverted by often cumbersome coordinate dependent statements.

One of the key reasons that tensor analysis is so useful in physics is that an equation written in a form which respects a basic set of tensor rules remains form invariant under changes in coordinates. Consequently, one can work within a simple set of coordinates, such as Cartesian, in order to establish results which are then easily generalizable to other coordinates. This result allows for much of the discussion in this chapter to employ Cartesian tensors, as in the work of Smagorinsky (1993) and Wajsowicz (1993). However, to facilitate the eventual transition to curvilinear coordinates, the approach taken here is to employ the notation of curvilinear tensor analysis (e.g., Aris 1962).

The purpose of this section is to summarize salient aspects of tensor analysis. Use of the following rules and ideas will prove sufficient.

- **Conservation of indices**: Lower and upper tensor indices are balanced across equal signs.

- **Einstein summation convention**: Repeated indices are summed, unless otherwise noted.

- **Metric tensor**: The metric tensor provides a means to measure the distance between two points on a manifold:

$$(ds)^2 \quad = \quad g_{mn}\, d\xi^m\, d\xi^n. \tag{9.66}$$

  In this expression, $(ds)^2$, often written ds^2, is the squared infinitesimal arc-length between the points, ξ^m is the coordinate for a point, and $m = 1, 2, 3$ labels the coordinate (m is *not* a power).

  The metric for spherical coordinates on a sphere is diagonal. With coordinates $(\xi^1, \xi^2, \xi^3) = (\lambda, \phi, r)$, where λ is longitude and ϕ latitude, the metric is

$$g_{mn} = \mathrm{diag}(g_{\lambda\lambda}, g_{\phi\phi}, g_{rr}) = \mathrm{diag}(r^2 \cos^2\phi, r^2, 1). \tag{9.67}$$

  The inverse metric components g^{mn} will also be needed, and they are given by

$$g^{mn} = \mathrm{diag}((r\,\cos\phi)^{-2}, r^{-2}, 1). \tag{9.68}$$

  In Cartesian coordinates, the metric tensor is given by

$$g_{mn} = \delta_{mn} = \delta^{mn} = \delta^m_n, \tag{9.69}$$

  where δ is the unit or Kronecker delta tensor. There is no distinction between raised and lowered indices in Cartesian coordinates, hence the ability to jettison the conservation of indices rule. For curvilinear coordinates, however, this rule is essential.

- **Covariant and contravariant**: A lower label is often termed "covariant" and an upper label "contravariant." The mnemonic "co-low" assists in remembering the terminology.

Modern tensor language jettisons this terminology, yet it will be sufficient for the following.

Covariant and contravariant tensors can be considered dual, where the connection is through the metric tensor. For example, the covariant components to the velocity vector u_m are related to the contravariant components through

$$u_m = g_{mn} u^n. \tag{9.70}$$

Some examples are useful. In Cartesian coordinates, the velocity vector takes the familiar form

$$(u^1, u^2, u^3) = (u_1, u_2, u_3) = \left(\frac{Dx}{Dt}, \frac{Dy}{Dt}, \frac{Dz}{Dt}\right), \tag{9.71}$$

where again there is no distinction between covariant and contravariant for Cartesian tensors. In spherical coordinates, however,

$$(u^1, u^2, u^3) = \left(\frac{D\lambda}{Dt}, \frac{D\phi}{Dt}, \frac{Dr}{Dt}\right), \tag{9.72}$$

whereas the covariant components are

$$(u_1, u_2, u_3) = \left((r\cos\phi)^2 \frac{D\lambda}{Dt}, r^2 \frac{D\phi}{Dt}, \frac{Dr}{Dt}\right). \tag{9.73}$$

- **Notation**: As the above indicates, for curvilinear tensor analysis the difference between a raised and lowered label is important. Additionally, in order to avoid confusion, partial derivatives will be denoted with a comma:

$$u_{m\,,n} = \frac{\partial u_m}{\partial \xi^n}. \tag{9.74}$$

- **Covariant derivative**: In order to account for nonconstant unit vectors on a curved manifold, it is necessary to generalize partial derivatives to so-called covariant derivatives. In particular, the strain tensor (described later) has components

$$e_{mn} = \frac{1}{2}(u_{m\,;n} + u_{n\,;m}), \tag{9.75}$$

where the comma has been generalized to a semi-colon.

For a "torsionless" manifold, such as a sphere, each component of the metric tensor has a vanishing covariant derivative

$$g_{mn;p} = 0. \tag{9.76}$$

This is a trivial property for Cartesian coordinates on a plane, in which case the metric is the constant unit tensor and the covariant derivative a partial derivative

$$\delta_{mn;p} = \delta_{mn,p} = 0. \tag{9.77}$$

However, for curvilinear coordinates $g_{mn;p} = 0$ is quite useful. For example, it provides for the convenient relation

$$\begin{aligned} u_{m;n} &= (g_{mp}\, u^p)_{;n} \\ &= g_{mp}\, u^p_{;n}. \end{aligned} \tag{9.78}$$

This result brings the strain tensor to the form

$$2\,e_{mn} \;=\; g_{mp}\,u^{p}_{;n} + g_{np}\,u^{p}_{;m}. \tag{9.79}$$

In general, the covariant derivative of a vector on a torsionless manifold is given by

$$u^{p}_{;n} \;=\; u^{p}_{,n} + \Gamma^{p}_{mn}\,u^{n}, \tag{9.80}$$

where Γ^{p}_{mn} are components to the Christoffel symbol, which is given by

$$\Gamma^{p}_{mn} \;=\; \frac{1}{2}\,g^{pq}\,(g_{qm,n} + g_{qn,m} - g_{mn,q}). \tag{9.81}$$

A more geometric means of understanding the Christoffel symbol is to note that they form the expansion coefficients of the partial derivative of the basis vectors for a manifold

$$\vec{e}_{a,b} = \Gamma^{m}_{ab}\,\vec{e}_{m}. \tag{9.82}$$

That is, the Christoffel symbol accounts for the nonzero changes in the basis vectors on a curved manifold. Note that it is symmetric on the lower two labels:

$$\Gamma^{p}_{mn} = \Gamma^{p}_{nm}, \tag{9.83}$$

which is the defining property of torsionless manifolds.

- **Transformation rules**: Under a coordinate transformation

$$\xi^{\overline{m}} = \xi^{\overline{m}}(\xi^{m}), \tag{9.84}$$

tensors transform as, for example,

$$e_{\overline{mn}} \;=\; \Lambda^{m}_{\overline{m}}\,\Lambda^{n}_{\overline{n}}\,e_{mn}, \tag{9.85}$$

where the transformation matrix is given by the partial derivatives

$$\Lambda^{m}_{\overline{m}} \;=\; \frac{\partial \xi^{m}}{\partial \xi^{\overline{m}}}. \tag{9.86}$$

Sometimes it is useful to write the transformation matrix in traditional matrix form. The convention is that the index which is placed a bit closer to the Λ denotes the row (m in $\Lambda^{m}_{\overline{m}}$), and the one pushed away a bit is the column ($\overline{m}$ in $\Lambda^{m}_{\overline{m}}$). The inverse transformation of a tensor takes the form

$$e_{mn} \;=\; \Lambda^{\overline{m}}_{m}\,\Lambda^{\overline{n}}_{n}\,e_{\overline{mn}}, \tag{9.87}$$

where the inverse transformation matrix is given by

$$\Lambda^{\overline{m}}_{m} \;=\; \frac{\partial \xi^{\overline{m}}}{\partial \xi^{m}}. \tag{9.88}$$

Generalizations to tensors of different order follow analogously.

9.4.2 Orthogonal coordinates

The metric tensor for orthogonal coordinates is diagonal

$$g_{ij} = \mathrm{diag}(g_{11}, g_{22}, g_{33}), \tag{9.89}$$

where the components $g_{ij} = g_{ij}(t, \xi^1, \xi^2, \xi^3)$ are generally functions of space-time. The infinitesimal arc-length measuring the distance between any two closely spaced points is therefore given by the diagonal quadratic-form

$$\begin{aligned} ds^2 &= g_{11}(d\xi^1)^2 + g_{22}(d\xi^2)^2 + g_{33}(d\xi^3)^2 \\ &= (h_1\, d\xi^1)^2 + (h_2\, d\xi^2)^2 + (h_3\, d\xi^3)^2, \end{aligned} \tag{9.90}$$

where the metric functions $g_{mm} = (h_m)^2$, with no sum, are often useful to introduce. Additionally, the relation between covariant and contravariant components of a tensor is given through a single multiplication. For example,

$$\begin{aligned} u_m &= g_{mn}\, u^n \\ &= g_{mm}\, u^m, \end{aligned} \tag{9.91}$$

relates the covariant velocity components u_m to the contravariant components u^m. Importantly, there is no sum on the m label in the last expression.

For the purposes of large-scale ocean modeling, it is usually sufficient to assume the simpler form of the metric

$$g_{ij} = \mathrm{diag}(g_{11}, g_{22}, 1), \tag{9.92}$$

where the nontrivial metric components are independent of time. This assumption follows from the quasi-hydrostatic approximation and will be made in the following.

In the following, the determinant of the metric tensor

$$\mathcal{G} = g_{11}\, g_{22}\, g_{33} \tag{9.93}$$

will appear quite frequently. With the quasi-hydrostatic approximation for which $g_{33} = 1$, the determinant is given by

$$\mathcal{G} = g_{11}\, g_{22} = (h_1\, h_2)^2. \tag{9.94}$$

9.4.3 Physical components of tensors

In many applications, it is useful to introduce the *physical components* of a tensor (see Section 7.4 of Aris or 4.8 of Weinberg). For example, the velocity field using spherical coordinates is often written

$$\begin{aligned} (u, v, w) &= \left(\sqrt{g_{\lambda\lambda}}\, u^\lambda, \sqrt{g_{\phi\phi}}\, u^\phi, \sqrt{g_{rr}}\, u^r\right) \\ &= \left(r\cos\phi\, \frac{D\lambda}{Dt}, r\, \frac{D\phi}{Dt}, \frac{Dr}{Dt}\right). \end{aligned} \tag{9.95}$$

Additionally, the infinitesimal displacements along the coordinate directions on the sphere are given by

$$(\delta x, \delta y, \delta z) = ((r\cos\phi)\, \delta\lambda, r\, \delta\phi, \delta r). \tag{9.96}$$

More generally, for any orthogonal coordinate system, the physical components of the displacement will be written

$$(\delta x, \delta y, \delta z) \quad = \quad (h_1\, \delta\xi^1,\, h_2\, \delta\xi^2,\, h_3\, \delta\xi^3) \tag{9.97}$$

Likewise, the physical components of the velocity are written

$$(u, v, w) \quad = \quad (h_1\, u^1,\, h_2\, u^2,\, h_3\, u^3). \tag{9.98}$$

As such, for example,

$$u^1_{,1} = \sqrt{g_{11}}\,(u/\sqrt{g_{11}})_{,x} = h_1\,(u/h_1)_{,x} \tag{9.99}$$

Note that the traditional Cartesian notation x, y, z is used for convenience; the coordinates are generally curvilinear.

The key property of the physical components of a tensor is that each has the same dimensions; e.g., length for the physical displacement components, length/time for the physical velocity components, etc. Importantly, the physical components are *not* components to a true tensor since the tensorial transformation rules are corrupted by the square root of the metric. Correspondingly, the physical components of the partial derivative operator do not necessarily commute; i.e., $\partial_x\,\partial_y = h_1^{-1}\partial_1\, h_2^{-1}\partial_2$ equals $\partial_y\,\partial_x$ only for a constant metric. Hence, it is best to perform mathematical manipulations with the tensor quantities, and only after establishing the final result should the physical components be introduced before discretizing. This is the approach taken in the following.

9.4.4 General form of the frictional stress tensor

In Cartesian coordinates, the stress tensor given in equation (9.54) can be written as the sum of two tensors, each defined on orthogonal subspaces

$$\tau^{ij} \quad = \quad \tau^{ij}_{tran} + \tau^{ij}_{vert}, \tag{9.100}$$

where

$$\begin{aligned} \tau^{ij}_{tran} &= \rho\, A\,(g^{ik}\, g^{jl} + g^{il}\, g^{jk} - g^{ij}\, g^{kl})\, e_{kl} \\ &= \rho\, A\,(2\, e^{ij} - g^{ij}\, e^k_k) \end{aligned} \tag{9.101}$$

is the transverse stress tensor, defined over the transverse coordinates $i, j, k, l = 1, 2$ and set to zero if one of the indices is 3.

$$\tau^{ij}_{vert} \quad = \quad 2\,\rho\,\kappa\, e^{i3}\, g^{j3} \tag{9.102}$$

is the vertical stress tensor, where $i = 1, 2$ and $\tau^{ji}_{vert} = \tau^{ij}_{vert}$. In these expressions, g^{ij} are the components to the inverse metric tensor, which is the Kronecker delta in Cartesian coordinates. The transformation to curvilinear coordinates of interest here maintains orthogonality of the coordinates and transverse isotropy about the third direction. Such a coordinate transformation maintains the form of the stress tensor given here, where the metric tensor is now generally nontrivial, and the strain tensor is computed using covariant derivatives as described in the next section.

It is notable that such a form for the stress tensor could have been "guessed" given the three constraints: (A) symmetry $\tau^{mn} = \tau^{nm}$, (B) separately trace-free in the two lateral directions and in the vertical direction; $\tau^1_1 + \tau^2_2 = 0 = \tau^3_3$, and (C) laterally isotropic. The previous analysis of the viscosity tensor, although more tedious than starting from these three assumptions, exposed more of the underlying properties of the stress tensor.

9.4.5 Horizontal tension and shearing rate of strain

Given the form for the stress tensor in equation (9.100), it is useful to follow Smagorinsky (1993) by introducing the *horizontal tension* D_T

$$\begin{aligned}
D_T &= e^1_1 - e^2_2 \\
&= u^1_{;1} - u^2_{;2} \\
&= u^1_{,1} - u^2_{,2} + u^m\,(\Gamma^1_{1m} - \Gamma^2_{2m}) \\
&= u^1_{,1} - u^2_{,2} + \frac{1}{2}\,u^m\,\partial_m\,\ln(g_{11}/g_{22}) \\
&= \sqrt{\frac{g_{22}}{g_{11}}}\left(u^1\sqrt{\frac{g_{11}}{g_{22}}}\right)_{,1} - \sqrt{\frac{g_{11}}{g_{22}}}\left(u^2\sqrt{\frac{g_{22}}{g_{11}}}\right)_{,2} + u^3\,\partial_3 \ln\sqrt{\frac{g_{11}}{g_{22}}} \\
&= \sqrt{g_{22}}\,(u/\sqrt{g_{22}})_{,x} - \sqrt{g_{11}}\,(v/\sqrt{g_{11}})_{,y},
\end{aligned} \tag{9.103}$$

where the last step wrote D_T in terms of the physical velocity and differential components, and the ∂_z term was dropped based on assuming the metric components are independent of depth as implied by the quasi-hydrostatic approximation (Section 9.3.7). In Cartesian coordinates, $D_T = u_{,x} - v_{,y}$. It is also useful to introduce the *horizontal shearing strain* D_S which is given by

$$\begin{aligned}
D_S &= 2\sqrt{\mathcal{G}}\,e^{12} \\
&= 2\sqrt{\mathcal{G}}\,g^{11}\,e^2_1.
\end{aligned} \tag{9.104}$$

A bit of work yields

$$\begin{aligned}
2\,e^2_1 &= u^2_{;1} + g^{22}\,g_{11}\,u^1_{;2} \\
&= u^2_{,1} + \Gamma^2_{1m}\,u^m + g^{22}\,g_{11}\,(u^1_{,2} + \Gamma^1_{2m}\,u^m) \\
&= u^2_{,1} + \frac{1}{2}\,g^{2d}\,(g_{1d,m} + g_{md,1} - g_{1m,d})\,u^m + g^{22}\,g_{11}\,u^1_{,2} \\
&+ \frac{1}{2}\,g^{22}\,g_{11}\,g^{1d}\,(g_{2d,m} + g_{md,2} - g_{2m,d})\,u^m \\
&= g^{22}\,(g_{11}\,u^1_{,2} + g_{22}\,u^2_{,1}),
\end{aligned} \tag{9.105}$$

which brings the horizontal shearing strain to

$$\begin{aligned}
D_S &= \mathcal{G}^{-1/2}\,(g_{11}\,u^1_{,2} + g_{22}\,u^2_{,1}) \\
&= \sqrt{g_{11}}\,(u/\sqrt{g_{11}})_{,y} + \sqrt{g_{22}}\,(v/\sqrt{g_{22}})_{,x},
\end{aligned} \tag{9.106}$$

where the last step introduced the physical components. In Cartesian coordinates, $D_S = u_{,y} + v_{,x}$. A similar calculation yields the strain component e^3_1

$$\begin{aligned}
2\,e^3_1 &= u^3_{;1} + g^{33}\,g_{11}\,u^1_{;3} \\
&= g^{33}\,(g_{11}\,u^1_{,3} + g_{33}\,u^3_{,1}) \\
&= g_{11}\,u^1_{,3} + u^3_{,1} \\
&\approx g_{11}\,u^1_{,3} \\
&= \sqrt{g_{11}}\,u_{,z},
\end{aligned} \tag{9.107}$$

where the last two steps follow from the quasi-hydrostatic approximation. In summary, these results bring the quasi-hydrostatic stress tensor to the form

$$\tau^{ij} = \rho \begin{pmatrix} A\, g^{11}\, D_T & A\, D_S/\sqrt{\mathcal{G}} & \kappa\,(u/\sqrt{g_{11}})_{,z} \\ A\, D_S/\sqrt{\mathcal{G}} & -A\, g^{22}\, D_T & \kappa\,(v/\sqrt{g_{22}})_{,z} \\ \kappa\,(u/\sqrt{g_{11}})_{,z} & \kappa\,(v/\sqrt{g_{22}})_{,z} & 0 \end{pmatrix}, \tag{9.108}$$

where again $g_{11,z} = g_{22,z} = 0$ was used.

9.4.6 The friction vector

The friction vector is given by the covariant divergence of the frictional stress tensor

$$\begin{aligned} \rho\, F^m &= \tau^{mn}_{;n} \\ &= \tau^{mn}_{,n} + \Gamma^m_{nc}\, \tau^{nc} + \Gamma^n_{nc}\, \tau^{mc}, \end{aligned} \tag{9.109}$$

where the covariant derivative of the second order stress tensor is a straightforward generalization of the result for a vector. Recall that in the quasi-hydrostatic limit, only the friction in the transverse directions is of interest, since the vertical momentum equation reduces to the inviscid hydrostatic equation.

The expression (9.109) is quite general. For the purpose of representing such friction in an ocean model, it is necessary to make this result a bit more explicit. For this purpose, it is useful to start from the equivalent expression

$$\rho\, F^m = (\sqrt{\mathcal{G}})^{-1} \left(\sqrt{\mathcal{G}}\, \tau^{mn}\right)_{,n} + \Gamma^m_{ab}\, \tau^{ab}. \tag{9.110}$$

This expression is valid for any metric. Its derivation is omitted here.

To proceed, employ the expression (9.81) for the Christoffel symbol written in terms of the metric, the expression (9.100) for the stress tensor, and the diagonal form of the metric tensor. First, the contraction $\Gamma^n_{ab}\, \tau^{ab}$ is given by

$$\begin{aligned} \Gamma^m_{ab}\, \tau^{ab} &= \frac{1}{2}\, g^{md}\, (g_{ad,b} + g_{bd,a} - g_{ab,d})\, \tau^{ab} \\ &= g^{mm}\, (g_{am,b} - \frac{1}{2}\, g_{ab,m})\, \tau^{ab} \\ &= g^{mm}\, g_{mm,b}\, \tau^{mb} - \frac{1}{2}\, g^{mm}\, g_{ab,m}\, \tau^{ab} \end{aligned} \tag{9.111}$$

where there is no sum on the m label. Plugging this result into the expression (9.110) for the friction vector yields

$$\rho\, F^m = (g_{mm}\, \sqrt{\mathcal{G}})^{-1} \left(g_{mm}\, \sqrt{\mathcal{G}}\, \tau^{mn}\right)_{,n} - \frac{1}{2}\, g^{mm}\, g_{ab,m}\, \tau^{ab}. \tag{9.112}$$

Now recall that $g_{33} = 1$ and $\tau^1_1 = -\tau^2_2$. Consequently, for $m = 1$ the friction is

$$\begin{aligned} \rho\, F^1 &= (g_{11}\, \sqrt{\mathcal{G}})^{-1} \left(g_{11}\, \sqrt{\mathcal{G}}\, \tau^{1n}\right)_{,n} - \frac{1}{2}\, g^{11}\, g_{ab,1}\, \tau^{ab} \\ &= \tau^{1n}_{,n} + \tau^{1n}\, \partial_n \ln(g_{11}\, \sqrt{\mathcal{G}}) - \tau^{11}\, \partial_1 \ln \sqrt{g_{11}} - \frac{1}{2}\, \tau^{22}\, g^{11}\, g_{22,1} \end{aligned}$$

$$
\begin{aligned}
&= \tau^{11}_{,1} + \tau^{11}\,\partial_1 \ln(g_{11}\sqrt{\mathcal{G}}) + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{12})_{,2} + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{13})_{,3} \\
&- \tau^{11}\,\partial_1 \ln\sqrt{g_{11}} + \tau^{11}\,\partial_1 \ln\sqrt{g_{22}} \\
&= \tau^{11}_{,1} + \tau^{11}\,\partial_1 \ln \mathcal{G} + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{12})_{,2} + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{13})_{,3} \\
&= \mathcal{G}^{-1}(\mathcal{G}\,\tau^{11})_{,1} + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{12})_{,2} + (g_{11}\sqrt{\mathcal{G}})^{-1}(g_{11}\sqrt{\mathcal{G}}\,\tau^{13})_{,3} \\
&= \frac{\sqrt{g_{11}}}{\mathcal{G}}(g_{22}\,\rho\,A\,D_T)_{,x} + \frac{1}{g_{11}\sqrt{g_{11}}}(g_{11}\,\rho\,A\,D_S)_{,y} + g_{11}^{-1/2}(\rho\,\kappa\,u_{,z})_{,z} \qquad (9.113)
\end{aligned}
$$

where the last step introduced the physical components, and the depth independence of the metric components has been used. Multiplying by $\sqrt{g_{11}}$ determines the physical component to the generalized zonal friction

$$
\rho\,F^x = g_{22}^{-1}(g_{22}\,\rho\,A\,D_T)_{,x} + g_{11}^{-1}(g_{11}\,\rho\,A\,D_S)_{,y} + (\kappa\,u_{,z})_{,z}. \qquad (9.114)
$$

Similar considerations lead to the second friction component

$$
\rho\,F^2 = \mathcal{G}^{-1}(\mathcal{G}\,\tau^{22})_{,2} + (g_{22}\sqrt{\mathcal{G}})^{-1}(g_{22}\sqrt{\mathcal{G}}\,\tau^{21})_{,1} + (g_{22}\sqrt{\mathcal{G}})^{-1}(g_{22}\sqrt{\mathcal{G}}\,\tau^{31})_{,3}. \qquad (9.115)
$$

Multiplying by $\sqrt{g_{22}}$ leads to the generalized meridional friction component

$$
\rho\,F^y = -g_{11}^{-1}(g_{11}\,\rho\,A\,D_T)_{,y} + g_{22}^{-1}(g_{22}\,\rho\,A\,D_S)_{,x} + (\rho\,\kappa\,v_{,z})_{,z}. \qquad (9.116)
$$

Again, for Boussinesq fluids, the factors of density can be canceled on both sides, since each are formally replaced by ρ_o. For non-Boussinesq fluids, the cancelation is also often performed, since the values of the kinematic viscosities are not precisely known.

9.4.7 Effects on kinetic energy

Although it has been built into the formalism, it is useful to explicitly show that the friction dissipates horizontal kinetic energy. For this purpose, recall that the kinetic energy for a parcel of fluid is the scalar quantity

$$
\begin{aligned}
2\,K &= \rho\,dV\,u_m\,u^m \\
&= \rho\,dV\,g_{mn}\,u^n\,u^m, \qquad (9.117)
\end{aligned}
$$

where $m = 1, 2$ represents the label for the horizontal coordinates. The evolution of this energy is given by

$$
\dot{K} = dV\,g_{mn}\,u^n\,(\rho\,f^m + T^{mp}_{;p}), \qquad (9.118)
$$

where mass conservation and Newton's Law were employed. Consequently, friction contributes to the evolution of kinetic energy through the term

$$
dV\,g_{mn}\,u^n\,\tau^{mp}_{;p} = \rho\,dV\,g_{mn}\,u^n\,F^m. \qquad (9.119)
$$

The question then arises as to whether $\rho\,dV\,g_{mn}\,u^n\,F^m$ integrated over the horizontal extent of the domain is negative semi-definite, which would be the case for dissipative friction. First note that the term $u_m\,(\kappa\,u^m_{,z})_{,z}$ is not at issue here; it appears in the same form as for Cartesian coordinates and has well known dissipative properties. Some manipulations using previous results from this section yield

$$
\int dV\,u_m\,\tau^{mp}_{;p} = \int dV\,[(u_m\,\tau^{mp})_{;p} - \tau^{mp}\,u_{m;p}]. \qquad (9.120)
$$

The first term integrates to a boundary contribution, which vanishes with a no-slip and/or no normal stress boundary condition. Hence,

$$\begin{aligned}\int dV\, u_m\, \tau^{mp}_{;p} &= -1/2 \int dV\, \tau^{mp}\, (u_{m\,;p} + u_{p\,;m}) \\ &= -\int dV\, \tau^{mp}\, e_{mp}, \end{aligned} \tag{9.121}$$

where the strain tensor e_{mp} was introduced. Use of the expression (9.101) for the transverse stress tensor leads to

$$\begin{aligned}2\,\rho\, A\, \tau_{mp}\, e^{mp} &= \tau_{mp}\, (\tau^{mp} + A\, \rho\, g^{mp}\, e^q_q) \\ &= \tau_{mp}\, \tau^{mp}, \end{aligned} \tag{9.122}$$

where $g^{mp}\, \tau_{mp} = 0$ was used. Hence, the contribution to kinetic energy from horizontal friction takes the form

$$\begin{aligned}\int dV\, u_m\, \tau^{mp}_{;p} &= -\int dV (2\,\rho\, A)^{-1}\, \tau^{mp}\, \tau_{mp} \\ &= -\int \rho\, dV\, A\, (D_T^2 + D_S^2), \end{aligned} \tag{9.123}$$

which shows that the kinetic energy is indeed dissipated by the chosen form of the friction, so long as the viscosity is non-negative. Since the dissipation is the scalar

$$\tau^{mn}\, \tau_{mn} = 2(\rho\, A)^2\, (D_T^2 + D_S^2), \tag{9.124}$$

it is coordinate invariant. Again, note that the indices m, n extend only over the horizontal directions $m, n = 1, 2$.

It is convenient here to point out a connection between the rate of kinetic energy dissipation and the Smagorinsky formulation for viscosity. In the Smagorinsky (1963) scheme (Section 34.7), viscosity A is determined as a function of the total amount of horizontal strain in the flow, as well as the grid spacing. By "total amount of strain", Smagorinsky means the scalar quantity

$$\begin{aligned}D^2 &= 2\,(2\,\rho\, A)^{-2}\, \tau^{mn}\, \tau_{mn} \\ &= D_T^2 + D_S^2. \end{aligned} \tag{9.125}$$

That is, $|D|$ represents the total rate of horizontal strain for the resolved motions. As D is constructed as a scalar quantity, its value is the same in any set of horizontal curvilinear coordinates. Hence, from a mathematical and numerical perspective, D is a sensible quantity to use for constructing viscosity.

9.4.8 Summary of second order friction

The fundamental assumptions that determine the form of the momentum friction are the following:

- Horizontal kinetic energy is dissipated by the friction.
- The friction does not introduce interior sources or sinks of angular momentum.
- The fluid motion is quasi-hydrostatic.

- The friction exhibits lateral or transverse isotropy in which gravity picks out the only special direction.

Each of these properties is satisfied by the following two components to the friction

$$\boxed{\rho\, F^x = g_{22}^{-1}\,(g_{22}\,\rho\, A\, D_T)_{,x} + g_{11}^{-1}\,(g_{11}\,\rho\, A\, D_S)_{,y} + (\rho\,\kappa\, v_{,z})_{,z}} \tag{9.126}$$

$$\boxed{\rho\, F^y = -g_{11}^{-1}\,(g_{11}\,\rho\, A\, D_T)_{,y} + g_{22}^{-1}\,(g_{22}\,\rho\, A\, D_S)_{,x} + (\rho\,\kappa\, v_{,z})_{,z}} \tag{9.127}$$

With the Boussinesq approximation, the factors of density are formally replaced by the constant ρ_o, and so cancel out from these expressions. The metric tensor is assumed to be diagonal and to define the infinitesimal distance between two points as

$$\boxed{ds^2 = (h_1\, d\xi^1)^2 + (h_2\, d\xi^2)^2 + dz^2 = dx^2 + dy^2 + dz^2} \tag{9.128}$$

In this expression, the metric components $g_{11} = h_1^2$ and $g_{22} = h_2^2$ are functions only of the transverse coordinates, and the physical displacements

$$dx = h_1\, d\xi^1 \tag{9.129}$$

$$dy = h_2\, d\xi^2 \tag{9.130}$$

have dimensions of length. The corresponding physical components of the partial derivative operators

$$\partial_x = h_1^{-1}\,\partial_1 \tag{9.131}$$

$$\partial_y = h_y^{-1}\,\partial_2 \tag{9.132}$$

bring the horizontal tension to the form

$$\boxed{D_T = h_2\,(u/h_2\,)_{,x} - h_1\,(v/h_1\,)_{,y}} \tag{9.133}$$

and the horizontal shearing strain

$$\boxed{D_S = h_1\,(u/h_1\,)_{,y} + h_2\,(v/h_2\,)_{,x}} \tag{9.134}$$

D_T and D_S are generically called the deformation rates, and the each have dimensions of t^{-1}. In spherical coordinates, $(\xi^a, \xi^2) = (\lambda, \phi)$, $h_1 = a\cos\phi$, $h_2 = a$, and $\partial_x = (a\cos\phi)^{-1}\,\partial_\lambda$, $\partial_y = a^{-1}\,\partial_\phi$. The viscosity A is generally a function of the fluid flow, and it has dimension L^2/t. The generalized curvilinear coordinates x, y, z are physical components, and so each has dimension of length. Likewise, the corresponding physical velocity components (u, v, w) have dimension L/t.

9.5 Biharmonic friction

The previous derivations were all concerned with second order, or *Laplacian*, friction. It is often useful to consider another method of dissipating momentum through use of a fourth order, or *biharmonic*, friction. Such friction acts more strongly on the small scales than the Laplacian friction, and less strongly on the large scales (see Section 34.4 for more discussion). Each property is desirable, especially when aiming to realize some form of a quasi-geostrophic

turbulent cascade in which enstrophy cascades to the small scales and energy to the large scales. Biharmonic friction is therefore commonly used in ocean modeling. It should be noted, however, that biharmonic friction is motivated mostly from numerical reasons and has no first principle derivation.

The goal of this section is to derive the appropriate form of the biharmonic operator which both dissipates kinetic energy yet does not introduce spurious sources of angular momentum. As with the previous derivations, some work is necessary in order to realize these properties on a sphere with a generally non-constant viscosity.

Recall that the quasi-hydrostatic approximation allowed for the separation of the transverse or horizontal subspace from the vertical subspace. Consequently, the vertical term $(\kappa \mathbf{u}_{,z})_{,z}$ was isolated from the other terms in the friction vector. As a result, the following will focus solely on deriving the biharmonic operator for use in the two-dimensional transverse subspace.

9.5.1 General formulation

The general formulation of biharmonic friction is a straightforward extension of the Laplacian friction given in the previous sections. What is done is to basically iterate twice on the Laplacian approach. More precisely, the components F_B^i of the biharmonic friction vector are derived from the covariant divergence

$$\rho F_B^m = \Theta^{mn}_{;n}, \tag{9.135}$$

where

$$\Theta^{mn} = -\rho B (2 E^{mn} - g^{mn} E^p_p), \tag{9.136}$$

$B > 0$ has units of $L^2/t^{1/2}$, and ρ is set to ρ_o when making the Boussinesq approximation. As shown in the next subsection, use of the "square-root" biharmonic viscosity in prompted by the desire to dissipate kinetic energy. This detail only matters for cases with a non-constant viscosity. Note that all labels in this section run over $m, n, p = 1, 2$. Θ^{mn} has the same form as the stress tensor used for second order friction discussed in Section 9.4.4, except with a minus sign. The components of the symmetric "strain" tensor are given by

$$E_{mn} = \frac{1}{2}(F_{m;n} + F_{n;m}). \tag{9.137}$$

The vector F^m is the friction vector determined through the second order frictional stress tensor

$$\begin{aligned} \rho F^m &= \tau^{mn}_{;n} \\ &= [B \rho (2 e^{mn} - g^{mn} e^p_p)]_{;n} \end{aligned} \tag{9.138}$$

as derived in the previous sections, where the only difference is that the viscosity used for computing the stress tensor τ^{mn} is now set to B, and the dimensions on F^m are $L\, t^{-3/2}$.

This approach ensures that the biharmonic friction is derived from the divergence of a symmetric tensor Θ^{mn}, hence ensuring a proper angular momentum budget. Additionally, the computational work necessary to compute the Laplacian friction is directly employed for the biharmonic friction. Finally, as shown in the next subsection, this form for biharmonic friction also dissipates kinetic energy.

9.5.2 Effects on kinetic energy

The manipulations necessary to show that the biharmonic friction dissipates kinetic energy are analogous to those used for second order friction in Section 9.4.8. As with that discussion, the relevant contribution from horizontal biharmonic friction is given by $dV\, u_m\, \Theta^{mn}_{;n}$. Assuming either no-slip or no-normal Θ stress at the boundaries brings this expression to the form

$$\begin{aligned} \int dV\, u_m\, \Theta^{mn}_{;n} &= -\int dV\, \Theta^{mn}\, e_{mn} \\ &= \int dV\, \rho\, B\, [2\, E^{mn}\, e_{mn} - e^n_n\, E^m_m]. \end{aligned} \tag{9.139}$$

For the product of traces, one has

$$\begin{aligned} \int dV\, \rho\, B\, e^n_n\, E^m_m &= \int dV\, \rho\, B\, e^n_n\, F^m_{;m} \\ &= \int dV\, [(\rho\, B\, e^n_n\, F^m)_{;m} - (\rho\, B\, e^n_n)_{;m}\, F^m]. \end{aligned} \tag{9.140}$$

The first term reduces to a boundary contribution, which will be assumed to vanish. For the contraction of the two strain tensors, one has

$$\begin{aligned} 2\int dV\, \rho\, B\, e^{mn}\, E_{mn} &= 2\int dV\, \rho\, B\, e^{mn}\, F_{m\,;n} \\ &= -2\int dV\, F_m\, (\rho\, B\, e^{mn})_{;n}, \end{aligned} \tag{9.141}$$

where the boundary term $(\rho\, B\, e^{mn}\, F_m)_{;n}$ was assumed to vanish. Combining the two contributions leads to

$$\begin{aligned} \int dV\, u_m\, \Theta^{mn}_{;n} &= -\int dV\, F_m\, [2\, B\, \rho\, e^{mn} - g^{mn}\, B\, \rho\, e^k_k]_{;n} \\ &= -\int dV\, \rho\, F_m\, F^m, \end{aligned} \tag{9.142}$$

which is non-positive.

If the viscosity B is distributed non-symmetrically, then the effects on kinetics energy are guaranteed to be dissipative only for the special case of constant viscosity. That is, in cartesian coordinates, the operator $\nabla_h \cdot B\, \nabla_h\, (\nabla_h \cdot B \nabla_h \psi)$ is dissipative for all $B > 0$, whereas $\nabla_h \cdot B\, \nabla_h\, (\nabla^2_h \psi)$ or $\nabla^2_h\, (\nabla_h \cdot B \nabla_h \psi)$ can be proven to be dissipative only for constant B. Until May 1999, this point was not recognized when implementing the biharmonic friction with non-constant viscosities in MOM.

9.6 Comments on frictional and advective metric terms

This section benefited from discussions with Bob Hallberg and Gavin Schmidt.

When discretizing physical processes in MOM, it is desirable to formulate these processes in terms of the finite difference of a flux across the faces of a grid cell. Hence, the forcing terms in the continuous equations should be in the form of a divergence of a flux. However, it is not possible to do so for the friction acting on the zonal and meridional momentum on a sphere. Likewise, it is not possible to do so for advection of zonal and meridional momentum, as mentioned in Section 4.2. The purpose of this section is to provide some discussion of these ideas.

As seen in Section 9.3.7, the hydrostatic approximation brings the vertical friction to the flux-form $(\kappa_m \mathbf{u}_{,z})_{,z}$, regardless of the details of the horizontal coordinates. For brevity, the following discussion will therefore omit the vertical portion of the friction and just focus on the lateral components.

9.6.1 Motion on an infinite plane

To get started, it is useful to consider fluid motion on an infinite flat plane. In the absence of external forces which act to make a particular horizontal direction special, the environment maintains translational symmetry in either of the horizontal directions. Hence, the total horizontal momentum in either direction is conserved. Mathematically, this result means that the momentum of a fluid parcel takes the form of a conservation equation. That is, the forces affecting the time tendency of this momentum are represented as a total divergence. These statements take their mathematical form as the time tendency for the momentum density

$$(\rho\, u^m)_{,t} \quad = \quad (T^{mn} - \rho\, u^m\, u^n)_{,n} + \rho\, f^m \tag{9.143}$$

where the tensor labels extend over the horizontal Cartesian coordinates x, y, and the conservation of mass was used in the form

$$\rho_{,t} + (\rho\, u^n)_{,n} = 0. \tag{9.144}$$

Additionally, the stress tensor has been written in the form

$$T^{mn} = \tau^{mn} - \delta^{mn}\, p, \tag{9.145}$$

which is the sum of the symmetric frictional stress tensor and diagonal pressure stress tensor. In the absence of external forces f^m, or in the case when these forces can be derived as the divergence of a scalar, the total horizontal momentum per unit volume $\int \rho\, u^m\, dV$ is a constant in time.

In addition to momentum in a particular direction, the discussion in Section 9.2.3 showed that so long as the stress tensor is symmetric and there is an absence of external forces, there is an angular momentum conservation law. For motion on the plane, this conservation law arises from symmetry of the unforced motion under rotations about the vertical axis. That is, angular momentum about the vertical direction is conserved in the absence of external forces or boundary effects. Mathematically, the conservation of angular momentum can be derived from the momentum equation in a similar manner to that used in Section 9.2.3. For completeness, the derivation is summarized. Recall that the angular momentum per unit volume is given by

$$\rho\, \mathcal{M}_m = \rho\, \epsilon_{mnp}\, x^n\, u^p. \tag{9.146}$$

Using the conservation of mass, the momentum equations, and symmetry of the stress tensor, it is straightforward to determine the conservation law

$$(\rho\, \mathcal{M}_m)_t + (\rho\, \mathcal{M}_m\, u^p)_{,p} \quad = \quad \epsilon_{mnp}\, [(x^n\, T^{pq})_{,q} + x^n\, \rho\, f^p]. \tag{9.147}$$

The first term on the right hand side takes the form of a total divergence, and the second term represents external torques. In the absence of external torques and boundary effects, $\int \rho\, \mathcal{M}_m\, dV$ is a constant in time.

9.6.2 Conservation of angular momentum about the north pole

In general, for unforced motion on a manifold which contains a translational symmetry, momentum in the direction of this symmetry is conserved. Likewise, if the manifold contains an axis of symmetry, the angular momentum about this axis is conserved. In either case, the form of the equation describing the evolution of the density of the conserved quantity will take the form of a conservation law. To be more specific, for unforced motion on a rotating sphere, it is angular momentum about the axis of rotation which is conserved. In contrast, zonal and meridional momentum are not conserved on the sphere, even in the absence of external forces, since the manifold is not flat. As such, the time tendency of the zonal and meridional momentum will not appear in a conservative form. This is the physical/geometric explanation for why the momentum equations on the sphere contain both "advective metric terms" and "frictional metric terms." The following discussion provides an elaboration of this point.

Consider the case of spherical coordinates for describing motion on a rotating sphere. In this subsection, it is sufficient to employ the more familiar vector notation introduced in Section 4.6. In this special case, the friction components (9.182) and (9.183) can be written in the compact form

$$F^u = (\cos\phi)^{-1}\,\nabla_h\cdot\mathbf{P} \tag{9.148}$$

$$F^v = (\cos\phi)^{-1}\,\nabla_h\cdot\mathbf{Q}, \tag{9.149}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are horizontal vectors given by

$$\mathbf{P} = A\cos\phi\,(\hat{\lambda}\,D_T + \hat{\phi}\,D_S) \tag{9.150}$$

$$\mathbf{Q} = A\cos\phi\,(\hat{\lambda}\,D_S - \hat{\phi}\,D_T). \tag{9.151}$$

As a result, the zonal momentum equation takes the form

$$\rho\,dV\,[u_{,t} + \mathbf{u}\cdot\nabla u - (u\,v/a)\tan\phi - fv] = dV(-p_{,x} + (\cos\phi)^{-1}\,\nabla_h\cdot\mathbf{P}), \tag{9.152}$$

where again $\rho\,dV$ is the conserved mass of a parcel. Notably, this equation cannot be written in the form of a conservation equation, as expected since the zonal momentum $\rho\,dV\,u$ is not conserved on a sphere. However, the angular momentum about the north pole can be written such, as now shown. For the purpose, multiply equation (9.152) by $\cos\phi$ and use the conservation of mass to find

$$(\partial_t + \mathbf{u}\cdot\nabla)\,[\rho\,dV\,a\cos\phi\,(u + \Omega\,a\cos\phi)] = a\,dV\,(-p_{,\lambda} + \nabla_h\cdot\mathbf{P}) \tag{9.153}$$

where Ω is the Earth's rotation rate. This equation states that the angular momentum per unit volume

$$\rho\mathcal{M} = \rho\,a\cos\phi\,(u + \Omega\,a\cos\phi) \tag{9.154}$$

satisfies the conservation equation

$$\partial_t(\rho\,\mathcal{M}) + \nabla\cdot(\rho\,\mathcal{M}\,\mathbf{u}) = -p_{,\lambda} + \nabla_h\cdot\mathbf{P}. \tag{9.155}$$

That is, for motion on a smooth sphere, $\int dV\,\rho\mathcal{M}$ is a constant in time.

9.6.3 The advective and frictional metric terms

It is useful to provide a more mathematical statement concerning the origin of the metric terms. For this purpose, consider the momemtum equations for fluid motion on an arbitrary manifold

$$\rho \frac{Du^m}{Dt} = T^{mn}_{;n} + \rho f^m. \tag{9.156}$$

The acceleration of a fluid parcel takes the form

$$\begin{aligned} \rho \frac{Du^m}{Dt} &= \rho(u^m_{,t} + u^n u^m_{;n}) \\ &= (\rho u^m)_{,t} + (\rho u^m u^n)_{;n}. \end{aligned} \tag{9.157}$$

To reach this result, the conservation of mass on a curved manifold has been used

$$\rho_{,t} + (\rho u^n)_{;n} = 0. \tag{9.158}$$

As such, the time tendency for the momentum density is given by

$$(\rho u^m)_{,t} = (T^{mn} - \rho u^m u^n)_{;n} + \rho f^m. \tag{9.159}$$

This equation is written in the same form as the Cartesian equivalent (9.143), except that now the derivatives are covariant and so contain information about the generally curved manifold. Expanding the covariant divergence using equation (9.110) yields

$$(\rho u^m)_{,t} = (\sqrt{\mathcal{G}})^{-1} [\sqrt{\mathcal{G}} (T^{mn} - \rho u^m u^n)]_{,n} + (T^{ab} - \rho u^a u^b) \Gamma^m_{ab} + \rho f^m. \tag{9.160}$$

The Christoffel symbol Γ^m_{np} accounts for the spatial dependence of the basis vectors. The $\Gamma^m_{ab} \tau^{ab}$ term is the "frictional metric term" and the $\rho u^a u^b \Gamma^m_{ab}$ term is the "advective metric term."

Integration of a quantity over the volume of a finite grid cell in arbitrary coordinates means performing an integral of the form

$$\int dV \psi = \int \sqrt{\mathcal{G}} d\xi^1 d\xi^2 d\xi^3 \psi, \tag{9.161}$$

where

$$dV = \sqrt{\mathcal{G}} d\xi^1 d\xi^2 d\xi^3 \tag{9.162}$$

is the invariant volume element. For example, in spherical coordinates

$$dV = a^2 \cos\phi \, d\lambda \, d\phi \, dz. \tag{9.163}$$

Hence, the metric terms cannot be written as the difference of fluxes across grid cells faces, whereas the remaining terms can. For flat space using Cartesian coordinates, the metric terms drop out, thus recovering the results discussed in Section 9.6.1.

9.7 Functional formalism

When implementing friction operators in a numerical model, it is important to maintain as much of the continuum properties as feasible. For constant viscosity models, a direct discretization based on the formulas of Bryan (1969) (see Section 9.8) seem to be sufficient. These formulas consist of a Laplacian plus extra "metric terms" which arise from the spherical geometry. When using non-constant viscosities, Wajsowicz (1993) pointed out that there are extra metric terms proportional to the derivatives of the viscosity (Section 9.8). The metric terms are cumbersome to discretize, and they generally introduce computational modes to the discreted friction operator on a B-grid. Hence, they are often ignored as in Cox (1984), but at the cost of no longer maintaining angular momentum conservation, as discussed previously.

For general orthogonal curvilinear coordinates, the metric terms are tedious to compute. In this case, it becomes even more clear that it is preferable to directly discretize the expressions (9.126) and (9.127). However, some attempts to do so on the B-grid have led to the presence of computational modes. Indeed, when using the Smagorinsky viscosity, these approaches can lead to unacceptably noisy solutions, more-so than found using the Laplacian plus metric approach.

In conclusion, either approach is unsatisfying, and so another approach is required. The purpose of this section is to introduce a different approach based on the same functional formalism applied to the isoneutral diffusion operator (Appendix C and Griffies, Gnanadesikan, Pacanowski, Larichev, Dukowicz, and Smith, 1998). This method provides a general framework that leads to a suitable discretization of the friction operator.

9.7.1 Continuum formulation

As shown in Section 9.4.7, the effects on kinetic energy dissipation from horizontal deformations in the fluid takes the form $-\rho_o \int dV\, A\, (D_T^2 + D_S^2)$, where a Boussinesq fluid has been assumed. One is therefore led to consider the functional

$$\mathcal{S} = -\rho_o \int \sqrt{\mathcal{G}}\, d\xi^1\, d\xi^2\, dz\, A\, (D_T^2 + D_S^2). \tag{9.164}$$

As shown in this section, the functional derivative $\delta\mathcal{S}/\delta u^a$ is proportional to the friction $g_{ab}\, F^b$. The connection between a functional and the friction is afforded through the self-adjointness of the friction operator. This result will then lead to a numerical discretization of the friction which is ensured to dissipate kinetic energy on the discrete lattice.

To make this method work, two assumptions are needed: (1) The viscosity is functionally independent of the velocity field. (2) The flow satisfies "natural boundary conditions", of which no-slip is one. The Smagorinsky viscosity does not satisfy the first assumption. Nonetheless, the functional approach will lead to a discretization inside of which one can employ the Smagorinsky viscosity. A similar assumption was used to discretize the isoneutral diffusion operator when diffusing active tracers.

Writing the functional as $\mathcal{S} = \int \sqrt{\mathcal{G}}\, d\xi^1\, d\xi^2\, dz\, \mathcal{L}$ leads to the variation

$$\delta\, \mathcal{S} = \int \sqrt{\mathcal{G}}\, d\xi^1\, d\xi^2\, dz\, \delta\mathcal{L}. \tag{9.165}$$

Note that the metric components are held fixed, since the only variation considered here is that of the velocity field $u^a \to u^a + \delta u^a$, not the underlying space-time geometry. Since $\mathcal{L}$ is a

function of the velocity and its derivative, $\mathcal{L}[u^a, u^a_{,b}]$, its variation leads to

$$\begin{aligned}\delta\,\mathcal{S} &= \int \sqrt{\mathcal{G}}\,d\xi^1\,d\xi^2\,dz\,\left[\frac{\delta\,\mathcal{L}}{\delta u^a}\,\delta u^a + \frac{\delta\,\mathcal{L}}{\delta u^a_{,b}}\,\delta u^a_{,b}\right] \\ &= \int d\xi^1\,d\xi^2\,dz\,\left[\sqrt{\mathcal{G}}\,\frac{\delta\,\mathcal{L}}{\delta u^a}\,\delta u^a + \partial_b\left(\sqrt{\mathcal{G}}\,\frac{\delta\,\mathcal{L}}{\delta u^a_{,b}}\,\delta u^a\right) - \partial_b\left(\sqrt{\mathcal{G}}\,\frac{\delta\,\mathcal{L}}{\delta u^a_{,b}}\right)\,\delta u^a\right], \quad (9.166)\end{aligned}$$

where an integration by parts has been performed. The total derivative reduces to a surface term, which vanishes when either $\delta u^a = 0$ on all boundaries, or $\hat{n}_b\,(\delta\,\mathcal{L}/\delta u^a_{,b})\,\delta u^a = 0$, where $\hat{n}_b$ are components to the outward normal at the boundaries. These two conditions define the "natural boundary conditions" mentioned above. If the velocity, and hence its variation, satisfy the no-slip condition, then $\delta u^a = 0$ on all boundaries and the total derivative can be dropped. More general boundary conditions can be derived from the second type of natural boundary condition, yet they are not considered here since MOM employs no-slip on the side boundaries. With natural boundary conditions, the variation of the functional takes the form

$$\delta\,\mathcal{S} = \int \sqrt{\mathcal{G}}\,d\xi^1\,d\xi^2\,dz\,\left[\frac{\delta\,\mathcal{L}}{\delta u^a} - \mathcal{G}^{-1/2}\,\partial_b\left(\sqrt{\mathcal{G}}\,\frac{\delta\,\mathcal{L}}{\delta u^a_{,b}}\right)\right]\,\delta u^a, \quad (9.167)$$

which then leads to the functional derivative

$$\frac{\delta\,\mathcal{S}}{\delta u^a} = \frac{\delta\,\mathcal{L}}{\delta u^a} - \mathcal{G}^{-1/2}\,\partial_b\left(\sqrt{\mathcal{G}}\,\frac{\delta\,\mathcal{L}}{\delta u^a_{,b}}\right). \quad (9.168)$$

To reach this result, it was necessary to use the identity

$$\frac{\delta u^a(\vec{x})}{\delta u^b(\vec{y})} = \delta^a_b\,\delta(\vec{x}-\vec{y}), \quad (9.169)$$

where $\delta(\vec{x}-\vec{y})$ is the Dirac delta-function. The delta-function has physical dimensions of inverse volume L^{-3}. Hence,

$$\int \sqrt{\mathcal{G}}\,d\xi^1\,d\xi^2\,dz\,\delta(\vec{x}-\vec{y}) = 1, \quad (9.170)$$

so long as the integration is over a domain containing the singular point $\vec{x} = \vec{y}$; otherwise, the integral vanishes.

Now that the general functional derivative of $\mathcal{S}$ has been computed, it remains to prove the connection to the friction vector. For this purpose, recall equation (9.103), for which is was shown that the horizontal tension can be written

$$D_T = u^1_{,1} - u^2_{,2} + u^m\,\partial_m\,\ln(h_1/h_2), \quad (9.171)$$

and the horizontal shearing strain can be written

$$D_S = \frac{h_1}{h_2}\,u^1_{,2} + \frac{h_2}{h_1}\,u^2_{,1}. \quad (9.172)$$

Hence, the horizontal tension is functionally dependent on both the velocity and its partial derivatives, whereas the shearing strain is dependent only on the velocity partial derivatives. These results lead to the functional derivatives

$$-\frac{\delta\mathcal{L}}{\delta u^a} = 2\,\rho_o\,A\,D_T\,[\,\delta^1_a\,\partial_1\,\ln(h_1/h_2) + \delta^2_a\,\partial_2\,\ln(h_1/h_2)\,], \quad (9.173)$$

and

$$
\begin{aligned}
-\frac{\delta \mathcal{L}}{\delta u^a_{,b}} &= 2\,\rho_o\,A\left(D_T\,\frac{\delta D_T}{\delta u^a_{,b}} + D_S\,\frac{\delta D_S}{\delta u^a_{,b}}\right) \\
&= 2\,\rho_o\,A\left(D_T\,(\delta^1_a\,\delta^b_1 - \delta^2_a\,\delta^b_2) + D_S\,(\frac{h_1}{h_2}\,\delta^1_a\,\delta^b_2 + \frac{h_2}{h_1}\,\delta^2_a\,\delta^b_1)\right). \qquad (9.174)
\end{aligned}
$$

For $a = 1$, these results lead to

$$
\begin{aligned}
\frac{1}{2\,\rho_o}\,\frac{\delta\,\mathcal{S}}{\delta u^1} &= -A\,D_T\,\partial_1\,\ln(h_1/h_2) + A\,D_T\,\partial_1\,\ln(h_1\,h_2) + (A\,D_T)_{,1} + \frac{1}{h_1\,h_2}\,(h_1^2\,A\,D_S)_{,2} \\
&= h_2^{-2}\,(h_2^2\,A\,D_T)_{,1} + \frac{1}{h_1\,h_2}\,(h_1^2\,A\,D_S)_{,2} \\
&= h_1^2\,F^1. \qquad (9.175)
\end{aligned}
$$

Similar manipulations with $a = 2$ lead to

$$
\frac{1}{2\,\rho_o}\,\frac{\delta\,\mathcal{S}}{\delta u^a} = g_{ab}\,F^b, \qquad (9.176)
$$

which is the desired general result.

9.7.2 Discrete formulation

The discrete formulation requires a number of details that are provided in Appendix D.

9.8 Old friction implementation

The purpose of this section is to summarize the old method for discretizing friction in MOM, which was to directly discretize the equations of Bryan (1969) and Wajsowicz (1993).

9.8.1 Spherical form of second order friction

In spherical coordinates, the metric takes the form

$$
\begin{aligned}
g_{ij} &= \mathrm{diag}\,(g_{\lambda\lambda}, g_{\phi\phi}, g_{rr}) \\
&= \mathrm{diag}\,(a^2\cos^2\phi, a^2, 1) \qquad (9.177)
\end{aligned}
$$

and

$$
(\delta x, \delta y, \delta z) = (a\cos\phi\,\delta\lambda, a\,\delta\phi, \delta r). \qquad (9.178)
$$

Consequently, it is only $g_{11} = g_{\lambda\lambda}$ which has nonzero spatial dependence. The friction then can be written

$$
F^x = \frac{\partial(A\,D_T)}{\partial x} + \frac{1}{\cos^2\phi}\,\frac{\partial(A\,\cos^2\phi\,D_S)}{\partial y} + (\kappa\,u_{,z})_{,z} \qquad (9.179)
$$

$$
F^y = \frac{\partial(A\,D_S)}{\partial x} - \frac{1}{\cos^2\phi}\,\frac{\partial(A\,\cos^2\phi\,D_T)}{\partial y} + (\kappa\,v_{,z})_{,z}, \qquad (9.180)
$$

where

$$\begin{aligned} D_T &= u_{,x} - v_{,y} - (v/a)\tan\phi \\ &= (a\cos\phi)^{-1}\left(u_{,\lambda} - v_{,\phi}\cos\phi - v\sin\phi\right) \\ D_S &= v_{,x} + \cos\phi\,(u/\cos\phi)_{,y} \\ &= (a\cos\phi)^{-1}\left(v_{,\lambda} + u_{,\phi}\cos\phi + u\sin\phi\right). \end{aligned} \tag{9.181}$$

The terms

$$F^u = \frac{1}{a\cos\phi}\frac{\partial(A\,D_T)}{\partial\lambda} + \frac{1}{a\cos^2\phi}\frac{\partial(A\cos^2\phi\,D_S)}{\partial\phi} \tag{9.182}$$

$$F^v = \frac{1}{a\cos\phi}\frac{\partial(A\,D_S)}{\partial\lambda} - \frac{1}{a\cos^2\phi}\frac{\partial(A\cos^2\phi\,D_T)}{\partial\phi} \tag{9.183}$$

can be massaged into the form presented by Bryan (1969) and Wajsowicz (1993); that is the purpose of the remainder of this section.

9.8.2 Zonal friction

The lateral friction acting on the zonal velocity takes the expanded form

$$\begin{aligned} a\,F^u &= \frac{1}{\cos\phi}\left(D_T\,\partial_\lambda A + A\,\partial_\lambda D_T\right) \\ &+ \frac{1}{\cos^2\phi}\left(D_S\,\partial_\phi A\cos^2\phi + A\,\partial_\phi D_S\cos^2\phi - 2\,A\,D_S\cos\phi\sin\phi\right) \\ &= \frac{1}{\cos\phi}\left(D_T\,\partial_\lambda A + D_S\,\partial_\phi A\cos\phi\right) \\ &- \frac{2A}{a\cos\phi}\left(v_{,\lambda}\tan\phi + u_{,\phi}\sin\phi + u\sin\phi\tan\phi\right) \\ &+ \frac{A}{a\cos\phi}\left(u_{,\lambda\lambda}\sec\phi + u_{,\phi\phi}\cos\phi + u_{,\phi}\sin\phi + u\sec\phi\right) \\ &= \frac{1}{\cos\phi}\left(D_T\,\partial_\lambda A + D_S\,\partial_\phi A\cos\phi\right) \\ &+ \frac{A}{a}\left(u_{,\lambda\lambda}\sec^2\phi + u_{,\phi\phi} - u_{,\phi}\tan\phi + u\,(\sec^2\phi - 2\tan^2\phi) - 2v_{,\lambda}\sec^2\phi\sin\phi\right) \\ &= \frac{1}{\cos\phi}\left(D_T\,\partial_\lambda A + D_S\,\partial_\phi A\cos\phi\right) \\ &+ \frac{A}{a}\left(u_{,\lambda\lambda}\sec^2\phi + \sec\phi\,(u_{,\phi}\cos\phi)_{,\phi} + u(1-\tan^2\phi) - 2v_{,\lambda}\sec^2\phi\sin\phi\right), \end{aligned} \tag{9.184}$$

which renders

$$\begin{aligned} F^u &= A\left(\nabla_h^2\,u + \frac{u(1-\tan^2\phi)}{a^2} - \frac{2v_{,\lambda}\sin\phi}{a^2\cos^2\phi}\right) \\ &+ \frac{1}{a\cos\phi}\left(D_T\,\partial_\lambda A + D_S\,\partial_\phi A\cos\phi\right). \end{aligned} \tag{9.185}$$

The second bracketed term in this expression arises from the spatial dependence of the viscosity coefficient, and should be present for any non-constant viscosity coefficient model. These non-constant viscosity coefficient terms amount to those identified by Wajsowicz (1993).

It is useful to perform one final step in the formulation in order to bring the non-constant viscosity coefficient inside the Laplacian. For this purpose, the Laplacian and non-constant viscosity coefficient terms are expanded to yield

$$\begin{aligned}
F^u &= \frac{A}{a^2\cos^2\phi}u_{,\lambda\lambda} + \frac{A}{a^2\cos\phi}(u_{,\phi}\cos\phi)_{,\phi} + \frac{A\,u\,(1-\tan^2\phi)}{a^2} - \frac{2\,A\,v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \\
&+ \frac{A_\lambda}{a^2\cos^2\phi}(u_{,\lambda} - v_{,\phi}\,\cos\phi - v\,\sin\phi) + \frac{A_{,\phi}}{a^2\cos\phi}(v_{,\lambda} + u_{,\phi}\,\cos\phi + u\,\sin\phi) \\
&= \frac{1}{a^2\cos^2\phi}(A\,u_{,\lambda\lambda} + A_{,\lambda}\,u_{,\lambda}) + \left(\frac{1}{a^2\cos\phi}(A\,u_{,\phi}\cos\phi)_{,\phi} - A_{,\phi}\,u_{,\phi}a^{-2}\right) \\
&+ \frac{A\,u\,(1-\tan^2\phi)}{a^2} - \frac{2\,A\,v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \\
&- \frac{A_{,\lambda}}{a^2\cos^2\phi}(v_{,\phi}\,\cos\phi + v\,\sin\phi) + a^{-2}\,A_\phi(v_{,\lambda}\sec\phi + u_{,\phi} + u\,\tan\phi) \\
&= \nabla_h\cdot(A\,\nabla_h u) + \frac{A\,u\,(1-\tan^2\phi)}{a^2} - \frac{2\,A\,v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \\
&- \frac{A_{,\lambda}}{a^2\cos\phi}(v_{,\phi} + v\,\tan\phi) + \frac{A_{,\phi}}{a^2\cos\phi}(v_{,\lambda} + u\,\sin\phi). \qquad (9.186)
\end{aligned}$$

This expression can be written as

$$\boxed{F^u = \nabla_h\cdot(A\,\nabla_h u) + old_metric^u + new_metric^u,} \qquad (9.187)$$

where

$$\nabla_h\cdot(A\,\nabla_h u) = \frac{1}{a^2\cos^2\phi}(Au_{,\lambda})_{,\lambda} + \frac{1}{a^2\cos\phi}(A\,u_{,\phi}\cos\phi)_{,\phi} \qquad (9.188)$$

is the horizontal Laplacian with the generally non-constant viscosity coefficient inserted. Note that this Laplacian is acting on the zonal velocity as if it was a scalar field. The term

$$old_metric^u = \frac{A\,u\,(1-\tan^2\phi)}{a^2} - \frac{2\,A\,v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \qquad (9.189)$$

is the metric term employed for constant horizontal viscosity coefficient (Bryan 1969), and

$$new_metric^u = -\frac{\partial_\lambda A}{a^2\cos\phi}(v_{,\phi} + v\,\tan\phi) + \frac{\partial_\phi A}{a^2\cos\phi}(v_{,\lambda} + u\,\sin\phi) \qquad (9.190)$$

is the metric term arising from spatial dependence in the viscosity coefficient (Wajsowicz 1993).

9.8.3 Meridional friction

Repeating the exercise just performed for the zonal friction yields the following lines of algebra for the meridional friction

$$\begin{aligned}
aF^v &= \frac{1}{\cos\phi}\,(D_S\,\partial_\lambda A + A\,\partial_\lambda\,D_S) \\
&- \frac{1}{\cos^2\phi}\,(D_T\,\partial_\phi A\,\cos^2\phi + A\,\partial_\phi D_T\,\cos^2\phi - 2\,A\,D_T\,\cos\phi\sin\phi)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\cos\phi}\,(D_S\,\partial_\lambda A - D_T\,\partial_\phi A\,\cos\phi) \\
&+ \frac{2A}{a\,\cos\phi}(u_{,\lambda}\tan\phi - v_{,\phi}\tan\phi\cos\phi - v\tan\phi\sin\phi) \\
&+ \frac{A}{a\,\cos\phi}\left(v_{,\lambda\lambda}\,\sec\phi + v_{,\phi\phi}\,\cos\phi + v_{,\phi}\,\sin\phi + v\,\sec\phi\right) \\
&= \frac{1}{\cos\phi}\,(D_S\,\partial_\lambda A - D_T\,\partial_\phi A\,\cos\phi) \\
&+ \frac{A}{a}\left(v_{,\lambda\lambda}\sec^2\phi + v_{,\phi\phi} - v_{,\phi}\tan\phi + v(\sec^2\phi - 2\tan^2\phi) + 2u_{,\lambda}\sec^2\phi\sin\phi\right) \\
&= \frac{1}{\cos\phi}\,(D_S\,\partial_\lambda A - D_T\,\partial_\phi A\,\cos\phi) \\
&+ \frac{A}{a}\left(v_{,\lambda\lambda}\sec^2\phi + \sec\phi\,(v_{,\phi}\cos\phi)_{,\phi} + v(1-\tan^2\phi) + 2u_{,\lambda}\sec^2\phi\sin\phi\right), \qquad (9.191)
\end{aligned}
$$

which renders

$$
\begin{aligned}
F^v &= A\left(\nabla_h^2\,v + \frac{v(1-\tan^2\phi)}{a^2} + \frac{2u_{,\lambda}\,\sin\phi}{a^2\cos^2\phi}\right) \\
&+ \frac{1}{a\,\cos\phi}\,(D_S\,\partial_\lambda A - D_T\,\partial_\phi A\,\cos\phi). \qquad (9.192)
\end{aligned}
$$

Again, it is useful to expand this friction one more step in order to bring the viscosity coefficient inside the Laplacian. This manipulation yields

$$
\boxed{F^v = \nabla_h\cdot(A\,\nabla_h v) + old_metric^v + new_metric^v,} \qquad (9.193)
$$

where

$$
old_metric^v = \frac{A\,v\,(1-\tan^2\phi)}{a^2} + \frac{2\,A\,u_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \qquad (9.194)
$$

is the metric employed for constant horizontal viscosity coefficient, and

$$
new_metric^v = \frac{\partial_\lambda A}{a^2\cos\phi}(u_{,\phi} + u\,\tan\phi) + \frac{\partial_\phi A}{a^2\cos\phi}(-u_{,\lambda} + v\,\sin\phi). \qquad (9.195)
$$

9.8.4 Old biharmonic algorithm

The following algorithm summarizes the steps used to compute the biharmonic friction vector in MOM prior to Summer 1999. This method for non-constant viscosities was incorrect as it distributed the viscosity only on the second part of the calculation, rather than evenly as necessitated by the constraints of kinetic energy dissipation (Section 9.5.2). Hence, this approach may lead to problems with non-constant viscosities, although none have been found, and it is thought that the differences will be small.

- **Second order calculation**: The physical components (F^x, F^y) of the second order friction vector are summarized in Section 9.4.8. The same calculation is made for the biharmonic scheme, only using here a unit viscosity. Define the physical components (F^u, F^v), which equal (F^x, F^y) sans the vertical derivative term $(\kappa\,\mathbf{u}_{,z})_{,z}$

$$
\begin{aligned}
F^u &= g_{22}^{-1}\,(g_{22}\,D_T)_{,x} + g_{11}^{-1}\,(g_{11}\,D_S)_{,y} \qquad (9.196) \\
F^v &= g_{11}^{-1}\,(g_{11}\,D_T)_{,y} - g_{22}^{-1}\,(g_{22}\,D_S)_{,x} \qquad (9.197)
\end{aligned}
$$

where the horizontal tension and strain are given by the usual forms

$$D_T = \sqrt{g_{22}}\,(u/\sqrt{g_{22}})_{,x} - \sqrt{g_{11}}\,(v/\sqrt{g_{11}})_{,y} \tag{9.198}$$
$$D_S = \sqrt{g_{11}}\,(u/\sqrt{g_{11}})_{,y} + \sqrt{g_{22}}\,(v/\sqrt{g_{22}})_{,x}. \tag{9.199}$$

- **Fourth order calculation**: Use again the results from the second order calculation, only now with the second order friction vector taking the place of the velocity vector and the viscosity A set to the biharmonic viscosity. That is, compute

$$F_B^u = g_{22}^{-1}\,(g_{22}\,A\,D_T^B)_{,x} + g_{11}^{-1}\,(g_{11}\,A\,D_S^B)_{,y} \tag{9.200}$$
$$F_B^v = g_{11}^{-1}\,(g_{11}\,A\,D_T^B)_{,y} - g_{22}^{-1}\,(g_{22}\,A\,D_S^B)_{,x} \tag{9.201}$$

where the new horizontal tension and strain are given by

$$D_T^B = \sqrt{g_{22}}\,(F^u/\sqrt{g_{22}})_{,x} - \sqrt{g_{11}}\,(F^v/\sqrt{g_{11}})_{,y} \tag{9.202}$$
$$D_S^B = \sqrt{g_{11}}\,(F^u/\sqrt{g_{11}})_{,y} + \sqrt{g_{22}}\,(F^v/\sqrt{g_{22}})_{,x} \tag{9.203}$$

For spherical coordinates in MOM, the following steps are taken:

- **Second order calculation**: The Laplacian friction vector is computed as in Section 9.8.2, only with a unit viscosity:

$$F^u = \nabla_h \cdot \nabla_h u + old_metric^u \tag{9.204}$$
$$F^v = \nabla_h \cdot \nabla_h v + old_metric^v, \tag{9.205}$$

where

$$old_metric^u = \frac{u\,(1-\tan^2\phi)}{a^2} - \frac{2\,v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi}, \tag{9.206}$$
$$old_metric^v = \frac{v\,(1-\tan^2\phi)}{a^2} + \frac{2\,u_{,\lambda}\,\sin\phi}{a^2\cos^2\phi}. \tag{9.207}$$

- **Fourth order calculation**: Now use the full biharmonic viscosity to compute the biharmonic friction

$$F_B^u = \nabla_h \cdot (A\,\nabla_h F^u) + old_metric^u + new_metric^u \tag{9.208}$$
$$F_B^v = \nabla_h \cdot (A\,\nabla_h F^v) + old_metric^v + new_metric^v. \tag{9.209}$$

The metric terms are given by

$$old_metric^u = \frac{A\,F^u\,(1-\tan^2\phi)}{a^2} - \frac{2\,A\,F^v_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \tag{9.210}$$
$$old_metric^v = \frac{A\,F^v\,(1-\tan^2\phi)}{a^2} + \frac{2\,A\,F^u_{,\lambda}\,\sin\phi}{a^2\cos^2\phi} \tag{9.211}$$
$$new_metric^u = -\frac{A_{,\lambda}}{a^2\cos\phi}(F^v_{,\phi} + F^v\,\tan\phi) + \frac{A_{,\phi}}{a^2\cos\phi}(F^v_{,\lambda} + F^u\,\sin\phi) \tag{9.212}$$
$$new_metric^v = \frac{A_{,\lambda}}{a^2\cos\phi}(F^u_{,\phi} + F^u\,\tan\phi) + \frac{A_{,\phi}}{a^2\cos\phi}(-F^u_{,\lambda} + F^v\,\sin\phi). \tag{9.213}$$

Part III

Code design

Chapter 10

Design Philosophy

10.1 Objective

The GFDL Modular Ocean Model was designed with one purpose in mind: to maximize scientific productivity within the research environment at GFDL. As indicated in the introduction, the computational environment at GFDL has undergone change with each new computer procurement. To keep pace, efforts have focused on developing one model capable of taking advantage of scalar, vector and multiple processors within this increasingly varied computational environment. At the same time, consideration has been given to organizing the model to allow a large number of options, diagnostics, and physics parameterizations to co-exist in a way that is understandable, extendable, and easily accessible to scientists. In one sense, the design was strongly influenced by having CRAY vector platforms as the computational workhorses at GFDL since 1990. Although changes have been introduced to take advantage of scalability on multiple processors, the focus has remained on factors that influence overall scientific productivity and the design continues to be motivated by a search for a better way to do science from the trenches of scientific programming. As such, the scientific generality and flexibility within MOM make it well suited for use by the general oceanographic community[1]. The following factors are considered to be important in maximizing overall scientific productivity.

10.1.1 Speed

In the past, speed was often thought of as being the equivalent of scientific productivity. In an operational setting where a model is rarely changed, it is justifiable to expend enormous effort to minimize wall clock time. In a research environment, it has become increasingly apparent that other considerations are important. This is particularly noticeable when changes introduced to take advantage of speed on a particular platform make implementation of science thereafter more difficult[2]. What is needed are changes which increase speed[3] but don't reduce clarity. This is best met by using more appropriate numerical algorithms, more optimizing

[1] Optimizing for the idiosyncrasies in computer environments outside GFDL is left to the researcher.

[2] As in the Cyber 205 experience.

[3] It is reassuring that the ideas influencing the design of MOM have not significantly altered speed when compared to MOM 1. Early comparisons were carried out using the standard test case resolution of 4° by 3° and 15 levels. Changes in the external mode of MOM 1 were necessary to assure the same accuracy as in MOM and there were no diagnostics enabled. MOM ranged from 3% slower to 6% faster (depending on size of the memory window) than MOM 1. Minimum memory configuration in MOM was 1% greater than in MOM 1.

compilers, and faster processors. In the long run, straightforward coding is best. Tricky coding introduced to increase speed is often rendered useless by newer compilers. Worse yet, these tricks may be detrimental on other platforms. Ultimately, speed should be the business of compilers and better algorithms, not physical scientists playing games to beat compilers.

Two philosophies of model building can be summarized by first stating the intent and then asking a question.

1. The aim is to do as much science as possible with this model. Now, how can it be made to execute as fast as possible?

2. The aim is to make this model execute as fast as possible. Now, what science can be done with it?

MOM is the result of focusing on the first.

10.1.2 Flexibility

To be a useful research tool, MOM needs to be easily configurable in many different ways. Also of importance is access to a large number of parameterizations for intercomparisons within the framework of one model. Using preprocessor "ifdefs" gives this flexibility. However, they must be used wisely. Indiscriminant use of preprocessor "ifdefs" can lead to a tangled mess of limited usefullness. With time, more and more constructs within MOM will be replaced by their Fortran 90 counterparts which should allow for even greater flexibility.

10.1.3 Modularity

MOM is continually being infused with new ideas and the resulting growth can present problems. For example, anyone who repeatedly changes or adds to a large model will appreciate that after time, the model can become unmanageable. At some point, inter-connectivity between sections of code increases to a point where making changes in one place inadvertently breaks something seemingly unrelated. Further limitations become apparent when previously added code acts as a road block to new development. To a large extent, modularity has been used as the key organizational approach to solve this problem and its use is explained in Chapter 15. The other part of the solution involves resisting temptation to make changes in a quick and dirty way for short term gains which inevitably turn into long term hindrances. Again, Fortran 90 with its well defined modules and interfaces should increase modularity.

10.1.4 Documentation

A good documentation aids in understanding the big picture as well as the little details which are necessary if a model is to be used and extended by many researchers. To this end, details of numerics right down to the subscripts within this documentation consistantly match what is found in the model code. This level of detail plus a straightforward coding style bolsters the scientific accessibility of MOM. It also provides another way of verifying that complicated numerics are correct. The manual should be considered a living document which actively reflects the current status of MOM as well as serving as a repository for details inappropriate for published papers and guidelines for usage of parameterizations. Therefore, understanding this manual will allow researchers to take a big step towards gaining a working knowledge of MOM.

10.1.5 Coding efficiency.

Inevitably, the size of a research model increases with time. However, economy of code is always desirable. Voluminous coding to support issues which are not central to science accumulates and, if not restrained, starts to account for the bulk of model code. This practice is to be discouraged, although there is fine line to be drawn and the answers are not always unambiguous. When adding a new feature, the question to be asked is: How much code is this idea worth and can it be justified with respect to the prevailing level of scientific approximations being made? Some areas within MOM have become overly large and complex but with questionable[4] gain. As time permits, "dead wood" will be eliminated and simplifications will follow.

10.1.6 Ability to upgrade.

It is vitaly important for researchers to be able to incorporate code changes (which may be of interest personally but not appropriate for general dissemination) into newer versions of a model. It is in this way that researchers are able to take advantage of new parameterizations while retaining local personal changes. Also of importance is the ability to incorporate "bug" fixes. In the past, both of these operations have presented significant difficulties. These difficulties have been largely eliminated by the tools and method described in Section 3.12.

[4]Cases in point were the old I/O manager and time manager modules.

Chapter 11

Uni-tasking

Uni-tasking refers to executing on one processor as opposed to multi-tasking which is executing on multiple processors. This chapter describes the dataflow used to integrate equations given in Chapter 4 on a single processor. The method allows problems to be solved which would otherwise be intractable due to their large size[1]. The method is extended to multiple processors in Chapter 12.

11.1 Why memory management is important

Productivity is related to total throughput of a computer system over time. The throughput is limited by how well a mix of jobs fits into available storage (memory and disk) and how fast the mix executes. At GFDL, the job mix is a combination of production models, analysis, development and interactive work. It is time dependent and is determined by guidelines intended to optimize total throughput. All jobs tie up a certain amount of memory[2]. The amount of memory consumed by high resolution models has a cruicial impact on overall productivity because it can easily exceed the computational storage capacity of the system. How much storage is enough? To admit most eddy structures would require a resolution of about $1/12^\circ$ which would take about 930MW for one time level of one variable in a global model assuming 100 vertical levels[3]. Even low resolution models that use memory wastefully limit the number of jobs in the system and therefore also impact overall throughput. The dataflow scheme described below utilizes a memory window to minimize model memory requirements. Without this memory window, memory requirements could easily increase by one or two orders of magnitude on a single processor! This is the scenerio for large jobs where the bulk of data must stored on a rotating or preferably solid state disk. For smaller jobs, the bulk of data may be stored on a ramdrive[4] and for this scenerio, memory requirements on a single processor would increase by at least 50% if a memory window were not employed.

[1]The size of the problem is related to the product of the number of grid cells in latitude, longitude, and depth. It is also related to which options are enabled and how many additional passive tracers are used.

[2]The assumption is that memory is a precious resource which is to be conserved. Historically, this has been true and the expectation is that it will continue to be in the future.

[3]At GFDL, the eight processor CRAY YMP had 32MW of central memory which had been upgraded to 64MW within the last year of its lifetime. Solid state disk space was 256MW.

[4]A ramdrive is a portion of memory used as a disk to speed up reading and writing of data. If solid state disk is not available, a ramdrive is a good alternative.

11.2 Minimizing the memory requirement

To integrate the equations detailed in Chapter 4, a volume of ocean is divided into a large number of rectangularly shaped cells within which equations are solved by finite difference techniques. Storage for each variable in addition to other quantities such as fluxes through cell faces must be allocated for each cell. As noted above, if storage were allocated entirely within memory, the maximum attainable resolution on a single processor would be severely limited. This restriction can be greatly relaxed by allocating the bulk of storage on a secondary device such as disk (preferably solid state) and allocating only enough memory to integrate equations for one slice through the ocean's volume at a time. Successively reading slices from disk into memory, integrating, and writing updated slices back to disk[5] allows equations to be solved for the entire volume of ocean with significantly less memory in comparison to total storage requirements. This is the essence of a memory window which is described in more detail below. For models with domain sizes which are small compared to available memory, or on architectures where disk access may be prohibitively slow, an option is available (Section 24.3) to turn the disk area into a ramdrive.

There are various ways to slice through a volume of data. As an example, refer to Figure 11.1a which illustrates a three dimensional block of data on disk arranged such that there are $i = 1 \cdots imt$ longitudes, $jrow = 1 \cdots jmt$ latitudes, and $k = 1 \cdots km$ depth levels. As indicated, slicing the volume of data along surfaces of constant longitude, latitude, and depth requires that when brought into central memory, the arrays needed to hold the slices be of size imt x km, jmt x km, or imt x jmt. Perhaps the most intuitive way of dimensioning arrays is Array(imt,jmt). However, in general, the number of model latitudes jmt is usually comparable (within a factor of 1/2) to the number of longitudes imt but the number of depth levels km is typically 1/5 to 1/10 the number of longitudes. Therfore, dimensioning slices as Array(imt,jmt) can be eliminated as being too wasteful of memory. Even though dimensioning slices as Array(jmt,km) requires the least space, it is less favorable based on other considerations: chiefly, the added cost of more vector startups (discussed below) and the desire to easily reference longitudinal sections along circles of constant latitude (i.e. for diagnostic purposes and polar filtering). Based on the above, slices are dimensioned as Array(imt,km).

The reason that slices are dimensioned as Array(imt,km) instead of as Array(km,imt) is largely historical and again based on speed issues: the inner dimension is the <u>vector</u> dimension in Fortran and longer vectors execute faster than shorter ones. In addition, vector startup time can be significant for short vectors. Therefore the idea is to have long vectors and as few of them as possible[6] on vector computers. Essentially, these ideas led to dimensioning slices as Array(imt,km) to represents a longitudinal section of data along a circle of constant latitude. In general, more than one latitude row is needed in memory to solve equations and so arrays are dimensioned by a local latitude index as in Array(imt,km,jmw) where the number of local latitudes is typically $jmw = 3$.

On cache (fast on-chip memory) processors of the type seen in distributed platforms, the size of cache has a significant impact on speed. Smaller sized arrays are more likely to fit

[5]The viability of this depends on disk access speed. Solid State Disk on the CRAY YMP, C90, and T90 is fast enough to allow this to work well. Slower disk access can also work if the reads from disk are buffered by the work involved in updating the slice. In practice this is not difficult to implement as long as the slices are to be accessed in a predetermined way.

[6]In general, two dimensional variables will not collapse into one long vector because operations along the first dimension typically do not include boundary cells $i = 1$ and $i = imt$. In the k dimension, limits are sometimes a function of i and j. Running indices over boundary cells leads to out of bounds references which is to be discouraged because it leads to programming errors and prevents use of the bounds checking feature on compilers.

within available cache than larger sized arrays and this results in speed improvements. In fact, when multi-tasking, fitting arrays into available cache is the reason for observed super linear speed ups as the number of processors are increased. In this case, vectors are not as important as fitting arrays into cache. However, performance is not as simple as just cache size. On-chip bottlenecks can occur for a variety of reasons which are beyond the scope of this documentation. Regardless of chip design details, the focus should be to write code which is scientifically clear and general enough to stand through time. Incorporating code to correct a compiler's shortcommings is to be discouraged since the coding quirk is likely to survive long after the compiler is gone. If carried to excess, accumulation of antiquated "speed ups" serve only to obscure the science behind the code. In the long run, faster integrations are the result of more appropriate numerical techniques, better optimizing compilers, and faster computers.

11.2.1 Slicing through the 3-D prognostic data

Since there is a large separation in speed between internal and external gravity wave modes, the equations of motion are divided into a set of internal mode (3-D or baroclinic) and external mode (2-D or barotropic) equations for computational efficiency. The external mode equations represent the vertical integral of the full equations while the internal mode equations represent deviations from the vertical mean. In support of this division, prognostic data is divided into 3-D and 2-D data.

Two time levels of 3-D prognostic data at $\tau - 1$ and τ are illustrated in Figure 11.1b. Two time levels are needed because it takes two time levels to integrate the equations forward in time. The dashed line marks a longitudinal slice at latitude index "jrow" through both time levels. Within each slice, data is arranged as follows: The zonal component of velocity (internal mode only) is first, followed by the meridional component of velocity (internal mode only) and then temperature (T) and salinity (S). In general, there may be $n = 1, nt$ tracers but typically $nt = 2$ with n=1 referring to T and n=2 referring to S. Therefore, each latitude row indexed by "jrow" contains $imt \cdot km \cdot (2 + nt)$ data points for two time levels.

Figure 11.1c is a schematic of all 3-D prognostic data arranged by two time levels of latitude slices on disk from south to north by increasing global index "jrow". The disk area may also be thought of as a ramdrive which is just an array in memory dimensioned as 3D(imt,km,variable,jmt,2). The "2" accounts for the two time levels. A third time level for $\tau + 1$ is not needed on disk because updated data at $\tau + 1$ can be written back over the $\tau - 1$ area. However, in the memory window discussed below, space for the third time level $\tau + 1$ is needed. This is the reason why it is best to arrange prognostic data on a ramdrive rather than opening the memory window up to contain all of the latitude rows.

11.3 The Memory Window

The memory window (MW) provides the framework for solving all tracer and baroclinic equations in a uni-tasking as well as a multi-tasking environment. It is not used for the barotropic equation since the total storage for 2D arrays is small compared to 3D arrays. In precursors to MOM, an array of data along longitude and depth was indexed as A(i,k). Within the MW, the same array would be indexed as A(i,k,j) where the "j" index is a latitude index local to the MW. This generalized approach is capable of simulating the older method but allows for much greater flexibility. Some of the advantages of the MW are:

- Higher order finite difference schemes and parameterizations which require access to more than three latitude rows can be implemented in a straight forward manner.
- There is a reduction in the number of names required for variables. For example, in MOM 1 and previous incarnations, tracers required three names: one for the row being computed, one for the row to the north and another for the row to the south. However, there is only need for one name: the row being computed is accessed by local index *j* and rows to the north and south are accessed by local row indices *j-1* and *j+1*.
- Essentially all prognostic variables are subscripted by three spatial indices (i,k,j) as if infinite central memory were available. However, the actual memory needed is controlled by the size of the memory window *jmw*. Equations and coding looks the same, regardless of how large or small the memory window is. In fact, the memory window can be opened all the way so that all dataflow between disk and memory vanishes. However, this is wasteful of memory when the number of latitude rows per processor is large since the memory window requires three time levels and only two time levels are needed on disk (or ramdrive).
- Increases in speed can be realized when opening up the memory window on a single processor. This is the case when lots of diagnostic options are enabled. The reason is that less redundant computations are required when the memory window is opened up.
- All 3D parameterizations are easily parallelizable with a minimum of communication calls.

The only disadvantage of opening the MW up beyond the minimim size is that memory will be aggressively consumed.

11.3.1 Detailed anatomy

The inner workings of the memory window will now be exposed so it will become clear how the window is used for multi-tasking. The memory window is a chunk of memory dimensioned to hold a few latitude rows of 3-D prognostic data plus some workspace. A detailed schematic of the minimum sized MW appropriate for second order numerics (i.e.. numerics which requires access to nearest neighboring cells) is given in Figure 11.2a. Arrays are dimensioned within the MW to hold two prognostic variables (velocity and tracer) plus a workspace area. Each prognostic variable is an array subscripted by 5 indices[7]. The first three *i,k,j* are used to denote longitude, depth, and latitude. The fourth denotes prognostic component (e.g. for velocity, a 1 pertains to the zonal component of velocity and a 2 pertains to the meridional component. For tracers, a 1 pertains to temperature and a 2 pertains to salinity) and the fifth index which denotes time level (e.g. typically, this index takes on values of $\{-1, 0, 1\}$ to represent discrete time levels $\{\tau - 1, \tau, \text{ or } \tau + 1\}$.). The latitude rows for which 3-D prognostic equations are solved (updated to time level $\tau + 1$) are indicated in red. The remaining latitude rows indicated in blue are used as buffer rows for boundary conditions on the computed row.

A work area is also indicated as part of the memory window. The storage required for this area varies depending on which options are enabled and may be comparable to the storage required for prognostic arrays. In general, space is needed to hold diffusive and advective fluxes of prognostic quantities defined on the faces of each cell. Arrays for these fluxes are also

[7]Reasons for this ordering are given in Section 11.2.

subscripted by indices *i,k,j* but without a fourth or fifth index since they are recalculated for each prognostic variable to conserve memory. Additionally, some options require diffusive coefficients with spatial dependence. In this case, three dimensional arrays are used for diffusive coefficients which are also defined on cell faces.

A simplified schematic of the MW is given on the right side of Figure 11.2a by collapsing all indices except for the local latitude index "j". Two important characteristics are indicated: the MW contains $jmw = 3$ latitude rows and prognostic equations are integrated to update prognostic variables to time level $\tau + 1$ starting on local row $j = 2$ and ending on local row $j = 2$. Rows on which prognostic equations are solved are indicated by an "'x" and buffer rows which hold data one row away are indicated by a "circle" in the simplified schematic. An example of a larger MW is given in Figure 11.2b. In this example, the window holds $jmw = 6$ rows and prognostic equations are solved for local rows $j = 2$ through $j = 5$. In both windows, local rows given by $j = 1$ and $j = jmw$ are needed as boundaries for the second order numerics. Note that the MW is always symmetrical in that there are as many boundary rows to the north of the computed rows as to the south. Why aren't prognostic equations solved on rows $j = 1$ and $j = jmw$? Because indexing into arrays required by the 2nd order numerics would reach outside the MW (i.e. $j < 1$ or $j > jmw$).

In general, the number of rows for which prognostic equations are solved within a MW is given by the size of the MW minus the number of northern and southern boundary or buffer rows "jbuf". Equations for intermediate quantities such as density or fluxes on cell faces are solved on these extra rows. However, details about which quantities are solved and where they are defined on the grid system within the MW are not needed here and are covered more thoroughly in Section 22.2.4. The important point for now is that the number of computed rows "ncrows" where prognostic equations are solved within the MW is given by

$$ncrows = jmw - 2 * jbuf \tag{11.1}$$

and prognostic equations are solved on every row within the MW from *j=js_calc* through *j=je_calc* where

$$js_calc = 1 + jbuf \tag{11.2}$$

$$je_calc = jmw - jbuf \tag{11.3}$$

In the case of second order numerics, jbuf=1, the minimum sized window is jmw=3, and there is only one row on which prognostic equations are solved: $js_calc = je_calc = 2$. For a larger window where $jmw > 3$, prognostic equations are solved from MW rows $js_calc = 2$ through $je_calc = jmw - 1$.

For third and fourth order numerics, nearest neighbors of nearest neighbor cells are required which means a larger minimum sized MW is needed. An example is given in Figure 11.3a which is the minimum size required when executing with third or fourth order numerics. In this case, $jbuf = 2$, the window holds $jmw = 5$ rows, and prognostic equations are solved from rows $js_calc = 3$ through $je_calc = 3$. As with the 2nd order MW, equations for intermediate quantities are solved on the buffer rows as long as the indexing does not reach outside of the MW.

11.3.2 Solving prognostic equations within the MW.

Refer to the example in Figure 11.4a. On disk (or ramdrive), there are $jmt = 8$ latitude rows arranged monotonically from south to north with $jrow = 1$ representing the southernmost row

and $jrow = jmt$ the northernmost one. Notice also that the first and last rows are marked with starting and ending indices "jstask" and "jetask". These define the limits of the processor's task and the processor does not reference data outside of these limits. The other set of indices "jscomp" and "jecomp" mark the starting and ending rows on which prognostic equations are solved (computed).

A minimum sized MW for 2nd order numerics is shown accessing data from latitude rows on disk (or ramdrive). Note that the global index $jrow = 4$ corresponds to local index $j = 2$ within the MW. The arrows pointing from the disk (or ramdrive) to the MW indicate that τ and $\tau - 1$ data in local MW rows $j = \{1, 2, 3\}$ are from disk latitude rows $jrow = \{3, 4, 5\}$. Prognostic equations are solved on latitude row $j = 2$ within the MW. The arrow pointing from the MW to the disk (or ramdrive) indicates that updated data at time level $\tau + 1$ is written back to disk over the $tau - 1$ slot. Note that the relation between local index j and global index $jrow$ is given by the offset $joff$ in the figure. In general, for any local index j, the mapping to global index $jrow$ is

$$jrow = j + joff \tag{11.4}$$

where offset $joff$ is simply how far the memory window has been moved northward from the latitude given by "jstask".

In general, prognostic equations are solved on all jrows from "jscomp" to "jecomp" by a northward moving MW which solves a small group of rows (usually one) at a time. All steps are illustrated in Figure 11.4b. The first MW loads data from $jrow = \{1, 2, 3\}$ and the offset is $joff = 0$. Prognostic equations are solved for local index $j = 2$ within the MW and written back to the $jrow = 2$ slot on the $\tau - 1$ disk (or ramdrive). Then the MW is moved northward by the following operation. All data within the MW on row $j = 2$ is copied to row $j = 1$ followed by all data on row $j = 3$ being copied into row $j = 2$. This is explained more fully below. Then two time levels of disk data from global latitude row with index $jrow = 4$ are read from disk (or ramdrive) into MW row $j = 3$ and the offset is bumped up by the number of computed rows which in this case is one. Therefore, $joff = 1$ for the second MW. After prognostic equations are solved for row $j = 2$, the updated data is written back to the $jrow = 3$ slot on the $\tau - 1$ disk (or ramdrive). The process continues until all rows from $jscomp = 2$ to $jecomp = jmt - 1$ have been updated to $\tau + 1$ values on the $\tau - 1$ disk (or ramdrive).

Figure 11.5 is the counterpart to Figure 11.4 for a fourth order window. Since, third and fourth order numerical schemes require two additional rows in the MW, the minimum size of the MW is *jmw=5*. Using five rows, the essential point is to calculate second order quantities on the three central rows $j = \{2, 3, 4\}$. Meridional differences of these second order quantities yield a fourth order result defined at the central row[8] $j = 3$. All schemes which use nearest neighbors of nearest neighbor cells require option *fourth_order_memory_window* which is automatically enabled in file *size.h* when any of the existing fourth order schemes are enabled. Any new parameterization which requires the above 4th order differencing must also enable option *fourth_order_memory_window*. In contrast to the 2nd order MW, $joff = -1$ for the first MW and prognostic equations are solved for the central row which is $j = 3$. Note also, that the $j = 1$

[8]When uni-tasking, since calculations proceed from south to north and the southern most latitude is land, the second order quantity can be set to zero at *jrow=1*. This may be utilized to reduce the minimum size of the MW to *jmw=4* which makes the MW assymetrical. This is no longer allowed because assymetrical MW's lead to complications when multi-tasking. To make the assymetrical MW work when multi-tasking requires extra communication calls and the management of the MW becomes more complicated. For simplicity, the MW is required to be symmetric with the same number of buffer rows to the north and south of the computed rows. The price is 25% more memory for the symmetric MW which buys simplicity and generality.

row in the first MW hangs below the physical domain which starts at $jrow = 1$. Data from row $j = 2$ is replicated in row $j = 1$ to fill this first MW. As can be seen in Figure 11.5b, prognostic equations are still solved from rows $jscomp = 2$ to $jecomp = jmt - 1$ as in the 2nd order MW case.

11.3.3 Moving the memory window

After prognostic equations have been integrated on rows within the MW and these rows have been written to disk (or ramdrive), the MW must be moved northward before computation can begin on the next group of rows. The movement takes place by copying quantities from the northernmost rows into the southernmost rows of the MW and then filling the remaining rows with data from disk (or ramdrive). The number of rows to be copied depends on the order of the MW. The ordering of the copies is important else data copied in one operation will be wiped out by the next copy. Which rows are to be copied is intimately tied into how arrays are dimensioned.

All arrays dimensioned within the MW must have their latitude dimension fall into one of the following four catagories:

```
dimension A(,,jmw)          !  all cells within the MW
dimension B(,,1:jmw-1)      !  all cells except j=jmw
dimension C(,,2:jmw)        !  all cells except j=1
dimension D(,,2:jmw-1)      !  all cells except j=1 and j=jmw
```

For further explanation and some examples of how array dimensions are determined, refer to Section 22.2.4. Assuming the above array structures, the copy part of the northward movement of the MW is demonstrated by the following segment of code which should be viewed in conjuction with Fig 11.6 to illustrate how the copy occurs for various array shapes and MW configurations. After prognostic equations have been solved on rows marked by "x", $\tau + 1$ data from those rows is written to disk. Figure 11.7 illustrates the copy and move operations when option *pressure_gradient_average* is enabled. Note that tracers are computed on one row to the north of where baroclinic velocities are computed. The extra row is to allow for a four point average of pressure gradients when computing baroclinic velocities. When option *pressure_gradient_average* is enabled, only rows for which baroclinic equations are solved are written to disk (or ramdrive). Updated tracers on the extra row are copied into the next MW and get written to disk (or ramdrive) with baroclinic velocities from that MW.

```
for j=1,num_rows_to_copy
     jfrom = jmw - num_rows_to_copy + j  !  MW row to copy data from
     jto   = j                           !  MW row to copy data to
     do k=1,km                           !  for arrays dimensioned (1:jmw)
         do i=1,imt
             A(i,k,jto) = A(i,k,jfrom)
         enddo
     enddo
     if (jfrom .le.  jmw-1) then         !  for arrays dimensioned (1:jmw-1)
         do k=1,km
             do i=1,imt
                 B(i,k,jto) = B(i,k,jfrom)
             enddo
```

```
            enddo
        endif
        if (jto .ge.  2) then                !  for arrays dimensioned (2:jmw)
            do k=1,km
                do i=1,imt
                    C(i,k,jto) = C(i,k,jfrom)
                enddo
            enddo
        endif
#if defined fourth_order_window            !  for arrays dimensioned (2:jmw-1)
        if (jto .ge.  2 .and.  jfrom .le.  jmw-1) then
            do k=1,km
                do i=1,imt
                    D(i,k,jto) = D(i,k,jfrom)
                enddo
            enddo
        endif
#endif
enddo
```

Note that the number of copied rows is typically[9]

$$num_rows_to_copy = 2 * jbuf \tag{11.5}$$

which is the total number of buffer rows in the MW. The number of buffer rows required on the northern side and the southern side of the MW depends on the order of the MW (i.e. $jbuf = n/2$ for an n-th order MW). Note also that data in array "D" needs to be copied only when the MW is fourth order. After these copies have been made, the remaining rows in the MW from $j = num_rows_to_copy + 1$ through $j = jmw$ must be read from disk (or ramdrive) to fill up the MW before prognostic equations can be solved.

11.3.4 Questions and Answers

Here are some questions and answers about memory window and related issues:

- How much space does the memory window require?

  There are 12 arrays required to hold three time levels of prognostic data for two horizontal velocity components and two tracers within the memory window. The arrays are dimensioned as A(imt,km,jmw). For the simplest options, the workspace part of the memory window must allow space for 3 advective velocities on the faces of T-cells, 3 advective velocities on the faces of U-cells, 3 temporary arrays for use as fluxes on cell faces, 1 array for density, and 2 arrays for land/sea masks. If it is assumed that all of these 12 arrays in the workspace part are dimensioned as above, then 50% of the space within the memory window is workspace and 50assumption when the memory window is fully opened (jmw=jmt for second order numerics). The percent of space taken by workspace arrays is significantly less when the window is minimum (jmw=3). But the minimum window is not the constraining configuration.

[9] When option *pressure_gradient_average* is enabled, the value of *num_rows_to_copy* must be increased by one.

- What are the basic ways to use the memory window and how do the space requirements compare?

  Refer to Fig 11.8a which is a typical situation when MOM 3 is configured to solve a problem which is too big to fit into memory on a conventional vector platform such as the CRAY T90 at GFDL. All latitude rows are placed on disk. This is viable only if the disk is a solid state device such as SSD on the CRAY[10]. The option for keeping data on solid state disk on the CRAY T90 is *ssread_sswrite*.

  Figure 11.8b indicates what happens when option *ramdrive* is used instead of option *ssread_sswrite*. This option is appropriate for platforms that do not have solid state disk. Note that in comparison to figure 11.8a, the memory has increased dramatically and no disk space is required. The ramdrive is just an array dimensioned large enough to hold all latitude rows. Reads and writes to the ramdrive are replaced by memory to memory copies.

  Figure 11.8c is the situation when the memory window is fully opened up. The options for this configuration is *radmrive* and *max_window*. In this case, all latitude rows are stored directly in the memory window. For large models with many latitude rows, the fully opened memory window takes three times as much memory space as the case in figure 11.8b. For more detail, refer to Section 38.1.

- When the memory window is opened all the way, is the global index "jrow" the same as the local window index "j"?

  Yes and no. When fully opened, the number of latitude rows in the memory window is given by $jmw = (jmt - 2) + 2 * jbuf$. Consider the case of one processor: global arrays are dimensioned as A(imt,1:jmt) and memory window arrays are dimensioned as B(imt,km,1:jmw). For a second order window (when equations require second order numerics), $jbuf = 1$ so $jmw = jmt$ and $jrow = 1$ references the same latitude row as $j = 1$ in the memory window array.

  However, the same is not true for a fourth order window. The reason is that $jbuf = 2$ which makes the memory window arrays dimensioned as B(imt,km,1:jmt+2) while global arrays are still dimensioned as A(imt,1:jmt). The relationship between indices is such that $jrow = 1$ in the global arrays references the same latitude row as $j = 2$ in the memory window arrays.

  To make the local and global latitude indices the same for all cases, the memory window arrays would have to be made allocatable and dimensioned as B(imt,km,jscomp-2*jbuf:jecomp+2*jbuf) where "jscomp" and "jecomp" are a function of processor. Allocating these arrays as such currently slows the model down by 30%.

- Can space be saved by dimensioning workspace arrays as B(imt,km) instead of B(imt,km,jmw)?

  Yes. All "j-loops" could be replaced by one large "j-loop" wrapped around all routines needed to compute internal modes and tracers. However, there are problems with this technique when message passing on multiple processors. Essentially, the computation becomes "serialized" when processor "n" must wait for adjacent processor "n-1" to finish before inter-processor communication can be used to extract correct quantities at domain boundaries. This happens, for instance, when quantities are computed incorrectly at domain boundaries because the proper inputs are unavailable. The standard solution

[10] Usually the wait time for rotating disk significantly affects efficiency. The option for using rotating disk is *fio*.

is to replace incorrectly computed quantities with correct ones from adjacent domains via inter-processor communication. Alternatively, the memory window can be bumped up to the next higher order to include more buffer rows. Another problem with using the single "j-loop" technique is the arrangement is not flexible enough to cover all possibilities. For instance, option *pressure_gradient_average* which requires tracers to be solved to the north of velocity cells before baroclinic velocities are solved is problematic.

- Can moving of data within the memory window be eliminated by using indices and rotating the indices as the window is moved northward?

  Yes. If a new set of indices "jm1,jc,jp1" are introduced to refer to rows "j-1,j,j+1" in the memory window, then instead of moving data, the new indices can be rotated. This technique would also save memory. There are two disadvantages: for higher order schemes, new indices need to be introduced (i.e. "jm2" and "jp2" for forth order schemes); and the scheme is prone to errors. For instance, if an array is referenced with "j+1" instead of "jp1", then errors result.

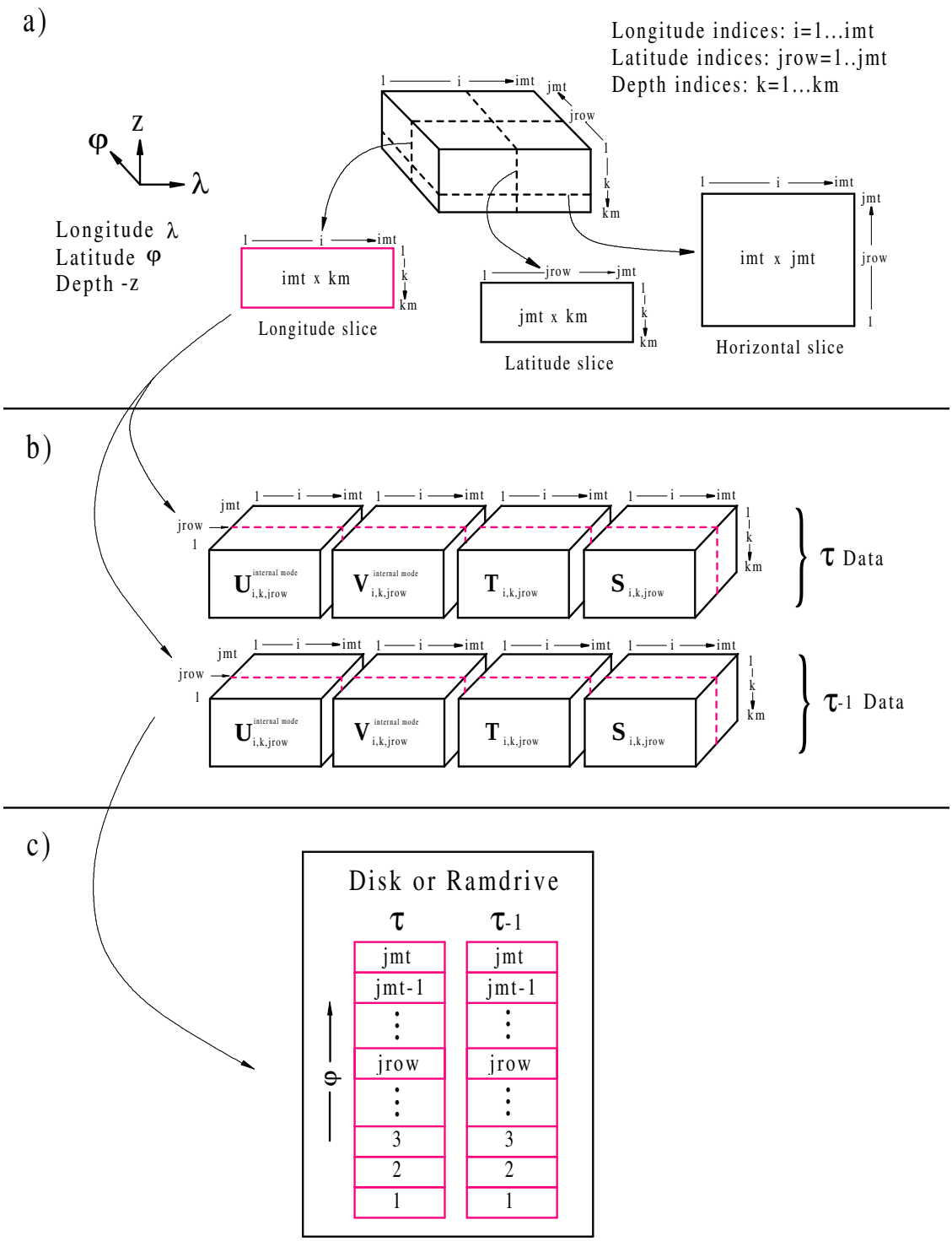


Figure 11.1: a) Slicing a volume along surfaces of constant latitude, longitude, and depth showing relative sizes of slices. b) A longitudinal slice through two time levels of prognostic data on disk or ramdrive with latitude index "jrow". c) Schematic arrangement of all slices arranged by latitude index "jrow" on disk or ramdrive area.

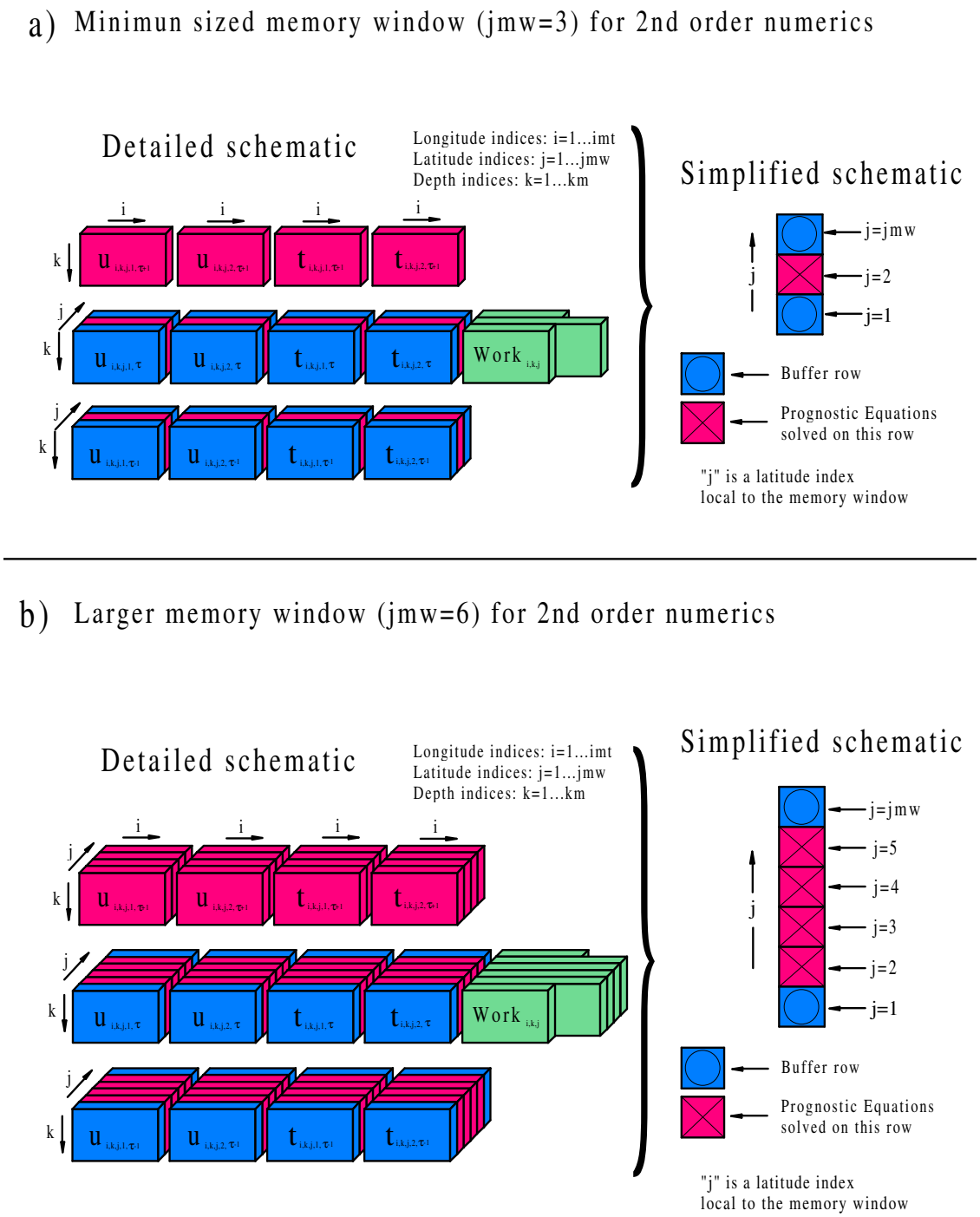


Figure 11.2: a) Detailed and simplified schematic of the minimum sized memory window with $jmw = 3$ latitude rows for 2nd order numerics. b) Memory window opened up to hold $jmw = 6$ latitude rows.

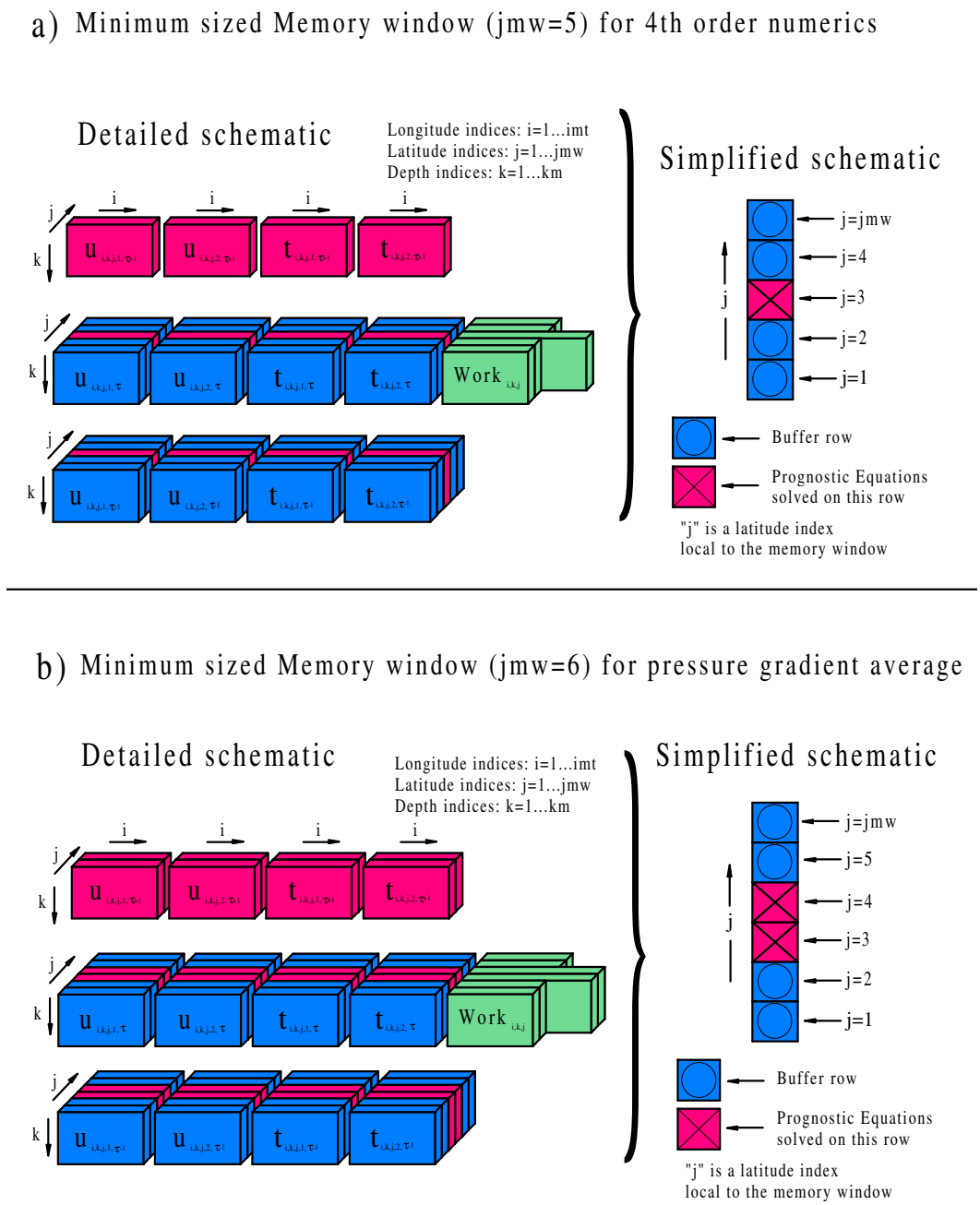


Figure 11.3: a) Minimum sized memory window for 4th order numerics.

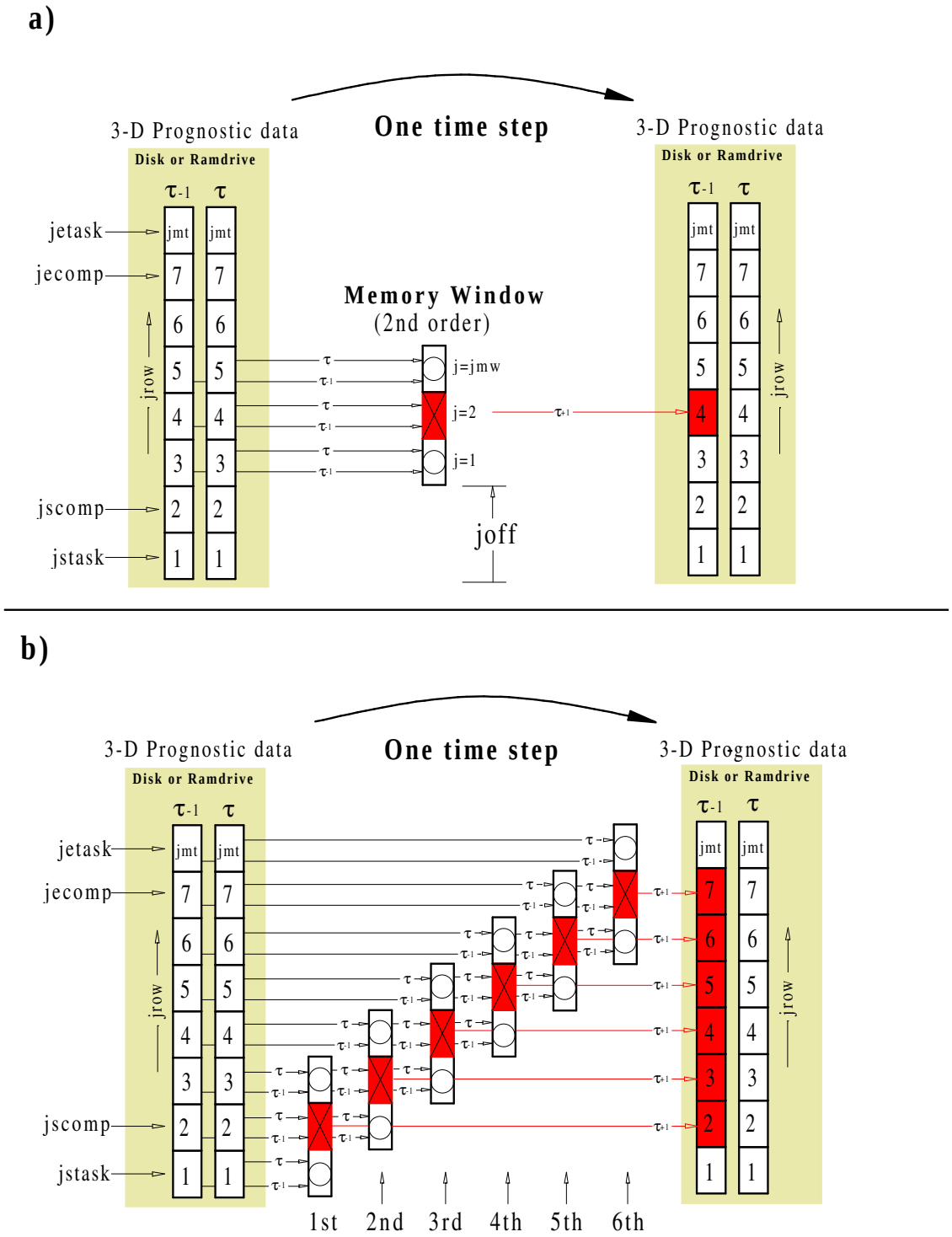


Figure 11.4: a) Loading the memory window, solving prognostic equations on the central row and writing $\tau + 1$ data back to $\tau - 1$ disk for latitude row *jrow* $= 4$. b) Solving prognostic equations for all latitude rows from "jscomp" to "jecomp" by moving the memory window northward.

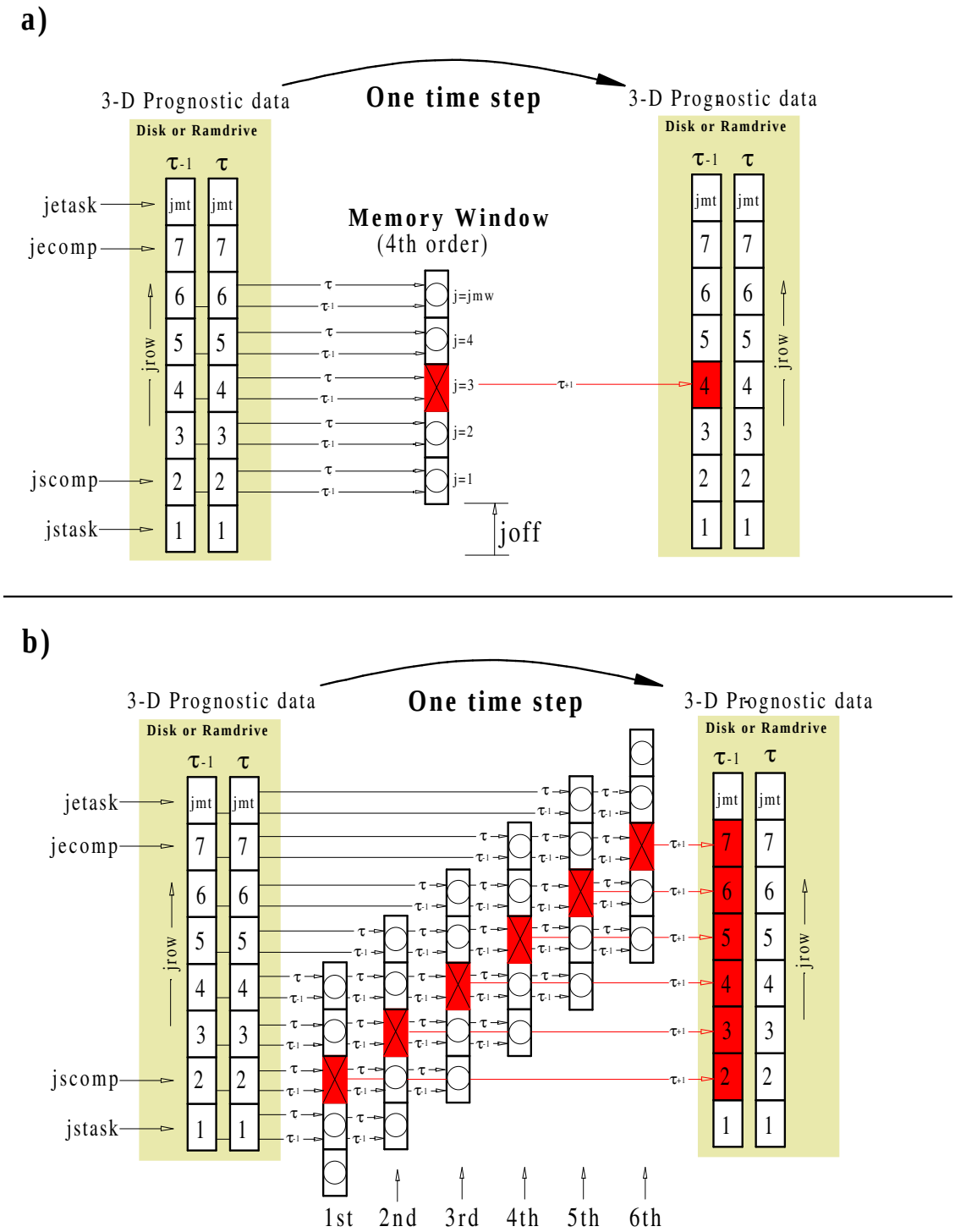


Figure 11.5: a) Loading the memory window, updating the central row and writing to disk for latitude row *jrow* = 4. b) Updating all latitude rows by moving the memory window northward.

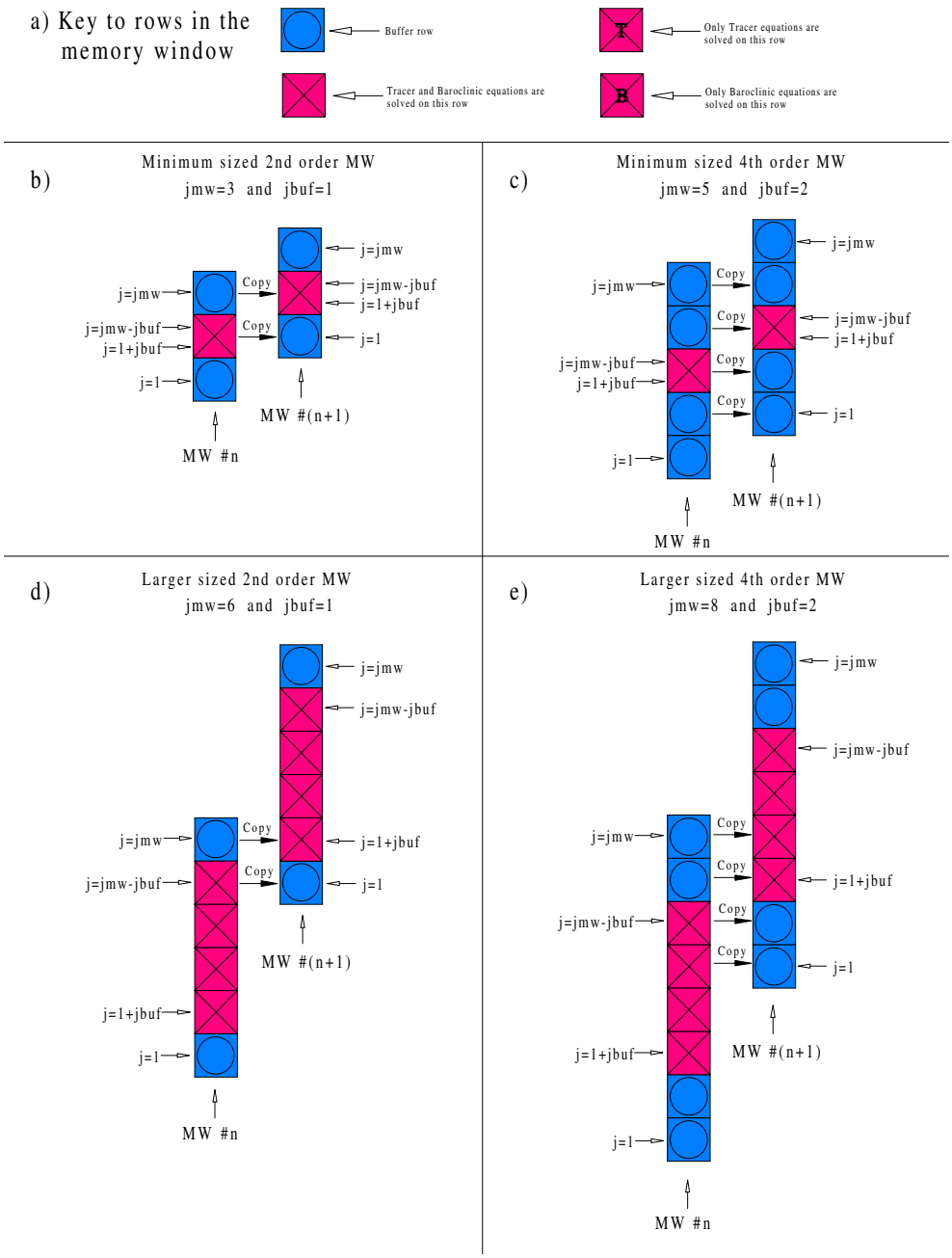


Figure 11.6: The memory window is moved northward by copying data from the northernmost rows into the southernmost rows followed by reading of new data. a) Various types of rows within the memory window. b) Copy operation for a 2nd order memory window with jmw=3. c) Copy operation for a 4th order memory window with jmw=5. d) Copy operation for a larger 2nd order memory window. e) Copy operation for a larger 4th order memory window.

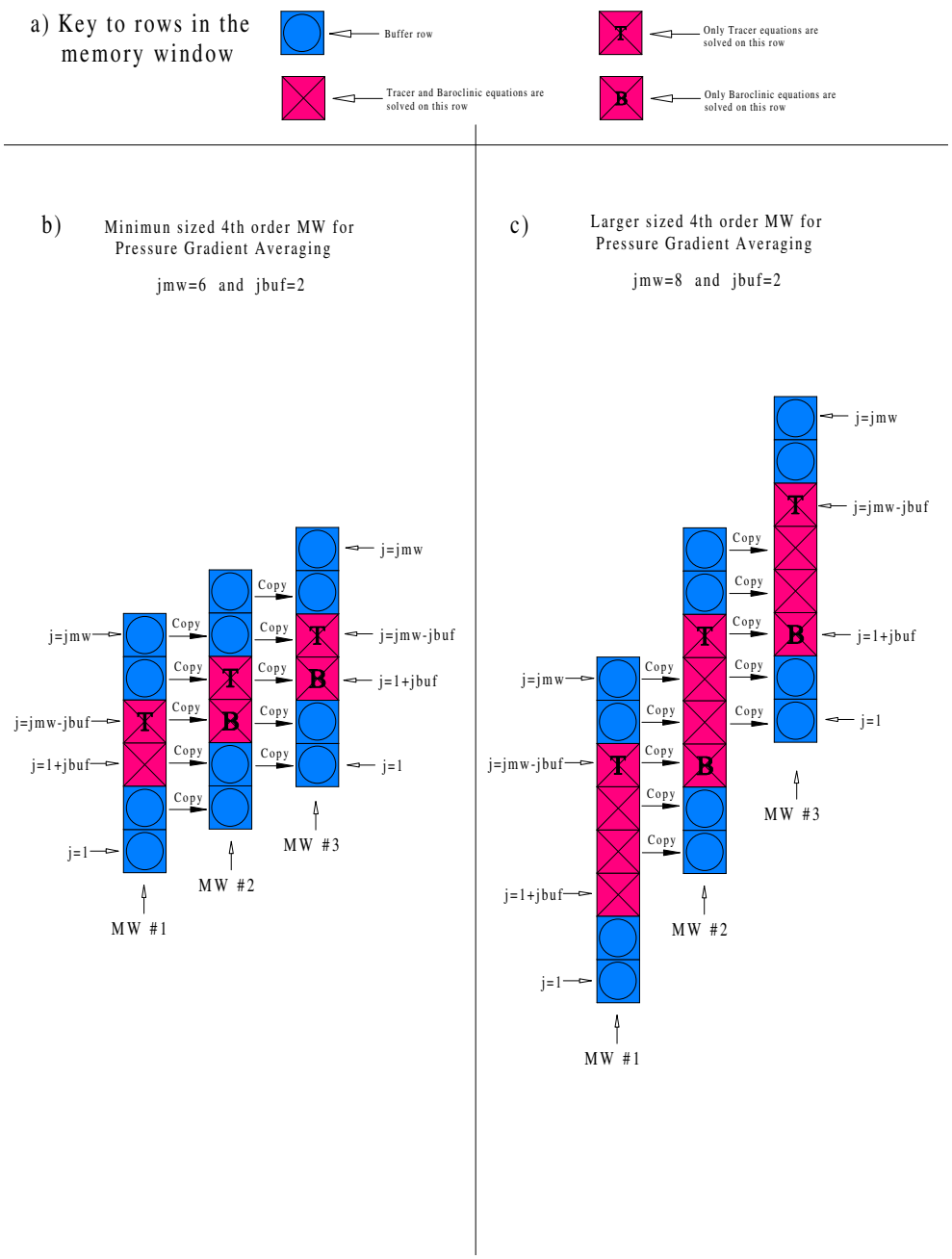


Figure 11.7: The memory window is moved northward by copying data from the northernmost rows into the southernmost rows followed by reading of new data. Note that only rows where baroclinic velocities are computed are written back to disk (or ramdrive). a) Various types of rows within the memory window. b) Copy operation for the first three memory windows of minimum size jmw=6 when option *pressure_gradient_average* is enabled. c) Copy operation for the first three memory windows of a larger size jmw=8 when option *pressure_gradient_average* is enabled.

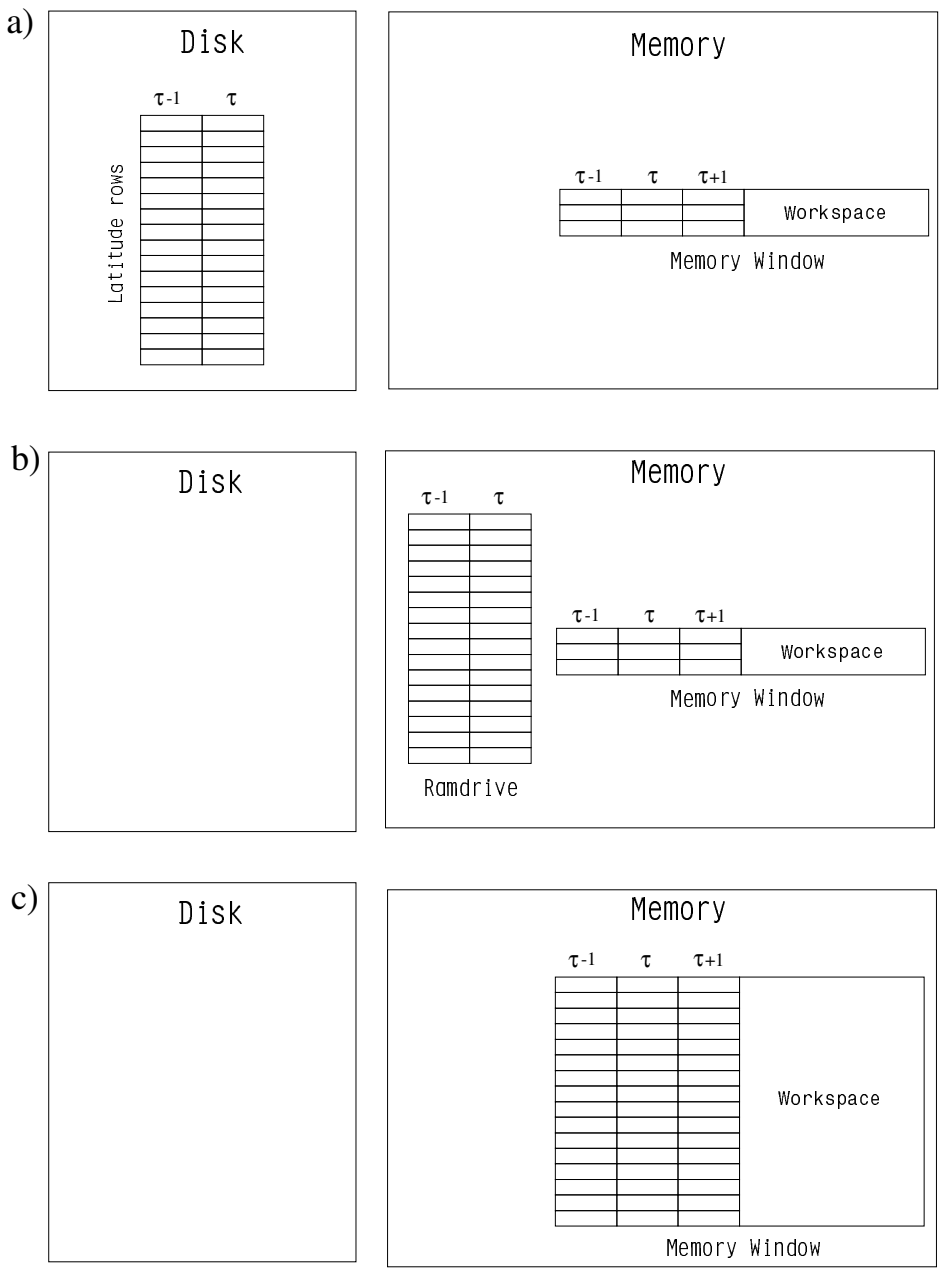


Figure 11.8: Various ways of configuring MOM illustrating the space used on disk and in memory. a) Options *fio* and *ssread_sswrite*. b) Option *ramdrive*. c) Options *ramdrive* and *max_window*.

Chapter 12

Multi-tasking

Since multi-tasking is a straightforward extension of uni-tasking, familarity with uni-tasking as described in Chapter 11 is assumed.

12.1 Scalability

When multi-tasking, multiple processors are deployed against a job to reduce the job's overall wall clock time. The cpu time (summed over all processors) is not reduced[1] compared to uni-tasking. The degree to which multiple processors are able to reduce a job's wall clock time is a measure of the job's parallel efficiency or scalability. For instance, if a job, which takes 10 hours of wall clock time on one processors, can be divided into pieces so that it will take one hour using 10 processors, then the job is 100% parallel. The job is then said to scale well which means it scales linearly with the number of processors. It should be kept in mind that a prefectly linear scaling with the number of processors assumes a dedicated environment. In a non-dedicated environment, other executables will steal processors and the job's captured processors must wait for stolen processors to be re-claimed before continuing. The wait time degrades scaling.

How parallel must a job be? Almost linear scaling may be good enough for a small number of processors, but it is not for large numbers of processors. For instance, if a job is 99.5% parallel, it will saturate about 14.8 out of 16 processors. That means the speedup from 16 processors is about 15 times as fast as the uni-tasked job. However, the same job will only give a speedup of 49 on 64 processors and on 1024 processors, the speedup is only 168! This is a consequence of Amdahl's law.

12.2 When to multi-task

If the number of processors exceed the number of jobs in the system, some processors stand idle and overall system efficiency degrades unless jobs are multi-tasked. In an operational environment, if a forecast is required every 4 hours but takes 8 hours on a single processor, then multi-tasking will be useful. Other examples include long running experiments that take too long to complete or when it's impossible to exhaust a monthly computer time allocation using one processor. In environments where multiple processors are dedicated to a single job, it is

[1] In fact, it may increase. However, the focus is changed from cpu time to wall clock time. Wall clock time is the time it takes to get results back.

essential to attain high parallel efficiency otherwise system efficiency and overall throughput suffers. In the past, multi-tasking has not been used in any significant way at GFDL. The reason is that there have always been enough jobs in the system to keep all processors busy. Typically, in the middle of 1997, there were about 30 batch jobs in GFDL's Cray T90 at all times and the system efficiency averaged over 24 hours was about 93%. At the point where a system becomes under saturated, overall system efficiency will drop significantly unless multi-tasking is used.

12.3 Approaches to multi-tasking

Two approaches to multi-tasking MOM have been experimented with at GFDL. They were:

- The *fine grained* approach to multi-tasking where parallelism is applied at the level of each *nested do loop*. This is also known as "autotasking". All processors work simultaneously on each *nested do loop* and there are many parallel regions.
- The *coarse grained* approach to multi-tasking where parallelism is applied at the level of latitude rows. For example, all work associated with solving equations for one latitude row is assigned to one processor. All work associated with solving the equations for another latitude row is assigned to a second processor . . . and so forth. Then, all processors work simultaneously and independently. This implies that there is only one parallel region.

In a nutshell, the fine grained approach to multi-tasking gives about 85% parallelism. That translates into a speedup of a little less than 4 on 8 processors. Why is the parallel efficiency so low? The reason is basically due to not enough work within the many parallel regions and the fact that the range on the parallel loop "j" index is not the same for all loops. Increasing the model resolution can somewhat address the first problem but not the second. The result is that processors stand idle and the parallel performance degrades. This is clearly not a long term solution. The coarse grained approach is the best and yields a near linear speedup with the number of processors.

12.4 The distributed memory paradigm

A distributed memory paradigm is one where each processor has its own chunk of memory but the memory is not shared among processors. This means that an array on one processor cannot be dimensioned larger than the processor's local memory and the processor cannot easily access arrays dimensioned in other processors memory. Accessing arrays in other processor's memory is possible by making "communication" calls to transfer data between processors.

MOM uses a distributed memory paradigm. The method builds on the coarse grained approach and assumes that both baroclinic and barotropic parts are divided among processors with distributed memories and therefore "communication" calls must be added to exchange boundary cells between processors at critical points within the code. The "communication" calls are made via a message passing module which supports SHMEM as well as MPI protocols. For details, refer to http:/www.gfdl.gov/ vb. The advantages of the distributed paradigm are:

1. Higher parallel efficiency is attainable.
2. No complicated "ifdef" structure is needed to partition code differently for uni-tasking or multi-tasking on shared or distributed memory platforms.
3. Only two time levels are required for 3-D prognostic data on disk (or ramdrive) as opposed to three time levels with the coarse grained shared memory method.

Most options have been tested in parallel. It bears stating that when an option is parallelized, it means that answers are the same to the last bit of machine precision regardless of the number of processors used[2]. Some options will probably never be parallelized. One example is the stream function method. This method requires land mass perimeter integrals which cut across processors in compilicated ways and this is a recipe for poor scaling. The implicit free surface method is better since it does not require perimeter integrals. However, global sums are still required which do degrade scaling but not as much as the island integrals. Improvement can be made to the existing global sum reductions because they are only a crude first attempt. However, even if global sum reductions were no problem, the accuracy of the method depends on the number of iterations which is tied to the grid size[3]. The best scaling is achieved by the explicit free surface option which does not require any global sum reductions and the number of iterations (sub-cycles) depends on the ratio of internal to external gravity wave speed independent of the number of grid cells.

12.5 Domain Decomposition

If all arrays were globally dimensioned[4] on all processors, available memory could easily be exceeded with even modest sized problems. Each processor should only dimension arrays large enough to cover the portion of the domain being worked on by the specific processor. Refer to Fig 12.1a which is an example of a domain with $i = 1, imt$ longitides and $jrow = 1, jmt$ latitudes divided among 9 processors arranged in a 2d (two dimensional) domain decomposition. Fig 12.1b gives the arrangement for a 1d (one dimensional) domain decomposition in latitude using the same 9 processors. In both cases, two dimensional arrays need only be dimensioned large enough to cover the area worked on by each processor. As discussed below, this area will actually be one or two cells larger at the processor boundaries to include boundary cells required by the numerics. Boundary cells on each processor must be updated with predicted values from the domain of the adjoining processor. The arrows indicate places where communication takes place across boundaries between processors.

Figure 12.1c indicates three cases: one for 9 processors, 100 processors, and 900 processors. Within each case, three domain shapes are considered: $imt = jmt$, $imt = 2 * jmt$, and $imt = jmt/2$. The tables indicate how many communication calls are required. Since both processors which share a boundary require data from the other processor, two communication calls are required at each boundary and the same amount of words are transferred across the boundary for each processor. Two things are worth noting. First, as the number of processors increases, the number of words transferred in the 2d domain decomposition is much less than in the 1d domain decomposition. Second, for large numbers of processors, 1d domain decomposition requires half as many communication calls as 2d domain decomposition. An

[2]This is necessary otherwise science may become processor dependent.
[3]As the number of grid cells increase, the number of iterations must increase to keep the same level of accuracy.
[4]Dimensioned by the full number of grid cells in latitude, longitude, and depth.

additional problem for 2d domain decomposition which is not indicated in Figure 12.1c is that the equations are not symmetric with respect to latitude and longitude. Specifically, a wrap around or cyclic condition is placed on longitudes for global simulations. No such condition is needed along latitudes. The implication is that many additional communication calls will be needed to impose a cyclic condition on intermediate computations for 2d domain decomposition. These communication calls are not needed in a 1d decomposition in latitude because each latitude strip including the cyclic boundary is on processor. To lessen the need for extra communication calls in 2d decomposition, additional points can be added near the cyclic boundary thereby extending the domain slightly.

Whether 1d or 2d decomposition is better depends on the the latency involved in issuing a communication call versus the time taken to transfer the data. Another factor is whether polar filtering is used because it would also require extra communications in a 2d decomposition. At this point, the 1d domain decomposition of Fig 12.1a is what is done in MOM.

Executing the 3 degree global test case #0 for MOM on multiple processors of a T3E using 1-D decomposition in latitude indicates that scaling falls off quickly when there are fewer than 8 latitude rows per processors. This result has dire implications for climate models. For instance, a 2 degree global ocean model will have about 90 latitiude rows. Efficient scaling implies only 11 processors can be used. On the GFDL T3E, it takes about 10 processors to equal the speed of one T90 processor. If it is assumed that processors on the next system are 2.5 times as fast as the T3E processors <u>and</u> the ratio of network speed to processor speed stays the same, a two degree climate model will only gain a factor of about 3 in speed over a T90 processor. If the processor speed increases faster than the network speed, then the situation gets worse. The implication is that 2-D domain decomposition will be needed for climate modeling if an order of magitude speed up is to be attained..

On the other hand, a 1/5 degree global ocean model with 50 levels has 900 latitude rows and requires 2 gigawords of storage with a fully opened memory window. Using 9 rows per processor implies that 100 processors can be used with 1-D decomposition. Each processor must have at least 20 megawords. Realistically, by the next procurement, each processor will have at least 32 megawords of memory. If the GFDL T3E had enough processors, then a speedup of 25 times that of one T90 processor would be realized. It seems that 2-D decomposition is more important for coarse resolution models.

Since the majority of simulations with MOM have been carried out on vector machines at GFDL, the idea has been to compute values over land rather than to incur the cost of starting extra vectors to skip around land cells. In an earlier implementation, extra logic was in place to skip computations on land cells. It turned out that the speed increase on vector machines was not worth the extra coding complexity so the logic was removed. On scalar machines, it may be beneficial to skip computation over land areas. A 2d domain decomposition would in principle more easily allow computation over land to be skipped by not assigning processors to areas containing all land cells. In a 1d decomposition, extra logic would need to be added to eliminate computation over land cells.

12.5.1 Calculating row boundaries on processors

The number of processors is read in through namelist as variable *num_processors*. The model domain is divided into *num_processors* pieces with each piece containing the same number[5] of

[5]The important point is to have the same amount of work on each processor. If more work is required on some rows than others, then the number of rows on each processsor should be different. It is assumed here that the same amount of work is on every row and that each processor should get the same number of rows.

latitude rows. Each processor is assigned the task of working on its own piece of the domain starting with latitude index $jrow = jstask$ and ending with latitude index $jrow = jetask$. Each processor has its own memory window to process only those latitude rows within its own task. The starting and ending rows of each processor's task are given by

$$jstask = nint((pn-1) * calculated_rows) + (2 - jbuf) \tag{12.1}$$

$$jetask = nint(pn * calculated_rows) + (1 + jbuf) \tag{12.2}$$

where pn is the processor number ($pn = 1 \cdots num_processors$), the number of buffer rows "jbuf" is explained in Section 11.3.1, and

$$calculated_rows = float(jmt - 2)/num_processors. \tag{12.3}$$

The latitude rows for which the tracer and baroclinic equations are solved within the processor's task are controlled by the starting and stopping rows

$$jscomp = jstask + jbuf \tag{12.4}$$

$$jecomp = jetask - jbuf \tag{12.5}$$

12.5.2 Communications

Figure 12.2a gives an example of multi-tasking with $num_processors = 3$ processors and $jmt = 14$ latitude rows for a second order memory window. Note that latitude rows on disk (or ramdrive) for each processor in Figure 12.2b look like a miniture version of Figure 11.4 where $jstask = 1$ and $jetask = jmt$. For example, on the disk (or ramdrive) of processor #1, the global latitude index runs from $jrow = 1$ to $jrow = 6$. On this processor, the task limits are $jstask = 1$ and $jetask = 6$ and the limits for integrating prognostic equations are from $jscomp = 2$ through $jecomp = 5$. On processor #2, the task limits are $jstask = 5$ and $jetask = 10$ and the limits for integrating prognostic equations are from $jscomp = 6$ through $jecomp = 9$.

Apart from dividing up the domain into pieces, the new aspect in Figure 12.2 is communication between processors indicated by short arrows pointing to the boundary rows. In the second order memory window, there is one boundary row at the borders of each task. Look at latitude row $jrow = 6$ on the updated disk of processor #1. This row cannot be updated to $\tau + 1$ by processor #1's MW because data from $jrow = 7$ is needed. Instead, data at $\tau + 1$ from $jrow = 6$ on processor #2 is copied into the $jrow = 6$ slot of processor #1 by a call to the communication routine after all processors have finished working on their tasks. Similarly, $jrow = 5$ on processor #2 is updated with $\tau + 1$data from $jrow = 5$ on processor #1 and so forth.

The situation with fourth order numerics is similar except that more communication is required at the end of the timestep. This is illustrated in Figure 12.3a. Note that since $jbuf = 2$ for forth order windows, the task for processor #1 ranges from $jstask = 1$ to $jetask = 7$. For processor #2, the task ranges from $jstask = 4$ to $jetask = 11$, and so forth. However, the rows on which prognostic equations are solved within each task are the same as in the case with a 2nd order memory window. Additional communication is indicated by the extra arrows which are needed because there are now two buffer rows on the borders of each task (i.e. $jbuf = 2$).

For all processor numbers from $pn = 2, num_processors$ the following prescribes the required communication for second order numerics:

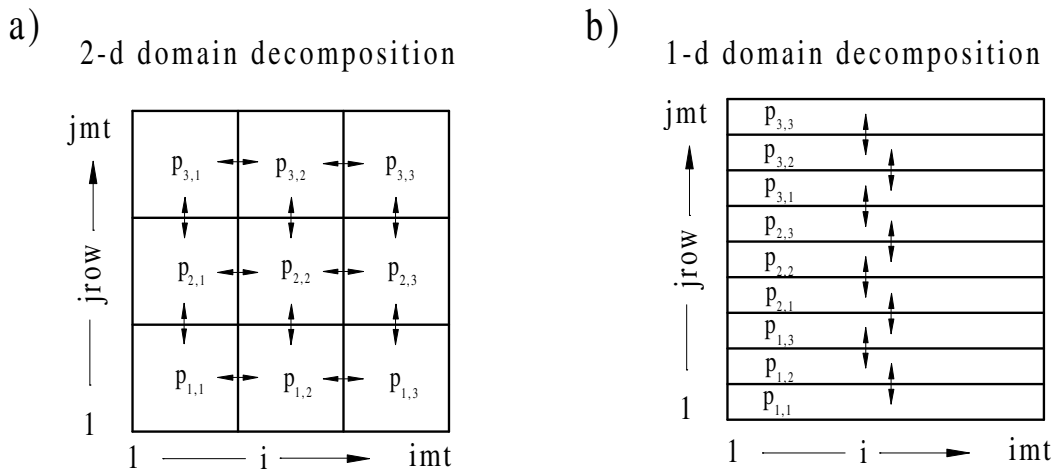
- copy all data from latitude row "jstask+1" on processor "pn" to latitude row "jetask" on processor "pn-1".
- copy all data from latitude row "jetask-1" on processor "pn-1" to latitude row "jstask" on processor "pn".

When a fourth order memory window is involved, the following communication is required:

- copy all data from latitude row "jstask+3" on processor "pn" to latitude row "jetask" on processor "pn-1".
- copy all data from latitude row "jstask+2" on processor "pn" to latitude row "jetask-1" on processor "pn-1".
- copy all data from latitude row "jetask-3" on processor "pn-1" to latitude row "jstask" on processor "pn".
- copy all data from latitude row "jetask-2" on processor "pn-1" to latitude row "jstask+1" on processor "pn".

12.5.3 The barotropic solution

The 2-D barotropic equation is divided into tasks in the same way as was done for the prognostic equation. Since the processor boundaries are the same, communication involves the same rows. Each processor dimensions arrays for only that part of the domain being worked on by the specific processor and the actual memory requirement is small. Therefore, within each processor's task no memory window is needed. All indexing into 2-D arrays is in terms of the absolute global index "jrow".



c)

3^2 processors

		Total number of words transferred		
	number of communication calls	when imt=jmt	when jmt = imt/2	when jmt = 2*imt
1-d	16	16*imt	16*imt	16*imt
2-d	24	8*imt	6*imt	12*imt

10^2 processors

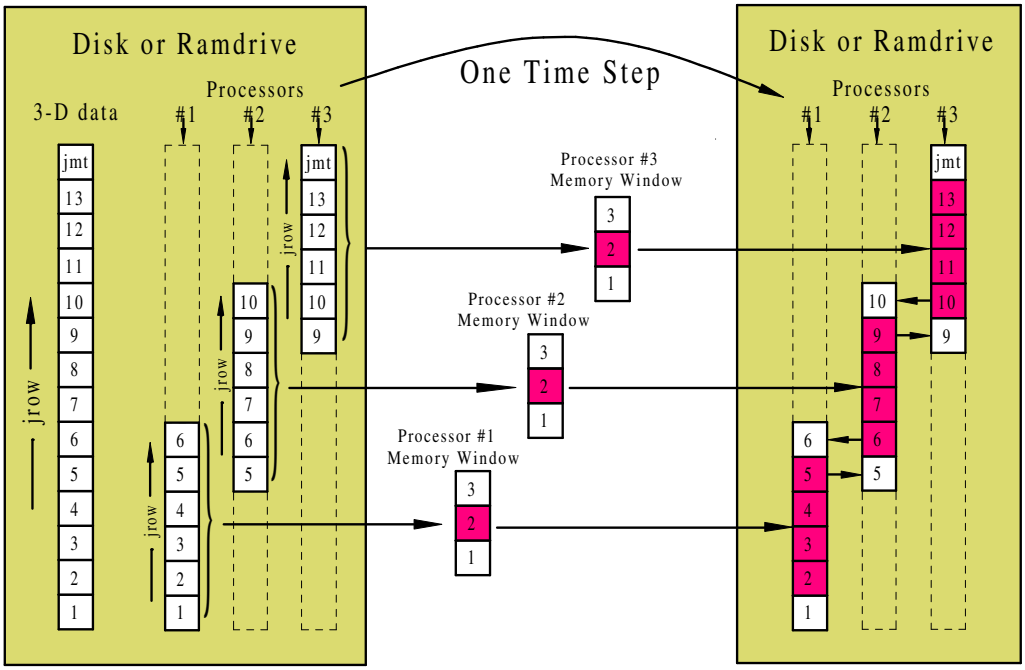
		Total number of words transferred		
	number of communication calls	when imt=jmt	when jmt = imt/2	when jmt = 2*imt
1-d	198	198*imt	198*imt	198*imt
2-d	360	36*imt	27*imt	54*imt

30^2 processors

		Total number of words transferred		
	number of communication calls	when imt=jmt	when jmt = imt/2	when jmt = 2*imt
1-d	1798	1798*imt	1798*imt	1798*imt
2-d	3480	116*imt	87*imt	174*imt

Figure 12.1: a) A 2d domain decomposition using 9 processors. b) Rearranging the 9 processors for a 1d domain decomposition in latitude. c) Comparison of 1d and 2d domain decomposition giving number of communication calls and words transferred for 9, 100, and 900 processors.

a) Multi-tasking with 3 processors and 2nd order numerics



b) Starting and ending row limits for each processor

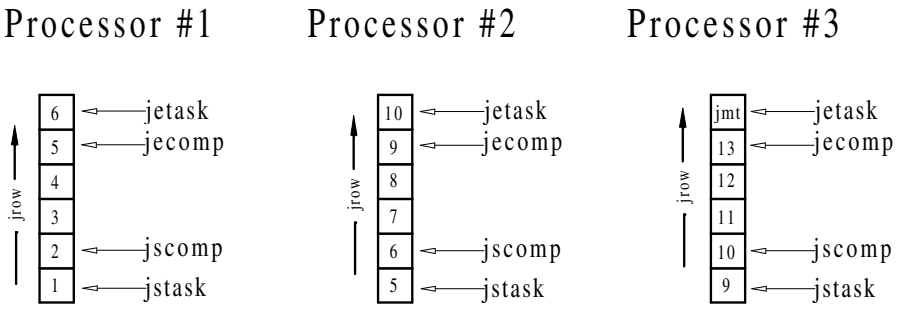


Figure 12.2: a) Integrating prognostic equations using 3 processors and 2nd order numerics. b) Task limits "jstask" and "jetask" as well as row limits "jscomp" and "jecomp" for each processor.

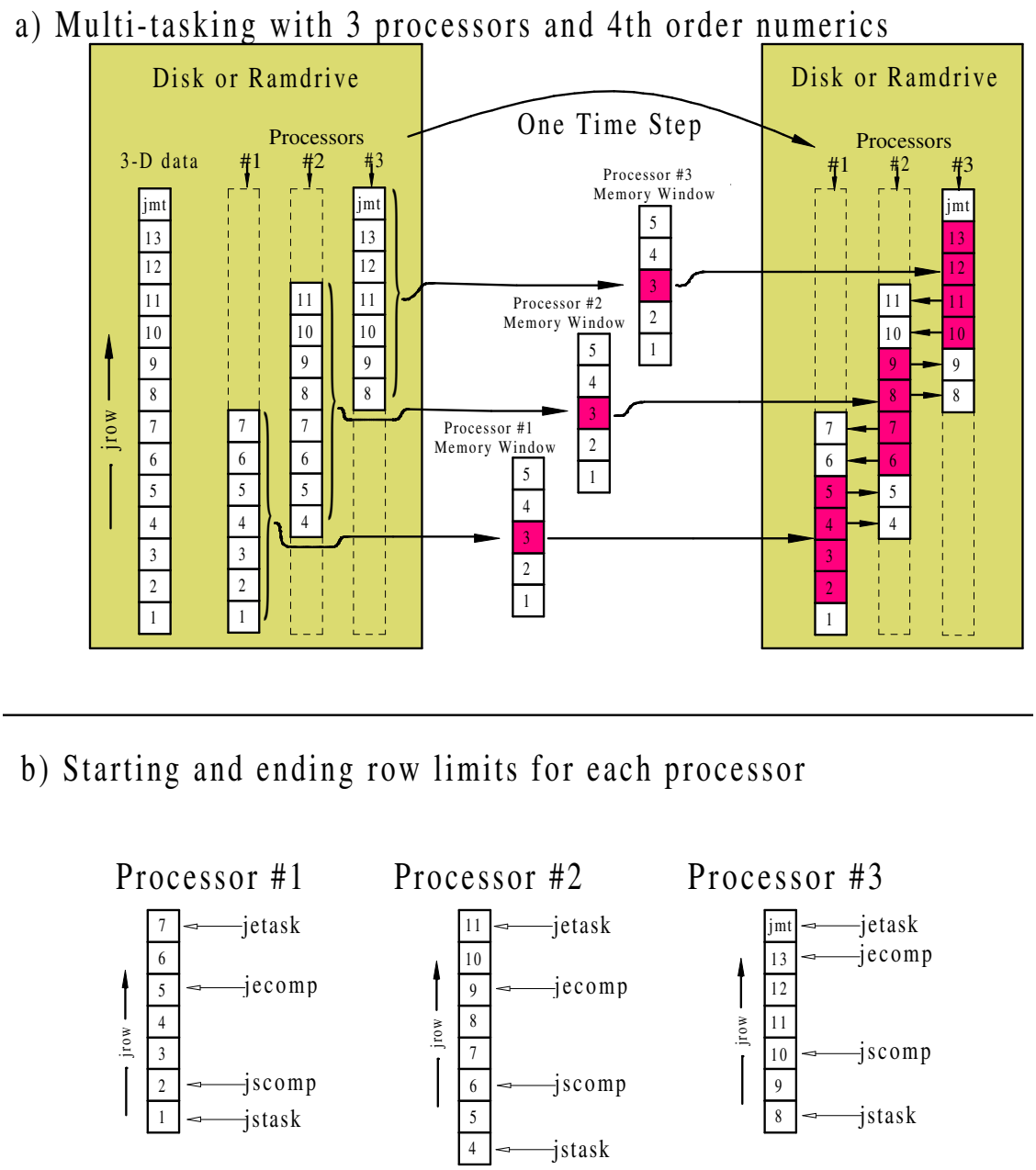


Figure 12.3: a) Integrating prognostic equations using 3 processors and 4th order numerics. b) Task limits "jstask" and "jetask" as well as row limits "jscomp" and "jecomp" for each processor.

Chapter 13

Database

A database is included with MOM. The database is not needed to start using MOM, although use of MOM is certainly limited without it. Refer to Chapter 3 for details on how to get this data and Section 3.2 for a description of programs which access it. This database is far from being complete. It should be considered as a starting point and act as a prototype for extensions using other datasets. GFDL is not a data center and there is no intention to extend this with other datasets.

13.1 Data files

The DATABASE contains approximately 160MB of IEEE 32bit data and the files are named as follows:

- *scripps.top* is the Scripps topography (265816 bytes).
- *hellerman.tau* is the Hellerman/Rosenstein (1983) windstress climatology (1757496 bytes).
- *oorts.air* is the Oort (1983) 1000mb air temperature climatology (294268 bytes).
- *levitus.mask* is the Levitus (1982) land/sea mask (8680716 bytes).
- *ann.temp* is the Levitus (1982) annual mean potential temperatures (8682036 bytes).
- *ann.salt* is the Levitus (1982) annual mean salinities (8682036 bytes).

  Then there are the Levitus (1982) monthly climatological potential temperatures. There are twelve monthly files where the three letter suffix gives the month.

- *jan.temp* is the Levitus (1982) monthly mean potential temperatures (4998748 bytes).

  The remaining eleven monthly temperature files are of the same size but use a different month as the suffix.

  Then there are the Levitus (1982) monthly climatological salinities. There are twelve monthly files where the three letter suffix gives the month.

- *jan.salt* is the Levitus (1982) monthly mean salinities (6314208 bytes).

  The remaining eleven monthly salinity files are of the same size but use a different month as the suffix.

Note that the file size for the monthly Levitus (1982) data is less than the file size for the annual means. This is because seasonal effects only penetrate down to 1000m or so and the monthly data only extends to this depth. Also, salinity has been interpolated from four seasonal values to twelve monthly values within this dataset. All Levitus (1982) data as well as Hellerman/Rosenstein (1983) and Oort (1983) data has been forced into land areas on their native grid resolutions by solving

$$\nabla^2 \xi_{i,jrow} = 0 \tag{13.1}$$

over land areas using values of $\xi_{i,jrow}$ over ocean areas as boundaries values where $\xi_{i,jrow}$ represents the various types of data. This method is similar to what was done with surface boundary conditions in Section 19.1.2.

Chapter 14

Variables

14.1 Naming convention for variables

The 6 character restriction on naming variables has been relaxed since MOM 2 although economy of characters in choosing names is always desirable. In an attempt to unify naming of variables, a convention is introduced which uses abbreviated forms of the fundamental conceptual pieces of the numerical oceanography. This convention also gives guidance for building the names for variables in future parameterizations and keeps an overall consistency throughout the model. Constructing names for variables is then a matter of assembling these abbreviations in various ways. Note that when names of variables are spelled out, the naming convention should not be applied. However, when names seem cryptic, the convention is intended to help decipher their meaning. A list of abbreviations is given below:

u =	Velocity point	d =	Delta	adv =	Advective
t =	Tracer point	f =	Flux	diff =	Diffusive
n =	North face	p =	Pressure	visc =	Viscous
e =	East face	c =	Coefficient	psi =	Stream function
b =	Bottom face	v =	Velocity	ps =	Surface pressure

Variables described subsequently within this chapter are build in terms of these abbreviations. All variables are positioned relative to the arrangement of grid cells discussed in Chapter 16. When an unknown variable is encountered in MOM, it may be possible to determine its meaning from context or the naming convention. For example, the variable *adv_vnt* is an advective velocity on the north face of T cells. If this fails, it may be useful to search all *include files* for comments using the UNIX command: *grep* as described in Section 3.1.

Operators are used in MOM to construct various terms in finite difference equations which are the counterpart to equations detailed in Chapter 4. Unlike arrays, operators are build using expressions which relate variables but require no explicit memory. The details of a particular operation are forced into the operator and buried from view which allows complicated equations to be written in terms of simple, easy to understand, pieces. At the same time, compilers can expand these pieces in-line into an un-intelligible mess (by human standards) and subsequently optimize it. In fact, there is no difference in speed on the CRAY (or any mature compiler) between writing an equation with operators and writing an equation with all terms expanded out. The big difference is that the former is easier to understand and work with for humans.

Instead of using operators, all terms in the equations could be computed once and stored in arrays. This would allow less redundant calculation but be prohibitive in memory usage. It is an interesting counter intuitive point that it is sometimes faster to do redundant calculation rather than save intermediate results in arrays. Depending on the speed of the load/store operations compared with the multiply/add operations, if redundant computation contains little work it may be faster to compute redundantly. In practice, this depends on the computer and compiler optimizations. This can be verified by executing script *run_timer* which exercises the timing utilities in module *timer* by solving a tracer equation in various ways. It is one measure of a mature compiler when the speed differences implied by solving the equations in various ways is relatively small. On the CRAY YMP the differences are about 10% whereas on the SGI workstation, they can exceed 50%.

Within operators, capital letters are reserved for the intended operation and small letters act as qualifiers. A list of abbreviations is given below:

ADV_T =	Advection of tracer	x =	longitude direction
DIFF_T =	Diffusion of tracer	y =	latitude direction
ADV_U =	Advection of velocity	z =	vertical direction
DIFF_U =	Diffusion of velocity		

As an example, tracer diffusion in the longitude direction at a given point is given by operator DIFF_Tx(i,k,j). These operators are defined in include files and may be located using the UNIX command *grep* as indicated in Section 3.1.

14.2 The main variables

14.2.1 Relating indices j and jrow

In MOM, index "jrow" is the index of a latitude row and it takes on values from "1" to "jmt". Index "j" refers to a latitude row within the memory window which takes on values from "1" to "jmw" which is the size of the memory window. The indices are related by a simple offset "joff" which measures how far the memory window has been moved northward as equations are being solved

$$j = jrow - joff \tag{14.1}$$

Why are some variables referenced by "j" where others are referenced by "jrow"? To save memory. Typically, one or two dimensional variables which are functions of latitude are dimensioned by "jmt" and referenced by index "jrow". Three dimensional variables are dimensioned by "jmw" and referenced with index "j". If memory size were no issue (but it is) then all variables could be dimensioned by "jmt" and referenced by one latitude index. When the memory window is opened up all the way then $jmw = jmt$, $joff = 0$, and $j = jrow$.

For a description of how the memory window moves northward, refer to Section 11.3.2. In MOM, whether an array is indexed by j or by $jrow$ is dictated entirely by whether it is dimensioned by jmw or jmt.

14.2.2 Cell faces

As described below, in addition to defining prognostic variables at grid points within T cells and U cells, it is useful to define other variables on the faces of these cells. This is because the

equations are written in finite difference flux form to allow quantities to be conserved numerically. Each cell has six faces and certain variables must be defined on all faces. For instance, advective velocities (zonal, meridional, and vertical) are located on cell faces defined normal to those faces and positive in a direction of increasing longitude, latitude and decreasing depth[1]. Additional variables defined on cell faces are diffusive coefficients, viscous coefficients, diffusive fluxes, and advective fluxes. Note that names for variables on cell faces are only needed on the east, north, and bottom faces of cells. This is because the west, south, and top faces of cell $T_{i,k,j}$ are the east face of cell $T_{i-1,k,j}$, the north face of cell $T_{i,k,j-1}$, and bottom face of cell $T_{i,k-1,j}$. A similar arrangement holds for U cells.

14.2.3 Model size parameters

The main dimensional parameters are:

- *imt* = number of longitudinal grid cells in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *jmt* = number of latitudinal grid cells in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *km* = number of grid cells between the ocean surface and the bottom in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *jmw* = number of latitude rows within the memory window. The minimum is three and the maximum is *jmt*.
- *nt* = number of tracers in the model. Usually set to two. The first is for potential temperature and the second is for salinity as described in Section 14.2.8. Additional tracers (when $nt > 2$) are passive and initialized to 1.0 at level $k = 1$ amd 0.0 for levels $k > 1$.

Some secondary ones are:

- *mnisle* = maximum number of unconnected land masses (islands).
- *maxipp* = maximum number of island perimeter (coastal) points. This includes coastlines from all land masses.

14.2.4 T cells

A complete description of grid cells is given in Chapter 16. For cells on the T grid given by $T_{i,k,j}$, the primary grid distances in longitude and latitude are given by:

- xt_i = longitudinal coordinate of grid point within cell $T_{i,k,j}$ in degrees. Index i increases eastward with increasing longitude.
- yt_{jrow} = latitudinal coordinate of grid point within cell $T_{i,k,j}$ in degrees. This is written in the equations of this manual as ϕ^T_{jrow}. Index *jrow* increases northward with increasing latitude.
- dxt_i = longitudinal width of equatorial cell $T_{i,k,j}$ in cm.

[1]Positive vertical velocity at the bottom of a cell points upward in the positive z direction

- dyt_{jrow} = latitudinal width of cell $T_{i,k,j}$ in cm.
- cst_{jrow} = cosine of latitude at the grid point within cell $T_{i,k,j}$. This is written in the equations of this manual as $\cos\phi^T_{jrow}$.

14.2.5 U cells

A complete description of grid cells is given in Chapter 16. For cells on the U grid given by $U_{i,k,j}$, the primary grid distances in longitude and latitude are given by:

- xu_i = longitudinal coordinate of grid point within cell $U_{i,k,j}$ in degrees. Index i increases eastward with increasing longitude.
- yu_{jrow} = latitudinal coordinate of grid point within cell $U_{i,k,j}$ in degrees. This is written in the equations of this manual as ϕ^U_{jrow}. Index *jrow* increases northward with increasing latitude.
- dxu_i = longitudinal width of equatorial cell $U_{i,k,j}$ in cm.
- dyu_{jrow} = latitudinal width of cell $U_{i,k,j}$ in cm.
- csu_{jrow} = cosine of latitude at the grid point within cell $U_{i,k,j}$. This is written in the equations of this manual as $\cos\phi^U_{jrow}$.

14.2.6 Vertical spacing

A complete description of grid cells is given in Chapter 16. For T cells and U cells, the primary grid distances in the vertical are given by:

- zt_k = distance from the surface to the grid point in cell $T_{i,k,j}$ or $U_{i,k,j}$ in units of cm. The T cells and U cells are not staggered vertically. Note that although zt_k is positive downwards, the coordinate z is positive upwards. Index k increases downward.
- zw_k = distance from the surface to the bottom face of cell $T_{i,k,j}$ or $U_{i,k,j}$ in cm. The T cells and U cells are not staggered vertically.
- dzt_k = vertical thickness of cell $T_{i,k,j}$ or $U_{i,k,j}$ in cm.
- dzw_k = vertical distance between the grid point within cell $T_{i,k,j}$ and the grid point within cell $T_{i,k+1,j}$ in cm. It's the same for the U cells.

14.2.7 Time level indices

As explained in Section 21.4, indices are used to point to various time levels on disk and in the memory window. Note that all of these indices are integers.
The indices which point to time levels within the memory window are:

- *taum1* which points to data at time level $\tau - 1$.
- *tau* which points to data at time level τ.
- *taup1* which points to data at time level $\tau + 1$.

The indices which point to time levels on disk are:

- *taum1disk* which points to data at time level $\tau - 1$.
- *taudisk* which points to data at time level τ.
- *taup1disk* which points to data at time level $\tau + 1$.

Note that typically there are only two disks needed (unless multi-tasking with the coarse grained parallel approach). In this typical case, the "taup1disk" takes on the value of "taum1disk".

14.2.8 3-D Prognostic variables

The three dimensional[2] prognostic variables are:

- $u_{i,k,j,n,\tau}$ = horizontal velocity components located at grid points within cell $U_{i,k,j}$
- $t_{i,k,j,n,\tau}$ = tracer components located at grid points within cell $T_{i,k,j}$

Here, subscript i is the longitude index of the cell from $i = 1$ (westernmost cell) to $i = imt$ (easternmost cell); subscript k is the depth index of the cells from $k = 1$ (surface cell) to $k = km$ (bottom cell); and subscript j is the latitude index of the cells from $j = 1$ (southernmost cell) to $j = jmw$ (northernmost cell)[3]. With regard to U cells, subscript n is the velocity component[4] in units of *cm/sec*. With regard to T cells, subscript n refers to a particular tracer [5]. Subscript τ in both cases refers to the discrete time level.

14.2.9 2-D Prognostic variables

The two dimensional prognostic quantities are:

- $psi_{i,jrow,\tau}$ = stream function defined at i, $jrow$ locations on the T grid in units of cm^3/sec, where $10^{12} cm^3/sec$ is one sverdrup. $psi_{i,jrow,\tau}$ is written as $\psi_{i,jrow,\tau}$ in this manual.
- $ps_{i,jrow,\tau}$ = prognostic surface pressure defined at i, $jrow$ locations on the T grid in units of $gram/cm/sec^2$. This variable is also used as the implicit free surface (in terms of pressure) and the units are the same.

[2]Referring to three spatial dimensions.

[3]The j index need only be dimensioned for the size of the memory window. Refer to Chapter 10 for a description of the memory window and Section 14.2 for a description of *jrow*.

[4]n=1 is the zonal and n=2 is the meridional velocity component. Vertical velocity is not a prognostic variable. It is a diagnostic variable defined as an advective velocity at the bottom face of cells. Note that since there are T cells and U cells, there are vertical velocities associated with each. For diagnostic purposes, the vertical velocity on the bottom of T cells is output.

[5]n=1 is for potential temperature in units of *degrees C*, n=2 is for salinity. The "model salinity unit" is (ppt-35)/1000 where "ppt" is "parts per thousand" or "grams of salt per kilogram of water". The "ppt" unit of salinity has been largely replaced by "practical salinity units" or "psu" in the literature which is based on conductivity measurements instead of measuring "grams of salt per kilogram of water". MOM uses $\rho_\circ = 1.035\ gm/cm^3$ for the Boussinesq approximation. The model salinity units can be converted back to "ppt" by adding 0.035 grams/cm^3 and multiplying by 1000. n $>$ 2 is for additional passive tracers.

14.2.10 3-D Workspace variables

These variables are determined diagnostically in the sense that they are computed from other prognostic quantities and are therefore not indexed by time level. The three dimensional diagnostic and workspace variables are listed below:

- Six variables are needed for advective velocities on T cells and U cells. Units are *cm/sec*

  1. $adv_vet_{i,k,j}$ = advective velocity on the east face of cell $T_{i,k,j}$
  2. $adv_vnt_{i,k,j}$ = advective velocity on the north face of cell $T_{i,k,j}$
  3. $adv_vbt_{i,k,j}$ = advective velocity on the bottom face of cell $T_{i,k,j}$
  4. $adv_veu_{i,k,j}$ = advective velocity on the east face of cell $U_{i,k,j}$
  5. $adv_vnu_{i,k,j}$ = advective velocity on the north face of cell $U_{i,k,j}$
  6. $adv_vbu_{i,k,j}$ = advective velocity on the bottom face of cell $U_{i,k,j}$

- Two variables are needed for density and hydrostatic pressure gradients

  1. $rho_{i,k,j}$ = density defined at T cell grid points. Units are in sigma units gm/cm^3 and represent departures from reference densities specific to each model level as described in Section 15.1.2. In this manual, $rho_{i,k,j}$ is written as $\rho_{i,k,j}$.
  2. $grad_p_{i,k,j,n}$ = hydrostatic pressure gradient. n=1 is for the zonal component and n=2 is for the meridional component. Units are in cm/sec^2 assuming $\rho_\circ = 1.035\ gm/cm^3$.

- Six variables are needed for fluxes through the faces of T cells and U cells. Note that there is no explicit T or U cell suffix on these names because they are re-calculated for each prognostic variable in T cells and U cells. Note also that since these are re-calculated for each prognostic variable, they are <u>not</u> candidates for being moved as the memory window moves northward. Units are in *cgs* and specific to the quantity being advected or diffused.

  1. $adv_fe_{i,k,j}$ = advective flux on the east face of a cell
  2. $adv_fn_{i,k,j}$ = advective flux on the north face of a cell
  3. $adv_fb_{i,k,j}$ = advective flux on the bottom face of a cell
  4. $diff_fe_{i,k,j}$ = diffusive flux on the east face of a cell
  5. $diff_fn_{i,k,j}$ = diffusive flux on the north face of a cell
  6. $diff_fb_{i,k,j}$ = diffusive flux on the bottom face of a cell

- There are three coefficients for diffusion of tracers across T cells and another three for diffusion (viscous transfer of momentum) across U cells. They may or may not be triply subscripted. It depends on the physics parameterization. For instance, options *consthmix* and*constvmix* use coefficients which are constant throughout the grid and therefore require no subscripts. Units are cm^2/sec.

  1. $diff_cet_{i,k,j}$ = diffusive coefficient on the east face of cell $T_{i,k,j}$
  2. $diff_cnt_{i,k,j}$ = diffusive coefficient on the north face of cell $T_{i,k,j}$
  3. $diff_cbt_{i,k,j}$ = diffusive coefficient on the bottom face of cell $T_{i,k,j}$

4. $visc_ceu_{i,k,j}$ = viscous coefficient on the east face of cell $U_{i,k,j}$
5. $visc_cnu_{i,k,j}$ = viscous coefficient on the north face of cell $U_{i,k,j}$
6. $visc_cbu_{i,k,j}$ = viscous coefficient on the bottom face of cell $U_{i,k,j}$

Additionally, other variables may be needed but these are dependent on enabled options and will not be listed here.

14.2.11 3-D Masks

- To promote vectorization, two fields are used to differentiate ocean and land cells.

  1. $tmask_{i,k,j}$ = mask for $T_{i,k,j}$. 1.0 for ocean, 0.0 for land
  2. $umask_{i,k,j}$ = mask for $U_{i,k,j}$. 1.0 for ocean, 0.0 for land

14.2.12 Surface Boundary Condition variables

The surface boundary condition quantities are contained in three arrays. Note that one is dimensioned by the total number of latitude rows (*jmt*) while two are dimensioned by the size of the memory window (*jmw*).

- $sbcocn_{i,jrow,m}$= surface boundary conditions which are dimensioned as *(imt,jmt,numsbc)* where *numsbc* is the number of surface boundary conditions. Refer to Section 19.3 for usage details.

- $smf_{i,j,n}$ = components of the surface momentum flux defined on the U grid for the latitude rows within the memory window. $n = 1$ is for zonal momentum flux (windstress in the zonal direction) and $n = 2$ is for meridional momentum flux (windstress in the meridional direction). This field is dimensioned by the memory window size as *smf(imt,jmw,2)*. Units are in $dynes/cm^2$.

- $stf_{i,j,n}$ = components of the surface tracer flux defined on the T grid for the latitude rows within the memory window. $n = 1$ is for heat flux in units of *langley/sec* where 1 *langley* $= 1\ cal/cm^2$, $n = 2$ is for salt flux in units of *grams/cm*^{2}*/sec*, and $n > 2$ is for additional tracers. This field is dimensioned by the memory window size as *stf(imt,jmw,nt)*.

14.2.13 2-D Workspace variables

These two dimensional quantities are functions of geometry and topography. Being independent of time, they are computed only once and are dimensioned by (imt, jmt):

- $kmt_{i,jrow}$= number of T cells between the surface and bottom of the ocean defined at i, *jrow* locations on the T grid. It is constructed by module *topog*.

- $kmu_{i,jrow}$= number of T cells between the surface and bottom of the ocean defined at i, *jrow* locations on the U grid. It is given by Equation (18.3).

- $mskhr_{i,jrow}$= horizontal regional mask number used for certain diagnostics as described in Section 39.5.

- $mskvr_k$= vertical regional mask number used for certain diagnostics as described in Section 39.5. This is only a function of k but is included here because of its close association with $mskhr_{i,jrow}$.

- $h_{i,jrow}$ = depth from the surface to the bottom of the ocean defined at i, $jrow$ locations on the U grid in units of cm. In this manual, $h_{i,jrow}$ is written as $H_{i,jrow}$ and is computed by Equation (18.4).

- $hr_{i,jrow}$ = reciprocal of $h_{i,jrow}$ defined at i, $jrow$ locations on the U grid in units of $1/cm$. For masking purposes, $hr_{i,jrow} = 0$ whenever $h_{i,jrow} = 0$

The following two dimensional quantities are diagnosed from other prognostic quantities and are therefore recomputed every time step. They are dimensioned by (imt, jmt):

- $zu_{i,jrow,n}$ = vertical average of time derivatives of the momentum equation used as forcing for the external mode equations at i, $jrow$ locations on the U grid. Units are cm/sec^2.

- $ztd_{i,jrow}$ = curl of $zu_{i,jrow,n}$ defined at i, $jrow$ locations on the T grid. Units are $1/sec^2$.

- $res_{i,jrow}$ = residual from the elliptic solver defined at i, $jrow$ locations on the T grid. Units depend on whether streamfunction or rigid lid surface pressure or implicit free surface method is used. Refer to these variables for units.

- $ptd_{i,jrow}$ = time rate of change of stream function, rigid lid surface pressure, or implicit free surface defined at i, $jrow$ locations on the T grid. Refer to these variables for units.

- $cf_{i,jrow,-1:1,-1:1}$ = coefficient arrays for solving the external mode elliptic equation with 5 point or 9 point numerics. Units are $1/sec/cm^3$. Normally, this is independent of time except when solving the Coriolis part of the external mode implicitly. Since this computation is done very fast, it is computed every time step even when the Coriolis term is solved explicitly. The third and fourth indices are deviations about i and $jrow$ respectively. They give access to cell (i, $jrow$) and all eight surrounding grid cells.

14.3 Operators

As outlined in Section 14.1, operators are used to construct various terms in the tracer and momentum equations. They are implemented as statement functions requiring no storage and their details are expanded out in Sections 22.9.5 and 22.8.7. It is important to note that for operators to work correctly, all items buried within the details of the operator must be defined correctly at the time when the operator is used. As with variables, operators also have a placement on the grid as explained below:

14.3.1 Tracer Operators

The details of the following operators used in solving the tracer equations are defined in file *fdift.h*.

- $ADV_Tx_{i,k,j}$ = the flux form of the zonal (x) advection of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $ADV_Ty_{i,k,j}$ = the flux form of the meridional (y) advection of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $ADV_Tz_{i,k,j}$ = the flux form of the vertical (z) advection of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $ADV_Txiso_{i,k,j}$ = the counterpart to $ADV_Tx_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gm_advect* is enabled. Units are in tracer units per second.

- $ADV_Tyiso_{i,k,j}$ = the counterpart to $ADV_Ty_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gm_advect* is enabled. Units are in tracer units per second.

- $ADV_Tziso_{i,k,j}$ = the counterpart to $ADV_Tz_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gm_advect* is enabled. Units are in tracer units per second.

- $DIFF_Tx_{i,k,j}$ = the flux form of zonal (x) diffusion of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $DIFF_Ty_{i,k,j}$ = the flux form of the meridional (y) diffusion of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $DIFF_Tz_{i,k,j}$ = the flux form of the vertical (z) diffusion[6] of tracer defined on the $T_{i,k,j}$ grid point. Units are in tracer units per second.

- $DIFF_Tziso_{i,k,j}$ = the flux form[7] of the vertical component of isoneutral tracer diffusion defined on the $T_{i,k,j}$ grid point. Only used when option *isoneutralmix* is enabled. Units are in tracer units per second.

14.3.2 Momentum Operators

The details of the following operators used in solving the momentum equations are defined in file *fdifm.h*.

- $ADV_Ux_{i,k,j}$ = the flux form of the zonal (x) advection of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.

- $ADV_Uy_{i,k,j}$ = the flux form of the meridional (y) advection of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.

- $ADV_Uz_{i,k,j}$ = the flux form of the vertical (z) advection of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.

- $ADV_metric_{i,k,j,n}$ = the metric term for momentum advection defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.

[6]This is the explicit portion of K^{33} indicated in 35.1 when option *isoneutralmix* is enabled. If option *implicitvmix* is enabled, then it is the explicit part of the vertical diffusion.

[7]These are the K^{31} and K^{32} tensor components indicated in Section 35.1.

- $DIFF_Ux_{i,k,j}$ = the flux form of the zonal (x) diffusion of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.
- $DIFF_Uy_{i,k,j}$ = the flux form of the meridional (y) diffusion of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.
- $DIFF_Uz_{i,k,j}$ = the flux form of the vertical (z) diffusion[8] of momentum defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.
- $DIFF_metric_{i,k,j,n}$ = the metric term for momentum diffusion defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.
- $CORIOLIS_{i,k,j,n}$ = Coriolis term defined on the $U_{i,k,j}$ grid point. Units are in cm/sec^2.

14.4 Input Namelist variables

Although MOM is configured in various ways using UNIX cpp options, the value of many of the variables and constants within MOM and its parameterizations are defaulted and their values can be over-ridden using *namelist*[9] input. Included below are the variables input through namelists categorized by namelist name.

14.4.1 Time and date

Namelist /*ictime*/

These variables are for use setting the time and date for initial conditions, referencing diagnostic calculations, and related items.

- *year0* = year of initial conditions (integer).
- *month0* = month of initial conditions (integer).
- *day0* = day of initial conditions (integer).
- *hour0* = hour of initial conditions (integer).
- *min0* = minute of initial conditions (integer).
- *sec0* = second of initial conditions (integer).
- *ryear* = user specified reference year (integer).
- *rmonth* = user specified reference month (integer).
- *rday* = user specified reference day (integer).
- *rhour* = user specified reference hour (integer).
- *rmin* = user specified reference min (integer).

[8] If option *implicitvmix* is enabled, then it is the explicit part of the vertical diffusion.

[9] This is a non-standard Fortran 77 feature that is very useful. Most compilers support it. Refer to any Fortran manual for usage.

- *rsec* = user specified reference sec (integer).
- *refrun* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is the starting time and date of each submitted job. For example, suppose each job submission integrates for one month starting at the beginning of a month but the number of days per month changes. Setting "refrun" = true and setting "timavgint" = (days in month)/3 will give 3 averaged outputs per month at intervals of approximately 10 days each. The averaging period *timavgper* may be set less than *timavgint* for shorter averages but the output is still every *timavgint* days. The only restriction is that "timavgint" divided into the integration time for each job should work out to be an integral number (because this diagnostic is an average over time). If not, then "timavgint" is reset internally to insure this condition.
- *refinit* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is the time and date of the initial conditions. For example, if term balances are desired every 20 days "trmbint" = 20.0, then they will be calculated and written out every 20 days starting from initial condition time.
- *refuser* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is a user specified time and date given by (ryear, rmonth, rday, rhour, rmin, rsec) described above.for comparing diagnostics from various experiments with different initial condition times, *refuser* = *true* is appropriate. Note that setting *refuser* = *true* and choosing the reference time to be the initial condition time is the same as specifying *refinit* = *true*.

  Note that one of these reference time Booleans must be set *true* and the other two set *false*.
- *eqyear* = Boolean which forces all years to have the same number of days (no leap years). If *false*, then a Julian calendar is used.
- *eqmon* = Boolean which forces all months to have the same number of days. If *false*, then the actual number of days per month will be used. This is only used when *eqyear* is *true*.
- *monlen* = the length of each month in days when *eqmon* = *true*.

14.4.2 Integration control

Namelist /*contrl*/

These variables are used for setting the integration time, when to initialize, and when to write restart files.

- *init* = Boolean for denoting whether the first time step of a run is to perform initializations such as reading in initial conditions, etc. When *init* = false, then the execution is started from a restart data file. Refer to *restrt* listed below.
- *runlen* = length of run in units given by *rununits*
- *rununits* = units for *runlen*. Either 'days', 'months' or 'years'.
- *segtim* = time in days for one segment of ocean or atmosphere. Only used with option *coupled*.

- *restrt* = Boolean for controlling whether a restart is to be written at the end of the run.
- *initpt* = Boolean for controlling whether particle trajectories are to be initialized on the first time step of a run. This is only used if option *trajectories* is enabled.

14.4.3 Surface boundary conditions

Namelist /*mbcin*/

These variables are for surface boundary conditions.

- *ocean_sbc* = names for surface boundary conditions chosen from the list within module *setup_sbc*. Note that if specifying surface boundary conditions from namelist, the entire list of ocean and atmos surface boundary conditions must be given.
- *atmos_sbc* = names for surface boundary conditions chosen from the list within module *setup_sbc*. Note that if specifying surface boundary conditions from namelist, the entire list of ocean and atmos surface boundary conditions must be given.
- *numpas* = maximum number of iterations used to extrapolate data into land regions on the model grids where it was constructed. This only applies when option *coupled* is enabled.
- *bwidth* = blending zone width in degrees when using limited domain ocean models with global atmosphere models. This only applies when option *coupled* is enabled.
- *taux0* = zonal windstress in $dynes/cm^2$ for idealized equatorial studies. Refer to Section 28.2.2.
- *tauxy* = meridional windstress in $dynes/cm^2$ for idealized equatorial studies. Refer to Section 28.2.3.

14.4.4 Time steps

Namelist /*tsteps*/

Historically, ocean time is measured in terms of tracer time steps. It should be noted that the equations in MOM can be solved asynchronously (Bryan 1984) using one timestep for the internal mode *dtuv*, another for the external mode *dtsf*, and a third for the tracers *dtts*. Each are input through namelist.

Basically asynchronous timestepping can be done because the three processes have different time scales. Since the timescale for adjustment of density is much greater than that of velocity, it is reasoned that integrations to equilibrium can be speeded up by taking a large time step on the tracer equations (within CFL restrictions) and letting the velocities come into geostrophic adjustment with the density. If the problem is linear and only the equilibrium solution is sought, the equilibrium solution is unique and it doesn't matter how the integration gets there. However, if the solution is non-linear enough to have multiple equilibria or the transient response is of interest, all three time steps should be synchronous with the rigid lid, and $dtts = dtuv$ for the explicit free surface.

A similar argument can be made for the adjustment time scale of the deep layers being much greater than that of the surface layers. An acceleration with depth factor $dtxcel_k$, initialized to 1.0 for all levels, is used to increase the length of the tracer time step with depth to reach equilibrium sooner (Bryan 1984).

Given the resolution defined by module *grids*, a time step can be estimated from the linear CFL condition (see Haltiner and Williams 1980)

$$\Delta t \leq \frac{\Delta_{min}}{2 \cdot c \cdot \sin(2\pi\Delta_{min}/L)} \tag{14.2}$$

where Δ_{min} is the minimum grid cell width, c is the wavespeed, and L is the scale of the wave. The most restrictive scale is $L = 4\Delta_{min}$. The external gravity wave is the fastest wave with $c = \sqrt{grav \cdot H}$ but this is filtered out of the equations by the rigid lid approximation. If the explicit free surface option is enabled, then this wave speed must be resolved and this accounts for the relatively short time step on the external mode. Next fastest are the low frequency external mode barotropic Rossby basin-scale waves with $c = -\frac{\beta}{k^2+l^2}$ where $k = \frac{2\pi}{L_x}$ and $l = \frac{2\pi}{L_y}$ are zonal and meridional wavenumbers. These waves limit the barotropic time step when using the rigid lid stream function method. Eastward traveling Rossby waves have small scales and their speed is too slow to be a limiting factor on the time step. Small scale internal mode gravity waves and the non-dispersive Kelvin waves travel with maximum speed $c \approx 3$ *m/sec*. In general, these are the waves that restrict the baroclinic time step. The trace time step is limited by advective velocity which can easily reach 1 or $2m/sec$ in boundary currents.

In some models, wavespeed is not the limiting factor for determining timestep length. For instance, when vertical resolution is approximately 10 meters thick, the time step may be limited by vertical velocity near the surface in regions such as the equator. Regardless of what limits the time step, it is recommended that diagnostic option *stability_tests* be enabled to show how close the model is to the local CFL condition and where that position is located. Large vertical diffusion coefficients can also limit the timestep length and when this is the case, option *implicitvmix* should be enabled to solve the vertical diffusion components implicitly.

Here are some rough examples from models run at GFDL.

- For a 2.4° by 2.4° mid latitude thermocline model, the following settings were used: $dtsf = dtuv = 3000.0$ sec; $dtts = 30000.0$ with acceleration of $dtts$ with depth using $dtxcel_1 = 1.0$ to $dtxcel_{km} = 5.0$.
- For a $\Delta_\lambda = 1°$ by $\Delta_\phi = 1/3°$ equatorial basin the following settings were used: $dtsf = dtuv = dtts = 3000.0$ sec.

The following variables set the time steps in seconds.

- *dtts* = time step length for tracers
- *dtuv* = time step length for internal mode velocities
- *dtsf* = time step length for external mode velocities

14.4.5 External mode

Namelist */riglid/*

These variables are for setting limits for the elliptic solvers.

- *mxscan* = maximum number of iterations.
- *tolrsf* = admissible error for the stream function in cm^3/sec. A reasonable starting point is 10^8. Refer to Section 29.2.
- *tolrsp* = admissible error for the surface pressure in $gram/cm/sec^2$. A reasonable starting point is 10^{-4}. Refer to Section 29.3.2.
- *tolrfs* = admissible error for the implicit free surface in $gram/cm/sec^2$. A reasonable starting point is 10^{-4}. Refer to Section 29.5.
- *land_mass_a* = the land mass number of the first land mass for specifying zero net transport between two land masses. The number is taken from the island map which is printed out when the model executes. Refer to Section 29.2.6.
- *land_mass_b* = the land mass number of the second land mass for specifying zero net transport between two land masses. The number is taken from the island map which is printed out when the model executes. Refer to Section 29.2.6.

14.4.6 Mixing

Namelist /*mixing*/

These variables are for setting mixing and related values.

- *am* = lateral viscosity coefficient in cm^2/sec for option *consthmix*. Refer to Bryan, Manabe, Pacanowski (1975) for estimating a value.
- *ah* = lateral diffusion coefficient in cm^2/sec for option *consthmix*.
- *ambi* = absolute value of lateral viscosity coefficient in cm^4/sec for option *velocity_horz_biharmonic*. Refer to Section 34.4 for estimating a value.
- *ahbi* = absolute value of lateral diffusion coefficient in cm^4/sec for option *tracer_horz_biharmonic*. Refer to Section 34.4 for estimating a value.
- *kappa_m* = vertical viscosity coefficient in cm^2/sec for option *constvmix*.
- *kappa_h* = vertical diffusion coefficient in cm^2/sec for option *constvmix*.
- *cdbot* = bottom drag coefficient which is unitless. A typical value would be around 2.5×10^{-3}.
- *aidif* = implicit vertical diffusion factor. In cases where the vertical mixing coefficients severely limit the time step (because of fine vertical resolution or large vertical eddy coefficients), this constraint on the time step can be relaxed by solving the vertical diffusion term implicitly. The vertical diffusion term is solved implicitly when option *implicitvmix* or *isoneutralmix* is enabled. Otherwise, $aidif$ is not used. When solving implicitly, $0 \leq aidif \leq 1$ with $aidif = 1$ giving fully implicit and $aidif = 0$ giving fully explicit treatment. Why should semi-implicit vertical diffusion be used? The recommendation from Haltiner and Williams (1980) is for $aidif = 0.5$ when solving semi-implicitly. This gives the Crank-Nicholson implicit scheme which is always stable. This setting is supposed to be the most accurate one. However, this is not the case when solving a time dependent problem as discussed in Section 38.5.

- *ncon* = number of passes on the convective adjustment routine. Only meaningful when option *fullconvect* is not enabled and *implicitvmix* is not enabled.
- *nmix* = number of time steps between mixing time steps. A mixing time step is either a Forward or Euler Backward time step as opposed to the normal leapfrog time step.
- *eb* = Boolean for using Euler backward mixing scheme used on mixing time steps.
- *acor* = implicit Coriolis factor for treating the Coriolis term semi-implicitly. For semi-implicit treatment, $0.5 < acor < 1$ and option *damp_inertial_oscillation* must be enabled. Refer to Section 27.2.1 for a discussion of when this is appropriate.
- *dampts* = Newtonian damping time scale in days used with option *restorst*. Note that damping time scale may by set differently for each tracer.
- *annlev* = Boolean for replacing seasonal sponge data by annual means when enabling option *sponges*.
- *filter_reflat_s* = southern latitude (degrees) below which polar filtering operates if option *fourfil* is enabled. The filter will remove zonal scales from the solution which are less than the zonal grid spacing at latitude *filter_reflat_s*.
- *filter_reflat_n* = northern latitude (degrees) above which polar filtering operates if option *fourfil* is enabled. The filter will remove zonal scales from the solution which are less than the zonal grid spacing at latitude *filter_reflat_n*.
- *rjfrst* = southern latitude (degrees) below which polar filtering is turned off when option *fourfil* is enabled. The purpose of this is to save computations over the land mass of Antarctica.

14.4.7 Diagnostic intervals

Namelist /*diagn*/

These variables are used for setting diagnostic intervals and related items. The interval is a real number and has a Boolean variable (switch) associated with it which is set by the time manager every time step. The Boolean variable is set to *true* when the model integration time *mod* the interval is less than or equal to half a timestep. Otherwise, it is set *false*.

To add a switch, three variables must be added to the common block in *switch.h*: an interval (real numbner), a Boolean variable which acts as the logical switch, and an integer variable used as an index within module *tmngr*. Refer to include file *switch.h* to see the structure. Refer to the section where alarms are set within module *tmngr* for examples of how to implement new switches.

- *tsiint* = interval in days between writing time step integrals. This is used with option *time_step_monitor*.
- *tavgint* = interval in days between writing data for use with option *tracer_averages*.
- *itavg* = Boolean for writing regional mask when used with option *tracer_averages*. It should be set true on the first time step of the first run and false thereafter. This allows datasets from multiple runs to be concatenated without regional mask information being duplicated.

- *tmbint* = interval in days between writing data for option *meridional_tracer_budget*.
- *tmbper* = period in days for producing time averaged data for use with option *meridional_tracer_budget*. This averaging period may be set shorter than the interval *tmbint*.
- *itmb* = Boolean for writing "msktmb" when used with option *meridional_tracer_budget*.It should be set true on the first time step of the first run and false thereafter. This allows datasets from multiple runs to be concatenated without regional mask information being duplicated.
- *stabint* = interval in days between doing stability analysis for use with option *stability_tests*.
- *cflons* = starting longitude (deg) for computing data for use with option *stability_tests*.
- *cflone* = ending longitude (deg) for computing data for use with option *stability_tests*.
- *cflats* = starting latitude (deg) for computing data for use with option *stability_tests*.
- *cflate* = ending latitude (deg) for computing data for use with option *stability_tests*.
- *cfldps* = starting depth (cm) for computing data for use with option *stability_tests*.
- *cfldpe* = ending depth (cm) for computing data for use with option *stability_tests*.
- *maxcfl* = maximum number of CFL violations before quitting for use with option *stability_tests*.
- *zmbcint* = interval in days between writing data for option *show_zonal_mean_of_sbc*.
- *glenint* = interval in days between writing data for option *energy_analysis*.
- *trmbint* = interval in days between writing data for option *term_balances*.
- *itrmb* = Boolean for writing regional masks when used with option *term_balances*.
- *vmsfint* = interval in days between writing data for option *meridional_overturning*.
- *igyre* = Boolean for writing regional masks when used with option *gyre_components*.
- *gyreint* = interval in days between writing data for option *gyre_components*.
- *exconvint* = interval in days between writing data for option *save_convection*.
- *cmixint* = interval in days between writing data for option *save_mixing_coeff*.
- *extint* = interval in days between writing data for option *show_external_mode*.
- *prxzint* = interval in days between writing data for use with option *matrix_sections*.
- *prlat* = latitudes for writing data for use with option *matrix_sections*.
- *prslon* = starting longitude (deg) for writing (xz) data for use with option *matrix_sections*.
- *prelon* = ending longitude (deg) for writing (xz) data for use with option *matrix_sections*.
- *prsdpt* = starting depth (cm) for writing (xz) data for use with option *matrix_sections*.

- *predpt* = ending depth (cm) for writing (xz) data for use with option *matrix_sections*.
- *slatxy* = starting latitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *elatxy* = ending latitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *slonxy* = starting longitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *elonxy* = ending longitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *trajint* = interval in days between writing data for use with option *trajectories*.
- *dspint* = interval in days between writing data for use with option *diagnostic_surf_height*.
- *dspper* = period in days for producing time averaged data for use with option *diagnostic_surf_height*. This averaging period may be set shorter than the interval *dspint*.
- *snapint* = interval in days between writing data for use with option *snapshots*.
- *snapls* = starting latitude (deg) for writing data for use with option *snapshots*.
- *snaple* = ending latitude (deg) for writing data for use with option *snapshots*.
- *snapds* = starting depth (cm) for writing data for use with option *snapshots*.
- *snapde* = ending depth (cm) for writing data for use with option *snapshots*.
- *timavgint* = interval in days between writing data for use with option *time_averages*.
- *timavgper* = period in days for producing time averaged data for use with option *time_averages*. This averaging period may be set shorter than the interval *timavgint*.
- *xbtint* = interval in days between writing data for use with option *save_xbts*.
- *xbtper* = period in days for producing time averaged data for use with option *save_xbts*. This averaging period may be set shorter than the interval *xbtint*.

14.4.8 Directing output

Namelist /*io*/

These variables are used for directing diagnostic output to either the model *printout* file which is in *ascii* or to 32 bit IEEE data files with suffixes *.dta*. These control variables will not direct output to NetCDF formatted files. NetCDF format files have suffixes *.dta.nc* and this format is controlled by options described under each diagnostic in Chapter 39. Control variables are integers and exert control as follows:

- *If* control variable < 0 then output is written to unformatted IEEE file and *stdout*.
- *If* control variable > 0 *and* $\neq 6$ then output is written to unformatted IEEE file.
- *If* control variable $= 6$ then output is written to *stdout* which is file *fort.6*. The script *run_mom* redirects *stdout* to file *results* and copies it to a printout file.

The namelist variables are:

- *expnam* = character*60 experiment name.
- *iotavg* = control variable for writing output from option *tracer_averages*.
- *iotmb* = control variable for writing output from option *meridional_tracer_budget*.
- *iotrmb* = control variable for writing output from option *term_balances*.
- *ioglen* = control variable for writing output from option *energy_analysis*.
- *iovmsf* = control variable for writing output from option *meridional_overturning*.
- *iogyre* = control variable for writing output from option *gyre_components*.
- *ioprxz* = control variable for writing output from option *matrix_sections*.
- *ioext* = control variable for writing output from option *show_external_mode*.
- *iodsp* = control variable for writing output from option *diagnostic_surf_height*.
- *iotsi* = control variable for writing output from option *time_step_monitor*.
- *iozmbc* = control variable for writing output from option *show_zonal_mean_of_sbc*.
- *iotraj* = control variable for writing output from option *trajectories*.
- *ioxbt* = control variable for writing output from option *save_xbts*.

14.4.9 Isoneutral diffusion

Namelist /*isopyc*/

These variables are for use with option *isoneutralmix*.

- *ahisop* = isoneutral diffusion coefficient in cm^2/sec.
- *athkdf* = GM90 diffusion coefficient in cm^2/sec. This is only used with option *gent_mcwilliams*.
- *abihrm* = Roberts and Marshall (1998) biharmonic diffusion coefficient in cm^4/sec. This is only used with option *biharmonic_rm*.
- *ahsteep* = Minimum diffusivity in cm^2/sec used for determining the strength of the lateral diagonal terms in the tracer mixing tensor.
- *slmx* = maximum slope of isopycnals.

14.4.10 Nonconstant isoneutral diffusivities

Namelist /*ncdiff*/

These variables are for use with either option *hl_diffusivity* or option *vmhs_diffusivity*. A full description of the namelist variables for these schemes is given in Sections 35.2.1 and 35.2.2.

14.4.11 Pacanowski/Philander mixing

Namelist /*ppmix*/

These variables are for use with option *ppvmix*.

- *wndmix* = min value for mixing at surface to simulate high frequency wind mixing in cm^2/sec. (if absent in forcing).
- *frcmx* = maximum mixing in cm^2/sec.
- *diff_cbt_back* = background diffusion coefficient in cm^2/sec.
- *visc_cbu_back* = background viscosity coefficient in cm^2/sec.
- *diff_cbt_limit* = limiting diffusion coefficient in cm^2/sec. This is to be used in regions of negative Richardson number.
- *visc_cbu_limit* = limiting viscosity coefficient in cm^2/sec. This is to be used in regions of negative Richardson number.

14.4.12 Smagorinsky mixing

Namelist /*smagnl*/

These variables are for use with option *smagnlmix*.

- *diff_c_back* = background diffusion coefficient which is added to predicted diffusion coefficient in cm^2/sec.
- *visc_c_back* = background viscosity coefficient which is added to predicted diffusion coefficient in cm^2/sec.
- *k_smag* = von Karman coefficient for calculating minimum resolvable wavelength.

14.4.13 Bryan/Lewis mixing

Namelist /*blmix*/

These variables are for use with option *bryan_lewis_vertical* and *bryan_lewis_horizontal*.

- Ahv_k = vertical diffusion coefficient for tracers as a function of depth in cm^2/sec.
- Ahh_k = horizontal diffusion coefficient for tracers as a function of depth in cm^2/sec.
- *afkph* = coefficient for setting up vertical dependence of Ahv_k. Refer to Section 33.2.2.
- *dfkph* = coefficient for setting up vertical dependence of Ahv_k. Refer to Section 33.2.2.
- *sfkph* = coefficient for setting up vertical dependence of Ahv_k. Refer to Section 33.2.2.
- *zfkph* = coefficient for setting up vertical dependence of Ahv_k. Refer to Section 33.2.2.

Chapter 15

Modules and Modularity

MOM was originally designed within Fortran 77 which did not support the concept of a module. Nevertheless, there was an attempt to emulate the module concept as implemented in languages other than Fortran (i.e. Turbo Pascal). For purposes of MOM, modules and modularity were used as key organizational tools to minimize inter-connectivity between various model components so that the model's structure remained relatively clear even when subjected to a large number of additions.

Since the adoption of Fortran 90 as a minimum requirement for MOM, the intent was to take advantage of the Fortran 90 modules where possible. Although most of the coding is still in the fortran 77 style, some of the latter additions are in the style of Fortran 90. Time does not permit re-writing MOM 3 to make better use of Fortran 90. A good example of a Fortran 90 module is the time manager which is described below.

15.1 List of Modules

The following sections contain a listing of modules by filename and a description of what they do. The run_scripts which activate these modules in a stand alone mode are all UNIX C shell scripts and should work on any workstation running UNIX. They are amenable to change and are intended as prototypes.

Recommendation

When experiencing problems in MOM relating to modules, the recommendation is to run the module in isolation (stand alone mode) as described below in an attempt to reproduce the problem in a simpler environment. For example, if a problem is suspected with I/O, try to replicate it within the driver for the I/O manager *iomngr*. Or if a diagnostic is not being saved at the intended time, try to replicate the problem in the driver for the time manager module *tmngr*. For additional information, refer to related options in Chapter 23 as indicated below.

15.1.1 convect.F

File *convect.F* contains the convection module. It may be exercised in a stand alone mode by executing script *run_convect* which enables option *test_convect*. The driver is intended to show what happens to two bubbles in a one dimensional column of fluid when acted on by the

old style explicit convection used in previous versions of MOM and a newer explicit convection scheme enabled by option *fullconvect*. Refer to Section 33.1 for details on both types of convection.

15.1.2 denscoef.F and MOM's density

This section provides a general discussion of density used in MOM. It also mentions the diagnosis of the *in situ* density and potential density available through the model diagnostic option *density_netcdf* (Section 40.2).

15.1.2.1 Bryan and Cox 1972

In Bryan and Cox (1972), a density is constructed as a third order polynomial approximation at each model level k by defining a set of coefficients $x_{\ell,k}, \ \ \ell = 1, 9$. The polynomial is written as

$$\begin{aligned} [\rho(T,S,Z_k) - \rho_{0,k} - 1] \cdot 10^3 &= x_{1,k}\delta T + x_{2,k}\delta S + x_{3,k}(\delta T)^2 + x_{4,k}(\delta S)^2 + x_{5,k}\delta T \delta S \\ &+ x_{6,k}(\delta T)^3 + x_{7,k}(\delta S)^2 \delta T + x_{8,k}(\delta T)^2 \delta S + x_{9,k}(\delta S)^3 \end{aligned} \quad (15.1)$$

where T is in-situ temperature, S is salinity, Z_k is the depth to the grid point within level k, $\rho_{0,k}$ is a reference density for level k, and $\rho(T,S,Z_k)$ is the density. Anomolies δT and δS are referenced to mean values $T_{0,k}$ and $S_{0,k}$ at the appropriate levels

$$\delta T = T - T_{0,k} \quad (15.2)$$

$$\delta S = S - S_{0,k} \quad (15.3)$$

$$\rho_{0,k} = \rho(T_{0,k}, S_{0,k}, Z_k) \quad (15.4)$$

MOM uses potential temperature instead of in-situ temperature. Because of this, there are some differences between what is described in the Cox and Bryan paper and what is actually done within module *denscoef*. The method used in module *denscoef* for calculating the references and coefficients is described below.

15.1.2.2 Computing density within MOM

For dynamical purposes, density in MOM is calculated as an anomoly $\rho_{\tilde{t},\tilde{s},k}$ relative to a reference density ρ_k^{ref} at each model level. The reason for calculating an anomaly instead of an actual density is accuracy since the anomaly $\rho_{\tilde{t},\tilde{s},k} << \rho_k^{ref}$. If an actual density were used, three significant digits in accuracy would be lost when taking gradients. Dynamically, there is no difference between using an anomoly and an actual density because only gradients of density are important and the horizontal gradient operator eliminates ρ_k^{ref} from the anomoly. When vertical gradients of density are needed, both densities in the derivative are referenced to the same local depth which again eliminates ρ_k^{ref}.

MOM uses a polynomial very similar to Equation (15.1) to calculate the density anomoly. The pressure effect with depth is incorporated into the polynomial coefficients $c_{k,\ell}, \ \ \ell = 1, 9$

for each model level k. The relevant equations are

$$\tilde{t} = t_{i,k,j,1,\tau} - T_k^{ref} \tag{15.5}$$

$$\tilde{s} = t_{i,k,j,2,\tau} - S_k^{ref} \tag{15.6}$$

$$\rho_{\tilde{t},\tilde{s},k} = (c_{k,1} + (c_{k,4} + c_{k,7} * \tilde{s}) * \tilde{s} + (c_{k,3} + c_{k,8} * \tilde{s} + c_{k,6} * \tilde{t}) * \tilde{t}) * \tilde{t} + (c_{k,2} + (c_{k,5} + c_{k,9} * \tilde{s}) * \tilde{s}) * \tilde{s} \tag{15.7}$$

The density $\rho_{\tilde{t},\tilde{s},k}$ is the result of calling the model's statement function $dens(\tilde{t}, \tilde{s}, k)$. The references T_k^{ref}, S_k^{ref}, ρ_k^{ref}, and the coefficients $c_{k,\ell}$, $\ell = 1, 9$, are computed by module *denscoef* for each model level k as follows. First, $(T_i^{\star\, min}, T_i^{\star\, max})$ and $(S_i^{\star\, min}, S_i^{\star\, max})$ are defined as in-situ temperature and salinity ranges at 33 equi-spaced levels (at depths given by $(i-1) * 250 meters \quad for \quad i = 1 \;\; to \;\; 33$) based on analysis of a world ocean hydrographic database as described in Bryan and Cox (1972). These ranges are interpolated to the depth of model levels $zt_k \quad for \quad k = 1, km$ by simply picking ranges from the equi-spaced depths that are nearest to model depths zt_k. The result is a new set of in-situ temperature ranges (T_k^{min}, T_k^{max}) and salinity ranges (S_k^{min}, S_k^{max}).

At each model level k, the range of specified in-situ temperatures is divided into 10 equi-spaced in-situ temperatures and the range of specified salinities into 5 equi-spaced salinities. From the 50 combinations, 50 in-situ densities are computed based on either the Knudsen-Ekman formula (if option *knudsen* is enabled) or the UNESCO equation of state (this is the default. For the appropriate equations, refer to Gill (1982), Appendix Three). Additionally, 50 potential temperatures corresponding to the 50 in-situ densities are computed. Then the average in-situ temperature, potential temperature, and salinity are constructed. The average potential temperature is used as the reference temperature T_k^{ref} and the average salinity is used as the reference salinity S_k^{ref}. A reference density ρ_k^{ref} is constructed from T_k^{ref} and S_k^{ref} at the depth of each model level zt_k using either the Knudsen-Ekman or the UNESCO (the default) equation of state.

Potential temperature anomolies $\tilde{t}$, salinity anomolies $\tilde{s}$, and in-situ density anomolies $\rho_{\tilde{t},\tilde{s},k}$ are then created by subtracting the reference values from the 50 potential temperatures, salinities, and in-situ densities within level k. Plugging these values into Equation 15.7 yields 50 equations and 9 unknown polynomial coefficients at each model level k. The system of equations is over determined at each level, and a best fit for the coefficients in the least squares sense is given by Hanson and Lawson (1969). In MOM, the solution is produced by a Jet Propulsion Laboratory subroutine "LSQL2".

From the above description, it should be clear that the values of the coefficients $c_{k,l}$ used in MOM are appropriate for potential temperatures and <u>not</u> in-situ temperatures. To check whether potential temperatures or salinities predicted by the model are outside the range of specified in-situ temperature and salinity ranges used to construct $c_{k,\ell}$, the in-situ temperature ranges are converted to potential temperature ranges for use within model diagnostics. When diagnostic option *stability_tests* is enabled, any predicted temperature or salinity outside of the ranges used to calculate $c_{k,\ell}$ will be flagged (although only on diagnostic time steps).

Normally, module *denscoef* is called from within a model execution to compute density coefficients and references. However, script *run_denscoef* enables option *drive_dencoef* and executes *denscoef.F* in a "stand alone mode" to produce a listing of the coefficients $c_{k,\ell}$ along with T_k^{ref}, S_k^{ref} and ρ_k^{ref}. In a model run, option *drive_dencoef* is <u>not</u> enabled.

15.1.2.3 *in situ* density and potential density

Even though the absolute density is not directly needed for the model's dynamics, it is common to diagnose *in situ* density or potential density. Option *density_netcdf* (Section 40.2) allows for such a diagnosis. The following discussion is relevant for that option. The *in situ* density at the model potential temperature $t_{i,k,j,1}$, salinity $t_{i,k,j,2}$, and depth k, is given by the sum of the model's dynamical density, the depth dependent reference value, and unity

$$\rho^{in\ situ}(t_{i,k,j,1}, t_{i,k,j,2}, k) = \rho_{\tilde{t},\tilde{s},k} + \rho_k^{ref} + 1, \tag{15.8}$$

where again $\tilde{t} = t_{i,k,j,1} - T_k^{ref}$, and $\tilde{s} = t_{i,k,j,s} - S_k^{ref}$. The dimensions of $\rho_{in\ situ}$ are g/cm^3. To get the density in sigma units, simply drop the one and multiply by 1000

$$\sigma^{in\ situ}(t_{i,k,j,1}, t_{i,k,j,2}, k) = 1000\,(\rho_{\tilde{t},\tilde{s},k} + \rho_k^{ref}). \tag{15.9}$$

The potential density referenced to a particular model depth level M is given by

$$\rho^{(M)}(t_{i,k,j,1}, t_{i,k,j,2}, k) = \rho^{(M)}_{\tilde{t},\tilde{s},k} + \rho^{ref}_{k=M} + 1, \tag{15.10}$$

where

$$\begin{aligned}
\rho^{(M)}_{\tilde{t},\tilde{s},k} &= (c_{M,1} + (c_{M,4} + c_{M,7} * \tilde{s}) * \tilde{s} + (c_{M,3} + c_{M,8} * \tilde{s} + c_{M,6} * \tilde{t}) * \tilde{t}) * \tilde{t} + \\
&\quad (c_{M,2} + (c_{M,5} + c_{M,9} * \tilde{s}) * \tilde{s}) * \tilde{s} && (15.11)\\
\tilde{t} &= t_{i,k,j,1} - T^{ref}_{k=M} && (15.12)\\
\tilde{s} &= t_{i,k,j,2} - S^{ref}_{k=M}. && (15.13)
\end{aligned}$$

In words, the potential density referenced to level M is computed by setting all the reference temperatures, salinities, densities, and expansion coefficients to those of the level $k = M$.

15.1.2.4 Linearized density and option linearized_density

For a number of idealized studies, it is useful to employ a linear equation of state which depends only on temperature

$$\rho = \rho_o\,[1 - \alpha\,(T - T_o)], \tag{15.14}$$

where T_o is some reference temperature. Option *linearized_density* invokes the following linear equation of state. Taking the reference temperature as $19°C$, the reference salinity as $35psu$, and the reference pressure as 0, Appendix Three of Gill (1982) gives

$$\begin{aligned}
\rho_o &= 1.025022\ g/cm^3 && (15.15)\\
\alpha &= 2489 \times 10^{-7}\ {}^\circ K^{-1}. && (15.16)\\
& && (15.17)
\end{aligned}$$

Hence, for all depths k, the expansion coefficients and equation of state are

$$\begin{aligned}
c_{k,1} &= -\alpha\,\rho_o = -2.55 \times 10^{-4} && (15.18)\\
c_{k,\ell} &= 0 \qquad \ell \neq 1 && (15.19)\\
\rho_{\tilde{t},\tilde{s},k} &= c_{k,1}\,\tilde{t} && (15.20)
\end{aligned}$$

The remaining reference values are given by

$$T_k^{ref} = 19 \tag{15.21}$$
$$S_k^{ref} = 0 \tag{15.22}$$
$$\rho_k^{ref} = \rho_o - 1 = 0.025022. \tag{15.23}$$

These values are used for all vertical levels, which means that pressure effects are removed. Recall that salinity is carried in MOM as a deviation from $35psu$

$$S_{model} = (S(psu) - 35)/1000, \tag{15.24}$$

which means that the reference salinity $S^{ref} = 0$ corresponds to $35psu$.

15.1.3 grids.F

File *grids.F* contains the *grids* module. Script *run_grids* uses option *drive_grids* to execute the module in a stand alone mode which is the recommended way to design a domain and grid resolution for MOM. The grid is specified in the USER INPUT section of file *grids.F*. When option *drive_grids* is not enabled, module *grids* is used by the model to install the specified grid information. Refer to Chapter 16 for a description of how to construct a grid.

15.1.4 iomngr.F

Th I/O manager can be tested by setting the appropriate computer PLATFORM variable in script *run_iomngr* and executing. The purpose of the I/O manager is to find unit numbers which are not already attached to other files and open then with the specified attributes.

The interface to subroutines is much the same as in the older version except that no abbreviations are allowed. As an example, consider writing data to a file named "test.dta" which is to be a sequential unformatted file. The following call

```
call getunit (nu,'test.dta', 'sequential unformatted rewind')
```

finds a unit number "nu" which is not attached to any other opened file, assigns it to file "test.dta", and performs an "open" statement with the requested file attributes. After writing data to unit "nu", the unit can be closed and the unit number released with the following call

```
call relunit (nu)
```

The file may subsequently be opened with an "append" attribute using

```
call getunit (nu,'test.dta', 'sequential unformatted append')
```

to append data after which the file can again be closed and the unit number released with another call to "relunit(nu)". Possible file keyword attributes include:

- "sequential" for sequential files, or "direct" for direct access files, or "word " (note the blank at the end) for CRAY word I/O files.

- "formatted" or "unformatted".
- "rewind" or "append".
- "words=" for specifying record length in words for direct access files.
- "sds" for solid state disk on CRAY computers outfitted with solid state disk.
- "ieee" for 32bit IEEE format on CRAY computers
- "cray_float" for reading restart data written by a CRAY YMP or C90 but read on a CRAY T90 which has a 64bit IEEE native format.

Note that the argument lists are the same as in the older version but that abbreviations for file attributes are not allowed.

15.1.5 poisson.F

poisson.F contains the elliptic equation solver module. It is exercised in a stand alone mode by using script *run_poisson* which enables option *test_poisson*. Module *poisson* uses the grid defined by module *grids* and the topography and geometry defined by module *topog*.

Only one method of solution is provided within module *poisson*: conjugate gradients. The conjugate gradient solver forms the basis of solving the external mode stream function and the rigid lid surface pressure and implicit free surface method developed by Dukowicz and Smith (1994). It is not used for the explicit free surface.

The stopping criterion for convergence within the elliptic solvers is input through a namelist (*tolrsf* for stream function, *tolrsp* for surface pressure, and *tolrfs* for the implicit free surface). This criterion behaves differently than the corresponding variable *crit* which was used in MOM 1. In MOM, the sum of the truncated series of future corrections to the prognostic variable is estimated assuming geometric convergence and the iteration is terminated when this sum is within the requested tolerance. This means that the solution differs from the true solution to within an error given by the tolerance. The tolerance is given in the same units as the external mode prognostic variable. In MOM 1, when the residual was smaller than *crit* the iteration was stopped. This, however, did not mean that the solution was within *crit* of the true solution. Before using one of these solvers, it must be decided whether 5 point or 9 point numerics are to be used. For the rigid lid surface pressure and implicit free surface method, only the 9 point numerics are appropriate. For the stream function, either the 5 point (option *sf_5_point*) or the 9 point (option *sf_9_point*) numerics is appropriate. The recommendation is to use option *sf_9_point*. The generation of the coefficient matrix for the elliptic equation is described in Section 31.2. The resulting matrix is slightly different that the one calculated in previous versions of the model but presumed to be more accurate.

There are other differences besides the coefficient matrix when comparing the way the elliptic equation for the stream function was solved in MOM 1 and MOM. The result of the differences (as measured in the conjugate gradient) is that the solvers in MOM converge faster and in less time than in MOM 1. How much is problem dependent but for the test cases measured, the difference is about 10 to 20% of the time taken by the external mode when solving to equivalent accuracy. Refer to Section 29.2.4 for details of the island equations.

15.1.6 vmix1d.F

File *vmix1d.F* executes vertical mixing schemes in a simple one dimensional model (depth versus time). Currently, it will exercise option *ppvmix* which is the scheme based on Pacanowski/Philander (1981) or option *kppvmix* which is the scheme based on Large, McWilliams, and Doney (1994) depending on which one is set in the run script *run_vmix1d*. Script *run_vmix1d* exercises this module in stand alone mode by enabling option *test_vmix* and *ppvmix* or *kppvmix*. The driver uses a simplified 1-D equation configuration with Coriolis and vertical diffusion terms at a specific latitude to indicate how mixing penetrates vertically. The 1-D equations are

$$\partial_t u - fv = \partial_z(\kappa_m \cdot \partial_z(u)) \tag{15.25}$$

$$\partial_t v + fu = \partial_z(\kappa_m \cdot \partial_z(v)) \tag{15.26}$$

$$\partial_t T = \partial_z(\kappa_h \cdot \partial_z(T)) \tag{15.27}$$

with values of κ_m and κ_h being predicted by the enabled scheme. Refer to Section 33.2.4 for details of the scheme.

15.1.7 timeinterp.F

File *timeinterp.F* contains the time interpolator module. It is exercised in a stand alone mode using script *run_timeinterp* which enables option *test_timeinterp*. The output indicates how surface boundary condition data (which may be monthly averages, daily averages, etc) is interpolated to the current time step in a simulated model integration. Keep in mind that interpolating surface boundary conditions in time only applies in the model when option *time_varying_sbc_data* is enabled.

Based on model time, the time interpolator decides when to read data from disk into memory buffers. When to read and which data to read is determined by the relation between the model time and the time at the centers of the data records. There are two memory buffers: one to hold data from the disk record whose centered time[1] precedes the model time (the previous data) and one to hold data from the disk record whose centered time has not yet been reached by the model time (the next data). Four alternative interpolation methods are supported as described in Section 19.2.

15.1.8 timer.F

File *timer.F* contains the timer module. It can be exercised in a stand alone mode by executing script *run_timer* which enables option *test_timer*. The module contains general purpose timing routines which are useful for optimizing any code. The timing routines consist of calls to"tic" and "toc" routines which are placed around code to be timed. Many levels of nesting are allowed as well as dividing times into categories and sub-categories. The driver (which is overly complex) simulates solving a tracer equation using various forms for calculating advection and diffusion. Timing results are given for each case. Experience shows there is no one form that is optimum on all computers, so if one wants to optimize speed for a particular environment, these routines will be useful. In MOM, most code sections are outfitted with calls to these timing routines. Enabling option *timing* in MOM will give an indication of how

[1]Date and time defined at the center of the data record. For instance, if the record was January (31 days long), the centered time would be at day 15.5 which is the center.

much time is taken by various options and sections. These routines take time themselves to execute[2] and so should be disabled once results are obtained.

15.1.9 Time manager

File *tmngr_mod.F90* contains the time manager module. MOM uses this module for working with times and dates but the module can also be exercised in stand alone mode by using script *run_tmngr* which activates the module's driver (main program). A specific clock and calendar, which may be changed, are defined within the module's driver along with a timestep length and specific intervals for certain events. The driver then integrates time forward (one timestep at a time) and indicates on which time steps the requested events take place. This time manager is well suited for use in any modelling project. Note that it depends heavily on Fortran 90 constructs. The following writeup was contributed by Jeff Anderson (jla@gfdl.com).

15.1.9.1 Introduction

This document describes a software package that provides a set of simple interfaces for modelers to perform computations related to time and dates. The package defines a "type" that can be used to represent discrete times (accurate to within one second) and to map these times into dates using a variety of calendars. A time is mapped to a date by representing the time since some base date. The base date is implicitly defined so researchers don't have to think about it.

15.1.9.2 Overview of interfaces

The time manager provides a single defined type, time_type, which is used to store time and date quantities. A time_type is a positive definite quantity that represents an interval of time. It can be most easily thought of as representing the number of seconds in some time interval. A time interval can be mapped to a date under a given calendar definition by using it to represent the time that has passed since some base date. A number of interfaces are provided to operate on time_type variables and their associated calendars. Time intervals can be as large as n days where n is the largest number represented by the default integer type on a compiler. This is typically considerably greater than 10 million years (assuming 32 bit integer representation) which is likely to be adequate for most applications. The description of the interfaces is separated into two sections. The first deals with operations on time intervals while the second deals with operations that convert time intervals to dates for a given calendar.

15.1.9.3 Time interfaces

If **t1**, **t2**, and **t3** are of type time_type, then following operators are allowed:

1. **t3 = t1 + t2** :: Sum of two time intervals
2. **t3 = t1 - t2** :: Difference of two time intervals. WARNING: The difference is a positive definite time interval; so **t1-t2** is the same as **t2-t1**.
3. **t2 = n * t1** :: Product of time interval with integer.

[2]This time is accounted for in the timer routines.

4. **t2 = t1 / n** :: Integer scalar divide.
5. **n = t2 / t1** :: Gives largest integer **n** for which **n * t1** $<$ **t2**
6. **d = t2 // t1** :: Gives double precision result of dividing **t2** by **t1**
7. **Relational operators** :: **t1** $>$ **t2**, **t1** $>=$ **t2**, **t1** $==$ **t2**, **t1** $<=$ **t2**, **t1** $<$ **t2**, **t1 /= t2**

The following are subroutine and function interfaces for working with times:

1. **time = set_time**(seconds, days)

   type(time_type) :: set_time
   integer, intent(in) :: seconds, days

   Given some number of seconds and days, set_time returns the corresponding time_type. The number of seconds can be greater than 86400.

2. **get_time**(time, seconds, days)

   type(time_type), intent(in) :: time
   integer, intent(out) :: seconds, days

   Given a time interval, returns the corresponding seconds (¡ 86400) and days.

3. **time = increment_time**(time, seconds, days)

   type (time_type) :: increment_time
   type(time_type), intent(in) :: time
   integer, intent(in) :: days, seconds

   Given a time and an increment of days and seconds (both positive definite, seconds not necessarily ¡ 86400) returns a time that adds this increment to an input time.

4. **time = decrement_time**(time, seconds, days)

   type (time_type) :: decrement_time
   type(time_type), intent(in) :: time
   integer, intent(in) :: days, seconds

   Given a time and a decrement of days and seconds (both positive definite, seconds not necessarily ¡ 86400) returns a time that subtracts this decrement from an input time. If the result is negative, it is considered a fatal error.

5. **b = interval_alarm**(time, time_interval, alarm, alarm_interval)

   logical :: incremental_alarm
   type(time_type), intent(in) :: time, time_interval, alarm_interval
   type(time_type), intent(inout) :: alarm_interval

   This is a specialized operation that is frequently performed in models. Given a time,

and a time interval, the interval alarm function is true if this is the closest time step to the alarm time. The actual computation is:
if ((alarm_time - time) <= (time_interval / 2))
If the function is true, the alarm time is incremented by the alarm_interval; WARNING, this is a featured side effect. Otherwise, the function is false and there are no other effects. CAUTION: if the alarm_interval is smaller than the time_interval, the alarm may fail to return true ever again.

6. **b = set_switch**(switch_interval, time_since, dt_time)

   logical :: set_switch
   type(time_type), intent(in) :: switch_interval, time_since, dt_time

   This is a specialized operation that is frequently performed within MOM. Given a switch interval, a time since some reference date, and a time step, the set_switch function is either true or false based on the following operation:
   set_switch = mod (time_since,switch_interval) <= dt_time/2

15.1.9.4 Calendar Interfaces

The following are subroutine and function interfaces for working with dates:

1. **call set_calendar_type**(type)

   integer, intent(in) :: type
   Sets the default calendar type for mapping time intervals to dates. At present, four integer constants are defined for setting the calendar type: THIRTY_DAY_MONTHS, JULIAN, NO_LEAP, and GREGORIAN. Note: the GREGORIAN calendar has not been fully implemented as of this writing.

2. **n = get_calendar_type**()

   integer :: get_calendar_type
   Returns the value of the default calendar type as an integer. Users should test this returned value using the predefined calendar type parameters: THIRTY_DAY_MONTHS, JULIAN, NO_LEAP, GREGORIAN, and NO_CALENDAR. NO_CALENDAR is the default value of the calendar type.

3. **time = set_date**(year, month, day, hours, minutes, seconds)

   type(time_type) :: set_date
   integer, intent(in) :: day, month, year
   integer, intent(in), optional :: seconds, minutes, hours

   Given an input date in seconds, minutes, etc., creates a time_type that represents this time interval from the internally defined base date. Optional arguments that are not present are set to 0. For the **julian** and **thirty_day_months** calendars, the base date is 0 seconds, 0 minutes, 0 hours, day 1, month 1, year 1. For the **Gregorian** calendar, the base date is 1 January, 1900, 0 seconds, 0 minutes, 0 hours. Negative time intervals are

not supported, so one cannot represent dates before the base dates. Illegal dates result in an error. The default calendar type as set in **set_calendar_type** is used. The functions **set_date_julian**, **set_date_gregorian**, **set_date_thirty**, and **set_date_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

4. **call get_date**(time, year, month, day, hours, minutes, seconds)

   type(time_type), intent(in) :: get_date
   integer, intent(out) :: seconds, minutes, hours, day, month, year

   Given a time_interval, returns the corresponding date under the default calendar (see set_date). Subroutines **get_date_julian**, **get_date_gregorian**, **get_date_thirty**, and **set_date_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

5. **time = increment_date**(time, years, months, days, hours, minutes, seconds)

   type(time_type) :: increment_date
   type(time_type), intent(in) :: time
   integer, intent(in), optional :: seconds, minutes, hours, days, months, years

   Increments the date represented by a time interval and the default calendar type by a number of seconds, etc. For all but the thirty_day_months calendar, increments to months and years must be made separately from other units because of the non-associative nature of the addition. All the input increments must be positive. Subroutines **increment_julian**, **increment_gregorian**, **increment_thirty**, and **increment_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

6. **time = decrement_date**(time, years, months, days, hours, minutes, seconds)

   type(time_type) :: decrement_date
   type(time_type), intent(in) :: time
   integer, intent(in), optional :: seconds, minutes, hours, days, months, years

   Decrements the date represented by a time interval and the default calendar type by a number of seconds, etc. For all but the thirty_day_months calendar, decrements to months and years must be made separately from other units because of the non-associative nature of addition. All the input decrements must be positive. Subroutines **decrement_julian**, **decrement_gregorian**, **decrement_thirty**, and **decrement_no_leap** are also available with the same interface allowing users to circumvent the default calendar type. If the result is a negative time (i.e. date before the base date) it is considered a fatal error.

7. **n = days_in_month**(time)

   integer :: days_in_month
   type(time_type), intent(in) :: time

   Given a time interval, gives the number of days in the month corresponding to the default calendar. Subroutines **days_in_month_julian**, **days_in_month_gregorian**, **days_in_month_thirty**

and **days_in_month_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

8. **b =leap_year**(time)

   logical :: leap_year
   type(time_type), intent(in) :: time

   Returns true if the year corresponding to the date for the default calendar is a leap year. Returns false for THIRTY_DAY_MONTHS and NO_LEAP. Subroutines **leap_year_julian**, **leap_year_gregorian**, **leap_year_thirty**, and **leap_year_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

9. **time = length_of_year**()

   type(time_type) :: time

   Returns the mean length of the year in the default calendar setting. Subroutines **length_of_year_julian**, **length_of_year_gregorian**, **length_of_year_thirty**, and **length_of_year_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

10. **n = days_in_year**(time)

    integer :: days_in_year
    type(time_type), intent(in) :: time

    Returns the number of days in the calendar year corresponding to the date represented by time for the default calendar. Subroutines **days_in_year_julian**, **days_in_year_gregorian**, **days_in_year_thirty**, and **days_in_year_no_leap** are also available with the same interface allowing users to circumvent the default calendar type.

11. **s = month_name**(n)

    character*9 :: month_name
    integer, intent(in) :: n

    Returns a character string containing the name of the month corresponding to month number n. Definition is the same for all calendar types.

15.1.9.5 Sample test program

The following is a sample program that shows some of the capabilities of the time manager. The driver within the module is different as illustrates how time computations are handled within MOM. The driver can be executed with script *run_tmngr*.

```
program main

use time_manager_mod
```

```
implicit none
type(time_type) :: dt, init_date, astro_base_date, time, final_date
type(time_type) :: next_rad_time, mid_date type(time_type) ::
repeat_alarm_freq, repeat_alarm_length integer :: num_steps, i, days,
months, years, seconds, minutes, hours integer :: months2, length real
:: astro_days


!Set calendar type
!call set_calendar_type(THIRTY_DAY_MONTHS)
!call set_calendar_type(NO_LEAP)
call set_calendar_type(JULIAN)

! Set timestep
dt = set_time(1100, 0)

! Set initial date
init_date = set_date(1992, 1, 1)

! Set date for astronomy delta calculation
astro_base_date = set_date(1970, 1, 1, 12, 0, 0)

! Copy initial time to model current time
time = init_date

! Determine how many steps to do to run one year
final_date = increment_date(init_date, years = 1)
num_steps = (final_date - init_date) / dt
write(*, *) 'Number of steps is ', num_steps

! Want to compute radiation at initial step, then every two hours
next_rad_time = time + set_time(7200, 0)

! Test repeat alarm repeat_alarm_freq = set_time(0, 1)
repeat_alarm_length = set_time(7200, 0)

! Loop through a year do i = 1, num_steps

! Increment time
   time = time + dt

! Test repeat alarm
   if (repeat_alarm(time, repeat_alarm_freq, repeat_alarm_length)) then
      write(*, *) 'REPEAT ALARM IS TRUE'
   endif

! Should radiation be computed? Three possible tests.
```

```
! First test assumes exact interval; just ask if times are equal
!   if(time == next_rad_time) then
! Second test computes rad on last time step that is <= radiation time
!   if((next_rad_time - time) < dt .and. time < next_rad) then
! Third test computes rad on time step closest to radiation time
   if (interval_alarm(time, dt, next_rad_time, set_time(7200, 0))) then
      call get_date(time, years, months, days, hours, minutes, seconds)
      write(*, *) days, month_name(months), years, hours, minutes,seconds

! Need to compute real number of days between current time and astro_base
      call get_time(time - astro_base_date, seconds, days)
      astro_days = days + seconds / 86400.
!     write(*, *) 'astro offset ', astro_days
   end if

! Can compute daily, monthly, yearly, hourly, etc. diagnostics as for rad
! Example: do diagnostics on last time step of this month
   call get_date(time + dt, years, months2, days, hours, minutes, seconds)
   call get_date(time, years, months, days, hours, minutes, seconds)
   if (months /= months2) then
      write(*, *) 'last timestep of month' write(*, *) days, months,
      years, hours, minutes, seconds
   endif

! Example: mid-month diagnostics; inefficient to make things clear
   length = days_in_month(time)
   call get_date(time, years, months, days, hours, minutes, seconds)
   mid_date = set_date(years, months, 1) + set_time(0, length) / 2
   if (time < mid_date .and. (mid_date - time) < dt) then
      write(*, *) 'mid-month time' write(*, *) days, months, years, hours, minutes, seconds
   endif
end do

end program main
```

15.1.9.6 Logical Switches

Once model time corresponding to the current time step has been calculated by module *tmngr*, a long list of logical variables are set based on input variables specified by the researcher and the current model time. The logical variables act as switches which indicate whether specific events (diagnostics, mixing time steps, end of month, end of week, Tuesday, etc.) will happen or not during the current time step. They are computed in subroutine *set_time_switches* located in file *switch.F*.

Include file *switch.h* contains all logical switches along with a long list of input variables used by the researcher to specify the interval in days between various requested events[3] or the

[3]The interval must be referenced to a starting time which may be specified as the beginning of the experiment, the beginning of a particular job, or a specific date in time.

period in days for averaging diagnostic quantities[4]. The list of input variables can be accessed through namelist and each input variable must have an associated logical switch. The logical switch is set to *true* when within half a time step of the requested interval or averaging period, otherwise it is set to *false*. Also associated with each switch is an internal variable containing the time when the logical switch will next be *true*.

In principle, any question regarding time can have a logical switch associated with it. It is easy to add new switches following the examples in files *switch.h* and *switch.F*. As a naming convention, all logical switches end in *ts*. One example is *snapts* which is the logical switch for determining whether a snapshot of the model solution is to be taken during the current time step. Use the UNIX *grep snapts ∗.[Fh]* to see where the switch *snapts* is defined and calculated, then replicate the form. Before adding a new switch, check through file *switch.h* because the switch may already be there. The following examples further indicate how to add a logical switch for various purposes in MOM:

How to set a Switch based on an interval.

Suppose the intent was to compute global energetics every 15.0 days. The following two variables would be added (they are already there) to the common block in file *switch.h*. Let *glen* be the desired mnemonic for global energetics, then the naming convention used in *switch.h* implies the following

- glenint = an interval (real) for diagnostic output.
- glents = a switch (logical) corresponding to the interval.

The researcher specifies the interval [e.g., glenint] for diagnostic output in units of days through namelist. Subroutine *set_time_switches* sets the corresponding logical switch [e.g., glents] every time step. It is set to true when within half a time step of the requested interval, otherwise it is false. All decisions relating to the interval [e.g., glenint] are based on the logical switch [e.g., glents]. The following statement placed inside the time manager in the section for switches based on an interval will calculate the logical switch.

glents = set_switch (glenint, time_since_base, dt_time)

In the above, variables *time_since_base* and *dt_time* are time structures related to a specified starting date and the model time step. Switch *glents* is updated every time step and is available anywhere in MOM by including file *switch.h* in the routine where the switch is needed. Other examples of ways to set switches can be seen in the driver for the time manager.

15.1.10 topog.F

File *topog.F* contains the topography module which is used to design a topography and geometry for the particular resolution specified by module *grids*. The module is exercised in stand alone mode by script *run_topog* which enables option *drive_topog*. This is the recommended way of generating topography and geometry. When option *drive_topog* is not enabled, the geometry and topography are constructed from with the model. Before executing module *topog*, the grid must be defined using module *grids*. Refer to Chapter 18 for a description of how to construct geometry and topography.

[4]Each event may have its own interval and averaging period.

15.1.11 util.F

File *util.F* contains the utility module. It is exercised in stand alone mode by script *run_util* which enables option *test_util*. The module is a collection of various subroutines or utilities used throughout MOM. All output from these utilities is passed through argument lists. Input to the utilities is also passed through argument lists, except for dimensional information which is passed by including file *size.h* within each utility. The include file is used for purposes of parallelization. If it were not for this include file, this module would be truly "plug compatible" and highly portable. The following is a summary of subroutines within the utility module:

15.1.11.1 indp

Function *indp* (index of nearest data point) searches an array to find which element is closest to a specified value and returns the index of that array element.

15.1.11.2 ftc

Subroutine *ftc* (fine to coarse) interpolates data defined on a fine resolution grid to a coarser resolution grid. It does this by averaging together all cells on the fine grid which lie within each coarse cell. Partial overlapping areas are taken into account.

15.1.11.3 ctf

Subroutine *ctf* (coarse to fine) linearly interpolates data defined on a coarse resolution grid to a fine resolution grid.

15.1.11.4 extrap

Subroutine *extrap* extrapolates data defined at ocean cells to land cells on the same grid. The intent is to force oceanographic quantities into land areas to allow reasonable values for interpolations involving coastlines. This is important in air/sea coupled models where coastlines and resolution may not match between models.

15.1.11.5 setbcx

Subroutine *setbcx* sets boundary conditions on arrays.

15.1.11.6 iplot

Subroutine *iplot* plots an integer array with characters thereby giving a quick "contour" map of the data.

15.1.11.7 imatrix

Subroutine *imatrix* prints values of an integer array in matrix format.

15.1.11.8 matrix

Subroutine *matrix* prints values of an real array in matrix format.

15.1.11.9 scope

Subroutine *scope* interrogates an array for the minimum, maximum and simple unweighted average. It also list their positions within the array.

15.1.11.10 sum1st

Subroutine *sum1st* performs a simple sum on the first index of and array for each value of the second index.

15.1.11.11 plot

Subroutine *plot* contours an array by dividing the array values into bins and assigning a character to each bin. It then prints the characters to produce a quick "contour" map. The array must be real.

15.1.11.12 checksum

Function *checksum* calculates a checksum for a two dimensional array and returns the value. The value is not printed.

15.1.11.13 print_checksum

Subroutine *print_checksum* prints a checksum for a two dimensional array.

15.1.11.14 wrufio

Subroutine *wrufio* writes an array as an unformatted fortran write. The purpose is to speed up writing by eliminating the indexed list when writing sub-sections of arrays.

15.1.11.15 rrufio

Subroutine *rrufio* reads an array as an unformatted fortran read. The purpose is to speed up reading by eliminating the indexed list when reading into sub-sections of arrays.

15.1.11.16 tranlon

Subroutine *tranlon* is only used in limited domain models to move the Greenwich meridian outside of the limited grid domain. It translates data in longitude and redefines longitudes to accommodate interpolating data which starts and ends at the Greenwich meridian. Recall that for interpolating data, grid coordinates must be monotonically increasing with increasing index.

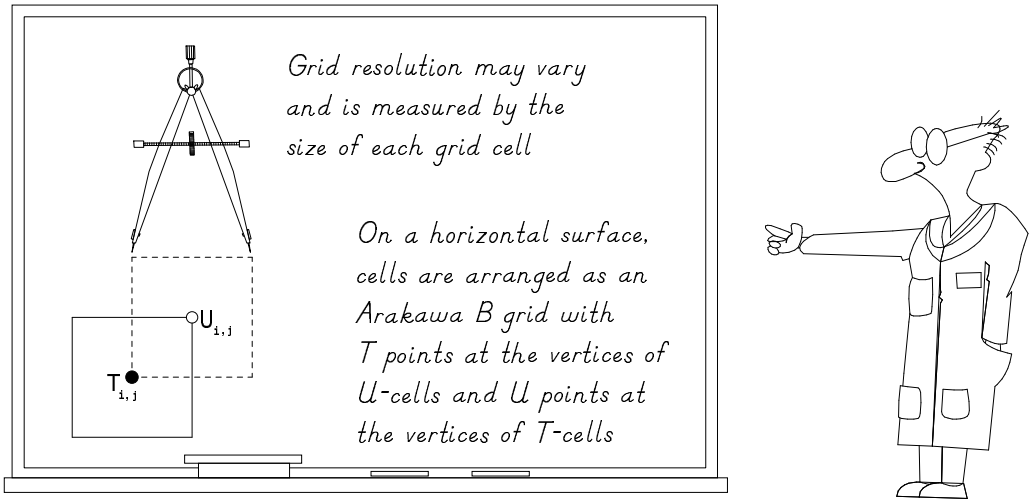
Part IV

Grids, Geometry, and Topography

Chapter 16

Grids

Before constructing surface boundary conditions, initial conditions, or executing a model, the model domain and resolution must be specified. The specification takes place within module *grids* which is contained in file grids.F.



16.1 Domain and Resolution

The model domain, which may be global, is defined as a thin shelled volume bounded by six coordinates: two latitudes, two longitudes, and two depths on the surface of a spherical earth. Embedded within this domain is a "rectangular" grid system aligned such that the

principle directions are along longitude λ (measured in degrees east of Greenwich), latitude ϕ (measured in degrees north of the equator), and depth zt (measured in centimeters from the surface of the spherical shell to the ocean bottom). Note that the vertical coordinate z increases upwards.

16.1.1 Regions

The domain may be further sub-divided into regions; each of which is similarly bounded by six coordinates: two latitudes, two longitudes, and two depths. Within each region, resolution can be specified as constant or non-uniform along each of the coordinate directions. Although non-uniform resolution is permitted, it is not allowed in the sense of generalized curvilinear coordinates. Resolution along any coordinate is constrained to be a function of position along that coordinate. For instance, vertical thickness of grid cells may vary with depth but not with longitude or latitude.

16.1.2 Resolution

The resolution within a region is determined by the width of the region and resolution at the bounding coordinates. Refer to Fig 16.1 as an illustration of specifying resolution within regions of longitude, latitude, and depth. Along any coordinate, if resolution at the bounding coordinates is the same, then resolution is constant across the region, otherwise it varies continuously from one boundary to the other according to an analytic function. The function describing the variation is prescribed to be a cosine. Although arbitrary, this function has two important properties: it allows the average resolution within any region to be calculated as an average of the two bounding resolutions; and it insures that the first derivative of the resolution vanishes at the region's boundaries. A vanishing first derivative allows regions to be smoothly joined. The only restriction is that there is an integral number of grid cells within a region.

To formalize these ideas, let a region be bounded along any coordinate direction (latitude, longitude, or depth) by two points α and β at which resolutions are Δ_α and Δ_β. The number of discrete cells N contained between α and β is given by

$$N = \frac{|\beta - \alpha|}{(\Delta_\alpha + \Delta_\beta)/2} \tag{16.1}$$

where N must be an integer and the resolution for any cell Δ_m is given by

$$\Delta_m = \frac{\Delta_\alpha + \Delta_\beta}{2} - \frac{\Delta_\beta - \Delta_\alpha}{2} \cos(\pi \frac{m - 0.5}{N}) \tag{16.2}$$

where $m = 1 \cdots N$. As an example, if α and β were longitudes, the western edge of the first cell would be a α and the eastern edge of cell N would be at β.

16.1.3 Describing a domain and resolution

The model domain is composed of one or more regions in latitude, longitude, and depth. Latitude and longitude are specified in degrees east of Greenwich and depth is in centimeters from the surface downwards. Each region is defined by its bounds and resolution at those bounds. Within each region, resolution may be constant or smoothly varying but there must be an integral number of grid cells contained within the region's bounds.

If the resolution at both bounds of a region is the same, then resolution within the region is constant. If the bounding resolutions differ, resolution varies smoothly across the region to make the transition. The functional for the variation is arbitrarily taken to be a cosine. Regions sharing a common boundary have the same resolution at that boundary and the first derivative is zero at the boundary to minimize effects of discrete jumps in the numerics. In the vertical, the last region allows a stretching factor applied to the analytic function to provide a more drastic fall off of resolution to the bottom if desired.

In each of the following examples, a grid domain and resolution is built by specifying bounding coordinates and resolution for each region.

16.1.3.1 Example 1: One resolution domain

Imagine a grid with a longitudinal resolution $\Delta_\lambda = 1^\circ$ encircling the earth and a meridional resolution $\Delta_\phi = 1/3^\circ$ equatorward of 10°N and 10°S, and a vertical grid spacing $\Delta_z = 10$ meters between the surface and a depth of 100 meters. This domain and resolution is specified in the following manner. Two bounding longitudes: one at 0°E and the other as 360°E with $\Delta_\lambda = 1^\circ$ at both longitudes; two bounding latitudes: one at -10° and the other at $+10^\circ$ with $\Delta_\phi = 1/3^\circ$ at both latitudes; and two bounding depths: one at 0cm and the other at 100x10^2cm with $\Delta_z = 10$x10^2cm at both depths. These specifications imply 360 grid cells in longitude, 60 grid cells in latitude, and 10 grid cells in depth[1]

The above is specified within the USER INPUT section of module *grids* by the following code:

```
c
c     "nxlons"  = number of bounding longitudes to define 1 region
c     "x_lon"   = bounding longitudes {0.0E, 360.0E}
c     "dx_lon"  = resolution centered at "x_lon" {1.0, 1.0}
c
       parameter (nxlons=2)
       data (x_lon(i), i=1,nxlons)  /0.0, 360.0/
       data (dx_lon(i),i=1,nxlons)  /1.0,   1.0/
c
c     "nylats"  = number of bounding latitudes to define 1 region
c     "y_lat"   = bounding latitudes {-10.0, 10.0}
c     "dy_lat"  = resolution centered at "y_lat" {1/3, 1/3}
c
       parameter (nylats=2)
       data (y_lat(j), j=1,nylats)  /-10.0,        10.0/
       data (dy_lat(j),j=1,nylats)  /  0.3333333,   0.3333333/
c
c     "nzdepths"  = number of bounding depths to define 1 region
c     "z_depth"   = bounding latitudes {0.0, 100.0m}
c     "dz_depth"  = resolution centered at "z_depth" {10m, 10m}
c
       parameter (nzdepths=2)
```

[1]Actually, two extra boundary cells are added to the grid domain in the latitude and longitude dimensions, but not in the vertical (historical reasons). Calculations range from $i = 2$ to $imt - 1$ in longitude, $jrow = 2$ to $jmt - 1$ in latitude, and $k = 1$ to km in depth where domain size is (imt x jmt x km) cells.

```
      data (z_depth(k), k=1,nzdepths)  / 0.0,   100.0e2/
      data (dz_depth(k),k=1,nzdepths)  /10.e2,   10.e2/
```

Note that "x_lon" marks the edges of T-cells in longitude. Similarly, "y_lat" marks the edges of T-cells in latitude, and "z_depth" marks the edges of T-cells in depth. Also, "dx_lon" is the desired grid resolution at "x_lon" which is the longitude of a U-cell (not the T-cell). Since MOM uses an Arakawa B grid, the U-cell grid points are at the vertices of the T-cells and visa versa when looking down at the grid toward increasing depth. A U-cell with coordinates (i,j) is on the northeast vertex of a T-cell with coordinates (i,j). In the vertical, U cells and T cells are at the same level. More on this later.

16.1.3.2 Example 2: Two resolution domains

In the preceeding example, if it were desired to extend the latitudinal domain poleward of 10°N and 10°S to 30°N and 30°S where the meridional resolution was to be $\Delta_\phi = 1°$, then the two previous bounding latitudes would need to be replaced by four: one at $-30°$ where $\Delta_\phi = 1°$, one at $-10°$ where $\Delta_\phi = 1/3°$, one at $+10°$ where $\Delta_\phi = 1/3°$ and one at $+30°$ where $\Delta_\phi = 1°$. Poleward of 10 degrees, meridional resolution would telescope from $\Delta_\phi = 1/3°$ to $\Delta_\phi = 1°$ over a span of 20°. The average meridional resolution in this region is calculated as the average of the bounding resolutions which is $\frac{1°+1/3°}{2} = 2/3°$. Therefore, there would be $\frac{20°}{2/3°} = 30$ additional grid cells in each hemisphere between latitudes 10° and 30°. This would be specified in the code as follows:

```
c
c     "nylats"  = number of bounding latitudes to define 3 region
c     "y_lat"   = bounding latitudes {-30, -10.0, 10.0, 30}
c     "dy_lat"  = resolution centered at "y_lat" {1, 1/3, 1/3, 1}
c
       parameter (nylats=4)
       data (y_lat(j), j=1,nylats)  /-30.0, -10.0,  10.0,      30.0/
       data (dy_lat(j),j=1,nylats)  /1.0, 0.3333333, 0.3333333, 1.0/
```

16.1.3.3 Example 3: Horizontally isotropic grid

Suppose it was desired to construct a square grid between latitude 65°S and 65°N with 3° resolution at the equator. The condition is that $\Delta_x = \Delta_y = 3°$ at the equator and in general Δ_y must shrink with latitude to match the decreasing Δ_x due to the convergence of the meridians. Consequently,

$$\Delta_y = \Delta_x \cdot \cos\phi. \tag{16.3}$$

Enabeling option *isotropic_grid* enforces the above condition and yields a square grid between the bounding longitudes which are set as follows:

```
c
c     "nxlons"  = number of bounding longitudes to define 1 region
```

```
c     "x_lon"   = bounding longitudes {0.0E, 360.0E}
c     "dx_lon"  = resolution centered at "x_lon" {3.0, 3.0}
c
       parameter (nxlons=2)
       data (x_lon(i), i=1,nxlons)  /0.0, 360.0/
       data (dx_lon(i),i=1,nxlons)  /3.0,   3.0/
c
c     "nylats"  = number of bounding latitudes to define 1 region
c     "y_lat"   = bounding latitudes {-65, 65}
c     "dy_lat"  = resolution centered at equator {3.0, 3.0}
c
       parameter (nylats=2)
       data (y_lat(j), j=1,nylats)  /-65.0, 65.0/
       data (dy_lat(j),j=1,nylats)  /  3.0,  3.0/
```

The above specification yields a regional domain with 58 grid cells in latitude between 65.284°N and 65.284°S. Resolution is 3.0 degrees on the equator and about 1.26 degrees at the latitudinal boundaries. Note that the regional bounds have been adjusted to fit an integral number of cells.

The isotropic regional domain can be extended to a global domain by enabeling option *extend_isotropic_grid*. Note that to fit an integral number of cells between 65.284° and the poles requires a slight increase in Δ_y poleward of 65. At the poles, $\Delta_y = 1.34$. In general, $Delta_y$ increases slighlty outside the isotropic region to insure the integral constraint.

16.2 Grid cell arrangement

The grid system is a rectangular Arakawa staggered B grid (Bryan 1969) containing T-cells and U-cells. In order to visualize this arrangement, it will be helpful to refer to Figs 16.2, 16.3, and 16.4 which depict grid cells within horizontal and vertical surfaces. Within each T cell is a T grid point which defines the location of tracer quantities. Similarly, each U cell contains a U grid point which defines the location of the zonal and meridional velocity components.

16.2.1 Relation between T and U cells

Within a horizontal surface at depth level k, grid points and cells are arranged such that a grid point $U_{i,k,j}$ (where subscript i is the longitude index, j is the latitude index, and k is the depth index) is located at the northeast vertex of cell $T_{i,k,j}$. Conversely, the grid point $T_{i,k,j}$ is located at the southwest vertex of cell $U_{i,k,j}$. The southwestern most column of T-cells has indices "i=1" and "jrow=1". Similarly, the northeasternmost column of T-cell within the grid has indices "i=imt" and "jrow=jmt". This horizontally staggered grid system is replicated and distributed vertically between the ocean surface and bottom of the domain. $\mathrm{T}_{i,k=1,j}$ is the first cell below the surface and $\mathrm{T}_{i,k=km,j}$ is the deepest cell. Unlike in the horizontal, T cells and U cells are not staggered vertically so all T cells and U cells with index k are at the same depth.

16.2.2 Regional and domain boundaries

As mentioned in Section 16.1, when specifying bounding coordinates and resolution, there must be an integral number of cells contained between these coordinates in each of the coor-

dinate directions. The integral number of cells refers specifically to T cells. In the horizontal plane, bounding surfaces of constant longitude and latitude define the location of U grid points and resolution at those surfaces is given to the corresponding U cells. Resolution varies continuously between bounding surfaces and is discretized, using Equations (16.1) and (16.2), to cell widths and heights. In the vertical plane, bounding surfaces are at the top or bottom of cells and resolution is discretized to cell thicknesses as in the horizontal plane. It should be noted that in the vertical, allowance is made for a stretching factor in the last region to provide for a more drastic fall off of resolution to the bottom if desired.

16.2.3 Non-uniform resolution

Within a region of constant resolution, all T and U grid points are located at the centers of their respective cells. When resolution is non-uniform, this is not the case. Within MOM 2, there are two methods to discretize non-uniform resolution onto T cells and U cells. Based on Equations (16.1) and (16.2) in section 16.1, and the averaging operator given by

$$\overline{(\xi)}^m = \frac{(\xi)_m + (\xi)_{m+1}}{2}, \tag{16.4}$$

the two methods are

1. $\Delta^T_m = \frac{\Delta^U_\alpha + \Delta^U_\beta}{2} - \frac{\Delta^U_\alpha - \Delta^U_\beta}{2}\cos(\pi\frac{m-0.5}{N}); \quad \Delta^U_m = \overline{\Delta^T_m}^m$
2. $\Delta^U_m = \frac{\Delta^U_\alpha + \Delta^U_\beta}{2} - \frac{\Delta^U_\alpha - \Delta^U_\beta}{2}\cos(\pi\frac{m}{N}); \quad \Delta^T_m = \overline{\Delta^U_m}^m$

where Δ^U_α and Δ^U_β are resolution of U cells at the bounding surfaces α and β, N is the number of cells given by Equation (16.1) in section 16.1, and the subscript m refers to longitude index i or latitude index j. These methods can also be applied to cells in the vertical, even though T and U cells are not staggered vertically. Refer to Figure 16.5 and assume phantom W cells (of thickness dzw_k) staggered vertically such that the W grid point within cell W_k lies at the bottom of cell T_k and the T grid point within cell T_k lies at the top of cell W_k. Now replace U by W in the expressions for both methods given above.

The motivation for method 2 is first to notice that on a non-uniform grid, advective velocities are a weighted average of velocities but the denominator is not the sum of the weights as indicated in Section 22.3. This form of the advective velocities is implied by energy conservation arguments as given in Section A.2.4. Secondly, the average of the quantity being advected is not defined coincident with the advecting velocity. Redefining the average operator in Equation (16.4) differently results in second moments not being conserved as indicated in Chapter A. Method 2 remedies both problems by simply redefining the location of grid points within grid cells. All equations remain the same. Both method 1 and 2 conserve second moments.

In method 1, U cell size is the average of adjacent T cell sizes. This means that T points are always centered within T cells, but U points are off center when the grid is non-uniform. This was the method used in model versions prior to MOM 2. In method 2, the construction is the other way around: T cell size is the average of adjacent U cell sizes. Accordingly, U points are always centered within U cells but T points are off center when the resolution is non-uniform. It should be noted that MOM 2 allows both methods. Although the default grid construction is by method 2, enabling option *centered_t*[2] when compiling will result in grid construction by method 1.

[2]Centered refers to the centering of the t grid point within the T grid cell.

16.2.3.1 Accuracy of numerics

When resolution is constant, the finite difference numerics are second order accurate. In this case, grid cells and grid points are at the same locations regardless of whether they are constructed using method 1 or method 2. However, contrary to widespread belief, when resolution is non-uniform, numerics are still second order accurate if the stretching is based on a smooth analytic function. See Treguier, Dukowicz, and Bryan (1995).

Even though methods 1 and 2 are second order accurate, is one slightly better than the other? In particular, does the horizontal staggering of grid cells implied by method 2 give slightly better horizontal advection[3] of tracers while the staggering implied by method 1 give more accurate horizontal advection of momentum? Also, since T cells and U cells are not staggered in the vertical, does method 2 gives more accurate vertical advection of tracers and momentum than method 1?

The reasoning behind these questions can be seen by referring to Figure 16.5 and noting the placement of T grid points within T cells[4]. Advective fluxes are constructed as the product of advective velocities and averages of quantities to be advected.

$$advective\ \ flux = W_k \cdot \frac{T_k + T_{k+1}}{2} \tag{16.5}$$

When resolution is constant, both advective velocity and averaged quantities lie on cell faces, but when resolution is non-uniform, averaged quantities may lie off the cell faces. Whether or not this happens depends on the placement of grid points within grid cells. Method 2 defines tracer points off center in such a way that the averaged tracer given by Equation (16.4) is placed squarely on the cell faces. Recall from Chapter A that defining the average operator differently than in Equation (16.4) will not conserve second moments.

Simple one dimensional tests using a constant advection velocity of 5 cm/sec to advect a gaussian shaped waveform through a non-uniform resolution varying from 2 to 4 degrees and back to 2 degrees suggests that method 2 is better than method 1. Also, a one dimensional thermocline model employing a stretched vertical coordinate indicates again that method 2 is better than method 1. In both cases, better means that the variance of the solution was closer to the variance of the analytic solution by a few percent. In short integrations using MOM, the effect showed up as less spurious creation of tracer extrema in grids constructed with method 2 as compared to method 1. Whether this difference is robust in all integrations has not been demonstrated. Nevertheless, in light of these results, the default grid construction in MOM is method 2. Method 1 can be implemented by using option *centered_t*.

It should be stressed, that regardless of which method of grid construction is used, the equations don't change and first and second moments are conserved. Of primary importance when constructing a grid is whether the physical scales are adequately resolved by the number of grid points. Beyond this, grid construction by method 1 or method 2 is of secondary importance.

16.3 Constructing a grid

The domain and resolution is constructed in MOM by calling module *grids* to construct a grid system. Executing MOM to design a grid is similar to using a sledge hammer to work with

[3] Of course, this is not to be compared to the significant gains due to higher order advective schemes.

[4] Although the argument is given for the vertical direction, it applies in the horizontal as well.

tacks. Instead, module *grids* can be executed in a much simpler environment (without the rest of MOM) by using script *run_grids*. This is a simple UNIX C shell script which assumes a Fortran 90 compiler. Although the script can be executed from the MOM_2 directory, it's a safer practice to use a MOM_UPDATES directory for each experiment as described in Section 3.2. Copy both *run_grids* and file *grids.F* into the MOM_UPDATES directory, and make changes to defined the grid there.

To define a grid, go to the USER INPUT section of module *grids*. After reading the information in this chapter and looking at the examples given in module *grids*, implementing a grid design should be straightforward. About the only potential problem might be that a particular specification leads to a non-integral number of T cells within a region. Recall that the number of T cells can be found by dividing the span of the region by the average resolution. Either the position of the bounding coordinates or the resolution at these coordinates may be changed to resolve the problem.

In the USER INPUT section, the bounding coordinates are specified by variables *x_lon*, *y_lat*, and *z_depth*. Variable *x_lon* is dimensioned by parameter *nxlons* which gives the number of bounding longitudes to define one or more regions in longitude. Units are in degrees of longitude measured at the equator and these points define the longitude of U grid points. Refer to Figs 16.2, 16.3, and 16.4 which indicate U grid points on bounding coordinates with an integral number of T cells inbetween. Similarly, variable *y_lat* is dimensioned by parameter *nylats* which gives the number of bounding latitudes to define one or more regions in latitude. These coordinates mark the position on U points in latitude. Variable *z_depth* is dimensioned by parameter *nzdepths* which gives the number of bounding depth coordinates to define one or more regions in depth. Units are in *cm* and mark where the position of the bases of T and U cells will be. Note that the bottoms of T and U cells define where the vertical velocity points are located.

Associated with variables *x_lon*, *y_lat*, and *z_depth* are variables which define the respective resolution *dx_lon*, *dy_lat*, and *dz_depth* of cells at the bounding coordinates. Units are in degrees for *dx_lon* and *dy_lat* but in *cm* for *dz_depth*. Note the variable *stretch_z* provides additional stretching for the last region in the vertical. When *stretch_z*= 1.0, there is no additional stretching. However, when *stretch_z* > 1.0 additional stretching is applied. To see how it works, set *stretch_z* $= 1.1$ and gradually increase it. It can be used to approximate an exponential fall off in the vertical.

When executed, script *run_grids* creates a sub directory and compiles module *grids* using option *drive_grids* to activate a driver as the main program. After executing, results are copied to a file in the directory from which script *run_grids* was executed and the sub-directory is eliminated. View the file *results_grids* with an editor. When satisfied, copy *size.h* from the MOM_2 directory and change the parameters according to the indicated directions. Any component of MOM which accesses the updated module *grids* and *size.h* will get the new grid. All components of MOM which use module *grids* and *size.h* perform a consistency check. If there is an inconsistency, MOM will give an error message and stop.

16.3.1 Grids in two dimensions

When the grid system is contracted to a minimum along one dimension, MOM is essentially reduced to a two dimensional model. For instance, if it is desirable to have a two dimensional model that is a function of latitude ϕ and depth z, then setting $nxlons = 2$ and specifying dx_lon_1, dx_lon_2 such that there are two grid cells between bounding surfaces x_lon_1, x_lon_2 will generate $imt = 4$ grid cells in the longitudinal direction. Two T cells $i = 2$ and $i = 3$

will be calculated and the two extra T cells $i = 1$ and $i = 4$ are for boundaries. Only one U cell, $i = 2$, is not in the boundary. If option *cyclic* is enabled, then the domain is zonally re-entrant. If the forcing and initial conditions are independent of longitude λ, then the solution is independent of λ and the model is two dimensional in ϕ and z. Obviously the relevance of this model depends on the scientific question being posed. This is just to demonstrate how the longitudinal dimension can be contracted to a minimum of $imt = 4$ cells. The memory window should be opened to $jmw = jmt$ to make it as efficient as possible but this will not be as fast as the two dimensional model in λ and z discussed below because of the short vector lengths in longitude.

A similar contraction can be performed in the latitudinal direction to end up with a two dimensional model in longitude λ and depth z. Here again, the minimum number of latitudes is $jmt = 4$ with $jrow = 2$ and $jrow = 3$ being ocean T cells and $jrow = 1$ and $jrow = 4$ being land cells. Note that there is only one latitude of ocean U cells. If these U cells are placed at the equator (with the two ocean T cell latitudes placed symmetrically about the equator) and there is no meridional variation in initial conditions or forcing, then the model is two dimensional in λ and z. Again, the scientific question being posed needs to be suited to this design. Note that this type of model really flies computationally because of the long vectors in the longitude dimension. The memory window should be opened to $jmw = jmt$ to make it as efficient as possible.

In the vertical, the minimum number of ocean levels is 2. Therefore the bounding surfaces and resolutions can be set to yield $km = 2$. Note that $kmt_{i,jrow} < 2$ is not allowed.

16.4 Summary of options

The following options are used by module *grids* when executing in a stand alone mode. They are enabled by compiling with options of the form *-Doption1 -Doption1* ... as in script *run_grids*.

- *drive_grids* turns on a small driver which allows the module *grids* to be executed in a simple environment (stand alone). This is only used for designing grids with the script *run_grids*. It is not appropriate when executing MOM.
- *generate_a_grid* generates a grid based on specifications in the USER INPUT section of module *grids*. It is used both in the script *run_grids* and when executing MOM.
- *read_my_grid* allows grids developed elsewhere to be imported into MOM.
- *write_my_grid* writes a copy of the grid information to file *grid.dta.out*.
- *centered_t* constructs a grid using method 1 from Section 16.2.3. If not enabled, the grid is constructed using method 2 from Section 16.2.3 which is different than the way it has been constructed in previous versions of the model.
- *isotropic_grid* constructs a square grid between two bounding latitudes.
- *extend_isotropic_grid* extends the isotropic grid to the poles using a nearly constant Δ_y.
- *bbl_ag* modifies the vertical grid to account for a bottom boundary layer.
- *symmetry* used when specifying a symmetry condition at the northern boundary of the domain. Note that symmetry can only be specified when the northern part of the domain is at the equator.

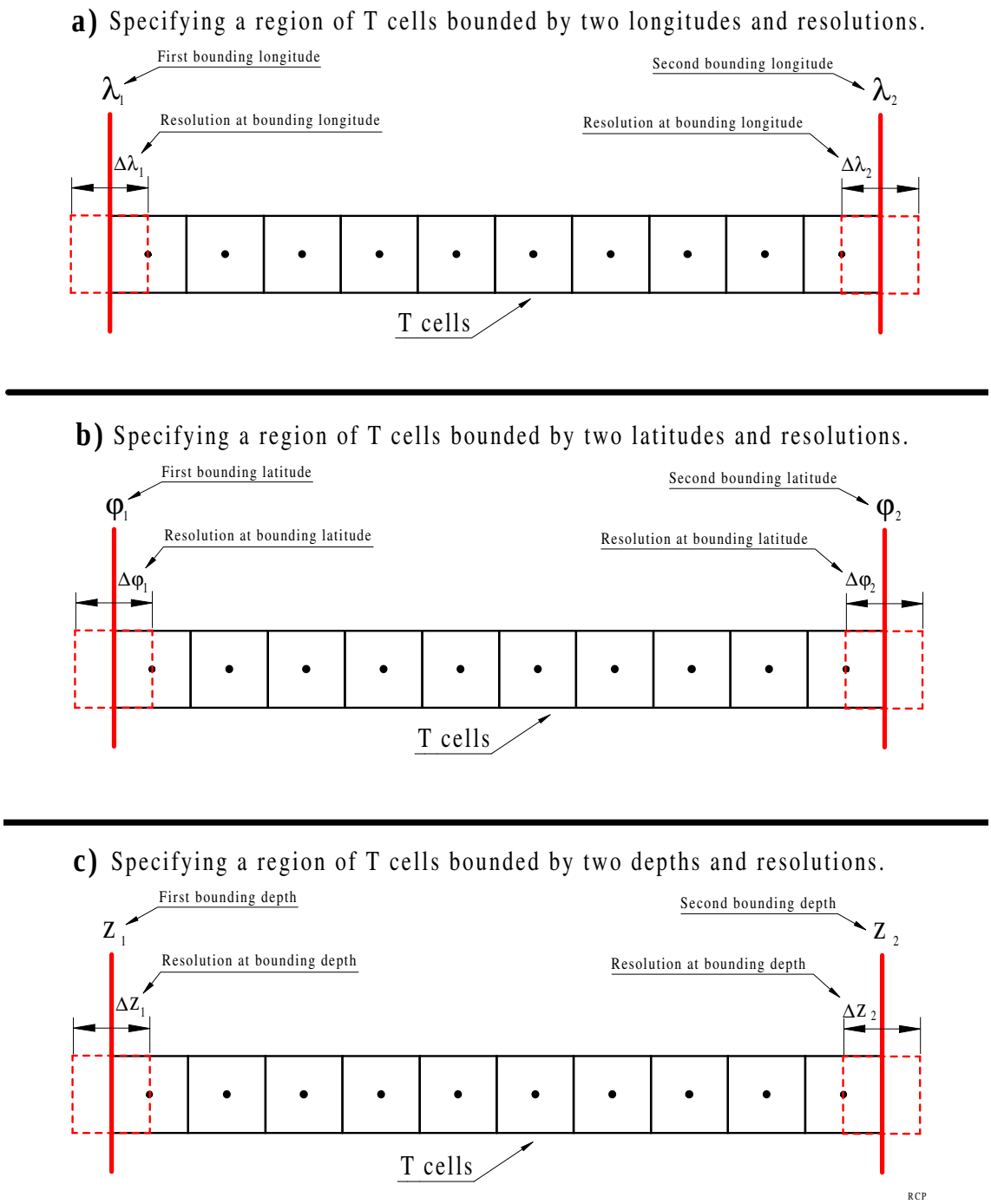


Figure 16.1: a) A region bounded by two longitudes. b) A region bounded by two latitudes. c) A region bounded by two depths.

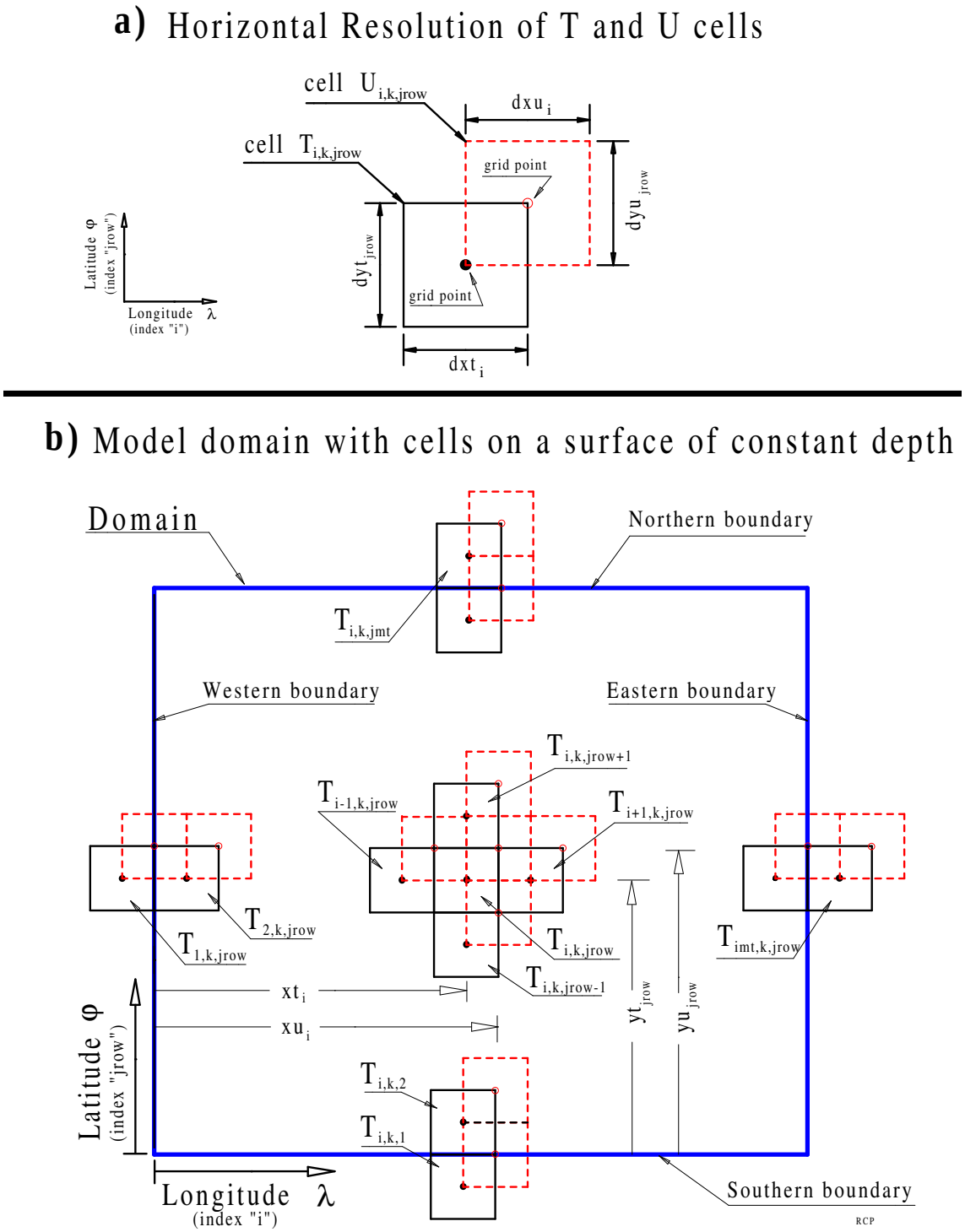


Figure 16.2: a) Horizontal resolution of grid cells. b) Grid cell arrangement on a longitude-latitude surface of constant depth.

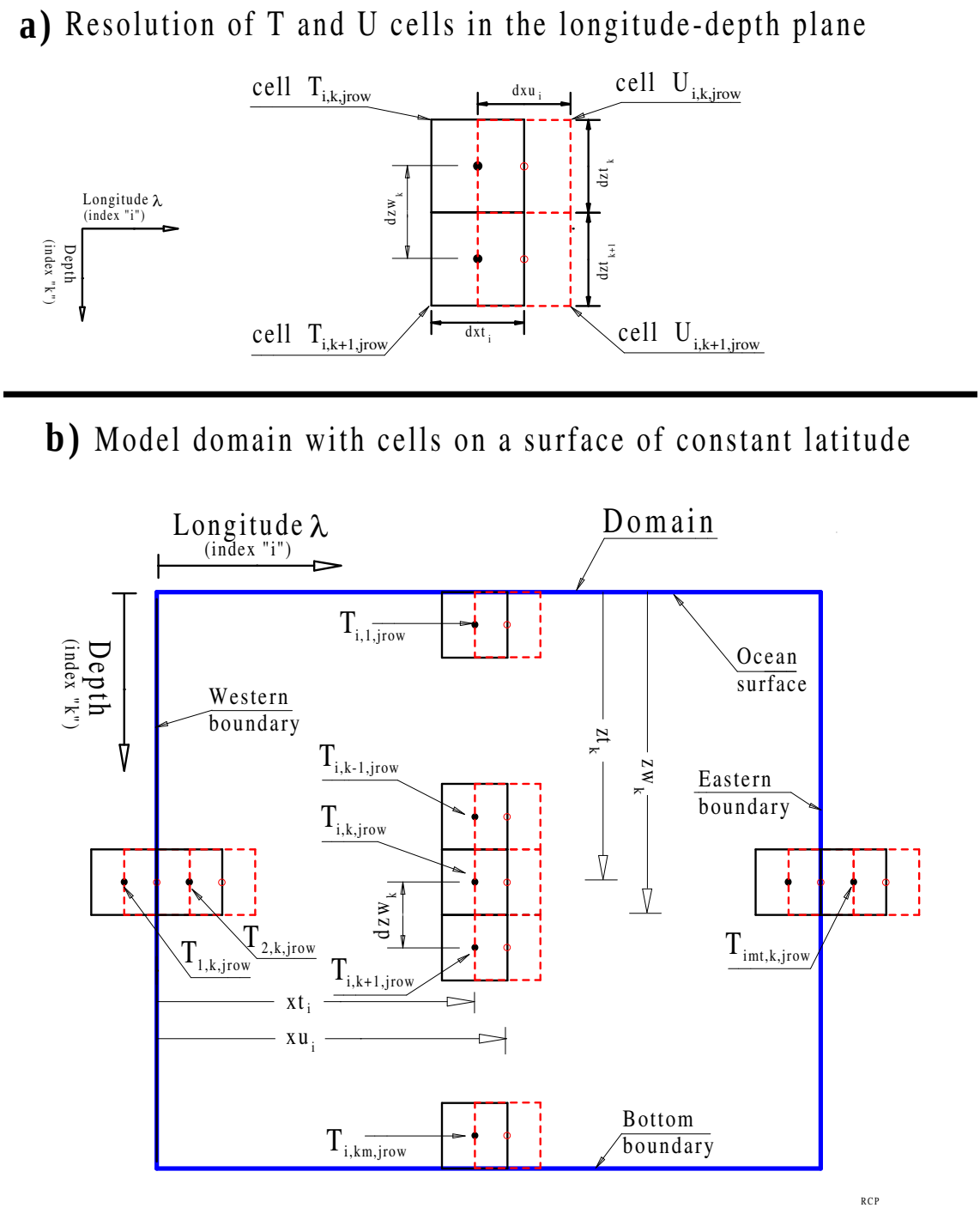


Figure 16.3: a) Vertical resolution of grid cells. b) Grid cell arrangement on a longitude-depth surface of constant latitude.

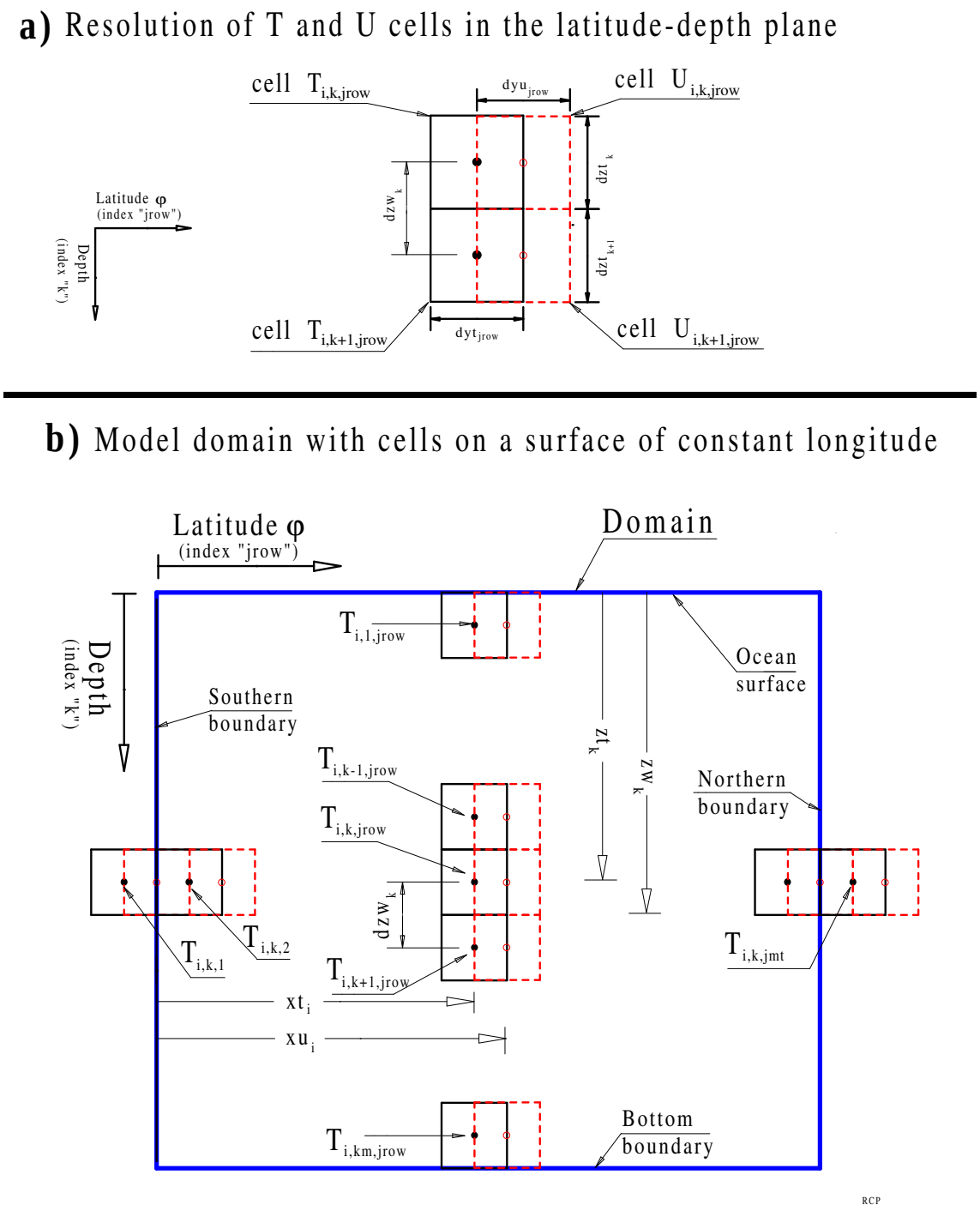


Figure 16.4: a) Vertical resolution of grid cells. b) Grid cell arrangement on a latitude-depth surface of constant longitude.

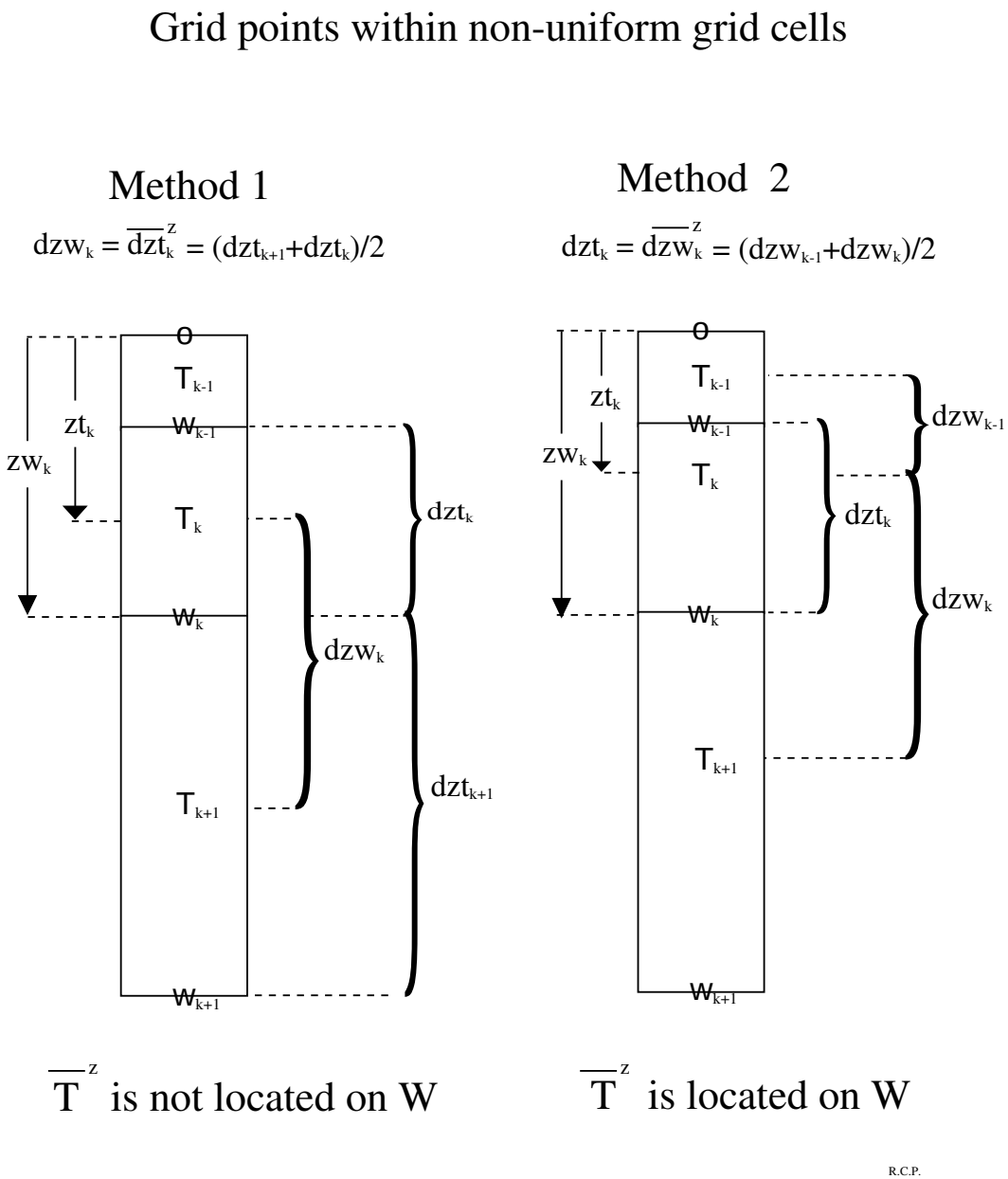


Figure 16.5: Comparison of two grid cell construction methods applied in the vertical dimension. Method 2 is the default in MOM.

Chapter 17

Grid Rotation

This chapter was contributed by Michael Eby (*eby@uvic.ca*). A grid rotation is one of the simplest types of grid transformations that can be applied to a spherically gridded model. This transform is particularly useful for studies of high latitude oceans where the convergence of lines of longitude may limit time steps or where the ocean contains a pole (as in the Arctic). The idea is to define a new grid in which the area of interest is far from the grid poles. In limited domain models, the pole can be rotated outside of the domain. For global models, one possibility is rotate the North Pole to (40° W,78° N) which puts the North pole in Greenland and keeps the South pole in Antarctica. Other uses include rotating the grid to match the angle of a coastline or to provide more flexibility in specifying lateral boundary conditions.

Option *rot_grid* modifies the Coriolis term to handle a rotated model grid which is specified using three Euler angles for solid body rotation. The Euler angles are computed by defining the geographic latitude and longitude of the rotated north pole and a point on the prime meridian as described below.

To make this option easier to use, several rotation routines are provided within module *rotation.F*. The driver may be used to help define the rotation, write a file of geographic latitudes and test the rotation with idealized data. Other routines demonstrate how to interpolate scalar and vector data from a geographically gridded data set, to a rotated model grid. To run the driver, use the script *run_rotation*.

17.1 Defining the rotation

Any spherical grid rotation can be specified by defining three solid body rotations. The angles which define the rotations are usually referred to as Euler angles (see "Classical Mechanics" by Goldstein, 1950 or a similar text). First, define the Z axis to be through the poles such that the X-Y plane defines the equator and the X axis runs through the prime meridian. In the routines (in rotation.F), the rotation angles are called *phir*, *thetar* and *psir*. The angle *phir* is defined as a rotation about the original Z axis. Angle *thetar* is defined as a rotation about the new X axis (after the first rotation) and angle *phir* is defined as a rotation about the final Z axis (see Figure 17.1).

It is helpful to have a globe to look at when thinking about this. Imagine that the globe has a clear sphere surrounding it, with only grid lines of latitude and longitude. By moving the outer sphere, the grid poles can be moved to line up with different points on the globe. Once the new poles are located, two of the rotation angles can be defined as follows. The definition for *phir* is 90 degrees minus the geographic longitude of the new north pole. This rotates the

Y axis under the new pole. To move the Z axis down, *thetar* is defined to be 90 degrees minus the geographic latitude of the new north pole. This places the original Z axis though the new north pole position.

To completely define the grid, a third rotation about the new Z axis, must be specified. The rotated grid longitude of any point on the geographic grid is still undefined. To specify this last rotation, choose a point on the geographic grid (the globe) to locate the rotated grid's prime meridian. Set angle *psir* to zero and calculate the longitude of this point on the rotated grid. This longitude is the final angle *psir*, the angle needed to rotate the point back to the prime meridian. The definition of *psir* is usually not very important since the new grid longitude is arbitrary, but it does make a difference in defining exactly where the new grid starts. This may be important if it is desirable to line up grids for nesting. Looking at Figure 17.1, it may appear that all of the angle definitions are of the opposite sign to what they should be, but this comes from thinking about rotating the axes rather than rotating the rigid body.

Generally, the idea is to move the poles so that they are 90 degrees away from the area of interest. For example, to set up a model with an equatorial grid over the Arctic and North Atlantic, the rotated grid north pole could be positioned at 0 N, 110 W and a prime meridian point at 0 N, O E. This defines a grid rotation in which the new grid equator is along the 20 W and 160 E meridians. The rotated grid longitude is east, north of the geographic equator and west to the south. On the rotated grid, North America is in the north and Europe in the south, and the geographic north pole is at 0 N, 90 E. It is more difficult if you want to specify an arbitrary grid rotation, but usually a few trials is enough to locate the necessary pole position.

Having defined the angles of rotation, an orthogonal transformation matrix can be written. The first rotation through an angle Φ about the z axis (counterclockwise looking down the -z direction) is given by

$$D = \begin{pmatrix} \cos\Phi & \sin\Phi & 0 \\ -\sin\Phi & \cos\Phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The second rotation through an angle Θ about the new x axis (counterclockwise looking down the -x direction) is given by

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Theta & \sin\Theta \\ 0 & -\sin\Theta & \cos\Theta \end{pmatrix}$$

and the final rotation through an angle Ψ about the z axis (counterclockwise looking down the -z direction) is given by

$$B = \begin{pmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Note that the total rotation A can be written as the product of the three rotations $A = BCD$ (ordering is important here).

$$A = \begin{pmatrix} \cos\Psi\cos\Phi - \cos\Theta\sin\Phi\sin\Psi & \cos\Psi\sin\Phi + \cos\Theta\cos\Phi\sin\Psi & \sin\Psi\sin\Theta \\ -\sin\Psi\cos\Phi - \cos\Theta\sin\Phi\cos\Psi & -\sin\Psi\sin\Phi + \cos\Theta\cos\Phi\cos\Psi & \cos\Psi\sin\Theta \\ \sin\Theta\sin\Phi & -\sin\Theta\cos\Phi & \cos\Theta \end{pmatrix}$$

Transforming points from the unrotated system to the rotated system (marked by primes) is given as

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = A \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{17.1}$$

and transforming points from the rotated system (marked by primes) to the unrotated system is given as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = A^{-1} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{17.2}$$

where the inverse transform A^{-1} is

$$A^{-1} = \begin{pmatrix} \cos\Psi\cos\Phi - \cos\Theta\sin\Phi\sin\Psi & -\sin\Psi\cos\Phi - \cos\Theta\sin\Phi\cos\Psi & \sin\Theta\sin\Phi \\ \cos\Psi\sin\Phi + \cos\Theta\cos\Phi\sin\Psi & -\sin\Psi\sin\Phi + \cos\Theta\cos\Phi\cos\Psi & -\sin\Theta\cos\Phi \\ \sin\Theta\sin\Psi & \sin\Theta\cos\Psi & \cos\Theta \end{pmatrix}$$

Note that the inverse transform A^{-1} is really just A with Ψ and Φ switched and all rotation angles made negative. In MOM, Equation (17.1) is used in routine "rotate" to produce coordinates with respect to the rotated frame.

17.2 Rotating Scalars and Vectors

Code changes to mom are limited to modifying the calculation and dimension of the Coriolis variable *cori*. Running the rotation driver will create the latitude data file needed to calculate the correct Coriolis terms, but the creation of all other data files is left to the researcher. The subroutine *rot_intrp_sclr* may be used to interpolate scalars (such as depths, surface tracers, etc.) while subroutine *rot_intrp_vctr* can be used to interpolate and correct the angles for vectors (such as wind stress).

17.3 Considerations

Although additional execution time used by this option is negligible, the option may complicate subsequent analysis by the researcher. Rotated model results can be interpolated back to the geographic grid for comparison, but this may involve recalculating many diagnostics (such as zonal integrals) since all diagnostics within MOM are only computed on the rotated model grid.

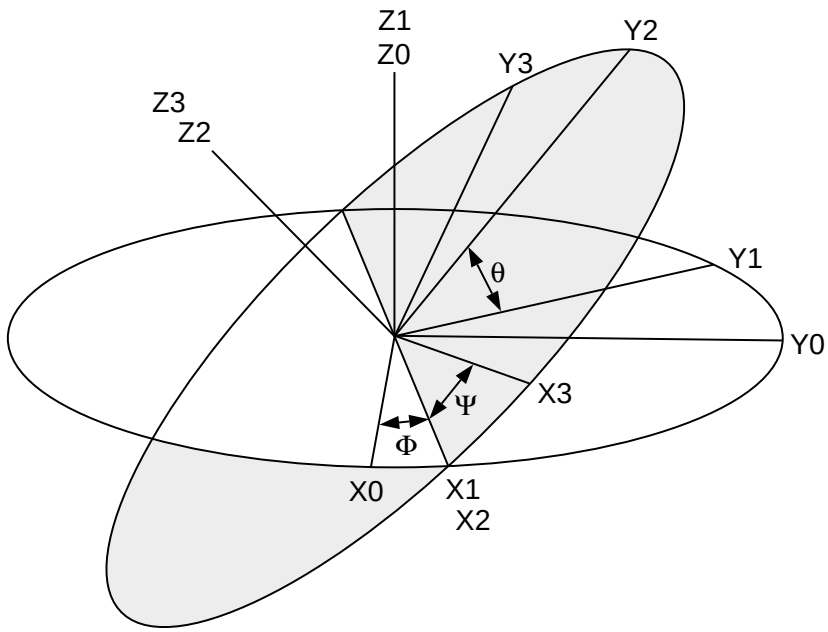


Figure 17.1: Euler solid body rotation angles and their associated axes for rotating the MOM latitude and longitude grid.

Chapter 18

Topography and geometry

As described in Chapter 16, the model domain (as defined in module *grids*) is populated with a horizontally rectangular arrangement of T-cells and U-cells arranged in an Arakawa B-grid fashion and stacked vertically on top of each other. The vertical stacking is such that T-cells overlay T-cells and U-cells overlay U-cells. In general, some of these grid cells will be treated as land cells and others as ocean cells. Geometry and topography determine whether individual cells are treated as land or ocean. This three dimensional land/ocean cell arrangement is encapsulated in a two dimensional array of integers $kmt_{i,jrow}$ (an example is given in Figure 18.1) which indicate the number of ocean cells stacked vertically between the surface and ocean bottom for each longitude and latitude coordinate xt_i and yt_{jrow} on the T-grid. For instance, the following settings

$$kmt_{i,jrow} = 0 \tag{18.1}$$

$$kmt_{i+1,jrow} = 6 \tag{18.2}$$

specify that there are no ocean cells[1] at coordinate location with index (i, $jrow$) but the location immediately to the east has 6 ocean T-cells[2]. On the U grid, there is a corresponding field of integers $kmu_{i,jrow}$ derived as the minimum of the four surrounding $kmt_{i,jrow}$ values.

$$kmu_{i,jrow} = min(kmt_{i,jrow}, kmt_{i+1,jrow}, kmt_{i,jrow+1}, kmt_{i+1,jrow+1}) \tag{18.3}$$

Vertical levels are indexed from the uppermost at $k = 1$ to the bottom of the domain at $k = km$. Beneath the ocean surface, land cells exist where $k > kmt_{i,jrow}$ on the T grid and where $k > kmu_{i,jrow}$on the U grid. The depth from the ocean surface to the bottom of the ocean is defined at the longitude and latitude of U-cells by

$$H_{i,jrow} = zw_{kbu} \quad where \;\; kbu = kmu_{i,jrow} \tag{18.4}$$

where zw_k is the depth from the ocean surface to the bottom face of the kth vertical level.

18.1 Designing topography and geometry

Although module *topog* is executed as part of MOM, designing $kmt_{i,jrow}$ by executing MOM is similar to using a sledge hammer to work with tacks. The preferred method is to do the

[1] All T-cells in the vertical at this location are treated as land cells.

[2] The cells below level 6 are treated as land cells.

construction by executing script *run_topog* from a MOM_UPDATES directory as described for module *grids* in Section 16.3. Put script *run_topog* in the MOM_UPDATES directory and it will execute module *topog* in a stand alone mode by enabling option *drive_topog*. The execution will produce a file *results_topog* which should be inspected. Once topography and geometry are judged as satisfactory, module *topog* along with the resulting topography and geometry are available to MOM during model executions. This is done by not including the option *drive_topog* in the list of options within the *run_mom* script.

18.2 Options for constructing the KMT field

Various ocean depth construction methods may be selected for use within MOM by enabeling options described below. Before being used, realistically derived depth fields must first be interpolated to the latitude and longitude coordinates for the domain defined by module *grids*. After the interpolation, ocean depth within the model is approximated as the nearest number of discrete vertical levels that comes closest to matching the interpolated ocean depth field. If option *partial_cell* is enabled, the deepest ocean cells have variable depth so the model ocean depth matches the interpolated ocean depth exactly. Refer to Chapter 26 for a discussion of partial cells. The resulting field of model levels is the base $kmt_{i,jrow}$ field defined at all latitude and longitude coordinates given by (xt_i, yt_{jrow}). One (and only one) of the following options for selecting the various forms of ocean depth fields must be enabled:

- *rectangular_box* constructs a flat bottomed rectangular box with $\mathrm{kmt}_{i,jrow} = km$ (deepest level) for all interior points on the grid ($i = 2, imt - 1$ and $jrow = 2, jmt - 1$) while setting $\mathrm{kmt}_{i,jrow} = 0$ on all boundary cells. Enabling option *cyclic* turns the box into a zonally re-entrant channel by re-setting

$$kmt_{1,jrow} = kmt_{imt-1,jrow} \tag{18.5}$$
$$kmt_{imt,jrow} = kmt_{2,jrow} \tag{18.6}$$

  for $jrow = 1, jmt$.

- *idealized_kmt* constructs an idealized version of the earth's geometry. Continental features are very coarse[3] but map onto whatever grid resolution is specified by module *grids*. They are built with the aid of subroutine *setkmt* which approximates continental shapes by filling in trapezoidal areas of $kmt_{i,jrow}$ with *zero*. Subroutine *setkmt* readily allows idealized geometries to be constructed. All that is required for input are the grid point coordinates, the coordinates of four vertices of a trapezoid, and a value for setting $kmt_{i,jrow}$ within the trapezoid. The bottom topography has a sinusoidal variation which is totally unrealistic. It is intended only to provide a test for the numerics in MOM. Since this option generates geometry and topography internally, no external data is required and therefore the DATABASE (explained in Chapter 3) is not needed. Typically, this option is useful when researching idealized geometries and topographies since arbitrary ones can be easily constructed. Also, this is useful when porting MOM to other computer platforms since no data files need be considered.

[3]Their scale is about 10 degrees in latitude and longitude.

- *gaussian_kmt* constructs a $kmt_{i,jrow}$ field derived from an analytical gaussian bump superimposed on a sloping ocean floor in a rectangular basin without geometry. The intent of this option is to provide a basis for regional idealized studies. The slope of the floor and scale of the bump can be set by modifying the USER INPUT section under the gaussian bump section in module *topog.F*. Further code modifications can easily turn the bump into a bowl or depression in the ocean floor if desired. This option requires no data and is therefore also highly portable. Results are interpolated to the resolution specified by module *grids*. The ocean bottom depth $ht_{i,jrow}$ is specified as

$$\begin{aligned} ht_{i,jrow} &= zw_{km} - b_0 \cdot exp^{-((xt_i - xu_{imt/2})^2 + (yt_{jrow} - yu_{jmt/2})^2)/b_1^2)} \\ &- slope_x \cdot (xt_i - xu_1) - slope_y \cdot (yt_{jrow} - yu_1) \end{aligned} \tag{18.7}$$

where

$$b_0 = zw_{km}/2 \tag{18.8}$$
$$b_1 = 0.25 \cdot min(xt_{imt-1} - xt_2,\ \ yt_{jmt-1} - yt_2) \tag{18.9}$$
$$slope_x = 10\ meter/deg \tag{18.10}$$
$$slope_y = 10\ meter/deg \tag{18.11}$$

and $ht_{i,jrow}$ is discretized to the nearest model level to construct the $kmt_{i,jrow}$ field.

- *scripps_kmt* reads the *scripps.top* 1° x 1° topography file (from the MOM DATABASE explained in Chapter 3) and interpolates to $kmt_{i,jrow}$ for the grid defined by module *grids*. Subroutine *scripp* within module *topog* makes an educated guess as to whether resolution specified by module *grids* is coarser or finer than the native Scripps resolution. If finer, it does a linear interpolation from Scripps data, otherwise it uses area averaging of Scripps data to estimate the value of $kmt_{i,jrow}$ for the resolution required by module *grids*.

- *etopo_kmt* reads the file *ETOPO5.NGDCunformat_ieee* which is a 1/12° x 1/12° topography dataset. The dataset may be purchased from the Marine Geology and Geophysics Division of the National Geophysical Data Center and is not included in the MOM DATABASE. Depths are interpolated to a $kmt_{i,jrow}$ field for the grid defined by module *grids*. Subroutine *etopo* within module *topog* does the interpolation. If model resolution is finer than 1/12°, a linear interpolation from the dataset is used, otherwise area averaging over the dataset is used to estimate the value of $kmt_{i,jrow}$ for the resolution required by module *grids*.

- *read_my_kmt* allows importing $kmt_{i,jrow}$ fields into MOM which have been exported from module *topog* using option *write_my_kmt*. One purpose of importing is to provide a hook for reading $kmt_{i,jrow}$ fields which have been constructed outside the MOM environment. Another is to allow direct editing of the $kmt_{i,jrow}$ field using a text editor as explained in Section 18.4.2. Still another use is when working with gigantic datasets like the 1/12° ETOPO5. Bringing these gigantic datasets into a model run simply to compute a $kmt_{i,jrow}$ field can adversely affect model turn around time. Expecially when trying a new model setup which requires multiple attempts to get problems resolved. It is better to export the resulting $kmt_{i,jrow}$ field from a stand alone execution of module *topog*. Then just import $kmt_{i,jrow}$ into a model execution instead of the huge dataset.

Note that the $kmt_{i,jrow}$ field only needs to be constructed or imported into the model once at initial condition time. After that, it is incorporated as part of the restart file *restart.dta*. Regardless of how $kmt_{i,jrow}$ is generated, module *topog* produces a checksum which can be used to verify that the $kmt_{i,jrow}$ field produced by script *run_topog* is the one being used by MOM.

18.3 Meta land masses

Figure 18.1 is an example of a base $kmt_{i,jrow}$ field. Note the cell pointed to by the "perimeter violation" arrow. When using option *stream_function*, it is incorrect[4] to have two distinct land masses seperated by a minimum of only one ocean T-cell. The reason is that if these two land masses are to remain unconnected, then the value of the stream function at the cell in question can be multivalued: one stream function value is implied by the integral of the pressure gradients around one land mass and the other is implied by the integral around the other land mass. But, since all four surrounding U-cells of the offending "perimeter violation" cell are land cells, no flow can exist between the land masses. So the integrals are not independent after all. The way to think of this is that the two seperate land masses are components of a "meta " land mass where all components are treated as a single mass. This is achieved by modifying the path of the island perimeters of the land masses so that the "meta" land mass has only one perimeter. Therefore "perimeter violation" cells are no longer a problem. They are automatically accounted for by constructing "meta" land masses. If desired, the "perimeter violation" cells can always be changed to land cells.

The difference between filling the "perimeter violation" cell in with land and leaving it as part of a "meta" land mass is that if the cell is made into land, then tracers cannot be diffused across it. However, as part of a "meta" land mass, two lateral faces of the "perimeter violation" cell connect with other ocean cells and diffusion of tracer quantities can occur. Therefore, if "perimeter violation" cells are left as an ocean cells, two ocean basins which cannot communicate advectively may still be able to communicate diffusively.

18.4 Modifications to KMT

The base $kmt_{i,jrow}$ field constructed above may have problem cells as illustrated in Figure 18.1. These include minimum depth violations and potentially troublesome areas such as isolated cells. Or possibly, the researcher may wish to modify the $kmt_{i,jrow}$ field by changing ocean cells to land cells or visa versa for various reasons.

18.4.1 Altering the code

Changes can be made by hard wiring $kmt_{i,jrow}$ to specific values in the USER INPUT section of module *topog*. For example

$$kmt_{37,82} = 0 \tag{18.12}$$

sets all T-cells at location given by indices $(i, jrow) = (37, 82)$ from $k = 1 \;\; to \;\; km$ to land cells. Setting large areas of $kmt_{i,jrow}$ to land or ocean cells may be handled using routine *setkmt* as

[4]This is not a restriction for the rigid lid surface pressure option, the implicit free surface option, or the explicit free surface option because they do not perform island integrals.

is done when generating the idealized dataset. Note that idealized geometries are readily constructed by blocking out areas of $kmt_{i,jrow}$ with the aid of subroutine *setkmt*. Refer to option *idealized_kmt* in Section 18.2 for usage. The two problems mentioned above are remedied automatically using guidance given in the form of options. Those options are:

- If option *fill_isolated_cells* is enabled, isolated T-cells are made into land cells. Isolated T-cells are deeper than nearest neighbor T-cells. Think of them as potholes or one cell wide trenches in the topography. They cannot communicate with horizontal neighbors through advection because all surrounding velocities are zero.

  A sub-option of *fill_isolated_cells* is *fill_isolated_dont_change_landmask* which restores any $kmt_{i,jrow}$ which was set to zero in the process of filling cells. Using this sub-option allows the land/sea areas to be preserved when filling isolated cells.

- Limit the minimum number of vertical levels in the ocean to parameter *kmt_min* = 2. This parameter may be increased but not decreased. There is some choice for altering $kmt_{i,jrow}$ to meet this condition. The choice is made by enabling one of the following options:

  - *fill_shallow* makes land cells where there are fewer than *kmt_min* vertical levels
  - *deepen_shallow* sets $kmt_{i,jrow} = kmt_min$ where there are fewer than *kmt_min* vertical levels
  - *round_shallow* sets $kmt_{i,jrow}$ to either 0 or *kmt_min* depending on which is closest

18.4.2 Directly editing the KMT field

Another way to change $kmt_{i,jrow}$ is to export the field using option *write_my_kmt* which saves $kmt_{i,jrow}$ to file kmt.dta under formatted control. Formatting is determined by the model's deepest level. File kmt.dta can then be edited with any text editor. Care must be exercised when editing so as not to change the format. If a character is deleted, a space or another character must be inserted in its place. When finished, the $kmt_{i,jrow}$ field can then be imported with option *read_my_kmt*.

18.5 Topographic instability

When using option *stream_function*, it sometimes happens that steep topographic gradients as described in Killworth (1987) are a source of problems. Essentially, a restriction on the external mode time step $dtsf$ is given by the condition

$$dtsf < \Delta\tau \tag{18.13}$$

where

$$\Delta\tau = \frac{(dxt_i \cos\phi^T_{jrow})^2}{A_m} \cdot \frac{(8\, h_1\, h_2)/(h_1+h_2)^2 + (dxt_i \cos\phi^T_{jrow})^2/(dyt_{jrow})^2}{4 + (dxt_i \cos\phi^T_{jrow})^2/(dyt_{jrow})^2} \tag{18.14}$$

$$h_1 = zw_{k_1} \tag{18.15}$$

$$h_2 = zw_{k_2} \tag{18.16}$$

$$k_1 = kmu_{i+1,jrow-1} \tag{18.17}$$

$$k_2 = kmu_{i+1,jrow} \tag{18.18}$$

The above condition on time step $\Delta\tau$ must be met at each point with index (i, *jrow*) within the domain. Since analysis requires knowledge of time step length and mixing coefficients, the above time step restriction is not detected in module *topog*. Rather, detection is left to MOM where all the information is known. Notably, the above condition is most restrictive in polar regions where convergence of meridians makes the effective zonal grid length $dxt_i \cos \phi^T_{jrow}$ go toward zero. The time step restriction can be ameliorated by making the mixing coefficient a function of latitude to compensate for the convergence of meridians. For using variable mixing coefficients, refer to Sections 34.6 and 34.6.2.

Locally smoothing the topography can also help by lessening topographic gradients in the meridional direction and thereby increasing the factor $(8\ h_1\ h_2)/(h_1 + h_2)^2$. This can be accomplished with option *smooth_topo*. When enabled, the smoothing is applied northward of a specified latitude by successively applying a 2D[5] finite impulse response symmetric filter where 1D weights are given by (1/4, 1/2, 1/4). Additional conditions are applied to insure that the original coastline is not changed[6] and topographic depths can only by decreased[7] by applying this filter. When option *smooth_topo* is enabled, latitudes northward of 85N are filtered by 10 passes of this filter. To change the smoothing latitude or the number of passes (both are arbitrary), look at the USER INPUT section of *topog.F* under the options *scripps_kmt* and *etopo_kmt*.

In the test case resolution with Scripps topography, options *varhmix* and *am_cosine* along with the above mentioned topographic filter will eliminate the Killworth time step restriction. Another alternative is to change the external mode solution technique from option *stream_function* to option *implicit_free_surface*.

18.6 Viewing results

The recommended way of viewing the resulting topography, $kmt_{i,jrow}$ field, and f/H field is to save these results using option *topog_diagnostic*. This is described in Section 40.20. A good way to visualize results is with Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (email: ferret@pmel.noaa.gov URL: http://www.pmel.noaa.gov/ferret/home.html).

As an alternative, a map of the resulting $kmt_{i,jrow}$ field is printed out as as part of the *results_topog* file from executing module *topog*. It is also possible to export the $kmt_{i,jrow}$ field and use an editor to view file kmt.dta.

18.7 Summary of options for topography

The following is a list of options used within the topography module. They are in no particular order.

- *topog_diagnostic*. Saves out topography and related fields in NetCDF or IEEE format for analysis.
- *drive_topog*. To execute module *topog* in a stand alone mode rather from within a model execution.

[5] Although the application of a 1D filter in the meridional direction will remove the topographic slopes, the 1D smoothing can introduce bizzare effects due to shallow water near coastlines. The 2D filter works best.

[6] The coastline is made invisible to the filter.

[7] Allowing the filter to incrase the depths requires realistic initial density profiles to be known below the realistic bottom. Therefore, the filter is restriced so that it can only shallow the bottom.

- *scripps_kmt*. Builds a $kmt_{i,jrow}$ field from Scripps 1 degree topography data.
- *etopo_kmt*. Builds a $kmt_{i,jrow}$ field from Etopo 1/12 degree topography data.
- *smooth_topo*. Smooths the interpolated topography from Etopo 1/12 degree or Scripps 1 degree datasets to help eliminate the Killworth Instability (Section 18.5). Such smoothing is also useful for removing grid scale power in the topography field. Note, however, that option *smooth_topo* will not allow for grid points to be deepened. To allow deepening, one must also turn on option *smooth_topo_allow_deepening*.
- *smooth_topo_allow_deepening*. When enabled with *smooth_topo*, this option will smooth the topography and will allow for this smoothing to deepen grid cells. The topography from both *smooth_topo* and *smooth_topo_allow_deepening* is somewhat smoother than the topography using only *smooth_topo*. It should be noted that if using Levitus initial conditions, the values used in the deepened grid points will need to be computed through an extrapolation procedure described in Section 28.1.4. Errors can arise from this extrapolation and so one should be aware of this issue when using the deepening option.
- *smooth_topog_after_user_changes*. This option is useful for those cases in which the researcher changes the land mask by adding or removing land points, but still wants to maintain a smooth topography to avoid numerical problems.
- *rectangular_box*. Builds a $kmt_{i,jrow}$ field for a flat bottomed rectangular box.
- *idealized_kmt*. Builds a $kmt_{i,jrow}$ field based on a highly idealized (not very accurate) geometry. Topography is un-realistic. It is useful for porting MOM to various architectures without having to worry about datasets. It can be easily modified for other idealized studies.
- *gaussian_kmt*. Builds the $kmt_{i,jrow}$ based on a gaussian bump on a sloping ocean floor in a rectangular domain. It is highly portable and can be easily modified for other idealized studies.
- *flat_bottom*. Resets the bottom within the ocean to the deepest level regardless of which method was used to construct the $kmt_{i,jrow}$ field.
- *partial_cell*. Allows bottom cells to be of variable thickness within the same vertical level.
- *bbl_ag*. Allows for a bottom boundary layer.
- *rough_mixing*. Calculates the bottom roughness by examining the curvature of the Etopo 1/12 degree dataset. If option *etopo_kmt* is not used, then the roughness is set to zero.
- *read_unformatted_kmt*. Imports $kmt_{i,jrow}$ from older kmt.dta files saved without format control.
- *read_my_kmt*. Imports $kmt_{i,jrow}$ from file kmt.dta under format control.
- *write_my_kmt*. Exports $kmt_{i,jrow}$ fo file kmt.dta under format control.
- *read_my_roughness*. Imports a roughness field from file rough.dta (unformatted).
- *write_my_roughness*. Exports a roughness field from file rough.dta (unformatted).

- *solid_walls*. Land boundaries along the northern, southern, eastern, and western limits of the domain.
- *cyclic*. Land boundaries along the northern and southern limits of the domain. However, what flows out of the eastern boundary enters through the western boundary and visa versa.
- *fill_shallow*. Fills ocean cells with land where the bottom is less than the specified minimum number of vertical levels.
- *round_shallow*. Where the bottom is less than the specified minimum number of vertical levels, the bottom is rounded to either zero depth or the minimum number of vertical levels.
- *deepen_shallow*. Where the bottom is less than the specified minimum number of vertical levels, the bottom is deepened to the minimum number of vertical levels.
- *fill_isolated_cells*. Fills one cell wide potholes and trenches with land.
- *fill_isolated_dont_change_landmask* does not allow the land/sea mask to change when filling isolated cells.
- *skip_island_map*. To eliminate land mass map indicating island perimeters.
- *skip_perimeter_details*. To eliminate listing the land mass perimeter points.
- *skip_kmt_map*. To eliminate printing the map of kmt.
- *skip_translation_details*. Interpolating data near the prime meridian is a problem for the interpolation routines because the longitude is not monotonic. Basically, it is difficult for the interpolation routines to decide in general if an error was made in specifying longitudes or not. So the topography data is translated in longitude to move the prime meridian so that longitudes defining real data are monotonic within the model domain. Normally, the original and translated longitudes are printed out. Use this option to eliminate the printing.

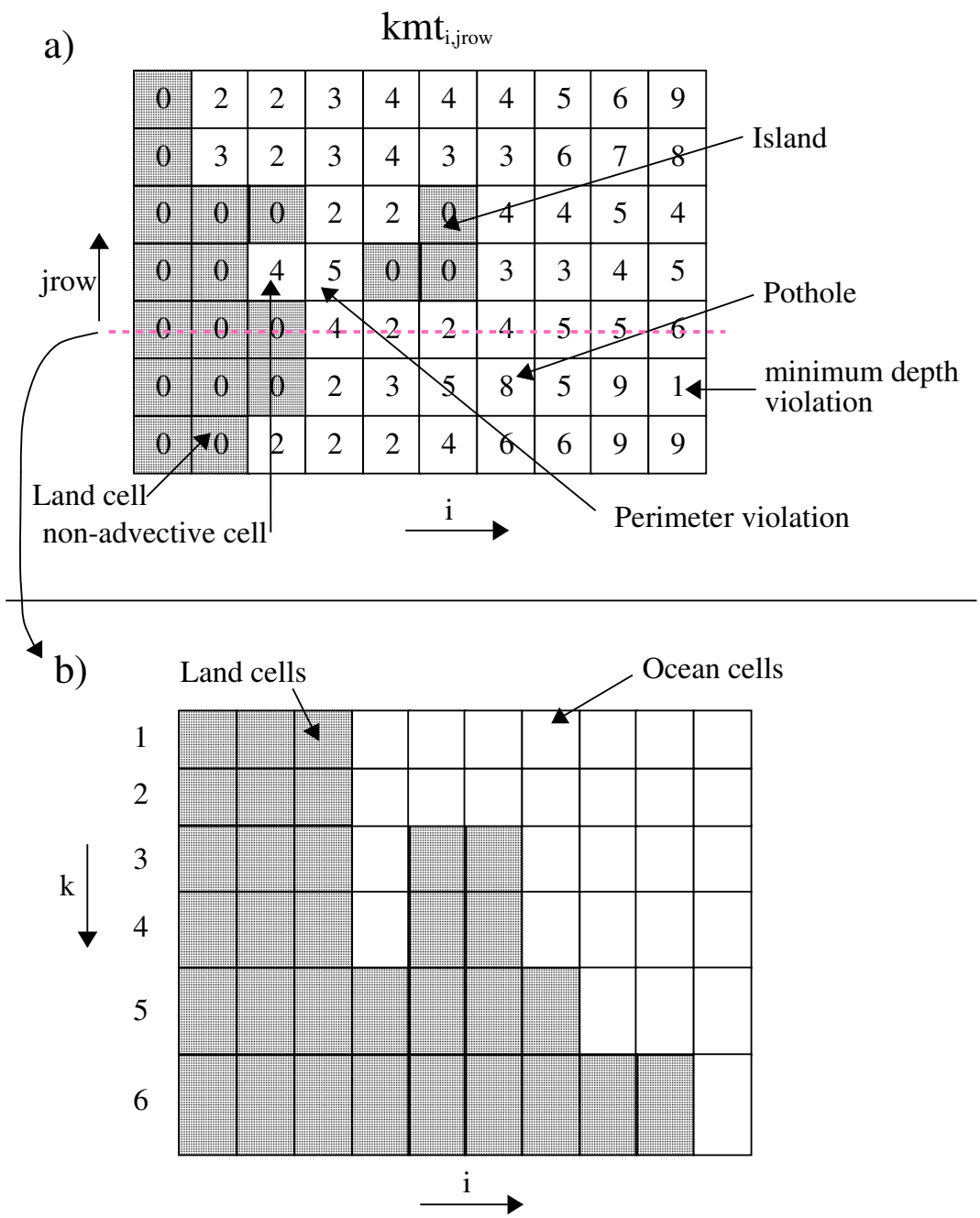


Figure 18.1: a) A portion of the *kmt* field indicating ocean and land areas with potential problems. b) Topography along the slice indicated in a)

Part V

Boundary Conditions

Chapter 19

Generalized Surface Boundary Condition Interface

The ocean is driven by surface boundary conditions[1] which come from the atmosphere and the atmosphere is in turn driven by surface boundary conditions[2] which come from the ocean. The difficulty in understanding the coupled system is that each component influences the other. For purposes of MOM, the atmosphere may be thought of as a hierarchy of models ranging from a simple idealized wind dataset fixed in time through a complicated atmospheric GCM[3]. Regardless of which is used, MOM is structured to accommodate both extremes as well as atmospheres of intermediate complexity as discussed in the following sections.

19.1 Coupling to atmospheric models

In this section, the general case will be considered where the atmosphere is assumed to be a GCM with domain and resolution differing from that of MOM and two-way coupling between MOM and the atmosphere model will be allowed. Option *coupled* configures MOM for this general case and the test case prototype is described in Sections 3.2 and 3.3 as CASE=3.

A flowchart of *driver*[4] is given in Figure 19.2. It begins by calling subroutine *setocn*[5] to perform initializations for *mom*[6]. Included in the list are such things as initializing variables, reading namelists to over-ride defaults, setting up the grid, topography, initial conditions, region masks, etc. In short, everything that needs to be done only once per model execution. Following this, a call is made to subroutine *setatm*[7] which completes whatever setup is required by *atmos*[8].

At this point, the integration is ready to begin. Integration time is divided into a number of equal length time segments[9] which determine the coupling period. In practice, this interval

[1]Wind, rain, heatflux, etc.
[2]Primarily SST.
[3]General circulation model.
[4]Contained within file driver.F. This is the main program for MOM.
[5]Contained within file setocn.F.
[6]Contained within file mom.F. This is the subroutine that does the time integration for the ocean model.
[7]Contained within file setatm.F in the MOM_2/SBC directories.
[8]Contained within file atmos.F in the SBC directories. This is the subroutine that does the time integration for the atmosphere model.
[9]The length of one time segment should be divisible by the length on one ocean time step to allow an integral number of calls to subroutine *mom*. The same holds true for the atmosphere time step.

should always be chosen short enough to adequately resolve time scales of coupled interaction. Typically this value would be one day[10]. Within the segment loop, *atmos* and *mom* are alternately integrated for each time segment while holding surface boundary conditions fixed. Note that this may require multiple calls to each model. In Figure 19.2, the loop variable *ntspas* stands for the number of time steps per atmosphere segment and the loop variable *ntspos* stands for the number of time steps per ocean segment. Products of each atmosphere segment include surface boundary conditions[11] for the ocean averaged over that segment. These are held fixed and applied to *mom* while it integrates over the same segment. Products of integrating *mom* include surface boundary conditions for the atmosphere which are also averaged over this segment. Subsequently, they are held fixed and applied to *atmos* on the following segment. This process is represented schematically as a function of time in Figure 19.1 and continues until all time segments are completed. Using asynchronous time segments[12] is possible with a small code modification but this is left to the researcher.

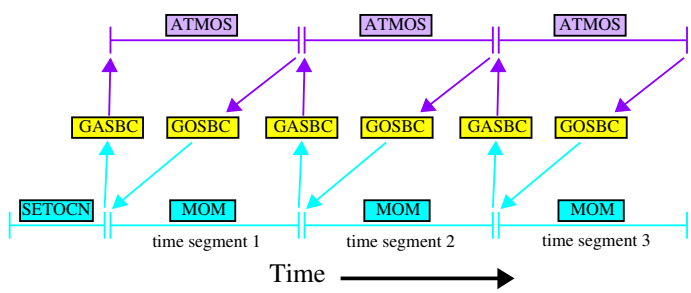


Figure 19.1: Schematic of two way coupling between an atmosphere model (ATMOS) and an ocean model (MOM) showing time segments.

In MOM and in the descriptions that follow, index j refers to any variable dimensioned by the number of rows in the memory window and index *jrow* refers to any variable dimensioned by the total number of latitude rows. They are related by an offset $jrow = j + joff$ which indicates how far the memory window has moved northward. Refer to Section 14.2 and also 11.3.2 for a more complete description.

19.1.1 GASBC

Getting surface boundary conditions for the atmosphere model defined on the atmosphere grid is the purpose of subroutine *gasbc*[13] which is an acronym for *get atmosphere surface boundary conditions*. All of the surface boundary conditions are two dimensional fields defined in longitude and latitude at model grid points. In the ocean, all quantities which are to be used as surface boundary conditions for the atmosphere are defined on the $T_{i,j}$ grid[14] in array

[10]If the diurnal cycle is included, the coupling period needs to be reduced to allow adequately resolution in time.
[11]The first set of surface boundary conditions for the atmosphere are products of *setocn*.
[12]Where, for example, an ocean segment is much longer than an atmosphere segment. This assumes the coupled system is linear with one equilibrium. Exercise caution if contemplating this!
[13]Contained in file gasbc.F.
[14]At grid locations given by xt_i and yt_{jrow}.

$sbcocn_{i,jrow,m}$ where subscript m refers to the ordering described in Section 19.3. Typically, only SST is needed but it may be desirable to let the atmosphere sense that the ocean surface is moving in which case the horizontal velocity components might also be used (Pacanowski 1987). In any event, these quantities are accumulated in $sbcocn_{i,jrow,m}$ within MOM and averaged at the end of each time segment. Basically what needs to be done in *gasbc* is to interpolate the time averaged $sbcocn_{i,jrow,m}$ fields to $sbcatm_{i',j',m}$ which is an array of the same surface boundary conditions except defined on the atmosphere boundary condition grid[15] $A_{i',j'}$. One of the duties of *setatm* is to define $A_{i',j'}$ as the atmospheric boundary condition grid which includes extra boundary points along the borders to facilitate these interpolations.

19.1.1.1 SST outside Ocean domain

There is a complication if the ocean is of limited extent: SST must be prescribed outside the ocean domain as a surface boundary condition for the atmosphere on the $A_{i',j'}$ grid. To accommodate this, the ocean domain $T_{i,j}$ must be known in terms of $A_{i',j'}$ and a buffer or blending zone must be established. Within this zone, SST from *mom* is blended with the prescribed SST outside the ocean domain as indicated in Figure 19.3. As a simplification, SST is prescribed outside the domain of *mom* as a constant which is obviously unrealistic and should be prescribed by the researcher as a function of space and time appropriately along with the width of the blending zone. SST within the blending zone is a linear interpolation between the two regions.

19.1.1.2 Interpolations to atmos grid

Each surface boundary condition is interpolated one at a time using essentially the following steps:

- For limited ocean domains, prescribe SST in $sbcatm_{i',j'}$ outside the *mom* domain as described above. When ocean and atmosphere domains match, this is not necessary.

- Extrapolate $sbcocn_{i,jrow,m}$ into land areas on the *mom* grid. This is done by solving $\nabla^2(SST_{i,jrow}) = 0$ over land cells using ocean $SST_{i,jrow}$as boundary conditions[16]. The idea is to get reasonable $SST_{i,jrow}$in land adjacent to coastlines but not necessarily to produce accurate $SST_{i,jrow}$in the middle of continents. Where land and ocean areas on the ocean and atmosphere grids are mismatched, this extrapolation will ameliorate the sensing of erroneous $SST_{i,jrow}$by the atmosphere. Such situations are common when coupling spectral atmospheres to MOM which has an Arakawa B-grid.

- Interpolate $sbcocn_{i,jrow,m}$ to $sbcatm_{i',j',m}$. The ocean resolution is typically higher than that of the atmosphere because the Rossby radius is larger in the atmosphere. To prevent aliasing, the interpolation is an area average of $sbcocn_{i,jrow,m}$ for those i, $jrow$ cells which fall within each $A_{i',j'}$ grid cell[17]. The interpolation is carried out using subroutine *ftc*[18]

[15] Grid locations given by $abcgx_{i'}$ and $abcgy_{j'}$.

[16] This method is arbitrary. However, when solving this equation iteratively, it is important not to zero out SST from previous solutions over land areas. Their purpose is to act as a good initial guess to limit the iterations needed for subsequent solutions.

[17] Partial ocean cells are accounted for to conserve the interpolated value.

[18] Acronym for fine to coarse resolution. It is an interpolation utility in module *util*.

which can easily be replaced by subroutine *ctf*[19] (or something else) if ocean resolution is less than atmosphere resolution.

- Set cyclic conditions on $sbcatm_{i',j',m}$ and convert to units expected by the atmosphere.
- Compute global mean of $sbcatm_{i',j',m}$

The structure of *gasbc* is arranged such that components can be removed or replaced with more appropriate ones if desired.

Caveat

In the process of constructing $sbcatm_{i',j',m}$ no attempt has been made at removing small scale spatial features from the grid which could potentially be a source of noise for the atmosphere model.

19.1.2 GOSBC

Getting surface boundary conditions for the ocean model defined on the grid of *mom* is the purpose of subroutine *gosbc*[20] which is an acronym for *get ocean surface boundary conditions* . This is the counterpart of *gasbc*. All of the surface boundary conditions are two dimensional fields as described in Section 19.1.1. In the atmosphere, all quantities which are to be used as surface boundary conditions for MOM are defined on the $A_{i',j'}$ grid in array $sbcatm_{i',j',m}$ where subscript m again refers to the ordering described in Section 19.3. Typically, they are quantities like windstress components, heatflux and precipitation minus evaporation. These must be accumulated in $sbcatm_{i',j',m}$ within the atmosphere model and averaged at the end of each time segment. Basically what needs to be done in *gosbc* is to interpolate the time averaged $sbcatm_{i',j',m}$ to $sbcocn_{i,jrow,m}$ which is defined on the ocean grid.

19.1.2.1 Interpolations to ocean grid

Unlike in Section 19.1.1, there are no complications since the ocean domain is assumed to be contained within the atmosphere domain as shown in Figure 19.4. Each surface boundary condition is interpolated one at a time with the essential steps being:

- Set cyclic conditions of $sbcatm_{i',j',m}$. This assumes a global atmosphere domain.
- Extrapolate $sbcatm_{i',j',m}$ into land areas on the atmosphere grid. This is done as in Section 19.1.1 by solving $\nabla^2\xi_{i',j'} = 0$ over land areas using values of $\xi_{i',j'}$ over non-land areas as boundaries where $\xi_{i',j'}$ represents windstress components, heatflux, etc[21]. The idea, as on the ocean grid, is to get reasonable $\xi_{i',j'}$ adjacent to coastlines while not trying to produce accurate $\xi_{i',j'}$ in the middle of continents. Where land and ocean areas on the ocean and atmosphere grids are mismatched, this will ameliorate the sensing of erroneous $\xi_{i',j'}$ by the ocean.

[19] Acronym for coarse to fine resolution. It is an interpolation utility in module *util*.

[20] Contained within file gosbc.F.

[21] This method is arbitrary. However, when solving this equation iteratively, it is important not to zero out ξ from previous solutions over land areas. Their purpose is to act as a good initial guess to limit the iterations needed for subsequent solutions.

- Interpolate $sbcatm_{i',j',m}$ to $sbcocn_{i,jrow,m}$. Since ocean resolution is typically higher than that of the atmosphere due to the difference in Rossby radius scales, the interpolation is linear. The interpolation is carried out using subroutine *ctf* from file *util.F*. Note that linear interpolation does not exactly conserve the quantity being interpolated. However, linear interpolation has been used without causing noticable drift in coupled integrations (without flux correction) of up to 20 years at GFDL.

  For integrations simulating thousands of years, linear interpolation may not be good enough. To conserve fluxes exactly, one possibility (although drastic) is to design the grids such that an integral number of ocean cells overlay each atmospheric grid cell. Then the quantity to be interpolated can simply be broadcast from each atmospheric cell to all underlying ocean cells. However, this is really not necessary. Assume that the value of the quantity being interpolated is constant over the entire atmospheric grid cell. For exact conservation, the value in each ocean cell is just an integral of the atmospheric quantity over the area of the ocean cell. It would be a matter of writing a subroutine to do it and substituting this subroutine for the call to *ctf*.

- Set cyclic conditions on $sbcocn_{i,jrow,m}$ and convert to units expected by the ocean. MOM expects units of *cgs*.

- Compute global mean of $sbcocn_{i,jrow,m}$

<u>Caveat</u>

In the process of constructing $sbcocn_{i,jrow,m}$ no attempt has been made at removing small scale spatial features from the grid which could potentially be a source of noise for the ocean model.

19.2 Coupling to datasets

In Section 19.1, the general case of two way coupling to an atmospheric GCM was considered. It is also useful to drive *mom* with atmospheric datasets representing simpler idealized atmospheres. One problem with doing this is that datasets act as infinite reservoirs of heat capable of masking shortcomings in parameterizations which only become apparent when two way coupling is allowed. Nevertheless, driving ocean models with surface boundary conditions derived from datasets is useful.

These datasets can be thought of as simple atmospheres prepared *a priori* such that data is defined at grid locations expected by *mom* for surface boundary conditions. All spatial interpolations are done beforehand, as described in Section 3.2. In this case, subroutines *gasbc* and *goabc* are bypassed because spatial interpolation to the *mom* grid is not needed. Also, the length of a time segment reduces to the length of one ocean time step. In general, the datasets contain either time mean conditions or time varying conditions and there are three options which configure these types of surface boundary conditions for *mom*:

- *simple_sbc* allows for the simplest of atmospheric datasets. All surface boundary conditions are a function only of latitude. No time dependence here although it could easily be incorporated. The test case prototype is described in Sections 3.2 and 3.3 as CASE=0. In this case, there is no dataset on disk and $sbcocn_{i,jrow,m}$ is not needed since all surface boundary conditions are generated internally. This is an appropriate option for building forcing functions for idealized surface boundary conditions .

- *time_mean_sbc_data* uses an atmosphere dataset defined in SBC/TIME_MEAN. The test case prototype is described in Sections 3.2 and 3.3 as CASE=1. The dataset resides on disk and each surface boundary condition is a function of latitude and longitude. The dataset should be prepared for the grid in MOM using scripts in PREP_DATA as described in Section 3.2. The surface boundary condition data is read into the surface boundary condition array $sbcocn_{i,jrow,m}$ once in *atmos* for all latitude rows where m gives the ordering of the boundary conditions as described in Section 19.3. On every time step, data from $sbcocn_{i,jrow,m}$ is loaded into the surface tracer flux array $stf_{i,j,n}$ and the surface momentum flux array $smf_{i,j,n}$. This is done in subroutine *setvbc* for rows $j = 2, jmw - 1$ in the memory window[22].

- *time_varying_sbc_data* uses an atmosphere dataset defined in SBC/MONTHLY. The test case prototype is described in Sections 3.2 and 3.3 as CASE=2. The dataset resides on disk as a series of records. Each record is a climatological monthly mean surface boundary condition as a function of latitude and longitude. This dataset should be prepared for the grid in *mom* using scripts in PREP_DATA as described in Section 3.2. The complication is one of interpolating the monthly values to the model time in *mom* for each time step. Basically, the model time must be mapped into the dataset to find which two months (data records) straddle the current model time[23]. For the purpose of interpolation, months are defined at the center of their averaging periods. Both months are read into additional boundary condition arrays in *atmos* and used to interpolate to array $sbcocn_{i,jrow,m}$ for the current model time. When the model time crosses the midpoint of a month, the next month is read into a boundary condition array[24] and the process continues indefinitely assuming the dataset is specified as periodic. If it is not, *mom* will stop when the ocean integration time reaches the end of the dataset. Although data is read into the additional boundary condition arrays only when ocean model time crosses the mid month boundary, data is interpolated into $sbcocn_{i,jrow,m}$ on every timestep. There are four allowable methods for interpolating these datasets in time:

  Method=0 is for no interpolation; the average value is used for all times within the averaging period. This is the simplest interpolation. It preserves the integral over averaging periods, but is discontinuous at period boundaries.

  Method=1 is for linear interpolation between the middles of two adjacent averaging periods. It is continuous but does not preserve integrals for unequal averaging periods.

  Method=2 is for equal linear interpolation. It assumes that the value on the boundary between two adjacent averaging periods is the unweighted average of the two average values. Linearly interpolates between the midperiod and period boundary. It is continuous but does not preserve integral for unequal periods.

  Method=3 is for equal area (midperiod to midperiod) interpolation. It chooses a value for the boundary between two adjacent periods such that linear interpolation be-

[22]Refer to Chapters 10 and 14. In CASE=0, the surface boundary conditions are zonally averaged time means. Since they are only functions of latitude, array $sbcocn_{i,jrow,m}$ does not exist and surface boundary conditions are set directly into arrays $stf_{i,j,n}$ and $smf_{i,j,n}$.

[23]It should be noted that there is enough generality to accommodate datasets with other periods such as daily, hourly, etc and treat them as climatologies (periodic) or real data (non periodic). Also datasets with differing periods may be mixed. For example: climatological monthly SST may be used with hourly winds.

[24]The one which is no longer needed.

tween the two midperiods and this value will preserve the integral midperiod to midperiod.

To explore how time interpolation works in a very simple environment, one can execute script *run_timeinterp* which exercises *timeinterp*[25] as a stand alone program.

As mentioned above, two additional boundary condition arrays are needed for each surface boundary condition. The memory increase may get excessive with large resolution models. In that case, option *minimize_sbc_memory* reduces these arrays from being dimensioned as (imt,jmt) to being dimensioned as (imt,jmw). The trade off is increased disk access which may be prohibitive if using rotating disk. If efficiency is an issue, asynchronous reads with look ahead capability would be worth trying. This look ahead feature has not been implemented.

19.2.1 Bulk parameterizations

Section 19.2 describes three simple atmospheres requiring very little computation. In all cases, the atmosphere is sensitive to SST but not sea surface salinity. Since fresh water flux into the ocean is not known very accurately, it is reasonable to damp sea surface salinity back to climatological values on some Newtonian time scale for a surface boundary condition. As mentioned in Section 19.3, restoring SST and sea surface salinity to data can be done by enabling option *restorst*. Note that damping time scale may by set differently for each tracer. However, instead of restoring SST, the next simplest atmosphere in the hierarchy may be thought of as being parameterized with bulk formulae as given by the idealized version in Philander/Pacanowski (1986) or the more complete version in Rosati/Miyakoda (1988).

Although these bulk parameterizations are not included in MOM as of this writing, they are easy to implement. The largest uncertainties are due to clouds. Caution must be used with this type of atmosphere since the global integral of heatflux into the ocean averaged over one seasonal cycle may be non-zero which may lead to a drift in the ocean heat content with time.

19.3 Surface boundary conditions

As mentioned in the previous sections, surface boundary conditions are contained within array $sbcocn_{i,jrow,m}$ and defined on the MOM grid. If option *coupled* is enabled, the same set is also contained within array $sbcatm_{i',j',m}$ defined on the atmosphere grid. The array $sbcatm_{i',j',m}$ only exists if option *coupled* is enabled. The total number of surface boundary conditions *numsbc* is partitioned into *numosbc* for the ocean and *numasbc* for the atmosphere. Surface boundary conditions are chosen from a list of possibilities maintained within module *sbc_info*. The module contains two routines: *setup_sbc* selects a particular surface boundary condition from the list of possible ones and a sequence of ordered calls to *setup_sbc* builds the desired ordered list of surface boundary conditions. The other routine *index_of_sbc* returns the index of any surface boundary condition within the ordered list. The ordering of surface boundary conditions is the same in array $sbcocn_{i,jrow,m}$ and array $sbcatm_{i',j',m}$; the only difference is that one set is on the MOM grid and the other is on the atmosphere grid.

[25]Contained in file timeinterp.F. This is the subroutine that does interpolations in time.

19.3.1 Default Surface boundary conditions

The default set of surface boundary conditions and ordering for the MOM test case is constructed by the following code in driver.F:

```
m = 0
call setup_sbc ('taux', m)
call setup_sbc ('tauy', m)
call setup_sbc ('heatflux', m)
call setup_sbc ('saltflux', m)
numosbc = m

call setup_sbc ('sst', m)
call setup_sbc ('sss', m)
numasbc = m - numosbc
numsbc  = numasbc + numosbc
```

where "numosbc" is the number of surface boundary conditions for the ocean, "numasbc" is the number of surface boundary conditions for the atmosphere, and "numsbc" is the total number. Given the above ordering of calls, the index for heatflux "i" can be determined from

```
i = index_of_sbc ('heatflux')
```

and the value of i will be $i = 3$. The above six boundary conditions within $sbcocn_{i,jrow,m}$and $sbcatm_{i',j',m}$will be indexed as follows:

- m=1 for "taux" which references $\vec{\tau} \cdot \hat{\lambda}$ which is the eastward windstress exerted on the ocean in units of $dynes/cm^2$. If the wind in the atmosphere model is positive (blowing towards the east) then the ocean and land exert a westward stress which decelerates the atmosphere. From the atmosphere point of view, this stress is negative. However, the ocean and land are being accelerated by the transfer of momentum through the boundary layer. Therefore the correct sign for the stress acting on the ocean is positive (towards the east).
- m=2 for "tauy" which references $\vec{\tau} \cdot \hat{\phi}$ which is the northward windstress exerted on the ocean in units of $dynes/cm^2$. A northward windstress is positive.
- m=3 for "heatflux" which references heat flux in units of *langley/sec* where 1 $langley = 1$ cal/cm^2. A positive heatflux means heat is being pumped into the ocean.
- m=4 for "saltflux" which references a salinity flux into the ocean in units of *grams/cm*^{2}*/sec*. In MOM, the rigid lid approximation implies that ocean volume is constant[26]. Therefore, when it rains, salt must change since ocean volume cannot. If the atmosphere model supplies a fresh water flux, this should be converted to a salt flux = -(P-E+R) S_{ref} where P-E+R represents a precipitation minus evaporation plus runoff rate in *cm/sec*, and S_{ref} is a

[26]When using option *implicit_free_surface* the ocean volume is allowed to change. This allows for a fresh water flux to be used directly. This has not been implemented as of this writing. The recommendation is to set the salinity flux to zero and add fresh water flux directly to the free surface elevation.

reference salinity in units of grams of salt per gram of water (units of *parts per part* such as 0.035). Depending upon the application of interest, S_{ref} may be a constant over the entire model domain or the locally predicted salinity of the uppermost model level. If the intent is for a global average P-E+R flux of zero to imply a zero trend in the ocean salt content, then S_{ref} must be chosen as a constant. A positive precipitation minus evaporation into the ocean means that fresh water is being added to the ocean which decreases the salinity. This implies that the salt flux is negative.

- m=5 references "sst" which is sea surface temperature in units of deg C.
- m=6 references "sss" which is sea surface salinity in units of (S-35.0)/1000.

Other surface boundary conditions are possible as defined within the module list. For instance ...

```
call setup_sbc ('short wave', m)
```

sets up a solar short wave flux in units of *langley/sec* where 1 *langley* $= 1\ cal/cm^2$. Normally this effect is included in the surface heat flux. However, solar short wave penetration into the ocean is a function of wavelength. The default clear water case (Jerlov turbidity type I) assumes energy partitions between two exponentials as follows: 58% of the energy decays with a 35 cm e-folding scale; 42% of the energy decays with a 23 m e-folding scale. If the thickness of the first ocean level $dzt_{k=1} = 50$ meters, then shortwave penetration wouldn't matter. However, for $dzt_{k=1} = 10$ meters, the effect can be significant and may be particularly noticeable in the summer hemisphere. See Paulson and Simpson (1977), Jerlov (1968) and Rosati (1988).

Surface boundary conditions such as solar shortwave "short wave" or fresh water flux "fresh wtr" require datasets that are not part of MOM. It is the responsibility of the researcher to supply these datasets.

When option *restorst* is enabled, surface tracers are restored to prescribed data $t^{\star}_{i,j,n,time}$[27] using a Newtonian damping time scale *dampts* in units of days input through a namelist. Refer to Section 14.4 for information on namelist variables. Note that the damping time scale may by set differently for each surface tracer. The Newtonian damping term is actually converted into a surface tracer flux as described under option *restorst* in Section 28.2.9.

19.3.2 Adding or removing surface boundary conditions

Look in module *setup_sbc* to see the list of possbile surface boundary conditions. The list can be extended indefinately. Then select the desired surface boundary conditions using a set of ordered calls in driver.F as indicated in Section 19.3. Alternatively, the set may be entered through namelist. If entering surface boundary condition names through namelist, make sure to enter the complete set of all names. The ordering of names determines how the surface boundary conditions will be stored. The only restrictions on ordering are:

[27] A product of PREP_DATA scripts operating on the DATABASE in CASE=1 and CASE=2. CASE=0 uses an idealized data generated internally as a function of latitude only. Interpolations to model time are explained under option *time_varying_sbc_data* in Section 19.2.

- Keep all surface boundary conditions for the ocean bundled together and put them first. Then bundle all surface boundary conditions for the atmosphere together and keep them last.
- Within each bundle. Keep the ordering the same as the variables for which the surface boundary conditions are used. For instance, if there are six tracers, then keep the six tracer surface boundary conditions ordered the same way as the tracers.

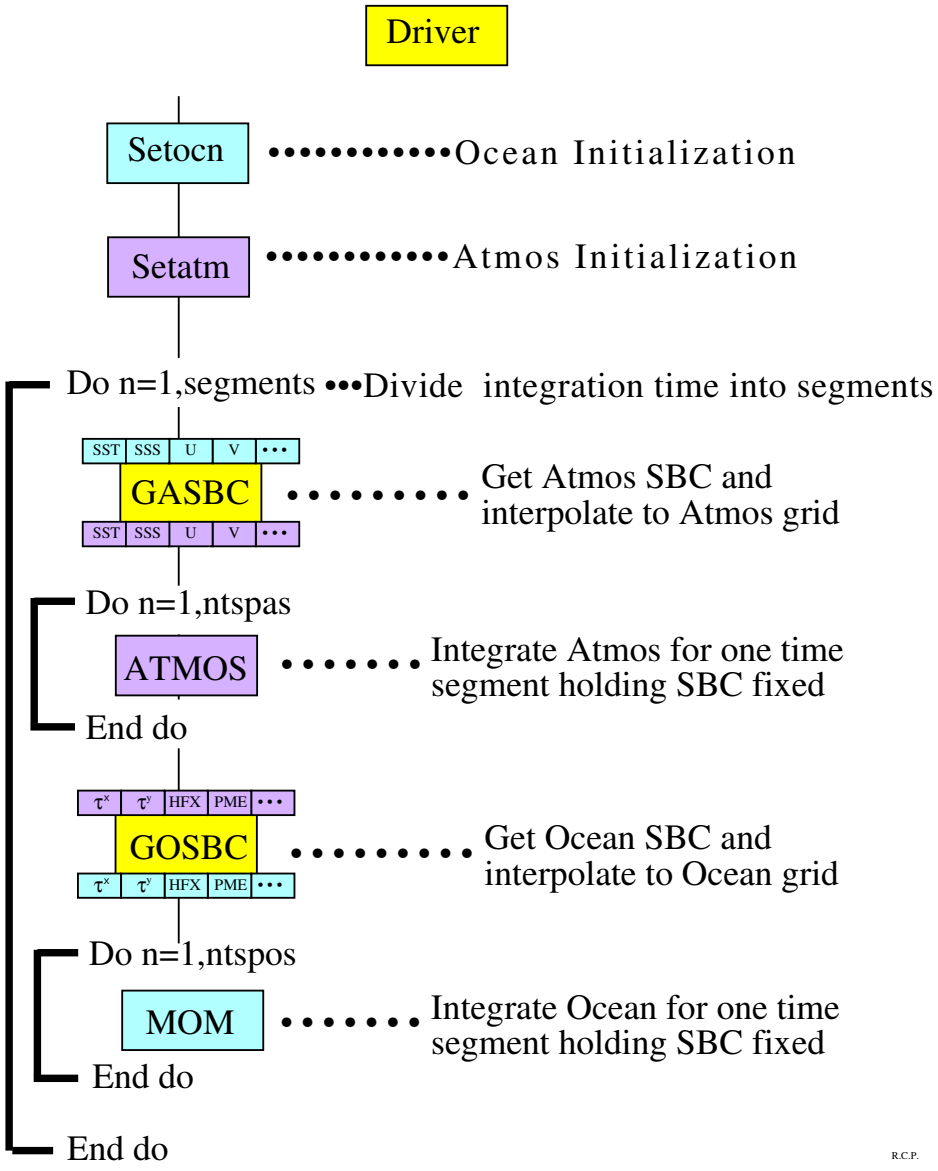


Figure 19.2: Flowchart for main program *driver.F* which controls surface boundary conditions in MOM

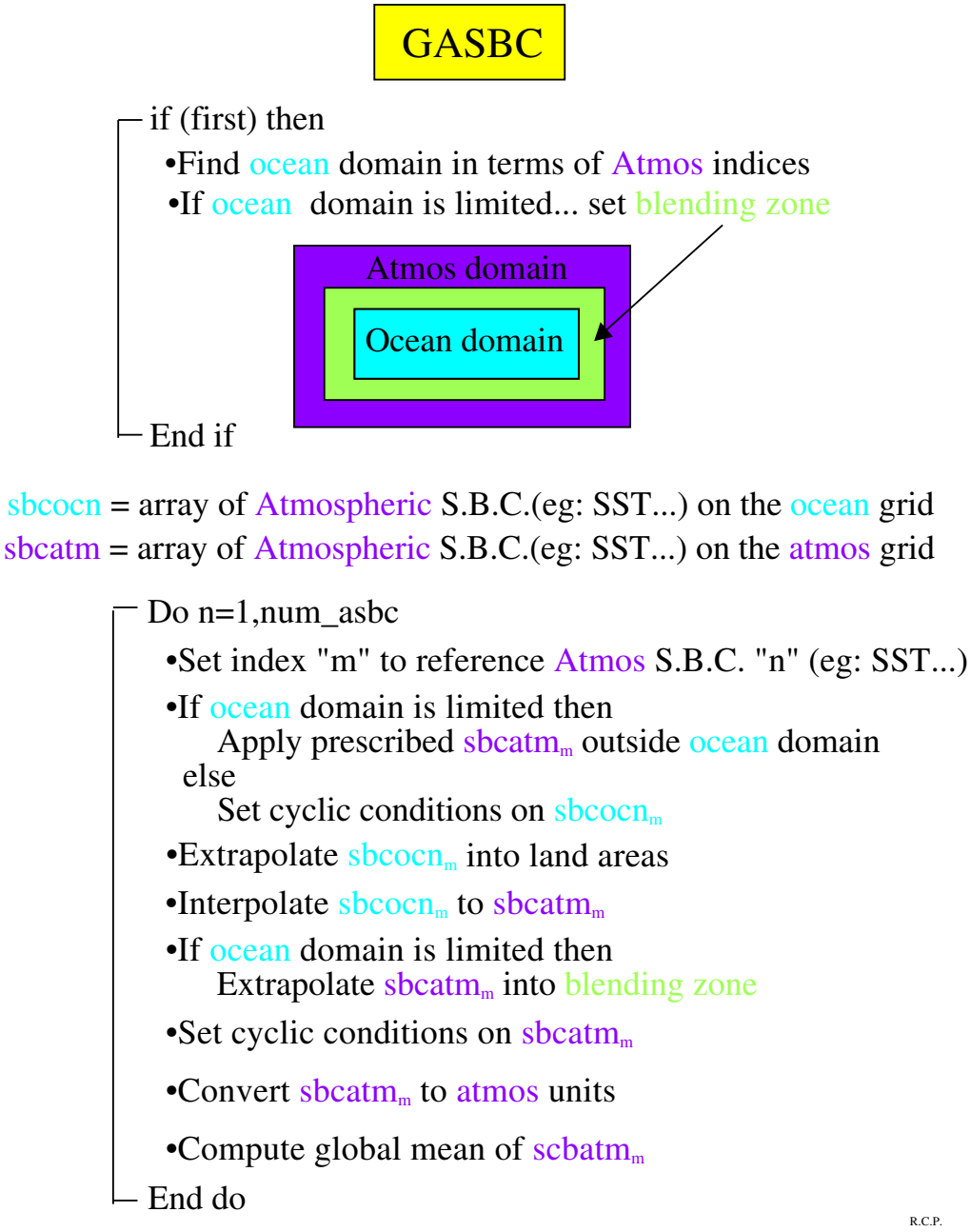


Figure 19.3: Flowchart for subroutine *gasbc.F*

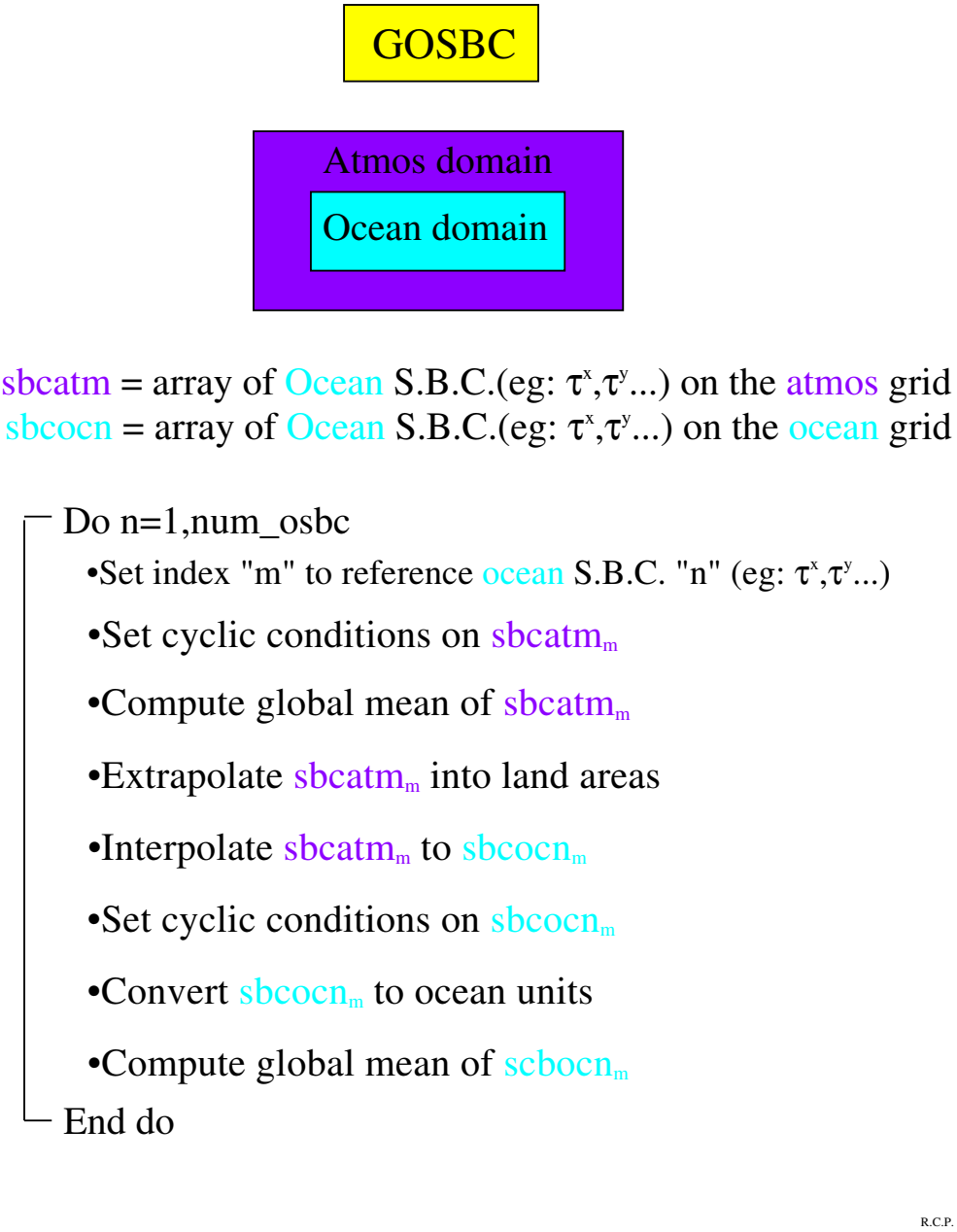


Figure 19.4: Flowchart for subroutine *gosbc.F*

Chapter 20

Stevens Open Boundary Conditions

This chapter was contributed by Arne Biastoch (abiastoch@ifm.uni-kiel.de). Open boundary conditions have been tested successfully in the *FRAM* model (Stevens 1991) and in the *community modeling effort (CME)*, (e.g. Döscher and Redler 1995). They have also been used in a GFDL-model of the North Atlantic in the framework of *MAST2-DYNAMO* (Dynamo 1994) as well as in a regional model of the subpolar North Atlantic (Redler and Böning 1996). The above mentioned experiments performed at Kiel were based on René Redler's[1] implementation in MOM 1. This chapter was written by Arne Biastoch[2] and describes an implementation of these open boundary conditions at the northern, southern, eastern and/or western boundary for a basin in MOM. The approach is based on the methodology of Stevens (1990).

There are two types of open boundary conditions: 'active' in which the interior is forced at inflow points by data prescribed at the boundary and 'passive' in which there is no forcing at the boundary and phenomena generated within the domain can propagate outward without disturbing the interior solution.

At open boundaries, baroclinic velocities are calculated using linearized horizontal momentum equations and the streamfunction is prescribed from other model results or calculated transports (e.g. directly or indirectly from the Sverdrup relation). Thus, the vertical shear of the current is free to adjust to local density gradients and internal waves are allowed to propagate out through the boundaries. Heat and salt are advected out of the domain if the normal component of the velocity at the boundary is directed outward. When the normal component of the velocity at the boundary is directed inward, heat and salt are restored to prescribed data ('active' open boundary conditions). During the course of a model integration, grid points can change from inflow to outflow (and vice-versa).

In contrast to 'active' open boundary conditions, 'passive' ones are characterized by not restoring tracers at inflow points. Additionally, a simple Orlanski radiation condition (Orlanski 1976) is used for the streamfunction.

In what follows, details are given for the northern and southern open boundaries. Open boundaries at the eastern and western end of the domain are handled in a similar manner except where noted.

WARNING: Tests for 'active' and 'passive' open boundary conditions are included as options. However, open boundary conditions is not an option which can be simply enabled to see what happens. Not only is data along the open boundaries needed (in the case of 'active'

[1]Check the mailing list for e-mail address.

[2]Dept. Theoretical Oceanography, Institut für Meereskunde, Düsternbrooker Weg 20, 24105 Kiel, Germany. Check the mailing list for e-mail address.

open boundaries) but topography along open boundary points must be chosen carefully. Additionally, prescribing a net tranport along the open boundary (via ψ_b), requires that changes be made in the stream function calculation. Although open boundary conditions have been tested in simple cases, attention must be given to details to insure that specific configurations are working properly. For more details the reader is referred to Stevens (1990) and a 'TO-DO' list given at the end of this chapter.

20.1 Boundary specifications

Geographically, open boundaries within MOM are defined at latitude rows $jrow = 2$ and $jrow = jmt - 1$ (or at longitudes given by $i = 2$ and $i = imt - 1$). This is in contrast to Stevens (1990) where the open boundaries reside directly at the sides of the domain. The advantage of this definition within MOM is that memory management, diagnostics, and the Poisson solver for the external mode remain unchanged. On the other side the model domain has to be enlarged by one row/column per open boundary.

During outflow conditions, new tracer values are calculated on the boundaries using

$$\frac{\partial T}{\partial t} + \frac{v_{ad} + c_T}{a}\frac{\partial T}{\partial \phi} = F^T \tag{20.1}$$

$$\text{where } c_T = -a\frac{\partial T}{\partial t}/\frac{\partial T}{\partial \phi},\ -a\Delta\phi/\Delta t \leq c_T \leq 0$$

where c_T is the phase speed of the tracer T (where the sign is appropriate for the southern open boundary), v_{ad} meridional velocity, a radius of the Earth and ϕ latitude. The limitation of c_T is due to the CFL criteria.

During inflow conditions, nothing is done unless the open boundary is 'active'. If 'active', tracer values at the boundary are set using a Newtonian restoring condition on a time scale defined by the researcher.

The baroclinic velocity components at $jrow = 2$ and $jrow = jmt - 2$ are calculated by a linearized momentum equation with only the advective terms being omitted.

Solving for the external mode stream function is similar to solving in a closed domain with walls at $jrow = 1$ and jmt. After each solution, two boundary rows are set ('active' open boundary conditions) or calculated ('passive' open boundary conditions) using a radiation condition after Orlanski (1976)

$$\frac{\partial \psi}{\partial t} = -\frac{c_\psi}{a}\frac{\partial \psi}{\partial \phi} \tag{20.2}$$

Because the Poisson solver calculates the updated stream function from $jrow = 3$ to $jrow = jmt - 2$ (In Stevens (1990) ψ is calculated from from $jrow = 2$ to $jrow = jmt - 1$) the first and the last two boundary rows are equal in ψ. This is desirable because there should be no barotropic velocity tangent to the boundary and a zero derivative across the first two rows assures this.

20.2 Options

When using open boundary conditions, it is advisable to configure a case using the simplest most straightforward options before moving to more complex ones. In this way, problems can more easily be isolated. The following options are available as *ifdefs*.

obc_south open boundary at the southern wall ($jrow = 2$).

obc_north open boundary at the northern wall ($jrow = jmt - 1$).

obc_west open boundary at the western wall ($i = 2$).

obc_east open boundary at the eastern wall ($i = imt - 1$).

obc is automatically defined if any of the above *ifdefs* are enabled.

orlanski Orlanski radiation condition for 'passive' open boundary conditions . If not set: 'active' open boundary conditions are assumed.

obcparameter writes open boundary conditions related parameters into snapshot file (only if **obc_south** and/or **obc_north** are enabled)

obctest configuration for 'passive' open boundary conditions test from Chap.5 of Stevens (1990). Requires **obc_north**, **obc_south** and **orlanski**. The option **no_sbc** (no surface boundary conditions) must also be enabled.

obctest2 configuration for 'active' open boundary conditions test from Chap.6 of Stevens (1990). Requires **obc_south** . Data for the southern boundary is taken from a run with a larger domain without **obc_south** enabled.

The following *options* have been tested with open boundary conditions

cray_ymp, cray_t90, generate_a_grid, read_my_grid, generate_a_kmt, read_my_kmt, etopo_kmt, flat_bottom, rectangular_box, timing, restorst, source_term, sponges, levitus_ic, time_varying_sbc_data, simple_sbc, constvmix, consthmix, biharmonic, fullconvect, stream_function, sf_9_point, conjugate_gradient and several diagnostic options

The following *options* do **not** work:

symmetry with **obc_north,obc_south**; **cyclic** with **obc_west,obc_east**

20.3 New Files

cobc.F This subroutine is used instead of `tracer.F` at northern and southern open boundaries. At boundary cells, new tracer values are calculated using (20.1) for outflow conditions and restored for inflow conditions (only for 'active' open boundary conditions). The subroutine is called by `mom.F`.

cobc2.F This subroutine is used instead of `tracer.F` at western and eastern open boundaries and is similar in most respects to `cobc.F`, but called from `tracer.F`.

addobcpsi.F This subroutine is used for prescribing values of ψ at open boundaries in the case of 'active' open boundary conditions. It is called by `topog.F`.

cobc.h Coefficient definitions are given for advective and phase velocities at open boundaries.

obc_data.h Definitions are given for open boundary data, pointers and damping coefficients (based on `sponge.h`).

20.4 Important changes to existing subroutines

To keep the structure of the model clear substantial changes to the code are contained in include files (`*.inc`) which are included when obc options are enabled.

baroclinic.F (baroclinic_obc.inc) Linearize velocity components at open boundaries (which simply means to omit the advective terms).

loadmw.F (loadmw_obc.inc) Set velocity values at $jrow = 1$ and $jrow = jmt - 1$ to prevent diffusion over the boundaries: $u_{i,k,1,n} = u_{i,k,2,n}, u_{i,k,jmt,n} = u_{i,k,jmt-1,n} = u_{i,k,jmt-2,n}$ (for τ and $\tau - 1$). Set rho values at $jrow = jmt$ to prevent errors in the calculation of the hydrostatic pressure gradients at the northern boundary: $\rho_{i,k,jmt,n} = \rho_{i,k,jmt-1,n}$.

mom.F (mom_obc.inc)
- first memory window (example for options *obc_south* and *obc_north*, for options *obc_west* and *obc_east* changes are in the routines itself):
  - `cobc` for $jrow = 2$
  - `tracer` for the rest of the memory window
  - call `baroclinic` as without obc (changes are **in** the subroutine itself)
- last memory window:
  - `tracer` for the first part of the window
  - `cobc` for $jrow = jmt - 1$
  - call `baroclinic` as without obc

setocn.F (setocn_obc.inc) Initialization of several quantities and reading of the open boundary data (only active open boundaries)

topog.F Potentially, open boundaries can reside at latitudinal rows $jrow = 2$ and $jrow = jmt - 1$ and at longitudinal columns $i = 2$ and $i = imt - 1$. In support of the open boundaries, virtual points are needed at latiudinal rows $jrow = 1$ and $jrow = jmt$ and at longitudinal columns $i = 1$ and $i = imt$. The topography on the virtual rows/columns must be identical to the topography on the corresponding open boundary row/column and module *topog* insures this. However, on the two rows/columns interior to the open boundary, the topography must not slope upwards toward the interior of the domain. Otherwise the calculation of phase velocities at the open boundary row would deliver wrong values at the bottom cells (example for a southern open boundary: $c_T \propto (t_{i,4,k} - t_{i,3,k})^{-1}$. E.g., if $kmt(i,3) = 4, kmt(i,4) = 3$, then the calculation at $k = 4$ would be point into bottom by $c_T \propto 1/(0 - t_{i,3,4})^{-1}$). Refer to Figure 20.1. Actually a minimum depth is chosen for option *obc_south* as in Figure 20.1d (e.g. $kmt_{i,2} = kmt_{i,3} = kmt_{i,4} = \min(kmt_{i,2}, kmt_{i,3}, kmt_{i,4})$, but the rearcher is free to change this.

tracer.F Before (in the case of an western open boundary) and after (eastern obc) the calculation of new tracer values subroutine `cobc2.F` is called.

barotropic.F (barotropic1_obc.inc, barotropic2_obc.inc) For the Orlanski (1976) radiation condition, new stream function values at the open boundary are calculated using (20.2). For active open boundaries, values for ψ are prescribed by calling subroutine `addobcpsi.F`.

20.5 Data Preparation Routines

obc.F This routine is based on the routine `PREP_DATA/sponge.F` and prepares the open boundary values for T and S. For input it uses the standard Levitus data (prepared by `ic.F`) or ASCII data written by FERRET (Hankin and Denham 1994). The damping factors have to be specified (search for `USER INPUT`). Two files, `obc1.mom` for zonal and `obc2.mom` for meridional open boundaries, are written. Following *ifdefs* can be chosen:

makeobc driver

readferret use FERRET-data as input (formatted listing); otherwise Levitus-data are used

obc_north, obc_south, obc_west, obc_east

write_netcdf additional netCDF output

obcpsi.F This routine prepares open boundary values for ψ, required for 'active' open boundaries. It uses output (formatted listing) of FERRET. Alternatively values for psi must be given directly in the code. Two files, `obcpsi1.mom` for zonal and `obcpsi2.mom` for meridional open boundaries, are written. The same *ifdefs* as in `obc.F` can be chosen.

20.6 TO-DO List (How to set up open boundaries)

This list is only a sketch of what is necessary to use the open boundary options in MOM 2. The interference with new options (e.g. new diffusion schemes) must be tested carefully. It must be kept in mind that the geographical position of the open boundaries and the prescribed data can substantially affect the solution.

1. Choose regions of open boundaries carefully at locations where the barotropic velocity is mostly perpendicular to the boundary.

2. Open the grid one cell further for open boundaries, so the open boundary conditions reside at the second row (column) in the domain. The first row(column) is only a virtual dummy point. Topography is set automatically by subroutine `topog.F` similar to the open boundary row(column). If the researcher does not want the drastic choice of setting the next three rows(columns) identically, the corresponding lines in `topog.F` must be commented out and it is up to the researcher to insure that the bottom topography is not sloping upward within the next two rows away from the boundary.

3. Choose a restoring time scale (`USER INPUT` in `obc.F`) and create tracer values at the open boundaries either from Levitus or FERRET written data.

4. Choose ψ values at the open boundaries in `obcpsi.F` or read FERRET written data to create ψ values.

5. If ψ values at the open boundaries indicate a net transport through the domain, land masses have to be set to a non-zero value. This has to be done at the end of the subroutine `adobcpsi.F`. In the following example (Fig. 20.2) open western and eastern boundaries are chosen, transporting 130 Sv through the domain. *psiwall_west* and *psiwall_east* rise from zero at the northern end to 130 Sv at the southern end, whereby the shapes of the curves can be different. To guide the current through the domain all southern land masses must set to 130 Sv.

```
do jrow=1,jpsimax+1
  do i=1,imt
    if (map(i,jrow) .ne. 0) then
      psi(i,jrow,1) = 130.e12
      psi(i,jrow,2) = 130.e12
    endif
  enddo
enddo
```

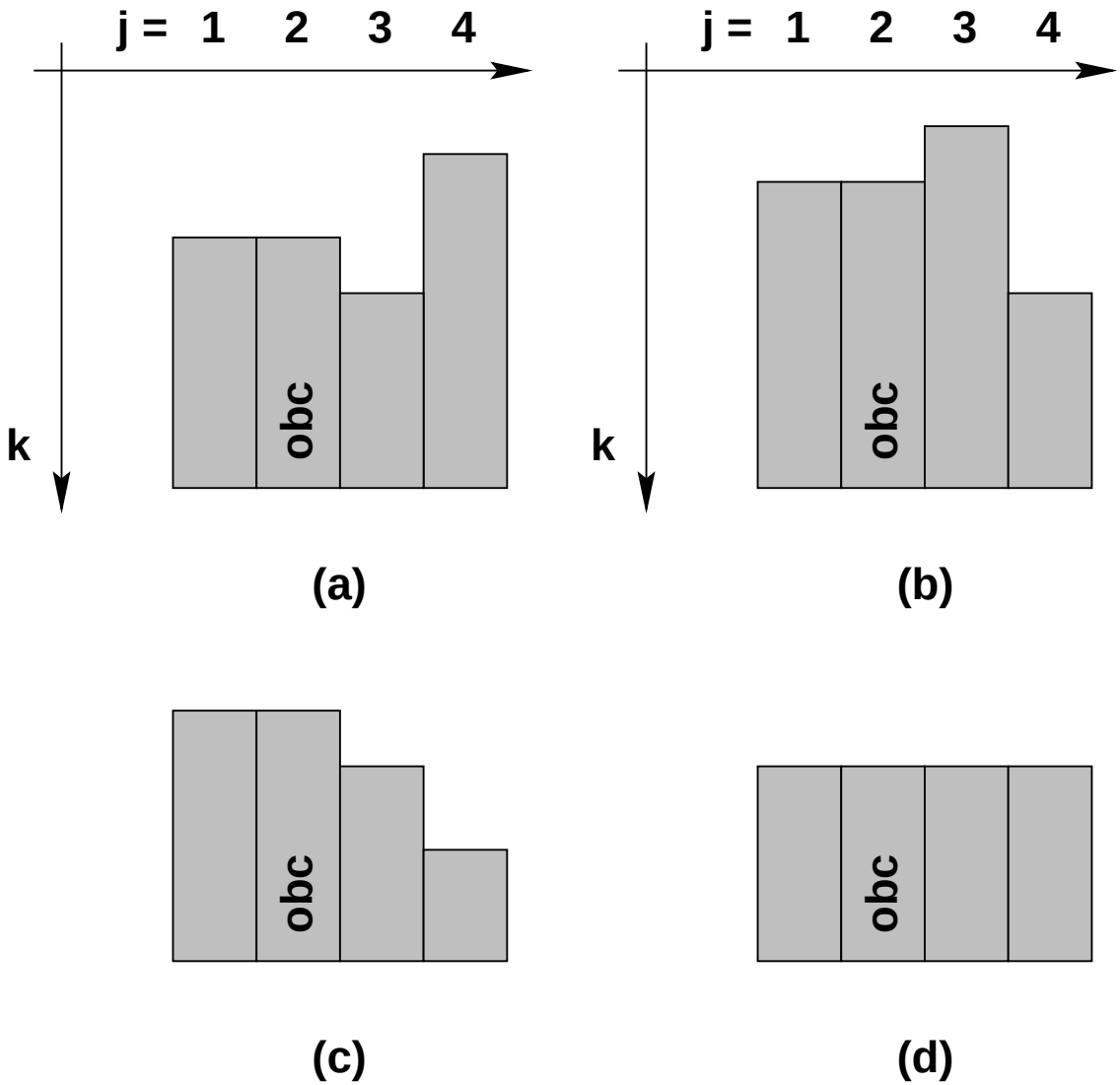


Figure 20.1: Topography at the open boundary (y-z-view). For southern obc, (c) and (d) are allowed configurations, but (a) and (b) are not. Configuration (d) is the default construced by *topog.F*.

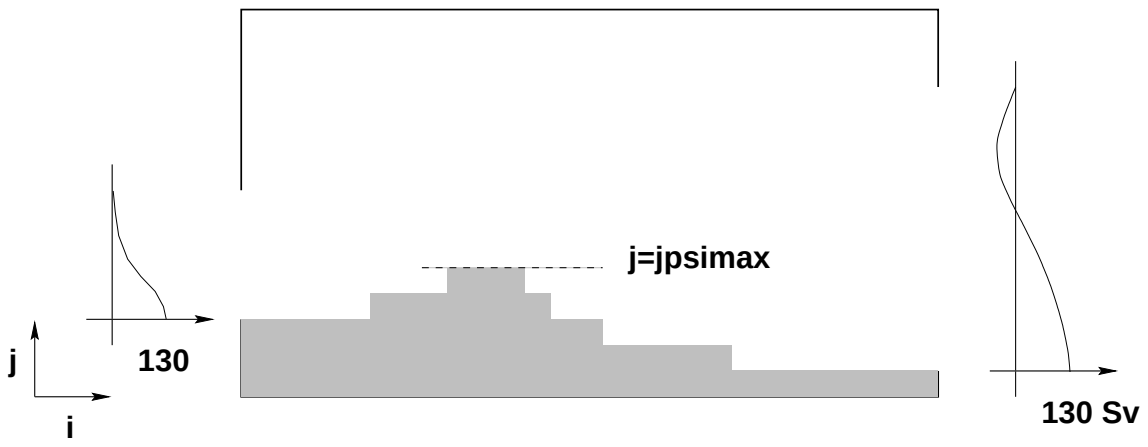


Figure 20.2: Example for throughflow of 130 *Sv* from western to eastern open boundary (x-y-view). *jpsimax* is the northernmost extension of the southern land mass

Part VI

Finite Difference Equations

Chapter 21

The Discrete Equations

This chapter details the time and space discretizations used for writing the finite difference counterpart of the continuous prognostic equations in Chapter 4. After these preliminaries, the finite difference equations are given for the simplest case of options *consthmix* and *constvmix*. Note that throughout all equations were appropriate, numerical indices are exposed within expressions so as to match exactly with those used in the fortran code. This extra bit of detail is insisted upon as another way of insuring that complex formulations are correct. MOM uses a wide variety of options to configure variants of the prognostic equations in Chapter 4. The available options are explained in Part VII and the steps for solving these equations are covered in Chapter 22.

21.1 Time and Space discretizations

The following operators are used to discretize equations on the grid system used in MOM. Before going further, the grid system should be firmly in mind. Refer to Section 16.2 if in doubt.

21.1.1 Averaging operators

Simple averaging operators are defined as follows:

$$\overline{\alpha_{i,k,j}}^{\lambda} = \frac{\alpha_{i+1,k,j} + \alpha_{i,k,j}}{2} \tag{21.1}$$

$$\overline{\alpha_{i,k,j}}^{\phi} = \frac{\alpha_{i,k,j+1} + \alpha_{i,k,j}}{2} \tag{21.2}$$

$$\overline{\alpha_{i,k,j}}^{z} = \frac{\alpha_{i,k+1,j} + \alpha_{i,k,j}}{2} \tag{21.3}$$

where α is any variable defined on grid points within T-cells or U-cells. It should be noted that the average is defined midway between the variables being averaged.

21.1.2 Derivative operators

Simple derivative operators in space and time are defined as follows:

$$\delta_\lambda(\alpha_{i,k,j}) = \frac{\alpha_{i+1,k,j} - \alpha_{i,k,j}}{a\Delta\lambda_i} \tag{21.4}$$

$$\delta_\phi(\alpha_{i,k,j}) = \frac{\alpha_{i,k,j+1} - \alpha_{i,k,j}}{a\Delta\phi_{jrow}} \tag{21.5}$$

$$\delta_z(\alpha_{i,k,j}) = -\frac{\alpha_{i,k+1,j} - \alpha_{i,k,j}}{\Delta z_k} \tag{21.6}$$

$$\delta_\tau(\beta_\tau) = \frac{\beta_{\tau+1} - \beta_{\tau-1}}{2\Delta\tau} \tag{21.7}$$

where grid distances (measured in *cm*) are determined by the distance between variables as indicated in Figures 16.2, 16.3, and 16.4 and discussed in Chapter 14. These operators are second order accurate with non-uniform resolution as long as the grid is constructed so that the stretching is based on a smooth analytic function. See Treguier, Dukowicz, and Bryan (1995). When $\alpha_{i,k,j}$ is defined at $T_{i,k,j}$grid points, then $a\Delta\lambda_i = dxu_i$ ($a = 6370$ x 10^5 *cm*) and when $\alpha_{i,k,j}$ is defined at $U_{i,k,j}$grid points, then $a\Delta\lambda_i = dxt_{i+1}$. Similarly, $a\Delta\phi_{jrow} = dyu_{jrow}$ when $\alpha_{i,k,j}$ is defined at $T_{i,k,j}$grid points and $a\Delta\phi_{jrow} = dyt_{jrow+1}$ when $\alpha_{i,k,j}$ is defined at $U_{i,k,j}$grid points. Note the negative sign in the vertical derivative. This is because z increases upwards while k increases downwards. The negative sign in Equation (21.6) is usually absorbed by reversing the indexing to give

$$\delta_z(\alpha_{i,k,j}) = \frac{\alpha_{i,k,j} - \alpha_{i,k+1,j}}{\Delta z_k} \tag{21.8}$$

In the finite difference approximation to the continuous time derivative, Equation (21.7) is appropriate for the normal leapfrog time steps where $2\Delta\tau$ is in seconds. As indicated in Section 21.4, on mixing time steps, the denominator is replaced by $\Delta\tau$.

21.2 Key to understanding finite difference equations

The grid distances (measured in *cm*) are determined by the distance between variables as indicated in Figures 16.2, 16.3, and 16.4. Before looking at any finite difference equations, the reader should be convinced of the following relations:

- If $\alpha_{i,k,j}$ is defined at the T cell grid point given by $T_{i,k,j}$, then the operation $\delta_\lambda(\overline{\alpha_{i,k,j}}^\phi)$ results in a quantity defined at the U cell grid point with index $U_{i,k,j}$.

- If $\beta_{i,k,j}$ is defined at the U cell grid point given by $U_{i,k,j}$, then the operation $\delta_\lambda(\overline{\beta_{i-1,k,j-1}}^\phi)$ results in a quantity defined at the T cell grid point given by $T_{i,k,j}$.

These operations reflect the nature of the staggered B grid. Once convinced of the above, the second thing to be aware of is that it's not sufficient to only know what a quantity is; where the quantity is defined is just as important. The <u>where</u> information is usually built into the name of the variable by the naming convention described in Chapter 14. There are a small number of interesting places on the grid. The grid point within a grid cell is one; the east, north, and bottom face of a cell are others. A quantity indexed by (i, k, j) may be defined at the grid point in $cell_{i,k,j}$ or on the east, north, or bottom face of $cell_{i,k,j}$. Note that if a variable $\alpha_{i,k,j}$

is defined on the east face of $cell_{i,k,j}$, its value on the west face of the cell is $\alpha_{i-1,k,j}$. Similarly, if defined on the north face of $cell_{i,k,j}$, then its value on the south face of the cell is $\alpha_{i,k,j-1}$. Likewise, if defined on the bottom face of $cell_{i,k,j}$, then its value on the top face of the cell is $\alpha_{i,k-1,j}$.

Note that this convention is a departure from the indexing used in Bryan (1969) where the faces of cells were referenced by half indexes (i.e., $i + \frac{1}{2}$). Half indexes do not map well into Fortran and are not used in this manual. The idea is that by looking at this manual it should be possible to determine if the code is wrong (or vice versa).

21.2.1 Rules for manipulating operators

In general, finite difference derivative and average operators <u>don't</u> commute unless the grid resolution is constant. Assuming that α_i is defined at grid points within T-cells, then the above condition is illustrated by the following

$$\delta_\lambda(\overline{\alpha_i}^\lambda) \neq \overline{\delta_\lambda(\alpha_i)}^\lambda \tag{21.9}$$

How is a term like $\delta_\lambda(\overline{\alpha_i}^\lambda)$ evaluated? It can be expanded from the inside out as

$$\begin{aligned} \delta_\lambda(\overline{\alpha_i}^\lambda) &= \delta_\lambda(\frac{\alpha_i + \alpha_{i+1}}{2}) \\ &= \frac{(\alpha_{i+1} + \alpha_{i+2})/2 - (\alpha_i + \alpha_{i+1})/2}{dxt_{i+1}} \\ &= \frac{\alpha_{i+2} - \alpha_i}{2 \cdot dxt_{i+1}} \end{aligned} \tag{21.10}$$

or from the outside in as

$$\begin{aligned} \delta_\lambda(\overline{\alpha_i}^\lambda) &= \frac{\overline{\alpha_{i+1}}^\lambda - \overline{\alpha_i}^\lambda}{dxt_{i+1}} \\ &= \frac{(\alpha_{i+1} + \alpha_{i+2})/2 - (\alpha_i + \alpha_{i+1})/2}{dxt_{i+1}} \\ &= \frac{\alpha_{i+2} - \alpha_i}{2 \cdot dxt_{i+1}} \end{aligned} \tag{21.11}$$

Both results are equal. Now expand the following:

$$\overline{\delta_\lambda(\alpha_i)}^\lambda = \frac{(\alpha_{i+1} - \alpha_i)/dxu_i + (\alpha_{i+2} - \alpha_{i+1})/dxu_{i+1}}{2} \tag{21.12}$$

Equation (21.12) is only equal to Equation (21.11) when

$$dxt_{i+1} = dxu_i = dxu_{i+1} \tag{21.13}$$

Also, it is worth remembering that the results of operators are displaced by the distance of a half cell width. For example, the single operator $\delta_\lambda(\alpha_{i,k,j})$ results in a quantity defined on the

eastern face[1] of cell $T_{i,k,j}$ and the double operator $\delta_\lambda(\delta_\lambda(\alpha_{i,k,j}))$ results in a quantity defined at the grid point within $T_{i+1,k,j}$. These results easily extend to two and three dimensions. Mixed double operators such as $\delta_\lambda(\overline{\alpha_{i,k,j}}^\phi)$ results in a quantity defined on the grid point within cell $U_{i,k,j}$.

21.2.2 Rules involving summations

A zero normal component of velocity on walls (material surfaces) implies the following identities

$$\sum_{i=2}^{imt-1} u_i\, \overline{\alpha_i}^\lambda dxu_i = \sum_{i=2}^{imt-1} \overline{u_{i-1}\, dxu_{i-1}}^\lambda\, \alpha_i \tag{21.14}$$

$$\sum_{i=2}^{imt-1} u_i\, \delta_\lambda(\alpha_i) dxu_i = -\sum_{i=2}^{imt-1} \delta_\lambda(u_{i-1})\, \alpha_i\, dxt_i \tag{21.15}$$

Similar identities hold in the vertical (rigid lid only) and meridional directions.

21.2.3 Other rules

There is no ambiguity in manipulating finite difference objects. As noted in Bryan (1969), there are formal rules and some are given below. They can be verified by substituting the basic finite difference derivative and averaging operators and expanding the terms. For illustrative purposes, consider <u>one</u> dimensional quantities α_i and γ_i (defined on longitudes of T cell grid points) and β_i (defined on longitudes of U cell grid points).

$$\overline{\alpha_i\gamma_i}^\lambda = \overline{\alpha_i}^\lambda\overline{\gamma_i}^\lambda + \frac{1}{4} dxu_i^2 \delta_\lambda(\alpha_i)\delta_\lambda(\gamma_i) \tag{21.16}$$

$$\delta_\lambda(\alpha_i\gamma_i) = \overline{\alpha_i}^\lambda\delta_\lambda(\gamma_i) + \overline{\gamma_i}^\lambda\delta_\lambda(\alpha_i) \tag{21.17}$$

$$dxt_{i+1}\cdot\delta_\lambda(\overline{\alpha_i}^\lambda\beta_i) = \overline{\beta_i\cdot dxu_i\cdot\delta_\lambda(\alpha_i)}^\lambda + \alpha_{i+1}\cdot dxt_{i+1}\cdot\delta_\lambda(\beta_i) \tag{21.18}$$

$$\overline{\overline{\alpha_i}^\lambda\beta_i}^\lambda = \alpha_{i+1}\cdot\overline{\beta_i}^\lambda + \frac{1}{4} dxt_{i+1}\delta_\lambda(\beta_i\cdot dxu_i\cdot\delta_\lambda(\alpha_i)) \tag{21.19}$$

These expressions also hold along other dimensions. In particular if ϕ is substituted[2] for λ or if z is substituted for λ. Consider further the <u>two</u> dimensional case where α_i is defined at T cell grid points, and β_i is defined at U cell grid points. The following rule may be derived by combining Equations (21.18) and (21.19)

$$\begin{aligned} & \alpha_{i,k,j}\cdot dxt_i\cdot\delta_\lambda(\overline{\beta_{i-1,k,j-1}}^\phi) + \overline{\beta_{i-1,k,j-1}dxu_{i-1}\cdot\overline{\delta_\lambda(\alpha_{i-1,k,j-1})}^\phi}^{\phi\lambda} \\ = \; & dxt_i\cdot\delta_\lambda(\overline{\alpha_{i-1,k,j}}^\lambda\overline{\beta i-1,k,j-1}^\phi) \\ + \; & \frac{1}{4} dyt_{jrow}\delta_\phi(\overline{\beta_{i-1,k,j-1}\cdot dyu_{jrow-1}\delta_\phi(dxu_{i-1}\delta_\lambda(\alpha_{i-1,k,j-1}))}^\lambda) \end{aligned} \tag{21.20}$$

[1]This is the the longitude of the grid point within $U_{i,k,j}$.
[2]The grid distances must also be replaced accordingly.

21.3 Primitive finite difference equations

The finite difference equations are given below for the simplest case of options *consthmix* and *constvmix* assuming explicit vertical diffusion and explicit coriolis terms. Note that the numerical indices are exposed within expressions so as to match exactly with those used in the fortran code. This extra bit of detail is insisted upon as another way of insuring that complex formulations are correct. The steps for solving these equations (plus other variants) are detailed in Chapter 22.

21.3.1 Momentum equations

Prognostic equations are written for the zonal (n=1) and meridional (n=2) component of velocity by breaking each component into internal and external modes. The internal modes for both components are solved for first. For each component, of the uncorrected internal mode velocity $u^{\star}_{i,k,j,n}$, the equation is

$$\delta_\tau(u^{\star}_{i,k,j,n}) = -grad_p_{i,k,j,n} + CORIOLIS_{i,k,j,n} - \mathcal{L}^U(u_{i,k,j,n,\tau}) + \mathcal{D}^U(u_{i,k,j,n,\tau-1}) \tag{21.21}$$

where the advection operator $\mathcal{L}^U$ and diffusion operator $\mathcal{D}^U$ for velocity components are given by

$$\mathcal{L}^U(u_{i,k,j,n,\tau}) = ADV_Ux_{i,k,j} + ADV_Uy_{i,k,j} + ADV_Uz_{i,k,j} - ADV_metric_{i,k,j,n} \tag{21.22}$$

$$\mathcal{D}^U(u_{i,k,j,n,\tau-1}) = DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_Uz_{i,k,j} - DIFF_metric_{i,k,j,n} \tag{21.23}$$

and

$$ADV_Ux_{i,k,j} = \frac{1}{2\cos\phi^U_{jrow}}\delta_\lambda(adv_fe_{i-1,k,j}) \tag{21.24}$$

$$ADV_Uy_{i,k,j} = \frac{1}{2\cos\phi^U_{jrow}}\delta_\phi(adv_fn_{i,k,j-1}) \tag{21.25}$$

$$ADV_Uz_{i,k,j} = \frac{1}{2}\delta_z(adv_fb_{i,k-1,j}) \tag{21.26}$$

$$ADV_metric_{i,k,j,n} = \mp\frac{\tan\phi^U_{jrow}}{radius}\, u_{i,k,j,1,\tau}\cdot u_{i,k,j,3-n,\tau} \tag{21.27}$$

$$DIFF_Ux_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(diff_fe_{i-1,k,j}) \tag{21.28}$$

$$DIFF_Uy_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}}\delta_\phi(diff_fn_{i,k,j-1}) \tag{21.29}$$

$$DIFF_Uz_{i,k,j} = \delta_z(diff_fb_{i,k-1,j}) \tag{21.30}$$

$$DIFF_metric_{i,k,j,n} = A_m\frac{1-\tan^2\phi^U_{jrow}}{radius^2}\, u_{i,k,j,n,\tau-1} \tag{21.31}$$

$$\mp A_m\frac{2\sin\phi^U_{jrow}}{radius^2\cdot\cos^2\phi^U_{jrow}}\delta_\lambda(\overline{u_{i-1,k,j,3-n,\tau-1}}^{\lambda}) \tag{21.32}$$

In the metric terms, the "-" sign is used when $n = 1$ and the "+" sign is used when $n = 2$. The fluxes on U-cell faces are

$$diff_fe_{i,k,j} = \frac{visc_ceu_{i,k,j}}{\cos\phi^U_{jrow}}\delta_\lambda(u_{i,k,j,n,\tau-1}) \tag{21.33}$$

$$diff_fn_{i,k,j} = visc_cnu_{i,k,j} \cdot \cos\phi^T_{jrow+1}\delta_\phi(u_{i,k,j,n,\tau-1}) \tag{21.34}$$

$$diff_fb_{i,k,j} = visc_cbu_{i,k,j} \cdot \delta_z(u_{i,k,j,n,\tau-1}) \tag{21.35}$$

$$adv_fe_{i,k,j} = 2\, adv_veu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^\lambda \tag{21.36}$$

$$adv_fn_{i,k,j} = 2\, adv_vnu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^\phi \tag{21.37}$$

$$adv_fb_{i,k,j} = 2\, adv_vbu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^z \tag{21.38}$$

Horizontal advective velocities on east and north U-cell faces "adv_veu" and "adv_vnu" are defined in Section 22.3.2 and vertical advective velocity on the bottom face of U-cells "adv_vbu" is computed by vertically integrating $\mathcal{L}^U(1) = 0$. Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in *adv_fe*) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. Refer to Section 22.3.3 for a discussion of bottom vertical velocity.

The Coriolis and pressure gradient terms are

$$CORIOLIS_{i,k,j,n} = \pm 2\Omega \sin\phi^U_{jrow} \cdot u_{i,k,j,3-n,\tau} \tag{21.39}$$

$$grad_p_{i,k,j,1} = \frac{1}{\rho_\circ \cdot \cos\phi^U_{jrow}}\delta_\lambda(\overline{p_{i,k,j}}^\phi) \tag{21.40}$$

$$grad_p_{i,k,j,2} = \frac{1}{\rho_\circ}\delta_\phi(\overline{p_{i,k,j}}^\lambda) \tag{21.41}$$

and the "+" sign is used when $n = 1$ and the "-" sign is used when $n = 2$. The pressure p is related to density by

$$\delta_z(p_{i,k,j}) = -grav \cdot \overline{\rho_{i,k,j}}^z \tag{21.42}$$

with gravity $grav = 980.6$ cm/sec^2. Therefore, the pressure at level "k" is given by

$$p_{i,k,j} = -grav \cdot \rho_{i,1,j} \cdot dzw_0 - grav \cdot \sum_{m=2}^{k} \overline{\rho_{i,m-1,j}}^z\, dzw_{m-1} \tag{21.43}$$

The uncorrected internal mode velocity is solved as

$$u'_{i,k,j,n,\tau+1} = u_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,n}) \tag{21.44}$$

and the corrected internal mode velocity (with bogus vertical mean removed) is constructed by

$$\hat{u}_{i,k,j,n,\tau+1} = u'_{i,k,j,n,\tau+1} - \frac{1}{H_{i,jrow}} \cdot \sum_{k=1}^{kb} dzt_k \cdot u'_{i,k,j,n,\tau+1} \tag{21.45}$$

where the bottom U-cell is at level $kb = kmu_{i,jrow}$. As detailed in Section 29.2.1, the external mode velocity components $\bar{u}_{i,j,n,\tau+1}$ are solved for by a variety of methods. Once the external mode is obtained, the full velocity $u_{i,k,j,n,\tau+1}$ is constructed by adding internal mode to external mode

$$u_{i,k,j,n,\tau+1} = \hat{u}_{i,k,j,n,\tau+1} + \bar{u}_{i,j,n,\tau+1} \tag{21.46}$$

21.3.2 Tracer equations

For each component[3] "n" of the tracer field $t_{i,k,j,n}$

$$\delta_t(t_{i,k,j,n,\tau}) = -\mathcal{L}^T(t_{i,k,j,n,\tau}) + \mathcal{D}^T(t_{i,k,j,n,\tau-1}) \tag{21.47}$$

where the advection operator $\mathcal{L}^T$ and diffusion operator $\mathcal{D}^T$ for tracers are

$$\begin{aligned}
\mathcal{L}^T(t_{i,k,j,n,\tau}) &= ADV_Tx_{i,k,j} + ADV_Ty_{i,k,j} + ADV_Tz_{i,k,j} && (21.48)\\
\mathcal{D}^T(t_{i,k,j,n,\tau-1}) &= DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + DIFF_Tz_{i,k,j} && (21.49)
\end{aligned}$$

and

$$\begin{aligned}
ADV_Tx_{i,k,j} &= \frac{1}{2\cos\phi^T_{jrow}}\delta_\lambda(adv_fe_{i-1,k,j}) && (21.50)\\
ADV_Ty_{i,k,j} &= \frac{1}{2\cos\phi^T_{jrow}}\delta_\phi(adv_fn_{i,k,j-1}) && (21.51)\\
ADV_Tz_{i,k,j} &= \frac{1}{2}\delta_z(adv_fb_{i,k-1,j}) && (21.52)\\
DIFF_Tx_{i,k,j} &= \frac{1}{\cos\phi^T_{jrow}}\delta_\lambda(diff_fe_{i-1,k,j}) && (21.53)\\
DIFF_Ty_{i,k,j} &= \frac{1}{\cos\phi^T_{jrow}}\delta_\phi(diff_fn_{i,k,j-1}) && (21.54)\\
DIFF_Tz_{i,k,j} &= \delta_z(diff_fb_{i,k-1,j}) && (21.55)
\end{aligned}$$

The fluxes on T-cell faces are

$$\begin{aligned}
adv_fe_{i,k,j} &= 2\,adv_vet_{i,k,j}\cdot\overline{t_{i,k,j,n,\tau}}^{\lambda} = adv_vet_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}) && (21.56)\\
adv_fn_{i,k,j} &= 2\,adv_vnt_{i,k,j}\cdot\overline{t_{i,k,j,n,\tau}}^{\phi} = adv_vnt_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau}) && (21.57)\\
adv_fb_{i,k,j} &= 2\,adv_vbt_{i,k,j}\cdot\overline{t_{i,k,j,n,\tau}}^{z} = adv_vbt_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau}) && (21.58)\\
diff_fe_{i,k,j} &= \frac{diff_cet_{i,k,j}}{\cos\phi^T_{jrow}}\delta_\lambda(t_{i,k,j,n,\tau-1}) && (21.59)\\
diff_fn_{i,k,j} &= diff_cnt_{i,k,j}\cdot\cos\phi^U_{jrow}\delta_\phi(t_{i,k,j,n,\tau-1}) && (21.60)\\
diff_fb_{i,k,j} &= diff_cbt_{i,k,j}\cdot\delta_z(t_{i,k,j,n,\tau-1}) && (21.61)
\end{aligned}$$

[3]Subscript n=1 denotes the temperature and n=2 denotes the salinity. $n > 2$ is for passive tracers

Horizontal advective velocities on east and north T-cell faces "adv_vet" and "adv_vnt" are defined in Section 22.3.1 and vertical advective velocity on the bottom face of T-cells "adv_vbt" is computed by vertically integrating $\mathcal{L}^T(1) = 0$. Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in *adv_fe*) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Tx) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. Refer to Section 22.3.3 for a discussion of bottom vertical velocity. Each tracer is solved by

$$t_{i,k,j,n,\tau+1} = t_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot \delta_t(t_{i,k,j,n,\tau}) \tag{21.62}$$

21.4 Time Stepping Schemes

Basically, MOM uses a ceneterd leapfrog time stepping scheme to integrate prognostic equations forward in time. To handle the computational mode characteristic of the leapfrog scheme, there are three choices: periodic application of a forward timestep, periodic application of an Euler backward timestep, or a Robert time filter applied every time step.

If option *robert_time_filter* is enabled, a Robert time filter is applied every time step. If option *robert_time_filter* is not enabled, then the type of mixing is determined by logical variable *eb* which if true indicates an Euler backward mixing scheme, otherwise a forward mixing scheme is used. Intervals between mixing timesteps are set by logical switch *mixts*. In the past, mixing time steps were applied every 17 time steps. This interval seems to work satisfactorily and has been retained. Of the two mixing schemes (forward and Euler backward), the Euler backward is more dissipative. It also damps spatial scales whereas the forward scheme does not (Haltiner and Williams 1980).

The focus now will be to explain how these schemes are implemented within MOM. The following description assumes that the idea of a memory window as outlined in Section 11.3.1 is understood. Refer to Figure 21.1 and note the four columns: *Time Step*, *Type of Time Step*, a partially opened memory window with two disk areas indicated bu column *jmw* < *jmt*, and a fully opened memory window with no disk area indicated by column *jmw=jmt*.

Two sets of indices are required to act as pointers to specific areas on disk and in the memory window: one set points to $\tau - 1$, τ, and $\tau + 1$ locations on disk and these indices are named taum1disk, taudisk, and taup1disk; the other set points to $\tau - 1$, τ, and $\tau + 1$ locations within the memory window and these indices are named taum1, tau, and taup1.

At the beginning of each time step, memory window indices are updated as shown schematically in the *Type of Time Step* column. The whole idea is to get data positioned properly inside the memory window so that the equations can be solved for various types of time steps with only minimal changes[4].

To actually see how disk indices are cycled for various types of time steps, enable option *trace_indices* as described in Chapter 39.

21.4.1 Leapfrog

Consider the partially opened memory window shown in the third column. On leapfrog time steps, the updating arrows indicate that $\tau - 1$ variables in the memory window are being filled with what was τ disk data on the previous time step. Likewise, τ variables in the memory

[4]The only change is whether a time step length is $\Delta\tau$ or $2\Delta\tau$

window are being filled with what was $\tau + 1$ disk data on the previous time step. The equation for a typical prognostic variable h indicates a central difference or leapfrog scheme in time. Note that the forcing term F is a function of τ for advective processes and $\tau - 1$ for diffusive processes. Before the leapfrog step, disk indices are set as follows:

- taum1disk $= mod(itt + 1, 2) + 1$
- tau $= mod(itt, 2) + 1$
- taup1disk = taum1disk

This formulation cyclically exchanges disk pointer indices every time step to assure that the correct disk data is read into and written from the proper memory window locations as described above. Memory window indices are always as follows when the memory window is partially opened:

- taum1 $= 1$
- tau $= 2$
- taup1 $= 3$

When the time step is complete, $\tau + 1$ data is written back over the $\tau - 1$ disk area. Looking to the fourth column, the fully opened memory window has no arrows therefore no movement of data. The disk pointers are not used and instead of being fixed in time, the memory pointers are cyclicly updated according to

- taum1 $= mod(itt + 0, 3) + 1$
- tau $= mod(itt + 1, 3) + 1$
- taup1 $= mod(itt + 2, 3) + 1$

which accomplishes renaming of areas within the memory window without moving data. Comparing time step n with $n + 1$ will clarify this.

21.4.2 Forward

On forward time steps when the memory window is partially opened as indicated by time step n+2, both $\tau - 1$ and τ variables are loaded with what was $\tau + 1$ disk data on the previous time step. In the third column, memory pointers are set to reflect this. The equation is the same as for the leapfrog time steps except that $2\Delta\tau$ is replaced by $\Delta\tau$.

21.4.3 Euler Backward

Euler backward steps are comprised of two half steps. The first is identical to a forward step and the second fills τ variables within the memory window with $\tau + 1$ data from the first step. When the memory window is opened all the way, an *euler shuffle* is required to get data properly aligned in preparation for the next time step. This is the only data movement required when the memory window is fully opened[5].

[5]This can be eliminated by recalculating the pointers differently on the time step after the Euler backward step. However, the calculation gets tricky and has complications.

21.4.4 Robert time filter

The *Robert* time filter (enabled by option *robert_time_filter*) is applied every time step in conjunction with using a leapfrog scheme. Schematically, this amounts to:

$$\overline{h^\tau} = h^\tau + smooth \cdot ((h^{\tau+1} + h^{\tau-1})/2 - h^\tau) \tag{21.63}$$

where $\overline{h^\tau}$ is the filtered value of the prognostic variable h at time level τ and *smooth* is an adjustable parameter which may be set via namelist. The default value is $smooth = 0.01$ which is a reasonably small value. The largest value possible is $smooth = 0.2$ which is a very heavy smoothing. The Robert time filter will give smoother solutions than periodic application of either forward or Euler backward mixing steps. The disadvantage of the Robert time filter is that it requires an extra read through of data on disk. However, if option *ramdrive* is enabled and disk data is actually held in memory, the extra overhead for reading is small. However, for other I/O options where disk data is stored on rotating disk, the overhead may result in a substantial increase in wall clock time.

Caveat: As implemented, the Robert time filter will not work correctly when applied to surface currents for the purpose of Doppler shilting windspeeds when restart files are involved for anything other than option *simple_sbc*. The reason is that on non-restart time steps, unfiltered values of u^τ are used. However, filtered values of u^τ are written to the restart file. Therefore, when re-starting from a restart file, filtered values of u^τ are loaded from disk and used on the first time step of a restart. This is difficult to remedy since the barotropic portion of $u^{\tau+1}$ is unknown at the time the filter is applied. Filtering the tracer fields will preserve answers across restart points. The difference in answers due to this error in velocity is very tiny judging from the observation that if the error is not accounted for in the tracer fields, the differences in surface tracer boundary conditions is the the 6th decimal place.

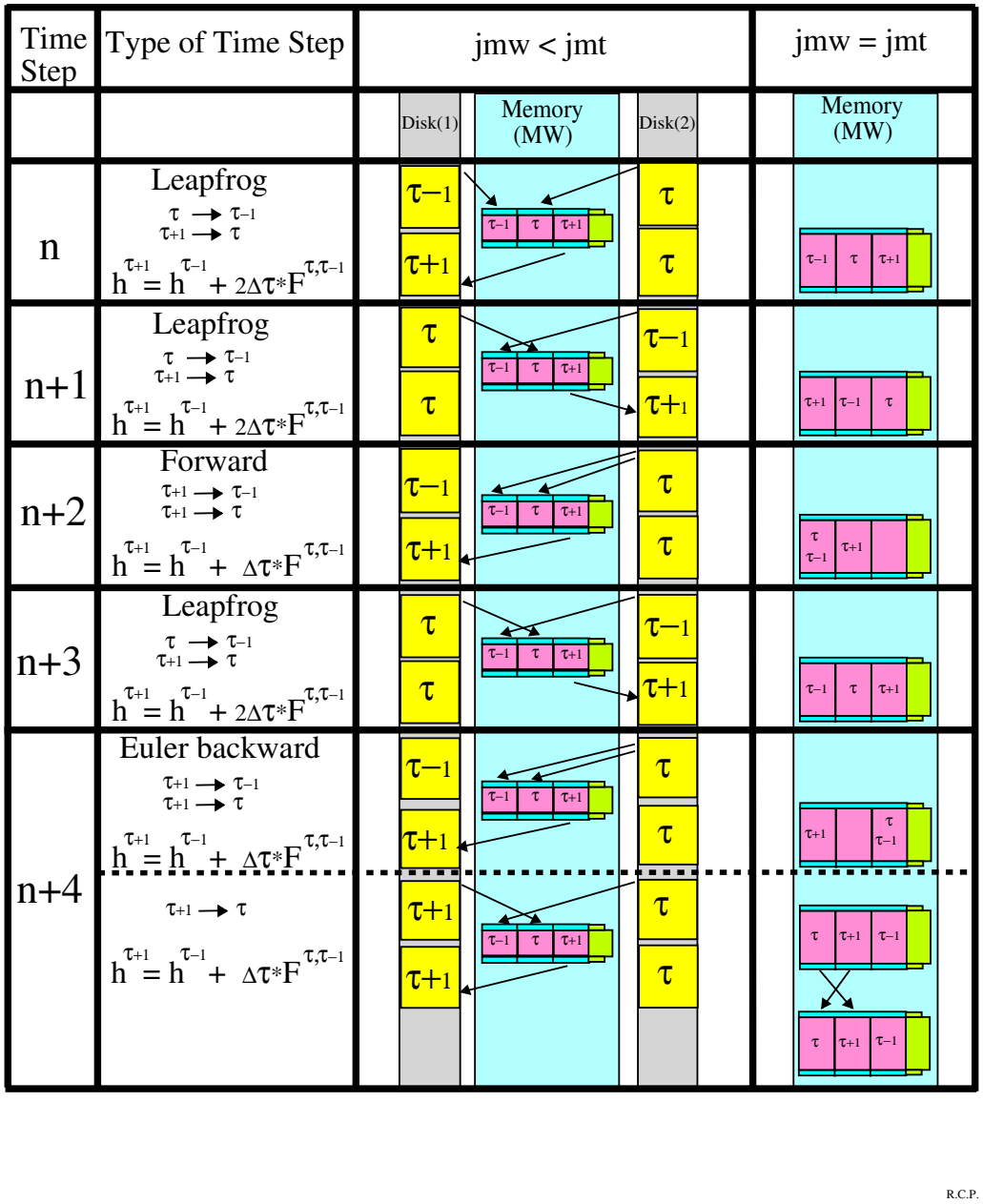


Figure 21.1: Time discretization used in MOM when memory window is partially opened and fully opened. components

Chapter 22

Solving the Discrete equations

The reader will note that all numerical indices are exposed within expressions so as to match exactly with those used in the Fortran code. This extra bit of detail is insisted upon as another way of insuring that complex formulations are correct.

Once surface boundary conditions are available for the ocean as described in Chapter 19, the ocean equations are integrated for one time step with each call to subroutine *mom*[1]. As indicated schematically in Figure 19.2 and described further in Section 19.1, *mom* may need to be called repeatedly until integration has been carried out for one time segment. For a flowchart indicating the calling sequence within *mom* for one time step, refer to Figure 22.1. The process starts by incrementing the ocean time step counter *itt* by one

$$itt = itt + 1 \tag{22.1}$$

and calling module *tmngr*[2] which increments ocean time by the number of seconds in one time step. Refer to Section 14.4.4 for choosing a time step length. Module *tmngr* additionally calculates a time and date as described in Section 15.1.9 and determines which events are to be activated on the current time step and which are not. Each event has an associated logical switch which is kept in file *switch.h*. An event might be something like writing a particular diagnostic for analysis, or doing a mixing time step, etc. After determining all logical switches, a call is made to the diagnostic initialization subroutine *diagi* which performs initializations for various diagnostics when required[3].

22.1 Start of computation within Memory Window

Equations are solved for a group of latitude rows within the memory window. The group may be as few as one[4] or as many as $(jmt - 2)$ latitudes[5]. The number of groups depends on the size of the memory window and each group is solved one at a time until all groups have been solved. The northward movement of the memory window is described in Section 11.3.1. The reader is referred to Chapter 14 for a description of the variables, Chapter 16 for a description of the grid system, and Chapter 11 for a description of the memory window.

[1]Contained in file *mom.F.*
[2]Contained in file tmngr.F.
[3]Assuming these diagnostics have been enabled by their options at compile time.
[4]Requiring three rows in the memory window.
[5]Requiring *jmt* rows in the memory window.

The number of memory window moves needed to solve all latitude rows is controlled by a "do loop" which essentially extends from Section 22.2 to Section 22.10. The subroutines called for each group of latitudes within the memory window are outlined in Figure 22.1 and explained in the following sections.

22.2 loadmw (load the memory window)

Subroutine *loadmw*[6] begins by moving the memory window northward if necessary[7]. This movement begins by copying data from the top two rows into the bottom two rows of the memory window as described in Section 11.3.2.

22.2.1 Land/Sea masks

At this point, land/sea masks $tmask_{i,k,j}$ for the T-cells and $umask_{i,k,j}$ for the U-cells are constructed from $kmt_{i,jrow}$ and $kmu_{i,jrow}$. They are set as

$$tmask_{i,k,j} = \begin{cases} 1.0 & \text{if cell } T_{i,k,j} \text{ is an ocean cell} \\ 0.0 & \text{if cell } T_{i,k,j} \text{ is a land cell} \end{cases} \tag{22.2}$$

and

$$umask_{i,k,j} = \begin{cases} 1.0 & \text{if cell } U_{i,k,j} \text{ is an ocean cell} \\ 0.0 & \text{if cell } U_{i,k,j} \text{ is a land cell} \end{cases} \tag{22.3}$$

The purpose of these masks is to promote vectorization by allowing computation everywhere but zeroing out the results within land cells. Note that when the memory window is fully opened ($jmw = jmt$), this calculation is only done once per run instead of redundantly every time step.

22.2.2 Reading latitude rows into the Memory window

After building the land/sea vectorization masks, the next latitude rows are read from disk to fill the remaining latitude slots within the memory window. Latitude rows are read from the τ and $\tau - 1$ disks. Time level indices depend on whether it is a normal leapfrog time step or a mixing time step as described in Section 21.4.

22.2.3 Constructing the total velocity

Once all data is in place within the memory window, preliminary calculations are ready to begin. The process starts by adding the external mode velocity to the internal mode velocity to construct the total velocity as in Equations (4.1) and (4.2). The finite difference counterpart to the external mode given in Equations (6.4) and (6.5) is

[6] Contained in file loadmw.F.

[7] It is necessary for all but the first group of latitudes. If the memory window is opened all the way (when $jmw = jmt$), then no movement is necessary.

$$\overline{u}_{i,jrow,1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau}}^\lambda) \tag{22.4}$$

$$\overline{u}_{i,jrow,2} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau}}^\phi) \tag{22.5}$$

The total velocity needs to be constructed for both time levels at this point because only the internal mode velocities[8] are kept in the latitude rows on disk.

$$u_{i,k,j,1,\tau-1} = \hat{u}_{i,k,j,1,\tau-1} - \frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau-1}}^\lambda) \tag{22.6}$$

$$u_{i,k,j,2,\tau-1} = \hat{u}_{i,k,j,2,\tau-1} + \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau-1}}^\phi) \tag{22.7}$$

$$u_{i,k,j,1,\tau} = \hat{u}_{i,k,j,1,\tau} - \frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau}}^\lambda) \tag{22.8}$$

$$u_{i,k,j,2,\tau} = \hat{u}_{i,k,j,2,\tau} + \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau}}^\phi) \tag{22.9}$$

where $\hat{u}$ is the internal mode velocity, ψ is the stream function, and H is the ocean depth defined over U cells by Equation (18.4).

Finally, the density $rho_{i,k,j}$ is computed by Equation (15.7) as a function of $t_{i,k,j,1,\tau}$, $t_{i,k,j,2,\tau}$, and depth of level k based on a third order polynomial fit to the equation of state for sea water. The density is actually a deviation from a reference density ρ_k^{ref} as described in Section 15.1.2.

22.2.4 Computing quantities within the memory window

The memory window (MW) is fundamental to solving 3-D baroclinic and tracer equations within MOM. It is not used for solving the 2-D barotropic equation. For a description of the MW, refer to Section 11.3. Figure 22.2 illustrates the horizontal grid within a minimum sized MW and shows sites where prognostic quantities are defined as well as sites populated by derived quantities. Recall that equations for prognostic quantities are solved only on row $j = 2$ within this minimum sized MW. Prognostic quantities on rows $j = 1$ and $j = 3$ are used only to supply access to neighboring cells required by the numerics for solving prognostic equations on row $j = 2$.

Prognostic quantities within grid cells such as temperature, salinity, and horizontal velocity components are defined at grid points within T-cells and U-cells on rows $j = \{1, 2, 3\}$ and therefore dimensioned by the size of the MW as in the following array:

```
dimension A(imt,km,jmw)    ! all cells within the memory window
```

Advective velocity on the north face of T-cells "$adv_vnt(i, k, j)$" is a derived quantity. It is defined for T-cells on rows $j = \{1, 2, 3\}$ and computed as a zonal average of meridional velocity

[8]After each group of internal mode velocities are solved within a time step, they are written back to disk. The external mode is unknown until all rows have been solved and subroutine *barotropic* solves for the external mode. The only way to get the total velocity on disk is to read all rows back into memory after the external mode has been solved. This is why the external mode is added at the beginning of each time step for time levels τ and $\tau - 1$.

components which are defined on U-cell grid points for rows $j = \{1, 2, 3\}$. Since all components for the computation exist within the MW, "$adv_vnt(i, k, j)$", must also be dimensioned as above.

Note that some derived quantities cannot occupy certain sites and these are indicated by a circle with at line across it. For instance, neither the advective velocity on the east face of T-cells for $j = 1$ nor the advective velocity on the south face of U-cells for $j = 1$ can be defined because their computation references quantities outside the MW. As a more detailed example, consider derived quantities such as the advective flux through the northern face of a T-cell "$adv_fn(i, k, j)$" which involve the product of an advective velocity on the northern T-cell face "$adv_vnt(i, k, j)$" and the average temperature on that cell face. Since the average temperature on the northern face of cell "j" is

$$\overline{T_{i,k,j}}^{\phi} = (T_{i,k,j} + T_{i,k,j+1})/2 \tag{22.10}$$

the advective flux can only be dimensioned as $adv_fn(imt, km, jmw - 1)$ otherwise the flux at index $j = jmw$ would require knowledge of temperature at $j = jmw + 1$ which is unknown because it is outside of the MW. Apart from quantities like density and advective velocity, other quantities such as Richardson number, mixing coefficients, etc. may be required by enabled options. In general, placement of derived quantities within the grid system is determined soley by numerics within finite difference equations.

The rule is simple. Any latitude row for which a quantitiy can be calculated should be populated by the calculation. Furthermore, the number of rows that can be populated within the memory window determines the dimension of the quantity. As the memory window is moved northward, all prognostic as well as derived quantities are copied from the northernmost rows into the southernmost rows (as described further in Section 11.3.3) so that no redundant computations are required. All arrays dimensioned within the memory window must have their latitude dimension fall into one of the following four catagories:

```
dimension A(,,jmw)          ! all cells within the memory window
dimension B(,,1:jmw-1)      ! all cells except j=jmw
dimension C(,,2:jmw)        ! all cells except j=1
dimension D(,,2:jmw-1)      ! all cells except j=1 and j=jmw
```

A few more examples follow:

22.2.4.1 Example 1: density

Density $\rho_{i,k,j}$ is defined at each grid point within T-cells by Equation (15.7). It is a function of temperature, salinity, and pressure (or equivalently depth) without regard to spatial gradients of temperature or salinity and so should be dimensioned as

```
dimension rho(imt,km,jmw)
```

The computation of density can therefore populate T-cells within rows "j=1" through "j=jmw" in the memory window. There is no need for redundant computations of "rho" since as the memory window is moved northward, rows within array "rho" are cycled by the prescription given in Section 11.3.3.

22.2.4.2 Example 2: Advective velocity on the eastern face of T-cells

The advective velocity on the eastern face of T-cells "$adv_vet_{i,k,j}$" is defined by Equation (22.11). Note that it cannot be computed for the first latitude row "j=1" in the memory window. Why? Because Equation (22.11) references zonal velocity at "j-1" which is outside of the dimensional bounds on zonal velocity and therefore outside the bounds of the memory window. However, "$adv_vet_{i,k,j}$" can be computed for rows "j=2" through "j=jmw" in the memory window. Therefore, it is dimensioned as

```
dimension adv_vet(imt,km,2:jmw)
```

and its computation must populate rows "2" through "jmw" in the memory window. Once computed, rows within array "*adv_vet*" are cycled according to the prescription given in Section 11.3.3 so that no redundant computations of advective velocity are involved as the window moves northward.

22.2.4.3 Example 3: Advective velocity on the bottom face of U-cells

Advective velocity on the bottom face of U-cells "$adv_vbu_{i,k,j}$" is defined as the vertically integrated divergence of horizontal advective velocities (which are on the faces of U-cells) by Equation (22.23). Inspection of the indices in the numerics indicates that "$adv_vbu_{i,k,j}$" cannot be computed within the first or last row of the memory window because this would require knowledge of quantities outside of the memory window. Therefore, "$adv_vbu_{i,k,j}$" can only be dimensioned as

```
dimension adv_vbu(imt,km,2:jmw)
```

22.3 adv_vel (computes advective velocities)

Subroutine *adv_vel*[9] constructs advective velocities on the north, east and bottom faces of tracer cells $T_{i,k,j}$ within the memory window. These advective velocities are then suitably averaged to construct advective velocities on the north, east and bottom faces of velocity cells $U_{i,k,j}$. Both sets of advective velocities will now be described.

22.3.1 Advective velocities for T cells

Advective velocities are defined in a direction normal to T-cell faces. Refer to Figure 22.3a which illustrates a T-cell with all six advective velocities and their indices. Note that a T grid point is within the T-cell and there are four surrounding U points; one on each corner of the T-cell. A horizontal slice through the plane containing the grid points is depicted in Figure 22.3c showing the relation between indices in the central T-cell and four surrounding U-cells. Advective velocity on the eastern face of a T cell is a meridional weighted average of the zonal velocities at the vertices of the face. A similar relation holds for the advective velocity on the northern face of a T cell but it involves a zonal average of the meridional velocities. In both cases the averaging takes place within the plane of the face.

[9] Contained within file adv_vel.F.

$$adv_vet_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot dyu_{jrow} + u_{i,k,j-1,1,\tau} \cdot dyu_{jrow-1}}{2 \cdot dyt_{jrow}} \tag{22.11}$$

$$adv_vnt_{i,k,j} = \cos\phi^{U}_{jrow} \frac{u_{i,k,j,2,\tau} \cdot dxu_i + u_{i-1,k,j,2,\tau} \cdot dxu_{i-1}}{2 \cdot dxt_i} \tag{22.12}$$

This form of the advective velocities is not arbitrary and comes from the condition that the work done by horizontal pressure forces must equal the work done by buoyancy. The details are given in Section A.2.4. Note that with non-uniform resolution, the denominator is not equal to the sum of the weights if the grid is constructed as:

$$dxu_i = \frac{dxt_i + dxt_{i+1}}{2} \tag{22.13}$$

$$dyu_{jrow} = \frac{dyt_{jrow} + dyt_{jrow+1}}{2} \tag{22.14}$$

which was done in MOM 1 and earlier implementations[10]. However, MOM allows grid construction[11] by

$$dxt_i = \frac{dxu_i + dxu_{i-1}}{2} \tag{22.15}$$

$$dyt_{jrow} = \frac{dyu_{jrow} + dyu_{jrow-1}}{2} \tag{22.16}$$

which guarantees that the denominator always equals the sum of the weighting factors. This is presumed to allow more accurate advective velocities when the grid is stretched in the horizontal although differences should be of second order. Refer to Chapter 16 for a further discussion on non-uniform grids and advection.

From the incompressibility condition expressed by Equation (4.3), the advective velocity on the bottom face of each cell is defined as the vertical integral of the convergence of the horizontal advective velocities on each cell face from the surface down to the bottom face of any cell. The rigid lid assumption sets the advective velocity at the top face of the first cell to zero. If option *implicit_free_surface* is enabled, the advective velocity at the top face of the first cell is diagnosed from $\frac{1}{rho_\circ \cdot grav} \cdot \delta_\tau(ps_{i,jrow})$. If the option *explicit_free_surface* is enabled, then the vertical velocity is diagnosed as the convergence of the vertically integrated transport. As discussed in Section 7.2.3, this convergence includes both the time tendency of the free surface height as well as the input of fresh water.

For points within the ocean, the vertical advection velocity is calculated diagnostically as the advective velocity at the bottom face of each cell through the expression

$$adv_vbt_{i,k,j} = \frac{1}{\cos\phi^{T}_{jrow}} \cdot \sum_{m=1}^{k} \left(\delta_\lambda(adv_vet_{i-1,m,j}) + \delta_\phi(adv_vnt_{i,m,j-1}) \right) \cdot dzt_m$$

[10]This was the motivation for exploring grid construction by method 2 as detailed in Section 16.2.3

[11]The old construction method 1 is enabled by option *centered_t*. Refer to Chapter 16 for a description of grid design.

$$= \frac{1}{\cos\phi^T_{jrow}} \cdot \sum_{m=1}^{k} \left(\frac{adv_vet_{i,m,j} - adv_vet_{i-1,m,j}}{dxt_i} + \frac{adv_vnt_{i,m,j} - adv_vnt_{i,m,j-1}}{dyt_{jrow}} \right) \cdot dzt_m \tag{22.17}$$

where ϕ^T_{jrow} refers to the latitude at point $T_{i,k,j}$.

22.3.2 Advective velocities for U cells

Another set of advective velocities are defined in a direction normal to U-cell faces. Refer to Figure 22.3b which illustrates a U-cell with all six advective velocities and their indices. Note that a U grid point is within the U-cell and there are four surrounding T points; one at each corner of the U-cell. A horizontal slice through the plane containing the grid points is depicted in Figure 22.3d showing the relation between indices in the central U-cell and four surrounding T-cells. To calculate advective velocities on faces of U cells, the averaging approach indicated by Dukowicz and Smith (1994) and subsequently described by Webb (1995) is used. Their formulae are:

$$adv_veu_{i,k,j} = \overline{adv_vet_{i,k,j}}^{\lambda\phi} \tag{22.18}$$

$$adv_vnu_{i,k,j} = \overline{adv_vnt_{i,k,j}}^{\lambda\phi} \tag{22.19}$$

$$adv_vbu_{i,k,j} = \overline{adv_vbt_{i,k,j}}^{\lambda\phi} \tag{22.20}$$

The above averaging works when the grid resolution is uniform. For MOM, a more general form is used to conserve volume within each U cell when resolution is non-uniform. Volume conservation is insured by taking a weighted average of T cell advective velocities within the plane of the cell face and a linear interpolation in a direction normal to the cell face. Refer to Figure 22.3c. Using the indicated grid distances, the resulting four point averaging operators are given by:

$$\begin{aligned} adv_vnu_{i,k,j} &= \frac{1}{dyt_{jrow+1} \cdot dxu_i} \times \\ &\Big[(adv_vnt_{i,k,j} \cdot duw_i + adv_vnt_{i+1,k,j} \cdot due_i) \cdot dus_{jrow+1} \\ &+ (adv_vnt_{i,k,j+1} \cdot duw_i + adv_vnt_{i+1,k,j+1} \cdot due_i) \cdot dun_{jrow} \Big] \end{aligned} \tag{22.21}$$

$$\begin{aligned} adv_veu_{i,k,j} &= \frac{1}{dyu_{jrow} \cdot dxt_{i+1}} \cdot \times \\ &\Big[(adv_vet_{i,k,j} \cdot dus_{jrow} + adv_vet_{i,k,j+1} \cdot dun_{jrow}) \cdot duw_{i+1} \\ &+ (adv_vet_{i+1,k,j} \cdot dus_{jrow} + adv_vet_{i+1,k,j+1} \cdot dun_{jrow}) \cdot due_i. \Big] \end{aligned} \tag{22.22}$$

The vertical velocity at the bottom face of U cells can be calculated by directly integrating the continuity equation

$$adv_vbu_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}} \cdot \sum_{m=1}^{k} \left(\delta_\lambda(adv_veu_{i-1,m,j}) + \delta_\phi(adv_vnu_{i,m,j-1}) \right) \cdot dzt_m \quad (22.23)$$

Equivalently, the vertical velocity at the bottom face of U cells can by calculated as an area preserving weighted average of the vertical velocities at the bottom face of four surrounding T cells.

$$\begin{aligned} adv_vbu_{i,k,j} &= \frac{1}{dyu_{jrow} \cdot dxu_i \cdot \cos\phi^U_{jrow}} \times \\ &\Big[\quad adv_vbt_{i,k,j} \cdot dus_{jrow} \cdot duw_i \cdot \cos\phi^T_{jrow} \\ &+ \quad adv_vbt_{i+1,k,j} \cdot dus_{jrow} \cdot due_i \cdot \cos\phi^T_{jrow} \\ &+ \quad adv_vbt_{i,k,j+1} \cdot dun_{jrow} \cdot duw_i \cdot \cos\phi^T_{jrow+1} \\ &+ \quad adv_vbt_{i+1,k,j+1} \cdot dun_{jrow} \cdot due_i \cdot \cos\phi^T_{jrow+1} \Big] \end{aligned} \quad (22.24)$$

This volume conserving averaging operation eliminates the problem of numerical de-coupling between advective velocities on the bottom of T cells and U cells in the presence of flow over topographic gradients. As discussed by Webb (1995), this de-coupling can introduce a large mis-match between the vertical velocity on T-cells and U-cells, with the U-cell velocity generally very noisy near topography. Implementations of vertical velocity in MOM 1 and previous versions of the GFDL ocean model suffered from such de-coupling and noise. It should be noted that the weighting factors *dus*, *dun*, *due*, and *duw* are always equal if resolution is uniform. In this case, the averaging used here reduces to that of Webb (1995).

Refer to Sections 22.9.5 and 22.8.7 for details on how advective velocities are used to compute the advective fluxes of momentum and tracers.

22.3.3 Vertical velocity on the ocean bottom

Before tracer or momentum equations can be solved, it is necessary to evaluate the vertical velocity at the ocean surface and bottom. More specifically, as discussed in Sections 22.3.1 and 22.3.2, vertical velocity at the ocean surface and bottom enters the continuity equation as an upper and lower boundary condition. Since the continuity equation is a first order differential constraint, only one boundary condition is needed to diagnose the vertical velocity $w(z)$. Either $w(0)$ or $w(-H)$ may be used. For historical reasons based on the early use of the rigid lid approximation, MOM uses vertical velocity at the ocean surface and integrates downward to construct $w(z)$ at all depths. The vertical velocity at the ocean bottom will now be given for the discrete equations. The result for integrating from the bottom up will also be given.

22.3.3.1 Summary of the continuum results

Before discussing the discrete results, it is useful to first recall the discussion of the vertical velocity within the context of the continuum equations as given in Chapter 7. The main result is that the vertical velocity at the ocean surface is given by (equation 7.39)

$$w(0) = -\nabla_h \cdot \mathbf{U}_0, \quad (22.25)$$

which in words means that $w(z=0)$ is due to the convergence of the vertically integrated velocity

$$\mathbf{U}_0 = \int_{-H}^{0} dz\, \mathbf{u}_h. \tag{22.26}$$

This result is general; i.e., it holds even when there is fresh water input to the free surface. For the rigid lid, there is identically zero convergence and so $w(0) = 0$. For the implicit free surface, only the case without fresh water is implemented in which $w(0) \approx \eta_t$ is assumed. For the explicit free surface, $w(0) = -\nabla_h \cdot \mathbf{U}_0$ is implemented. For the ocean bottom, recall the discussion from Section 4.3.1, in which it was shown that the vertical velocity at the ocean bottom is given through the kinematic boundary condition

$$w \;=\; -\mathbf{u}_h \cdot \nabla_h H \qquad z = -H(\lambda, \phi). \tag{22.27}$$

Each of these results will now be discussed within the context of the discrete ocean model in which nontrivial bottom topography is allowed.

22.3.3.2 Discrete vertical velocity at the ocean bottom

As described in Chapter 18, the discretized ocean bottom is defined by land T-cells beneath the deepest ocean T-cells. The deepest ocean T-cells are given by $T_{i,kb,j}$ where $kb = kmt_{i,jrow}$ and

$$2 \le kb \le km\ . \tag{22.28}$$

Refer to Fig 22.4a which illustrates the ocean bottom using land T-cells (the land cells are drawn with solid lines). The bottom face of the deepest ocean T-cells[12] is marked with an "x". In total, all exposed faces of these land T-cells define the material surface across which no flow can pass (i.e. normal velocity component at the material surface is zero). To keep the figure simple, only one vertical column of T-cells extending from the ocean surface to the ocean bottom is illustrated (the ocean T-cells are drawn with dashed lines). Let the location of this vertical column be given by coordinate indices "i,j" where cell $T_{i,kb,j}$ is the deepest ocean T-cell in the column and the bottom face of cell $T_{i,kb,j}$ is marked with an "x".

Vertical velocity at the bottom face of any ocean T-cell can be found by vertically integrating as in Equaton (22.17) from the resting ocean surface $z = 0$ to the bottom face of the cell in question. Vertical velocity on the bottom face of cell $T_{i,kb,j}$ (marked by "x") is zero because bottom faces are horizontally flat and the normal velocity component on a material surface which is fixed in time is zero. That is, the model's bottom topography is piece-wise flat on the T-cell grid and so the vertical velocity must vanish there. Hence, the vertical component to the advective velocity on the T-cell vanishes at the bottom of the T-cell grid

$$\begin{aligned} adv_vbt_{i,kb,j} &= adv_vbt_{i,0,j} + \frac{1}{\cos\phi^T_{jrow}} \sum_{k=1}^{kb} (\delta_\lambda(adv_vet_{i-1,k,j}) + \delta_\phi(adv_vnt_{i,k,j-1})) \cdot dzt_k \\ &= 0. \end{aligned} \tag{22.29}$$

The vertical advection velocity at the ocean surface $adv_vbt_{i,0,j}$ vanishes when using a rigid lid option, but is generally nonzero if using a free surface option. Similarly, vertical velocity is zero on all other bottom T-cell faces marked by "x".

[12] This cell face is also the top face of the first land T-cell below the ocean bottom.

Now consider the "black dot" in Figure 22.4a. This dot marks the bottom face of the deepest ocean U-cell having coordinate indices "i,j". The vertical column of U-cells extending from the ocean surface to the deepest ocean cell $U_{i,kbu,j}$ is indicated in Fig 22.4b (the ocean U-cells are drawn with dashed lines). The depth index "kbu" is calculated as the minimum of the four surrounding T-cell depth indices:

$$kbu = kmu_{i,jrow} = min(kmt_{i,jrow},\ kmt_{i,jrow+1},\ kmt_{i+1,jrow},\ kmt_{i+1,jrow+1}) \tag{22.30}$$

Importantly, note that the bottom face of the U-cell $U_{i,kbu,j}$ is *not a material surface*. This fact is evident in Figure 22.4c which illustrates partial ocean U-cells existing below the bottom face of cell $U_{i,kbu,j}$ (the U-cells are drawn with dashed lines). These ocean U-cells are only partially full grid cells since they are truncated by the material surfaces of the surrounding land T-cells.

Since the bottom face of cell $U_{i,kbu,j}$ is not a material surface, it follows that the vertical velocity at the "black dot" in Figure 22.4a is not generally zero

$$adv_vbu_{i,kbu,j} \neq 0. \tag{22.31}$$

Instead, using Equation (22.17) to vertically integrate from the surface downwards to the "black dot" in Figure 22.4b yields

$$adv_vbu_{i,kbu,j} \quad = \quad \overline{adv_vbt_{i,0,j}}^{\lambda\phi} + \frac{1}{\cos\phi^{U}_{jrow}} \sum_{m=1}^{kbu} \left(\delta_\lambda(adv_veu_{i-1,m,j}) + \delta_\phi(adv_vnu_{i,m,j-1}) \right) \cdot dzt_m \tag{22.32}$$

The above considerations always started from the ocean surface. Volume conservation ensures that one can equivalently start from the bottom of the U-cells and integrate upwards. For this purpose, it is necessary to define the deepest partially full cell $U_{i,kmax,j}$ as shown in Figure 22.4c. The depth index "kmax" for this cell is calculated as the maximum of the four surrounding T-cell depth indices

$$kmax = max(kmt_{i,jrow},\ kmt_{i,jrow+1},\ kmt_{i+1,jrow},\ kmt_{i+1,jrow+1}) \tag{22.33}$$

The vertical advective velocity $adv_vbu_{i,kmax,j}$ vanishes since it represents the vertical velocity at the flat material surface which is the bottom of a U-cell column. Integrating Equation (22.17) upwards from the bottom of the U-cell column at *kmax* to the "black dot " at *kbu* yields

$$adv_vbu_{i,kbu,j} \quad = \quad -\frac{1}{\cos\phi^{U}_{jrow}} \sum_{m=kbu+1}^{kmax} \left(\delta_\lambda(adv_veu_{i-1,m,j}) + \delta_\phi(adv_vnu_{i,m,j-1}) \right) \cdot dzt_m. \tag{22.34}$$

Volume conservation ensures that this result is identical to equation (22.32).

The question now arises as to why Equation (22.34) does not have the appearance of a discretized version of Equation (22.27)? The answer is that Equation (22.27) should be discretized at the location of the "open circle" in Figure 22.4 rather than at the "black dot" itself. The reason is that location of the "open circle" is determined as the maximum distance above the bottom face of cell $U_{i,kmax,j}$ for which *both* the northern and western cell faces are material surfaces.

For purposes of completeness, the discrete form of Equation (22.27) can be written for the location of the "open circle" through the following considerations. Let "ko" be the vertical

index of the U-cell whose top face contains the "open circle". Also, let the height of the "open circle" from the base of cell $U_{i,kmax,j}$ be given by

$$H^b = zw(kmax) - zw(ko). \tag{22.35}$$

Integrating upwards from the bottom face of level "kmax" to "ko" yields

$$adv_vbu_{i,ko,j} = -\frac{1}{\cos\phi^U_{jrow}} \sum_{m=ko}^{kmax} \left(\delta_\lambda(adv_veu_{i-1,m,j}) + \delta_\phi(adv_vnu_{i,m,j-1}) \right) \cdot dzt_m \tag{22.36}$$

Defining the average advective velocity normal to cell faces as

$$U^b_{i,j} = \frac{1}{H^b} \sum_{m=ko}^{kmax} adv_veu_{i,m,j}\, dzt_m \tag{22.37}$$

$$V^b_{i,j} = \frac{1}{H^b} \sum_{m=ko}^{kmax} adv_vnu_{i,m,j}\, dzt_m \tag{22.38}$$

and substituting into Equation (22.36) yields

$$adv_vbu_{i,ko,j} = -\frac{1}{\cos\phi^U_{jrow}} \left(H^b \delta_\lambda(U^b_{i-1,j}) + H^b \delta_\phi(V^b_{i,j-1}) \right) \tag{22.39}$$

Since the north and west face of the cells summed over in Equation (22.36) are material surfaces

$$U^b_{i-1,j} = 0 \tag{22.40}$$

$$V^b_{i,j} = 0 \tag{22.41}$$

which leads to the finite difference counterpart of Equation (22.27).

$$\begin{aligned} adv_vbu_{i,ko,j} &= \frac{1}{\cos\phi^U_{jrow}} \left(-U^b_{i,j} \frac{H^b}{dxu_i} + V^b_{i,j-1} \frac{H^b}{dyu_{jrow}} \right) \\ &= \frac{1}{\cos\phi^U_{jrow}} \left(U^b_{i,j} \cdot \delta_\lambda(H^b) + V^b_{i,j-1} \cdot \delta_\phi(H^b) \right). \end{aligned} \tag{22.42}$$

22.4 isopyc (computes isoneutral mixing tensor components)

Subroutine *isopyc*[13] computes the isoneutral mixing tensor components. Refer to Sections 35.1.5, 35.1.6, and 35.1.8 for the details.

22.5 vmixc (computes vertical mixing coefficients)

Subroutine *vmixc*[14] computes vertical mixing coefficients for momentum ($diff_cbu_{i,k,j}$) <u>and</u> tracers ($diff_cbt_{i,k,j}$) according to the scheme selected by options at compile time. Subscripts for the mixing coefficients depend on which option was enabled. Options are *constvmix* for constant

[13]Contained in file isopyc.F.
[14]Contained in file vmixc.F.

vertical mixing coefficients, *ppvmix* for Richardson dependent vertical mixing coefficients as in Pacanowski/Philander (1981), and *tcvmix* for the second order turbulence closure of Mellor/Yamada as given in Rosati and Miyakoda (1988). However, *tcvmix* is not available as of this writing but is being implemented by Rosati at GFDL. One of these must be enabled and all adjustable values are input through namelist. Refer to Section 14.4 for information on namelist variables.

Additionally, there are two hybrid options allowed which supply mixing coefficients in the vertical for tracers but not momentum. Option *isoneutral* is for mixing of tracers according to the structure of the isoneutral directions, as described in Section 35.1, and option *bryan_lewis_vertical* (Bryan/Lewis 1979) sets constant values as a function of depth for the vertical mixing coefficient. When option *isoneutralmix* is enabled, mixing coefficients for tracers from other options such as *constvmix*, *ppvmix*, or *bryan_lewis_vertical* are used as background values and added to the vertical diffusion coefficient[15] from option *isoneutralmix*.

22.6 hmixc (computes horizontal mixing coefficients)

Subroutine *hmixc*[16] computes horizontal mixing coefficients for momentum ($diff_ceu_{i,k,j}$ and $diff_cnu_{i,k,j}$) and tracers ($diff_cet_{i,k,j}$ and $diff_cnt_{i,k,j}$) according to the scheme selected by options at compile time. Currently, options include *consthmix* which use constant horizontal mixing coefficients appropriate for ∇^2 mixing, *biharmonic* which use constant mixing coefficients appropriate for higher order ∇^4 mixing, and *smagnlmix* for Smagorinsky nonlinear mixing coefficients after Smagorinsky (1963).

As in Section 22.5, coefficients from hybrid schemes for mixing of tracers but not mixing of momentum are also allowed. They are *bryan_lewis_horizontal*, *isoneutralmix* and *held_larichev* as referred to in Section 22.5. When option *isoneutralmix* is enabled, mixing coefficients for tracers from *consthmix*, *smagnlmix*, or *bryan_lewis_vertical* are used as background values and added to the horizontal diffusion coefficients[17] from option *isoneutralmix*. For details on specific schemes, consult their options.

22.7 setvbc (set vertical boundary conditions)

Subroutine *setvbc*[18] sets vertical boundary conditions for momentum $smf_{i,j,n}$ and tracers $stf_{i,j,n}$ at the ocean surface and their counterparts defined at the ocean bottom $bmf_{i,j,n}$ and $btf_{i,j,n}$. Also, $bmf_{i,j,n}$ can be set to zero (free slip) or a linear bottom drag condition with the drag coefficient being input through namelist. Refer to Section 14.4 for information on namelist variables. Refer to Chapter 19 for further discussion on surface boundary conditions.

22.8 tracer (computes tracers)

Subroutine *tracer* is contained in file tracer.F and computes tracers given by $t_{i,k,j,n,\tau+1}$ for $n = 1$ to nt, where nt is the number of tracers. Index $n = 1$ corresponds to potential temperature

[15] This is effectively done by adding diffusive fluxes from the basic scheme to the isoneutral flux across the bottom face of the T cell.

[16] Contained in file hmixc.F.

[17] This is effectively done by adding diffusive fluxes from the basic scheme to the isoneutral flux across the northern and eastern face of the T cell.

[18] Contained in file setvbc.F.

in Equation (4.5) and index $n = 2$ is for salinity in Equation (4.6). These first two tracers are dynamically active in the sense that they determine density by Equation (15.7). Other tracers may be added by setting parameter $nt > 2$ but the additional tracers are passive. In previous versions of the model, tracers were solved after the baroclinic velocities. However, in order to accommodate option *pressure_gradient_average* tracers are now computed before baroclinic velocities.

22.8.1 Tracer components

Each tracer is considered as a component of one tracer equation and solved separately by a "do $n = 1, nt$" loop extending to Section 22.8.10. As with the finite difference momentum equations given in Section 22.9.5, the finite difference tracer equation is written in flux form to conserve first and second moments as discussed in Chapter A.

22.8.2 Advective and Diffusive fluxes

Advective and diffusive fluxes across the north, east, and bottom faces of T cells within the memory window are calculated for use by the advection and diffusion operators described in Sections 21.3.2 and 22.8.7. The canonical forms of the fluxes are given in Section 21.3.2 and expanded below. The actual form of the diffusive flux may differ depending on which option is enabled. Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in adv_fe) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. Also, a cosine factor has been absorbed into the definition of $adv_vnt_{i,k,j}$ for reasons of speed. Fluxes are typically computed from $j = 2$ through $j = jmw - 1$, except for the last memory window, which may only be partially full. Meridional operators require meridional flux to be computed for $j = 1$, which is why the lower limit of the j index is not identical for the flux across the north, east, and bottom cell faces. Refer to Figure 11.2 for a schematic of the memory window to help clarifiy the latitude indexing limits.

$$adv_fe_{i,k,j} = adv_vet_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}) \;, j = 2, jmw - 1 \tag{22.43}$$

$$adv_fn_{i,k,j} = adv_vnt_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau}) \;, j = 1, jmw - 1 \tag{22.44}$$

$$adv_fb_{i,k,j} = adv_vbt_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau}) \;, j = 2, jmw - 1 \tag{22.45}$$

$$diff_fe_{i,k,j} = diff_cet_{i,k,j} \cdot \frac{t_{i+1,k,j,n,\tau-1} - t_{i,k,j,n,\tau-1}}{\cos\phi^{T}_{jrow} dxu_i} \;, j = 2, jmw - 1 \tag{22.46}$$

$$diff_fn_{i,k,j} = diff_cnt_{i,k,j} \cdot \cos\phi^{U}_{jrow} \frac{t_{i,k,j+1,n,\tau-1} - t_{i,k,j,n,\tau-1}}{dyu_{jrow}} \;, j = 1, jmw - 1 \tag{22.47}$$

$$diff_fb_{i,k,j} = diff_cbt_{i,k,j} \cdot \frac{t_{i,k,j,n,\tau-1} - t_{i,k+1,j,n,\tau-1}}{dzw_k} \;, j = 2, jmw - 1 \tag{22.48}$$

At the top and bottom, boundary conditions are applied to the vertical fluxes where the bottom cell is at level $kb = kmt_{i,jrow}$. Also, the advective flux through the bottom of the last level

$kb = kmt_{i,jrow}$ is zero[19]. Note that the advective flux through the bottom of the last vertical cell $k = km$ is set to zero. Also, when option *stream_function* is enabled, $adv_vbt_{i,0,j} = 0$ because of the rigid lid but $adv_vbt_{i,0,j} \neq 0$ when option *implicit_free_surface* is enabled.

$$kb = kmt_{i,jrow} \tag{22.49}$$
$$diff_fb_{i,0,j} = stf_{i,j,n} \tag{22.50}$$
$$diff_fb_{i,kb,j} = btf_{i,j,n} \tag{22.51}$$
$$adv_fb_{i,0,j} = adv_vbt_{i,0,j} \cdot (t_{i,1,j,n,\tau} + t_{i,1,j,n,\tau}) \tag{22.52}$$
$$adv_fb_{i,km,j} = 0.0 \tag{22.53}$$

22.8.3 Isoneutral fluxes

When option *isoneutralmix* is enabled, the horizontal components of the isoneutral diffusive flux in Section 35.1 are added to the horizontal diffusive flux in Equations (22.46) and (22.47). The complete diffusive flux across the northern and eastern face of a T cell then is given by

$$diff_fe_{i,k,j} = diff_fe^{old}_{i,k,j} + diff_fet^{iso}_{i,k,j} \tag{22.54}$$
$$diff_fn_{i,k,j} = diff_fn^{old}_{i,k,j} + diff_fnt^{iso}_{i,k,j} \tag{22.55}$$

where the isoneutral diffusion fluxes $diff_fet^{iso}_{i,k,j}$ and $diff_fnt^{iso}_{i,k,j}$ are given in Section 35.1. The background fluxes are given by

$$diff_fe^{old}_{i,k,j} = diff_cet_{i,k,j} \cdot \frac{t_{i+1,k,j,n,\tau-1} - t_{i,k,j,n,\tau-1}}{\cos\phi^T_{jrow} dxu_i} \tag{22.56}$$
$$diff_fn^{old}_{i,k,j} = diff_cnt_{i,k,j} \cdot \cos\phi^U_{jrow} \frac{t_{i,k,j+1,n,\tau-1} - t_{i,k,j,n,\tau-1}}{dyu_{jrow}} \tag{22.57}$$

where $diff_cet_{i,k,j}$ and $diff_cnt_{i,k,j}$ are the coefficients from whatever subgrid scale mixing parameterization option has been enabled. The $[K^{31}]$ and $[K^{32}]$ parts of the diffusive flux across the bottom face of the T cell is given as $diff_fbiso_{i,k,j}$ which is the vertical flux term $-F^z_{i,k,j}$ from Section 35.1 minus the $[K^{33}]$ piece.

$$diff_fbiso_{i,k,j} = -F^z_{i,k,j} - K^{33}_{i,k,j}\delta_z T_{i,k,j} \tag{22.58}$$

Note that the isoneutral diffusion coefficient A_I has been absorbed into the tensor components $[K^{31}]$, $[K^{32}]$, $[K^{33}]$. The $[K^{33}]$ component is handled implicitly as indicated in Section 22.8.7.

22.8.4 Source terms

It is possible to introduce sources into the tracer equation by enabling option *source_term*. If this option is enabled, the source term is initialized to zero for each component of the tracer equation.

[19] Zero to within roundoff. Note that unlike $\text{adv_vbt}_{i,kb,j}$, $\text{adv_vbu}_{i,kb,j}$ can be non-zero if there is a bottom slope.

$$source_{i,k,j} = 0.0 \tag{22.59}$$

Adding new sources or sinks to the tracer equations is a matter of calculating them and adding to $source_{i,k,j}$ as indicated in Sections 22.8.5 and 22.8.6.

22.8.5 Sponge boundaries

If option *sponges* is enabled, a Newtonian damping term is added to the tracer equation through the source term near northern and southern boundaries where γ^{-1} is a Newtonian damping time scale input through a namelist and $t^{\star}_{i,k,j,n}$ is interpolated in time from data prepared by the scripts in PREP_DATA. If option *equatorial_sponge* is enabled, then the profile from Equation (28.1) is used for $t^{\star}_{i,k,j,n}$ instead of data prepared in PREP_DATA. Refer to Section 14.4 for information on namelist variables. The time interpolation method is the same as described in Section 19.2.

$$source_{i,k,j} = source_{i,k,j} - \gamma \cdot (t_{i,k,j,n,\tau-1} - t^{\star}_{i,k,j,n}) \tag{22.60}$$

22.8.6 Shortwave solar penetration

If option *shortwave* is enabled, the divergence of shortwave penetration is also added to the source term using

$$source_{i,k,j} = source_{i,k,j} + sbcocn_{i,jrow,m_{i,jrow,isw}} \cdot divpen_k \tag{22.61}$$

where subscript *isw* points to the shortwave surface boundary condition. This only applies for $n = 1$. Refer to Section 28.2.10 for more details about the divergence of shortwave penetration.

22.8.7 Tracer operators

Finite difference tracer operators (see Section 14.1 for naming convention) are used for the various terms when solving for $t_{i,k,j,n,\tau+1}$ and parallel those used in the solution of the internal mode velocities. Operators are implemented as statement functions and so require negligible memory allocation. These operators are also used in diagnostics throughout MOM. The definitions of the operators are given in Section 21.3.2 and their canonical forms are expanded as:

$$ADV_Tx_{i,k,j} = \frac{adv_fe_{i,k,j} - adv_fe_{i-1,k,j}}{2\cos\phi^T_{jrow} dxt_i} \tag{22.62}$$

$$ADV_Ty_{i,k,j} = \frac{adv_fn_{i,k,j} - adv_fn_{i,k,j-1}}{2\cos\phi^T_{jrow} dyt_{jrow}} \tag{22.63}$$

$$ADV_Tz_{i,k,j} = \frac{adv_fb_{i,k-1,j} - adv_fb_{i,k,j}}{2\, dzt_k} \tag{22.64}$$

$$DIFF_Tx_{i,k,j} = \frac{diff_fe_{i,k,j} \cdot tmask_{i+1,k,j} - diff_fe_{i-1,k,j} \cdot tmask_{i-1,k,j}}{\cos\phi^T_{jrow} \cdot dxt_i} \tag{22.65}$$

$$DIFF_Ty_{i,k,j} = \frac{diff_fn_{i,k,j} \cdot tmask_{i,k,j+1} - diff_fn_{i,k,j-1} \cdot tmask_{i,k,j-1}}{\cos\phi^T_{jrow} \cdot dyt_{jrow}} \tag{22.66}$$

$$DIFF_Tz_{i,k,j} = \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} \tag{22.67}$$

where the vertical diffusion operator needs no masking because the masking effect has been built into boundary conditions $stf_{i,j,n}$ and $btf_{i,j,n}$ at the top and bottom of the column. Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in adv_fe) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Tx) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux.

22.8.7.1 Implicit vertical diffusion

When option *implicitvmix* is enabled, the vertical diffusion operator in Equation (22.67) becomes

$$DIFF_Tz_{i,k,j} = (1 - aidif) \cdot \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} \tag{22.68}$$

where the implicit factor $aidif$ is used to separate the term into explicit and implicit parts.

22.8.7.2 Isoneutral mixing

When option *isoneutralmix* is enabled, the vertical diffusion operator in Equation (22.67) becomes

$$\begin{aligned} DIFF_Tz_{i,k,j} &= (1 - aidif) \cdot \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} + \\ &\quad \frac{diff_fbiso_{i,k-1,j} - diff_fbiso_{i,k,j}}{dzt_k} \end{aligned} \tag{22.69}$$

where the implicit factor is $aidif$ and the K^{33} term is within $diff_fb_{i,k-1,j}$ which is solved implicitly. The K^{31} and K^{32} terms are handled explicitly (no implicit treatment) within $diff_fbiso_{i,k,j}$ which is defined by Equation (22.58). The horizontal isoneutral fluxes are given by Equations (22.54) and (22.55). Refer to Section 35.1 for further details on isoneutral mixing.

22.8.7.3 Gent-McWilliams advection velocities

If option *isoneutralmix* and option *gent_mcwilliams* and option *gm_advect* are enabled, the advective operators for Equation (22.75) are given by

$$ADV_Txiso_{i,k,j} = \frac{adv_vetiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{\lambda} - adv_vetiso_{i-1,k,j} \cdot \overline{t_{i-1,k,j,n,\tau-1}}^{\lambda}}{\cos\phi^T_{jrow} dxt_i} \tag{22.70}$$

$$ADV_Tyiso_{i,k,j} = \frac{adv_vntiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{\phi} - adv_vntiso_{i,k,j-1} \cdot \overline{t_{i,k,j-1,n,\tau-1}}^{\phi}}{\cos\phi^T_{jrow} dyt_{jrow}} \tag{22.71}$$

$$ADV_Tziso_{i,k,j} = \frac{adv_fbiso_{i,k-1,j} - adv_fbiso_{i,k,j}}{2dzt_k} \tag{22.72}$$

where twice the advective flux at the bottom of the T cell is

$$adv_fbiso_{i,k,j} = 2(adv_vbiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{z}) \tag{22.73}$$

As with the normal advective flux, the factor of two is for reasons of computational speed. Note also, that a cosine factor has been absorbed within $adv_vntiso_{i,k,j}$ to mimic the regular meridional advective velocity. The horizontal components are treated differently than the vertical component to save memory at the expense of speed. If memory is not a problem, the advective flux across the north and east face of the T cell can be computed separately to eliminate the redundancy. The total advection becomes the regular advection plus the Gent-McWilliams advection given as

$$\mathcal{L}^{total}(t_{i,k,j,n,\tau}) = \mathcal{L}^{T}(t_{i,k,j,n,\tau}) + \mathcal{L}^{gm}(t_{i,k,j,n,\tau-1}) \tag{22.74}$$

where the Gent-McWilliams advective operator is

$$\mathcal{L}^{gm}(t_{i,k,j,n,\tau-1}) = ADV_Txiso_{i,k,j} + ADV_Tyiso_{i,k,j} + ADV_Tziso_{i,k,j} \tag{22.75}$$

It should be noted that when option *fct* is enabled, Equation 22.75 is not used in *tracer.F*. Instead, the Gent-McWilliams advection velocities are added to the regular advective velocities within the flux corrected transport calculations.

22.8.8 Solving for the tracer

The tracer equation is solved using a variety of time differencing schemes (leapfrog, forward, Euler backward) as outlined in Section 21.4. A complication arises due to the vertical diffusion term which may be solved explicitly or implicitly. The implicit treatment is appropriate when large vertical diffusion coefficients (or small vertical grid spacing) limit the time step. The following discussion pertains to the leapfrog scheme. The other schemes amount to changing $2\Delta t$ to Δt in what follows.

22.8.8.1 Explicit vertical diffusion

Using the above operators, the tracer is computed directly as:

$$\begin{aligned} t_{i,k,j,n,\tau+1} = t_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot (\; & DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + DIFF_Tz_{i,k,j} \\ - \; & ADV_Tx_{i,k,j} - ADV_Ty_{i,k,j} - ADV_Tz_{i,k,j} \\ + \; & source_{i,k,j}) \cdot tmask_{i,k,j} \end{aligned} \tag{22.76}$$

When options *isoneutralmix* and *gent_mcwilliams* are enabled, the right hand side of Equation (22.76) also includes the flux form of the advection terms[20] given by Equations (22.70), (22.71), and (22.72). In effect, combining Equations (21.48) and (22.75) gives the total advection as in Equation (22.74).

[20]These could have been added directly to the advective fluxes computed previously. For now they are kept separately for diagnostic reasons.

22.8.8.2 Implicit vertical diffusion

In general, vertical diffusion is handled implicitly when using option *implicitvmix* or option *isoneutralmix*. In either case, Equation (22.76) is solved in two steps. The first calculates an explicit piece $t^{\star}_{i,k,j,n}$ using all terms except the portion of vertical diffusion which is to be solved implicitly. The equation is

$$\begin{aligned} t^{\star}_{i,k,j,n} = t_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot (\; & DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + (1 - aidif) \cdot DIFF_Tz_{i,k,j} \\ - \; & ADV_Tx_{i,k,j} - ADV_Ty_{i,k,j} - ADV_Tz_{i,k,j} \\ + \; & source_{i,k,j}) \cdot tmask_{i,k,j} \end{aligned} \tag{22.77}$$

where $aidif$ is the implicit vertical diffusion factor. Setting *aidif = 1.0* gives full implicit treatment and setting *aidif = 0* gives full explicit treatment for the vertical diffusion term. Intermediate values give semi-implicit treatment. The second step involves solving the implicit equation

$$t_{i,k,j,n,\tau+1} = t^{\star}_{i,k,j,n} + 2\Delta\tau \cdot \delta_z(aidif \cdot diff_cbt_{i,k,j} \cdot \delta_z(t_{i,k,j,n,\tau+1})) \tag{22.78}$$

Notice that $t_{i,k,j,n,\tau+1}$ appears on both sides of the equation. Solving this involves inverting a tri-diagonal matrix and the technique is taken from pages 42 and 43 of Numerical Recipes in Fortran (1992). For details, refer to Section 38.5.

22.8.9 Diagnostics

At this point, diagnostics are computed for tracer component n. For a description of the diagnostics, refer to Chapter 39.

22.8.10 End of tracer components

At this point, the "do $n = 1, nt$" loop issued in Section 22.8.1 is ended and all tracers have been computed.

22.8.11 Explicit Convection

The condition for gravitational instability is given by

$$\delta_z(\rho_{i,k,j}) < 0 \tag{22.79}$$

If option *implicitvmix* is enabled, the instability is handled by implicit vertical diffusion using huge vertical diffusion coefficients $diff_cbt_limit$ input through namelist. Refer to Section 14.4 for information on namelist variables. Otherwise, when option *implicitvmix* is <u>not</u> enabled, the temperature and salinity within each vertical column of T cells is stabilized where needed by one of two explicit convection methods. These methods involve mixing predicted temperature and salinity between two adjacent levels ($t_{i,k,j,n,\tau+1}$ and $t_{i,k+1,j,n,\tau+1}$ for $n = 1, 2$). If option *fullconvect* is enabled, the convection scheme of Rahmstorf (1993) is used. Otherwise, the original scheme involving alternate mixing of odd and even pairs of levels is used. Refer to Section 33.1. The mixing in the latter case may be incomplete and multiple passes through the scheme is allowed by variable *ncon* which is also input through namelist. If explicit convection is active, additional diagnostics are computed. Refer to Section 15.1.1 for the details.

22.8.12 Filtering

If the time step constraint imposed by convergence of meridians is to be relaxed, $t_{i,k,j,n,\tau+1}$ is filtered in longitude by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil* or finite impulse response filtering enabled by option *firfil*. Both should be used with caution and only when necessary at high latitudes. *firfil* is much faster than *fourfil*. Refer to Section 27.1.

22.8.13 Accumulating $sbcocn_{i,jrow,m}$

Finally, if required as a surface boundary condition for the atmosphere, $t_{i,1,j,n,\tau}$ is accumulated and averaged over one time segment. The result is stored in $sbcocn_{i,jrow,m}$where m relates the ordering of n in the array of surface boundary conditions as discussed in Section 19.3.

Note that in a coupled mode where the average over a segment contains many time steps, it does not matter much whether τ or $\tau+1$ values are accumulated because the average over a segment will be about the same in both cases. However, when the segment contains only one time step, the τ time level of SST should be used. The τ value is appropriate for Test Case 1 and 2 when vertical mixing is explicit (i.e. option *implicitvmix* is not enabled). When the τ time level is accumulated, the boundary condition for vertical diffusion (on the next time step) uses values from $\tau-1$ which is correct because explicit vertical diffusion should be lagged by one time step. If option *implicitvmix* is used however, then the $\tau+1$ SST values should be used in the implicit vertical diffusion ... but $\tau+1$ values are unknown so τ values are used instead as the best available approximation. To achieve this, $\tau+1$ instead of τ values must be accumulated during the previous time step.

22.9 baroclinic (computes internal mode velocities)

Subroutine *baroclinic*[21] computes the baroclinic or internal mode velocities $\hat{u}_{i,k,j,n,\tau+1}$ for velocity components $n=1$ (the zonal velocity) and 2 (the meridional velocity). They are the finite difference counterparts of the velocities in Equations (4.1) and (4.2) after the vertical means have been removed. The finite difference equations given below are written in flux form to allow conservation of first and second moments as discussed in Chapter A. The solution starts by computing the hydrostatic pressure gradient terms.

22.9.1 Hydrostatic pressure gradient terms

Both horizontal pressure gradient terms in Equations (4.1) and (4.2) are computed first. The unknown barotropic surface pressure gradients exerted by the rigid lid at the ocean surface are not needed since only the internal mode velocities are being solved for[22]. The discretized forms are given by

$$-\frac{1}{\rho_\circ a\cdot\cos\phi}p_\lambda \approx -grad_p_{i,k,j,1} = -\frac{1}{\rho_\circ\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{p_{i,k,j}}^\phi) \tag{22.80}$$

$$-\frac{1}{\rho_\circ a}p_\phi \approx -grad_p_{i,k,j,2} = -\frac{1}{\rho_\circ}\delta_\phi(\overline{p_{i,k,j}}^\lambda) \tag{22.81}$$

[21]Contained in file baroclinic.F.

[22]Indeed, lack of knowledge about these barotropic surface pressure gradients is the very reason the internal mode velocities are being solved for.

The discrete form of the hydrostatic Equation (4.4) is given is given by Equation (21.42) with $grav = 980.6$ cm$/sec^2$ and is used to eliminate p from Equations (22.80) and (22.81). Note that the hydrostatic pressure is an anomaly because $\rho_{i,k,j}$ is an anomaly as described in Section 15.1.2. The hydrostatic pressure gradient terms at the depth of the grid point in the first level are

$$grad_p_{i,1,j,1} = \frac{grav \cdot dzw_0}{\rho_\circ \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\rho_{i,1,j}}^\phi) \tag{22.82}$$

$$grad_p_{i,1,j,2} = \frac{grav \cdot dzw_0}{\rho_\circ} \delta_\phi(\overline{\rho_{i,1,j}}^\lambda) \tag{22.83}$$

which assumes that density at the ocean surface is the same as at depth $zt_{k=1}$[23]. Pressure gradient terms at successive levels 2 through $kmu_{i,jrow}$ are built by partial sums

$$grad_p_{i,k,j,1} = \sum_{m=2}^{k} \frac{grav \cdot dzw_{m-1}}{\rho_\circ \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\frac{\rho_{i,m,j} + \rho_{i,m-1,j}}{2}}^\phi) \tag{22.84}$$

$$grad_p_{i,k,j,2} = \sum_{m=2}^{k} \frac{grav \cdot dzw_{m-1}}{\rho_\circ} \delta_\phi(\overline{\frac{\rho_{i,m,j} + \rho_{i,m-1,j}}{2}}^\lambda) \tag{22.85}$$

When option *pressure_gradient_average* is enabled, ρ in Equations (21.42) through (22.85) is replaced by the time averaged value $\tilde{\rho}$ given by Equation (38.1) and discussed further in Section 38.3. Note that since the integration is to levels of constant depth, hydrostatic pressure gradients at depth are zero when ρ is a function of z only. Note also, that the pressure gradients are different for grid construction methods 1 and 2 as discussed in Chapter 16.2.3.

22.9.2 Momentum components

After constructing both components of the hydrostatic pressure gradients, the remaining terms on the right hand side of the momentum equations are computed first for the zonal component of momentum given by Equation (4.1) and then for the meridional component of momentum given by Equation (4.2). This is accomplished by a "do $n = 1, 2$" loop extending to Section 22.9.9. All workspace arrays used in the momentum operators are re-computed for each component of momentum. Therefore all momentum operators can only be used from within this n loop which implies all diagnostics using these operators are issued from within this n loop.

22.9.3 Advective and Diffusive fluxes

Advective and diffusive fluxes across the north, east, and bottom faces of U-cells within the memory window are calculated for use by the advection and diffusion operators described in Sections 21.3.1 and 22.9.5. The canonical forms of the fluxes are given in Section 21.3.1 and expanded below. The actual form for the diffusive flux may differ depending upon which option is enabled. Note that advective flux is missing a factor of $\frac{1}{2}$ which is incorporated in the advection operator for speed reasons. Also for reasons of speed, a cosine factor has been absorbed into the definition of $adv_vnu_{i,k,j}$. Fluxes are typically computed from $j = 2$ through $j = jmw - 1$, except for the last memory window, which may be only partially full. Meridional

[23] This is the depth of the grid point within the first T cell and U cell.

operators require meridional flux to be computed for $j = 1$, which is why the lower limit of the j index is not identical for all fluxes. Refer to the simplified schematic in Figure 11.2 to see what rows are available within the memory window.

$$adv_fe_{i,k,j} = adv_veu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i+1,k,j,n,\tau}) \ , j = 2, jmw - 1 \tag{22.86}$$

$$adv_fn_{i,k,j} = adv_vnu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i,k,j+1,n,\tau}) \ , j = 1, jmw - 1 \tag{22.87}$$

$$adv_fb_{i,k,j} = adv_vbu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i,k+1,j,n,\tau}) \ , j = 2, jmw - 1 \tag{22.88}$$

$$diff_fe_{i,k,j} = diff_ceu_{i,k,j} \cdot \frac{u_{i+1,k,j,n,\tau-1} - u_{i,k,j,n,\tau-1}}{\cos \phi^{U}_{jrow} dxt_{i+1}} \ , j = 2, jmw - 1 \tag{22.89}$$

$$diff_fn_{i,k,j} = diff_cnu_{i,k,j} \cdot \cos \phi^{T}_{jrow+1} \frac{u_{i,k,j+1,n,\tau-1} - u_{i,k,j,n,\tau-1}}{dyt_{jrow+1}} \ , j = 1, jmw - 1 \tag{22.90}$$

$$diff_fb_{i,k,j} = diff_cbu_{i,k,j} \cdot \frac{u_{i,k,j,n,\tau-1} - u_{i,k+1,j,n,\tau-1}}{dzw_k} \ , j = 2, jmw - 1 \tag{22.91}$$

Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in adv_fe) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. The top and bottom boundary conditions are applied to the vertical fluxes where the bottom cell is at level $kb = kmu_{i,jrow}$. Also, the advective flux through the bottom of the last level $k = km$ is set to zero since there can be no bottom slope at the bottom of the deepest level.

$$kb = kmu_{i,jrow} \tag{22.92}$$

$$diff_fb_{i,0,j} = smf_{i,j,n} \tag{22.93}$$

$$diff_fb_{i,kb,j} = bmf_{i,j,n} \tag{22.94}$$

$$adv_fb_{i,0,j} = adv_vbu_{i,0,j} \cdot (u_{i,1,j,n,\tau} + u_{i,1,j,n,\tau}) \tag{22.95}$$

$$adv_fb_{i,km,j} = 0.0 \tag{22.96}$$

22.9.4 Source terms

There is the possibility of introducing sources into the momentum equation by enabling option *source_term*. If this option is enabled, the source term is initialized to zero for each component of the momentum equations.

$$source_{i,k,j} = 0.0 \tag{22.97}$$

Adding new sources or sinks to the momentum equations is a matter of calculating new terms and adding them to $source_{i,k,j}$. For instance, suppose it was desirable to add a Rayleigh damping in some region of the domain. The damping could be calculated as

$$rayleigh_{i,k,j,n} = -\gamma \cdot u_{i,k,j,n,\tau-1} \cdot ray_mask_{i,jrow} \tag{22.98}$$

Where $ray_mask_{i,jrow}$ is a mask of ones and zeros to limit the region of damping and γ is the reciprocal of the damping time scale in seconds. This term could be added to the momentum equations using

$$source_{i,k,j} = source_{i,k,j} + rayleigh_{i,k,j,n} \tag{22.99}$$

and additional sources and sinks could be stacked in a likewise manner.

22.9.5 Momentum operators

Finite difference momentum operators (see Section 14.1 for naming convention) are used for various terms when solving for the time derivatives in Equations (4.1) and (4.2). Operators are implemented as statement functions and so require negligible memory allocation. These operators are also used in diagnostics throughout MOM. Note that the arrays embedded within the operators must be defined when the operators are invoked. The definitions of the operators are given in Section 21.3.1 and their canonical forms are expanded as:

$$DIFF_Ux_{i,k,j} = \frac{diff_fe_{i,k,j} - diff_fe_{i-1,k,j}}{\cos\phi^U_{jrow} \cdot dxu_i} \tag{22.100}$$

$$DIFF_Uy_{i,k,j} = \frac{diff_fn_{i,k,j} - diff_fn_{i,k,j-1}}{\cos\phi^U_{jrow} \cdot dyu_{jrow}} \tag{22.101}$$

$$DIFF_Uz_{i,k,j} = \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} \tag{22.102}$$

$$\begin{aligned} DIFF_metric_{i,k,j,n} &= A_m \frac{1-\tan^2\phi^U_{jrow}}{radius^2} u_{i,k,j,n,\tau-1} \\ &\mp A_m \frac{2\sin\phi^U_{jrow}}{radius \cdot \cos^2\phi^U_{jrow}} \frac{u_{i+1,k,j,3-n,\tau-1} - u_{i-1,k,j,3-n,\tau-1}}{dxt_i + dxt_{i+1}} \end{aligned} \tag{22.103}$$

$$ADV_Ux_{i,k,j} = \frac{adv_fe_{i,k,j} - adv_fe_{i-1,k,j}}{2\cos\phi^U_{jrow} dxu_i} \tag{22.104}$$

$$ADV_Uy_{i,k,j} = \frac{adv_fn_{i,k,j} - adv_fn_{i,k,j-1}}{2\cos\phi^U_{jrow} dyu_{jrow}} \tag{22.105}$$

$$ADV_Uz_{i,k,j} = \frac{adv_fb_{i,k-1,j} - adv_fb_{i,k,j}}{2dzt_k} \tag{22.106}$$

$$ADV_metric_{i,k,j,n} = \mp \frac{\tan\phi^U_{jrow}}{radius} u_{i,k,j,1,\tau} \cdot u_{i,k,j,3-n,\tau} \tag{22.107}$$

$$\tag{22.108}$$

where the earth's rotation rate[24] $\Omega = \frac{\pi}{43082.0} sec^{-1}$, and the mean radius of the earth (from Gill, page 597) is given by $radius = 6371.0x10^5 cm$. The $\mp$ indicates a minus sign for $n = 1$ and a plus sign for $n = 2$. Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in adv_fe) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining

[24]The 43082.0 sec is arrived at assuming approximately $86400 * (1 - \frac{1}{366}) = 86164$ seconds in one sidereal day. The 366 is used to account for a 1 day co-rotation.

it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux.

In terms of the above operators, the finite difference advection plus advection metric term in Equations (4.1) and (4.2) is given as

$$\mathcal{L}^U(u_{i,k,j,n,\tau}) = ADV_Ux_{i,k,j} + ADV_Uy_{i,k,j} + ADV_Uz_{i,k,j} + ADV_metric_{i,k,j,n} \quad (22.109)$$

22.9.5.1 Coriolis treatment

The Coriolis term may be handled explicitly or semi-implicitly. Refer to Section 27.2.1 for a discussion of the semi-implicit treatment of the Coriolis term and when it is warranted. When solving explicitly, the Coriolis term is given by

$$CORIOLIS_{i,k,j,n} = \pm 2\Omega \sin \phi^U_{jrow} \cdot u_{i,k,j,3-n,\tau} \quad (22.110)$$

When solving implicitly, option *damp_inertial_oscillation* is enabled and the semi-implicit Coriolis factor *acor* is used. In the semi-implicit treatment of the Coriolis term, the explicit part is given by

$$CORIOLIS_{i,k,j,n} = \pm 2\Omega \sin \phi^U_{jrow} \cdot u_{i,k,j,3-n,\tau-1} \quad (22.111)$$

and the handling of the implicit part is detailed below in Section 22.9.10. For both explicit and semi-implicit treatment, the $\pm$ indicates a plus sign for $n = 1$ and a minus sign for $n = 2$.

22.9.6 Solving for the time derivative of velocity

The vertical diffusion term in the momentum equations may be solved explicitly or implicitly. The implicit treatment is appropriate when large diffusion coefficients (or small vertical grid spacing) limit the time step.

22.9.6.1 Explicit vertical diffusion

Using the above operators (and flux terms calculated previously), the time derivative of velocity is computed by including all terms except the unknown surface pressure gradients exerted by the rigid lid at the ocean surface. If the vertical diffusion term is handled explicitly, then the time derivative of velocity is given by:

$$\begin{aligned}\delta_\tau(u^\star_{i,k,j,n}) = \quad & (DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_Uz_{i,k,j} \\ & + DIFF_metric_{i,k,j,n} - ADV_Ux_{i,k,j} - ADV_Uy_{i,k,j} \\ & - ADV_Uz_{i,k,j} + ADV_metric_{i,k,j,n} - grad_p_{i,k,j,n} \\ & + CORIOLIS_{i,k,j,n} + source_{i,k,j}) \cdot umask_{i,k,j}\end{aligned} \quad (22.112)$$

22.9.6.2 Implicit vertical diffusion

When solving the vertical diffusion implicitly, option *implicitvmix* must be enabled and the implicit diffusion factor *aidif* is used. The vertical diffusion operator in Equation (22.102) is replaced by

$$DIFF_Uz_{i,k,j} = (1 - aidif) \cdot \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} \tag{22.113}$$

and the time derivative of velocity is computed in two steps. The first computes the derivative as

$$\begin{aligned}\delta_\tau(u^{\star\star}_{i,k,j,n}) = \quad & (DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_Uz_{i,k,j} \\ & + DIFF_metric_{i,k,j,n} - ADV_Ux_{i,k,j} - ADV_Uy_{i,k,j} \\ & - ADV_Uz_{i,k,j} + ADV_metric_{i,k,j,n} - grad_p_{i,k,j,n} \\ & + CORIOLIS_{i,k,j,n} + source_{i,k,j}) \cdot umask_{i,k,j}\end{aligned} \tag{22.114}$$

and an intermediate velocity $u^{\star\star}_{i,k,j,n}$ is constructed as

$$u^{\star\star}_{i,k,j,n} = u_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot \delta_t(u^{\star\star}_{i,k,j,n}) \tag{22.115}$$

The next step solves the following equation by inverting a tri-diagonal matrix as given in pages 42 and 43 of Numerical Recipes in Fortran (1992)[25].

$$u^{\star}_{i,k,j,n,\tau+1} = u^{\star\star}_{i,k,j,n} + 2\Delta\tau \cdot \delta_z(aidif \cdot diff_cbu_{i,k,j} \cdot \delta_z(u^{\star}_{i,k,j,n,\tau+1})) \tag{22.116}$$

The time derivative of velocity is then given by

$$\delta_\tau(u^{\star}_{i,k,j,n}) = \frac{u^{\star}_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau} \tag{22.117}$$

The recommendation from Haltiner and Williams (1980) is for $aidif = 0.5$ which gives the Crank-Nicholson implicit scheme which is always stable. This setting is supposed to be the most accurate one. However, this is not the case when solving a time dependent problem as discussed in Section38.5. Note that in the model code, $\delta_\tau(u^{\star}_{i,k,j,n})$ is temporarily stored in $u_{i,k,j,n,\tau+1}$ as a space saving measure until the internal modes are computed as indicated below.

22.9.7 Diagnostics

At this point, diagnostics are computed for velocity component n using previously defined operators. For a description of the diagnostics, refer to Chapter 39.

22.9.8 Vertically averaged time derivatives of velocity

The vertical average of the time derivative of the velocity components is computed as the last item within the loop over velocity components. The vertical averages are constructed as

$$zu_{i,jrow,n} = \frac{1}{H_{i,jrow}} \cdot \sum_{k=1}^{km} dzt_k \cdot \delta_\tau(u^{\star}_{i,k,j,n}) \tag{22.118}$$

and later used to construct the forcing for the external mode.

[25]The surface boundary condition $smf_{i,j,n}$ should be at $\tau + 1$ but that value isn't known. Instead, the τ value is used. In MOM 1, The Richardson number in *ppmix.F* was calculated at $tau - 1$ in the implicit case. It should be at $\tau + 1$ but that is unknown. In MOM, the Richardson number can easily be changed to τ. The effect of this on the solution should be small but has not been explored.

22.9.9 End of momentum components

At this point, the "do $n = 1, 2$" loop issued in Section 22.9.2 is ended. After completion of this loop, all right hand terms in Equations (4.1) and (4.2) have been computed.

22.9.10 Computing the internal modes of velocity

The internal modes of velocity are solved using a variety of time differencing schemes (leapfrog, forward, Euler backward) as outlined in Section 21.4. When the time derivatives of both velocity components have been computed using Equations (22.112) through (22.117), the uncorrected velocities[26] $u'_{i,k,j,1,\tau+1}$ and $u'_{i,k,j,2,\tau+1}$ are computed. A complication arises due to the vertical diffusion term which may be solved explicitly or implicitly. The implicit treatment is appropriate when large vertical viscosity coefficients (or small vertical grid spacing) limit the time step. The following discussion pertains to the leapfrog scheme. The other schemes amount to changing $2\Delta t$ to Δt in what follows.

22.9.10.1 Explicit Coriolis treatment

If the Coriolis term is treated explicitly, the solution is given by

$$u'_{i,k,j,1,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,1}) \tag{22.119}$$

$$u'_{i,k,j,2,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,2}) \tag{22.120}$$

22.9.10.2 Semi-implicit Coriolis treatment

If the Coriolis term is treated semi-implicitly (refer to Section 27.2.1 for a discussion of the semi-implicit treatment and when it is warranted), the equations are

$$u'_{i,k,j,1,\tau+1} - 2\Delta\tau \cdot acor \cdot f_{jrow} \cdot u'_{i,k,j,2,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,1}) \tag{22.121}$$

$$u'_{i,k,j,2,\tau+1} + 2\Delta\tau \cdot acor \cdot f_{jrow} \cdot u'_{i,k,j,1,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,2}) \tag{22.122}$$

and the solution, after a little algebra, is given by

$$u'_{i,k,j,1,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \frac{\delta_\tau(u^\star_{i,k,j,1}) + (2\Delta\tau \cdot f \cdot acor) \cdot \delta_\tau(u^\star_{i,k,j,2})}{1 + (2\Delta\tau \cdot f \cdot acor)^2} \tag{22.123}$$

$$u'_{i,k,j,2,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \frac{\delta_\tau(u^\star_{i,k,j,2}) - (2\Delta\tau \cdot f \cdot acor) \cdot \delta_\tau(u^\star_{i,k,j,1})}{1 + (2\Delta\tau \cdot f \cdot acor)^2} \tag{22.124}$$

The pure internal modes $\hat{u}_{i,k,j,n,\tau+1}$ are then calculated by removing the incorrect vertical means

$$\hat{u}_{i,k,j,n,\tau+1} = u'_{i,k,j,n,\tau+1} - \frac{1}{H_{i,jrow}} \cdot \sum_{k=1}^{kb} dzt_k \cdot u'_{i,k,j,n,\tau+1} \tag{22.125}$$

[26] Velocities with incorrect vertical means due to the missing unknown surface pressure.

where $kb = kmu_{i,jrow}$. In the MOM code, $\hat{u}_{i,k,j,n,\tau+1}$ is actually stored in array $u_{i,k,j,n,\tau+1}$ and it is these internal mode velocities that are saved to disk from the memory window. The full velocity is actually not available until the next timestep when the external mode velocity is added to the internal mode velocity in subroutine *loadmw*.

22.9.11 Filtering

If the time step constraint imposed by convergence of meridians is to be relaxed, the internal modes are filtered by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil*, or finite impulse response filtering enabled by option *firfil*. Both should be used with caution and only when necessary at high latitudes. *firfil* is much faster than *fourfil*.

22.9.12 Accumulating $sbcocn_{i,jrow,m}$

Finally, if required as a surface boundary condition for the atmosphere (Pacanowski 1987)[27], surface velocity components $u_{i,1,j,n,\tau}$ are accumulated and averaged over one time segment. The result is stored in $sbcocn_{i,jrow,m}$where m relates the ordering of n in the array of surface boundary conditions as discussed in Section 19.3.

Note that in a coupled mode where the average over a segment contains many time steps, it does not matter much whether τ or $\tau+1$ values are accumulated because the average over a segment will be about the same in both cases. However, when the segment contains only one time step, the τ time level of velocity should be used. The τ value is appropriate for Test Case 1 and 2 when vertical mixing is explicit (i.e. option *implicitvmix* is not enabled). When the τ time level is accumulated, the boundary condition for vertical diffusion (on the next time step) uses values from $\tau-1$ which is correct because explicit vertical diffusion should be lagged by one time step. If option *implicitvmix* is used however, then the $\tau+1$ velocity values should be used in the implicit vertical diffusion ...but $\tau+1$ values are unknown so τ values are used instead as the best available approximation. To achieve this, $\tau+1$ instead of τ values must be accumulated during the previous time step.

22.10 End of computation within Memory Window

As each group of latitudes is solved within the memory window, the updated baroclinic velocities and tracers are written to disk as explained in Section 11.3.1 and the next group issued in Section 22.1 is started. After all memory windows have been solved, the forcing for the barotropic velocity has been constructed in Section 22.9.8 and the external mode equations can now be solved.

22.11 barotropic (computes external mode velocities)

After the baroclinic velocities have been computed for all latitude rows in the domain, the barotropic or external mode velocity can be computed. Subroutine *barotropic*[28] computes the barotropic velocity $\bar{u}_{i,k,j,n,\tau+1}$ for $n=1$ and 2 by one of three options: *stream_function* described

[27] Although the paper explored the affect on SST primarily through Ekman divergence, there may also be an impact on evaporative flux in certain regions where the ocean surface velocities are large.

[28] Contained in file tropic.F.

in Section 29.2, *prognostic_surface_pressure* described in Section 29.3, or *implicit_free_surface* described in Section 29.4. Refer to the above sections for details.

22.12 diago

After the external mode has been solved, the remaining diagnostics can be calculated and written out. This is done in subroutine *diago*[29] and described in Chapter 39.

[29]Contained in file diago.F.

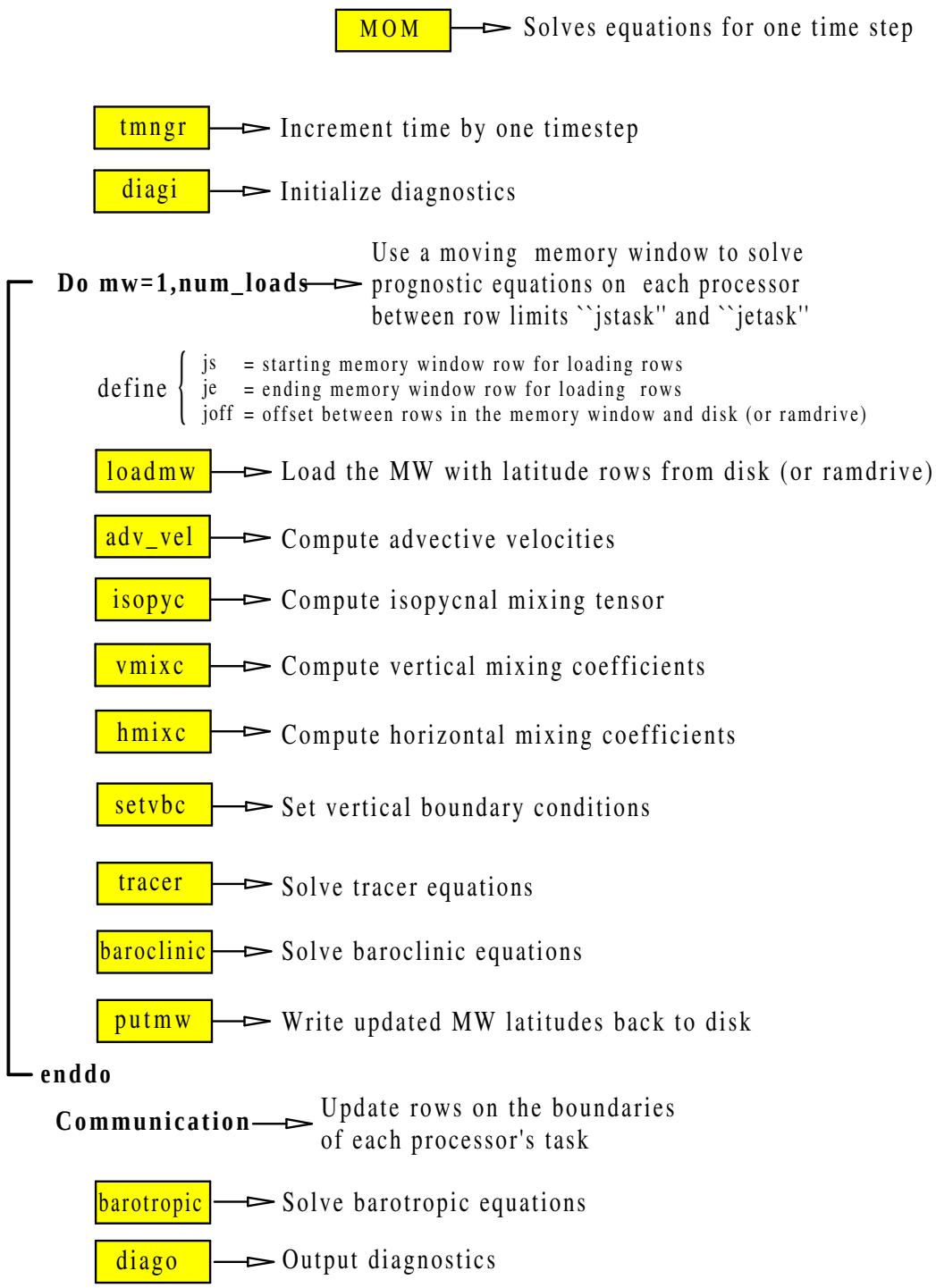


Figure 22.1: Flowchart for subroutine *mom.F* showing order of components

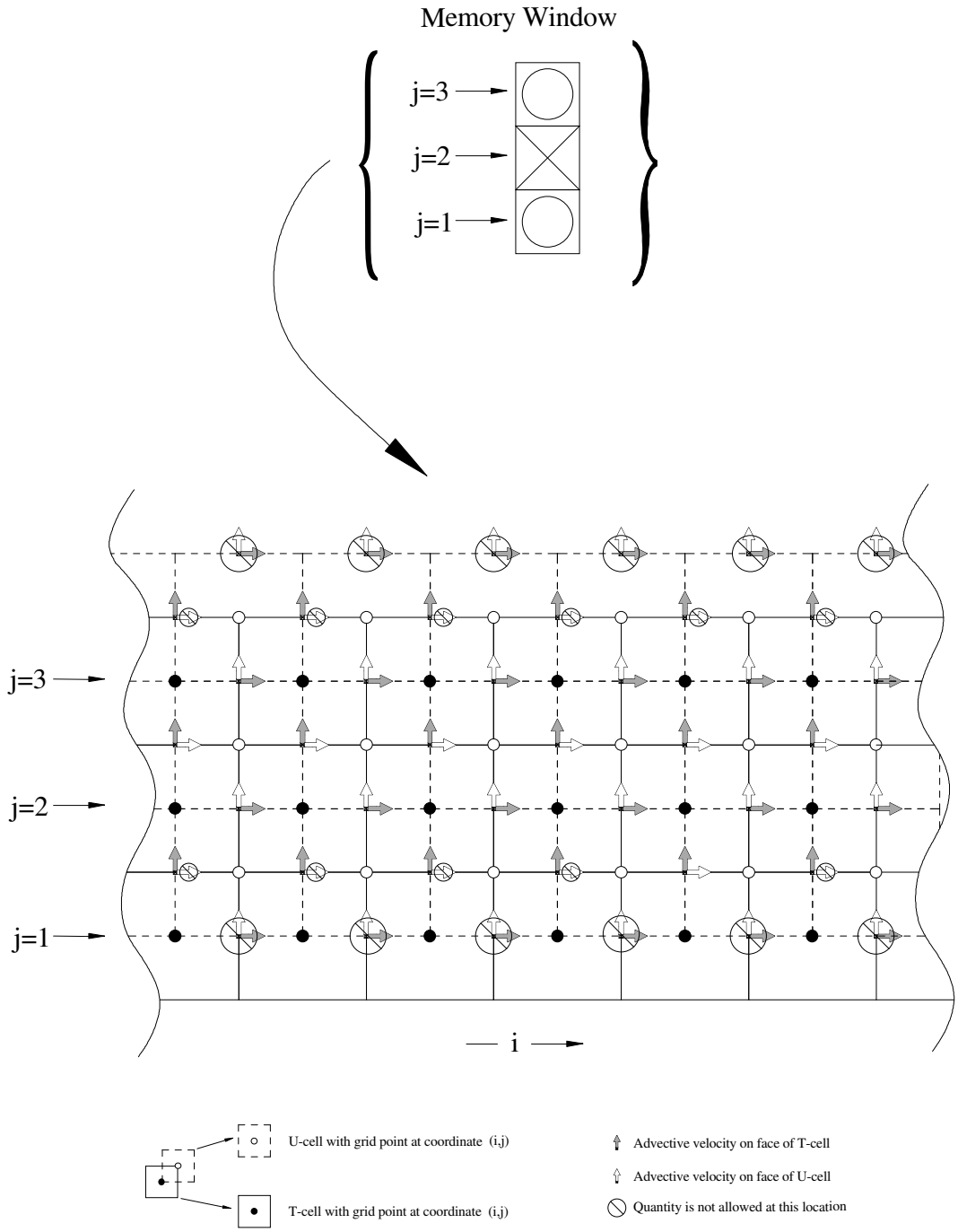


Figure 22.2: The horizontal grid layout within a memory window of size $jmw = 3$ indicating cells, grid points, and sites where derived quantities can be defined. Allowable sited determine how arrays must be dimensioned within the memory window.

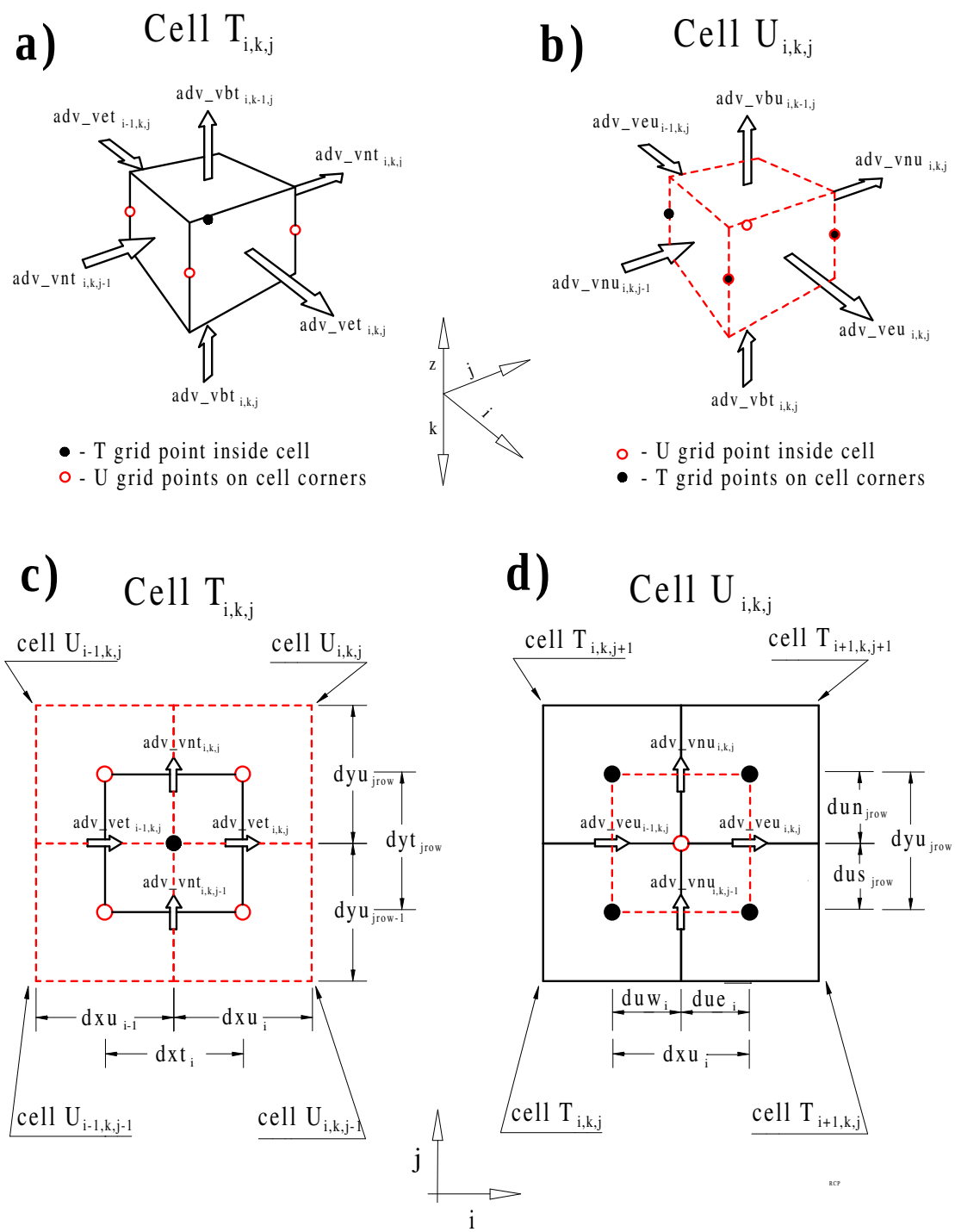


Figure 22.3: a) Advective velocities on a T-cell. b) Advective velocities on a U-cell. c) Horizontal slice through a T-cell showing grid points and surrounding U-cells d) Horizontal slice through a U-cell showing grid points and surrounding T-cells

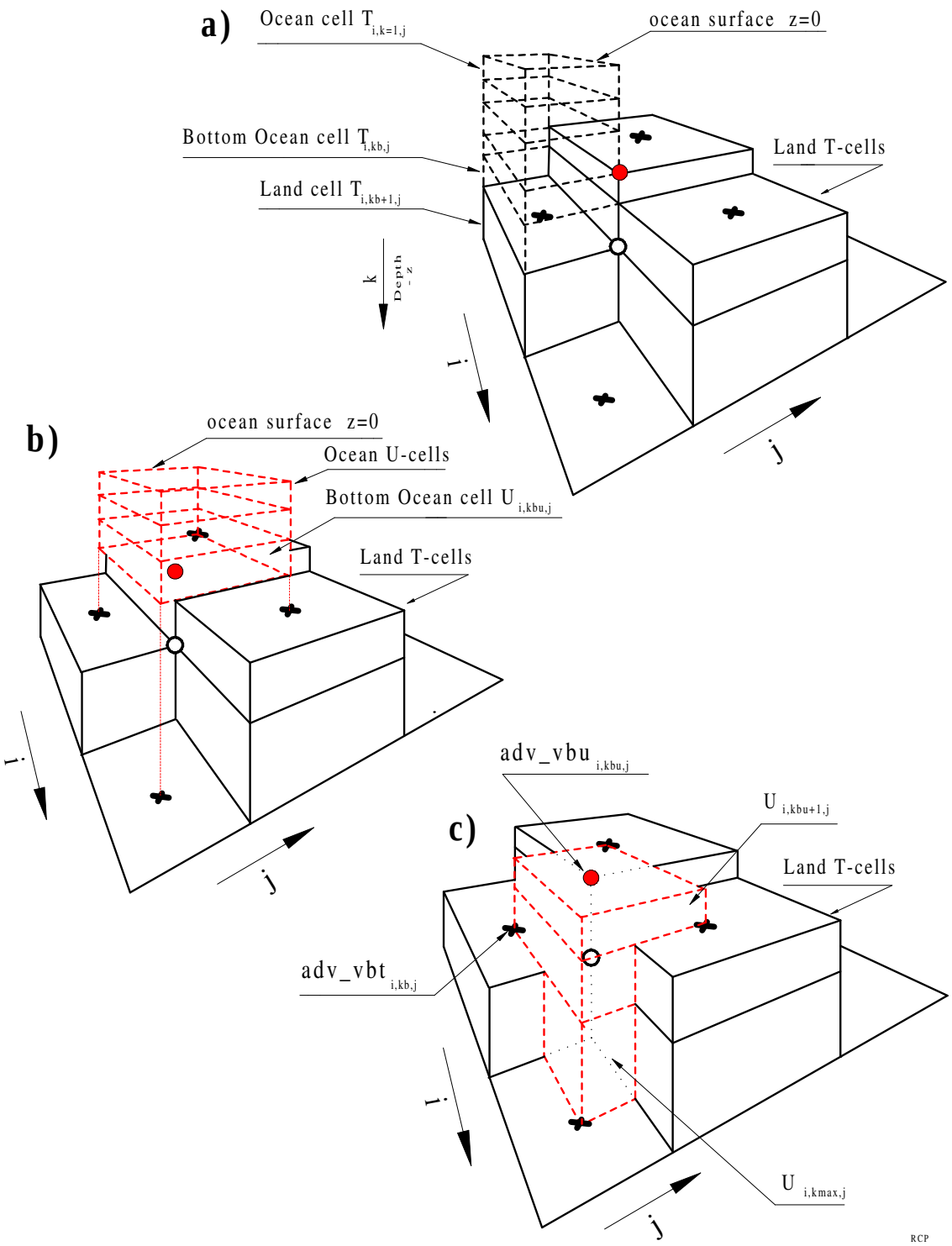


Figure 22.4: a) Column of ocean T-cells in relation to bottom topography. "x" denotes vertical velocity at the base of ocean bottom T-cells. "black dot" denotes vertical velocity at the base of an ocean bottom U-cell. "open circle" denotes position of bottom vertical velocity from continuous equation. b) Corresponding column of ocean U-cells. c) Truncated ocean U-cells below bottom of the U-cell column.

Part VII

General model options

MOM is configured in various ways through the use of options which are enabled by setting preprocessor type directives. The directives are usually placed on a preprocessor or compile statement in a run_script and take the form *-Doption1 -Doption2* etc. Note that the *-D* must be used as a prefix to the option name. For an example, refer to script *run_mom*. Care should be exercised when enabling options because there is no check for mispelled options. When enabled, options eliminate or include portions of code thereby implementing desired features. As MOM executes, a summary of all enabled options is given.

Some options are incompatible with others. Subroutine *checks*[30] looks for all conflicting options and if any are found, writes all error messages to the printout file before stopping. Various additional checking is done and warning messages may also be issued. These are less serious than error messages and allow MOM to continue. This does not mean they should be ignored. They often give information about possibly inappropriate values being used. In fact, it is a good idea to search the printout file to locate any "warning" messages and verify they are harmless before continuing. Refer to Section 3.1 for details on how to perform searches.

In the rest of this part of the manual, MOM options will be grouped into catagories and given brief explanations along with guidelines for useage. Diagnostic options are discussed separately in Part VIII. Note that all numerical indices are exposed within expressions given in this documentation so as to closely match with those used in the Fortran code. This extra bit of detail is insisted upon as another way of insuring that complex formulations are correct.

[30] contained in file checks.F

Chapter 23

Options for testing modules

Modules and the usage of modules are described in Chapter 15. Within each module is a driver intended to exercise the module in a simple environment. This means executing in a "stand alond mode" as opposed to from within a model execution. Associated with each module is a *run_script* which activates the included driver with an option. These options are listed below and are not to be enabled from within a model execution (because the model becomes the driver).

23.1 test_convect

This option activates a driver for the convection module in file convect.F. Refer to Section 15.1.1 for details.

23.2 drive_denscoef

When executing the model, required density coefficients are automatically computed[1] from within by calling module *denscoef*. However, script *run_denscoef* activates a driver in "stand alone mode" which gives details about density coefficients. Executing *run_denscoef* uses information from module *grids* and file *size.h* to construct density coefficients which depend on the vertical discretization of the grid cells. Refer to Section 15.1.2 for details on density.

23.3 drive_grids

This option activates a driver for the grids module in file grids.F. Refer to Section 15.1.3 for details.

23.4 test_iomngr

This option activates a driver for the I/O manager module in file iomngr.F. Refer to Section 15.1.4 for details.

[1] It is no longer required to construct density coefficients before executing the model.

23.5 test_poisson

This option activates a driver for the poisson module in file poisson.F. For more details refer to Sectionsubsection:poisson.F.

23.6 test_vmix

This option activates a driver for testing the vertical mixing scheme enabled by option *ppvmix* or *kppvmix*. Refer to Section 15.1.6 for details.

23.7 test_rotation

The model grid may be rotated using a set of Euler angles for solid body rotation. The Euler angles are computed by defining the geographic latitude and longitude of the rotated north pole and a point on the prime meridian. This option activates the driver which gives results from a sample rotation of the model grid. It indicates how scalars and vectors are interpolated and rotated before being used on the rotated model grid. Refer to Section 25.2.1 for details.

23.8 test_timeinterp

This option activates a driver for the the time interpolation module in file timeinterp.F. Refer to Section 15.1.7 for details.

23.9 test_timer

This option activates a driver for the timing module in file timer.F. For more details refer to Sectionsubsection:timer.F.

23.10 test_tmngr

This option activates a driver for the time manager module in file tmngr.F. Refer to Section 15.1.9 for details.

23.11 drive_topog

This option activates a driver for the topography and geometry module in file topog.F. Refer to Section 15.1.10 for details.

23.12 test_util

This option activates a driver for the utility module in file util.F. Refer to Section 15.1.11 for details.

Chapter 24

Options for the computational environment

24.1 Computer platform

Options are used to invoke minor changes in MOM which are computer platform specific thereby enabling MOM to execute on various platforms. These changes tend to be concentrated in the I/O manager, timing, and NetCDF routines. Only one platform must be specified.

24.1.1 cray_ymp

This option configures MOM for the CRAY YMP which no longer exists at GFDL.

24.1.2 cray_c90

This option configures MOM for the CRAY C90 which no longer exists at GFDL.

24.1.3 cray_t90

This option configures MOM for the CRAY T90 which is the current computational workhorse at GFDL. The operating system is Unicos 9.1.0.1 or later which does not have a "cf77" compiler. Therefore all scripts assume an Fortran 90 compiler.

24.1.4 cray_t3e

This option configures MOM for the CRAY T3e at GFDL and is necessary to insure the variables on the heaps are aligned for communication purposes.

24.1.5 sgi

This option configures MOM for Silicon graphics workstations at GFDL. Fortran 90 is assumed.

24.2 Compilers

The intent is for an Fortran 90 compiler to be used in all scripts.

24.3 Dataflow I/O Options

Various options for handling I/O between disk and the memory window are available. Each has its advantages and disadvantages as described in the following sections. One and only one of these options must be enabled. It should be noted that regardless of which option is used, details are implemented at the lowest level routines. This allows the higher level code structure (where reading and writing takes place) to remain the same for all options.

Characteristics for each file are controlled by specifying file attributes (i.e. whether the file is sequential, direct access, cray wordio, unformatted, etc.) in a character string which is passed into the I/O manager *iomngr.F*. The I/O manager then assigns a unit number for each file and opens the file with the specified attributes. Refer to Section 15.1.4 for details on the I/O manager.

Otions relating to I/O are set at compile time with UNIX "cpp" directives of the form -Doption. The available I/O options are "crayio", "fio", "ramdrive" or "ssread_swrite" and are described below. The platform options are "cray_ymp", cray_c90", "cray_t90", or "sgi" and are described in Section 24.1.

The I/O within MOM occurs during initialization, integration, diagnostic analysis, and shutdown. Initialization consists of reading namelist files, and possibly a restart file, boundary condition data, and data for sponge zones along artificial boundaries. The namelist files have the attributes "formatted sequential" and these files allow an easy way to override default settings of coefficients and miscellaneous variables (i.e. mixing coefficients, time step lengths, etc). They only amount to O(100) words and are therefore unimportant as far as efficiency is concerned. The biggest part of initialization is the reading of a restart file which may be huge (it contains two time levels of prognostic data plus the topography mask). The restart file has the attributes "unformatted sequential" and on CRAY systems is buffered through the "cachea layer" by an assign issued from within the I/O manager. The size of the buffer is calculated within MOM and passed within the attribute character string. The attributes given to boundary condition and sponge data are similar but contain the additional attribute " ieee". The "ieee" attribute specifies that the data was created as 32 bit IEEE. The large initialization files are buffered through the "cachea layer" on CRAY systems.

Prognostic data from the restart file is placed on "tau files" which take various forms depending on whether option "ramdrive", "crayio", "fio", or "ssread_sswrite" was enabled. The "tau files" are used during the integtration phase. By default, all of these options (except for ramdrive) use attribute "sds" to instruct the I/O manager to put the files on CRAY Solid state disk. The assign statements in the I/O manager can be changed to put the files on rotating disk if desired (although this is very inefficient compared with solid state disk).

During the integration phase, diagnostic data is also written out. It may be of two forms: Either 32bit IEEE or NetCDF. The recommended way is NetCDF which is specified by option "netcdf". The 32bit IEEE form is older and has not been extended to use buffering. The shutdown phase basically writes the last image of the "tau files" onto a restart file with the previously specified characteristics.

24.3.1 ramdrive

This one is similar in concept to the usage of the word *ramdrive* in the IBM PC world. It concerns configuring a portion of memory to behave like disk which speeds up programs that are I/O intensive particularly when disks are relatively slow. The catch is that there must be enough memory available to contain the disk data. In MOM, this option defines a huge

memory array and replaces all reads and writes to disk with copying variables into or out of locations in this huge array. This is done at the lowest subroutine levels so that the higher level routines look the same regardless of whether this I/O option is used or not. This option is not specific to any computer platform and therefore makes MOM highly portable.

24.3.2 crayio

This option uses the CRAY specific *wordio* package to read and write data to and from disk. As currently configured, option *crayio* directs the *iomngr* to assign scatch files to CRAY solid state disk. So solid state disk is assumed. If the "sds" attribute is removed from the file specification, then the files will reside on rotating disk. Script *run_iomngr* executes the driver in *iomngr.F* to exercise this form of I/O in a simple environment.

24.3.3 ssread_sswrite

This option uses the CRAY specific *ssread/sswrite* package to read and write data to and from solid state disk. Solid state disk is assumed and this is the fastest way to more data between solid state disk and memory. The performance is comparable to option *ramdrive*. All I/O is done in blocks of 512 words and all blocking and buffering is done in *iomngr.F* and *odam.F*. On the CRAY YMP, option *crayio* was sufficient for good performance. On the CRAY T90 it is not. Script *run_iomngr* executes the driver in *iomngr.F* to exercise this form of I/O in a simple environment.

24.3.4 fio

This option uses Fortran direct access I/O to read and write data to and from disk. It works on CRAY, SGI workstations and presumably any other platform which supports Fortran direct access I/O. Record size is specified in words which is converted to bytes within the *iomngr*. If solid state disk is not availbale, this option is very inefficient. However, it does allow the model to run on non-CRAY systems which do not have enough memory to use option *ramdrive* on a particular problem size. Script *run_iomngr* executes the driver in *iomngr.F* to exercise this form of I/O in a simple environment.

24.4 Parallelization

Currently, there is only one way to do multi-tasking in MOM as discussed in Chapter 12.

24.4.1 parallel_1d

Option *parallel_1d* is automatically added to the option list in the run_mom_script if the number of processors is set greater than one. This option enforces a high level parallelism as described in Section 12. When option *parallel_1d* is enabled, the message passing toolkit "mpt" must be available on CRAY T90 platforms. All communication is in terms of Cray specific "shmem" calls because it is faster than MPI or PVM on Cray platforms. When executing on the CRAY T3E, option *cray_t3e* must be enabled.

Chapter 25

Options for grid, geometry and topography

25.1 Grid generation

How to design grids is detailed in Chapter 16. The options for implementing these ideas are summarized below.

25.1.1 drive_grids

Script *run_grids* uses option *drive_grids* to execute the grids module in "stand alone mode". This is the recommended method for designing grids. Option *drive_grids* is not appropriate when executing a model because the model itself becomes the driver.

25.1.2 generate_a_grid

This option is used for generating a grid domain and resolution as opposed to importing one. Either this one or option *read_my_grid* must be enabled.

25.1.3 read_my_grid

This is the option to use when importing a domain and resolution generated from outside of the MOM environment. Either this one or option *generate_a_grid* must be enabled.

25.1.4 write_my_grid

For exporting a domain and resolution generated by module *grids*.

25.1.5 centered_t

This is for constructing a grid by method 1 as described in Chapter 16. If this option is <u>not</u> enabled, grid construction is by method 2 which is the default method for MOM.

25.2 Grid Transformations

25.2.1 rot_grid

Option *rot_grid* allows the grid system defined by module *grids* to be rotated so that the pole is outside the area of interest. Refer to Chapter 17 for details.

25.3 Topography and geometry generation

The options available for constructing topography and geometry are covered in detail in Chapter 18 and summarized below. One and only one of the following options must be enabled: *rectangular_box*, *idealized_kmt*, *gaussian_kmt*, *scripps_kmt*, *etopo_kmt*, or *read_my_kmt*. All others are optional.

25.3.1 rectangular_box

This option constructs a flat bottomed rectangular domain on the grid resolution specified by module *grids*. All interior ocean $kmt_{i,jrow}$ are set to level *km*. If used with option *cyclic* it configures a zonally re-entrant channel. If used with option *solid_walls* then all boundary $kmt_{i,jrow}$ cells are land cells.

25.3.2 idealized_kmt

This option constructs an idealized map of the world's geometry with a bottom depth that is not realistic. It requires no data and is highly portable. The geometry and topography are interpolated to the resolution specified by module *grids*.

This option is also a good starting point for investigations where idealized geometries need to be constructed. Other idealized geometries can be readily constructed with the aid of subroutine *setkmt* which approximates continental shapes by filling in trapezoidal areas of $kmt_{i,jrow}$ with *zero*. All that is required for input are the grid point coordinates, the coordinates of four vertices of a trapezoid, and a value for setting /kmt/ within the trapezoid.

25.3.3 gaussian_kmt

This option constructs a $kmt_{i,jrow}$ field with a gaussian bump superimposed on a sloping ocean floor in a rectangular basin without geometry. The slope of the floor and scale of the bump can be set by modifying the USER INPUT section under the gaussian bump section in module *topog.F*. This option requires no data and is therefore highly portable. Results are interpolated to the resolution specified by module *grids*. Refer to Section 18.2 for more details.

25.3.4 scripps_kmt

This generates geometry and topography interpolated from SCRIPPS data (in the DATABASE described in Section 3.2) to the resolution specified by module *grids*.

25.3.5 etopo_kmt

This generates geometry and topography interpolated from a 1/12° x 1/12° dataset[1] to the resolution specified by module *grids*.

25.3.6 read_my_kmt

This option imports a topography and geometry $kmt_{i,jrow}$ field from outside of the MOM environment or one produced from option *write_my_kmt*.

25.3.7 write_my_kmt

This option exports a topography and geometry $kmt_{i,jrow}$ field by writing it to disk.

25.3.8 flat_bottom

To deepen all ocean $kmt_{i,jrow}$ cells to the maximum number of levels given by level *km*, enable this option. It works with option *idealized_kmt* or *scripps_kmt*.

25.3.9 fill_isolated_cells

This option converts all isolated ocean T cells into land cells. An ocean T cell $T_{i,k,j}$ is isolated if the four surrounding U cells $U_{i,k,j}$, $U_{i-1,k,j}$, $U_{i-1,k,j-1}$, and $U_{i,k,j-1}$ at the same vertical level k are land cells. Potholes as well as trenches which are one T cell wide fall into this category.

25.3.10 fill_shallow

This option converts ocean cells to land cells in regions where the ocean depth is shallower than what is allowed.

25.3.11 deepen_shallow

This option increases the number of ocean cells in regions where the ocean depth is shallower than what is allowed.

25.3.12 round_shallow

This option rounds the number of ocean cells to zero or the minimum number of allowable ocean levels in regions where the ocean depth is shallower than what is allowed.

25.3.13 fill_perimeter_violations

This option builds a land bridge between two distinct land masses which are seperated only by a distance of one ocean T cell. There must be at least two ocean T cells separating distinct land masses.

[1] This dataset may be purchased from the Marine Geology and Geophysics Division of the National Geophysical Data Center and is not included in the MOM DATABASE.

25.3.14 widen_perimeter_violations

This option removes land cells between two distinct land masses which are seperated only by a distance of one ocean T cell. There must be at least two ocean T cells separating distinct land masses.

Chapter 26

Partial Bottom Cells

Traditionally, all cells within a vertical level in MOM have the same vertical thickness. Option *partial_cell* allows bottom ocean cells within any given level to be of different vertical thickness so as to more accurately resolve the physical bottom (Pacanowski and Gnanadesikan, 1998). This option must be used right from the beginning of an experiment. It is not intended as an option which can be enabled in the middle of an experiment. Nor is it intended to be disabled in the middle of an experiment. Either a model is constructed using full-cells (the default) or partial-cells (option *partial_cell*).

26.1 Motivation

Bottom topography is discretized to model levels as described in Chapter 18. Fig 26.1a is an example of a typical section of bottom topography compared with the resulting discretized bottom. The solid line represents the "true" bottom which the discretization is trying to capture. Ocean T-cells are indicated with grid points and land T-cells are shaded. The same bottom discretized with partial bottom cells is indicated in Fig 26.1b where the bottom T-cell in each column is allowed to be partially filled with land. Note that the topography is significantly more accurately approximated when partial cells are used. Most of the change in regions of steep slopes is captured by changes in the number of levels. Even so, partial cells give a better estimation of the true slope in these regions. In general, T-cell (and U-cell) thickness becomes a function of latitude, longitude and depth when partial cells are used. Note how the grid points follow the topography in Fig 26.1b. In effect, a one level "sigma coordinate" has been added by the partial cells.

Refer to Fig 26.2 which indicates what happens when horizontal resolution in Fig 26.1 is doubled but vertical resolution remains unchanged. Comparing Fig 26.1a with Fig 26.2a indicates that discretized bottom topography using full cells does not significantly improve when horizontal resolution is increased. In fact, it remains about the same. However, comparing Fig 26.1b with Fig 26.2b, the improvement using partial cells is significant.

In addition to ocean volume and f/H being more accurately represented by partial-cells, topographic wave speeds are also more accurate and therefore so is the dispersion relation for topographic waves as discusssed in Pacanowski and Gnanadesikan (1997). The disadvantage of partial-cells is that the integration time is 10 to 15% longer than for full-cells for the same number of vertical levels. However, to achieve the same accuracy, partial-cells are significantly more computationally efficient than using full-cells and increasing the number of vertical lev-

els[1]. There is also a pressure gradient error with partial-cells but this is tiny as discussed in Pacanowski and Gnanadesikan (1997). In the interior of the ocean, equations are second order accurate (Treguier et al., 1996) but reduce to first order at bottom boundaries because the thickness of partial-cells breaks the analytical stretching of grid cell thickness in the vertical. Although equations drop from second order in the interior to first order for non full-cells at the bottom, a leading order error in the position of the topography has been corrected. Globally, the solution remains second order accurate.

26.2 Discrete Equations

The essential point is to note that part of the northern, southern, eastern, or western face of a grid cell may be partially land and this must be taken into account when fluxing information across cell faces. The discrete equations in Section 21.3 must be altered to account for vertical cell thickness which is a function of λ, ϕ, and depth. In general, vertical cell thickness dzt_k is replaced by $dht_{i,k,j}$ which is the vertical thickness of T-cells and $dhu_{i,k,j}$ which is the vertical thickness of U-cells. The thickness of T-cells $dht_{i,k,j}$ is determined when bottom topography is discretized. Although ocean cells may be partial, all land cells are considered as full-cells. Since material surfaces are defined by faces of T-cells, it follows that

$$dhu_{i,k,j} = min\ (dht_{i,k,j},\ dht_{i+1,k,j},\ dht_{i,k,j+1},\ dht_{i+1,k,j+1}) \tag{26.1}$$

Refer Fig 26.4 depicts the relationship between partial bottom T-cells and U-cells.

26.2.1 Momentum equations

To account for a generalized variation in cell thickness, a factor of $dhu_{i,k,j}$ must be included in horizontal diffusive and advective operators. The following equations should be compared with the corresponding set in Section 21.3.1:

$$ADV_Ux_{i,k,j} = \frac{1}{2\cos\phi^U_{jrow}\, dhu_{i,k,j}} \delta_\lambda(adv_fe_{i-1,k,j}) \tag{26.2}$$

$$ADV_Uy_{i,k,j} = \frac{1}{2\cos\phi^U_{jrow}\, dhu_{i,k,j}} \delta_\phi(adv_fn_{i,k,j-1}) \tag{26.3}$$

$$DIFF_Ux_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}\, dhu_{i,k,j}} \delta_\lambda(diff_fe_{i-1,k,j}) \tag{26.4}$$

$$DIFF_Uy_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}\, dhu_{i,k,j}} \delta_\phi(diff_fn_{i,k,j-1}) \tag{26.5}$$

where the horizontal viscous fluxes on U-cell faces become

$$diff_fe_{i,k,j} = \frac{visc_ceu_{i,k,j}}{\cos\phi^U_{jrow}}\, min(dhu_{i,k,j}, dhu_{i+1,k,j})\ \delta_\lambda(u_{i,k,j,n,\tau-1}) \tag{26.6}$$

$$diff_fn_{i,k,j} = visc_cnu_{i,k,j} \cos\phi^T_{jrow+1}\, min(dhu_{i,k,j}, dhu_{i,k,j+1})\ \delta_\phi(u_{i,k,j,n,\tau-1}) \tag{26.7}$$

[1]Increasing the number of vertical levels does allow interior flows to be better resolved.

Because the lateral boundary condition is no-flux for tracers but no-slip for velocity, the horizontal viscosity $\frac{1}{h}\nabla \cdot (h\ \nabla(\vec{u}))$ has zonal and meridional components given by

$$
\begin{aligned}
\mathcal{VISC}^{\lambda} &= \frac{A_m}{dhu_{i,k,j}\cos^2\phi^U}\ \delta_\lambda(\zeta^{U\lambda}\ \delta_\lambda u)\ +\ \frac{A_m}{dhu_{i,k,j}\cos\phi^U}\ \delta_\phi(\cos\phi^T\ \zeta^{U\phi}\ \delta_\phi u) \\
&+\ A_m\frac{1-\tan^2\phi^U}{a^2}\ u\ -\ A_m\frac{2\sin\phi^U}{a^2\ \cos^2\phi^U}\ \delta_\lambda(\overline{v}^\lambda) \\
&+\ S(u) \qquad (26.8) \\
\mathcal{VISC}^{\phi} &= \frac{A_m}{dhu_{i,k,j}\cos^2\phi^U}\ \delta_\lambda(\zeta^{U\lambda}\ \delta_\lambda v)\ +\ \frac{A_m}{dhu_{i,k,j}\cos\phi^U}\ \delta_\phi(\cos\phi^T\ \zeta^{U\phi}\ \delta_\phi v) \\
&+\ A_m\frac{1-\tan^2\phi^U}{a^2}\ v + A_m\frac{2\sin\phi^U}{a^2\ \cos^2\phi^U}\ \delta_\lambda(\overline{u}^\lambda) \\
&+\ S(v)\ . \qquad (26.9)
\end{aligned}
$$

where A_m is the lateral viscosity coefficient[2]. The above form differs from Bryan (1969) due to the inclusion of effective cell face heights $\zeta^{U\lambda}$ and $\zeta^{U\phi}$ and a sink term due to a no-slip lateral boundary condition given by

$$
\begin{aligned}
\zeta^{U\lambda}_{i,k,j} &= min(dhu_{i,k,j},\ dhu_{i+1,k,j}) \qquad (26.10) \\
\zeta^{U\phi}_{i,k,j} &= min(dhu_{i,k,j},\ dhu_{i,k,j+1}) \qquad (26.11) \\
S(\beta) &= -\frac{A_m}{dhu_{i,k,j}\cos^2\phi^U dxu_i}\left[\frac{dhu_{i,k,j}-\zeta^{U\lambda}}{dxt_{i+1}} + \frac{dhu_{i,k,j}-\zeta^{U\lambda}_{i-1}}{dxt_i}\right]\beta \\
&- \frac{A_m}{dhu_{i,k,j}\cos\phi^U dyu_{jrow}}\left[\frac{\cos\phi^T_{j+1}(dhu_{i,k,j}-\zeta^{U\phi})}{dyt_{jrow+1}} + \frac{\cos\phi^T(dhu_{i,k,j}-\zeta^{U\phi}_{j-1})}{dyt_{jrow}}\right]\beta \qquad (26.12)
\end{aligned}
$$

which is zero where U-cell thickness is constant (i.e. $dhu_{i,k,j} = \zeta^{U\lambda}$) within a vertical level but acts effectively as a bottom drag

$$
S(\beta) \propto (A_m/\Delta^2 x^U)\ \beta \quad \cdots\ assuming\ \ dxu_i = dyu_{jrow} \qquad (26.13)
$$

where U-cell thickness varies within a vertical level. The constant of proportionality $((dhu_{i,k,j} - \zeta^{U\lambda})/dhu_{i,k,j})$ represents the fraction of cell face height over which a no-slip condition is applied.

There is no change in the form of the horizontal advective flux given by Equations (22.86) and (22.87) because advective velocities $adv_vnt_{i,k,j}$ and $adv_vet_{i,k,j}$ absorb a factor $dhu_{i,k,j}$ and become advective transports given by Equations (26.40) and (26.41). Horizontal advective velocities for the U-cells are then given by Equations (22.22) and (22.21).

The only change in the vertical advective and vertical diffusive operators is to replace dzt_k in the vertical derivative with $dhu_{i,k,j}$. When defining vertical fluxes at the bottom cell faces, dhw_k in the vertical derivatives is replaced by $min\ (dhut_{i,k,j},\ dhut_{i+1,k,j},\ dhut_{i,k,j+1},\ dhut_{i+1,k,j+1})$.

[2]The formulation is for constant A_m. Variable A_m requires additional terms.

26.2.2 Pressure gradient

When vertical thickness of cells is a function of λ, ϕ, and depth, extra terms are needed to construct the pressure gradient. Refer to Pacanowski and Gnanadesikan for a discussion of the pressure gradient. Another way to look at it is the following: With the aid of Leibnitz's Rule (Equation (4.65), the zonal and meridional pressure gradients are written as

$$-\frac{grav}{\rho_\circ a \cdot \cos\phi}\int_z^0 \frac{\partial\rho}{\partial\lambda}dz' = -\frac{grav}{\rho_\circ a \cdot \cos\phi}\Big(\frac{\partial}{\partial\lambda}\big(\int_z^0 \rho\, dz'\big) - \rho\frac{\partial z}{\partial\lambda}\Big) \tag{26.14}$$

$$-\frac{grav}{\rho_\circ a}\int_z^0 \frac{\partial\rho}{\partial\phi}dz' = -\frac{grav}{\rho_\circ a}\Big(\frac{\partial}{\partial\phi}\big(\int_z^0 \rho\, dz'\big) - \rho\frac{\partial z}{\partial\phi}\Big) \tag{26.15}$$

where ρ is density and the acceleration due to gravity is $grav = 980.6$ cm$/sec^2$. Let $\mathcal{P}_x(k)$ symbolize the the discrete form for the right hand side of the zonal pressure gradient in Equation (26.14) and $\mathcal{P}_y(k)$ symbolize the discrete form for the right hand side of the meridional pressure gradient in Equation (26.15). The discrete forms are

$$\mathcal{P}_x(k) = -\frac{1}{\rho_\circ \cos\phi^U_{jrow}}\Big(\overline{\delta_\lambda(p_{i,k,j}) - grav\cdot\overline{\rho_{i,k,j}}^\lambda\cdot\delta_\lambda(zt_{i,k,j})}^\phi\Big) \tag{26.16}$$

$$\mathcal{P}_y(k) = -\frac{1}{\rho_\circ}\Big(\overline{\delta_\phi(p_{i,k,j}) - grav\cdot\overline{\rho_{i,k,j}}^\phi\cdot\delta_\phi(zt_{i,k,j})}^\lambda\Big) \tag{26.17}$$

where the $\overline{()}^\phi$ in $\mathcal{P}_x(k)$ and the $\overline{()}^\lambda$ in $\mathcal{P}_y(k)$ are used to move the results onto a U-cells. The pressure $p_{i,k,j}$ is defined on T-cells and is calculated by vertically integrating the density from the surface to the depth of the T-grid point at level "k".

$$p_{i,k,j} = grav\cdot\rho_{i,1,j}\cdot dhw_{i,0,j} + grav\cdot\sum_{m=2}^{k}\overline{\rho_{i,m-1,j}}^z\, dhw_{i,m-1,j} \tag{26.18}$$

where $dhw_{i,k,j}$ is the vertical distance between a T-grid point at level "k" and its neightbor at "k+1" for any coordinate index "i,j". In the first term, $dhw_{i,0,j}$ is the distance between the ocean surface and the grid point within level $k = 1$. Partial cells are only allowed for levels $k > 1$ and so $dhw_{i,0,j}$ is constant for all "i,j". Using Equation (26.18), the term $\delta_\lambda(p_{i,k,j})$ in $\mathcal{P}_x(k)$ can be written as

$$\delta_\lambda(p_{i,k,j}) = grav\cdot dhw_{i,0,j}\cdot\delta_\lambda(\rho_{i,1,j}) + grav\cdot\delta_\lambda\Big(\sum_{m=2}^{k}\overline{\rho_{i,m-1,j}}^z\, dhw_{i,m-1,j}\Big) \tag{26.19}$$

Refer to Fig 26.3 which illustrates two vertical columns of T-cells with one partial bottom cell at level "k". Consider the vertically stratified case where density is a function of depth only. This condition implies

$$\delta_\lambda(\rho_{i,k',j}) = \delta_\phi(\rho_{i,k',j}) = 0 \quad for\ \ k' < k \tag{26.20}$$

$$\overline{\rho_{i,k',j}}^\lambda = \overline{\rho_{i,k',j}}^\phi = \rho_{i,k',j} \quad for\ \ k' < k \tag{26.21}$$

which reduces Equation (26.19) to

$$\delta_\lambda(p_{i,k,j}) \quad = \quad grav \cdot \delta_\lambda\left(\overline{\rho_{i,k-1,j}}^z \, dhw_{i,k-1,j}\right) \tag{26.22}$$

Distributing the derivative and reducing further yields

$$\begin{aligned} \delta_\lambda(p_{i,k,j}) &= grav \cdot \left(\overline{\rho_{i,k-1,j}}^{z\lambda} \cdot \delta_\lambda(dhw_{i,k-1,j}) \; + \; \overline{dhw_{i,k-1,j}}^\lambda \cdot \delta_\lambda(\overline{\rho_{i,k-1,j}}^z)\right) \\ &= \frac{grav}{2} \cdot \left((\overline{\rho_{i,k-1,j}}^\lambda + \overline{\rho_{i,k,j}}^\lambda) \cdot \delta_\lambda(zt_{i,k,j}) \; + \; \overline{dhw_{i,k-1,j}}^\lambda \cdot \delta_\lambda(\rho_{i,k,j})\right) \end{aligned} \tag{26.23}$$

Substituting the above form for $\delta_\lambda(p_{i,k,j})$ into Equation (26.16) yields

$$\mathcal{P}_x(k) = -\frac{grav}{2\,\rho_\circ \cos\phi^U_{jrow}} \left(\overline{\overline{\rho_{i,k-1,j}}^\lambda \cdot \delta_\lambda(zt_{i,k,j}) + \overline{dhw_{i,k-1,j}}^\lambda \cdot \delta_\lambda(\rho_{i,k,j}) - \overline{\rho_{i,k,j}}^\lambda \cdot \delta_\lambda(zt_{i,k,j})}^{\phi}\right) \tag{26.24}$$

In general, density can be divided into two pieces: one that varies linearly with depth and another that varies non-linearly with depth. The non-linear part will lead to non-zero pressure gradients in the vicinity of partial cells. The linear portion of the density variation with depth goes to zero.

26.2.2.1 Example where density varies linearly with depth

If density varies linearly with depth, then

$$\delta_z(\rho_{i,k,j}) = constant \tag{26.25}$$

and density $\rho_{i,k,j}$ can be expanded in terms of density in the upper level $\rho_{i,k-1,j}$ as

$$\rho_{i,k,j} = \rho_{i,k-1,j} + \delta_z \rho_{i,k-1,j} \cdot dhw_{i,k-1,j} \; . \tag{26.26}$$

Using the above expansion to eliminate $\rho_{i,k,j}$ in Equation (26.24) results in the following expansions:

$$\delta_\lambda(\rho_{i,k,j}) \quad = \quad \delta_z(\rho_{i,k-1,j}) \cdot \delta_\lambda(zt_{i,k,j}) \tag{26.27}$$

$$\overline{\rho_{i,k,j}}^\lambda \quad = \quad \rho_{i,k-1,j} + \delta_z(\rho_{i,k-1,j}) \cdot \overline{dhw_{i,k-1,j}}^\lambda \; . \tag{26.28}$$

Substituting the above expansions into Equation (26.24) results in

$$\mathcal{P}_x(k) = 0 \tag{26.29}$$

26.2.2.2 Computing density in partial bottom cells

The dependence of density on pressure is linear below a few hundred meters. As was shown above, there are no pressure gradients induced by partial bottom cells due to linear variations in density with depth. With a linear equation of state, bottom flows induced by partial bottom cells are within machine roundoff. Density anomoly using a non-linear equation of state as given by Equation (15.7) is not correct at grid points within partial cells because the polynomial coefficients, reference temperatures and reference salinities are defined only at the depth of grid points within discrete model levels. Effectively, partial bottom cells imply an infinite number of vertical levels. Density anomoly at the grid point within a partial cell can be approximated by linearly interpolating the polynomial coefficients, reference temperatures and reference salinities used in the equation of state (Equation 15.7) to the depth of the partial bottom cell grid point.

Without correcting the density anomoly for the depth of the partial bottom cells, a linear stratification of temperature in a non-linear equation of state generates error flows of about 1 cm/sec along topography (gaussian bump) in a 1° resolution model simulation. However, by accounting for the depth of the grid point within the polynomial approximation to the equation of state, the error flows are reduced by an order of magnitude.

26.2.3 Tracer equations

To account for a generalized variation in cell thickness, a factor of $dht_{i,k,j}$ must also be included in horizontal diffusive and advective operators for tracers. The following equations should be compared with the corresponding set in Section 21.3.2:

$$ADV_Tx_{i,k,j} = \frac{1}{2\cos\phi^T_{jrow}\, dht_{i,k,j}} \delta_\lambda(adv_fe_{i-1,k,j}) \tag{26.30}$$

$$ADV_Ty_{i,k,j} = \frac{1}{2\cos\phi^T_{jrow}\, dht_{i,k,j}} \delta_\phi(adv_fn_{i,k,j-1}) \tag{26.31}$$

$$DIFF_Tx_{i,k,j} = \frac{1}{\cos\phi^T_{jrow}\, dht_{i,k,j}} \delta_\lambda(diff_fe_{i-1,k,j}) \tag{26.32}$$

$$DIFF_Ty_{i,k,j} = \frac{1}{\cos\phi^T_{jrow}\, dht_{i,k,j}} \delta_\phi(diff_fn_{i,k,j-1}) \tag{26.33}$$

$$\tag{26.34}$$

The diffusive fluxes on T-cell faces are

$$diff_fe_{i,k,j} = \frac{diff_cet_{i,k,j}}{\cos\phi^T_{jrow}}\, min(dht_{i,k,j}, dht_{i+1,k,j})\ \delta_\lambda(t^{int}_{i,k,j,n,\tau-1}) \tag{26.35}$$

$$diff_fn_{i,k,j} = diff_cnt_{i,k,j} \cdot \cos\phi^U_{jrow}\, min(dht_{i,k,j}, dht_{i,k,j+1})\ \delta_\phi(t^{int}_{i,k,j,n,\tau-1}) \tag{26.36}$$

The superscript "int" in the tracer quantity denotes a linear interpolation in the vertical to the minimum depth of the two points being considered in the horizontal derivative. The form of the linear interpolation is

$$\begin{aligned}\delta_\lambda(t^{int}_{i,k,j,n,\tau-1}) = \quad & \Big(\quad t_{i+1,k,j,n,\tau-1} - min(dhwt_{i+1,k,j},\ dhwt_{i,k,j}) \cdot \frac{t_{i+1,k-1,j,n,\tau-1} - t_{i+1,k,j,n,\tau-1}}{dhwt_{i+1,k-1,j}} \\ & - \quad (t_{i,k,j,n,\tau-1} - min(dhwt_{i+1,k,j},\ dhwt_{i,k,j}) \cdot \frac{t_{i,k-1,j,n,\tau-1} - t_{i,k,j,n,\tau-1}}{dhwt_{i,k-1,j}}) \\ & \Big) \quad / dxu_i \end{aligned} \tag{26.37}$$

where $dhwt_{i,k,j}$ is the vertical distance between the grid point in cell $T_{i,k,j}$ and cell $T_{i,k+1,j}$.
If the above linear interpolation is not done, horizontal diffusion leads to an enhanced vertical diffusion in the vicinity of partial cells. This can easily be demonstrated by defining a temperature and salinity stratification which is only a linear function of depth. Horizontal diffusion should be zero. However, in the vicinity of partial cells, it will not be zero unless the above linear interpolation is used.

The form of the advective fluxes given by Equations (22.43) and (22.44) does not change. However, horizontal advective velocities $adv_vet_{i,k,j}$ and $adv_vet_{i,k,j}$ absorb a factor of $dhu_{i,k,j}$ to become horizontal advective transports given by Equations (26.40) and (26.41).

26.3 Conservation of energy

26.3.1 Changes in Kinetic energy due to partial bottom cells

When vertical thickness of cells is a function of λ, ϕ, and depth, extra terms are needed to construct the change in kinetic energy due to pressure forces. Multiplying the zonal pressure gradient given in Equation (26.14) by u and the meridional pressure gradient given in Equation (26.15) by v and integrating over the entire ocean volume yields the change in kinetic energy. The finite difference equivalent of this expression is

$$\begin{aligned} & -\frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} dvol_{i,k,j} \Bigg(u_{i,k,j,1,\tau} \frac{\delta_\lambda(\overline{p_{i,k,j}}^{\phi})}{\cos\phi^U_{jrow}} + u_{i,k,j,2,\tau} \cdot \delta_\phi(\overline{p_{i,k,j}}^{\lambda}) \\ - \quad & u_{i,k,j,1,\tau} \frac{grav\ \overline{\overline{\rho_{i.k.j}}^{\lambda} \delta_\lambda(zt_{i,k,j})}^{\phi}}{\cos\phi^U_{jrow}} - u_{i,k,j,2,\tau}\ grav\ \overline{\overline{\rho_{i.k.j}}^{\phi} \delta_\phi(zt_{i,k,j})}^{\lambda} \Bigg) \end{aligned} \tag{26.38}$$

where the velocity cell volume element is

$$dvol_{i,k,j} = dxu_i \cos\phi^U_{jrow}\ dyu_{jrow}\ dhu_{i,k,j}\ . \tag{26.39}$$

Note that there are four terms in Equation (26.38). The first two are similar to Equation (A.89) except that dzt_k in the U-cell volume element has been replaced by $dhu_{i,k,j}$ which is the vertical thickness of a U-cell as given by Equation (26.1).
The reduction of Equation (26.38) is similar to the one given for Equation (A.89) except for differences needed to account for spatial dependence of $dhu_{i,k,j}$. In Equation (A.91), dzt_k must be removed and $dhu_{i,k,j}$ is placed inside both derivatives with subscripts matching those on the velocity. In Equation (A.92), dzt_k is eliminated and $dhu_{i,k,j}$ is placed inside both averages with subscripts matching those on the velocity. The horizontal advective velocites become advective transports defined as

$$adv_vnt_{i,k,j} = \frac{\overline{u_{i-1,k,j,2,\tau} \cdot dxu_{i-1} \cdot dhu_{i-1,k,j}}^{\lambda}}{dxt_i} \cos\phi^U_{jrow} \tag{26.40}$$

$$adv_vet_{i,k,j} = \frac{\overline{u_{i,k,j-1,1,\tau} \cdot dyu_{jrow-1} \cdot dhu_{i,k,j-1}}^{\phi}}{dyt_{jrow}} \tag{26.41}$$

but the vertical advective velocities remain as velocities and the continuity equation becomes

$$\frac{adv_vet_{i,k,j} - adv_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} + \frac{adv_vnt_{i,k,j} - adv_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} + adv_vbt_{i,k-1,j} - adv_vbt_{i,k,j} = 0 \tag{26.42}$$

which leads to a change in kinetic energy given by

$$-\frac{grav}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} adv_vbt_{i,k-1,j} \cdot \overline{\rho_{i,k-1,j}}^{z}\, dxt_i \cos\phi^T_{jrow}\, dyt_{jrow}\, dhw_{i,k-1,j} \tag{26.43}$$

Note that thickness factor dzw_{k-1} in Equation (A.99) has been replaced by its spatially dependent counterpart $dhw_{i,k-1,j}$ which is the vertical distance between grid points within cells $T_{i,k,j}$ and $T_{i,k-1,j}$. The change in kinetic energy due to the third and fourth terms is given next.

26.3.2 Additional kinetic energy change due to boundary effects

After cancelling the cosine, the third term in Equation (26.38) is written as

$$-\frac{grav}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} u_{i,k,j,1,\tau}\, \overline{\overline{\rho_{i.k.j}}^{\lambda} \delta_\lambda(zt_{i,k,j})}^{\phi}\, dxu_i\, dyu_{jrow}\, dhu_{i,k,j} \tag{26.44}$$

where $zt_{i,k,j}$ is the depth from the ocean surface to the point within cell $T_{i,k,j}$. Using the latitudinal version of Equation (21.14) to re-arrange terms in the summation on "jrow" gives

$$-\frac{grav}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \overline{u_{i,k,j-1,1,\tau} \cdot dhu_{i,k,j-1} \cdot dyu_{jrow-1}}^{\phi} \cdot \overline{\rho_{i.k.j}}^{\lambda} \delta_\lambda(zt_{i,k,j})\, dxu_i \tag{26.45}$$

and substitution from Equation (26.41) yields

$$-\frac{grav}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} adv_vet_{i,k,j} \cdot \overline{\rho_{i,k,j}}^{\lambda} \cdot \delta_\lambda(zt_{i,k,j})\, dxu_i\, dyt_{jrow} \tag{26.46}$$

Again, using Equation (21.15) to re-arrange terms in the longitudinal summation on"i" yields

$$\frac{grav}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \delta_\lambda(adv_vet_{i-1,k,j} \cdot \overline{\rho_{i-1.k.j}}^{\lambda})\, zt_{i,k,j}\, dxt_i\, dyt_{jrow} \tag{26.47}$$

The fourth term in Equation (26.38) is

$$-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} u_{i,k,j,2,\tau}\cdot\overline{\overline{\rho_{i.k.j}}^{\phi}\ \delta_\phi(zt_{i,k,j})}^{\lambda}\ dxu_i\cos\phi^U_{jrow}\ dyu_{jrow}\ dhu_{i,k,j} \tag{26.48}$$

A similar manipulation as given for the third term above reduces this fourth term to

$$\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\delta_\phi(adv_vnt_{i,k,j-1}\overline{\rho_{i.k.j-1}}^{\phi})\ zt_{i,k,j}dxt_i\ dyt_{jrow} \tag{26.49}$$

Combining Equations (26.43), (26.47) and (26.49) gives the total change in kinetic energy due to pressure terms as

$$\begin{aligned}&\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\left(\delta_\lambda(adv_vet_{i-1,k,j}\overline{\rho_{i-1,k,j}}^{\lambda})+\delta_\phi(adv_vnt_{i,k,j-1}\overline{\rho_{i.k.j-1}}^{\phi})\right)\ zt_{i,k,j}\ dxt_i\ dyt_{jrow}\\&-adv_vbt_{i,k-1,j}\ \overline{\rho_{i,k-1,j}}^{z}\ dxt_i\cos\phi^T_{jrow}\ dyt_{jrow}\ dhw_{i,k-1,j}\end{aligned} \tag{26.50}$$

26.3.3 Changes in Potential energy due to partial bottom cells

When vertical thickness of bottom cells at a given depth is a function of λ and ϕ then extra terms are needed to construct the change in potential energy due to advection. As a consequence of absorbing a vertical grid cell thickness factor into the advective transports of Equations (26.41) and (26.40) , the advection operator for T-cells takes the form

$$\begin{aligned}\mathcal{L}^T(\alpha_{i,k,j}) &= \frac{\delta_\lambda(adv_vet_{i-1,k,j}\cdot\overline{\alpha_{i-1,k,j}}^{\lambda})+\delta_\phi(adv_vnt_{i,k,j-1}\cdot\overline{\alpha_{i,k,j-1}}^{\phi})}{dht_{i,k,j}\cos\phi^T_{jrow}}\\&+\ \delta_z(adv_vbt_{i,k-1,j}\cdot\overline{\alpha_{i-1,k,j}}^{\lambda})\end{aligned} \tag{26.51}$$

and Equation (A.100) is re-written as

$$\begin{aligned}&-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} zt_{i,k,j}\ (dxt_i\cos\phi^T_{jrow}\ dyt_{jrow}\ dht_{i,k,j})\times\\&\left(\frac{\delta_\lambda(adv_vet_{i-1,k,j}\cdot\overline{\rho_{i-1,k,j}}^{\lambda})+\delta_\phi(adv_vnt_{i,k,j-1}\cdot\overline{\rho_{i,k,j-1}}^{\phi})}{dht_{i,k,j}\cos\phi^T_{jrow}}\right.\\&+\ \left.\delta_z(adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z})\right)\end{aligned} \tag{26.52}$$

where zt_k has been replaced by $zt_{i,k,j}$ and dzt_k has been replaced by $dht_{i,k,j}$. Note the extra factor of $dht_{i,k,j}$ in the denominator. Because of these changes, the summation of the first two terms over a horizontal surface will not vanish. After some cancellations, the work done by horizontal advection of density is

$$-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} zt_{i,k,j}\, dxt_i\, dyt_{jrow} \times$$
$$\left(\delta_\lambda(adv_vet_{i-1,k,j}\cdot\overline{\rho_{i-1,k,j}}^{\lambda}) + \delta_\phi(adv_vnt_{i,k,j-1}\cdot\overline{\rho_{i,k,j-1}}^{\phi})\right). \tag{26.53}$$

which exactly compensates for the loss of kinetic energy due to the first two terms in Equation (26.50). The third term in Equation (26.52) may be re-arranged in the vertical summation to yield

$$\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z}\cdot dhw_{i,k-1,j}\, dxt_i \cos\phi^T_{jrow}\, dyt_{jrow} \tag{26.54}$$

which is the work due to buoyancy and exactly compensates for the remaining change in kinetic energy in Equation (26.50).

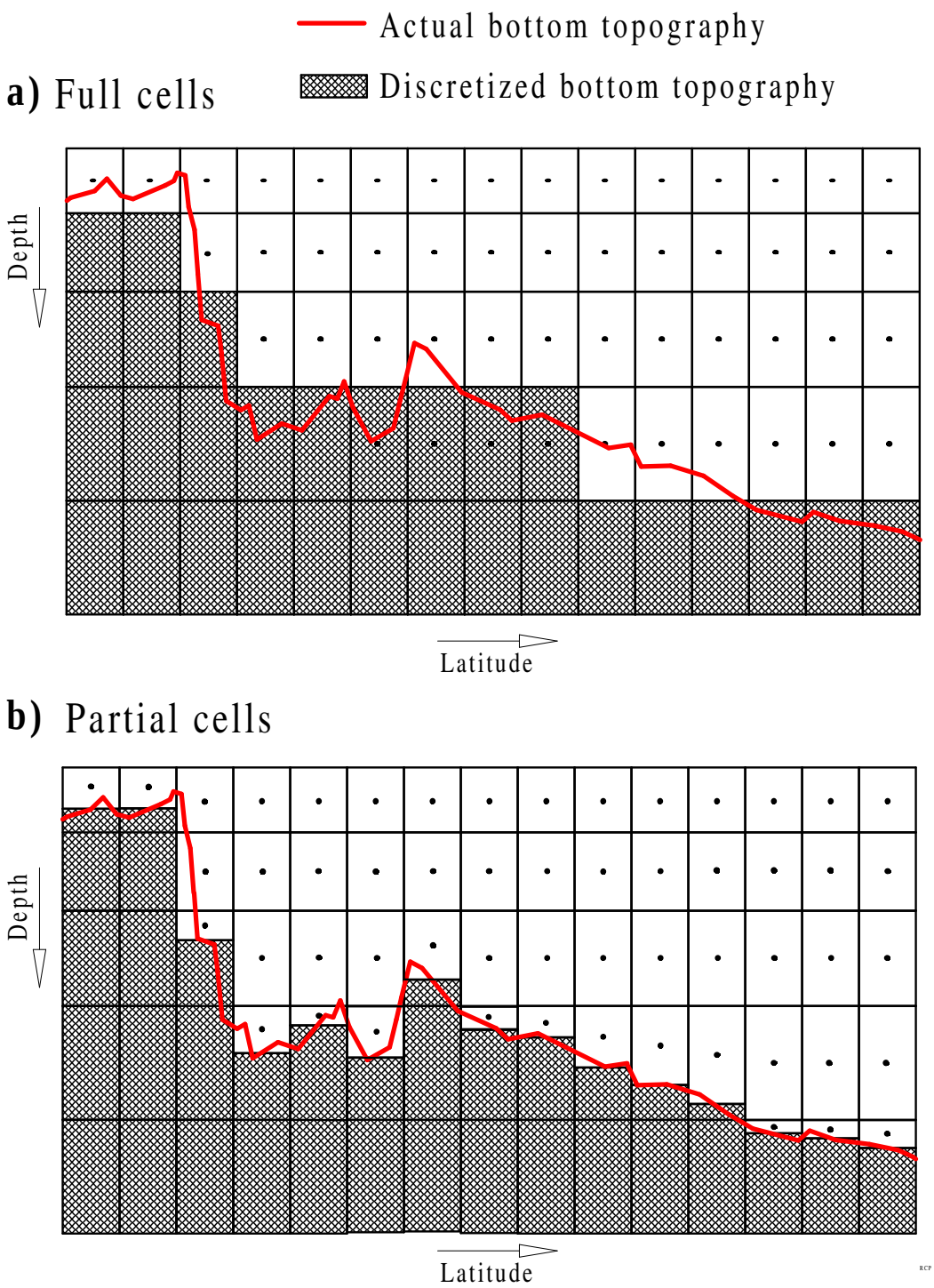


Figure 26.1: Comparing bottom topography for a 1x horizontal resolution a) Using full cells. b) Using partial bottom cells

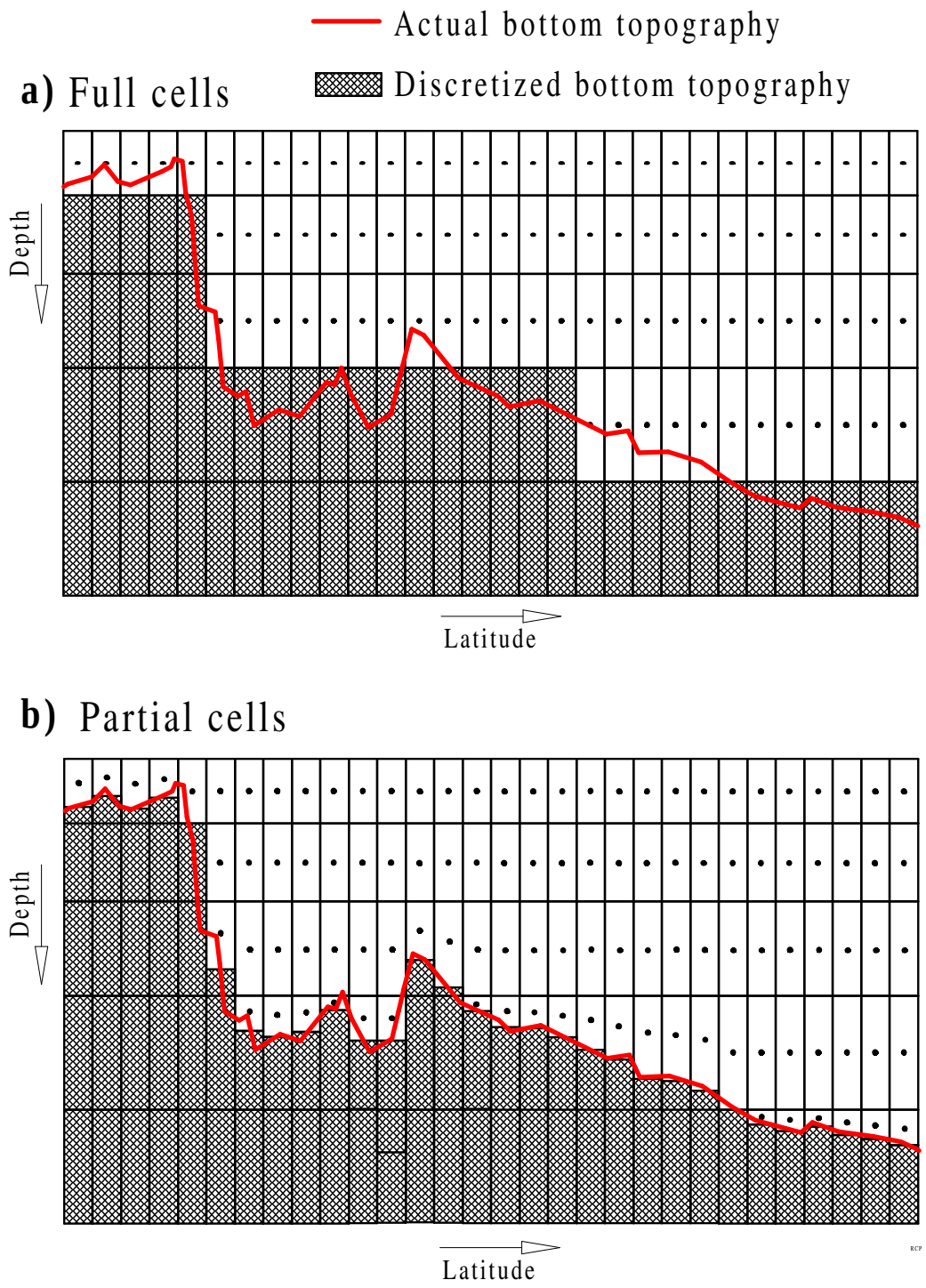


Figure 26.2: Comparing bottom topography for a 2x horizontal resolution a) Using full cells. b) Using partial bottom cells

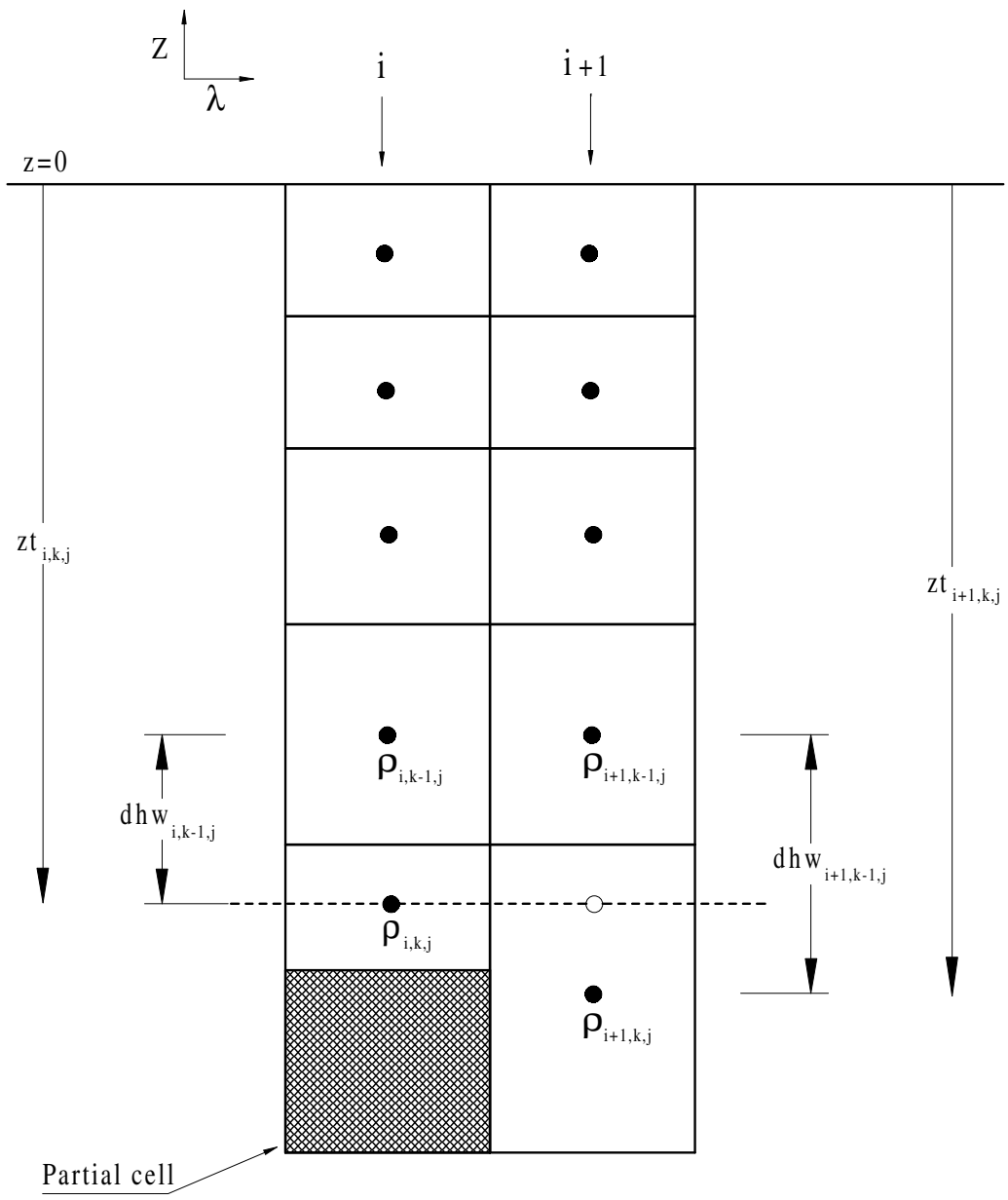


Figure 26.3: Two vertical columns of T-cells with one partial bottom cell

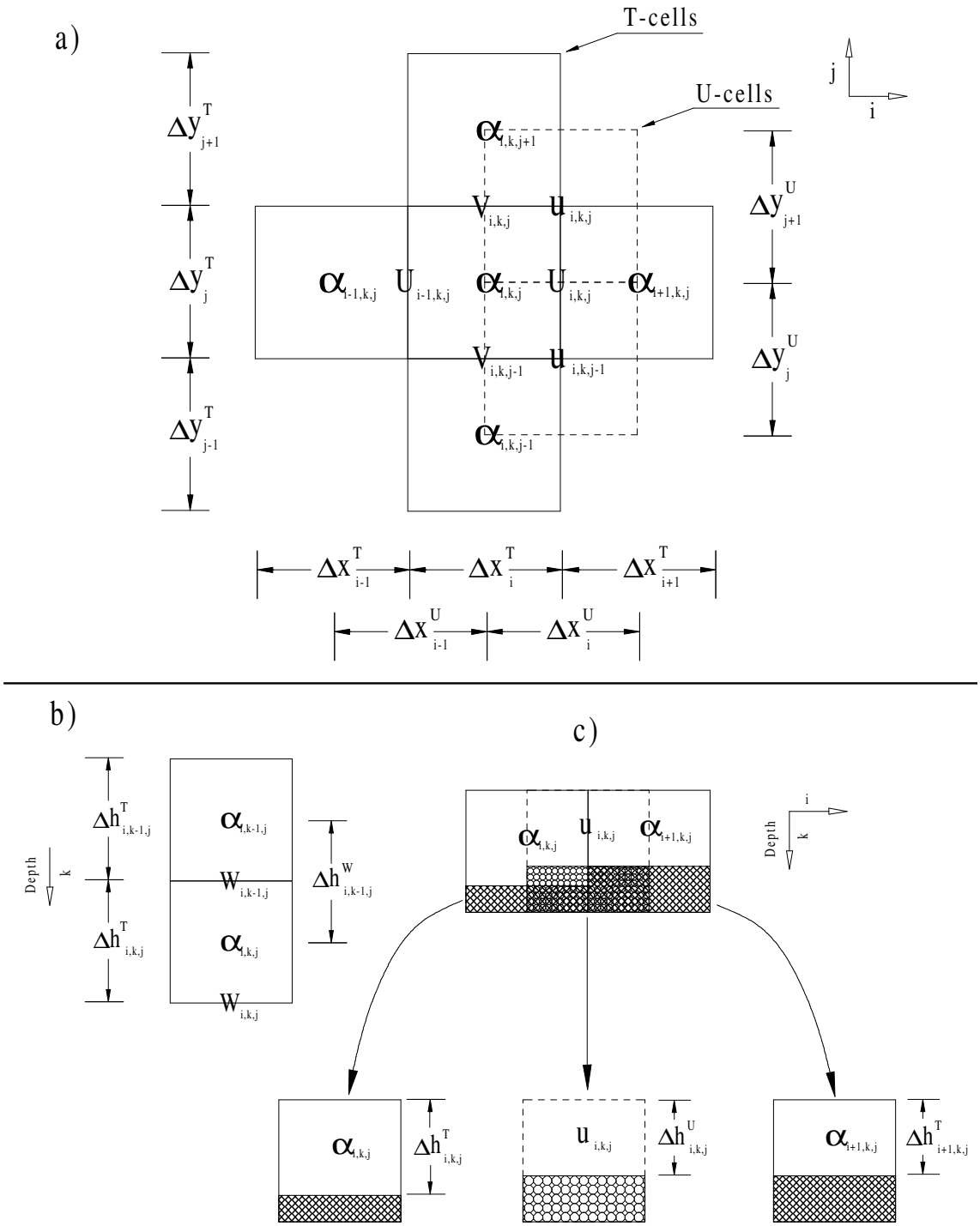


Figure 26.4: a) Horizontal arrangement of T-cells and U-cells. b) Two partial bottom T-cells with a partial bottom U-cell. In general, the partial bottom U-cell thickness is determined by the minimum thickness of four surrounding partial bottom T-cells.

Chapter 27

Filtering

In general, the equations of motion are filtered in various ways to remove components from the solution which are physically unimportant but nevertheless limit the length of the time step.

27.1 Convergence of meridians

Because the model is formulated in spherical coordinates, convergence of meridians near the earth's poles reduces the effective grid size in longitude. At high latitude, the solution may start to become unstable because of too large a time step. Instead of decreasing the time step, an alternative is to filter the solution at high (polar) latitudes to remove unstable wavenumber components. There are two filtering methods described below for suppressing these components. Option *fourfil* uses Fourier filtering and option *firfil* uses a finite impulse response filter to accomplish the same thing. Either one of these is typically applied polward of a reference latitude determined by the researcher. In the southern hemisphere, the reference latitude is given by *filter_reflat_s* and in the northern hemisphere, the reference latitude is given by *filter_reflat_n*. To save computation over the land mass of Antarctica, filtering is turned off southward of latitude *rjfrst*. These variables can be set through namelist. Refer to Section 14.4.6.

An alternative to filtering is to rotate the grid so that the convergence of meridians takes place outside to the model domain. Refer to Section 25.2.1 for details.

27.1.1 fourfil

Due to the convergence of meridians on a sphere, longitudinal grid resolution decreases toward zero as the poles are approached. For global domains this may severely limit the length of the time step due to the CFL constraint given in Section 14.4.4. The instability can be removed by filtering (Bryan, Manabe, and Pacanowski, 1975 and Takacs, Balgovind, 1983) the smallest scales out of the solution since they impose the most severe restriction on the time step.

Filtering is not to be encouraged and is not recommended unless absolutely necessary. When necessary, it should be restricted to the northernmost and southernmost polar latitudes in the domain. Typically, the filtering works on strips of latitude defined by the researcher. In the southern hemisphere, the reference latitude is given by *filter_reflat_s* and in the northern hemisphere, the reference latitude is given by *filter_reflat_n*. By default, these values are set to 70° N and 70° S latitude for the test case resolution. With higher resolution, these reference

latitudes can and should be pushed further poleward. To save computation over the land mass of Antarctica, filtering is turned off southward of variable *rjfrst* which is defaulted to 81° S. If needed, these variables can be set through namelist. Refer to Section 14.4.6.

Option *fourfil* does a Fourier smoothing of prognostic variables in the longitudinal direction by truncating wavenumbers larger than a critical one given by

$$N = im \frac{\cos \phi^T_{jrow}}{\cos filter_reflat_s} \tag{27.1}$$

where N is the number of waves to retain, im is the length of a strip of ocean data along the latitude given by $\cos \phi^T_{jrow}$, and $filter_reflat_s$ is a reference latitude row in the southern hemisphere ($filter_reflat_n$ would be used for the northern hemisphere.). If data were being filtered on the latitude of U-cells, then $\cos \phi^T_{jrow}$ would be replaced by $\cos \phi^U_{jrow}$. Because of geometry and topography break up each latitude circle, each filtering latitude is composed of strips defined by starting and stopping longitudes for ocean data and each strip is therefore a function of latitude and depth. There are a lot of different sized strips which rules out the use of an FFT. Also, FFT's are inherently square whereas for filtering purposes, fewer than im wavenumbers are generally needed. Instead, a fourier transform is used which has been optimized in prehistoric times. As such, it is difficult to understand[1] exactly how this fourier decomposition and systhesis works by looking at the code since no reference was left by the original designer (Mike Cox who is now deceased). In principle the filter is doing the following.

Any variable α_i where $i = 1, im$ can be decomposed and re-constructed in terms of a discrete Fourier transform given by

$$\tilde{\alpha}_i = A_\circ + \sum_{\ell=1}^{N/2} w_\ell A_\ell \cos \frac{2\pi i \ell}{im} + \sum_{\ell=1}^{N/2-1} w_\ell B_\ell \sin \frac{2\pi i \ell}{im} \tag{27.2}$$

where $\tilde{\alpha}_i$ is the filtered value of α composed of N wavenumbers ($N \leq im$) given by Equation (27.1) and the even and odd Fourier coefficients A_ℓ and B_ℓ are

$$A_\circ = \frac{1}{im} \sum_{i=1}^{im} \alpha_i \tag{27.3}$$

$$A_\ell = \frac{2}{im} \sum_{i=1}^{im} \alpha_i \cos \frac{2\pi i \ell}{im} \quad for\ \ell = 1, 2, \cdots \leq \frac{N}{2} - 1 \tag{27.4}$$

$$A_{im/2} = \frac{1}{im} \sum_{i=1}^{im} \alpha_i \cos(i\pi) \tag{27.5}$$

$$B_\ell = \frac{2}{im} \sum_{i=1}^{im} \alpha_i \sin \frac{2\pi i \ell}{im} \quad for\ \ell = 1, 2, \cdots \leq \frac{N}{2} - 1 \tag{27.6}$$

The window w_ℓ is given by the boxcar function

$$w_\ell = 1 \tag{27.7}$$

[1] If any researcher understands what is being done then please contact me. (rcp@gfdl.gov)

which can lead to oscillations when wavenumbers greater than N are truncated. These oscillations are known as the Gibbs effect which is particularly pronounced if energy exists at the truncation wavenumber N. The Gibbs effect can increase variance and this is a drawback. The Gibbs effect can be minimized by multiplying the wavenumbers by a window. A typical cosine window is

$$w_\ell = \cos((\pi/2)\ell/N). \tag{27.8}$$

Tracers are filtered using even Fourier coefficients (cosines) because the boundary condition at edges of the tracer strips is no-flux. Odd Fourier coefficients (sines) are used for velocity components because of the no-slip boundary condition at the edges of the velocity strips. Vorticity ($ztd_{i,jrow}$) is filtered with even Fourier coefficients[2]. Strips which completely encircle the earth are filtered using the full series of sines and cosines using a cyclic boundary condition. Any vector which is filtered must first remove the wavenumber which is due to the changing direction of a latitude circle. This is wavenumber one for all latitudes. For instance, the x-component of constant cross polar flow will exhibit a wavenumber one influence. After the filtering operation, this wavenumber one due to the changing direction of the latitude circle must be put back.

Apart from the Gibbs problem, another drawback of fourier filtering is that two strips of different length on the same latitude circle will contain different wavenumbers after filtering. The reason is that truncation wavenumber is a function of strip length and not total distance around the latitude circle. If the two strips were on adjacent latitudes, then taking the divergence of horizontal velocity might introduce ficticious vertical velocities (because of differing scales on the strips. Note that option *firfil* does not have this problem.). Also, fourier filtering can be a real time burner and FFT's are of little help because of the arbitrary strip widths and the need to enforce proper boundary conditions at the side walls (i.e. zero filling of land will not work). It should be noted that filtering a function with a fourier transform and the cosine window given above yields results very similar to the much faster finite impulse filter described below. Also, the finite impulse filter does not induce a ficticious vertical velocity because the filtering is not a function of strip size.

27.1.2 firfil

For the reasons given above, it may be desirable to filter the solution in polar latitudes. Option *firfil* uses multiple passes with a simple symmetric finite impulse response filter (Hamming 1977) to accomplish the filtering. Its advantage over fourier filtering with option *fourfil* is speed and there is also no ficticiously induced vertical velocity (described above) because the amount of filtering is not a function of strip size. The finite impluse filter is also designed to handle land points without the need for code to pull out the various strips as is needed for fourier filtering. Additionally, option *firfil* will not increase variance (and there is no introduction of negative values into a field that is all positive).

As with fourier filtering, the amount of filtering is a function of latitude and is controlled by the number of passes of the filter. The number of passes is arbitrairly given by

$$numflt = \frac{\cos filter_reflat_s}{\cos \phi^T_{jrow}} \tag{27.9}$$

[2]Strips for the vorticity do not include coastal ocean cells because of the island integrals.

where $filter_reflat_s$ is the southern reference latitude and *numflt* is the number of filter applications per prognostic variable per latitude and the $\cos\phi^T_{jrow}$ is the cosine of that latitude. If filtering in the northern hemisphere, then $filter_reflat_n$ would be used instead of $filter_reflat_s$. Variable *numflt* is for T cell latitudes and there is a corresponding *numflu* for the number of passes used for variables on U cell latitudes. Equation (27.9) is arbitrary and should be adjusted by the researcher to get the desired effect.

A finite impulse filter can be written as

$$\tilde{\alpha}_i = \sum_{\ell=-M}^{M} c_\ell \alpha_{i+\ell} \tag{27.10}$$

and its transfer function is

$$H(\omega) = c_\circ + 2\sum_{\ell=1}^{M} C_\ell \cos(\omega) \;\; for\ \omega = 0\ to\ \pi \tag{27.11}$$

A very simple one is used with $M = 1$ and weights given by

$$\tilde{\alpha}_i = \frac{1}{4}\alpha_{i-1} + \frac{1}{2}\alpha_i + \frac{1}{4}\alpha_{i+1} \tag{27.12}$$

The transfer function is

$$H(\omega) = \frac{1}{2}(1 + \cos(\omega)) \tag{27.13}$$

The effect of multiple applications of this filter can be easily calculated. If the filter is applied $numflt$ times then the transfer function is $H^{numflt}(\omega)$.

It is useful to demonstrate that the $M = 1$ filter (a "1-2-1" filter) does not change the zonal mean of a field, so long as the zonal grid spacing is uniform and the field has cyclic symmetry, or the field satisfies the no-flux condition on the zonal land/ocean boundaries. For this purpose, consider the zonal sum of the filtered field

$$\begin{aligned}
\langle\tilde{\alpha}\rangle_x &= \sum_{i=2}^{imt-1} \tilde{\alpha}_i \\
&= \sum_{i=2}^{imt-1} (\frac{1}{4}\alpha_{i-1} + \frac{1}{2}\alpha_i + \frac{1}{4}\alpha_{i+1}) \\
&= \frac{1}{2}\sum_{i=2}^{imt-1} \alpha_i + \frac{1}{4}\sum_{i=2}^{imt-1} (\alpha_{i-1} + \alpha_{i+1}).
\end{aligned} \tag{27.14}$$

The first term represents the zonal sum of the unfiltered field

$$\langle\alpha\rangle_x = \sum_{i=2}^{imt-1} \alpha_i. \tag{27.15}$$

The other two terms are almost this zonal sum, except for some boundary points

$$\begin{aligned}
\sum_{i=2}^{imt-1} \alpha_{i-1} &= \sum_{i=1}^{imt-2} \alpha_i \\
&= \alpha_1 - \alpha_{imt-1} + \sum_{i=2}^{imt-1} \alpha_i \\
&= \alpha_1 - \alpha_{imt-1} + \langle\alpha\rangle_x,
\end{aligned} \tag{27.16}$$

and likewise

$$\begin{aligned}\sum_{i=2}^{imt-1} \alpha_{i+1} &= \sum_{i=3}^{imt} \alpha_i \\ &= -\alpha_2 + \alpha_{imt} + \sum_{i=2}^{imt-1} \alpha_i \\ &= -\alpha_2 + \alpha_{imt} + \langle \alpha \rangle_x. \end{aligned} \tag{27.17}$$

Therefore, the zonal sum of the filtered field is given by

$$\langle \tilde{\alpha} \rangle_x = \langle \alpha \rangle_x + (\alpha_1 - \alpha_2 + \alpha_{imt} - \alpha_{imt-1}), \tag{27.18}$$

and the zonal sum is unchanged if the unfiltered field satisfies the zonally cyclic conditions:

$$\alpha_1 = \alpha_{imt-1} \tag{27.19}$$

$$\alpha_{imt} = \alpha_2, \tag{27.20}$$

or if the field satisfies the no-flux condition on the zonal walls

$$\alpha_1 = \alpha_2 \tag{27.21}$$

$$\alpha_{imt} = \alpha_{imt-1}. \tag{27.22}$$

27.1.3 An analysis of polar filtering

In the past, test case integrations for 5 years using a global domain with roughly 3° resolution and realistic geometry, topography, initial conditions, and forcing yielded results which were different depending on whether fourier filtering or finite impulse filtering was used. In general, results using the finite impulse filter were inferior to those using the fourier filter based on a comparison with an integration using very small time steps and no filtering. For many years, the reason for this result was unknown and thought to be due to a code bug. However, the reason was not due to a code bug. The result can be explained with the following argument.

Assume time is discretized as $t = n\Delta\tau$ where n is the time step. At time $t = 0$, assume that quantity $T_{n=0}$ is known. To predict ($T_{n=1}$), a change in T over the time step is computed as $\delta T_{0,1}$ (details are unimportant) and

$$T_{n=1} = T_{n=0} + \delta T_{0,1} \tag{27.23}$$

Then $T_{n=1}$ is filtered yielding

$$\tilde{T}_{n=1} = \tilde{T}_{n=0} + \delta \tilde{T}_{0,1} \tag{27.24}$$

To predict $T_{n=2}$, the next change in T is computed as $\delta T_{1,2}$ and

$$T_{n=2} = \tilde{T}_{n=1} + \delta T_{1,2} \tag{27.25}$$

After filtering, the result is

$$\tilde{T}_{n=2} = \tilde{\tilde{T}}_{n=1} + \delta \tilde{T}_{1,2} \tag{27.26}$$

which can be expanded to yield

$$\tilde{T}_{n=2} = \tilde{\tilde{T}}_{n=0} + \delta\tilde{\tilde{T}}_{0,1} + \delta\tilde{T}_{1,2} \tag{27.27}$$

Consider the term $\tilde{\tilde{T}}_{n=0}$. If a fourier filter is used with wavenumbers truncated above some critical cutoff, then two or more applications of this filter is the same as one application. The reason is that "m" applications of the filter is the same as multiplying "m" response functions together. However, since the response fuction of the simple finite impulse filter with weights (1/4, 1/2, 1/4) is a cosine, multiple applications yield successively more smoothing at lower wavenumbers.

An alternative to filtering prognostic variables is to filter the time tendencies. Using the above example, after $\delta T_{0,1}$ is computed, it can be filtered to yield $\delta\tilde{T}_{0,1}$. Then $\tilde{T}_{n=1}$ can be constructed as

$$\tilde{T}_{n=1} = T_{n=0} + \delta\tilde{T}_{0,1} \tag{27.28}$$

After computing and filtering to yield the next time tendency $\delta\tilde{T}_{1,2}$, the updated and filtered variable $\tilde{T}_{n=2}$ can be constructed as

$$\tilde{T}_{n=2} = \tilde{T}_{n=1} + \delta\tilde{T}_{1,2} \tag{27.29}$$

which can be expanded to yield

$$\tilde{T}_{n=2} = T_{n=0} + \delta\tilde{T}_{0,1} + \delta\tilde{T}_{1,2} \tag{27.30}$$

Comparing Equations 27.27 and 27.30, it can be seen that there are effectively fewer filter applications when time tendencies are filtered compared to when prognostic variables are filtered. Fourier filtering with a sharp cutoff gives essentially the same results using either method for the reason given above. However, the finite impulse filter will produce much less smoothing of long waves when time tendencies are filtered. Results using the explicit free surface and realistic forcing bear this out. For stability reasons within the explicit free surface scheme, it turns out that tracer tendencies and barotropic velocity tendencies can be filtered but baroclinic velocity must be filtered instead of the baroclinic tendency. Using the explicit free surface method and the above described filtering combination yields solutions which compare well with the unfiltered case. However, the filtering must be tunned to each geometric configuration.

27.1.4 Recommendation for tuning the polar filter

Tests have been carried out using the tuning method given below and the explicit free surface option. Similar tests have not been carried out with the stream function option. Use of the explicit free surface method is recommended. The following steps are recommended for tuning the polar filter. For reasons given above, the finite impulse response filter (-Dfirfil) is the recommended filter of choice.

- Make a short integration of a few months using the intended model configuration except shorten the density, baroclinic, and barotropic time steps so that no filtering is needed. Use the same time step for density and baroclinic velocity. Do not use any polar filtering and save a snapshot at the end of the integration.

- Set the filtering latitudes as far poleward as possible. In the test case, the polar filtering latitudes start at 70°. This is for the purpose of studying the interaction of filtering on shelf processes and is for testing purposes only. Based on testing, it may be desirable to move the filtering poleward of 80°. This will eliminate filtering in the southern hemisphere and keep the filter away from coastal shelf areas in the Arctic.

- Turn on filtering with the finite impulse filter (-Dfirfil) and repeat the run using the same small timesteps. Examine the differences in snapshot files. Change the filtering latitudes and number of filter applications (numflt and numflu in setocn.F) until results are acceptable. Testing in this way shows what damage is done by the filtering. If there is alot of small scale information in the initial density stratification, then the initial T and S will have to be filtered more to remove it. It is counter intuitive but if small scales are left in the initial conditions, then increasing the amount of polar filtering in the integration will cause the model to blow up faster. The reason is that when filtering tendencies, increasing the number of filtering passes kills off the small scales in the tendencies which prevents them from wiping out small scales in the initial conditions. These small scales persist and eventually contaminate the solution.

- Increase the time steps to the expected values consistant with filtering latitudes. (The model output will give an analysis of what time steps should be ok). Repeat the run. If it blows up, increase the filtering on tracers first (numflt). If it still blows up, increase the filtering on velocity (numflu).

27.2 Inertial period

Another limiting factor on the time step is the earth's inertial period. The time step for coarse global models (i.e. grid size greater than 5 deg) is severly restricted by having to resolve the inertial period near the poles. Option *damp_inertial_oscillation*, described below, filters the Coriolis term and removes this restriction.

27.2.1 damp_inertial_oscillation

Option *damp_inertial_oscillation* damps inertial oscillations by treating the Coriolis term semi-implicitly. Why treat the Coriolis terms semi-implicitly? It only makes sense in coarse resolution ($\Delta_x \geq 5°$) global models where the time step allowed by the CFL condition does not resolve the inertial period which is 1/2 day at the poles. Treating the Coriolis term semi-implicitly damps the inertial oscillation and allows a longer time step. For global models with $\Delta_x \leq 2°$, the time step allowed by the CFL condition is typically small enough (less than 2 hours) to resolve the inertial period at the poles and so semi-implicit treatment is not needed. Consider the simple system for inertial oscillations

$$u_t - fv = 0 \tag{27.31}$$
$$v_t + fu = 0 \tag{27.32}$$

where $f = 2\Omega \sin\phi$, u is zonal velocity, and v is meridional velocity. When discretizing and solving the Coriolis term explicitly, the solution is given by

$$u^{\tau+1} = u^{\tau-1} + 2\Delta\tau \cdot fv^{\tau} \tag{27.33}$$
$$v^{\tau+1} = v^{\tau-1} - 2\Delta\tau \cdot fu^{\tau} \tag{27.34}$$

When treating the Coriolis term semi-implicitly, an implicit Coriolis factor $0.5 \leq acor < 1$ is used and the system becomes

$$u^{\tau+1} - 2\Delta\tau \cdot acor \cdot fv^{\tau+1} = u^{\tau-1} + 2\Delta\tau \cdot (1 - acor) \cdot fv^{\tau-1} \tag{27.35}$$
$$v^{\tau+1} + 2\Delta\tau \cdot acor \cdot fu^{\tau+1} = v^{\tau-1} - 2\Delta\tau \cdot (1 - acor) \cdot fu^{\tau-1} \tag{27.36}$$

The solution, after a little algebra, is given by

$$u^{\tau+1} = u^{\tau-1} + 2\Delta\tau \cdot \frac{fv^{\tau-1} - (2\Delta\tau \cdot acor \cdot f) \cdot fu^{\tau-1}}{1 + (2\Delta\tau \cdot acor \cdot f)^2} \tag{27.37}$$
$$v^{\tau+1} = v^{\tau-1} - 2\Delta\tau \cdot \frac{fu^{\tau-1} + (2\Delta\tau \cdot acor \cdot f) \cdot fv^{\tau-1}}{1 + (2\Delta\tau \cdot acor \cdot f)^2} \tag{27.38}$$

Note that option *damp_inertial_oscillation* will also damp external Rossby waves.

Chapter 28

Initial and boundary conditions

This chapter documents the initial and boundary conditions offered by MOM.

28.1 Initial Conditions

Typically, at initial condition time, the model is at rest with a specified density distribution. This is accomplished by zeroing all velocities (internal and external mode) and setting potential temperature and salinity at each grid cell using one of the options described below. All passive tracers are initialized to unity in the first level ($k = 1$) and zero for all other levels. In principle, initial conditions could also come from a particular MOM restart file although this is not the intent of the following options. One and only one of these options must be enabled.

28.1.1 ideal_thermocline

An idealized equatorial thermocline from Philander and Pacanowski (1980) is approximated by a function dependent on depth but not latitude or longitude. Salinity is held constant at 35 ppt and the temperature profile is given by

$$\bar{t}_k = T_\circ \left[1 - \tanh\left(\frac{zt_k - H_\circ}{Z_\circ}\right)\right] + T_1 \left(1 - \frac{zt_k}{zt_{km}}\right) \tag{28.1}$$

where $H_\circ = 80\text{x}10^2$ cm, $Z_\circ = 30\text{x}10^2 cm$, $T_\circ = 7.5\ ^\circ C$ and $T_1 = 10.0\ ^\circ C$. The intent is to initialize idealized equatorial models. It is also useful to use option *sponges* which damps the solution back to Equation (28.1) along the northern and southern boundaries to kill off Kelvin waves. Simple idealized windstress can also be set using options *constant_taux* and *constant_tauy*. Refer to these options for more details.

28.1.2 ideal_pycnocline

This option prescribes a linear profile for salinity which increases with depth

$$s_k = (35 - s_{ok})/1000 \tag{28.2}$$

where

$$s_{ok} = s_1 (1 - 100\, h\, z/z_{bot}), \tag{28.3}$$

$h = 1/200$, $s_1 = 35.2$, and z_{bot} is the depth of the ocean. More sophisticated profiles can easily be prescribed through modifying this option.

28.1.3 idealized_ic

This constructs an idealized temperature and salinity distribution based on zonal averages of annual means from the Levitus (1982) data. The distribution is only a function of latitude and depth and is contained totally within MOM so no external datasets are needed. This option makes MOM easily portable to various computer platforms.
Recommendation: As a matter of strategy, it is recommended that any new model be first set up using option *idealized_ic* to eliminate problems which could potentially come from more realistic initial conditions. Only after the model sucessfully executes and the solution is judged reasonable should more realistic initial conditions be considered.

28.1.4 levitus_ic

This option accesses three dimensional temperature and salinity data which have previously been prepared by scripts in PREP_DATA for the resolution specified by module *grids* as discussed in Section 3.2. The particular month from the Levitus (1982) climatology which is to be used as initial conditions must be accessed by pathname from within script *run_mom*. The researcher should supply the pathname.

The flag values in the Levitus analyzed dataset have been replaced by values extrapolated from valid Levitus ocean points. The extrapolation was accomplished by solving the following two-dimensional elliptic boundary value problem on land points for each horizontal level within the Levitus dataset.

$$\nabla_h^2 T(\lambda, \phi) \quad = \quad 0 \qquad (\lambda, \phi) \in \text{land}, \tag{28.4}$$

The reason for the above was to provide data values everywhere within the dataset to ease the procedure of interpolating Levitus data to ocean points within arbitrary model grid resolutions. As long as the boundary between ocean and land cells in the model domain is not too different than the boundary in the Levitus dataset, the extrapolation yields reasonably good values. However, if model topography is modified by any means to produce ocean cells far from Levitus ocean points, the extrapolation may not be good enough. It is up to the researcher to verify that data is reasonable for particular configurations of model geometry and topography.

Ideally, the above procedure will not result in any extrema. The two-dimensional approach used in this algorithm can be limiting. For example, say one needs to prescribe data inside of a deep canyon for a high resolution model, or perhaps inside of a wide seamount whose top has been chopped off in order to smooth topography. In both cases, the two-dimensional approach will effectively extrapolate from values horizontally outside the canyon or seamount, through the rock, to the point of interest. No information from adjacent vertical points are used. A more complete algorithm would involve solving a three-dimensional Laplacian in order to incorporate vertical information as well. The three-dimensional approach has not been taken due to the added computational complexity.

The importance of these issues depends on how important the initial conditions are to the model experiment. For many climate models, initial conditions are almost irrelevant since the models are integrated for many thousands of years to equilibrium. For higher resolution models which are not integrated for so long, initial conditions can be important and something more general than the above procedure might be relevant.

28.2 Surface Boundary Conditions

As detailed in Chapter 19, surface boundary conditions are viewed as comming from a hierarchy of atmospheric models: the simplest of which are datasets of varying complexity and the most complicated ones are the GCM's. The following is a summary of the various options relating to surface boundary conditions with further details being left to Chapter 19. One and only one option of the following options must be enabled: *simple_sbc*, *time_mean_sbc_data*, *time_varying_sbc_data*, or *coupled*.

28.2.1 simple_sbc

These boundary conditions are based on zonal averages of Hellerman and Rosenstein annual mean windstress (1983). If option *restorst* is enabled, the surface temperature and salinity are damped back to initial conditions on a time scale given by *dampts* in days as given by Equation (28.23) in Section 28.2.9. This option together with options *idealized_ic* and *restorst* allow MOM to be easily portable to various computer platforms.
Recommendation: When setting up a new model for the first time, it is recommended that option *simple_sbc* be used. Only after verifying that results are reasonable should more suitable options be considered.

28.2.2 constant_taux

This option is meant to specify a constant windstress of *taux0* $dynes/cm^2$ in the zonal direction for idealized studies

$$\tau^{\lambda} = taux0. \tag{28.5}$$

It will replace the zonal windstress given by option *simple_sbc*. and its default value of $taux0 = -0.5$ can be changed via *namelist.mbcin*. Refer to Section 14.4 for discussion of namelist parameters.

28.2.3 constant_tauy

This option is meant to specify a constant windstress of *tauy0* $dynes/cm^2$ in the meridional direction for idealized studies

$$\tau^{\phi} = tauy0. \tag{28.6}$$

It will replace the meridional windstress given by option *simple_sbc*. Its default value of $tauy0 = 0.0$ can be changed via *namelist.mbcin*. Refer to Section 14.4 for discussion of namelist parameters.

28.2.4 analytic_zonal_winds

F. Bryan (1987) introduced an idealized wind stress which has been used in many subsequent studies (e.g., Weaver and Sarachik 1991)

$$\begin{aligned}
\tau^{\lambda} &= 0.2 - 0.8\sin(6\,|\phi|) - 0.5\,[1 - \tanh(10\,|\phi|)] - 0.5\left(1 - \tanh[10(\pi/2 - |\phi|)]\right) \\
&= -0.8\,[\sin(6\,|\phi|) + 1] + \frac{\tanh(5\pi - 10|\phi|) + \tanh(10|\phi|)}{2} \qquad (28.7) \\
\tau^{\phi} &= 0. \qquad (28.8)
\end{aligned}$$

The units are $dyne/cm^2$ for the wind stress, and the latitude ϕ is in radians. The curl of this wind stress is given by

$$\begin{aligned} \nabla \wedge \vec{\tau} &= -\hat{z}\, a^{-1}\, \partial_\phi\, \tau^\lambda \\ &= -\frac{\hat{z}}{a}\left[\frac{24}{5}\cos(6\phi) + 5\,\mathrm{sech}^2(5\pi - 10|\phi|) - 5\,\mathrm{sech}^2(10\phi)\right]. \end{aligned} \tag{28.9}$$

Figure 28.1 shows a plot of the wind stress and its curl.

28.2.5 linear_tstar

For restoring the surface buoyancy in idealized ocean models using temperature alone, it is often useful to employ a linear profile. The papers by Cox and Bryan (1984) and Cox (1985) used the density profile

$$\sigma^* = 23 + \frac{6|\phi|}{65} \tag{28.10}$$

where ϕ is latitude in degrees and $\sigma = 1000\,(\rho - 1)$. The profile is symmetric across the equator. For MOM, it is necessary to translate this density profile into a temperature. For this purpose, assume a linear equation of state as discussed in Sections 15.1.2.4 and 15.1.2.4

$$\rho = \rho_o\,[1 - \alpha\,(T - T_o)], \tag{28.11}$$

which implies

$$T^* = T_o + \frac{\sigma_o - \sigma^*}{1000\,\alpha\rho_o}. \tag{28.12}$$

The following reference values are chosen (see Appendix 3 of Gill 1982)

$$T_o = 19^\circ C \tag{28.13}$$
$$S_o = 35\text{psu} \tag{28.14}$$
$$p_o = 0\text{dbar} \tag{28.15}$$
$$\rho_o = 1.025022\ g/cm^3 \tag{28.16}$$
$$\sigma_o = 1000\,(\rho_o - 1) = 25.022 \tag{28.17}$$
$$\alpha = 2489 \times 10^{-7}\ {}^\circ K^{-1}. \tag{28.18}$$

As a result,

$$T^* = 26.93 - .3618\,|\phi|, \tag{28.19}$$

where again, ϕ is in degrees latitude, and T^* is in Celcius. With σ^* given by (28.10), the restoring temperature takes the values

$$T^*(\phi = 0) = 26.9^\circ C \tag{28.20}$$
$$T^*(\phi = 65) = 3.4^\circ C. \tag{28.21}$$

Figure 28.2 shows the temperature profile across the northern hemisphere obtained with this option. This profile can easily be changed in the model code (routine *setvbc.F*).

28.2.6 time_mean_sbc_data

This option is used to supply annual mean Hellerman and Rosenstein windstress and annual mean Levitus (1982) sea surface temperature and salinity for use with option *restorst*. Data is prepared for the resolution in *mom* as described in Section 3.2. Refer to Section 19.2 for further discussion. If option *restorst* is enabled, surface temperature and salinity are damped back to this annual mean data as given by Equation (28.23) in Section 28.2.9.

28.2.7 time_varying_sbc_data

This option is used to supply monthly mean Hellerman and Rosenstein windstress and monthly mean Levitus (1982) sea surface temperature and salinity for use with option *restorst*. Data is prepared for the resolution specified in module *grids* as described in Section 3.2. It is interpolated to the time step level in *mom* as described in Section 19.2. If option *restorst* is enabled, surface temperature and salinity are damped back to this monthly mean data as given by Equation (28.23) in Section 28.2.9.

28.2.8 coupled

This option is used to couple *mom* to an atmospheric model assumed to have resolution differing from that of *mom*. It allows for two way coupling as described in Section 19.1.

28.2.9 restorst

Ocean tracers are driven directly by surface tracer fluxes. In some cases, specification of surface tracer flux may lead to surface tracers drifting far away from observed data. This may be due to errors in the data or in paramaterizations not capturing the proper physics. When this happens, it is desirable to restrict how closely sea surface tracers remain to prescribed data by use of a restoring term. The prescribed data typically comes from options *simple_sbc*, *time_mean_sbc_data*, or *time_varying_sbc_data* as described above. Enabeling option *restorst* provides the restoring by means of Newtonian damping.

The damping is actually converted to a sea surface tracer flux and enters the tracer equation as a top boundary condition for vertical diffusion. The amount of damping is defined by a time scale *dampts* in days which is input through namelist (Refer to Section 14.4). Note that the damping time scale may by set differently for each tracer.

A Newtonian damping term applied to the first vertical level may be converted into a surface tracer flux by vertically summing as in

$$\sum_{k=1}^{km}(\kappa t_z)_z \Delta z = -\sum_{k=1}^{km}\delta^{1,k}\cdot\frac{1}{dampts_n\cdot 86400.0}(t-t^\star)\Delta z \tag{28.22}$$

where t is temperature and $\delta^{1,k}$ is the Kronecker delta ($\delta^{1,1}=1$ and $\delta^{1,k>1}=0$). The left hand side of Equation (28.22) equates to $stf - bmf$ where stf is the surface tracer flux and bmf is the bottom tracer flux (which is taken as zero). Applying the indexing terminology of MOM, the equation becomes

$$stf_{i,j,n} = -\frac{dzt_{k=1}}{dampts_n\cdot 86400.0}(t_{i,1,j,n,\tau-1} - t^\star_{i,j,n,time}) \tag{28.23}$$

where $t^{\star}_{i,j,n,time}$ represents prescribed surface data for tracer n and $dampts_n$ is the damping time scale (Note that "stf" entered the tracer equation as a newtonian damping term in MOM 1 so the factor $dzt_{k=1}$ was not there.). The effective damping coefficient is $\rho_{\circ} c_p \cdot dzt_{k=1}/(dampts_n \cdot 86400.0)$ and is printed out when the model executes. The damping coefficient is printed in units of "Watts/m^2/deg C" for $n = 1$ and "Kg/m^2/sec/model salinity unit" for $n = 2$. Recall that a "model salinity unit" is (ppt-35)/1000 where "ppt" is "parts per thousand" or "grams of salt per kilogram of water". The "ppt" unit of salinity has been largely replaced by "practical salinity units" or "psu" in the literature which is based on conductivity measurements instead of measuring "grams of salt per kilogram of water". For $n > 2$, the "model salinity unit" is replaced by "tracer unit". It should be apparent that two models with differing $dzt_{k=1}$ require differing values of $dampts_n$ to insure both models are getting the same tracer flux across the top layer.

28.2.10 shortwave

Heatflux at the ocean surface is composed of latent, sensible, longwave, and shortwave components. When applied as an upper boundary condition at the ocean surface, all of the flux is absorbed within the first vertical model level. If vertical resolution is less than 25m, this may lead to too much heating within the first level resulting in SST that is too warm. Option *shortwave* allows some of the solar shortwave energy to penetrate below the first level. This penetration below the surface is a function of wavelength. For the case of clear water, it is assumed that energy partitions between two exponentials with a vertical dependency given by

$$pen_k = A \cdot e^{-zw_k/\ell_1} + (1 - A) \cdot e^{-zw_k/\ell_2} \quad for\ k = 1\ to\ km \tag{28.24}$$

Coefficients are set for the case of clear water using $A = 0.58$, $\ell_1 = 35$ cm, and $\ell_2 = 2300$ cm. This means that 58% of the energy decays with a 35 cm e-folding scale and 42% of the energy decays with a 23 m e-folding scale. If the thickness of the first ocean level $dzt_{k=1} = 50$ meters, then penetration of solar shortwave wouldn't matter. However, for the case where $dzt_{k=1} = 10$ meters, the effect can be significant and may be particularly noticeable as warm SST in the summer hemisphere.

The divergence of the vertical penetration pen_k is calculated as

$$divpen_k = (pen_{k-1} - pen_k)/dzt_k, \quad for\ k = 1\ to\ km \tag{28.25}$$

and the sub-surface heating due to the divergence is added to the temperature component of the tracer equation through a source term (refer to Section 22.8.5) given by

$$source_{i,k,j} = source_{i,k,j} + sbcocn_{i,jrow,m_{i,jrow,isw}} \cdot divpen_k \tag{28.26}$$

where subscript *isw* points to the shortwave surface boundary condition[1]. In general, since surface heatflux $stf_{i,j,1}$ already contains a solar shortwave component, the penetration function at the ocean surface is set to zero ($pen_0 = 0$) to prevent the shortwave component from being added in twice: once through the surface boundary condition ($stf_{i,j,1}$) for vertical diffusion (given by Equation (22.50)) and once through the source term given above. For further information refer to Paulson and Simpson (1977), Jerlov (1968) and Rosati (1988).

[1]Note that solar shortwave data is not supplied with MOM and so must be supplied by the researcher.

28.2.11 minimize_sbc_memory

This option re-dimensions buffer arrays used with option *time_varying_sbc_data* for time interpolations. Instead of being dimensioned as (imt,jmt) these arrays are dimensioned as (imt,jmw) and there are two buffer fields for each surface boundary condition. The resulting savings in memory is approximately $2 \cdot numobc \cdot imt \cdot jmt$ where *numobc* is the number of ocean surface boundary conditions . However, the price to be paid is increased disk access which will significantly slow execution if conventional rotating disk is used. The intent of this option is for very high resolution models which cannot fit into available memory and require a very conservative approach to memory usage. For additional space saving measures, refer to Section 38.12.

28.3 Lateral Boundary Conditions

Neumann conditions are assumed for heat and salt on lateral boundaries (no-flux across boundaries). For momentum, a Dirichlet condition is assumed (no-slip). There are some minor variations on these at the domain boundaries as described in the following sections. Note that land cells are where $kmt_{i,jrow} = 0$. In the following set of options, a choice must be made between enabeling option *cyclic* or *solid_walls*. The others are optional.

28.3.1 cyclic

This makes a semi-infinite domain. It implements cyclic conditions in longitude. Whatever flows out of the (eastern,western) end of the basin enters the (western,eastern) end. Normally, the computations in longitude proceed from $i = 2$ to $imt - 1$ after which cyclic conditions are imposed. In latitude, the computations proceed from $jrow = 2$ to $jmt - 1$. The cyclic boundary condition is

$$\begin{aligned} kmt_{1,jrow} &= kmt_{imt-1,jrow} \\ kmt_{imt,jrow} &= kmt_{2,jrow} \end{aligned} \tag{28.27}$$

Note, there is no option for the doubly cyclic case (cyclic in latitude also) and on the northern and southernmost cells

$$\begin{aligned} kmt_{i,1} &= 0 \\ kmt_{i,jmt} &= 0 \end{aligned} \tag{28.28}$$

28.3.2 solid_walls

This implements solid walls at domain boundaries in longitude and latitude. The condition is

$$\begin{aligned} kmt_{1,jrow} &= 0 \\ kmt_{imt,jrow} &= 0 \\ kmt_{i,1} &= 0 \\ kmt_{i,jmt} &= 0 \end{aligned} \tag{28.29}$$

The domain is finite and closed.

28.3.3 symmetry

This implements a symmetric boundary condition across the equator. Specifically, the second last row of velocity points must be defined on the equator ($\phi^U_{jmt-1} = 0.0$). Note that the equator is at the northern end of the domain. The condition applies when j in the memory window corresponds to $jrow = jmt - 1$ and is given by

$$\begin{aligned} t_{i,k,j+1,n} &= t_{i,k,j,n} \\ u_{i,k,j+1,1} &= u_{i,k,j,1} \\ u_{i,k,j+1,2} &= -u_{i,k,j,2} \\ psi_{i,jrow+1} &= -psi_{i,jrow} \\ kmt_{i,jrow+1} &= kmt_{i,jrow} \end{aligned} \tag{28.30}$$

28.3.4 sponges

This implements a poor man's open boundary condition along the northernmost and southernmost artificial walls in a limited domain basin. A Newtonian damping term is added to the tracer equations which damps the solution back to data within a specified width from the walls. The form is given by Equation 22.60 and the data is generated by script *run_sponge*[2] in PREP_DATA.

If option *equatorial_thermocline* is enabled, then the profile from Equation (28.1) is used instead of data prepared in PREP_DATA. The meridional width of the sponge layer *spng_width* is hard wired to 3 degrees and the reciprocal of the damping time scale *spng_damp* is hard wired to 1/5 days. Their purpose is to damp Kelvin waves in idealzed equatorial models. As indicated in Section 3.1, use UNIX *grep* to find their location if changes are to be made.

The current implementation uses data defined at the latitude of the northern and southern walls as the data to which the solution is damped. This data varies monthly, but the annual mean values can be used instead by setting variable *annlev* in the namelist. Refer to Section 14.4 for information on namelist variables. The width of the sponge layers is determined by a Newtonain damping time scale that is a function of latitude and set in subroutine *sponge* which is executed by script *run_sponge*.

If a more realistic sponge layer is desired, data from latitude rows within the sponge layers needs to be saved instead of just the data at the latitude of the walls. This is a bit more I/O intensive and is not an option as of this writing. Refer to Sections 3.2 and 22.8.5 for further details.

28.3.5 obc

Open boundary conditions are based on the methodology of Stevens (1990). There are two types of open boundary conditions: 'active' in which the interior is forced by data prescribed at the boundary and 'passive' in which there is no forcing at the boundary and phenomena generated within the domain can propagate outward without disturbing the interior solution.

Open boundaries may be placed along the northern, southern, eastern and/or western edges of the domain. At open boundaries, baroclinic velocities are calculated using linearized

[2]Note that this script can only be run after script *run_ic* which prepares temperature and salinity data for all latitude rows.

horizontal momentum equations and the streamfunction is prescribed from other model results or calculated transports (e.g. directly or indirectly from the Sverdrup relation). Thus, the vertical shear of the current is free to adjust to local density gradients. Heat and salt are advected out of the domain if the normal component of the velocity at the boundary is directed outward. When the normal component of the velocity at the boundary is directed inward, heat and salt are either restored to prescribed data ('active' open boundary conditions) or not ('passive' open boundary conditions).

In contrast to the above described 'active' open boundary conditions, 'passive' ones are characterized by not restoring tracers at inflow points. Additionally, a simple Orlanski radiation condition (Orlanski 1976) is used for the streamfunction.

Refer to Chapter 20 for all the options and details.

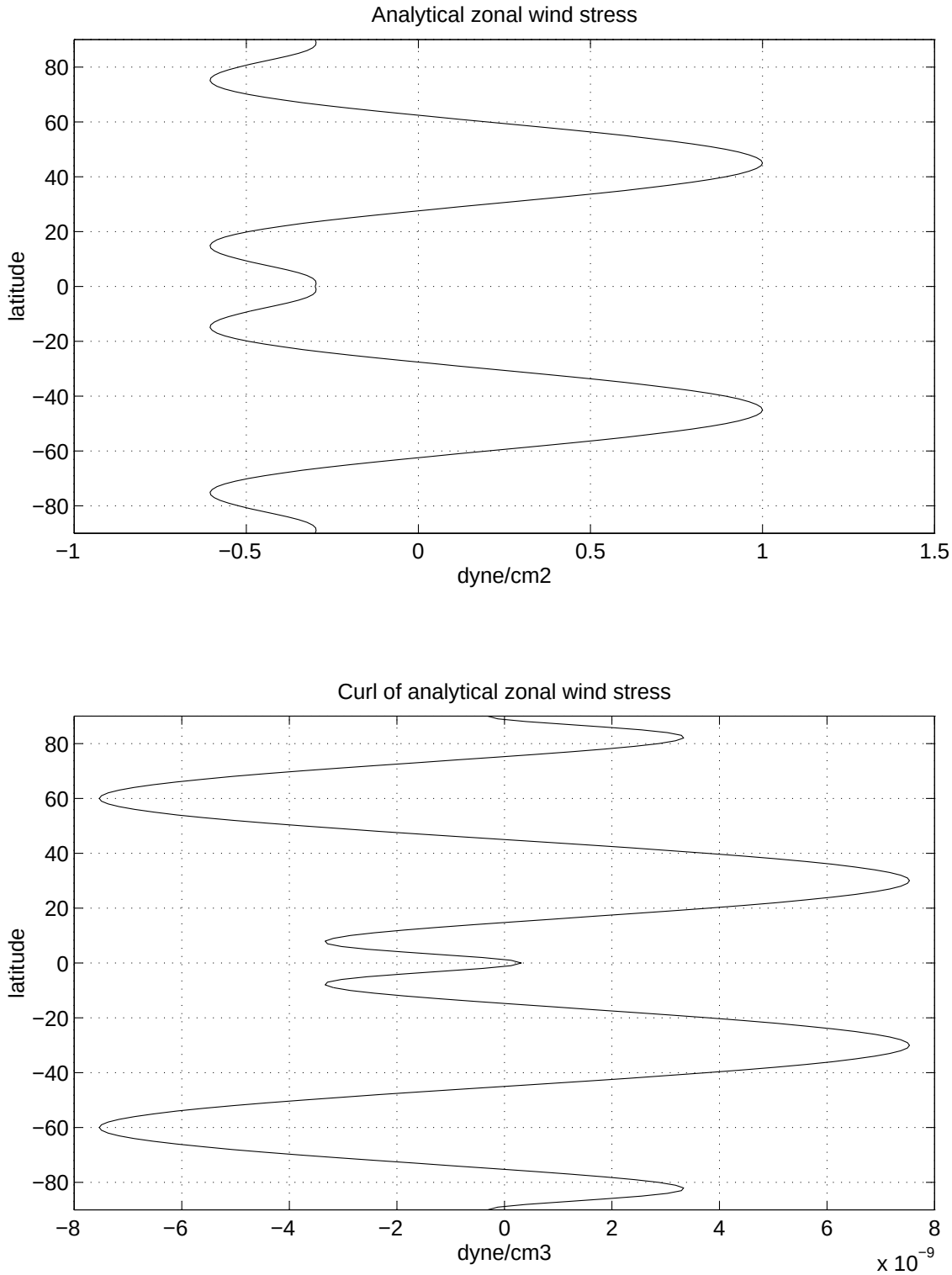


Figure 28.1: Upper panel: Analytic zonal wind stress from F. Bryan (1987). Lower panel: Curl of the wind stress.

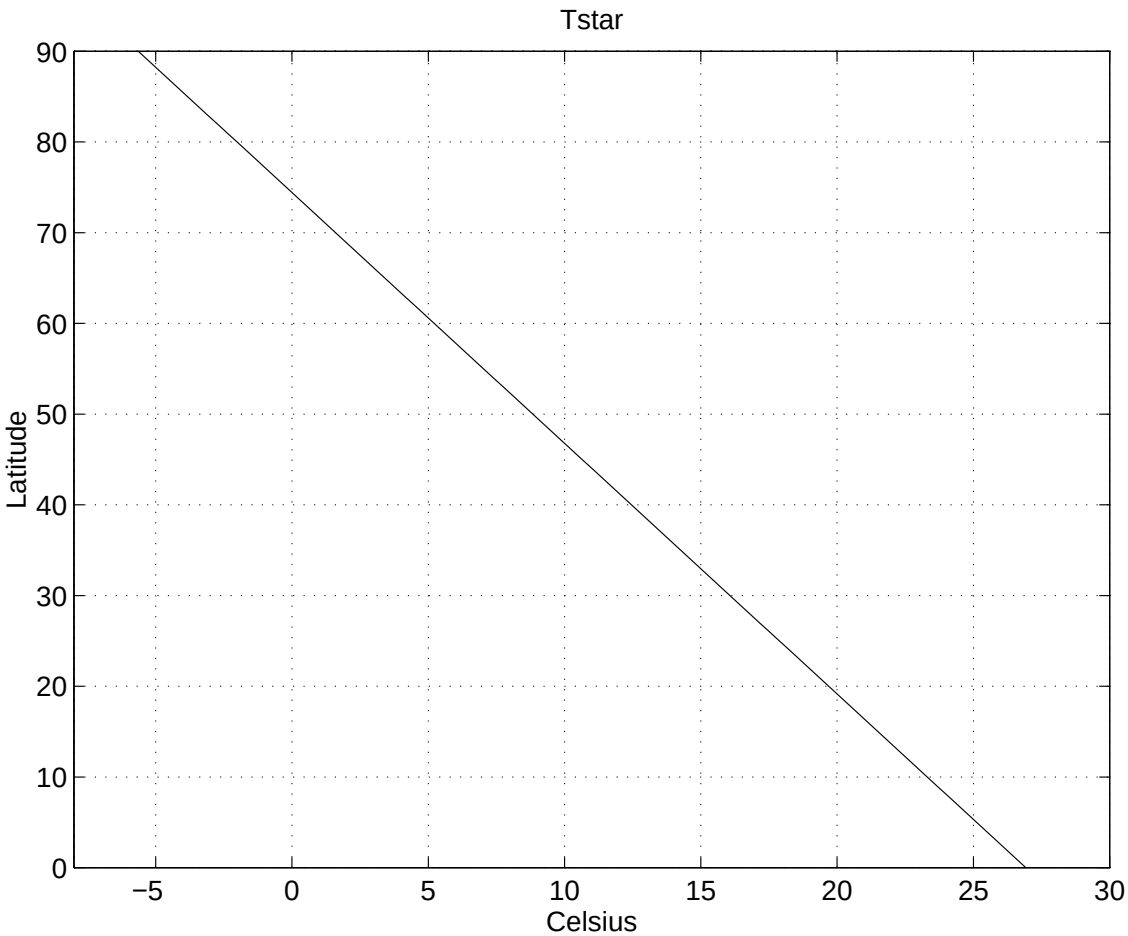


Figure 28.2: Linear profile of temperature from Cox and Bryan (1984).

Chapter 29

Old options for the external mode

There are various ways to solve for the external mode (depth integrated) velocities in MOM. The purpose of this chapter, as well as Chapter 30, is to describe the numerical issues involved. Many of the theoretical issues were discussed in Chapter 4. Starting this chapter are some comments concerning the main points to be considered when choosing a particular method. Additionally, within the discussion of each method, certain caveats, which are mostly suggestive and not rigorous, are mentioned. In general, stability of the external mode in ocean modeling tends to be the very sensitive, especially in global models with realistic geography and topography. A great deal of energy has gone into deriving various methods for solving the external mode. The present methods are not perfect, and development continues.

29.1 Concerning which external mode option to use

The methods available for updating the velocity field are the following:

- Option *stream_function* uses the Bryan (1969) rigid lid streamfunction approach.
- Option *rigid_lid_surface_pressure* uses the rigid lid surface pressure approach of Dukowicz, Smith, and Malone (1993).
- Option *implicit_free_surface* employs the implicit free surface method of Dukowicz and Smith (1994).
- Option *explicit_free_surface* and *explicit_eb* or *explicit_fb* employ the free surface method of Killworth, Stainforth, Webb and Paterson (1991).
- Option *explicit_free_surface* employs an explicit free surface method which fully incorporates the undulating surface height within the baroclinic momentum and tracer equations. It also allows for longer tracer time steps than the Killworth *et al* method.

Researchers at GFDL are using the last option (explicit free surface) most frequently (Summer 1999). This method is the only one which will be available in post MOM 3 releases of MOM.

29.1.1 Wave processes

The explicit free surface was written with the spreading of surface gravity waves and/or Kelvin waves as one of the main physical phenomenon of interest (Killworth *et al.* 1991).

The rigid lid stream function and rigid lid surface pressure approaches completely filter out these waves. The implicit free surface is very diffusive and so may not provide a satisfactory simulation of these waves for certain purposes, such as tidal studies.

In addition, note that even for climate studies, the assumptions of a rigid lid might not be valid for equatorial dynamics. Namely, the rigid lid approximation makes the speed of all external gravity waves infinite and therefore instantly equilibrates them. This is reasonable in mid and high latitudes where there is a large time scale separation between gravity waves and Rossby waves. However, this separation of time scales is not the case in the equatorial domain. Both explicit and implicit free surface methods resolve Rossby and gravity waves within the equatorial region, while equilibrating higher frequency gravity waves at mid and high latitudes. It remains to be shown if this difference between rigid lid and free surfaces is significant.

29.1.2 Surface tracer fluxes

An incresingly large number of modelers are paying close attention to how tracers are forced. For example, ocean biogeochemical modeling requires some dozens of tracers with varying surface and interior sources. The central problem with the rigid lid approximation is the manner in which fresh water enters the ocean model, and consequently how tracers contained in precipitation and river runoff enter as well. Namely, fresh water enters not as a fresh water flux, but as a "virtual salt flux." The reason it must do so in a rigid lid approximation can be seen by looking at the kinematic boundary condition at the ocean surface. For a free surface, the kinematic boundary condition in the presence of fresh water fluxes Q_w is given by

$$\eta_t = w + q_w - \vec{u}_h \cdot \nabla_h \eta \qquad z = \eta, \tag{29.1}$$

where η is the deviation of the surface height from $z = 0$. This equation is derived in Section 7.2.2. For a rigid lid, the surface velocity is set to zero, and the surface elevation is a constant $\eta = 0$. Therefore, $Q_w = 0$ everywhere on the ocean surface, which precludes the introduction of a fresh water flux through a rigid lid. Instead, for the purpose of garnering some sort of dynamical input of fresh water, it is necessary to introduce an unphysical salt flux in the salinity equation. Huang (1993) provides a thorough critique of this approach and proposes an alternative within the context of a generalized rigid lid model for which the vertical velocity is not set to zero, yet there is no explicit equation for the free surface height. Since $w(z = 0)$ is not set to zero, an extra elliptical problem needs to be solved for the stream potential, which is necessary since the vertically integrated velocity is no longer divergence-free when $w(z = 0) \neq 0$. The presence of two elliptic equations in a climate model can potentially be quite costly, and so this approach has not been implemented in MOM.

The use of a free surface with proper accounting of fresh water flux is arguably the most physically and numerically satisfying approach. The option *explicit_free_surface* allows for a conservative model with fresh water input.

29.1.3 Killworth topographic instability

Killworth (1987) identified a fundamental problem with the rigid lid streamfunction when used with steeply sloping bottom topography. The condition places a limit on the time step based on lateral viscosity, resolution, and topographic slope. This condition and methods for relaxing it are given in Section 18.5.

Basically, due to a factor of topographic factor of $1/H$ in the elliptic equation for the rigid lid streamfunction (Section 29.2.1), there is a sensitivity to rapid changes in topography[1]. As pointed out by Dukowicz and Smith (1994), the rigid lid surface pressure and implicit free surface approaches are better conditioned than the streamfunction because the factor $1/H$ in the stream function is replaced by a factor of H (see Section 29.4).

The explicit free surface does not have an elliptic equation to solve, and so is not restricted by the Killworth topographic condition. Qualitatively, since the explicit free surface's barotropic mode can change height, it is not "squashed" like the barotropic mode in the rigid lid cases in the presence of rapidly changing bottom topography. Hence, both free surface options are not subject to the Killworth topographic condition.

29.1.4 Wave speed considerations

Use of rigid lid and implicit free surface methods allow relatively long time steps to be used by eliminating fast external gravity waves. The next fastest waves are large scale external Rossby waves which limit the time step for the external mode. The fastest remaining waves are internal gravity waves due to vertical differences in density and they travel at speeds less than 3 m/sec. Since the inertial period at the pole is 1/2 day, time steps greater than about 2 hours are unstable unless the inertial oscillation is filtered out. Typically, either internal gravity waves or inertial oscillations limit the time step for the internal mode in coarse resolution models. Density is limited by advective velocity which is usually less than 2 m/sec. As an example, in a 4-degree model, time steps as long a few days are typical for the density (i.e., temperature and salinity). For large-scale ocean modeling in which the momentum field is close to geostrophic balance, the density time sets the "time" for the ocean model. Therefore, maintaining a long density time step is central to quickly spinning up global ocean models (see Bryan 1984 and Killworth, Smith, and Gill 1984 for more details on acceleration techniques). However, as mentioned by Killworth *et al.* (1991), it is necessary to greatly reduce the density time step in the explicit free surface due to a bleeding of the barotropic mode into the baroclinic mode. This bleeding effectively makes the density time step constrained by the first baroclinic Rossby wave speed which results in roughly a factor of 10 reduction in the density time step from that available in the rigid lid and implicit free surface. Empirical tests with MOM are consistent with this constraint.

29.1.5 Polar filtering

The rigid lid stream function and explicit free surface are much more tolerant of polar filtering than the implicit free surface. Nonetheless, whenever polar filtering, there is a tendency for the vertical velocity field to become noisy, especially when running with fourier filtering. Note that when applying a polar filter with option *firfil* or *fourfil*, the forcing for the rigid lid stream function is also filtered. Details of polar filtering are discussed further in Chapter 27.

In the explicit free surface, polar filtering must be applied within every barotropic time step. For the rigid lid and the implicit free surface, polar filtering is applied only once before the elliptic equation is solved. Therefore, when using the explicit free surface, it is important

[1]This instability has been associated with problems in various versions of the GFDL coupled model in the Drake Passage region. For example, a traditional time step analysis indicates that the ocean component of the 4-degree/R15 GFDL coupled model should work fine with a 6 hour barotropic time step. However, it is unable to run with time steps much larger than 2 hours. The consensus is that the Killworth instability is the dominant factor determining the time step in this rigid lid model (Keith Dixon and Ron Stouffer, personal communication).

to move the latitude where filterting starts as far poleward as possible since polar filtering is expensive.

Tests with the implicit free surface indicate that polar filtering of the external mode velocities leads to a problem in the filtered latitudes. Removing the filtering has been shown to eliminate the problem. However, the standard TEST CASE 0 will not run without filtering and so the filtering has been left in the code.

Finally, it should be noted that polar filtering may not always be necessary, even for global models. Due to its rather unphysical properties, initial tests should be run for each model configuration to see just how much polar filtering is required and whether its affects are detrimental to the physics of interest. In particular, experience at GFDL indicates that for some model configurations, it is economical to run the explicit free surface without polar filtering the barotropic mode. Although the absence of polar filtering requires taking smaller barotropic time steps, there are nontrivial savings due to the absence of polar filtering on every barotropic time step.

29.1.6 Parallelization

There is currently a lot of attention paid to parallelizing MOM (Chapter 12 and Section 24.4). The presence of island integrals in the rigid lid streamfunction approach (Section 29.2) renders the rigid lid unsuitable for parallelism due to nonlocal processes (i.e. island integrals). Both the explicit (Section 29.5 Killworth *et al.* 1991) and implicit (Section 29.4, Dukowicz and Smith 1994) free surfaces do not require island integrals, hence their use offers strong advantages over the rigid lid stream function. Preliminary experience indicates that the explicit free surface is better scalable to larger numbers of processors. This is especially true if gravity can be reduced by a factor of 10 in the explicit free surface. This is under investigation.

Both the implicit and explicit free surfaces, have been parallelized. Work to increase the efficiency of these schemes is ongoing.

29.2 stream_function

Option *stream_function* enables the time honored standard approach which eliminates surface pressure from the momentum equations by vertically integrating, taking the curl, and assuming that vertical velocity vanishes at the ocean surface which is taken at $z = 0$. This *rigid lid* approach was developed by Bryan (1969) and has remained the choice of numerous ocean modelers for almost 30 years. However, for the reasons mentioned in Section 29.1, the rigid lid is being displaced by the more physically complete and computationally efficient (with regard to parallelism) free surface approaches. This section discusses the computational implementation of the rigid lid. The theoretical formulation is given in Bryan (1969).

Boundary conditions for the rigid lid barotropic streamfunction are Dirichlet, which necessitate solving island equations as given in Section 29.2.4. Artificially setting the flow between land masses to zero may be implemented, as described in Section 29.2.6. Changes[2] to the coefficient matrices and in the handling of islands in the elliptic solvers have resulted in faster convergence rates than in MOM 1. The external mode velocities given by the stream function approach are guaranteed to be divergence free even if the solution for change in stream function $\Delta\psi$ is not accurate. The accuracy of the solution for $\Delta\psi$ is governed by "tolrsf" which is input through namelist. Typically, the value is 10^8 cm^3/sec.

[2]Worked out by Charles Goldberg.

29.2.1 The equation

The stream function equation is generated by taking the curl of the vertically averaged momentum equations to knock out the unknown surface pressure terms. This is done by vertically averaging the Equations (4.1) and (4.2), expressing the averaged velocities in terms of a stream function ψ, and taking the $\hat{\mathbf{k}} \cdot \nabla \times$ of these equations yielding:

$$\nabla \cdot (\frac{1}{H} \cdot \nabla \psi_t) - J(acor \cdot \frac{f}{H}, \psi) = \hat{\mathbf{k}} \cdot \nabla \times F \tag{29.2}$$

where J is the Jacobian[3], *acor* is the implicit Coriolis factor[4], the Coriolis term $f = 2\Omega \sin \phi$, and F is the vertically averaged forcing computed in subroutine *baroclinic*. The boundary condition is that the normal component of the gradient of the stream function $\hat{n} \cdot \nabla \psi = 0$ on lateral boundaries. Actually, since the viscous terms in the vertically averaged forcing are of the form $\nabla^2 u$ and $\nabla^2 v$, another boundary condition is necessary. The additional boundary condition is that the tangential component of the stream function $\hat{t} \cdot \nabla \psi = 0$ on lateral boundaries.

Bryan(1969) gives the discretization of Equation (29.2) in terms of five point numerics. This means that the discretized equation at grid point with subscripts $(i, jrow)$ involves the four nearest neighboring points with subscripts $(i+1, jrow)$,$(i-1, jrow)$, $(i, jrow+1)$, and $(i, jrow-1)$. Semtner (1974) derives the nine point equivalent of Bryan's external mode equation which additionally involves the four nearest neighboring corner points $(i+1, jrow+1)$,$(i-1, jrow+1)$, $(i+1, jrow-1)$, and $(i-1, jrow-1)$. A similar approach[5] is given below.

The finite difference counterpart of Equation (29.2) is arrived at by starting with the vertically averaged finite differenced momentum equations

$$\begin{aligned} \frac{\bar{u}_{i,j,1,\tau+1} - \bar{u}_{i,j,1,\tau-1}}{2\Delta\tau} - \tilde{f}_{jrow} \cdot (\bar{u}_{i,j,2,\tau+1} - \bar{u}_{i,j,2,\tau-1}) &= \frac{-1}{\rho_\circ \cdot \cos \phi^U_{jrow}} \cdot \delta_\lambda(\overline{p^s_{i,jrow}}^{\phi}) \\ &+ zu_{i,jrow,1} \end{aligned} \tag{29.3}$$

$$\begin{aligned} \frac{\bar{u}_{i,j,2,\tau+1} - \bar{u}_{i,j,2,\tau-1}}{2\Delta\tau} + \tilde{f}_{jrow} \cdot (\bar{u}_{i,j,1,\tau+1} - \bar{u}_{i,j,1,\tau-1}) &= \frac{-1}{\rho_\circ} \cdot \delta_\phi(\overline{p^s_{i,jrow}}^{\lambda}) \\ &+ zu_{i,jrow,2} \end{aligned} \tag{29.4}$$

where p^s is the unknown surface pressure and $zu_{i,jrow,n}$ contains the vertically averaged advection, diffusion, hydrostatic pressure gradients, and explicit part of the Coriolis term. The implicit part of the Coriolis term is given by

$$\tilde{f}_{jrow} = \quad acor \cdot 2\Omega \sin \phi^U_{jrow} \tag{29.5}$$

When the finite difference curl is taken, particular attention must be taken to assure that the unknown surface pressure terms are eliminated even when the grid is non-uniform. Here is an outline of the steps needed to do this. Starting with Equations (29.3) and (29.4), make the following substitutions:

[3]The Jacobian is given as $J(A, B) = \frac{1}{a^2 \cos\phi}(\frac{\partial A}{\partial \lambda}\frac{\partial B}{\partial \phi} - \frac{\partial B}{\partial \lambda}\frac{\partial A}{\partial \phi})$.

[4] *acor* = zero implies that the Coriolis term is handled explicitly. Otherwise, $0.5 \leq acor < 1.0$ implies implicit handling of the Coriolis term. This is useful for coarse models with global domains where the time step is limited by the inertial period $1/f$.

[5]This approach was first worked out by Charles Goldberg (personal communication) using algebraic manipulations. The derivation given here is in terms of finite difference operators.

$$\bar{u}_{i,j,1,\tau-1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau-1}}^\lambda) \tag{29.6}$$

$$\bar{u}_{i,j,2,\tau-1} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau-1}}^\phi) \tag{29.7}$$

$$\bar{u}_{i,j,1,\tau+1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau+1}}^\lambda) \tag{29.8}$$

$$\bar{u}_{i,j,2,\tau+1} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau+1}}^\phi) \tag{29.9}$$

$$\Delta\psi_{i,jrow} = \psi_{i,jrow,\tau+1} - \psi_{i,jrow,\tau-1} \tag{29.10}$$

where the horizontal stream function $\psi_{i,jrow}$ is defined on T cells. Next, take

$$dxu_i \cdot \cos\phi^U_{jrow} \times \text{equation(29.3)} \tag{29.11}$$

$$dyu_{jrow} \cdot \times \text{equation(29.4)} \tag{29.12}$$

Then take the finite difference equivalent of the curl operation using:

$$-dyt_{jrow} \cdot \delta_\phi(\overline{\text{equation(29.11)}}^\lambda) + dxt_i \cdot \delta_\lambda(\overline{\text{equation(29.12)}}^\phi) \tag{29.13}$$

The result is the finite difference counterpart of Equation (29.2) which is valid for non-uniform grids

$$\begin{aligned}
&\frac{1}{2\Delta\tau} \cdot \left[dyt_{jrow} \cdot \delta_\phi \left(\overline{\left(\frac{dxu_{i-1} \cdot \cos\phi^U_{jrow-1}}{H_{i-1,jrow-1} \cdot dyu_{jrow-1}}\right) \cdot dyu_{jrow-1} \cdot \delta_\phi(\overline{\Delta\psi_{i-1,jrow-1}}^\lambda)}^\lambda \right) \right. \\
&\left. + dxt_i \cdot \delta_\lambda \left(\overline{\left(\frac{dyu_{jrow-1}}{H_{i-1,jrow-1} \cdot \cos\phi^U_{jrow-1} \cdot dxu_{i-1}}\right) \cdot dxu_{i-1} \cdot \delta_\lambda(\overline{\Delta\psi_{i-1,jrow-1}}^\phi)}^\phi \right) \right] \\
&+ \left[dyt_{jrow} \cdot \delta_\phi \left(\overline{\frac{\tilde{f}_{jrow-1}}{H_{i-1,jrow-1}} \cdot dxu_{i-1} \cdot \delta_\lambda(\overline{\Delta\psi_{i-1,jrow-1}}^\phi)}^\lambda \right) \right. \\
&\left. - dxt_i \cdot \delta_\lambda \left(\overline{\frac{\tilde{f}_{jrow-1}}{H_{i-1,jrow-1}} \cdot dyu_{jrow-1} \cdot \delta_\phi(\overline{\Delta\psi_{i-1,jrow-1}}^\lambda)}^\phi \right) \right] \\
&= \left[-dyt_{jrow} \cdot \delta_\phi(\overline{dxu_{i-1} \cdot \cos\phi^U_{jrow-1} \cdot zu_{i-1,jrow-1,1}}^\lambda) \right. \\
&\left. + dxt_i \cdot \delta_\lambda(\overline{dyu_{jrow-1} \cdot zu_{i-1,jrow-1,2}}^\phi) \right]
\end{aligned} \tag{29.14}$$

For each $\psi_{i,jrow}$, Equation (29.14) involves nine points of ψ and $\tilde{f}_{jrow}$ is given by Equation (29.5). Comparing this to Equation (29.2), $\nabla \cdot (\frac{1}{H} \cdot \nabla\psi_t)$ corresponds to the first bracket, $-J(acor \cdot \frac{f}{H}, \psi)$ corresponds to the second bracket and $\hat{\mathbf{k}} \cdot \nabla \times F$ corresponds to the third bracket. In the model, this third bracket is stored as array $ztd_{i,jrow}$.

$$
\begin{aligned}
ztd_{i,jrow} &= -dyt_{jrow} \cdot \delta_\phi(\overline{dxu_{i-1} \cdot \cos\phi^U_{jrow-1} \cdot zu_{i-1,jrow-1,1}}^{\lambda}) \\
&+ dxt_i \cdot \delta_\lambda(\overline{dyu_{jrow-1} \cdot zu_{i-1,jrow-1,2}}^{\phi})
\end{aligned} \tag{29.15}
$$

If the time step constraint imposed by convergence of meridians needs to be relaxed, $ztd_{i,jrow}$ can be filtered in longitude by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil* or finite impulse response filtering enabled by option *firfil*. Both should be used with caution[6] and only when necessary at high latitudes. *firfil* is much faster than *fourfil*.
The boundary condition is no slip on lateral boundaries. This is expressed as

$$
\delta_\lambda(\overline{\psi_{i,jrow}}^{\phi}) = 0 \tag{29.16}
$$

$$
\delta_\phi(\overline{\psi_{i,jrow}}^{\lambda}) = 0 \tag{29.17}
$$

which implies that $\psi_{i,jrow} = constant$ on all land masses and their associated coastal ocean T cells. In domains containing disconnected land masses (islands), the value of $\psi_{i,jrow}$ on each land mass m is a different constant ψ^m. The solution of Equation (29.14) contains an arbitrary constant which means that the value of ψ can be specified arbitrarily on any one land mass. The remaining values of ψ on other land masses are determined and cannot be specified (actually the way to do this is given below). Refer to Section 29.2.4 for how to solving for ψ^m on islands.

29.2.2 The coefficient matrices

Equation (29.14) involves nine values of $\Delta\psi$ centered at $\Delta\psi_{i,jrow}$ which may be written as

$$
\sum_{i'=-1}^{1} \sum_{j'=-1}^{1} coeff_{i,jrow,i',j'} \Delta\psi_{i+i',jrow+j'} = ztd_{i,jrow} \tag{29.18}
$$

where the 3rd and 4th subscripts on the coefficient matrix refer to coefficients on neighboring cells. For example, $i' = -1$ and $j' = 0$ refers to the coefficient of $\Delta\psi_{i-1,jrow}$ (the value on the western neighboring cell). The coefficient of $\Delta\psi_{i+1,jrow+1}$ on the northeast neighboring cell is given by $i' = 1$ and $j' = 1$. When option *sf_9_point* is enabled, MOM calculates the coefficient matrix $coeff_{i,jrow,i',j'}$ for Equation (29.18) using summation formulas given in Section 31.2. This nine point coefficient matrix differs slightly from the one in MOM 1 and is more accurate. The elliptic solvers also converge in fewer iterations using this coefficient matrix. The $\bar{u}_{i,j,1,\tau+1}$ and $\bar{u}_{i,j,2,\tau+1}$ derived from the solution of Equation (29.14) are exact solutions[7] of the finite difference vertically averaged momentum Equations (29.3) and (29.4).

When option *sf_5_point* is enabled, the nine point coefficient matrix is approximated by a five point coefficient matrix involving five non-zero coefficients $coeff_{i,jrow,i',j'}$ for $i' = 0$ or $j' = 0$ as in Bryan (1969). It is arrived at by averaging terms used to construct the nine point coefficient matrix in a different way resulting in a coefficient matrix that differs from the one used by Bryan (1969). The five point coefficient matrix is not as accurate as the nine point

[6] This filtering induces spurious vertical velocities because each latitude if filtered independently and the strips are of varying length due to topography.

[7] To within roundoff.

matrix although the nine point matrix has a checkerboard null mode. Refer to Appendix E for a discussion on null modes. However, in the stream function, this null mode is largely suppressed because $\psi_{i,jrow}$ is constant along boundaries. Also, spatial derivatives of $\psi_{i,jrow}$ remove the this null mode and so it is of no dynamical consequence. The five point matrix does not have this checkerboard null mode. However, both five and nine point operators have an arbitrary unspecified constant null mode. Therefore, the value of ψ on one land mass can be arbitrarily specified (usually set to zero) and all stream function values referenced to this land mass value. Either the five point or nine point operator must be chosen by enabling options *sf_5_point* or *sf_9_point*.

29.2.3 Solving the equation

After choosing whether five point numerics enabled by option *sf_5_point* or nine point numerics enabled by option *sf_9_point* is to be used with Equation (29.14), the elliptic equation is inverted by the method of conjugate gradients. Note that the conjugate gradient solver may fail to converge for large values of the implicit Coriolis parameter *acor*, which de-symmetrize the equations.

29.2.4 Island equations

When solving Equations (29.3) and (29.3) by the method of stream function, Dirichlet boundary conditions on velocity are used. Specifically, both components of vertically integrated velocity are set to zero on all land U cells. This implies

$$\delta_\lambda(\overline{\psi_{i,jrow}}^{\phi}) = 0 \tag{29.19}$$

$$\delta_\phi(\overline{\psi_{i,jrow}}^{\lambda}) = 0 \tag{29.20}$$

which further implies that $\psi_{i,jrow} = \psi^m$ where ψ^m is a constant on all T cells within land mass m and surrounding ocean coastal perimeter cells. Now pick any land mass m and change to a directional notation letting cell $(i, jrow)$ be referred to as cell ℓ. The central coefficient $coeff_{i,jrow,0,0}$ for cell ℓ from Equation (29.18) is referred to as C°_ℓ, the coefficient $coeff_{i,jrow,1,0}$ at the eastern face of cell ℓ is referred to as C^e_ℓ, the coefficient $coeff_{i,jrow,1,1}$ at the northeastern corner of cell ℓ is referred to as C^{ne}_ℓ, and so forth. Using this notation, Equation (29.18) may be written for the ℓ'th T cell as

$$\begin{aligned} C^n_\ell \cdot \psi^n_\ell + C^e_\ell \cdot \psi^e_\ell + C^s_\ell \cdot \psi^s_\ell + C^w_\ell \cdot \psi^w_\ell + C^\circ_\ell \cdot \psi^\circ_\ell + \\ C^{ne}_\ell \cdot \psi^{ne}_\ell + C^{nw}_\ell \cdot \psi^{nw}_\ell + C^{se}_\ell \cdot \psi^{se}_\ell + C^{sw}_\ell \cdot \psi^{sw}_\ell = ztd^\circ_\ell \end{aligned} \tag{29.21}$$

where superscript n indicates the cell to the north of cell ℓ with index $(i, jrow + 1)$, superscript ne indicates the cell to the northeast of cell ℓ with index $(i + 1, jrow + 1)$ and so forth. Superscript $\circ$ refers to the the ℓ'th cell with index $(i, jrow)$.
The central coefficient C°_ℓ is related to the surrounding coefficients by

$$c^\circ_\ell = -(C^n_\ell + C^s_\ell + C^e_\ell + C^w_\ell + C^{ne}_\ell + C^{nw}_\ell + C^{se}_\ell + C^{sw}_\ell) \tag{29.22}$$

An equation for land mass m is generated by summing Equation (29.21) over all cells within the land mass including coastal ocean perimeter cells. At each cell ℓ within the island proper, the left hand side of Equation (29.21) is zero because of Equation (29.22) and the condition that

$$\psi_\ell^\circ = \psi_\ell^n = \psi_\ell^s = \psi_\ell^e = \psi_\ell^w = \psi_\ell^{ne} = \psi_\ell^{nw} = \psi_\ell^{se} = \psi_\ell^{sw} = constant \tag{29.23}$$

After summing over all cells within land mass m and it's ocean perimeter, the only locations with non-zero contributions to the left hand side of Equation (29.21) are ocean perimeter cells. The result in general can be expressed as the sum of nine products summed from $\ell = 1$ to L which is the number of perimeter cells for land mass m.

$$\begin{aligned}\sum_{\ell=1}^{L}(C_\ell^n \cdot \psi_\ell^n + C_\ell^e \cdot \psi_\ell^e + C_\ell^s \cdot \psi_\ell^s + C_\ell^w \cdot \psi_\ell^w + C_\ell^\circ \cdot \psi^\circ + \\ C_\ell^{ne} \cdot \psi_\ell^{ne} + C_\ell^{nw} \cdot \psi_\ell^{nw} + C_\ell^{se} \cdot \psi_\ell^{se} + C_\ell^{sw} \cdot \psi_\ell^{sw}) = \sum_{\ell=1}^{L} ztd_\ell^\circ\end{aligned} \tag{29.24}$$

Without loss of generality, all coefficients adjacent to land cells on the left hand side may be set to zero which leaves only ψ° and values of ψ exterior to the island perimeter. Indeed, this is the very reason why reciprocals of H are set to zero on land (to zero out the coefficients there) in the code.

Recall from Equation (29.15) that $ztd_{i,jrow}$ is in fact the finite difference version of the curl of $zu_{i,jrow,n}$. By Stokes theorem, summing the curl over any area leaves only values of $zu_{i,jrow,n}$ at the outer boundary of the perimeter cells. And at these locations, values of $zu_{i,jrow,n}$ are well defined by Equation (22.118).
Solving Equation (29.24) for $\psi_\circ$ yields

$$\psi_\circ = (\sum_{\ell=1}^{L} ztd_\ell^\circ - \sum_{\ell=1}^{L}(C_\ell^n \cdot \psi_\ell^n + C_\ell^e \cdot \psi_\ell^e + \cdots))/\sum_{\ell=1}^{L} C_\ell^\circ \tag{29.25}$$

where $\psi^m = \psi_\circ$ is the value of the stream function on island m. In the code, $zu_{i,jrow,n}$ is set to zero on land cells so that $\sum_{\ell=1}^{L} ztd_\ell^\circ$ picks up contributions only from values of $zu_{i,jrow,n}$ in the ocean.

The solution of Equation (29.2) is determined only to within an arbitrary constant. If the domain is multiply connected by two or more distinct land masses (islands), the value of the stream function can be chosen arbitrarily on one of the land masses. In MOM 1, the value on the main continent is held fixed at zero and each iteration involved calculating an integral around each other island. In MOM, this option is retained, but it has been determined that the solution converges more quickly if the stream function values on all land masses are allowed to "float". Afterwards, the entire solution is adjusted to make the stream function zero on the main continent. Choosing a stream function value of zero on the main continent <u>and</u> on an island amounts to an over specification of the problem and is not correct.

29.2.4.1 Another approach

Another approach suggested by Charles Goldberg leads to the same result. Equation (31.12) indicates that the right hand side of Equation (29.14) centered at a land or ocean perimeter cell $T_{i,j}$ contains values of $zu_{i,jrow,n}$ on land that are not available. In fact, $forc_{i,jrow}$ is the line

integral of $zu_{i,jrow,n}$ around the boundary of cell $T_{i,j}$. The island equation for land mass m is formed by summing Equation (29.14) over all land and ocean perimeter T cells of land mass m.

Since each right hand side is a line integral around the boundary of one T cell, in the sum of such line integrals over all of land mass m and its surrounding ocean perimeter cells, all contributions from interior edges cancel, leaving only known values of $zu_{i,jrow,n}$ at ocean U cells.

On the left side of an island equation, Equations (31.9) and (31.10) and the fact that $\psi_{i,jrow} = \psi_m$ on all land and ocean perimeter T cells of land mass m imply that all contributions from land T cells are zero as follows. At a land T cell, all nine values of ψ are the constant value ψ_m, so by pulling all terms that don't involve i' or j' out of the i' and j' summations, Equation (31.9) reduces to

$$\begin{aligned}
& \sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^{U}_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddyu_{i',j'} \\
+ & \sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^{U}_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddxu_{i',j'} \\
= & \;=0
\end{aligned} \tag{29.26}$$

since the sum of the partial derivative coefficients $cddxu$ and $cddyu$ are zero. Similarly, at land points, the contributions to the island equation from the implicit Coriolis terms is also zero because Equation (31.10) reduces to

$$\begin{aligned}
& \sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[-cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddxu_{i',j'} \\
+ & \sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[-cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddyu_{i',j'} \\
= & \;0
\end{aligned} \tag{29.27}$$

Because of these simplifications, the island equation for land mass m may be calculated by summing Equations (29.14) only over the ocean perimeter T cells of land mass m.

In fact, in the Fortran code of MOM , the full island equation never appears. Instead, each set of contributions to the island equation from an island perimeter cell $T_{i,j}$ is stored in $coeff_{i,jrow,i'',j''}$. The sum over the island perimeter is done in the elliptic solver.

29.2.5 Symmetry in the stream function equation

This section on symmetry in the stream function equation was contributed by Charles Goldberg. Conjugate gradient solvers work by transforming the system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{29.28}$$

into minimizing the quadratic form

$$\mathbf{Q}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{29.29}$$

This transformation is justified as long as $\mathbf{A}$ is symmetric, that is, $\mathbf{A}_{\alpha,\beta} = \mathbf{A}_{\beta,\alpha}$ for all α and β.

29.2.5.1 Symmetry of the explicit equations

In Equation (29.14), the three major brackets are expanded into summation notation as given by Equations (31.9), (31.10), and (31.12). Interpreting these equations in the formalism given above, the "vector" $\mathbf{x}$ is $\Delta\Psi_{i,jrow}$ at all mid-ocean points and island values $\Delta\Psi_m$, the linear operator $\mathbf{A}$ is the array $coeff_{i,jrow,i'+i'',j'+j''}$, and the subscripts are $\alpha = (i, jrow)$ and $\beta = (i^\star, j^\star)$, where $i^\star = i + i' + i''$ and $j^\star = jrow + j' + j''$ for some values of $i', j' \in \{0, 1\}$ and $i'', j'' \in \{-1, 0\}$. More simply, $\beta = (i^\star, j^\star)$ is one of the eight nearest neighbors of $\alpha = (i, jrow)$ and $\alpha = (i, jrow)$ is one of the eight nearest neighbors of $\beta = (i^\star, j^\star)$. This section shows that the contributions to the coefficients arising from the first brackets, Equation (31.9), are symmetric.

If $i^\star \neq i$, then i' and i'' must both be at their upper limits or both must be at their lower limits. In either case, $i' = i'' + 1$. Similarly, if $j^\star \neq j$, then $j' = j'' + 1$. If both are unequal, that is, if $(i, jrow)$ and $(i^\star, j^\star)$ are diagonal neighbors, then $i + i'' = i + i' + i'' - i' = i^\star - i' = i^\star + (-1 - i'')$, and similarly $jrow + j'' = j^\star + (-1 - j'')$. Note that the expressions $(i^\star)' = 1 - i'$, $(j^\star)' = 1 - j'$, $(i^\star)'' = -1 - i''$, and $(j^\star)'' = -1 - j''$ describe the transition in the opposite direction, so the relations $i + i'' = i^\star + (-1 - i'')$ and $jrow + j'' = j^\star + (-1 - j'')$ show that the evaluation point of most of the factors in Equation (31.9) is the same going both ways. The remaining factors, $cddyu_{i',j'}$, $cddyt_{i'',j''}$, $cddxu_{i',j'}$, and $cddxt_{i'',j''}$ all change sign when i' is replaced by $1 - i'$, j' is replaced by $1 - j'$, i'' is replaced by $-1 - i''$, and j'' is replaced by $-1 - j''$. Thus the product of any two of these is unchanged, and each diagonal coeffiecient at $(i, jrow)$ is equal to the opposite diagonal coefficient at $(i^\star, j^\star)$.

If $i = i^\star$, there are two possibilities: either $i' = i'' = 0$ or $i' = -i'' = 1$. Assume that $j \neq j^\star$. Otherwise, $(i, jrow)$ and $(i^\star, j^\star)$ are not neighbors, but identical. The transitions back to $(i, jrow)$ are in this case: $i = i^\star = i^\star + i' + i''$ and $jrow = j^\star + (1 - j') + (-1 - j'')$, so $i + i'' = i^\star + i''$ and, as above, $jrow + j'' = j^\star + (-1 - j'')$. This time, only j' changes to $1 - j'$ and j'' changes to $-1 - j''$. As a result, the factors $cddyu_{i',j'}$ and $cddyt_{i'',j''}$ change sign, but the factors $cddxu_{i',j'}$, and $cddxt_{i'',j''}$ remain unchanged. All other factors remain evaluated at the same points, so again symmetry holds in that the northern or southern coefficient in Equation (31.9) centered at $(i, jrow)$ is the same as the opposite coefficient in Equation (31.9) centered at $(i^\star, j^\star)$. The case where $j = j^\star$ is proved similarly.

29.2.5.2 Anti-symmetry of the implicit Coriolis terms

If one applies the same arguments to the implicit Coriolis terms in Equation (31.10), the result is a proof of antisymmetry rather than symmetry. First, the diagonal coefficients in the implicit Coriolis terms are zero, so this case need not be considered. In the northern and southern terms (i.e., when $i = i^\star$), the evaluation points given by $(i + i'', jrow + j'') = (i^\star + i''), j^\star + (-1 - j''))$ still remain the same, but each term has one $cddx$ coefficient and one $cddy$ coefficient, so the product of these two factors changes sign. The antisymmetry of the eastern and western implicit Coriolis coefficients is proved similarly.

Since conjugate gradient solvers are derived under the assumption of symmetry of coefficients, the larger the implicit Coriolis terms, the less suitable a conjugate gradient solver is for the elliptic Equations (29.14). For large values of the implicit Coriolis parameter *acor*, the conjugate gradient solver will not converge.

29.2.5.3 Island equations and symmetry

If one of the T cells, $\alpha = (i, jrow)$ or $\beta = (i^\star, j^\star)$ is an island perimeter cell, the arguments in the above section on symmetry of the explicit equations must be made more carefully, since each

island equation is a sum of Equations (29.14). Note that it cannot happen that both α and β are island perimeter T cells because islands must be separated by at least two ocean T cells.

Without loss of generality, we may assume that β is an island perimeter cell and the α is not. In this case, β may not be the only island perimeter cell of land mass m that is a nearest neighbor of α. The coefficient of the island perimeter value $\Delta\Psi_m$ in the elliptic Equation (29.14) centered at α is the sum of the contributions from all nearest neighbors of α that are in the island perimeter of land mass m. Each contribution will be shown equal to a corresponding contribution of $\Delta\Phi_\alpha$ to the island equation for land mass m. Individually, each coefficient[8] of the equation centered at cell α, say, the coefficient in the direction of cell β, is equal to the opposite coefficient in the contribution to the island equation arising from β. Since multiple island perimeter cells as nearest neighbors of α lead to a sum of their individual contributions, and since the island equation is the sum of the individual equations centered at island perimeter points β, both summations lead to the same result[9]. Thus even the coefficient values involving islands satisfy the symmetry relation $\mathbf{A}_{\alpha,\beta} = \mathbf{A}_{\beta,\alpha}$ if the implicit Coriolis parameter is zero.

29.2.5.4 Asymmetry of the barotropic equations in MOM 1

The barotropic equations presented to the solvers in MOM 1 were not symmetric, even when the implicit Coriolis parameter *acor* was set to zero. In an attempt to optimize the code to save a few floating point operations in the calculation of $\mathbf{Ax}$, each equation was divided by its diagonal coefficient, $coeff_{i,jrow,0,0}$. Since these diagonal coefficients depend on topography and grid factors, they are not equal at neighboring cells, and the resulting equations are not symmetric. The asymmetry between a mid-ocean cell and a neighboring island perimeter is likely to be especially severe. It is possible that some of the problems arising with the use of conjugate gradient solvers in MOM 1 may be attributed to MOM 1's "normalization" of the elliptic equations and the resulting de-symmetrization of the coefficient array.

29.2.6 zero_island_flow

Specifying the value of the stream function ψ on two or more distinct land masses amounts to an over-specification of conditions for solving the stream function equation. This over-specification is strictly not correct. Nevertheless, it is sometimes useful to be able to specify a zero net transport between two land masses. Option *zero_island_flow* effectively combines the coastal perimeter cells from two arbitrary land masses into one perimeter which implies that the stream function will have the same value on both land masses; even though the land masses are not physically connected. Therfore, there is no net transport between the land masses.

Option *zero_island_flow* requires "land_mass_a" and "land_mass_b" to be specified through namelist. Values for "land_mass_a" and "land_mass_b" can be taken from the island map which is printed out when MOM 2 executes. Refer to Section 14.4 for information on namelist variables. It must always be kept in mind that option *zero_island_flow* is an over-specification which may have side effects. To judge whether these side effects are significant or not, a companion experiment should be run without specifying the flow between land masses. Although this option is only written to specify zero net transport between two land masses, it can be easily extended to handle more land masses.

[8] Only the first brackets are being done here. The implicit Coriolis terms stand no chance of being symmetric.

[9] Note that remote island perimeter cells neither appear in the equation centered at α, nor do they contain references to cell α in their (up to) nine terms.

Note: Specifying a non-zero net transport between arbitrary land masses has been tried by re-setting ψ after it has been predicted by the conjugate gradient solver. This method seems to work for land masses with relatively short perimeters but fails for land masses with long perimeters. The right way to specify flow is to modify the conjugate gradient solver by specifying the net flow between land masses within the "scan" loop. In principle, a modification of this kind should work but, so far, attempts have failed.

29.3 rigid_lid_surface_pressure

Option *rigid_lid_surface_pressure* enables the method developed by Smith, Dukowicz, and Malone (1992) which is hereafter referred to as SDM. Instead of taking the curl of the vertically integrated momentum equations to drop the surface pressure, the divergence is taken which yields an elliptic equation for the surface pressure instead of a stream function. Its main advantage is that Neumann boundary conditions apply instead of Dirichlet boundary conditions, and there is no need to solve island integrals which perform poorly on SIMD[10] parallel computers. However, elliptic equation solvers converge very slowly using this method.

29.3.1 The equations

Basically, the SDM approach is to define the external mode velocities $U^{\tau+1}$ and $V^{\tau+1}$ in terms of auxiliary velocities $\hat{U}$ and $\hat{V}$ and a time difference of surface pressure Δps as follows

$$U^{\tau+1}_{i,jrow} = \hat{U}_{i,jrow} - \frac{2\Delta\tau}{\cos\phi^{U}_{jrow}}\delta_\lambda(\overline{\Delta ps_{i-1,jrow-1}}^{\phi}) \tag{29.30}$$

$$V^{\tau+1}_{i,jrow} = \hat{V}_{i,jrow} - 2\Delta\tau\cdot\delta_\phi(\overline{\Delta ps_{i-1,jrow-1}}^{\lambda}) \tag{29.31}$$

$$\Delta ps_{i,jrow} = ps^{\tau}_{i,jrow} - ps^{\tau-1}_{i,jrow} \tag{29.32}$$

The momentum equations can then be re-written in terms of $\hat{U}$ and $\hat{V}$ as

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} - 2\Delta\tau\omega\delta_\phi(\overline{\Delta ps_{i-1,jrow-1}}^{\lambda}) \tag{29.33}$$

$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} + 2\Delta\tau\omega\delta_\lambda(\overline{\Delta ps_{i-1,jrow-1}}^{\phi}) \tag{29.34}$$

$$\tilde{U}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,1} - \frac{1}{\cos\phi^{U}_{jrow}}\delta_\lambda(\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\phi})) \tag{29.35}$$

$$\tilde{V}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,2} - \delta_\phi(\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\lambda})) \tag{29.36}$$

$$zu_{i,jrow,1} = f(gcor\cdot V^{\tau}_{i,k,j} + (1-gcor)\cdot V^{\tau-1}_{i,k,j}) + G_{i,jrow,1} \tag{29.37}$$

$$zu_{i,jrow,2} = -f(gcor\cdot U^{\tau}_{i,k,j} + (1-gcor)\cdot U^{\tau-1}_{i,k,j}) + G_{i,jrow,2} \tag{29.38}$$

where $\omega = 2\Delta\tau\cdot acor\cdot f$ and the forcing terms $G_{i,jrow,1}$ and $G_{i,jrow,2}$ contain all remaining terms which are known. If solving the Coriolis term explicitly ($acor = 0$), then $gcor = 1$ otherwise $gcor = 0$ for implicit treatment ($acor > 1/2$).

[10] Single Instruction stream–Multiple Data streams

Equations (29.33) and (29.34) can be solved if terms involving Δps are dropped. The justification given by SDM is that these terms are the same order of magnitude as the time truncation error which is $O(\tau^3)$. This is the operator splitting technique of SDM which leads to a self-adjoint elliptic equation and therefore a symmetric coefficient matrix which can be solved with efficient conjugate gradient techniques. Multiplying Equations (29.30) and (29.31) by the total depth H and taking the divergence gives the second order elliptic equation for Δps in terms of known quantities $\hat{U}$ and $\hat{V}$.

$$\begin{aligned} & \delta_\lambda(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}}\delta_\lambda(\overline{\Delta ps_{i-2,jrow-2}})^\phi}^\phi) + \delta_\phi(\overline{H_{i-1,jrow-1}\cos\phi^U_{jrow-1}\delta_\phi(\overline{\Delta ps_{i-2,jrow-2}}^\lambda)}^\lambda) \\ = \; & \frac{1}{2\Delta\tau}(\delta_\lambda(\overline{H_{i-1,jrow-1}\hat{U}_{i-1,jrow-1}}^\phi) + \delta_\phi(\cos\phi^U_{jrow-1}\overline{H_{i-1,jrow-1}\hat{V}_{i-1,jrow-1}}^\lambda)) \end{aligned} \tag{29.39}$$

Equation (29.39) is solved using a conjugate gradient technique with option *sf_9_point*. Note that the number of islands is set to zero ($nislsp = 0$) because no island equations are to be solved. After solving for Δps, a checkerboard null space and mean are removed after which the predicted surface pressure is

$$ps^{\tau+1}_{i,jrow} = ps^{\tau-1}_{i,jrow} + \Delta ps_{i,jrow} \tag{29.40}$$

The above equations are written with a leapfrog time step in mind. During mixing time steps (forward or first pass of an Euler backward), quantities at time level $\tau - 1$ are replaced by their values at τ and $2\Delta\tau$ is replaced by $\Delta\tau$. The second pass of an Euler backward mixing time step is as described in Section 21.4 and indicated in Figure 21.1.

29.3.2 Remarks

29.3.2.1 Boundary conditions

It should be noted that Equation (29.39) only requires a Neumann boundary condition at the boundaries instead of the Direchlet boundary condition required for the elliptic equation of the stream function method. The implication is that there are no island equations to be solved and hence this method should be faster than the stream function method on massively parallel computers.

29.3.2.2 Conditioning of the elliptic operator

The second point to be made is that Equation (29.39) contains a factor $H_{i,jrow}$ whereas the elliptic equation for the stream function contains the factor $1/H_{i,jrow}$ which should make this method less prone to stability problems than the stream function equation when topography contains steep slopes.

29.3.2.3 Non-divergent barotropic velocities

The third point to be noted is that the barotropic velocities $U^{\tau+1}_{i,jrow}$ and $V^{\tau+1}_{i,jrow}$ given by this method are not non-divergent. The degree of non-divergence is related to how accurately Equation (29.39) is solved for the change in surface pressure Δps and the accuracy depends on the tolerance variable *tolrsp* which is input through namelist and typically set to $10^{-4} gram/cm/sec^2$. Refer to Section 14.4 for information on namelist variables.

29.3.2.4 Polar filtering

Use of polar filtering on $\hat{U}$ and $\hat{V}$ leads to a problem. Removing the filtering eliminates the problem. So the filtering has been removed for this method.

29.3.2.5 Checkerboarding in surface pressure

The surface pressure field in the rigid lid - surface pressure method will sometimes exhibit a checkerboard pattern due to the use of the 9-point Laplacian operator. Namely, this operator contains a zero frequency eigenmode ("null mode") which has a checkerboard pattern. This null mode can be excited in general, and so must be subtracted out. Further discussion is given in Smith, Dukowicz, and Malone (1992).

29.4 implicit_free_surface

Option *implicit_free_surface* enables the method developed by Dukowicz and Smith (1994) which is hereafter referred to as DS. This work is an extension of their earlier work described in Section 29.3. The rigid lid assumption is replaced by a free surface which exerts a surface pressure at $z = 0$. As in the rigid lid surface pressure approach, an elliptic equation can be derived for the change in surface pressure Δps but the resulting equation is more diagonally dominant than the rigid lid surface pressure equation. The implication is that the DS method converges faster than the rigid lid surface pressure method. As with the rigid lid surface pressure approach, DS is well suited for SIMD parallel computers because it requires Neumann boundary conditions and thus needs no island integrals.

Chapter 7 discusses the general mathematical issues concerning the formulation of the free surface. This section discusses the numerical details of how to implement the implicit solution for the surface pressure, and thus the surface elevation. Note that the implicit free surface implemented in MOM has been frozen in its MOM 2 version. The updates to include fresh water forcing have only been done for the explicit free surface discussed in Section 29.5.

29.4.1 The equations

The equations are given as (E1), (E2) and (E3) in DS and will be repeated here for convienence. They are

$$
\begin{aligned}
(\mathbf{I} + \alpha'\tau\mathbf{B})(\hat{\mathbf{u}} - \mathbf{u}^{n-1}) &= \tau(\mathbf{F} - g \cdot \mathbf{G}(\tilde{\gamma}\eta^n + (1-\tilde{\gamma})\eta^{n-1}) \\
&\quad -\mathbf{B}(\tilde{\gamma}'\mathbf{u}^n + (1-\tilde{\gamma}')\mathbf{u}^{n-1}) && (29.41)\\
(\mathbf{DHG} - \frac{2}{\alpha\theta g\tau^2}\mathbf{I})\Delta\eta &= \frac{1}{\alpha\theta g\tau}\mathbf{DH}(\theta\hat{\mathbf{u}} + (1-\theta)\mathbf{u}^{n-1} + \mathbf{u}^n) && (29.42)\\
\mathbf{u}^{n+1} &= \hat{\mathbf{u}} - \alpha\tau g\mathbf{G}\Delta\eta && (29.43)
\end{aligned}
$$

where $\mathbf{B}$ is the Coriolis operator, $\mathbf{D}$ is the divergence operator, and $\mathbf{G}$ is the gradient operator. The centering coefficients $(\alpha, \alpha', \gamma, \gamma', \theta)$ and the quantities $\mathbf{I}, \mathbf{u}, \Delta\eta, g$ and τ are as defined in DS. The above set of equations is very similar to the set in SDM (1992) except there is an extra divergence term and three centering coefficients needed to damp two computational modes. Because of the similarities, the surface pressure and implicit free surface methods share much of the same code in MOM.

In the terminology of MOM, the centering coefficients ($alph, gam, theta$) are set to $(1, 0, 1)$ for the rigid lid surface pressure method and $(1/3, 1/3, 1/2)$ for the implicit free surface method. The relevant equations corresponding to Equations (29.41), (29.42), and (29.43) are as follows

$$
\begin{aligned}
U^{\tau+1}_{i,jrow} &= \hat{U}_{i,jrow} - 2\Delta\tau \cdot \frac{apgr}{\cos\phi^U_{jrow}} \delta_\lambda(\overline{\Delta ps_{i,jrow}}^{\phi}) - U^{\tau}_{i,jrow} && (29.44)\\
V^{\tau+1}_{i,jrow} &= \hat{V}_{i,jrow} - 2\Delta\tau \cdot apgr \cdot \delta_\phi(\overline{\Delta ps_{i,jrow}}^{\lambda}) - V^{\tau}_{i,jrow} && (29.45)\\
\Delta ps_{i,jrow} &= ps^{\tau}_{i,jrow} - ps^{\tau-1}_{i,jrow} && (29.46)\\
\hat{U}_{i,jrow} &= \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} + U^{\tau}_{i,jrow} && (29.47)\\
\hat{V}_{i,jrow} &= \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} + V^{\tau}_{i,jrow} && (29.48)\\
\tilde{U}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,1}\\
&- \frac{1}{\cos\phi^U_{jrow}} \delta_\lambda((1-gam) \cdot \overline{ps^{\tau-1}_{i-1,jrow-1}}^{\phi} + gam \cdot \overline{ps^{\tau}_{i-1,jrow-1}}^{\phi})) && (29.49)\\
\tilde{V}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,2}\\
&- \delta_\phi((1-gam) \cdot \overline{ps^{\tau-1}_{i-1,jrow-1}}^{\lambda} + gam \cdot \overline{ps^{\tau}_{i-1,jrow-1}}^{\lambda})) && (29.50)\\
zu_{i,jrow,1} &= f(gcor \cdot V^{\tau}_{i,k,j} + (1-gcor) \cdot V^{\tau-1}_{i,k,j}) + G_{i,jrow,1} && (29.51)\\
zu_{i,jrow,2} &= -f(gcor \cdot U^{\tau}_{i,k,j} + (1-gcor) \cdot U^{\tau-1}_{i,k,j}) + G_{i,jrow,2} && (29.52)
\end{aligned}
$$

where U and V are vertically averaged velocity components and terms involving $\Delta ps_{i-1,jrow-1}$ have been dropped from Equations (29.47) and (29.48) to obtain the operator splitting as discussed in DS. Also, $\omega = 2\Delta\tau \cdot acor \cdot f$ and the forcing terms $G_{i,jrow,1}$ and $G_{i,jrow,2}$ contain all remaining terms which are known. The right hand side of Equation 29.42 is constructed by taking the divergence of **H** times Equations 29.47 and 29.48. The resulting elliptic equation takes the form

$$
\begin{aligned}
&\delta_\lambda(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}} \delta_\lambda(\overline{\Delta ps_{i-2,jrow-2}})^{\phi}}^{\phi}) + \delta_\phi(\overline{H_{i-1,jrow-1}\cos\phi^U_{jrow-1}\delta_\phi(\overline{\Delta ps_{i-2,jrow-2}}^{\lambda})}^{\lambda})\\
- &\frac{\cos\phi^T_{jrow} dyt_{jrow}}{apgr \cdot 2\Delta\tau^2 \cdot grav} \Delta ps_{i,jrow}\\
= &\frac{1}{apgr \cdot 2\Delta\tau}(\delta_\lambda(\overline{H_{i-1,jrow-1}\hat{U}_{i-1,jrow-1}}^{\phi}) + \delta_\phi(\cos\phi^U_{jrow-1}\overline{H_{i-1,jrow-1}\hat{V}_{i-1,jrow-1}}^{\lambda}))
\end{aligned}
\tag{29.53}
$$

Note that the piece $-\frac{\cos\phi^T_{jrow} dyt_{jrow}}{apgr \cdot 2\Delta\tau^2 \cdot grav} \Delta ps_{i,jrow}$ is included only when the implicit free surface method is used. Also, the coefficient $apgr$ is set to $alph$ for leapfrog time steps and $theta$ for mixing time steps. Equation (29.53) is solved a conjugate gradient technique with option *sf_9_point* with the number of islands $nislsp$ set to zero because no island equations are being solved. After solving for Δps, the predicted surface pressure is given by

$$
ps^{\tau+1}_{i,jrow} = ps^{\tau-1}_{i,jrow} + \Delta ps_{i,jrow} \tag{29.54}
$$

and the free surface elevation at time level $\tau + 1$ is

$$\eta_{i,jrow} = \rho_\circ \cdot grav \cdot ps^{\tau+1}_{i,jrow} \tag{29.55}$$

but η is not explicitly needed in the code and so is not calculated. The vertical velocity at the top of the free surface is

$$adv_vbt_{i,k=0,j} = (ps^{tau+1}_{i,jrow} - ps^{tau-1}_{i,jrow})/(g\Delta\tau) \tag{29.56}$$

29.4.1.1 Modifications for various kinds of time steps

The following setting are needed on various types of time steps:

Leapfrog time steps

The equations are as given above and the centering coefficients are given as

- implicit free surface method: Centering coefficient $apgr = alph$. If solving the Coriolis term explicitly ($acor = 0$), then centering coefficient $gcor = 1$. If solving the Coriolis term implicitly ($acor \neq 0$), then $acor$ is reset to $acor = alph$ and centering coefficient $gcor = gam$.
- rigid lid surface pressure method: Centering coefficient $apgr = alph$. If solving the Coriolis term explicitly ($acor = 0$), then centering coefficient $gcor = 1$ otherwise $gcor = 0$.

Mixing time steps (Forward and Euler backward)

The equations for forward mixing time steps and the first step of an Euler backward are modified to:

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} \tag{29.57}$$

$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} \tag{29.58}$$

$$\tilde{U}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,1} - \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{ps^\tau_{i-1,jrow-1}}^\phi)) \tag{29.59}$$

$$\tilde{V}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,2} - \delta_\phi(\overline{ps^\tau_{i-1,jrow-1}}^\lambda)) \tag{29.60}$$

The equations for the second step of an Euler backward are modified to:

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} \tag{29.61}$$

$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} \tag{29.62}$$

$$\begin{aligned}\tilde{U}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,1} \\ &- \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda((1-theta)\cdot\overline{ps^\tau_{i-1,jrow-1}}^\phi + theta\cdot\overline{ps^{\tau+1}_{i-1,jrow-1}}^\phi))\end{aligned} \tag{29.63}$$

$$\begin{aligned}\tilde{V}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,2} \\ &- \delta_\phi((1-theta)\cdot\overline{ps^\tau_{i-1,jrow-1}}^\lambda + theta\cdot\overline{ps^{\tau+1}_{i-1,jrow-1}}^\lambda))\end{aligned} \tag{29.64}$$

In both cases of mixing time steps, the time step factor $2\Delta\tau$ is replaced by $\Delta\tau$ (also in ω) and the centering coefficients are given as

- implicit free surface method: Centering coefficient $apgr = theta$. Centering coefficient $gcor = 0$. For implicit treatment of the Coriolis term, $acor$ is reset to $acor = theta$. Also on the second pass of an Euler backward time step the term $\frac{dyt_{jrow}\cos\phi^T_{jrow}dxt_i}{apgr\cdot g\cdot 2\Delta\tau^2}(ps^{\tau+1}_{i,jrow} - ps^{\tau}_{i,jrow})$ must be added to the right hand side of Equation (29.53).
- rigid lid surface pressure method: Centering coefficient $apgr = theta$. Centering coefficient $gcor = 1$ when the Coriolis term is handled explicitly but $gcor = 0$ when the Coriolis term is handled implicitly.

29.4.2 Remarks

29.4.2.1 Boundary conditions

It should be noted that Equation (29.39) only requires a Neumann boundary condition at the boundaries instead of the Dirichlet boundary condition required for the elliptic equation of the stream function method. The implication is that there are no island equations to be solved and hence this method should be faster than the stream function method on massively parallel computers.

29.4.2.2 Conditioning with topography

Another point to be made is that Equation (29.39) contains a factor $H_{i,jrow}$ whereas the elliptic equation for the stream function contains the factor $1/H_{i,jrow}$. Therefore, the implicit free surface method should be less prone to stability problems than the stream function method when topography contains steep gradients. This has yet to be verified.

29.4.2.3 Barotropic velocities

It should also be noted that the barotropic velocities $U^{\tau+1}_{i,jrow}$ and $V^{\tau+1}_{i,jrow}$ given by this method are not non-divergent due to the rising and falling of the sea level (e.g., see discussion in Section 7.2.1).

29.4.2.4 Polar filtering

Use of polar filtering on $\hat{U}$ and $\hat{V}$ in some cases leads to a problem in the external mode in the filtered latitudes. Removing the filtering has been shown to eliminate the problem.

29.4.2.5 Checkerboarding in surface pressure

The surface pressure will sometimes exhibit a checkerboard pattern when using the implicit free surface method. As mentioned in Section 29.3.2.5, the rigid lid - surface pressure method has a zero frequency eigenmode associated with the 9-point Laplacian, and this null mode has the spatial structure of a checkerboard pattern. This mode, which is global in extent, is strictly eliminated with the free surface method (Dukowicz and Smith 1994). However, according to Rick Smith, there might be other low frequency modes of similar structure which are present in the free surface operator. Experience indicates that these modes are more local than the rigid

lid's null mode. In general, the Los Alamos group has found that the checkerboard patterns with the free surface have not presented a problem so far as long-term stability of the run is concerned.

29.5 The Killworth *et al* explicit_free_surface

This section was contributed by Martin Schmidt (*mschmidt@paula.io − warnemuende.de*). Note: as of January, 2000, this method has been removed from MOM 3. The standard MOM 3 explicit free surface is the replacement, and is described in Chapter 30. This section remains for those with the older code wishing to still use the Killworth et al. approach.

Option *explicit_free_surface*, along with one of the two options *explicit_eb* or *explicit_efb* enables the method introduced by Killworth, Stainforth, Webb and Paterson (1991). The free sea surface elevation is treated as an additional prognostic variable calculated together with the vertically integrated barotropic velocity. One major difference between the implicit free surface and explicit free surface code is a small timestep used to resolve the linear components of the barotropic equations. The reason for the smaller time steps in the explicit free surface is due to the allowance of external gravity waves (with wave speeds on the order of $100 - 200\, m/sec$). Note that the method has been found suitable also for open boundary conditions where measured sea levels are prescribed.

29.5.1 The numerical implementation

29.5.1.1 Time stepping

There are two time step implementations discussed by Killworth *et al.* (1989,1991), and Figure 29.1a indicates how the explicit free surface fits into the leapfrog scheme of the baroclinic mode. The barotropic mode starts from level τ, after the baroclinic flow is updated for $\tau + 1$. The barotropic flow is thence updated to baroclinic time level $\tau + 1$. The forcing comes from the τ level, except for the friction terms, which are taken from $\tau - 1$ as required by linear stability. The barotropic mode does not use the leapfrog scheme of the baroclinic mode, and so there is no barotropic mixing time step. Additionally, the Coriolis contribution is updated on each barotropic time step, hence it does not appear in the vertically integrated forcing.

It does not matter much whether the barotropic turbulent momentum exchange is updated for each barotropic substep (Killworth *et al.* (1989)). A considerable amount of CPU time can be saved if the turbulent momentum flux is kept constant over the whole series of the barotropic sub-steps. The differences can be assessed with the option *explhmix*, in which case the barotropic turbulent momentum flux for every sub-step is calculated from the actual barotropic velocity. Note that option *explhmix* requires the use of constant viscosity Laplacian friction.

Two barotropic time step schemes are implemented: a full Euler backward scheme and an Euler forward-backward. The full Euler backward scheme is performed in two steps. In the first step intermediate values η', U' and V' are calculated:

$$\frac{\eta' - \eta}{\Delta t} + \frac{1}{a\cos\phi}\left(\delta_\lambda\left(\overline{U}^\phi\right) + \delta_\phi\left(\overline{\cos\phi V}^\lambda\right)\right) = 0, \tag{29.65}$$

$$\frac{U' - U}{\Delta t} - f\frac{V' + V}{2} + \frac{gH}{a\cos\phi}\delta_\lambda\left(\overline{\eta}^\phi\right) = X_0, \tag{29.66}$$

$$\frac{V' - V}{\Delta t} + f\frac{U' + U}{2} + \frac{gH}{a}\delta_\phi\left(\overline{\eta}^\lambda\right) = Y_0. \tag{29.67}$$

Note the semi-implicit treatment of the Coriolis term. In the second part of a barotropic time step these intermediate values are used to calculate the new barotropic fields η'', U'' and V'',

$$\frac{\eta'' - \eta}{\Delta t} + \frac{1}{a\cos\phi}\left(\delta_\lambda\left(\overline{U'}^\phi\right) + \delta_\phi\left(\overline{\cos\phi V'}^\lambda\right)\right) = 0, \tag{29.68}$$

$$\frac{U'' - U}{\Delta t} - f\frac{V'' + V}{2} + \frac{gH}{a\cos\phi}\delta_\lambda\left(\overline{\eta'}^\phi\right) = X_0, \tag{29.69}$$

$$\frac{V'' - V}{\Delta t} + f\frac{U'' + U}{2} + \frac{gH}{a}\delta_\phi\left(\overline{\eta'}^\lambda\right) = Y_0. \tag{29.70}$$

Then the next barotropic sub-step starts.

In the forward-backward version the sea surface elevation η is updated in a forward step and the new value is used to update the barotropic velocities in a backward step. Explicitly, this scheme takes the form

$$\eta_{\tau+1} = \eta_\tau - \Delta t\left(\frac{1}{a\cos\phi}\left(\delta_\lambda\left(\overline{U_\tau}^\phi\right) + \delta_\phi\left(\overline{\cos\phi V_\tau}^\lambda\right)\right)\right) \tag{29.71}$$

$$U_{\tau+1} = U_\tau + \frac{f\,\Delta t}{2}\left(V_{\tau+1} + V_\tau\right) - \frac{g\,H\,\Delta t}{a\cos\phi}\delta_\lambda\left(\overline{\eta_{\tau+1}}^\phi\right) + \Delta t\,X_0 \tag{29.72}$$

$$V_{\tau+1} = V_\tau - \frac{f\,\Delta t}{2}\left(U_{\tau+1} + U_\tau\right) - \frac{g\,H\,\Delta t}{a}\delta_\phi\left(\overline{\eta_{\tau+1}}^\lambda\right) + \Delta t\,Y_0. \tag{29.73}$$

The properties of both schemes are discussed in Killworth *et al.* (1989, 1991). The full backward scheme numerically damps barotropic waves, except the geostrophic modes. This damping reduces the feedback of barotropic surface waves to the baroclinic mode sampled with a baroclinic time step, which would act as a noise generator. For possible limitations of the tracer time step due to the phase speed of planetary Rossby waves contained in the divergence of the geostropic barotropic velocity field, refer to the Killworth *et al.*. Nevertheless, the Killworth *et al.* explicit free surface method is used mostly for marginal seas of small horizontal extend where the β-effect is small.

If the barotropic waves are a direct point of interest, i.e. for tidal waves, the forward-backward scheme can be used, which does not damp the barotropic waves. Due to the feedback from the barotropic to the baroclinic mode, considerably smaller baroclinic timesteps are necessary. Otherwise the model generates much energy in checkerboard waves and may become unstable.

29.5.1.2 The delplus - delcross filter

The discretization of gradients in the surface pressure introduces computational modes to the surface pressure field. This is a common problem with B-grid implementations, and the result can be grid noise in the free surface height and the vertical velocity $w(z = 0)$. For many situations, this noise is mild. However, when adding fresh water to the model, the noise tends to increase. Additionally, with added realism and length of integration times, the noise appears to be more prominent.

In order to remove the grid splitting which allows for the noisy checkerboard patterns, Killworth *et al.* (1991) found that the implementation of a spatial filter, motivated by similar filters used for atmospheric models, was sufficient. When this filter is applied every barotropic

time step, the computational cost is large. When added only every baroclinic time step, the cost is negligible. The frequency necessary for using this filter depends on the choice for time stepping the free surface. If *explicit_eb* nor *explicit_efb* are enabled, the filter is applied every time step as specified by Killworth *et al.*.

For the surface pressure, the basic idea is to introduce a damping of the form

$$\partial_t\, p_s^{new} = \partial_t\, p_s^{old} + wght\, g\, H\, \Delta t_{bt}\, (dxt_i\, dyt_j\, \cos\phi^T_{jrow})^{-1}\, (\Delta_+ - \Delta_\times)\; p_s^{old}, \tag{29.74}$$

where g is the gravitational acceleration, H is the total model depth, Δt_{bt} is the barotropic time step, $dxt_i\, dyt_j\, \cos\phi^T_{jrow}$ the horizontal area of the $(i, jrow)$ tracer grid cell,

$$\Delta_+\psi = \psi_{i,jrow+1} + \psi_{i+1,jrow} + \psi_{i-1,jrow} + \psi_{i,jrow-1} - 4\,\psi_{i,jrow}, \tag{29.75}$$

is a difference operator for the usual 5-point Laplacian on a uniform grid, and

$$2\,\Delta_\times\psi = \psi_{i+1,jrow+1} + \psi_{i-1,jrow+1} + \psi_{i+1,jrow-1} + \psi_{i-1,jrow-1} - 4\,\psi_{i,jrow} \tag{29.76}$$

is the difference operator for a Laplacian using the diagonal neighbours. Note that in the model, appropriate masking is used for these operators to ensure volume conservation. For long waves, the difference $(\Delta_+ - \Delta_\times)\, p_s^{old}$ is very small and the filter will only mildly influence these waves. For short waves, and especially for checkerboard patterns, the difference can be large. The result is a dissipation of the short waves through a volume transfer between the $+$ and the $\times$ grid. The dimensionless weight factor *wght* rules the strength of the filter and is of order unity. The factor $(dxt_i\, dyt_j\, \cos\phi^T_{jrow})^{-1}$ ensures that the filter conserves volume.

The implementation of the filter follows the code described in Killworth —em et al., which is volume conserving. In order to keep perturbations arising from the filter near coasts small, special artificial values for the surface elevation are prescribed at perimeter land points before the filter is applied. Another implementation of a delplus - delcross filter can be found in the *MOMA* code of the OCCAM group.

Note that the perimeter points used by this filter are land points, but the perimeter points used for the Poisson solvers in MOM are ocean points. Since these ocean perimeter points are never needed if the option *explicit_free_surface* is enabled, the same variables are used to store the land perimeter points for the delplus-delcross filter.

At one point, applying the filter only for each baroclinic time step was tried, but not found suitable with the Killworth *et al.* method. The changes necessary to do so are nonetheless summarized. In this case, at the end of all barotropic time steps, the surface pressure should be updated according to

$$p_s^{new} = p_s^{old} + wght\, g\, H(\Delta t_{bt})^2\, (dxt_i\, dyt_j\, \cos\phi^T_{jrow})^{-1}\; (\Delta_+ - \Delta_\times)\; p_s^{old}. \tag{29.77}$$

The extra factor of Δt_{bt} is needed since the filter is applied here to the pressure and not the tendency of the pressure as in the case above. For some reasonably fine global models (e.g., better than 1°) in which there is polar filtering, it may be necessary to put $wght = min(\cos\phi^T)$ in order to stabilize an instability which appears in the high latitudes. The mechanism for the instability is not well understood, but it might be related to the large differences between cosine of the latitude when approaching the model's highest latitude, hence the reduction of *wght* according to the smallest value of cosine. The $min(\cos\phi^T)$ factor does not appear necessary when running without polar filtering, and is unnecessary in some coarse global models. The default is to place $wght = 1$ in the model. Tuning of this parameter may be necessary for individual needs.

29.5.1.3 Interaction with subroutine *baroclinic*

The coupling to subroutine *baroclinic* differs from the implicit free surface procedure only in the removal of the Coriolis force from the vertically integrated forcing field. Instead, the Coriolis force is introduced explicitly to the barotropic system (see Section 7.2.4).

The horizontal convergence of the vertically integrated velocity is used to calculate the vertical velocity at the surface for the next baroclinic time step. In particular, the vertical velocity at the ocean surface on tracer cells is constructed through discretizing $w(z=0) = -\nabla_h \cdot \mathbf{U}$, which takes the form in the model

$$\begin{aligned} w0_{i,jrow} &= (2\,dxt_i\,dyt_{jrow}\cos\phi^T_{jrow})^{-1} \times \\ &\Big(dyu_{jrow}(U_{i,jrow} - U_{i-1,jrow}) + dyu_{jrow-1}(U_{i,jrow-1} - U_{i-1,jrow-1}) \\ &+ dxu_i(\cos\phi^U_{jrow} V_{i,jrow} - \cos\phi^U_{jrow-1} V_{i,jrow-1}) \\ &+ dxu_{i-1}(\cos\phi^U_{jrow} V_{i-1,jrow} - \cos\phi^U_{jrow-1} V_{i-1,jrow-1}) \Big) \end{aligned} \tag{29.78}$$

The extra factors of dxu and dyu account for the area of the sides of the grid cells. The vertical velocity on velocity cells is determined through the averaging procedure discussed in Section 22.3.2.

29.5.2 Energy analysis

Since there is no barotropic mixing time step, minor changes in the energy analysis are necessary.

The vertical velocity does not vanish at the free surface, and so there is a contribution from the surface level to the work by buoyancy. So the work by surface pressure forces has to be included to guarantee the consistency of the energy analysis. This is also necessary for the implicit solution method.

29.5.3 Options

The options available in connection with the Killworth *et al.* explicit free surface code are the following:

- *explicit_free_surface* enables an explicit free surface scheme. All other barotropic solution schemes have to be disabled.
- *explicit_eb* enables the full Euler backward scheme according to Killworth *et al.*
- *explicit_efb* a forward step is used to calculate the free surface elevation, and the velocity fields are calculated with a full Euler backward step. This again is according to the method of Killworth *et al.*
- *explicit_dpdc* enables the delplus - delcross filter. This method is only available with the Killworth *et al.* explicit free surface. It is not available with the MOM default free surface.
- *explicit_polar_filter* enables the polar filtering on the external mode. Filtering is applied every barotropic time step to the free surface height and the vertically integrated velocity field. Either *firfil* or *fourfil* are possible, though *firfil* is recommended. Polar filtering can

be expensive, and so it is suggested that one reduce the barotropic time step to compare costs of simply resolving the dynamics.

- *explhmix* calculates the turbulent mixing for every barotropic substep. This option is only available with the Killworth *et al.* explicit free surface.

- *explicit_fresh_water_flux* inserts fresh water into the free surface height equation. Conservation of total salt is not realized with the Killworth *et al.* free surface.

29.5.4 Compatibility with other model options

The majority of model options are not influenced by how the barotropic mode is treated. Some obviously incompatible or useless combinations are excluded in *checks*. *rot_grid* is implemented as for the implicit free surface but has not been tested yet. If *explhmix* is enabled, then neither biharmonic nor Smagorinsky mixing is available.

All variables from the island algorithm in the rigid lid version (*nisle, nippts* , ...) are used to identify land perimeter points for the delplus-delcross filter. In future versions of MOM, in which the rigid lid is removed, then so will the delplus-delcross filter.

29.5.5 Test cases

A possible test case is a Rossby adjustment problem, which consists of the following situation. Consider a long zonally oriented rectangular basin with an initial step-like surface elevation in the left part of the basin. Then, a front of barotropic surface gravity waves propagates from the initial step in the surface elevation to the left and the right leaving behind a barotropic flow. The wave front is reflected at the left boundary and later at the right boundary. The initial potential energy is transformed into kinetic energy and back into potential energy if the wave group is reflected at the right boundary. The gross flow patterns have been found to be similar for all free surface options.

29.5.6 Open boundary conditions and river inflow

Another version of open boundary conditions is inspired by the implementation of the Orlanski scheme by Stevens. An arbitrary number of boundaries is possible which need not to be at the outer model rows. Basically, these boundary conditions are similar to those provided by Redler at al., however, the code cannot be merged and the new option *obc_iow* had to be introduced. A. Mutzke has developed a more refined method to calculate the phase speed of waves which may be a source of instability of the Stevens scheme. Moreover, the Sommerfeld radiation condition does not contain the complete information necessary to fix an asymptotic value of the sea level in a basin with an open boundary. Another method is used to decide whether a Sommerfeld radiation condition applies or relaxation to prescribed values takes place. So ill-posed situations as self discharging basins are avoided.

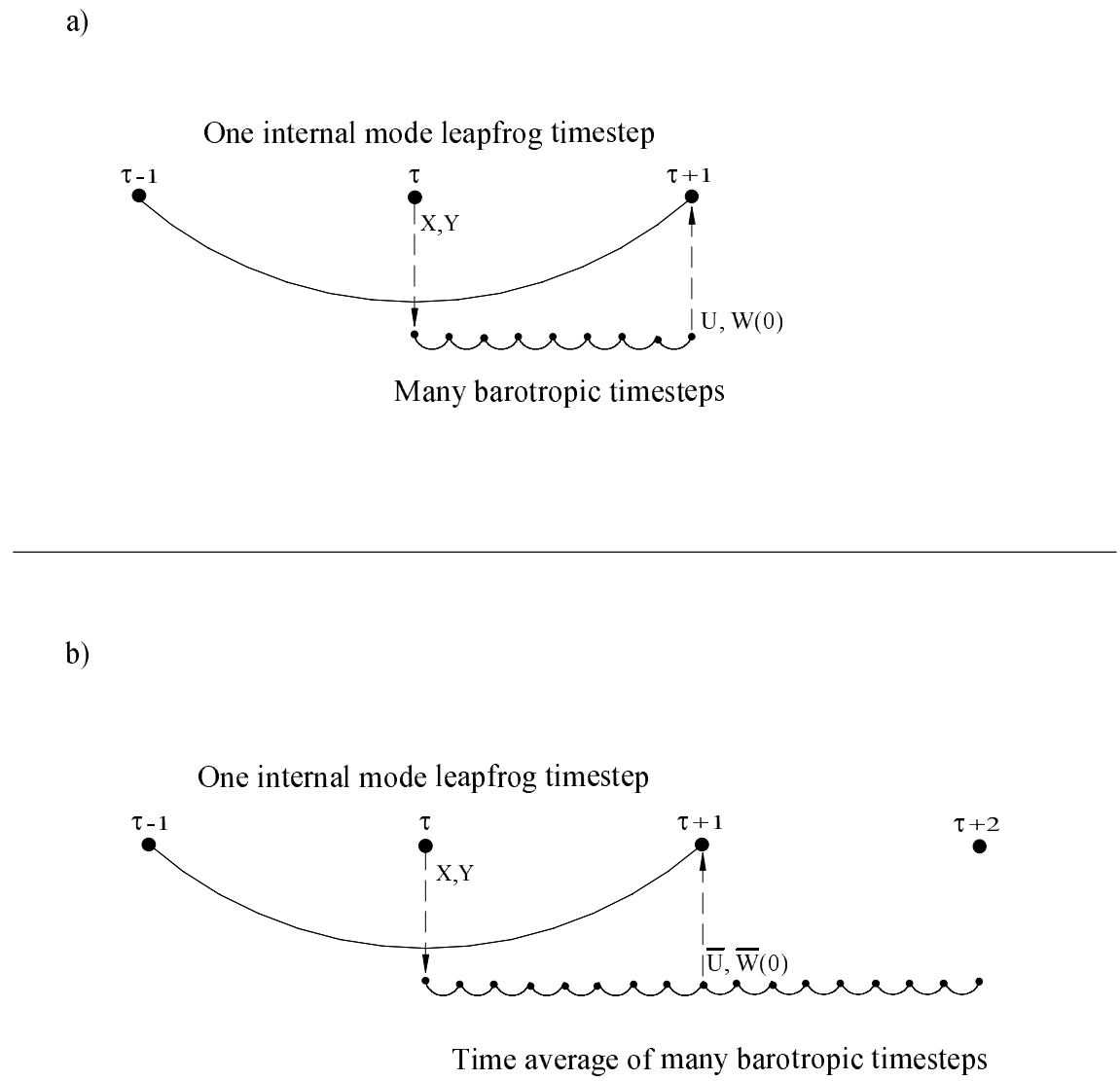


Figure 29.1: (a) Method according to Killworth *et al.* (1991). (b) The standard MOM method which allows stretching the tracer timestep for global climate simulations. This method is described in Chapter 30.

Chapter 30

Explicit free surface and fresh water

Neither of the two time stepping schemes of the Killworth *et al.* (1991) method (see Section 29.5) allows the tracer timestep to be stretched significantly longer than the baroclinic timestep. Taking a tracer timestep which is 10 to 20 times longer than the baroclinic timestep is important for spinning up global ocean simulations on climate time scales. It is for this reason that another time stepping scheme has been implemented, and it has been found to be substantially more stable.

The relation between baroclinic and barotropic timesteps is shown in Figure 29.1b. The barotropic equations are integrated by the previously mentioned forward-backward scheme, but the integration is over two baroclinic timesteps from τ to $\tau + 2$. The solution is time averaged over this period, which centers the barotropic fields at baroclinic time step $\tau + 1$. The time averaged barotropic fields at $\tau + 1$ are used in the rest of the code as the effective barotropic solution, *and* for purposes of initializing the subsequent barotropic integration.

As of Summer 1999, the MOM free surface has incorporated the effects of the undulating surface height into the baroclinic and tracer equations. This element allows the model to be conservative, even in the presence of fresh water forcing. Full details of the algorithm are documented in Griffies, Pacanowski, Schmidt, and Balaji (2000). Much of this chapter is taken from this paper, except for Sections 30.10 and 30.11, which were written by Martin Schmidt (martin.schmidt@io-warnemuende.de) and Stephen Griffies.

The presence of a free surface which undulates as the surface height varies introduces the possibility for much more physically realistic surface boundary conditions on the ocean model. In particular, tracers can be input to the model with values which are distinct from the tracer value in the surface ocean cell. This situation may arise, for example, when coupling the ocean model to a river. Section 30.11 discusses, in detail, the issues of fresh water forcing in MOM, and introduces a means for interfacing MOM to river models.

30.1 Free surface options

The options available in connection with the Griffies, Pacanowski, Schmidt, and Balaji (2000) MOM explicit free surface code are the following:

- *explicit_free_surface* enables the explicit free surface scheme. All other barotropic solution schemes have to be disabled. So long as neither *explicit_eb* nor *explicit_efb* are enabled, the algorithm is that discussed in this section, and is *not* that of Killworth *et al.* This option

must also have either *explicit_free_surface_nonlinear* or *explicit_free_surface_linear* enabled. There is no default.

- *explicit_free_surface_nonlinear* provides the full effects of the surface height are incorporated. In order to do so, it is necessary to also enable option *partial_cell*. This method will conserve total tracer amount.

- *explicit_free_surface_linear* enables a free surface in which the baroclinic and tracer equations employ constant volume upper model cells. This option yields a linearized free surface which is similar in principle to that of Killworth *et al* (1991) and Dukowicz and Smith (1994). This method will not conserve total tracer amount, which is a limitation shared by the Killworth *et al* (1991) and Dukowicz and Smith (1994) methods.

- *explicit_eta_laplacian* enables a Laplacian filter to reduce the checkerboard null mode in the sea surface height. Filtering is done after a sequence of barotropic time steps. More details can be found in Section 30.12.

- *explicit_eta_dpdc* enables a "del-plus / del-cross" filter to reduce checkerboard null mode in the sea surface height. As default filtering is done after a sequence of barotropic time steps. Option *explicit_eta_dpdc_all* enables filtering for each barotropic time step. More details can be found in Section 30.12.

- *explicit_polar_filter* enables the polar filtering on the external mode. Filtering is applied every barotropic time step to the free surface height and the vertically integrated velocity field. Either *firfil* or *fourfil* are possible, though *firfil* is recommended. Polar filtering can be expensive, and so it is suggested that one reduce the barotropic time step to compare costs of simply resolving the dynamics. Polar filtering the barotropic fields also introduces a lot of noise to the solution, as discussed in Sections 30.12 and 30.13.

- *explicit_fresh_water_flux* inserts fresh water into the free surface height equation. Conservation of total salt is realized so long as the tracer and baroclinic time steps are equal. See Section 30.11 for further options related to water forcing.

- *salinity_psu* computes model salinity in units of psu, rather than the traditional model units $(s - 35)/1000$. Option *salinity_psu* is necessary with *explicit_fresh_water_flux* in order to get the correct sign in the salinity equation in regions where salinity is less than 35.

Options which are not available with the MOM free surface are *explhmix*, *explicit_eb*, *explicit_efb*.

The remainder of this chapter presents the details of the MOM explicit free surface method. More complete treatment, with examples, is provided in Griffies, Pacanowski, Schmidt, and Balaji (2000).

30.2 Momentum equations

Figure 30.1 details a zonal-depth cross-section of the *rectangular* control volumes over which the model's surface equations are discretized. Notably, the position of a grid point within a cell is assumed to be fixed in time, hence maintaining the Eulerian nature of MOM. However, the corresponding grid cell volume generally changes according to temporal undulations of the surface height. We do *not* make the common approximation that the surface height is small

relative to any other length scale in the model, including the ocean depth. Consequently, the resulting barotropic equations represent nonlinear shallow water equations. In the following, we focus on the zonal momentum budget as it is sufficient to expose the essence of the numerical algorithm.

Ignoring meridional gradients for brevity, the continuous time budget for the zonal momentum of a Boussinesq fluid occupying a rectangular velocity cell is given by

$$\begin{aligned} \partial_t (\rho_o \, A \, h^u \, u)_i &= (\rho_o \, A \, h^u \, f \, v)_i - h^u \, dy \, (p_{i+1} - p_i) \\ &- \rho_o \, dy \, (F^x_{i+1} - F^x_i) - \rho_o \, A \, (F^z_{k-1} - F^z_k). \end{aligned} \tag{30.1}$$

In this equation, $A = dx\,dy$ is the horizontal cross-sectional area of the velocity cell, h^u is its thickness, i denotes the zonal grid point and k the vertical, and these labels are exposed only when needed. Metric terms arising from momentum advection and friction on a sphere (e.g., Wajsowicz 1993, Griffies and Hallberg 1999) have been omitted for brevity, but they are present in the numerical code. The constant Boussinesq density, ρ_o, is not set to $1\,\mathrm{g\,cm^{-3}}$, as previously done in the Bryan-Cox-Semtner model (Bryan 1969, Cox 1984, Semtner 1974) or previous versions of MOM (Pacanowski, Dixon, and Rosati 1991). Instead, $\rho_o = 1.035\,\mathrm{g\,cm^{-3}}$, from which density in the World Ocean generally deviates by less than 2% (Gill 1982, page 47), whereas using $\rho_o = 1.0\,\mathrm{g\,cm^{-3}}$ is less accurate.

F^x represents the *thickness weighted* zonal advective and turbulent momentum fluxes passing across the vertical faces of the cell. To compute these fluxes with space-time varying cell thicknesses, we exploit the numerical methods developed by Pacanowski and Gnanadesikan (1998) for use with a partial bottom cell representation of topography. Notably, those methods provide a means to discretize the model equations when the cell thicknesses are functions of horizontal position. It is a straightforward extension of their work to allow the cell thicknesses to be functions of time. For further numerical details, refer to Pacanowski and Gnanadesikan (1998).

F^z represents the vertical advective and turbulent momentum flux passing across the horizontal cell faces. In particular, $F^z_{k=0}$ represents the momentum flux passing through the ocean surface at $z = \eta$. It consists of the usual vertical turbulent momentum flux and a contribution from the vertical advection of momentum relative to the moving sea surface,

$$F^z_{k=0} = F^{z,turb}_{k=0} - q_w \, u_{k=0}. \tag{30.2}$$

The surface flux $F^z_{k=0}$ must be prescribed by the boundary condition that the total vertical momentum flux through the air sea interface is continuous at $z = \eta$. That is, $F^z_{k=0} = \mathcal{M}$, where the prescribed momentum flux $\mathcal{M}$ takes the form

$$\mathcal{M} = -(\rho_o^{-1} \, \tau^x_{winds} + q_w \, u_w), \tag{30.3}$$

which has contributions from wind stress and momentum flux with fresh water. Although the total momentum flux is continuous at $z = \eta$, the two components need not be. For example, the fresh water velocity u_w may be different from $u_{k=0}$, although most climate models assume they are equal and take them equal to the horizontal velocity $u_{k=1}$ in the top model grid cell. This point is further discussed below.

On the B-grid used in MOM, the offset in the horizontal between velocity and tracer/density points means that the hydrostatic pressure at the western face of a velocity cell is given by

$$p_i = g \, \overline{\rho_i \, (|z_1|/2 + \eta_i)}^{\,y} + p^{atm}_i, \tag{30.4}$$

where $\overline{\alpha}^y$ is a meridional grid average. $p_b = (g\,|z_1|/2)\,\overline{\rho_i}^y$ is a discretization of the baroclinic pressure, whose horizontal gradients arise from baroclinicity in the density field. Note that z_1 has been assumed independent of horizontal position, hence its removal from the meridional average operator. $p_s = g\,\overline{\rho_i\,\eta_i}^y$ is a discretization of the surface pressure, whose gradients arise from those in the density weighted free surface height. p_i^{atm} represents an atmospheric pressure contribution. It is dropped in the following for brevity, yet is maintained in the numerical code when coupling to atmospheric models.

For Boussinesq models, it is common to approximate the surface pressure as $p_s \approx g\,\rho_o\,\overline{\eta_i}^y$, rather than the hydrostatic form $g\,\overline{\rho_i\,\eta_i}^y$. With this approximation, surface pressure gradients arise *solely* from gradients in the free surface height. To maintain self-consistency with the hydrostatic baroclinic pressure field, while incurring only trivial computational expense, we maintain the hydrostatic form $p_s = g\,\overline{\rho_i\,\eta_i}^y$ in the model.

As discussed in the next subsection, time stepping of the baroclinic velocity in MOM is performed using the average cell velocity. Dividing the budget (30.1) by the generally time dependent volume of the velocity cell $h^u\,dx\,dy$, and performing the product rule on the time derivative $\partial_t(h^u\,u)$, leads to the zonal velocity equation

$$\partial_t\,u_k = f\,v_k - \partial_x\,p_s/\rho_o + \tilde{G}_k \tag{30.5}$$

where $\tilde{G}_k$ represents the forcing

$$\begin{aligned} h_k^u\,\tilde{G}_k &= h_k^u\,G_k - \delta_{k1}\,u_k\,\partial_t\eta \\ &= -h_k^u\,\partial_x\,p_b/\rho_o - \partial_x\,F^x - (F_{k-1}^z - F_k^z + \delta_{k1}\,u_k\,\partial_t\eta). \end{aligned} \tag{30.6}$$

The $u_1\,\partial_t\eta$ term and the time dependent surface cell thicknesses represent fundamentally new elements relative to the traditional rigid lid analogs of these equations. Notably, the cell thicknesses h^u used for computing the terms in G_k are taken at baroclinic time τ, as are the other inviscid contributions such as pressure and advection.

The fresh water velocity u_w in the surface momentum flux (30.3) requires additional efforts to specify the complete boundary conditions. However, the momentum flux with fresh water is in many cases smaller than the uncertainty in the parameterized wind stress. So, simple approximations for u_w may exploited to reduce the model complexity. Considering

$$u_1\,\partial_t\eta + F_{k=0}^z = -\rho_o^{-1}\,\tau_{winds}^x - u_1\,\nabla_h\cdot\mathbf{U} + q_w\,(u_1 - u_w)\,, \tag{30.7}$$

it is seen that the approximation $u_w \approx u_1$ removes the explicit dependence of baroclinic momentum on the fresh water flux. Fresh water does influence baroclinic momentum indirectly, however, through its affects on the convergence $-\nabla_h\cdot\mathbf{U}$ of the vertically integrated velocity. The approximation $u_w \approx u_1$ should be well justified for many cases, and generalizations for special cases such as heavy rainfall are straightforward.

Instead of time stepping the barotropic velocity, MOM time steps the vertically integrated transport $U = \sum_k h_k\,u_k$, as is common in shallow water models. Its evolution takes the form

$$\begin{aligned} \partial_t U &= u_1\,\partial_t\eta + \sum_k h_k\,\partial_t u_k \\ &= f\,V - D\,\partial_x\,p_s/\rho_o + G, \end{aligned} \tag{30.8}$$

where equation (30.5) was used for $\partial_t u_k$, the vertically integrated forcing is given by $G = \sum_k h_k\,G_k$ with G_k defined in equation (30.6), and $D = \sum_k h_k$ is the time dependent ocean depth. Once the transport and ocean depth are updated, the updated barotropic velocity can be diagnosed through $\overline{u} = U/D$.

30.3 Time stepping algorithm

The focus in this subsection is on time and depth discretization, with Figure 30.2 summarizing the following algorithm. For purposes of brevity, the horizontal spatial discretization discussed in the previous subsection will not be exposed. Also, discrete baroclinic times and time steps will be denoted by the Greek τ and $\Delta\tau$, respectively, whereas the barotropic analogs will use the Latin t and Δt.

The basic idea is to split the velocity at an arbitrary depth level k and baroclinic time $\tau' = \tau + \Delta\tau$ into two components

$$\begin{aligned} u_k(\tau') &= B_{km}(\tau)\, u_m(\tau') + (\delta_{km} - B_{km}(\tau))\, u_m(\tau') \\ &\equiv \hat{u}_k(\tau, \tau') + \overline{u}(\tau, \tau'). \end{aligned} \tag{30.9}$$

The following "baroclinicity operator" is used to affect this split

$$B_{km}(\tau) = \delta_{km} - D(\tau)^{-1}\, h^u_m(\tau), \tag{30.10}$$

where δ_{km} is the Kronecker delta, summation over the repeated vertical level index m is implied, and

$$D(\tau) = \sum_k h^u_k(\tau) = D_o + \eta^u(\tau) \tag{30.11}$$

is the ocean depth at baroclinic time τ over a column of velocity points, with D_o the resting ocean depth.

The introduction of two baroclinic time labels to equation (30.9) is necessitated by the freedom afforded the ocean depth to change in time. This property is in contrast to the case with a rigid lid, or the fixed volume free surface models of Killworth *et al.* (1991) and Dukowicz and Smith (1994).

Equation (30.9) is an identity which is valid for any baroclinic times τ' and τ. Its utility depends on the ability to render a relatively clean and stable split between the fast and slow dynamics. The form of the baroclinicity operator $B_{km}(\tau)$ is motivated by an attempt to perform such a split. That is, it projects out the approximate baroclinic portion of a field, where the projection is based on the distribution of cell thicknesses at time τ.

If the split introduced in equation (30.9) is successful, the baroclinic velocity $\hat{u}_k(\tau, \tau')$ will evolve on a slow time scale $\Delta\tau$. In turn, the barotropic velocity $\overline{u}(\tau, \tau')$ will evolve on the fast time scale $\Delta t = 2\, N^{-1}\, \Delta\tau$, with N determined by the ratio of external to internal gravity wave speeds ($N \approx 100$ for climate models). The method therefore proceeds by separately updating $\hat{u}_k(\tau, \tau')$ and $\overline{u}(\tau, \tau')$ by exploiting the time scale split. Upon doing so, the right hand side of the identity (30.9) will be specified, hence allowing for an update of the full velocity field $u_k(\tau + \Delta\tau)$. The following discussion details these ideas.

By construction, evolution of the baroclinic mode is unaffected by vertically independent forces, such as those from surface pressure gradients. Therefore, it is sufficient to update the "primed" velocity

$$u'_k(\tau + \Delta\tau) = u^R_k(\tau - \Delta\tau) + 2\Delta\tau\, [f\, v_k(\tau) + \tilde{G}_k(\tau)], \tag{30.12}$$

which represents a temporal discretization of the full momentum equation (30.5), yet without the surface pressure gradient. The lagged velocity $u^R_k(\tau - \Delta\tau)$ is a Robert time filtered version of the full velocity field. A weak form of such filtering has been found sufficient to suppress a splitting between the two branches of the leap-frog (e.g., Haltiner and Williams 1980). It is also

useful to dampen any fast dynamics which may partially leak through the baroclinicity operator due to the imperfect baroclinic/barotropic split discussed in Section 5.1.1. The baroclinic piece of the primed velocity $u'_k(\tau + \Delta\tau)$ is equivalent to the updated baroclinic velocity

$$\hat{u}_k(\tau, \tau + \Delta\tau) = B(\tau)_{km}\, u'_m(\tau + \Delta\tau), \tag{30.13}$$

thus specifying one half of the identity (30.9).

To update the barotropic velocity $\overline{u}(\tau, \tau + \Delta\tau)$, we employ a forward-backward time stepping scheme (e.g., Haltiner and Williams 1980). For this scheme, the surface height is time stepped using the small barotropic time step Δt

$$\eta^*(t_{n+1}) = \eta^*(t_n) - \Delta t\, [\nabla_h \cdot \mathbf{U}^*(t_n) - q_w(\tau)], \tag{30.14}$$

with $t_n = n\,\Delta t$ the barotropic time, and $n \in [0, N]$. The asterisk is used to denote intermediate values of the barotropic fields, each of which are updated on the barotropic time steps. For stability purposes, it is important to take the initial condition $\eta^*(0)$ as the time average $\overline{\eta}^*$ from the previous barotropic integration (time averaging is defined by equation (30.17) discussed below). Note that it is assumed that the fresh water flux q_w is constant over the small barotropic time steps, since the hydrological fluxes are typically updated at a period no shorter than the baroclinic time step.

Having the surface height η^* updated to the new barotropic time step allows for an update of the transport

$$\begin{aligned} U^*(t_{n+1}) &= U^*(t_n) + f\,\Delta t \left(\frac{V^*(t_n) + V^*(t_{n+1})}{2} \right) \\ &+ \Delta t\, [G(\tau) - D(\tau)\, \partial_x\, \tilde{p}^*_s(t_{n+1})], \end{aligned} \tag{30.15}$$

where $\tilde{p}^*_s(t_{n+1}) = g\,\eta^*(t_{n+1})\,\rho(\tau)/\rho_o$ is the surface pressure normalized by the Boussinesq density. Both the ocean depth $D(\tau)$ and depth integrated forcing $G(\tau)$ are assumed to evolve on the baroclinic time scale, and so are held constant over the extent of the barotropic time steps from τ to $\tau + 2\Delta\tau$. The Coriolis force is computed using a Crank-Nicholson semi-implicit time stepping scheme (e.g., Haltiner and Williams 1980).

After N barotropic time steps, the vertical transport and surface height are time averaged to produce

$$\overline{U}^*(\tau + \Delta\tau) = \frac{1}{N+1} \sum_{n=0}^{N} U^*(t_n) \tag{30.16}$$

$$\overline{\eta}^*(\tau + \Delta\tau) = \frac{1}{N+1} \sum_{n=0}^{N} \eta^*(t_n). \tag{30.17}$$

The time averaged fields are centered on the baroclinic time step $\tau + \Delta\tau$, so long as N is an even integer. Equating the barotropic velocity $\overline{u}(\tau, \tau + \Delta\tau)$ to $\overline{U}^*(\tau + \Delta\tau)/D(\tau)$ allows for the full velocity field $u(\tau + \Delta\tau)$ to be updated according to equation (30.9). The time averaged field $\overline{\eta}^*(\tau + \Delta\tau)$ is used to initialize the next suite of barotropic integrations from $\tau + \Delta\tau$ to $\tau + 3\Delta\tau$.

To begin the next baroclinic time step, it is necessary to determine the updated surface height $\eta(\tau + \Delta\tau)$ and vertically integrated velocity $U(\tau + \Delta\tau)$. A natural choice is to equate these fields to the time averages of the intermediate fields η^* and U^* given by equations (30.16)

and (30.17). Unfortunately, this choice does not lead to a self-consistent and conservative algorithm. The reason is that it is essential to maintain consistency between the updated ocean depth, full velocity, barotropic velocity, all while maintaining basic conservation properties. The following approach has been found to be sufficient for these purposes.

As discussed in Section 30.9, since the tracer concentration is time stepped with a leap-frog scheme, quasi-conservation of tracer results if the surface height is similarly time stepped

$$\eta(\tau + \Delta\tau) = \overline{\eta}^*(\tau - \Delta\tau) - 2\,\Delta\tau\,[\nabla_h \cdot \mathbf{U}(\tau) - q_w(\tau)]. \tag{30.18}$$

To maintain stability and smoothness of the solutions, it has been found necessary to use the lagged surface height $\overline{\eta}^*(\tau - \Delta\tau)$ rather than a more traditional Robert filtered height. As so defined, $\eta(\tau + \Delta\tau)$ is used to update thicknesses of the surface grid cells $h_k^t(\tau + \Delta\tau)$ and $h_k^u(\tau + \Delta\tau)$. Given these new thicknesses, the updated transport $U(\tau + \Delta\tau)$ can be diagnosed from the known full velocity $u_k(\tau + \Delta\tau)$ through

$$U(\tau + \Delta\tau) = \sum_k h_k^u(\tau + \Delta\tau)\, u_k(\tau + \Delta\tau), \tag{30.19}$$

thus ensuring self-consistency between the updated full velocity and the updated vertically integrated velocity. That is, with time dependent thicknesses, $U(\tau + \Delta\tau)$ differs from the time averaged field $\overline{U}^*(\tau + \Delta\tau)$ because of the changing level thicknesses. To complete the time stepping, the updated barotropic velocity is diagnosed through

$$\overline{u}(\tau + \Delta\tau) = U(\tau + \Delta\tau)/D(\tau + \Delta\tau), \tag{30.20}$$

where $D(\tau + \Delta\tau) = \sum h_k^u(\tau + \Delta\tau)$ is the new depth of a velocity cell column.

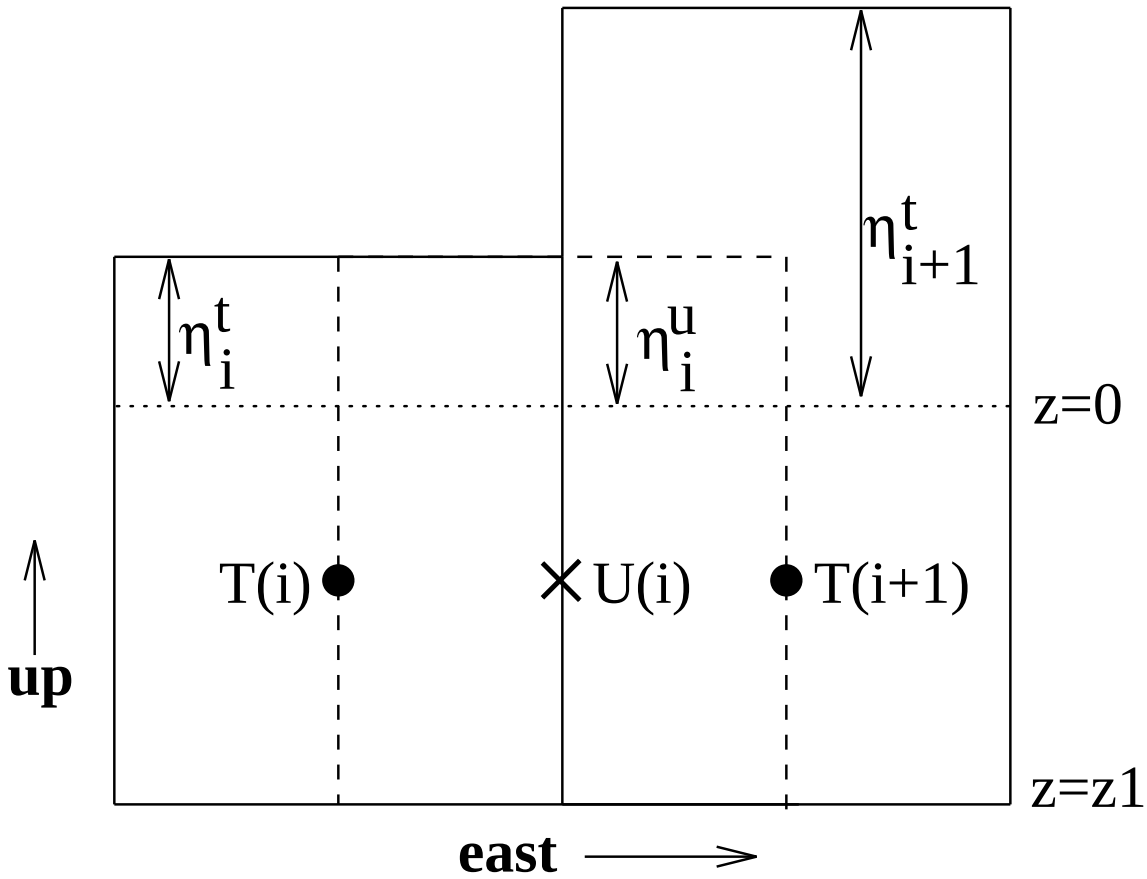


Figure 30.1: Schematic of rectangular surface tracer and velocity cells with the free surface. The tracer points are denoted by a solid dot and the tracer cells are enclosed by solid lines; a velocity point is denoted by an "x" and is enclosed by dashed lines. These points have fixed vertical position $z = z_1/2 < 0$, *regardless* of the value for the free surface height. The thickness of a tracer cell is $h^t = -z_1 + \eta^t$, where η^t is the prognostic surface height determined through volume conservation (i.e., the vertically integrated continuity equation (7.18)). The thickness of a velocity cell is $h^u = -z_1 + \eta^u$, where η^u is determined by taking the minimum height of the four surrounding tracer cells, as prescribed by the partial cell approach of Pacanowski and Gnanadesikan (1998). Thicknesses of the cells are assumed to be positive, so that $-z_1 + \eta^t > 0$. In models without an explicit representation of tides, this constraint in practice means that $|z_1|$ must be greater than roughly 2m. For models with tides, $|z_1|$ may need to be somewhat larger, depending on the tidal range considered.

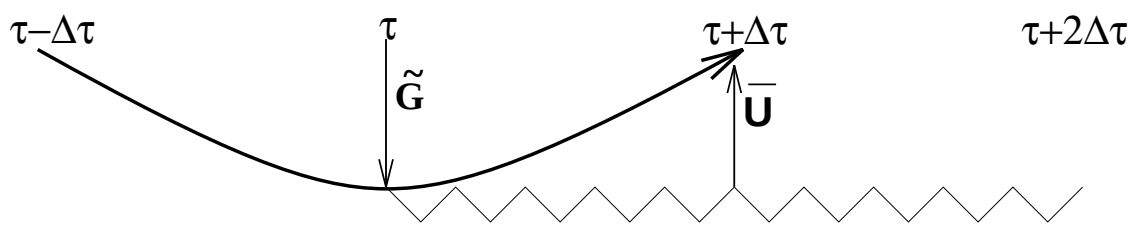


Figure 30.2: Schematic of the split-explicit time stepping scheme. Time increases to the right. The baroclinic time steps are denoted by $\tau - \Delta\tau, \tau, \tau + \Delta\tau$, and $\tau + 2\Delta\tau$. The curved line represents a baroclinic leap-frog time step, and the smaller barotropic time steps $N\,\Delta t = 2\,\Delta\tau$ are denoted by the zig-zag line. First, a baroclinic leap-frog time step updates the baroclinic mode to $\tau + \Delta\tau$. Then, using the vertically integrated forcing $G(\tau)$ (equation 30.8) computed at baroclinic time step τ, a forward-backward time stepping scheme integrates the surface height and vertically integrated velocity from τ to $\tau + 2\Delta\tau$ using N barotropic time steps of length Δt, while keeping $G(\tau)$ and the ocean depth $D(\tau)$ fixed. Time averaging the barotropic fields over the $N + 1$ time steps (endpoints included) centers the vertically integrated velocity and free surface height at baroclinic time step $\tau + \Delta\tau$. Note that for accurate centering, N is set to an even integer.

30.4 Vertical velocities

Vertical velocities are diagnosed at baroclinic time steps using volume conservation within a grid cell. In particular, volume conservation over a surface cell indicates that the vertical velocity at the bottom face of this cell arises from the horizontal convergence of volume in this cell, time tendencies in the thickness of this cell, and volume passing across the top face from fresh water fluxes.

It is useful to see precisely how these velocities are determined. For this purpose, consider a vertical integral of the continuity equation $\nabla \cdot \mathbf{u} = 0$ over a rectangular top model grid cell

$$w_{k=1} = w(\eta) + \int_{z_1}^{\eta} dz \, \nabla_h \cdot \mathbf{u}_h. \tag{30.21}$$

Use of the surface kinematic boundary condition

$$(\partial_t + \mathbf{u}(\eta) \cdot \nabla_h) \, \eta = w(\eta) + q_w, \tag{30.22}$$

and Leibnitz's Rule with $\nabla_h z_1 = 0$ leads to

$$\begin{aligned} w_{k=1} &= \partial_t \eta - q_w + \nabla_h \cdot \int_{z_1}^{\eta} dz \, \mathbf{u}_h \\ &= -\nabla_h \cdot \mathbf{U} + \nabla_h \cdot \int_{z_1}^{\eta} dz \, \mathbf{u}_h, \end{aligned} \tag{30.23}$$

where the last step used the vertically integrated continuity equation (7.18). This exact expression for $w_{k=1}$ is approximated in the model with a discretized version of

$$w_{k=1} \approx -\nabla_h \cdot \mathbf{U} + \nabla_h \cdot (h \, \mathbf{u}_h), \tag{30.24}$$

where $h = \eta + |z_1|$ is the surface cell thickness, and the horizontal velocity $\mathbf{u}_h$ is that in the surface cell $k = 1$. Notably, many implementations of the free surface linearize the surface kinematic boundary condition by dropping the surface height advection $\mathbf{u}(\eta) \cdot \nabla_h \eta$. No such approximation has been made here.

Given an expression for the convergence $-\nabla_h \cdot \mathbf{U}$, diagnosing $w_{k=1}$ in this manner allows for the remaining interior vertical velocities to be successively found through further integration of the continuity equation downward through a vertical column. On a B-grid, $-\nabla_h \cdot \mathbf{U}$ is centered on a tracer point. Hence, equation (30.24) yields the vertical velocity on the bottom face of the surface tracer cell. The vertical velocity on the surrounding velocity cells is constructed as a volume conserving average of the surrounding tracer cell vertical velocities.

In MOM, the bottom of the ocean on tracer cells is a flat surface, representing the "lopped off" surfaces of topography[1]. Hence, the vertical velocity must vanish at this location. A self-consistency check on how accurate the model's numerics conserve volume amounts to testing how well this property is satisfied when integrating downwards from the ocean surface, starting from the vertical velocity given by equation (30.24). Adding fresh water to the model provides a nontrivial test of these properties. The present scheme produces zero vertical velocities at the bottom of tracer cells, to within computer roundoff, regardless of the topography or surface forcing.

[1]The bottom velocity cell generally does not sit on the ocean bottom, and so can support a vertical velocity due to sloping topography. Details of how MOM handles this velocity are given in Webb (1995) and Section 22.3.3.

30.5 Comments on the algorithm

The time averaged portion of the algorithm provides a fairly heavy filter of the fast barotropic gravity waves active on the barotropic time scales. Since these waves are not of interest here, such filtering is justified. As might be expected, time averaging greatly stabilizes the scheme so that stretching of tracer time steps to values larger than the baroclinic time step is available. Note that the Killworth *et al.* (1991) scheme has no time averaging, and correspondingly we have found it to be quite unstable with stretched tracer time steps.[2] The stretching of tracer time steps is ubiquitous in coarse resolution ocean models (e.g., Bryan 1984, Killworth *et al.* 1984, Danagasoglu *et al.* 1996), and so we consider the added stability of the present scheme to be of great practical value.

Allowing the grid cell thicknesses to evolve introduces a fundamentally new element to the traditional algorithms relevant for constant cells in z-coordinate models. However, since the model does not integrate the thickness equation at each vertical depth, as required in an isopycnal layered model, and since we impose positive cell thicknesses for all cells, including the surface, modifications to the constant cell approach are relatively modest.

Although the immediate application of the algorithm is for the free surface method, the algorithm is general so that any thickness within a vertical column can vary in time, within the constraints of positive cell thicknesses mentioned above. In particular, it is thought that this approach will find additional use for models with a time varying bottom boundary layer thickness, such as that proposed by Killworth and Edwards (1999).

30.6 Discrete tracer budgets

The purpose of this section is to derive the discrete tracer budgets within the free surface ocean model with a time dependent top cell volume.

As for the formulation of the discrete momentum equations in Section 30.2, the continuous time budget for the total amount of tracer within a given tracer cell (Figure 30.1) is given by

$$\partial_t \left(A\, h^t\, T \right) = -dy \left(F^x_{i+1} - F^x_i \right) - A \left(F^z_{k-1} - F^z_k \right), \tag{30.25}$$

where T is the tracer per unit volume, i.e., a tracer concentration, A is now the horizontal area of the tracer cell, h^t is the thickness of the tracer cell, meridional gradients are omitted for brevity, and the thickness weighted horizontal advective and diffusive fluxes F^x are computed as in Pacanowski and Gnanadesikan (1998) to account for the generally different adjacent cell thicknesses.

F^z_k is the vertical turbulent and advective tracer flux through the cell interface k. The special term $F^z_{k=0}$ for the vertical tracer flux at the sea surface $z = \eta$ involves vertical tracer advection, relative to the undulating sea surface, by fresh water, as well as the usual parameterized turbulent flux

$$F^z_{k=0} = F^{z,turb}_{k=0} - q_w\, T_{k=0}. \tag{30.26}$$

The tracer flux $F^z_{k=0}$ must be calculated from a boundary condition which equates this flux with the total tracer flux Q_T through the surface. The flux Q_T generally has a contribution

[2]Tests conducted at the Southampton Oceanography Center indicate a need to perform a time average over barotropic time steps, even when using equal tracer and baroclinic time steps, to suppress an instability associated with aliased barotropic Rossby waves near the equator (David Webb, personal communication).

from parameterized turbulence as well as a tracer flux with fresh water,

$$Q_T = Q_T^{turb} - q_w \, T_w, \tag{30.27}$$

where T_w is the tracer concentration in the fresh water. Although T_w and $T_{k=0}$ may be of the same order of magnitude, the terms $q_w \, T_w$ and $q_w \, T_{k=0}$ stand for different physical processes. The term $q_w \, T_w$ represents the amount of tracer passing through the air sea interface with fresh water, whereas $q_w \, T_{k=0}$ represents the advection of tracer in the ocean relative to the sea surface. Note that for the special case of salt, the total surface salt flux Q_s is well approximated to be zero for climate modeling. More will be said about salt below.

In general, specification of the surface tracer flux involves details of how the air-sea interface is modeled. A simple example of a "closure" for the turbulence term is a restoring condition $Q_T^{turb} = \gamma \, h^t \, (T_1 - T^*)$, where γ is an inverse damping time and T_1 is the time lagged surface tracer concentration. For the advection term, a simple choice is to set the tracer concentration T_w in the fresh water equal to the surface cell concentration T_1, now without time lag as it represents an advective contribution. Other forms are generally appropriate when employing ocean models with more realistic surface boundary conditions, such as for coupled climate simulations.

30.7 Time discretization of the tracer budgets

The question arises whether it is more suitable to time step the thickness weighted tracer concentration $h^t \, T$, or the tracer concentration T. Indeed, this question is also relevant for the baroclinic velocity equations. As discussed in Section 30.2, we found time stepping the baroclinic velocity, rather than the thickness weighted baroclinic velocity, to be suitable and to involve only a small number of changes to the equations relevant for the rigid lid version of MOM. Energetic consistency, in which buoyancy effects in the density equation balance the work done by pressure from the momentum equation, and convenience motivate a similar temporal discretization of the tracer budget.

Our starting point is the tracer concentration budget, which is obtained by dividing the tracer budget (30.25) by the tracer cell cross-sectional area, and performing the product rule on $\partial_t(h^t \, T)$

$$h^t \, \partial_t \, T = -\partial_x \, F^x - (\delta_{k1} \, T_k \, \partial_t \eta + F^z_{k-1} - F^z_k). \tag{30.28}$$

The $\delta_{k1} \, T_k \, \partial_t \eta$ term, as well as the time dependent thickness h^t, distinguish this budget from that considered with the partial bottom cells of Pacanowski and Gnanadesikan (1998). Note that as for the momentum equations, the thickness is computed at the present time step $h^t(\tau)$ as are other non-diffusive terms.

The tracer concentration budget (30.28) is time stepped with a leap-frog scheme along with the same Robert time filter used with the baroclinic integration. Upon setting the tracer to unity and omitting boundary fluxes, the tracer budget reduces to volume conservation. This self-consistency between tracer and volume conservation must be maintained with the discretized equations in order to allow for tracer conservation (Section 30.9).

30.8 Further comments on surface fluxes and the case of salt

It is useful to further elucidate the different terms describing the surface tracer fluxes. Additionally, the special case of salt deserves some comments.

Use of the free surface equation (7.18) and the surface tracer flux expression (30.27) leads to the relation

$$\begin{aligned} T_1\,\partial_t\eta + F^z_{k=0} &= T_1\,\partial_t\eta + Q_T \\ &= Q_T^{turb} - T_1\,\nabla_h\cdot\mathbf{U} + q_w\,(T_1 - T_w)\,. \end{aligned} \tag{30.29}$$

Hence, only the balance of the advective and turbulent tracer fluxes enters the model equations. Notably, neither the tracer $T_{k=0}$ nor the turbulent flux $F^{z,turb}_{k=0}$ are explicitly needed. Relatedly, the distinction should be made between $q_w\,T_1$, which naturally appears here in the relation (30.29), and $q_w\,T_{k=0}$, which appears in the expression for $F^z_{k=0}$ in equation (30.26).

The relation (30.29) is analogous to the surface momentum flux relation given by equation (30.7). Assuming that the tracer concentration in the fresh water is the same as in the surface cell, $T_w \approx T_1$, makes the tracer time tendency independent of any explicit fresh water forcing. Instead, the dependence is restricted to the affects that fresh water has on the convergence $\nabla_h\cdot\mathbf{U}$. This approximation simplifies the setup of boundary conditions for the tracer flux. However, in general T_w and T_1 are distinct and so T_w may need to be specified explicitly from data or another component model.

For salt, due to a large hydration energy, the air-sea interface effectively acts as an impenetrable barrier. Therefore, neglecting the formation of sea spray and the interchange of salt with sea ice, and in the absence of salt entering through the solid earth boundaries, total ocean salt is constant. In turn, the salt concentration $\rho_o\,s$ within a grid cell, where s is salinity, changes only through advective and turbulent fluxes from the interior ocean, as well as time tendencies in the volume of the grid cell. From the flux expression (30.27), a zero salt flux through the air-sea interface formally represents a balance

$$F^z_{k=0} = F^{z,turb}_{k=0} - q_w\,s_{k=0} = Q_s = 0 \tag{30.30}$$

between the turbulent flux and advective flux. Discussion of this balance has been given by Huang (1993), where the turbulent flux was called an *anti-advective flux*. Specifying $Q_s = 0$ in the relation (30.29) indicates that surface salinity is affected only through the $s_1\,\partial_t\eta$ term. Alternatively, but perhaps less fundamentally, one sees that $s_1\,\partial_t\eta$ is the only relevant term when one sets the salinity of the fresh water to zero, $s_w = 0$, and drops the turbulent flux term $Q_s^{turb} = 0$.

For purposes of numerical accuracy, MOM has traditionally time stepped salinity deviation $s^{mom} = (s - 35)/1000$ rather than the non-negative salt concentration s. s^{mom} is useful for models where the salinity does not deviate much from 35 psu. Turbulent fluxes of s^{mom}, as they depend on spatial gradients, are scaled by 1000 from fluxes of salinity s. For a Boussinesq fluid in the absence of fresh water fluxes, advective fluxes scale likewise. Hence, s^{mom} can be used for a rigid lid or free surface model, so long as there are no fresh water fluxes.

In the presence of fresh water fluxes, however, salinity can deviate significantly from 35 psu, especially when using high resolution ocean models with a realistic hydrological cycle. Additionally, in order to represent the proper effect on salinity from the input of fresh water, it is more straightforward to employ salinity s in the model instead of the deviation s^{mom}. To see why, consider the case of zero interior fluxes for which the continuous time salinity equation in a surface cell takes the form $\partial_t\,s = -(q\,s)/h$. If s is less than 35 psu, then $q\,s^{mom}$ has the opposite sign of $q\,s$. Time stepping $\partial_t\,s^{mom} = -(q\,s^{mom})/h$ would hence lead to the unphysical result of increasing s^{mom} when adding fresh water.

Allowing salinity to deviate significantly from 35psu motivates a more general approach than that discussed by Bryan and Cox (1972) for computing ocean density. Bryan and Cox approximate the full UNESCO equation of state by different cubic polynomials for each model depth level. This approach is warranted for models in which salinity is close to 35psu and for which the model's vertical cell thickness is independent of horizontal position. The latter assumption is broken by the Pacanowski and Gnanadesikan (1998) partial bottom cells, and the former is broken by realistic fresh water fluxes. Hence, for the purpose of computing the model's hydrostatic pressure, we employ the full UNESCO equation of state as formulated by Jackett and McDougall (1995). In this approach, *in situ* density is computed as

$$\rho(\tau) = \rho[\theta(\tau), s(\tau), p(\tau - \Delta\tau)], \tag{30.31}$$

where θ is the model's potential temperature, s is salinity, and pressure is lagged one time step instead of performing the iterative approach described by Dewar *et al.* (1998). The added cost of this approach is modest ($5 - 10\%$ depending on the model configuration).

This is an opportune point to recall the salt budget employed in rigid lid streamfunction models. In that case, one assumes that $\partial_t \eta = \nabla_h \cdot \mathbf{U} = 0$, which means that the only way to affect salinity when there is a surface water flux is to do so indirectly through the addition of a non-zero salt flux

$$Q_s = s_o \, q_w, \tag{30.32}$$

where s_o is a global constant representing the mean ocean salinity, which is typically taken as 35 psu. The use of a globally constant normalization s_o is required in order to conserve both salt and fresh water. Referring to the more realistic zero salt flux given by equation (30.30) for the free surface, the rigid lid salt flux, often termed a *virtual salt flux*, can be considered a linearization of the turbulent salt flux Q_{turb} about the mean salinity state of the ocean. Barnier (1998) provides a detailed and pedagogical discussion of virtual salt fluxes.

Within the context of a constant volume model, Huang (1993) provides a discussion of how steadily forced ocean models can behave when forced with fresh water fluxes. We are also intrigued by the possible differences in response under transient forcing, such as occurs in realistic coupled climate models where potentially large local departures of salinity from the global mean s_o are associated with river runoff, evaporation, precipitation, and ice melting/freezing.

30.9 Discrete conservation properties

As mentioned previously, it is essential that the free surface method provide for a physically relevant set of discrete conservation laws. This section discusses these issues.

30.9.1 Volume conservation

To conserve model volume, it is sufficient to ensure that the surface height η evolves in a conservative manner. Both explicit time stepping discretizations (30.14) and (30.18) trivially satisfy such conservation. All model tests indicate that volume is conserved to within computer roundoff.

30.9.2 Energetic consistency

Energetic consistency of the methods used here have been shown elsewhere. First, the arguments given by Bryan (1969) account for the momentum advection terms, which are discretized using second order centered differences. The result is a redistribution of local kinetic energy, yet a preservation of its global integral. Second, the arguments in the appendix of Pacanowski and Gnanadesikan (1998) show that the partial cell methods ensure that the change in energy due to horizontal pressure forces balances potential energy change when density is linearly dependent on temperature and salinity. Importantly, this consistency is realized only when incorporating the undulating surface grid cell thickness into *both* the tracer and baroclinic velocity equations, as done here. Third, Appendix B of Dukowicz and Smith (1994) provide a complementary analysis of the energetic consistency within their implicit free surface method, much of which is relevant for the present considerations.

30.9.3 Tracer quasi-conservation

An analysis of tracer conservation for the present method has not been given elsewhere, and so warrants discussion. For this purpose, multiply the tracer budget (30.28) by the area of a tracer cell A_{ij} and sum over all cells. All flux terms reduce to boundary fluxes, which are assumed for the following purposes to vanish. The result is the balance

$$\sum_{ijk} A_{ij} \left(h_k^t(\tau)\, \partial_t T_k(\tau) + \delta_{k1}\, T_k(\tau)\, \partial_t \eta(\tau) \right) = 0. \tag{30.33}$$

A simplified form of this budget was discussed in the Introduction. Using a leap-frog time stepping scheme for both the tracer concentration and surface height leads to the discrete time budget

$$\sum_{ijk} A_{ij}\, h_k^t(\tau) \left((T_k(\tau + \Delta\tau) - T_k^R(\tau - \Delta\tau) \right) = -\sum_{ij} A_{ij}\, T_1(\tau) \left(\eta(\tau + \Delta\tau) - \overline{\eta}^*(\tau - \Delta\tau) \right). \tag{30.34}$$

Note that it was necessary to assume that the tracer and baroclinic time steps are equal in order to reach this result. In this expression, T_k^R represents the Robert time filtered tracer concentration, and $\overline{\eta}^*$ is the time averaged surface height. The presence of the Robert time filtering and the time averaging makes this discrete balance only a quasi-conservation law. That is, in the absence of boundary fluxes, the total tracer

$$\sum_{ijk} A_{ij} \left(h_k(\tau)\, T_k(\tau + \Delta\tau) + \delta_{k1}\, T_k(\tau)\, h_k(\tau + \Delta\tau) \right) \tag{30.35}$$

is not exactly constant in time. Instead, time truncation errors allow for only partial conservation. Note that even if one were to employ a time stepping scheme for the surface height which did not use time averaging, or if one time stepped thickness weighted tracer instead of tracer concentration, the use of a Robert filter to suppress the leap-frog splitting for the tracer concentration precludes exact tracer conservation. Alternative approaches, such as Hallberg (1997), can be constructed which use only two-time levels and which do not admit time splitting. In this case, an exact discrete form of tracer conservation can be given. Given these caveats, it has nonetheless been found that for all tests conducted, including realistic simulations with mesoscale eddies, the total tracer given by equation (30.35) remains steady with the present free surface scheme. That is, the tests show near exact conservation with no noticeable trend.

Even though we have found the model to be numerically stable when stretching tracer time steps longer than baroclinic steps, as commonly done in rigid lid models, doing so precludes tracer conservation within top model grid cells since their volumes temporally vary. This is not a constraint shared by rigid lid models, and so it warrants some explanation.

Technically, the constraint of equal baroclinic and tracer time steps is seen by noting that without equating them, it is not possible to identify a physically relevant quasi-conserved property such as the total tracer given by equation (30.35). Fundamentally, the constraint arises from the coupling between tracer and volume conservation. For example, with a rigid lid, the volume of each cell is temporally constant, and so there is no constraint on tracer time steps so far as tracer or volume conservation are concerned. In contrast, when the volume of a cell is allowed to dynamically change, this change must occur in concert with the change in tracer concentration in order to conserve total tracer. It is interesting to note that attempts at time stepping the thickness weighted tracer equation (30.25), which yields the more conventional quasi-conserved discrete quantity $h(\tau)\, T_k(\tau)$, results in numerical instabilities when stretching the tracer time step longer than the baroclinic time step. This approach is therefore severely limited in its utility for coarse resolution climate studies.

Although it is important to conduct realistic tests of this conservation property, and such tests have been done, it is useful to mention here some idealized cases, each of which start with uniform temperature and salinity fields. Blowing winds causes the free surface to undulate and hence to redistribute volume and generate barotropic currents. However, pure wind forcing will not cause the temperature or salinity within a grid cell to change, and it will not cause the total model heat, total salt, or total volume to change. In effect, this experiment checks the consistency of the discretized tracer and volume budgets. If instead of blowing winds, there is fresh water added with temperature equal to that in the ocean, but with zero salinity, then the temperature in each cell will stay constant, the heat increase, the volume increase, the salinity decrease, and the total model salt remain constant.

Tracer conservation issues are relevant for spinning up coarse resolution z-coordinate climate models to a thermodynamic equilibrium. The rigid lid procedure, as discussed by Bryan (1984), Killworth *et al.* (1984), and Danagasoglu *et al.* (1996), typically employs a stretched tracer time step. Doing so with the current free surface will not allow for tracer conservation in the surface grid cells. However, preliminary idealized restoring boundary condition tests indicate that this subtlety does not alter the equilibrium solution as compared to that found with the rigid lid method. Further tests will be necessary to determine whether this insensitivity is robust when employing more realistic boundary conditions, including fresh water.

30.10 Detailed treatment of surface tracer budgets

The purpose of this section is to revisit, in detail, the issues of surface tracer budgets, and in particular to describe the various decisions that have been made in the model implementation. One of the main goals is to provide a conservative numerical scheme which circumvents the peculiarity of advection and diffusion operators acting on different time levels. There is some overlap with the discussion in Section 30.6, although, again, the present section is closely tied to specific choices made in the numerical code.

30.10.1 Summary of the surface tracer fluxes

Recall the continuous time tracer budget

$$h^t \, \partial_t \, T = -\partial_x \, F^x - (\delta_{k1} \, T_k \, \partial_t \eta + F^z_{k-1} - F^z_k). \tag{30.36}$$

For all levels except the sea surface where $k = 1$, the tracer fluxes can be calculated from the model variables. At the surface, the vertical flux $F^z(z = \eta)$, or the discrete counterpart $F^z_{k=0}$, must be specified in order to solve the vertical advection-diffusion equation. Conservation of tracer indicates that the surface tracer flux entering the ocean must equal that flux leaving other component models, such as the atmosphere, river, ice, etc. For substances such as salt, gases, or trace metals, the total tracer mass is conserved. For heat flux, the latent heat from condensation and evaporation provides a heat source at the sea surface which can be included in the heat, or enthalpy, budget.

30.10.1.1 Flux between the ocean model and other model components

Letting Q_T symbolize the flux leaving the other component models and entering the ocean, tracer conservation leads to the surface boundary condition

$$F^z_{k=0} = Q_T. \tag{30.37}$$

This equation acts to close the surface tracer budget associated with vertical transfer, where the flux Q_T is prescribed from data, determined via a restoring procedure, or calculated from a model which resolves details of the planetary boundary layer. In general, Q_T has a contribution from parameterized turbulent flux as well as an advective tracer flux coming in with fresh water,

$$Q_T = -(Q^{turb}_T + q_w \, T_w). \tag{30.38}$$

In this expression, T_w is the tracer concentration in the fresh water, and q_w is the fresh water volume per time per area (units of velocity) entering or leaving the ocean. Both q_w and the turbulent tracer flux Q^{turb}_T are counted as positive if tracer enters the surface cell, and the sign convention corresponds to that employed in the numerical model.

For the case of salt, if one neglects sea spray and brine formation in sea ice, the surface salt flux, due to the large hydration energy, is zero,

$$Q_s = 0. \tag{30.39}$$

Notably, knowledge of the detailed dynamics of wave breaking and wind stirring, which generally are not resolved in an ocean climate model, is not needed for a correct salt budget. For heat and fresh water fluxes, the specification of the surface flux Q_T appears to be more complex than salt. Firstly, the surface fluxes are sensitive to the sea surface temperature (SST). Secondly, due to the skin effect and the complex surface radiation balance, the SST may differ substantially from the average temperature of the surface model cell. The following discussion assumes that this very difficult problem in boundary layer physics is solved, or approximately solved, so that Q_T is considered a known quantity.

30.10.1.2 Surface flux in the ocean model

The vertical flux $F^z_{k=0}$ involves vertical tracer advection by fresh water, as well as the usual parameterized turbulent flux

$$F^z_{k=0} = F^{z,turb}_{k=0} + F^{z,adv}_{k=0}. \tag{30.40}$$

Both components, $F^{z,turb}_{k=0}$ and $F^{z,adv}_{k=0}$, are governed by special processes in the ocean-atmosphere boundary layer which are generally not resolved by the ocean model. In particular, the advective flux

$$F^{z,adv}_{k=0} = -q_w \, T_{k=0} \tag{30.41}$$

represents that flux of tracer advected by fresh water relative to the undulating sea surface. To compute this flux, the surface tracer concentration $T_{k=0}$ is required. This concentration depends on small scale processes in the ocean-atmosphere boundary layer, and it is generally different from the surface cell tracer concentration $T_{k=1}$. For example, with a large fresh water flux, surface salinity may be substantially reduced in a very thin surface layer. Additionally, due to the complex surface radiation balance, such as implied by the skin effect, the "true" SST may differ substantially from the temperature of the surface model cell. In general, however these flux components are discretized, they must maintain conservation of tracer.

The identification (30.37) indicates that

$$F^{z,turb}_{k=0} + F^{z,adv}_{k=0} = -(Q^{turb}_T + q_w \, T_w). \tag{30.42}$$

Notably, it is not generally correct to separately identify the two turbulent terms and two advective terms. For example, a zero surface salt flux, $Q_s = 0$, yields the balance at the ocean surface between advection and turbulence

$$F^{z,turb}_{k=0} = -F^{z,adv}_{k=0}, \tag{30.43}$$

where each term is generally non-zero (see discussion surrounding equation (30.30)). However, the concentration of salt is well approximated by zero for rivers, rain, and evaporation, and so $Q_s = 0$ and $s_w = 0$ implies

$$q_w \, s_w = Q^{turb} = 0. \tag{30.44}$$

Hence, attempts to generally identify $F^{z,turb}_{k=0}$ with $-Q^{turb}_T$ and $F^{z,adv}_{k=0}$ with $-q_w \, T_w$ can yield incorrect results.

30.10.2 Advection and diffusion on different time slices

MOM updates tracer and baroclinic velocities with a leapfrog time step. Likewise, as discussed in Section 30.3, the sea surface elevation is also updated with a leapfrog time step, and it requires the fresh water flux $q_w(\tau)$ for calculation of sea level $\eta(\tau + 1)$. Consequently, both the advective and turbulent pieces of Q_T (equation (30.38)) must come from time level τ.

A complication arises when considering the diffusive piece of the surface tracer flux. For numerical stability reasons, explicitly computed diffusive fluxes are taken at time level $\tau - 1$ instead of τ. Additionally, for the vertical diffusive fluxes, it is more common to employ an implicit time stepping scheme in order to allow realistically large vertical fluxes active in boundary layers. In this case, the tracer flux is determined at time step $\tau + 1$. For the surface fluxes, this situation generally leads to a separate temporal specification of the advective and the diffusive tracer flux components in equation (30.40), $F^{z,turb}_{k=0}$ and $F^{z,adv}_{k=0}$. Thus, fresh water and tracer flux may become temporally inconsistent, which compromises tracer conservation in the ocean model. The following considerations aim to eliminate this problem.

For this purpose, return to equation(30.36). The total surface contribution to the tracer time tendency is given by

$$\begin{aligned} F^{z*}_{k=0} &= F^{z,turb}_{k=0} + F^{z,adv}_{k=0} + T_1 \, \partial_t \eta \\ &= F^{z,turb}_{k=0} + F^{z,adv*}_{k=0}, \end{aligned} \tag{30.45}$$

where

$$F^{z,adv*}_{k=0} = F^{z,adv}_{k=0} + T_1\,\partial_t\eta \tag{30.46}$$

represents a generalized non-turbulent, or "advective", contribution to the evolution of the surface tracer concentration. As previously discussed, the turbulent and generalized advective terms are handled quite differently in the model, with the turbulent term handled via vertical diffusion, and the advective term via a surface advective flux. Use of the identification $F^z_{k=0} = Q_T$ (equation (30.37)) allows for these two terms to be written in the alternative form, each taken at time step τ,

$$\begin{aligned} F^{z,turb}_{k=0}(\tau) &= F^z_{k=0} - F^{z,adv}_{k=0} \\ &= Q_T(\tau) + q_w(\tau)\,T_{k=0}(\tau) \\ &= -Q^{turb}_T(\tau) + q_w(\tau)\,\left(T_{k=0}(\tau) - T_w(\tau)\right), \end{aligned} \tag{30.47}$$

$$\begin{aligned} F^{z,adv*}_{k=0}(\tau) &= -q_w(\tau)\,T_{k=0}(\tau) + T_1(\tau)\,\partial_t\eta(\tau) \\ &= -T_1(\tau)\nabla_h\cdot\mathbf{U}(\tau) + q_w(\tau)\,\left(T_1(\tau) - T_{k=0}(\tau)\right), \end{aligned} \tag{30.48}$$

where the free surface height equation $\partial_t\eta(\tau) = -\nabla\cdot\mathbf{U}(\tau) + q_w(\tau)$ was used to reach the last equality.

As mentioned earlier, the surface tracer concentration $T_{k=0}$ is not a model variable, and so an approximation is needed to specify the fluxes (30.47) and (30.48). So long as this approximation is made consistently in both $F^{z,turb}_{k=0}$ and $F^{z,adv*}_{k=0}$, the total tracer flux $F^z_{k=0}$ is unchanged, and the details of the approximation itself should be unimportant. In many cases it should be reasonable to set

$$T_{k=0}(\tau) \approx T_1(\tau). \tag{30.49}$$

This approximation results in the elimination of the fresh water term in the generalized surface advective flux (30.48), which is very convenient for many purposes. Other, more detailed, approximations may be considered, such as those which somehow account for the structure of a surface layer. However, the approximations outlined here should be sufficient for most regional or global climate modeling purposes. Approximations for the tracer concentration in the fresh water, T_w, are considered in Sections 30.10.5 and 30.11.

30.10.3 Multiple sources and sinks of fresh water

The boundary conditions (30.47) and (30.48) are valid under the special assumption that there is only one fresh water source. However, there may be a fresh water flux between ocean and atmosphere, river discharge, and sea ice formation all within the same model grid cell. Consequently, it is important to consider generalizations of the surface fluxes (30.47) and (30.48) to the case of multiple sources

$$\begin{aligned} F^{z,turb}_{k=0}(\tau) &= Q_T(\tau) + q_w(\tau)\,T_{k=0}(\tau) \\ &= -Q^{turb}_T(\tau) + q_w(\tau)\,T_{k=0}(\tau) - \sum_{l=a,r,i} q_l(\tau)T_l(\tau), \end{aligned} \tag{30.50}$$

$$\begin{aligned} F^{z,adv*}_{k=0}(\tau) &= -q_w(\tau)\,T_{k=0}(\tau) + T_1(\tau)\,\partial_t\eta(\tau) \\ &= -T_1(\tau)\nabla_h\cdot\mathbf{U}(\tau) + q_w(\tau)\,\left(T_1(\tau) - T_{k=0}(\tau)\right), \end{aligned} \tag{30.51}$$

where q_l is the fresh water flux contribution from the atmosphere (a), rivers (r) and sea ice (i), T_l is the corresponding tracer concentration, and q_w is the total fresh water flux,

$$q_w = \sum_{l=a,r,i} q_l. \tag{30.52}$$

Likewise, the turbulent flux Q_T^{turb} may consist of several contributions.

30.10.4 The special case of salt

As mentioned earlier, salt is a special tracer in so far that the large hydration energy precludes a nontrivial salt flux through the sea surface,

$$Q_s = 0. \tag{30.53}$$

Additionally, the salinity of rivers, rain, and evaporation is also approximately zero,

$$s_r = s_a = 0. \tag{30.54}$$

The turbulent and advective surface salt fluxes are thence given by

$$F_{k=0}^{z,turb}(\tau) = q_w(\tau)\, s_{k=0}(\tau) \tag{30.55}$$

$$F_{k=0}^{z,adv*}(\tau) = -s_1(\tau)\nabla_h \cdot \mathbf{U}(\tau) + q_w(\tau)\left(s_1(\tau) - s_{k=0}(\tau)\right). \tag{30.56}$$

The approximation $s_{k=0}(\tau) \approx s_{k=1}(\tau)$ further simplifies these fluxes. Note that in the case of sea ice, salt fluxes through the sea surface are often important to consider. Thus, the surface boundary condition of no salt flux should not be "hard wired" in the model.

Verifying the conservation of salt has proven to be an important test case for the accuracy of the numerical scheme. Any new implementation of component models, or changes in the form of the surface fluxes, should be tested to see that they do not adversely affect salt conservation.

30.10.5 Neutral tracer fluxes

In many cases, the tracer concentration in the incoming freshwater is unknown. For example, the river temperature is typically not included in river discharge datasets, and atmospheric component models typically do not carry the temperature of precipitation as a variable separate from the air temperature. Hence, approximations are needed to specify the surface boundary conditions (30.47) and (30.48).

If the tracer concentration in the fresh water is the same as in the surface box, the fresh water flux does not change the surface tracer concentration,

$$T_l = T_1(\tau). \tag{30.57}$$

This type of tracer flux is termed "neutral" in the following. It is clear that salt should never be approximated as a "neutral" tracer since the salinity of fresh water is effectively zero.

For the special case that all fresh water borne tracer flux components, i.e. atmosphere, rivers and sea ice, are "neutral", then the surface boundary conditions (30.47) and (30.48) become

$$F_{k=0}^{z,turb}(\tau) = -Q_T^{turb}(\tau) - q_w(\tau)\left(T_1(\tau) - T_{k=0}(\tau)\right), \tag{30.58}$$

$$F_{k=0}^{z,adv*}(\tau) = -T_1(\tau)\nabla_h \cdot \mathbf{U}(\tau) + q_w(\tau)\left(T_1(\tau) - T_{k=0}(\tau)\right). \tag{30.59}$$

With the further approximation $T_{k=0}(\tau) \approx T_1(\tau)$, the fresh water flux does not appear in the surface flux

$$F_{k=0}^{z,turb}(\tau) \approx -Q_T^{turb}(\tau), \tag{30.60}$$

$$F_{k=0}^{z,adv*}(\tau) \approx -T_1(\tau)\nabla_h \cdot \mathbf{U}(\tau). \tag{30.61}$$

Although absent from the surface tracer flux, fresh water does influence tracers through its effects on the volume of the surface tracer cell. In general, such neutral tracers have proven to be important for tests of the numerical schemes, and they provide a suitable approximation for many modeling purposes.

The more general case when only one or two components are "neutral" is straightforward. For example, consider a neutral atmosphere, non-neutral rivers, and no sea ice. In this case, the turbulent surface flux component is given by

$$F_{k=0}^{z,turb}(\tau) = -Q_T^{turb}(\tau) - q_a(\tau)T_1(\tau) - q_r(\tau)T_r(\tau) + q_w(\tau)T_{k=0}(\tau). \tag{30.62}$$

Approximating $T_{k=0}(\tau) \approx T_1(\tau)$ indicates that the atmosphere fresh water flux does not contribute to the surface tracer flux

$$F_{k=0}^{z,turb}(\tau) = -Q_T^{turb}(\tau) - q_r(\tau)\left(T_r(\tau) - T_1(\tau)\right). \tag{30.63}$$

(Note that the latent heat for temperature needs special consideration, and should be included in $Q_T^{turb}(\tau)$.) For neutral river but non-neutral atmosphere tracers, one has

$$F_{k=0}^{z,turb}(\tau) = -Q_T^{turb}(\tau) - q_a(\tau)\left(T_a(\tau) - T_1(\tau)\right). \tag{30.64}$$

30.11 Implementation of fresh water fluxes and rivers in MOM

This section documents the means for interfacing MOM to various models which input fresh water into the ocean. In particular, a river interface is described. The discussion here builds much on that given in Section 30.10.

30.11.1 How river fluxes are input to MOM

For the case of rivers, one may conclude that the most realistic implementation would be an open boundary condition in the ocean model with prescribed values for the mass transport, the heat flux, and other tracer fluxes entering or leaving through river mouths. However, such a coupling of rivers with the ocean model is very complex and not necessary for most large-scale modeling purposes. One particular simplification that is made in MOM when coupling to rivers is to assume that it is unimportant whether a momentum or tracer flux enters the surface cell vertically through the sea surface or horizontally as a lateral flux. Consequently, in practice, all fluxes from other component models enter vertically through the surface boundary condition.

30.11.2 Approximations for the surface boundary conditions

To summarize much of the discussion in Section 30.10, note that the following approximations and defaults are used for the implementation of fresh water fluxes in MOM:

- For the update of tracer concentration and sea surface height, fresh water and surface tracer fluxes come from time level τ.

- The surface tracer concentration is approximated by

$$T_{k=0}(\tau) \approx T_1(\tau). \tag{30.65}$$

- MOM provides interfaces for the interaction of the ocean model with other component models, such as an atmospheric model, a sea-ice model, and a model of river discharge. The interface calculates the surface boundary condition for the tracer equations. The most general expression for the advective and diffusive flux through the sea surface for all tracers is

$$\begin{aligned} adv_fb_{i,k=0,j} &= F^{z,adv*}_{k=0} \\ &= -T_1(\tau)\nabla_h \cdot \mathbf{U}(\tau), \end{aligned} \tag{30.66}$$

$$\begin{aligned} diff_fb_{i,k=0,j} &= -F^{z,turb}_{k=0} \\ &= Q^{turb}_T(\tau) - q_w(\tau)\, T_1(\tau) + \sum_{l=a,r,i} q_l(\tau) T_l(\tau). \end{aligned} \tag{30.67}$$

  The sum includes the atmosphere (a), river (r) and sea-ice (i) contribution.

- For salt $Q^{turb}_T = 0$. For ocean-atmosphere and river flux $s_a = s_r = 0$. Sea-ice flux can allow $s_i \neq 0$.

- As the default, the surface fluxes of heat and other tracers are approximated as a "neutral" flux; i.e., it does not change the surface tracer concentration. With the options *river_tracer_explicit*, *atmos_tracer_explicit* and *ice_tracer_explicit* an explicit specifiction of the tracer concentration in the fresh water can be enabled.

- For the surface tracer flux, the following diagnostic is printed to snapshots and time mean files

$$stf = Q^{turb}_T(\tau) + \sum_{l=a,r,i} q_l(\tau) T_l(\tau). \tag{30.68}$$

- Option *simple_sbc* permits river inflow but no sea ice. The ocean - atmosphere tracer fluxes are specified directly in setvbc. The river inflow is specified in subroutine river. Only "neutral" fresh water flux is allowed for *simple_sbc*.

- With option *time_mean_sbc_data* and *coupled*, arbitrary surface fluxes may be specified from an atmosphere, river and ice model. The different fluxes are specified prior the call of the ocean model in the subroutine *get_ocean_sbc*. If no other option is specified, "neutral" fluxes are the default.

30.11.3 New files and changed subroutines

Rivers, and extensive accounting of fresh water fluxes, were not part of the MOM 3.0.0 distribution. Updates to certain files were necessary to incorporate these effects. In particular, files *checks.F*, *datamod.F90*, *derived_options.h*, *driver.F*, *loadmw.F*, *mw.h*, *setocn.F*, *setvbc.F* and *tracer.F* have code which considers fresh water flux. Additionally, *get_ocean_sbc.F* is new (it was not part of MOM 3.0.0), and the files *setffs.F* and *getriver.F*, which were part of MOM 3.0.0, are now obsolete and must be removed or replaced by dummy subroutines. The module *tinterp.F90* replaces a part of *timeinterp.F*.

To model river inflow, a set of subroutines is provided which read river flux data from a database, interpolate to model time, and assign the data to the corresponding surface grid cells of the ocean model. These subroutines could be replaced by a separate hydrological model of river runoff without any changes to the MOM interface.

The following files are provided for modeling river inflow: *check_river.F*, *river.F*, *river.h*, *rivermod.F90*, *setriver.F*. These are described in Section 30.11.7.

30.11.4 Changed and new variables for the surface boundary conditions

sfft The total fresh water flux q_w in $\mathrm{cm\,s^{-1}}$ at tracer points. It is defined on the processor domain.

sfft_expl The fresh water flux contributions from those sources, where the default "neutral" tracer flux is not desired and the corresponding tracer fluxes are calculated explicitly. If, e.g., river tracers are calculated explicitly, $sfft_expl = q_r$. If all tracer fluxes are assumed to be "neutral," then this variable is not defined and not allocated. It is defined on the processor domain.

sffu The total fresh water flux q_w in $\mathrm{cm\,s^{-1}}$ at velocity points. It is defined on the processor domain.

strf The turbulent surface tracer flux, and contributions of tracer flux with fresh water which are not assumed to be "neutral." Instead, they are calculated explicitly. This field is defined on the processor domain.

stf The total surface tracer flux. The field is needed for diagnostic purposes only. It is defined in the memory window.

stf_turb The turbulent part of the surface tracer flux tracer flux as in eq. (30.67). It is defined in the memory window.

30.11.5 Data flow between the model components

Here is an outline of the data flow between the model components.

1. The fresh water fluxes from atmosphere, rivers and sea ice are updated on different time scales. First the atmosphere model (or in some cases a boundary layer model) runs over one time segment and provides averaged surface fluxes between ocean and atmosphere or ocean and sea ice respectively. *gosbc* manages the data flow from the atmosphere model to the variable *sbcocn* which contains the corresponding surface fluxes for the ocean model.

2. The subroutine *river* is called after each atmosphere segment and provides the river fresh water flux as well as river tracer fluxes averaged over one time segment.

3. Then the ocean model is called several times during one time segment. If there is a sea ice model it is called in turns with the ocean model. Hence, the procedure *gosbc* as provided in MOM is not sufficient to manage explicit fresh water flux coupled with other model components. This is done now by subroutine *get_ocean_sbc* which collects all surface boundary conditions for the ocean model prior to when an ocean time step is executed. The subroutine *setffs*, previously called from *driver*, is now part of *get_ocean_sbc*.

4. The fields *sfft*, *sfft_expl* and *strf* collect the fresh water flux and all tracer fluxes from the atmosphere (provided in *sbcocn*), the rivers and from sea ice. These fields are needed in subroutine *setvbc* to calculate *stf* and *stf_turb*.

This complex structure is needed especially for the river inflow. In principle, the output of the river model could be used directly in *setvbc*. However, since *setvbc* works in the memory window, a check for river inflow would be necessary for each grid cell at each time step. The present implementation avoids this at the price of more memory consumption.

30.11.6 New model options

The explicit consideration of fresh water flux in the ocean model requires one to enable the option *explicit_fresh_water_flux*. Also, recall that the option *sailinty_psu* is required in order to handle salinity in the presence of fresh water fluxes (see Section 30.8 for discussion). The following options can be used to specify details of the treatment of fresh water flux in MOM:

river_inflow
Enables the inclusion of river runoff in the surface fresh water flux.

river_tracer_explicit
As default the tracer concentration in the river is assumed to be the same as in the ocean surface grid cell ("neutral" tracers). In this case the river discharge leaves the tracer concentration in the ocean unchanged, except for salinity, in which case, for example, the salinity is reduced upon increasing the volume of the surface cell. For *simple_sbc* this is "hard wired". If *river_tracer_explicit* is enabled, the tracer values in the river are read from a database. This option does not influence salt.

atmos_tracer_explicit
As default the tracer concentration in the atmospheric fresh water flux is assumed to be the same as in the ocean surface grid cell ("neutral" tracers). In this case the fresh water flux leaves the tracer concentration in the ocean unchanged. For *simple_sbc* this is "hard wired". If *atmos_tracer_explicit* is enabled, the tracer values must be provided from atmos. This option does not influence salt.

need_sfft_expl
This option is defined in *derived_option.h* if *atmos_tracer_explicit* or *river_tracer_explicit* is enabled. It is used in *setocn* to detect whether or not *sfft_expl* must be allocated. It must not be specified from a run script.

skip_river_map
The printout of an ASCII map showing the position of the rivers is suppressed.

show_river_details
Enables detailed output to follow the data flow.

diag_river
Enables diagnostics for river inflow. For each river the fresh water flux, the tracer flux and the time integrals of both are written to a separate file.

debug_river
Enables debugging of the river code.

30.11.7 The river code

This section details the river code and provides examples of how to set up river discharge into MOM.

30.11.7.1 Files

All files needed for river inflow can be found in the directory *RIVER*, and they include *river.F*, *river.h*, *setriver.F*, *rivermod.F90*, and *check_river.F*. The file *example_data.dat* provides an example of the data format expected for tracer data from a river. Two kinds of information on a river are needed. The geographic location of rivers is defined in terms of the ocean model grid. The user must edit file *river.h* to provide this information for MOM. The time dependent fresh water flux and the tracer concentration in the river is provided in separate files.

30.11.7.2 Setup of the river geometry

In *river.h* the following parameters must be specified:

nriv	The number of rivers in the model domain,
ntriv	The number of tracers in the fresh water. This parameter is valid only if option *simple_sbc* is disabled. For *simple_sbc* neutral rivers are assumed and no tracer values are needed.
nboxmax	The maximum numbers of surface grid cells used for one river.
rivertrname	A name for each tracer. This parameter is required only if river tracers are not "neutral". The names are used only to trace the data flow to the S.B.C in *get_ocean_bc*.
monthly_data	The river database may contain data with arbitrary time scales. For monthly river data the database may not be suitable for leap years. If monthly_data is .true., the length of the February is adjusted automatically to the calendar defined for the model run. Setting monthly_data to .false. disables this adjustment. Monthly data must be set to .false. in any case when the river data are not monthly data.

The following example configures two rivers, the Amazon and Zaire rivers:

```
      parameter (nriv = 2)
# ifndef simple_sbc
      parameter (ntriv = 2)
# endif
      parameter (nboxmax = 4)
!
      integer nboxriv(nriv)
      integer iriv(nriv,nboxmax), jriv(nriv,nboxmax), riverindex(nriv)
!
      character rivername(nriv) *16
      data (riverindex(nr),nboxriv(nr),
     & (iriv(nr,nb),jriv(nr,nb),nb=1,nboxmax),
     & rivername(nr), nr=1,nriv)
     & / 1, 3, 10,98, 10,99, 11,98, 0,0, 'Amazon'
     &,   2, 2, 89,80, 89,81, 0,0, 0,0, 'Zaire'
     &/
# ifndef simple_sbc
      character rivertrname(ntriv) *12
      data (rivertrname(n),n=1,ntriv) /'temperature','salinity'/
# endif
```

```
logical monthly_data
data monthly_data /.true./
```

The first item in the data statement is a running number used to identify the river. The second item indicates the number of surface grid cells used for the river, three for the Amazon River, two for the Zaire River. The next numbers specify the zonal and meridional grid cell index of these river boxes. The grid cell index can be found from the output of *grids* and *topog*. Here an arbitrary example is used! Also note that upon changing model resolution, this part of the river information generally needs to be altered. Since nboxmax is 4, two additional cells with index (0,0) must be specified. The river name is used for transparent logging of river inflow. monthly_data /.true./ specifies that monthly data are in the database and an adjustment of the February length to the model calendar is desired.

If option *river_inflow* is enabled, *topog* calls subroutine *check_river* which analyzes the setup in *river.h*. For rivers on land points error messages are generated, rivers located in the open ocean are tolerated but a warning is printed.

30.11.7.3 The river - ocean interface

The data flow for rivers is organized in terms of grid cells instead of rivers. This makes parallel processing simple and avoids checks for river inflow at all grid cells for every ocean time step. The setup is done in subroutine *setriver*. In a first step each grid cell is checked for river inflow. If a grid cell belongs to a river, i.e., the grid cell index is found in *river.h*, the counter of river grid cells is incremented and the cell index is stored. If all grid cell are checked each PE knows it's number of grid cells with river inflow, rivbox_in_pe, and the index of these grid cells in the ocean grid.

Now a field of the dimension rivbox_in_pe is allocated. Each field element belongs to one river grid cell on the PE and is a specific data structure which contains all information needed to update the surface fresh water flux and the tracer flux in this grid cell. The complex data type river_type is defined in the module *rivermod.F90* and has the form

```
type, public :: river_type
  integer              :: index    ! identifies river
  character*16         :: name     ! name of the river
  integer              :: ir,jr    ! zonal and meridional grid index
  real                 :: area     ! area of all boxes of the river
  character*32         :: dstamp   ! time stamps marking the end of each record
  real, pointer        :: rff      ! river fresh water flux
  real, pointer        :: rtf(:)   ! river tracer flux(tracer index)
  character*20, pointer :: trname(:)! Name of the tracers ( tracer index)
end type river_type
```

The field river_rec of type river_type and of dimension rivbox_in_pe is the interface between the river model and the ocean model. It provides all information on the river fresh water and tracer flux for later use by subroutine *get_ocean_bc* at time level τ.

30.11.7.4 Time dependent fresh water and tracer data management

The purpose of the subroutine *river* is to provide the data in river_rec used in subroutine *get_ocean_bc*. Although many coupled models calculate river discharge from hydrological

model components, the river data used here must come from a database and *river* has to do only some time interpolation. Before this can be done subroutine *setriver* reads the database and prepares the time interpolation algorithm.

The database must provide a volume flux $R(t)$ as function of time given, e.g., in units $\mathrm{cm}^3\,\mathrm{s}^{-1}$. It is useful to prescribe $R(t)$ instead of q_r since the volume flux as well as the tracer concentration is independent of the model geometry and the data files may be used for models with various configurations. All data of each river are collected in one file and the file name convention *"rivername".dat* is used. Thus, the example configuration would require the files *Amazon.dat* and *Zaire.dat*. The files are organized in data records averaged over a certain period, say one month. Each line contains one record, starting with a time stamp with the end time of the averaging period followed by the averaging period in days, the fresh water flux in $\mathrm{cm}^3\mathrm{s}^{-1}$, the river temperature, salinity and concentration of other tracers. The number of data records and tracers is defined in the file header. The subsequent example file is for a Swedish river and contains 6 records with four tracers.

```
monthly discharge of river Angermansalv (zero salinity)
6
4
yy,mm,dd,hh,mm,ss,per(days),tr(cm**3/s),temp,sal,a,n
1970 2 1 0 0 0 31.0000    .1149E+10  1.1   0.0   0.6   2.4
1970 3 1 0 0 0 28.2425    .1065E+10  0.0   0.0   0.5   2.4
1970 4 1 0 0 0 31.0000    .9798E+09  0.0   0.0   0.6   3.5
1970 5 1 0 0 0 30.0000    .1663E+10  3.0   0.0   1.7   9.5
1970 6 1 0 0 0 31.0000    .3330E+10  3.1   0.0   0.8   4.2
1970 7 1 0 0 0 30.0000    .1785E+10  6.2   0.0   0.4   1.3
```

The volume flux is distributed uniformly over the **total** area A_R of the river which may involve one or more grid cells. In these surface cells the fresh water flux due to the river is,

$$q_R = \frac{R(t)}{A_R}. \tag{30.69}$$

q_R has the dimension of a velocity and has to be added to the surface fresh water flux velocity,

$$q_w \rightarrow q_w + q_R. \tag{30.70}$$

The fluxes of heat and additional tracers entrained with the fresh water are,

$$\begin{aligned} Q_{RT} &= \frac{R(t)}{A_R} T_R \\ &= q_R\, T_R, \end{aligned} \tag{30.71}$$

where T_R is the tracer concentration in the river water. They must be added to the fresh water driven surface fluxes,

$$Q_{wT} \rightarrow Q_{wT} + Q_{RT}. \tag{30.72}$$

The additional momentum entrained with rivers can be estimated from the river volume flux rate, $R(t)$, and the vertical river cross section, C_R. When the cross river circulation in the boundary cells is zero, the average velocity is through the vertical cross section is

$$\mathbf{u}_R = \frac{R(t)}{\rho C_R}\,\hat{\mathbf{C}}_R. \tag{30.73}$$

$\hat{\mathbf{C}}_R$ is the normal unit vector of the river cross section, C_R. The momentum advected through the river cross section with the river flux is

$$\begin{aligned} \mathbf{M}_R &= \rho\, C_R\, |\mathbf{u}_R|\, \mathbf{u}_R \\ &= \rho\, R(t)\, \mathbf{u}_R. \end{aligned} \tag{30.74}$$

The components of the equivalent surface stress, which gives the same momentum input in form of a vertical momentum flux, are

$$\begin{aligned} \tau_R^\lambda &= \rho \frac{R(t)}{A_R} u_R, \\ &= \rho\, q_R\, u_R, \end{aligned} \tag{30.75}$$

$$\tau_R^\phi = \rho\, q_R\, v_R. \tag{30.76}$$

These terms are not implemented yet (as of Feb, 2000) but will be added later.

30.11.7.5 Initializing the river procedures

For each river grid cell a data structure is allocated which is used to organize the time interpolation of river data. The data type is defined in the module *rivermod.F90*

```
type, public              :: river_data_type
   integer                :: index        ! identifies river
   character*16           :: name         ! name of the river
   integer                :: ir,jr        ! zonal and meridional grid index
   real                   :: area         ! area of all boxes of the river
   integer                :: mrecriv      ! number of data records
   logical                :: perriv       ! true for periodic data
   integer                :: ninterp      ! data index for timeinterp
   real, pointer          :: aprec(:)     ! period lengths
   real, pointer          :: tdrec(:)     ! times of data records
   type(time_type)        :: start_time, end_time
   character*32, pointer  :: dstamp(:)    ! time stamp of record end time
   real, pointer          :: rff(:)       ! river fresh water flux (time)
   real, pointer          :: rtf(:,:)     ! river tracer flux (tracer index, time)
   character*20, pointer  :: trname(:)    ! Name of the tracers ( tracer index)
end type river_data_type
```

After determining, grid index, river name and area of all river grid cells, *setriver* prepares the river data for time interpolation. For each river grid cell the corresponding river data file is read. The fresh water flux velocity q_R as well as the tracer fluxes Q_{RT} are calculated for each time record and are stored in the grid cell specific data set.

This procedure looks overly complex but it is of fundamental importance for parallel architectures. On a massive parallel machine all PEs do this initialization procedure separately. Each PE generates a private list of it's river grid cells and knows where to get the required data. Thus, a PE selects and stores the information only for those river grid cells which are located on the PE's domain. If a river is on a domain boundary nothing special has to be done. Both neighboring PEs get the information for their river grid cells independently and no message passing is needed. For later use in the ocean model it is important, that each PE has a

list of the river grid cells where fresh water flux has to be added. Hence, it is not necessary to check at each time step and for all grid cells, whether or not they belong to one of the rivers. However, this is of advantage only, if the

30.11.8 The time interpolation

The fresh water data are available in most cases as monthly data sets. Thus, time interpolation is necessary which is done with subroutine *timeinterp*. This procedure is provided with MOM to interpolate monthly atmosphere data sets and can be used in the same manner for the river data. The fluxes for each river grid cell form one dataset. The fields required for the time interpolation, aprec and tdrec, are part of the fields of type river_data_type.

get_ocean_bc expects fresh water flux at a time which corresponds to time level τ of the circulation model. If *river* is called within the ocean loop, it interpolates the river data to time level *tau*. However, such frequent calls of *river* are not needed in many cases and as the default *river* is called prior an ocean segment. In this case *river* returns data from time

$$model_time + 0.5\,(segtim - dtts). \tag{30.77}$$

The interpolation of river data may not be the only use of *timeinterp*, other boundary data as surface boundary conditions or data for open lateral boundaries may be interpolated with the same subroutine. To avoid conflicts between calls from different model components *timeinterp* has been slightly changed. *timeinterp* uses a parameter n to identify the dataset to be interpolated. Specifying $n = -1$ *timeinterp* assumes that this dataset is used the first time and provides an unique index to identify this dataset for the next interpolation steps. For rivers this identifier, ninterp, is part of the data field river_data. See the initialization of *timeinterp* in *setriver.F* and the comments in *tinterp.F90* for more details.

The amount of river data is small and they can be kept in the memory during the model run. Thus, river_data works like a ram disk and the interpolation procedure can be simplified. The index iprevdriv points to the "left" dataset used for interpolation, inextdriv to the "right" dataset respectively.

30.11.9 Limitations of the river code

The treatment of river inflow with the aforementioned method has limitations which warrant some remarks.

- Data on river runoff are provided mostly as climatological data, i.e. only monthly or seasonal time scales are resolved. However, big rivers may have estuaries with a large storage capacity. Due to wind surge or tides, the river outflow may be temporarily blocked and fresh water enters the ocean intermittently in form of drifting river plumes. To implement such features, the processes producing intermittency in the river outflow must be resolved by the model. Thus, the estuary must be part of the model and the river runoff must enter the model area more upstream.

- The river is assumed to be unstratified and horizontally uniform. Improvements require a higher resolution of the river component. Nonuniform distribution of the fresh water flux over the river grid cells can be added in *setriver.F*

- River inflow is confined to the surface cells. For rivers much deeper than the surface layer, this may introduce too much stratification and an intensified estuarine circulation.

If this is not desired, two or more surface layers should be mixed explicitly at the cells with river inflow. This requires code changes and is not part of the basic MOM code.

- The momentum flux may be incorrect. The method distributes the mass flux over several tracer points and establishes surface pressure gradients which reflect the model setup but not the dynamics in a river mouth. If the dynamical features of the outflow are important, a sufficiently large part of the river must be included in the model.

  In order to prevent unrealistic currents due to different position of velocity and tracer cells in the Arakawa B-grid, the fresh water flux and the tracer flux should be distributed at least over two adjacent tracer cells, the momentum flux should be added in the corresponding velocity cell between the tracer cells. Figures 30.3 and 30.4 give some hints for how the rivers could be configurated. However, the method does not require an adjustment of the coastline to configure rivers so long as land-points or non-advective cells are avoided.

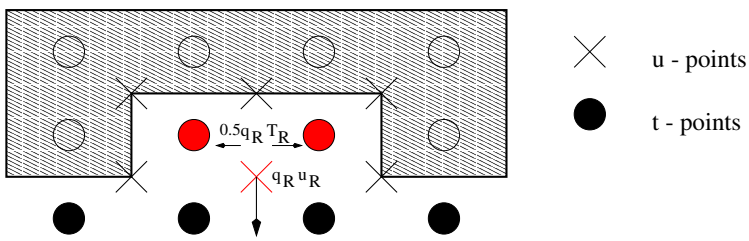


Figure 30.3: Sketch of the cell arrangement for a river flowing southward. The fresh water and tracer flux is distributed over the two red tracer points, the momentum flux is added in the red velocity point.

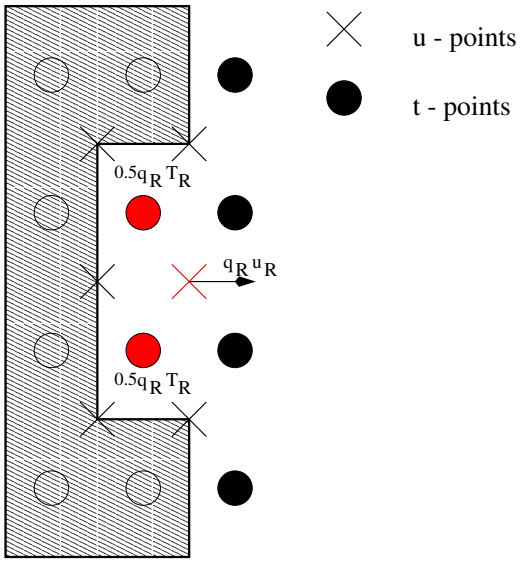


Figure 30.4: Sketch of the cell arrangement for a river flowing eastward. The fresh water and tracer flux is distributed over the two red tracer points, the momentum flux is added in the red velocity point.

30.12 Checkerboard null mode

As discussed by Killworth *et al.* (1991), discretization of gravity waves on a B-grid can admit a stationary grid scale checkerboard pattern (see also Messinger 1973 and Jancic 1974 for atmospheric discussions). This pattern is associated with an unsuppressed grid splitting that can be initiated through grid scale forcing, such as topography. That is, the discrete equations admit a checkerboard null mode. Beckers (1999) also discusses subtleties associated with traditional linear stability. In our experience, when the model goes unstable due to CFL violations, this grid mode is the spatial pattern associated with the instability.

Strictly, the grid mode is present *only* when the surface pressure gradient takes the form $\nabla_h p_s = \overline{\rho} g \nabla_h \eta$, where $\overline{\rho}$ is an averaged surface density. For example, in the implementation of Killworth *et al.* (1991), $\nabla_h p_s = \rho_o g \nabla_h \eta$, and so the null mode is always present. Tests with the alternative expression $p_s = \rho_{k=1} g \eta$ used in MOM, in which surface pressure gradients arise from gradients in both the surface density $\rho_{k=1}$ and surface height, suggest that the null mode is slightly suppressed. A more significant means to suppress the mode, however, is provided through the use of time averaging over the barotropic time steps, as well as with nonlinearity in the shallow water system present when the undulating surface height is fully incorporated to the dynamics.

Hence, for many of our simulations, the grid mode is absent or quite mild. Nonetheless, some experiments did realize the grid mode in the surface height, and so we considered various means of suppressing it. The following relates our experience with this mode and provides suggestions and caveats.

30.12.1 Experiences with the checkerboard null mode

Under certain circumstances, Killworth *et al.* (1991) noted that the checkerboard pattern can appear in the barotropic velocity. In order to dissipate this mode, they suggested that horizontal friction acting on the barotropic velocity be updated each barotropic time step, instead of only during the baroclinic time steps. This approach was found to be especially appropriate in simulations with strong eddy activity.

Updating friction on each time step can be quite costly. Therefore, in practice, one is restricted to the use of a constant viscosity Laplacian friction for both the barotropic and baroclinic modes. Notably, use of a biharmonic viscosity, updated each barotropic time step, could be prohibitively costly. As discussed by Semtner and Minz (1977) and recently by Griffies and Hallberg (1999), Laplacian friction for eddy-permitting simulations significantly under-utilizes the model grid as it strongly suppresses mesoscale eddy formation. We therefore believe that restricting the model to constant viscosity Laplacian friction is not an acceptable solution to the grid splitting problem. Relatedly, barotropically updating a depth dependent viscosity, such as that arising from the Smagorinsky scheme (Smagorinsky 1963, 1993, and Griffies and Hallberg 1999), is very cumbersome due to the need to maintain self-consistency between friction applied in the barotropic and baroclinic equations.

For the time stepping procedure described in this paper, regions where the barotropic fields are evolving experience an effective spatial averaging associated with the barotropic time averaging. Eddy-rich regions are therefore preferentially smoothed, in contrast to the algorithm of Killworth *et al.* in which there is no time averaging. Consequently, we conjecture that time averaging provides for an effective suppression of the grid splitting in both the surface height and vertically integrated velocity. Indeed, we have rarely observed noise in the vertically integrated velocity with the time filtering scheme, and so have not been motivated to update the

friction on each barotropic time step.

Since the equations for the surface height are inviscid, the checkerboard pattern tends to appear most prominently in this field. To suppress the noise, Killworth *et al* proposed the addition of the so-called "del-plus / del-cross" operator to the prognostic equation for the free surface height. Their approach is an extension of that suggested by Jancic (1974). The operator is basically the difference between Laplacians computed in two different manners, and it preferentially removes the checkerboard pattern by coupling the grids. This coupling occurs in a scale-selective manner so that the larger scales are only modestly affected. The construction of this operator requires the definition of time dependent image points over land to allow for it to conserve volume.

30.12.2 A caveat concerning filtering the surface height

In quiescent regions, such as embayments, enclosed seas, and idealized steady state models with nontrivial topography and geography, we have found the checkerboard pattern sometimes is visible in the free surface height. As such, various filters were tested, such as the del-plus / del-cross operator, as well as a two-dimensional Shapiro filter designed to conserve volume with arbitrary geography. Either spatial filter successfully removed noisy patterns from the surface height. However, we found their effects on other model fields, such as vertical velocity, to be quite dependent on the strength of the filter. In particular, for many cases where the filter was sufficient to suppress noise in the surface height, it had detrimental effects on the vertical velocity.

The reason for this sensitivity can be seen by considering the equations for gravity waves on a B-grid using a forward-backward time stepping scheme (note that the following considerations do not depend on the choice of time stepping schemes). To better expose our point, assume $p_s = \rho_o g \eta$, $D = H + \eta \approx H$, and omit fresh water forcing. The discretized equations take the form

$$\eta(t_{n+1}) = \eta(t_n) - \Delta t \, [\nabla_h \cdot \mathbf{U}(t_n) - F(\eta)] \tag{30.78}$$

$$U(t_{n+1}) = U(t_n) - H g \Delta t \, \delta_x \overline{\eta(t_{n+1})}^y \tag{30.79}$$

$$V(t_{n+1}) = V(t_n) - H g \Delta t \, \delta_y \overline{\eta(t_{n+1})}^x , \tag{30.80}$$

where $F(\eta)$ represents a spatial filter applied to the surface height, and the temporal notation has been streamlined from the more complete notation used in Section 30.2.

Consider first a case in which the free surface contains power in the checkerboard pattern, as well as longer waves, yet there is no spatial filter and so $F = 0$. The discretized surface pressure gradients do not feel the checkerboard pattern since this pattern is a null mode of the discrete gradient operators $\delta_x \overline{()}^y$ and $\delta_y \overline{()}^x$. Therefore, discrete pressure gradients are determined only by the longer waves, and so the gradients are smooth. Since the checkerboard pattern is stationary in time, it does not contribute to the vertical velocity. Neglecting the time varying top cell volume in the baroclinic and tracer equations, as in the free surface methods of Killworth *et al.* (1991) and Dukowicz and Smith (1994), renders the checkerboard pattern dynamically irrelevant. Instead, it is simply an esthetic nuisance. Yet when solving the more general equations with undulating top cell thicknesses, it cannot be ignored as it potentially will influence the other model fields.

Now add the spatial filter which is designed to preferentially remove the checkerboard grid noise. Unlike the unfiltered η, the filter F will generally not be invisible to the discretized pressure gradients, and so these gradients are affected by an unphysical source. In practice,

we have found that these sources can be nontrivial, especially when using the Shapiro filter, which is the strongest available grid scale filtering. For the del plus / del minus filter, using the weighting factor suggested by Killworth *et al.* (1991), we found a large amount of noise was transferred to the vertical velocity field. Again, the reason is that this filter is not invisible to the discretized gradients, and so while it acts to smooth the surface height, it correspondingly adds noise to the surface pressure gradients.

One attempt to suppress the transfer of noise was to also filter $\nabla_h \cdot \mathbf{U}$ in order to counteract the noisy pressure gradients. Unfortunately, this approach is problematic, since doing so breaks the precise connection between the barotropic and baroclinic mode systems. As a result, spatially filtering $\nabla_h \cdot \mathbf{U}$ produces a nontrivial spurious vertical velocity throughout the ocean, and in particular it creates unacceptable advective fluxes through the ocean bottom.

30.12.3 Suggestions

As mentioned at the beginning of this section, we find the need for filtering to be greatly reduced due to the use of a nonlinear free surface in which the barotropic time steps are time averaged. Nevertheless, for those cases in which filtering is needed, we have found it sufficient to add a straightforward Laplacian to the free surface height equation with a reasonably small viscosity. The more sophisticated Shapiro filter or del-plus minus del-minus operator were no better, and often much worse in their affects on vertical velocity. Note that a Laplacian, using no-flux boundary conditions, conserves volume. Therefore, adding a Laplacian to the free surface equation is trivial to implement relative to the del plus / del minus operator, which requires the specification of time dependent image points in order to preserve volume in arbitrary geometry.

30.13 Polar filtering

The CFL time step requirements for the barotropic mode can be quite stringent when reaching the pole on a spherical grid. Instead of reducing the time steps, it is common to prescribe a spatial filtering in order to remove the small scale features. This filtering traditionally takes the form of Fourier filtering (e.g., Bryan *et al.* 1975). We have instead found one-dimensional Shapiro filtering preferable due to its increased speed. Nevertheless, as such filtering is necessary for each of the small barotropic mode time steps, the computational cost can be high and it can render strong load imbalances between adjacent parallel computer processors. Indeed, the cost can be higher than the cost of simply reducing the barotropic time step in order to satisfy the CFL constraint. Additionally, for reasons related to the mechanism discussed above, tests have shown that polar filtering of the surface height can result in a nontrivial amount of grid noise in the vertical velocity field within the polar filtering regions. As a result, we do not recommend polar filtering the barotropic fields. The more general solution of pole displacement using non-spherical coordinates (e.g., Smith *et al,* 1995, Murray 1996, Madec *et al.* 1998) is currently under investigation.

Chapter 31

Options for solving elliptic equations

The external mode elliptic equation is solved by conjugate gradients with two possible forms for the numerics of the coefficient matrix. One form of the coefficient matrix must be chosen. In previous versions of MOM, a solution by sucessive over relaxation was available but this has been eliminated in favor of the conjugate gradient technique.

The accuracy of the external mode solution for the stream function $\Delta\psi = \psi^{\tau+1} - \psi^{\tau-1}$ is determined by setting a tolerence (see Section 14.4.5). The preferred method of solution is by conjugate gradients which is an iterative technique. The iteration is ended and the equation is considered solved when the estimated sum of future corrections to $\Delta\psi$ (the truncated ones) is less than the specified tolerence. The estimated sum is approximated assuming a geometric decrease in the maximum correction to $\Delta\psi$ with time.

Typically, as a rule of thumb, the number of scans taken to solve the external mode equation should not be $<< max\ (imt,\ jmt)$ when the forcing (i.e. windstress) is time dependent. If the number of scans is much less than this, it may indicate that the specified tolerence is set too large. As an example, if the specified tolerence is 10^8, then in 10,000 time steps the computed solution is guaranteed to be within 1 Sverdrup ($1x10^{12}$) of the true solution at every point. This is a worst case assuming systematic errors always of the same sign. If time steps are very small, then $\Delta\psi$ will be small and the tolerence should be decreased particularly if long running experiments are involved. The amount of error that is tolerable is dependent on the goal of the experiment.

In cases where the windstress is set constant or zero, the number of scans may eventually get very small as equilibrium is approached. The reason is that a guess at the solution is made from the previous solutions. If the solution is not evolving, then eventually, the guess is within the specified tolerence and zero scans result. It is left up to the researcher to determine the appropriate value of the tolerence. The number of times that the scans are $< max\ (imt,\ jmt)$ are counted and a message is printed at the end of the execution.

31.1 conjugate gradient

This section was contributed by Charles Goldberg. The conjugate gradient technique is used to invert the elliptic equation for external mode options other than *explicit_free_surface* using either five point or nine point operators, selected by options *sf_9_point* or *sf_5_point*. The algorithm below is a Fortran 90 version of the algorithm in Smith, Dukowicz, and Malone (1992).

```
subroutine congrad (A, guess, forc, dpsi, iterations, epsilon)
```

```
use matrix_module

intent (in)     ::  A, guess, forc, epsilon
intent (out)    ::  dpsi, iterations

type(dpsi_type) ::  guess, dpsi, Zres, s
type(res_type)  ::  res, As, forc
type(operator)  ::  A
type(inv_op)    ::  Z
dimension (0:max_iterations) ::  dpsi, res, s, As, beta, alpha

dpsi(0) = guess
res(0)  = forc - A * dpsi(0)
beta(0) = 1
s(0)    = zerovector()
do k = 1 to max_iterations
       Zres(k-1) = Z * res(k-1)
       beta(k)   = res(k-1) * Zres(k-1)
       s(k)      = Zres(k-1) + (beta(k)/beta(k-1)) * s(k-1)
       As(k)     = A * s(k)
       alpha(k)  = beta(k) / (s(k) * As(k))
       dpsi(k)   = dpsi(k-1) + alpha(k) * s(k)
       res(k)    = res(k-1) - alpha(k) * As(k)
       estimated_error = err_estimate(k, alpha(k), s(k))
       if (estimated_error) < epsilon) exit
end do
if (k > max_iterations) then
       print *, 'did not converge in ',k,' iterations'
       stop 'congrad'
end if

iterations = k
dpsi = dpsi(k)

end
```

In MOM, all land masses are usually treated as islands when using this option with option *stream_function* because it has been found that this treatment speeds convergence. Island sums are turned off when solving for surface pressure by setting *nisle*=0. The test for convergence has also been changed from that used in MOM 1 and previous versions. In MOM, the sum of all estimated future corrections to the prognostic variable is calculated assuming geometric decrease of the maximum correction, and the iteration is terminated when this sum is within the requested tolerance. Tests indicate that this solver converges significantly faster than the one in MOM 1[1].

[1]Because of the difference in meaning between the MOM 1 tolerance *crit* and the MOM tolerance *tolrsf*, care must be taken to assure that both are converging to the same tolerance when doing these tests.

31.2 sf_9_point

This option uses the full nine point numerics for the coefficient matrix in Equation (29.18) when inverting the external mode elliptic Equation (29.14). This matrix is slightly different than the one used in MOM 1 and earlier versions. Particular attention has been given to assuring that the operator remains symmetric with respect to islands. This form exactly eliminates[2] the surface pressure term from the momentum equations when used with option *stream_function*. It has a checkerboard null space which in most cases is not a problem because of the Dirichlet boundary conditions.

Options *rigid_lid_surface_pressure* and *implicit_free_surface* require the use of a nine point coefficient matrix[3], and in these cases, option *sf_9_point* must also be enabled. The null space is a problem when option *rigid_lid_surface_pressure* is chosen, and it must be removed. Adding the missing divergence from the *rigid_lid_surface_pressure* to the central term constructs the *implicit_free_surface* which suppresses the null space. Details of these methods appear later.

Constructing the coefficient matrix for Equation (29.14) can be difficult. The mathematician among us (Goldberg) points out that the finite difference operators can be written in terms of matrices as follows:

$$dxt_i \cdot \delta_\lambda(\overline{\alpha_{i,jrow}}^{\phi}) = \sum_{j'=-1}^{0} \sum_{i'=-1}^{0} cddxt_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{31.1}$$

$$dyt_{jrow} \cdot \delta_\phi(\overline{\alpha_{i,jrow}}^{\lambda}) = \sum_{j'=-1}^{0} \sum_{i'=-1}^{0} cddyt_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{31.2}$$

$$dxu_i \cdot \delta_\lambda(\overline{\alpha_{i,jrow}}^{\phi}) = \sum_{j'=0}^{1} \sum_{i'=0}^{1} cddxu_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{31.3}$$

$$dyu_{jrow} \cdot \delta_\phi(\overline{\alpha_{i,jrow}}^{\lambda}) = \sum_{j'=0}^{1} \sum_{i'=0}^{1} cddyu_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{31.4}$$

where $cddxt_{i',j'}$, $cddyt_{i',j'}$, $cddxu_{i',j'}$, and $cddyu_{i',j'}$ are defined as 2x2 matrices:

$$cddxt_{i',j'} = \left(a_{i',j'}\right) = \begin{pmatrix} a_{-1,0} & a_{0,0} \\ a_{-1,-1} & a_{0,-1} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{31.5}$$

$$cddyt_{i',j'} = \left(a_{i',j'}\right) = \begin{pmatrix} a_{-1,0} & a_{0,0} \\ a_{-1,-1} & a_{0,-1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \tag{31.6}$$

$$cddxu_{i',j'} = \left(a_{i',j'}\right) = \begin{pmatrix} a_{0,1} & a_{1,1} \\ a_{0,0} & a_{1,0} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{31.7}$$

$$cddyu_{i',j'} = \left(a_{i',j'}\right) = \begin{pmatrix} a_{0,1} & a_{1,1} \\ a_{0,0} & a_{1,0} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \tag{31.8}$$

[2] To within roundoff.

[3] However, in these cases, the elliptic equations for the prognostic surface pressure p^s are, of course, different than those given above for the prognostic stream function Ψ.

Using Equations (31.1) – (31.4), the first brackets in Equation (29.14) can be rewritten as

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\Bigg[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^{U}_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}$$
$$+\,cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^{U}_{jrow+j''}\cdot 2\Delta\tau}\Bigg]\Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{31.9}$$

and in a similar fashion, the implicit Coriolis contributions from the second brackets in Equation (29.14) can be rewritten as:

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\Bigg[-cddxu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}$$
$$-cddyu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\Bigg]\Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{31.10}$$

which provides a very compact and efficient way of calculating the coefficient matrix $coeff_{i,jrow,i^\star,j^\star}$.

$$coeff_{i,jrow,i^\star,j^\star}=$$
$$\sum_{i'=0}^{1}\sum_{i''=-1}^{0}\sum_{j'=0}^{1}\sum_{j''=-1}^{0}\delta_{i'+i'',i^\star}\delta_{j'+j'',j^\star}\Bigg[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^{U}_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}$$
$$+\,cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^{U}_{jrow+j''}\cdot 2\Delta\tau}$$
$$-cddxu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}$$
$$-cddyu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\Bigg] \tag{31.11}$$

where $\delta_{i'+i'',i^\star}$ is the Kronecker Delta function which is 1 when $i'+i''=i^\star$ and 0 when $i'+i''\neq i^\star$. The entire coefficient array is calculated by six nested loops on i, $jrow$, i', j', i'', and j'', with each iteration adding the terms above to the proper coefficient bucket[4].

The matrix for right hand side of Equation (29.14) can be written as:

$$forc_{i,jrow}=\sum_{i'=0}^{1}\sum_{j'=0}^{1}\Big[-cddyt_{i',j'}\cdot zu_{i+i',jrow+j',1}\cdot dxu_{i+i'}\cdot\cos\phi^{U}_{jrow+j'}$$
$$+cddxt_{i',j'}\cdot zu_{i+i',jrow+j',2}\cdot dyu_{jrow+j'}\Big] \tag{31.12}$$

[4]We have come to call these loops "Amtrack Normal Form," because the indentation of the DO loops and the long executable statement in the middle resemble the "Amtrack" logo. Properly ordered, these nested loops vectorize very well on a Cray YMP.

The coefficient matrix is symmetric[5] except for the implicit Coriolis term (the second bracket in Equation (29.14)). The preferred method of solving Equation (29.14) is by conjugate gradients as described in Dukowicz, Smith and Malone (1993). Refer to Section 31.1 for more detail on the elliptic solvers and to Appendix 29.2.4 for more details on the island equations.

31.3 sf_5_point

This section was contributed by Charles Goldberg. The five point form approximates averages of contributions to the corner points of the nine point operator to produce a five point operator that has coefficients only in the center, north, east, south, and west places in the coefficient matrix for inverting the external mode elliptic equation. This option can be used with option *stream_function* but is not appropriate for options *rigid_lid_surface_pressure* or *implicit_free_surface* because of energy leakage.

In Equation (31.9), corresponding to the first bracket of Equation (29.2), all 32 contributions to the elliptic operator are calculated as in the 9 point operator, except that in the first group, i.e., in the terms originating in a second difference in the δ_ϕ direction, the coefficient of the northeast variable $\Delta\phi_{i,jrow,1,1}$ is averaged with the coefficient of the northwest variable $\Delta\phi_{i,jrow,-1,1}$, and both are applied to the northern variable $\Delta\phi_{i,jrow,0,1}$. This averaging centers the coefficient on the northern edge of the central T cell, so that continuous derivatives in the ϕ direction are well approximated. Similarly, the southeast and southwest contributions from the first group are applied to the southern variable $\Delta\phi_{i,jrow,0,-1}$. The second group of terms originate in a second difference in the δ_λ direction, and here the northeastern and southeastern contributions are averaged and applied to the eastern variable $\Delta\phi_{i,jrow,1,0}$, and the northwestern and southwestern contributions are averaged and applied to the western variable $\Delta\phi_{i,jrow,-1,0}$, giving good approximations to the continuous derivatives in the λ direction. The resulting summations are

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^{U}_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_{i,jrow+j'+j''}$$
$$+\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^{U}_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_{i+i'+i'',jrow} \tag{31.13}$$

Bryan (1969) uses a five point approximation to $(\frac{1}{H}\cdot\nabla\psi_t)$ which as implemented in MOM 1 takes the form

$$\frac{1}{\cos\phi^{T}_{jrow}dxt_i\,dyt_{jrow}}\left[dxt_i\cdot dyt_{jrow}\cdot\delta_\phi\left(\frac{\cos\phi^{U}_{jrow-1}}{\overline{H_{i-1,jrow-1}}^{\lambda}\cdot dyu_{jrow-1}}\,dyu_{jrow-1}\cdot\delta_\phi\Delta\psi_{i,jrow-1}\right)\right.$$
$$\left.+\,dyt_{jrow}\cdot dxt_i\cdot\delta_\lambda\left(\frac{1}{\overline{H_{i-1,jrow-1}}^{\phi}\cdot\cos\phi^{T}_{jrow}\,dxu_{i-1}}\,dxu_{i-1}\cdot\delta_\lambda\Delta\psi_{i-1,jrow}\right)\right] \tag{31.14}$$

[5]Refer to Section 29.2.5.

In this notation, the five point approximation used in MOM takes the form

$$\frac{1}{2\Delta\tau}\cdot \left[dyt_{jrow}\cdot\delta_\phi\left(\overline{\left(\frac{dxu_{i-1}\cdot\cos\phi^U_{jrow-1}}{H_{i-1,jrow-1}\cdot dyu_{jrow-1}}\right)}^{\lambda}\cdot dyu_{jrow-1}\cdot\delta_\phi\Delta\psi_{i,jrow-1}\right)\right.$$
$$\left. + dxt_i\cdot\delta_\lambda\left(\overline{\left(\frac{dyu_{jrow-1}}{H_{i-1,jrow-1}\cdot\cos\phi^U_{jrow-1}\cdot dxu_{i-1}}\right)}^{\phi}\cdot dxu_{i-1}\cdot\delta_\lambda\Delta\psi_{i-1,jrow}\right)\right] \tag{31.15}$$

For comparison with MOM, Bryan's form must be multiplied by $(\cos\phi^T_{jrow}dxt_idyt_{jrow})/(2\Delta\tau)$. The principal differences between these two forms arise from Bryan's use of averages of reciprocals of the form $\frac{2}{H_{i,jrow}+H_{i-1,jrow}}$ where MOM uses $\frac{1}{2}(\frac{1}{H_{i,jrow}}+\frac{1}{H_{i-1,jrow}})$. These differ by a factor of two in the second order term in $H_{i,jrow}-H_{i-1,jrow}$. Other differences arise on a nonuniform grid.

The second bracket, the implicit Coriolis terms, may be seen to already be in five point form because each corner coefficient consists of two terms of equal magnitude, but opposite sign.

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[-cddxu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right.$$
$$\left.-cddyu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right]\Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{31.16}$$

The right hand side of Equation (29.14) also remains the same in the five point equations as in the nine point equations.

$$forc_{i,jrow,}=\sum_{i'=0}^{1}\sum_{j'=0}^{1}\Big[-cddyt_{i',j'}\cdot zu_{i+i',jrow+j',1}\cdot dxu_{i+i'}\cdot\cos\phi^U_{jrow+j'}$$
$$+cddxt_{i',j'}\cdot zu_{i+i',jrow+j',2}\cdot dyu_{jrow+j'}\Big] \tag{31.17}$$

Although the five point operator approximates the continuous differential equation (29.2) well, its solutions are not exact solutions of the finite difference momentum equations (29.3) and (29.4). Moreover, the time saved by calculating a five point operator instead of a nine point operator in two places per iteration in a conjugate gradient solver is not large, and the nine point equations usually take fewer iterations to converge.

Chapter 32

Options for advecting tracers

The advection of tracers and momentum discussed in this chapter refer to advection by that portion of the velocity field which is explicitly computed by the ocean model. Depending on the model resolution, there may be need to parameterize the effects from unresolved parts of the spectrum through one or more of the schemes discussed in Chapters 35

In MOM, the advection of momentum always uses second order accurate centered differencing in space and time. This approach ensures that the first and second moments of momentum, namely the total momentum and kinetic energy, are preserved in conservative flow situations. In addition to this second order accurate scheme, other schemes are available for the advection of tracers. If no scheme is explicitly enabled, then the default second order accurate scheme is also used for tracers.

Basically, each tracer advective scheme has its advantages and disadvantages. As mentioned above, the standard second order scheme conserves first (mean) and second (variance) moments. However, it produces over(under)-shoots and phase errors when scales in the tracer or velocity field are not well resolved by the grid resolution. The fourth order advection does a better job by reducing the amplitude of the over(under)-shoots and the phase errors but does not preserve second moments in general. The flux corrected transport scheme is positive definite (no over(under)-shoots), does not conserve second moments, and tends to sharpen fronts. It is also expensive to use. The third order scheme (quicker) balances the diffusion of the scheme by its dispersion. It does not conserve second moments and is not positive definite but the over(under)-shoots are drastically reduced compared to second or fourth order schemes. It is interesting to note that "quicker" can result in less dissapation than the second order scheme because of the non-linear action of convection on the over(under)-shoots of the second order scheme.

32.1 Considerations of accuracy in one-dimension

The purpose of this section is to introduce some basic ideas regarding how well advection on the discrete lattice approximates advection in the continuum. Much of the intuition for advection comes from the many studies that have been done using a single space dimension, and this section is restricted to such systems. In this case, the continuum equation for the advection of a quantity ψ is given by

$$\psi_t + U\,\psi_x = 0. \tag{32.1}$$

In the following, the solution $\psi(x,t)$ to this equation will be termed the *continuum* field; it is defined at each continuous point (x,t). This field will be compared to the various solutions Ψ_i^n which live on the discrete space time lattice, with $x_i = i\,\Delta x$ and $t_n = n\,\Delta t$ the spatial and temporal grid points, respectively. The goal is to make the differences between the continuum and lattice fields small using discrete approximations to the continuum advection equation (32.1). Note that for the following, all considerations will be for uniform space-time lattices.

For a single space dimension, an incompressible fluid corresponds to a constant advecting velocity U. Hence, the simplest case to consider is tracer advection by a constant velocity U discretized on a uniform spatial and temporal grid $(\Delta x, \Delta t)$.

32.1.1 Lattice and continuum operators

A good reference for the methods used in this subsection is chapter 2 from the book by Mitchell and Griffiths (1980).

Consider the continuum field ψ evaluated at an arbitrary lattice point

$$\psi_i^n = \psi(x_i, t_n). \tag{32.2}$$

Using a Taylor series expansion, the relation between ψ_i^n and ψ_{i+p}^n, where $x_p = p\,\Delta x$, is given by

$$\begin{aligned} \psi_{i+p}^n &= \left(1 + p\,\Delta x\,\partial_x + \frac{1}{2!}\,(p\,\Delta x)^2 \partial_x^{(2)} + \frac{1}{3!}\,(p\,\Delta x)^3 \partial_x^{(3)} + \ldots\right) \psi_i^n \\ &= \exp(p\,\Delta x\,\partial_x)\,\psi_i^n. \end{aligned} \tag{32.3}$$

The linear operator $\exp(p\,\Delta x\,\partial_x)$ can be thought of as a spatial translation operator; a terminology familiar to those having studied quantum mechanics. Similarly, the temporal translation operator $\exp(q\,\Delta t\,\partial_t)$ connects ψ_i^n to another point in time $t_q = q\,\Delta t$ through the relation

$$\psi_i^{n+q} = \exp(q\,\Delta t\,\partial_t)\,\psi_i^n. \tag{32.4}$$

In the following, it will prove useful to derive relations between the continuum differential operators ∂_x and ∂_t and various finite difference or lattice operators. To start, consider the familiar centered difference spatial operator as defined by

$$\delta_x^C \psi_i^n = \frac{\psi_{i+1}^n - \psi_{i-1}^n}{2\Delta x}. \tag{32.5}$$

Using the spatial translation operators, this relation takes the form

$$\delta_x^C \psi_i^n = \left(\frac{\sinh(\Delta x\,\partial_x)}{\Delta x}\right) \psi_i^n. \tag{32.6}$$

Since ψ_i^n is arbitrary, this equation yields the relation between the centered difference lattice operator and the continuous partial derivative

$$\Delta x\,\delta_x^C = \sinh(\Delta x\,\partial_x). \tag{32.7}$$

Inverting this relation yields

$$\Delta x\,\partial_x = \sinh^{-1}(\Delta x\,\delta_x^C). \tag{32.8}$$

Similar relations hold for the temporal centered difference, or *leap frog*, operator

$$\Delta t\, \delta_t^C = \sinh(\Delta t\, \partial_t) \tag{32.9}$$
$$\Delta t\, \partial_t = \sinh^{-1}(\Delta t\, \delta_t^C). \tag{32.10}$$

The forward difference lattice operator is also quite common

$$\begin{aligned} \delta_x^F \psi_i^n &= \frac{\psi_{i+1}^n - \psi_i^n}{\Delta x} \\ &= [\exp(\Delta x\, \partial_x) - 1]\, \psi_i^n. \end{aligned} \tag{32.11}$$

This relation leads to the operator equalities

$$\Delta x\, \delta_x^F = \exp(\Delta x\, \partial_x) - 1 \tag{32.12}$$
$$\Delta x\, \partial_x = \ln(1 + \Delta x\, \delta_x^F), \tag{32.13}$$

with analogous results for the temporal forward difference operator. The relation between other finite difference operators and the continuum operator can be derived similarly.

32.1.2 Leap frog in time and centered in space

The previous relations between continuum and lattice operators render the differential advection equation $\psi_t + U\, \psi_x = 0$ equivalent to the finite difference advection equation

$$\left[\sinh^{-1}(\Delta t\, \delta_t^C) + \frac{U\, \Delta t}{\Delta x}\, \sinh^{-1}(\Delta x\, \delta_x^C)\right] \psi_i^n = 0, \tag{32.14}$$

where the lattice operators are each centered difference operators. Various finite difference methods can be obtained by expanding the $\sinh^{-1}$ functions and truncating at some desired order. For example, keeping only the first terms yields

$$(\delta_t^C + U\, \delta_x^C)\Psi_i^n = 0. \tag{32.15}$$

This equation defines the lattice field Ψ_i^n, which is an approximation to the continuum field ψ_i^n. In order to determine the accuracy of the approximation, consider the error field

$$D_i^n = \psi_i^n - \Psi_i^n. \tag{32.16}$$

This field satisfies the equation

$$\begin{aligned} (\delta_t^C + U\, \delta_x^C)\, D_i^n &= (\delta_t^C + U\, \delta_x^C)\, \psi_i^n \\ &= \left[\frac{1}{\Delta t} \sinh(\Delta t\, \partial_t) + \frac{U}{\Delta x} \sinh(\Delta x\, \partial_x)\right] \psi_i^n \\ &= \left(\frac{(\Delta t)^2\, \partial_t^{(3)} + U\, (\Delta x)^2\, \partial_x^{(3)}}{3!} + \ldots\right) \psi_i^n. \end{aligned} \tag{32.17}$$

where $(\delta_t^C + U\, \delta_x^C)\Psi_i^n = 0$ and $(\partial_t + U\, \partial_x)\psi_i^n = 0$ were used. This result defines the truncation error

$$E[\psi_i^n] \equiv \left(\frac{(\Delta t)^2\, \partial_t^{(3)} + U\, (\Delta x)^2\, \partial_x^{(3)}}{3!}\right) \psi_i^n + \ldots, \tag{32.18}$$

which is seen to be second order in both Δx and Δt. This expression can be written solely in terms of spatial derivatives using the relation satisfied by the continuum field

$$\partial_t^{(p)}\psi \quad = \quad (-U)^p\, \partial_x^{(p)}\psi, \tag{32.19}$$

which follows from successive differentiation of $\partial_t\psi = -U\,\partial_x\psi$. As such, the truncation error takes the form

$$E[\psi_i^n] = \frac{U\,(\Delta x)^2}{3!}\left[1 - \left(\frac{U\,\Delta t}{\Delta x}\right)^2\right]\partial_x^{(3)}\,\psi_i^n + \ldots \tag{32.20}$$

As seen from the above analysis, the lattice field Ψ_i^n only approximates the continuous field ψ due to the nonzero truncation error. A complementary issue concerns the properties of the continous field $\tilde{\psi}$ that exactly corresponds to Ψ_i^n. Namely, consider a continuum field $\tilde{\psi}$ which satisfies the discrete equation

$$(\delta_t^C + U\,\delta_x^C)\,\tilde{\psi}_i^n = 0. \tag{32.21}$$

Substituting the relations between the lattice and continuous operators into this equation yields the differential equation satisfied by $\tilde{\psi}$

$$\left(\frac{1}{\Delta t}\sinh(\Delta t\,\partial_t) + \frac{U}{\Delta x}\sinh(\Delta x\,\partial_x)\right)\tilde{\psi}_i^n \quad = \quad 0. \tag{32.22}$$

Expanding, rearranging, and dropping lattice labels reveals

$$\begin{aligned}(\partial_t + U\,\partial_x)\tilde{\psi} \quad &= \quad -\left(\frac{(\Delta t)^2\,\partial_t^{(3)}}{3!} + \frac{U(\Delta x)^2\partial_x^{(3)}}{3!} + \ldots\right)\tilde{\psi} \\ &= \quad -E[\tilde{\psi}].\end{aligned} \tag{32.23}$$

Consequently, the lattice field Ψ_i^n corresponds exactly to the continuous field $\tilde{\psi}$, where $\tilde{\psi}$ satisfies the advection equation with a linear source term determined by minus the truncation error $E[\tilde{\psi}]$. Note that in $E[\tilde{\psi}]$, it is not possible to exactly eliminate the $\partial_t^{(n)}$ operator as was done for $E[\psi]$, since $\tilde{\psi}_t \neq -U\,\tilde{\psi}_x$.

32.1.3 A critique of upwind advection

Another method to discretize the advection equation is the so-called *upwind method*. Upwind advection was used in early numerical weather models due to its good stability properties, and it still finds use in idealized ocean box models (e.g., Stommel 1961) as well as some general circulation models. It is available in MOM for use in advecting tracers along the ocean bottom (Section 32.7) and within the parameterized bottom boundary layer (Chapter 37). When using upwind, however, one should realize that it is highly diffusive, for reasons first discussed by Molenkamp (1968) and outlined in the following.

The upwind scheme uses backward space differences if the velocity is in the positive x direction, and forward space differences for negative velocities

$$\begin{array}{ll} U\,\delta_x^F\Psi_i^n & \text{if } U < 0 \\ U\,\delta_x^B\Psi_i^n & \text{if } U > 0, \end{array} \tag{32.24}$$

where $\delta_x^B \Psi_i^n = (\Psi_i^n - \Psi_{i-1}^n)/\Delta x$ is the backward difference operator. The name *upwind* denotes the use of upwind, or upstream, information in determining the form for the finite difference; downstream information is ignored.

In order to analyze the accuracy of the upwind scheme, consider the case with $U > 0$ for definiteness. In this case, leap frog in time and backward difference in space yields the finite difference equation

$$(\delta_t^C + U\,\delta_x^B)\Psi_i^n = 0. \tag{32.25}$$

The error field $D_i^n = \psi_i^n - \Psi_i^n$ satisfies the equation

$$\begin{aligned}
(\delta_t^C + U\,\delta_x^B)\,D_i^n &= (\delta_t^C + U\,\delta_x^B)\,\psi_i^n \\
&= \left(\frac{1}{\Delta t}\sinh(\Delta t\,\partial_t) + \frac{U}{\Delta x}[1 - \exp(-\Delta x\,\partial_x)]\right)\psi_i^n \\
&= \left(\frac{(\Delta t)^2\,\partial_t^{(3)}}{3!} - \frac{U\,\Delta x\,\partial_x^{(2)}}{2} + \ldots\right)\psi_i^n \\
&= -\frac{U\,\Delta x}{2}\left(1 + \frac{(U\,\Delta t)^2\,\partial_x}{3\,\Delta x} + \ldots\right)\partial_x^{(2)}\,\psi_i^n \\
&\equiv E[\psi_i^n].
\end{aligned} \tag{32.26}$$

The truncation error is seen to be second order in Δt and first order in Δx.

To determine the differential equation to which the finite difference equation corresponds, introduce the continuum field $\tilde{\psi}$ which satisfies

$$\begin{aligned}
(\delta_t^C + U\,\delta_x^B)\tilde{\psi}_i^n &= \left(\frac{1}{\Delta t}\sinh(\Delta t\,\partial_t) + \frac{U}{\Delta x}[1 - \exp(-\Delta x\,\partial_x)]\right)\tilde{\psi}_n^i \\
&= 0.
\end{aligned} \tag{32.27}$$

Expanding the transcendental functions and rearranging yields the differential equation

$$\begin{aligned}
(\partial_t + U\,\partial_x)\tilde{\psi} &= \left(-\frac{(\Delta t)^2\,\partial_t^{(3)}}{3!} + \frac{U\,\Delta x\,\partial_x^{(2)}}{2} + \ldots\right)\tilde{\psi} \\
&= -E[\tilde{\psi}].
\end{aligned} \tag{32.28}$$

Since $\tilde{\psi}$ does not solve the continuum advection equation, it is not possible to exactly eliminate the $\partial_t^{(3)}$ operator in terms of the $\partial_x^{(3)}$ operator as done for $E[\psi]$. Regardless, the main point here is that the lowest order derivative in the truncation error represents a dissipative, or diffusive, term where the numerical or effective diffusivity is given by

$$\kappa = (U\,\Delta x)/2. \tag{32.29}$$

For a one-degree model at the equator with horizontal velocity $U = 20cm/sec$, the numerical diffusivity is roughly $10^8 cm^2/sec$. If upwind advection is used in the vertical, with $U = 10^{-3}cm/sec - 10^{-2}cm/sec$ and $\Delta z = 1000cm$, the vertical numerical diffusivity is roughly $0.5cm^2/sec - 5cm^2/sec$. Both diffusivities are huge. Granted, it is possible that the higher order terms in the truncation error will alter these numbers. However, there is no reason to believe that they will systematically reduce the diffusivities to the extent necessary to bring them more in line with

observations. The paper by Griffies, Pacanowski, and Hallberg (1999) further discusses these points for upwind and other advection schemes.

The analysis of Molenkamp (1968) considered the case of upwind advection with forward time differences. In this case, the finite difference equation for $U > 0$ is

$$(\delta_t^F + U\,\delta_x^B)\Psi_i^n = 0. \tag{32.30}$$

This equation corresponds to the differential equation

$$\left(\frac{1}{\Delta t}[\exp(\Delta t\,\partial_t) - 1] + \frac{U}{\Delta x}[1 - \exp(-\Delta x\,\partial_x)]\right)\tilde{\psi}_i^n = 0. \tag{32.31}$$

Expanding and rearranging yields

$$(\partial_t + U\,\partial_x)\tilde{\psi} \;=\; \frac{U\,\Delta x}{2}\left(\partial_x^{(2)} - \frac{\Delta t\,\partial_t^{(2)}}{U\,\Delta x} + \ldots\right)\tilde{\psi}. \tag{32.32}$$

If one drops derivatives higher than second, or does some back substitution, it is possible to approximately eliminate the $\partial_t^{(2)}$ term in favor of the $\partial_x^{(2)}$. In this way, the equation satisfied by $\tilde{\psi}$ is given by

$$(\partial_t + U\,\partial_x)\tilde{\psi} \;=\; \frac{U\,\Delta x}{2}\left(1 - \frac{U\,\Delta t}{\Delta x}\right)\partial_{xx}\,\tilde{\psi} + \ldots. \tag{32.33}$$

This result corresponds to equation (15) of Molenkamp (1968) and equation (32.88) discussed in the FCT section of this chapter. Note that stability of the upwind scheme with forward time difference requires $|U|\,\Delta t/\Delta x < 1$, thus leading to a positive effective diffusivity. It is seen that the effective diffusivity is systematically reduced from that resulting from the leap frog scheme. Interestingly, as the CFL number $|U|\,\Delta t/\Delta x$ approaches unity, the leading term in the truncation error vanishes, and so the upwind scheme, with forward time differences, becomes more accurate. This fortuitous situation, however, is not general and so is not justification for using upwind.

In general, as argued here and in many other places, the upwind advection scheme is too diffusive to be of direct use for realistic ocean general circulation modeling. However, the upwind scheme in combination with a less dissipative scheme, such centered differences, can be quite useful for many purposes. The discussion of FCT in Section 32.6 clarifies this point. Additionally, for use in advecting tracers along the ocean bottom (Section 32.7) and within the parameterized bottom boundary layer (Chapter 37).

32.2 second_order_tracer_advection

This option is automatically enabled in file *size.h* if no other advective schemes are enabled. Do not directly enable it with a preprocessor directive. Twice the advective flux on the eastern, northern, and bottom sides of cell $T_{i,k,j}$is given by

$$adv_fe_{i,k,j} \;=\; adv_vet_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}) \tag{32.34}$$

$$adv_fn_{i,k,j} \;=\; adv_vnt_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau}) \tag{32.35}$$

$$adv_fb_{i,k,j} \;=\; adv_vbt_{i,k,j}\cdot(t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau}) \tag{32.36}$$

Note that twice the advective flux is computed in the model. For purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in adv_fe) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux.

32.3 linearized_advection

It is sometimes useful to linearize about a state of no motion as in Philander and Pacanowski (1980). The state of no motion is given by

$$t = t(z) \tag{32.37}$$

$$\rho = \rho(t) \tag{32.38}$$

where t is temperature and the density ρ is linearized as in option *linearized_density*. If the advective velocities are thought of as being composed of a mean and deviation, then linearizing about this state eliminates the advective terms in the momentum equations. In the temperature equation, only the advective term $w \cdot \bar{t}_z$ remains where $\bar{t}$ is an initial equatorial stratification given by Equation 28.1. Note that this stratification is the initial condition. Options *levitus_ic* and *idealized_ic* are therefore incompatible and must not be enabled with option *linearized_advection*. It is not necessary to enable option *equatorial_thermocline* for linearized advection.

32.4 fourth_order_tracer_advection

Option *fourth_order_tracer_advection* replaces the second order advective scheme with a fourth order scheme and requires option *fourth_order_memory_window* to be enabled. This is automatically done when option *fourth_order_tracer_advection* is enabled. Consider any quantity q_i for $i = 1$ *to* N points. Expanding q_{i+1}, q_{i-1}, q_{i+2}, and q_{i-2} in a Taylor series about q_i yields

$$q_{i+1} = q_i + \frac{\partial q_i}{\partial i} + \frac{1}{2}\frac{\partial^2 q_i}{\partial^2 i} + \frac{1}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{1}{24}\frac{\partial^4 q_i}{\partial^4 i} + \cdots \tag{32.39}$$

$$q_{i-1} = q_i - \frac{\partial q_i}{\partial i} + \frac{1}{2}\frac{\partial^2 q_i}{\partial^2 i} - \frac{1}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{1}{24}\frac{\partial^4 q_i}{\partial^4 i} + \cdots \tag{32.40}$$

$$q_{i+2} = q_i + 2\frac{\partial q_i}{\partial i} + 2\frac{\partial^2 q_i}{\partial^2 i} + \frac{8}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{16}{24}\frac{\partial^4 q_i}{\partial^4 i} + \cdots \tag{32.41}$$

$$q_{i-2} = q_i - 2\frac{\partial q_i}{\partial i} + 2\frac{\partial^2 q_i}{\partial^2 i} - \frac{8}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{16}{24}\frac{\partial^4 q_i}{\partial^4 i} + \cdots \tag{32.42}$$

The above expansions can be combined to yield

$$(q_{i+1} + q_i) - (q_i + q_{i-1}) \approx 2\frac{\partial q_i}{\partial i} + \frac{1}{3}\frac{\partial^3 q_i}{\partial^3 i} \tag{32.43}$$

$$(q_{i+2} + q_{i-1}) - (q_{i+1} + q_{i-2}) \approx 2\frac{\partial q_i}{\partial i} + \frac{7}{3}\frac{\partial^3 q_i}{\partial^3 i} \tag{32.44}$$

Multiplying Equation (32.43) by 7, subtracting Equation (32.44) and changing to grid spacing Δx yields

$$\partial q_i/\partial x \approx \frac{F_i - F_{i-1}}{\Delta x} \tag{32.45}$$

$$F_i = A(q_{i+1} + q_i) - B(q_{i+2} + q_{i-1}) \tag{32.46}$$

where $A = 7/12$ and $B = 1/12$ for fourth order accuracy and $A = 1/2$ and $B = 0$ for second order accuracy. Since the argument is the same for zonal, meridional, and vertical directions, consider the zonal direction. A fully fourth order advective scheme would set

$$q_{i,k,j} = \overline{adv_vet_{i-1,k,j}}^{\lambda} \cdot t_{i,k,j,n,\tau} \tag{32.47}$$

However, as implemented in MOM, the scheme is only psuedo fourth order[1] because the advecting velocity is left second order while the average tracer on the cell faces is expanded to fourth order using

$$F_i = adv_vet_{i,k,j} \cdot (A(t_{i+1,k,j,n,\tau} + t_{i,k,j,n,\tau}) - B(t_{i+2,k,j,n,\tau} + t_{i-1,k,j,n,\tau})) \tag{32.48}$$

Note that for ocean T cells adjacent to land cells, the scheme is reduced to second order. This is easily accomplished by using a land/sea mask to select the appropriate coefficients A and B for each T cell. In principle, a fully fourth order scheme could be implemented by expanding fluxes according to Equation (32.47) but this has not been done. Also, this scheme has only been implemented for the case of constant resolution.

32.5 quicker

This is a third order advection scheme for tracers which significantly reduces the over-shooting inherent in the second order center differenced advection scheme. The cost is less than a 10% increase in overall time. It is enabled by option *quicker* and is based on the "quick" scheme of Leonard (1979), but has been modified to lag the upstream correction by putting it on time level $\tau - 1$. This approach was recommended by Jeurgen Willebrand (personal communication, 1995) who demonstrated it in a one dimensional advection diffusion model. It has also been incorporated into the NCAR ocean model as described by Holland *et al.* (1998). The advantage of this lagging in time is that it allows the same time step to be used as for second order advection. If not done, the scheme is unstable unless the time step is reduced by about one half. The discretization of Farrow and Stevens (1995) has been followed but not their predictor corrector method since the lagged correction mentioned above solves the stability problem. The formulation given in NCAR (1996) is recovered by changing the $\tau - 1$ to τ in all upstream corrective terms. This is done by additionally enabling option *ncar_upwind3*. Each direction is discretized independently of others and so the scheme is un-compensated for multi-dimensions. In the zonal direction, twice the advective flux out of the eastern face of the T cell is given by

$$u^+ = (adv_vet_{i,k,j} + |adv_vet_{i,k,j}|)/2 \tag{32.49}$$

$$u^- = (adv_vet_{i,k,j} - |adv_vet_{i,k,j}|)/2 \tag{32.50}$$

[1]The idea of a psuedo fourth order technique was taken from the GFDL SKYHI stratospheric GCM.

$$\begin{aligned} adv_fe_{i,k,j} &= adv_vet_{i,k,j} \cdot (quick^{x}_{i,1} \cdot t_{i,k,j,n,\tau} + quick^{x}_{i,2} \cdot t_{i+1,k,j,n,\tau}) \\ &- u^{+} \cdot (curv^{x+}_{i,1} \cdot t_{i+1,k,j,n,\tau-1} + curv^{x+}_{i,2} \cdot t_{i,k,j,n,\tau-1} + curv^{x+}_{i,3} \cdot t_{i-1,k,j,n,\tau-1} \\ &- u^{-} \cdot (curv^{x-}_{i,1} \cdot t_{i+2,k,j,n,\tau-1} + curv^{x-}_{i,2} \cdot t_{i+1,k,j,n,\tau-1} + curv^{x-}_{i,3} \cdot t_{i,k,j,n,\tau-1}) \end{aligned} \tag{32.51}$$

where the coefficients are given by

$$quick^{x}_{i,1} = 2 \cdot dxt_{i+1}/(dxt_{i+1} + dxt_{i}) \tag{32.52}$$

$$quick^{x}_{i,2} = 2 \cdot dxt_{i}/(dxt_{i+1} + dxt_{i}) \tag{32.53}$$

$$curv^{x+}_{i,1} = 2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i-1} + 2 \cdot dxt_{i} + dxt_{i+1}) \cdot (dxt_{i} + dxt_{i+1})) \tag{32.54}$$

$$curv^{x+}_{i,2} = -2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i} + dxt_{i+1}) \cdot (dxt_{i-1} + dxt_{i})) \tag{32.55}$$

$$curv^{x+}_{i,3} = 2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i-1} + 2 \cdot dxt_{i} + dxt_{i+1}) \cdot (dxt_{i-1} + dxt_{i})) \tag{32.56}$$

$$curv^{x-}_{i,1} = 2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i} + 2 \cdot dxt_{i+1} + dxt_{i+2}) \cdot (dxt_{i+1} + dxt_{i+2})) \tag{32.57}$$

$$curv^{x-}_{i,2} = -2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i+1} + dxt_{i+2}) \cdot (dxt_{i} + dxt_{i+1})) \tag{32.58}$$

$$curv^{x-}_{i,3} = 2 \cdot dxt_{i} * dxt_{i+1}/((dxt_{i} + 2 \cdot dxt_{i+1} + dxt_{i+2}) \cdot (dxt_{i} + dxt_{i+1})) \tag{32.59}$$

In the meridional direction, twice the advective flux out of the northern face of the T cell is given by

$$v^{+} = (adv_vnt_{i,k,j} + |adv_vnt_{i,k,j}|)/2 \tag{32.60}$$

$$v^{-} = (adv_vnt_{i,k,j} - |adv_vnt_{i,k,j}|)/2 \tag{32.61}$$

$$\begin{aligned} adv_fn_{i,k,j} &= adv_vnt_{i,k,j} \cdot (quick^{y}_{jrow,1} \cdot t_{i,k,j,n,\tau} + quick^{y}_{jrow,2} \cdot t_{i,k,j+1,n,\tau}) \\ &- v^{+} \cdot (curv^{y+}_{jrow,1} \cdot t_{i,k,j+1,n,\tau-1} + curv^{y+}_{jrow,2} \cdot t_{i,k,j,n,\tau-1} + curv^{y+}_{jrow,3} \cdot t_{i,k,j-1,n,\tau-1} \\ &- v^{-} \cdot (curv^{y-}_{jrow,1} \cdot t_{i,k,j+2,n,\tau-1} + curv^{y-}_{jrow,2} \cdot t_{i,k,j+1,n,\tau-1} + curv^{y-}_{jrow,3} \cdot t_{i,k,j,n,\tau-1}) \end{aligned} \tag{32.62}$$

where the coefficients are given by

$$quick^{y}_{jrow,1} = 2 \cdot dyt_{jrow+1}/(dyt_{jrow+1} + dyt_{jrow}) \tag{32.63}$$

$$quick^{y}_{jrow,2} = 2 \cdot dyt_{jrow}/(dyt_{jrow+1} + dyt_{jrow}) \tag{32.64}$$

$$\begin{aligned} curv^{y+}_{jrow,1} &= 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow-1} + 2 \cdot dyt_{jrow} + dyt_{jrow+1}) \\ &\cdot (dyt_{jrow} + dyt_{jrow+1})) \end{aligned} \tag{32.65}$$

$$\begin{aligned} curv^{y+}_{jrow,2} &= -2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + dyt_{jrow+1}) \\ &\cdot (dyt_{jrow-1} + dyt_{jrow})) \end{aligned} \tag{32.66}$$

$$\begin{aligned} curv^{y+}_{jrow,3} &= 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow-1} + 2 \cdot dyt_{jrow} + dyt_{jrow+1}) \\ &\cdot (dyt_{jrow-1} + dyt_{jrow})) \end{aligned} \tag{32.67}$$

$$\begin{aligned} curv^{y-}_{jrow,1} &= 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + 2 \cdot dyt_{jrow+1} + dyt_{jrow+2}) \\ &\cdot (dyt_{jrow+1} + dyt_{jrow+2})) \end{aligned} \tag{32.68}$$

$$
\begin{aligned}
curv^{y-}_{jrow,2} &= -2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow+1} + dyt_{jrow+2}) \\
&\quad \cdot (dyt_{jrow} + dyt_{jrow+1})) && (32.69) \\
curv^{y-}_{jrow,3} &= 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + 2 \cdot dyt_{jrow+1} + dyt_{jrow+2}) \\
&\quad \cdot (dyt_{jrow} + dyt_{jrow+1})) && (32.70)
\end{aligned}
$$

Note that the indices in the above expressions require that option *fourth_order_memory_window* be enabled. This is automatically done when option *quicker* is enabled. Also, for ocean cells next to land cells (and the surface), the correction term on the face parallel to the land boundary is dropped thereby reducing the flux to second order. Normal flux on faces shared by land cells is set to zero. Masking (not shown) is used to enfore this.

In the vertical direction, twice the advective flux through the bottom face of the T cell is given by

$$
\begin{aligned}
w^+ &= (adv_vbt_{i,k,j} + |adv_vbt_{i,k,j}|)/2 && (32.71) \\
w^- &= (adv_vbt_{i,k,j} - |adv_vbt_{i,k,j}|)/2 && (32.72) \\
adv_fb_{i,k,j} &= adv_vbt_{i,k,j} \cdot (quick^z_{k,1} \cdot t_{i,k,j,n,\tau} + quick^z_{k,2} \cdot t_{i,k+1,j,n,\tau}) \\
&- w^- \cdot (curv^{z+}_{k,1} \cdot t_{i,k+1,j,n,\tau-1} + curv^{z+}_{k,2} \cdot t_{i,k,j,n,\tau-1} + curv^{z+}_{k,3} \cdot t_{i,k-1,j,n,\tau-1} \\
&- w^+ \cdot (curv^{z-}_{k,1} \cdot t_{i,k+2,j,n,\tau-1} + curv^{z-}_{k,2} \cdot t_{i,k+1,j,n,\tau-1} + curv^{z-}_{k,3} \cdot t_{i,k,j,n,\tau-1}) \\
& && (32.73)
\end{aligned}
$$

Note the way w^- and w^+ are used to account for a z axis which is positive upwards. The coefficients are given by

$$
\begin{aligned}
quick^z_{k,1} &= 2 \cdot dzt_{k+1}/(dzt_{k+1} + dzt_k) && (32.74) \\
quick^z_{k,2} &= 2 \cdot dzt_k/(dzt_{k+1} + dzt_k) && (32.75) \\
curv^{z+}_{k,1} &= 2 \cdot dzt_k * dzt_{k+1}/((dzt_{k-1} + 2 \cdot dzt_k + dzt_{k+1}) \cdot (dzt_k + dzt_{k+1})) && (32.76) \\
curv^{z+}_{k,2} &= -2 \cdot dzt_k * dzt_{k+1}/((dzt_k + dzt_{k+1}) \cdot (dzt_{k-1} + dzt_k)) && (32.77) \\
curv^{z+}_{k,3} &= 2 \cdot dzt_k * dzt_{k+1}/((dzt_{k-1} + 2 \cdot dzt_k + dzt_{k+1}) \cdot (dzt_{k-1} + dzt_k)) && (32.78) \\
curv^{z-}_{k,1} &= 2 \cdot dzt_k * dzt_{k+1}/((dzt_k + 2 \cdot dzt_{k+1} + dzt_{k+2}) \cdot (dzt_{k+1} + dzt_{k+2})) && (32.79) \\
curv^{z-}_{k,2} &= -2 \cdot dzt_k * dzt_{k+1}/((dzt_{k+1} + dzt_{k+2}) \cdot (dzt_k + dzt_{k+1})) && (32.80) \\
curv^{z-}_{k,3} &= 2 \cdot dzt_k * dzt_{k+1}/((dzt_k + 2 \cdot dzt_{k+1} + dzt_{k+2}) \cdot (dzt_k + dzt_{k+1})) && (32.81)
\end{aligned}
$$

32.6 fct

This section was contributed by Ruediger Gerdes (*rgerdes@AWI − Bremerhaven.de*). The main disadvantage of the widely used central differences advection scheme (or other higher order schemes) is the numerical dispersion that is most noticeable near large gradients in the advected quantity. Non-physical oscillations or "ripples" (under and overshoots) and negative concentrations of positive definite quantities may occur. Addition of explicit diffusion in the coordinate directions is required to reduce or eliminate this problem. The one-dimensional advection diffusion equation

$$U\frac{\partial T}{\partial x} = A\frac{\partial}{\partial x}(\frac{\partial T}{\partial x}) \tag{32.82}$$

discretized with central differences

$$\frac{U}{2\Delta x}(T_{i+1} - T_{i-1}) = \frac{A}{\Delta x^2}(T_{i+1} - 2T_i + T_{i-1}) \tag{32.83}$$

has solutions of the form $T_i = \xi^i$ which when put into Equation (32.83) results in a quadratic equation with roots

$$\xi = 1 \quad and \quad \xi = (2 + Pe)/(2 - Pe) \tag{32.84}$$

where $Pe = U\Delta x/A$ is the Péclèt number for the grid scale Δx. The second solution changes sign from grid point to grid point (two grid point noise) unless the grid Péclèt number is less than two. Simple estimates demonstrate that the required diffusion in a typical ocean model is rather large. For a current of 10 cm/s and a grid distance of 100 km, a diffusion coefficient of 5x10^7 cm^2/s is implied. A moderate vertical velocity of 10^{-5} cm/s and a grid distance of 100m would require a vertical diffusion coefficient of 0.25 cm^2/s. Note that in the deep ocean grid distances are usually much larger and vertical velocities can easily be one or two orders of magnitude larger. In the standard case of constant diffusion coefficients the numerical requirements in regions of strong currents determine the magnitude of the coefficients. In quiet regions this implies a diffusively dominated tracer balance that is not physically justified.

The above analysis only consideres a one dimensional advective diffusive balance and the requirements on diffusion to suppress two grid point noise can be relaxed somewhat in three dimensions. However, the problem is indeed of great practical importance as shown, among others, by Weaver and Sarachik 1990, Gerdes et al. 1991, Farrow and Stevens 1995.

The upstream scheme is an equally simple advection scheme that is free from the dispersive effects mentioned above. However, it has very different numerical errors. The main disadvantage of the only first order accurate scheme is its large amount of implicit diffusion. Here one-sided upstream differences are used to calculate the advective fluxes. The upstream discretized advective diffusive balance in one dimension is

$$\frac{U + |U|}{2\Delta tx}(T_i - T_{i-1}) + \frac{U - |U|}{2\Delta x}(T_{i+1} - T_i) = \frac{A}{\Delta x^2}(T_{i+1} - 2T_i + T_{i-1}) \tag{32.85}$$

and the solution is as given above for the central differences scheme except that the grid Péclèt number is replaced by

$$Pe' = (2U\Delta x)/(2A + |U|\Delta x) \tag{32.86}$$

that is always less than two. The truncation error of the advection scheme in multi-dimensions is

$$\sum_i \frac{\partial}{\partial_i} 0.5(|u_i|\Delta x_i - \Delta t u_i^2) + \sum_{i \neq j} 0.5\Delta t u_i u_j \frac{\partial T}{\partial x_j} \tag{32.87}$$

which can be interpreted as implicit diffusion with diffusion coefficients given by

$$A^i_{impl} = 0.5(|u_i|\Delta x_i - \Delta t u_i^2) \tag{32.88}$$

For small time steps (small compared to the maximum time step allowed by the CFL criterion) the effective diffusion is such that the grid Péclèt number is two. It should be noted that the tracer balance is thus always advectively dominated. Therefore the upstream scheme might actually be less diffusive than the central differences scheme with explicit diffusion in the larger part of the model domain.

Central differences and upstream algorithms represent, in a sense, opposite extremes, each minimizing one kind of error at the expense of another. A linear compromise between both schemes may be useful in certain cases (e.g. Fiadeiro and Veronis 1977) but will in general exhibit dispersive effects. A nonlinear compromise is the flux-corrected transport (FCT) algorithm (Boris and Book 1973; Zalesak 1979. A comparison of the central differences, upstream and FCT schemes for (oceanographic) two- and three-dimensional examples is given in Gerdes et al. 1991). Here the flux difference (anti-diffusive flux) between a central difference scheme (or any other higher order scheme) and an upstream scheme is computed. Adding the anti-diffusive flux in full to the upstream flux would maximally reduce diffusion but introduce dispersive ripples. The central idea is to limit the anti-diffusive flux locally such that no under and overshoots are introduced.

One possible criterion is to insist that from one time step to the next no new maxima or minima around one grid cell are created by advection. As remarked by Rood (1987), the FCT is more a philosophy rather than an explicit algorithm, as the crucial limiting step is essentially left to the user's discretion. Depending on the choice of the limiting step the results will be closer to those of either the upstream or the central differences scheme. The amount of implicit mixing does, therefore, depend on a subjective choice. With this limitation in mind, the FCT algorithm may be regarded as a way to find the minimum mixing that is consistent with the thermodynamical constraint. With the FCT scheme, the model can be run without any explicit diffusion and the author suggests running a case with all tracer diffusion coefficients set to zero to appreciate the effects of the advection scheme. This offers an opportunity to study cases where the tracer balance is advectively dominated everywhere. Furthermore, the scheme allows use of physically motivated mixing that will not be swamped by numerically necessary explicit diffusion or large implicit diffusion of the advection scheme.

The recommended options (all of which should be specified) for the FCT scheme are *fct*, *fct_dlm1*, and *fct_3d*. An alternative option to *fct_dlm1* is *fct_dlm2* which specifies the limiter as originally proposed by Zalesak (1979). Changes in tracer due to FCT, (i.e. FCT minus central differences), are written in NetCDF format to file *fct.yyyyyy.mm.dd.dta.nc* when the diagnostic option *fct_netcdf* is enabled. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. Two additional options *tst_fct_his* and *tst_fct_los* were introduced mainly for debugging purposes. Option *tst_fct_his* suppresses the limitation of the anti-diffusive fluxes and thus results in the high-order scheme. With this option, the model should reproduce the results of the central differences scheme. Option *tst_fct_los* forces a complete limitation of the anti-diffusive fluxes thus realizing the upstream scheme. However, all intermediate steps of the algorithm are performed. So for performance reasons, it is not recommended to use *tst_fct_los* to implement an upstream scheme.

With option *fct* enabled, the advective fluxes are calculated in subroutine *adv_flx_fct*. The implementation closely follows the FCT algorithm as given by Zalesak (1979). The low-order (upstream) fluxes are calculated first and a preliminary upstream solution is given by

$$t^{low}_{i,k,j,n} = t_{i,k,j,n,\tau-1} - 2\Delta t(ADV_Tx_{i,k,j} + ADV_Ty_{i,k,j} + ADV_Tz_{i,k,j}) \tag{32.89}$$

where the advective operator is defined by Equations (22.62), (22.63), (22.64) except that the fluxes are given by

$$\begin{aligned} adv_fe^{ups}_{i,k,j} &= adv_vet_{i,k,j}(t_{i,k,j,n,\tau-1} + t_{i+1,k,j,n,\tau-1}) \\ &+ |adv_vet_{i,k,j}|(t_{i,k,j,n,\tau-1} - t_{i+1,k,j,n,\tau-1}) \qquad (32.90) \\ adv_fn^{ups}_{i,k,j} &= adv_vnt_{i,k,j}(t_{i,k,j,n,\tau-1} + t_{i,k,j+1,n,\tau-1}) \\ &+ |adv_vnt_{i,k,j}|(t_{i,k,j,n,\tau-1} - t_{i,k,j+1,n,\tau-1}) \qquad (32.91) \\ adv_fb^{ups}_{i,k,j} &= adv_vbt_{i,k,j}(t_{i,k,j,n,\tau-1} + t_{i,k+1,j,n,\tau-1}) \\ &+ |adv_vbt_{i,k,j}|(t_{i,k+1,j,n,\tau-1} - t_{i,k,j,n,\tau-1}) \qquad (32.92) \end{aligned}$$

Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in *adv_fe*) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. A forward time step over $2\Delta t$ is used because the usual forward time step turned out to be unstable in the ocean model. For stability reasons, discretization by Equation (32.89) is not allowed with option *damp_inertial_oscillation* which treats the Coriolis term implicitly.

32.6.1 Sub-options fct_dlm1 and fct_dlm2

The next step involves limitation of anti-diffusive fluxes using options *fct_dlm1* or *fct_dlm2*. The anti-diffusive fluxes are limited for each coordinate direction separately. Flux limitation in three dimensions is (optionally) done afterwards. This procedure was recommended by Zalesak (1979) in the case that a tracer is transported in a direction perpendicular to a large gradient in the tracer. In ocean models, the possible range of the solution is frequently given by a large variation in the vertical direction while the largest anti-diffusive fluxes occur in the horizontal direction. Experience shows that using only three-dimensional limiting results in very noisy fields although the solution is free from overshoots and undershoots[2].

As an example, the following presents details of the algorithm for the one-dimensional limiter in the x-direction. The procedure is the same for the other coordinate directions. Assume that the solution is required to stay within bounds given by Tr_i^{max} and Tr_i^{min}. There are currently two different ways to calculate these bounds and are selected by options *fct_dlm1* and *fct_dlm2*. For option *fct_dlm1* these bounds are specified as

$$\begin{aligned} Tr_i^{max} &= max(\frac{t_{i-1,k,j,n,\tau} + t_{i,k,j,n,\tau}}{2}, \frac{t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}}{2}, t^{low}_{i,k,j,n}) \\ Tr_i^{min} &= min(\frac{t_{i-1,k,j,n,\tau} + t_{i,k,j,n,\tau}}{2}, \frac{t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}}{2}, t^{low}_{i,k,j,n}) \qquad (32.93) \end{aligned}$$

while option *fct_dlm2* employs

[2]Experimentation with the limitation process can be useful: The combination of two-dimensional limiting in the horizontal and one-dimensional limiting in the vertical is likely to generate less implicit diffusion than the implemented scheme.

$$
\begin{aligned}
Tr_i^{max} &= max(t_{i-1,k,j,n}^{low}, t_{i,k,j,n}^{low}, t_{i+1,k,j,n}^{low}) \\
Tr_i^{min} &= min(t_{i-1,k,j,n}^{low}, t_{i,k,j,n}^{low}, t_{i+1,k,j,n}^{low})
\end{aligned} \tag{32.94}
$$

which is the original formula of Zalesak (1979). The upstream solution at all neighbouring points enters the version given by Equation (32.94) which requires additional storage for the meridional direction. With the version given by Equation (32.93), the current values of the tracers at neighbouring points are used instead of the upstream solution that enters only at the central point. Experimentally, the author has found that differences in solutions using Equations (32.93) and (32.94) are very small and thus option *fct_dml1* is recommended in general. Obviously, Equations (32.93) and (32.94) are not the only possible choices for upper and lower bounds on the solution. Narrower bounds will make the solution more diffusive. Specification of Tr_i^{max} and Tr_i^{min} can be used to keep the solution within a certain range (always positive for example).

To calculate the limiters, the possible change of the solution in either direction is determined by considering the sum of anti-diffusive fluxes into and out of the grid cell. For the x-direction:

$$
P_{i,k,j}^{+} = max(0, A_e_{i-1,k,j}) - min(0, A_e_{i,k,j}) \tag{32.95}
$$

$$
P_{i,k,j}^{-} = max(0, A_e_{i,k,j}) - min(0, A_e_{i-1,k,j}) \tag{32.96}
$$

where

$$
A_e_{i,k,j} = 2\Delta t \frac{anti_fe_{i,k,j}}{2\cos\phi_{jrow}^{T} dxt_i} \tag{32.97}
$$

and

$$
anti_fe_{i,k,j} = adv_fe_{i,k,j} - adv_fe_{i,k,j}^{ups} \tag{32.98}
$$

is the anti-diffusive flux at the eastern edge of the tracer cell. It is worth noting that in the vertical direction, the k index in the expressions for $P_{i,k,j}^{+}$ and $P_{i,k,j}^{-}$ would take the form

$$
P_{i,k,j}^{+} = max(0, A_b_{i,k,j}) - min(0, A_b_{i,k-1,j}) \tag{32.99}
$$

$$
P_{i,k,j}^{-} = max(0, A_b_{i,k-1,j}) - min(0, A_b_{i,k,j}) \tag{32.100}
$$

where $A_b_{i,k,j}$ is the couterpart to Equation (32.97) at the bottom of the cell face. The change in k indices is due to the fact that the z axis is positive upwards but the index k is positive downwards. The maximum permitted positive or negative changes in the solution due to the divergence of the delimited anti-diffusive fluxes are

$$
\begin{aligned}
Q_{i,k,j}^{+} &= Tr_i^{max} - t_{i,k,j}^{low} \\
Q_{i,k,j}^{-} &= t_{i,k,j}^{low} - Tr_i^{min}
\end{aligned} \tag{32.101}
$$

so that with the ratios[3]

$$
\begin{aligned}
R^{+}_{i,k,j} &= min(1, \frac{Q^{+}_{i,k,j}}{P^{+}_{i,k,j} + \epsilon}) \\
R^{-}_{i,k,j} &= min(1, \frac{Q^{-}_{i,k,j}}{P^{-}_{i,k,j} + \epsilon})
\end{aligned} \tag{32.102}
$$

the limiters can be defined as

$$
\begin{aligned}
C_e_{i,k,j} &= min(R^{+}_{i+1,k,j}, R^{-}_{i,k,j}) \quad for \quad A_e_{i,k,j} \geq 0 \\
C_e_{i,k,j} &= min(R^{+}_{i,k,j}, R^{-}_{i+1,k,j}) \quad for \quad A_e_{i,k,j} < 0
\end{aligned} \tag{32.103}
$$

32.6.2 Sub-option fct_3d

Limitation of the anti-diffusive fluxes in the coordinate directions separately does not guarantee that the solution stays in the permitted range. To assure that no undershoots and overshoots appear a three-dimensional limitation of the anti-diffusive fluxes must be performed. This is accomplished with option *fct_3d*. This option is, however, not automatically enabled because the one-dimensional limitation is sufficient in many cases and has slightly less implicit diffusion than the full scheme.

The one-dimensional scheme shown above easily generalizes to multiple dimensions. For instance, the possible increase in the solution by anti-diffusive fluxes into the grid cell becomes

$$
\begin{aligned}
P^{+}_{i,k,j} &= max(0, A_e_{i-1,k,j}) - min(0, A_e_{i,k,j}) \\
&+ max(0, A_b_{i,k,j}) - min(0, A_b_{i,k-1,j}) \\
&+ max(0, A_n_{i,k,j-1}) - min(0, A_n_{i,k,j})
\end{aligned} \tag{32.104}
$$

where the "A"s are defined analogously to Equation (32.97). For option *fct_dlm2*, the upper bound for the solution is

$$
Tr^{max}_{i,k,j} = max(t^{low}_{i-1,k,j,n}, t^{low}_{i+1,k,j,n}, t^{low}_{i,k-1,j,n}, t^{low}_{i,k+1,j,n}, t^{low}_{i,k,j-1,n}, t^{low}_{i,k,j+1,n}, t^{low}_{i,k,j,n}) \tag{32.105}
$$

The additional computational load due to the three-dimensional limiter is moderate because most of the needed maxima and minima have already been computed during the calculation of the one-dimensional limiters. Total advective fluxes are given by

$$
\begin{aligned}
adv_fe_{i,k,j} &= adv_fe^{ups}_{i,k,j} + C_e_{i,k,j} \cdot anti_fe_{i,k,j} \\
adv_fn_{i,k,j} &= adv_fn^{ups}_{i,k,j} + C_n_{i,k,j} \cdot anti_fn_{i,k,j} \\
adv_fb_{i,k,j} &= adv_fb^{ups}_{i,k,j} + C_b_{i,k,j} \cdot anti_fb_{i,k,j}
\end{aligned} \tag{32.106}
$$

[3]Where ϵ is a small value O(10^{-25}) to avoid division by zero.

Note that for purposes of speed in the Fortran code, an extra factor of 2 appears in all advective fluxes (i.e. in *adv_fe*) to cancel a factor of 2 in the averaging operator. The cancelled factor of 2 is reclaimed in the divergence operator (i.e. ADV_Ux) by combining it with a grid spacing from the derivative so as to allow one less multiply operation when computing the divergence of flux. In comparing the program code with this description, it will noticed that most quantities are computed for row *j+1* instead of row *j*. The anti-diffusive flux $anti_fn_{j+1}$ is needed to compute R_{j+1} which enters Equation (32.103) for the meridional direction. If option *fct_dlm2* is used, the low-order solution is needed in Equation (32.94). For economic reasons, zonal and vertical fluxes for row *j+1* are already calculated and delimited at row *j*. The meridional flux $anti_fn_{j+1}$ is calculated but not yet limited at row j while $anti_fn_j$ has already been calculated the row before and is now delimited at row *j*.

The scheme is expensive in terms of computer time. The time spent in subroutine tracer increases by a factor 2.5. In a typical coarse resolution model with potential temperature and salinity subroutine tracer may require 40% of the total time, in which case the CPU time of the ocean model with FCT will be 1.6 times larger than without.

32.7 bottom_upwind

One of the most pernicious problems with models that use realistically small mixing in the ocean interior is that this mixing is not sufficient to eliminate problems with numerical "digging" next to realistic bottom topography. Digging is the accumulation of unphysical tracer extrema which are localized near topography. Partial cells (Chapter 26), unfortunately, have not eliminated digging. Smoothing the bottom topography does reduce digging, and smoothing is a method of choice for many modelers.

Another approach to reducing digging is to allow the amount of mixing next to the bottom to be enhanced. Such is perhaps occurring in the real ocean, and so might be a step towards a more realistic simulation. Indeed, in the bottom boundary layer (Chapter 37), an enhanced amount of mixing is enabled through the use of upwind tracer advection. As shown in Section 32.1.3, the mixing from upwind is larger when the currents are stronger, which is arguably a realistic feature. Use of the bottom boundary layer has resulted in a reduced amount of digging as a result of such mixing from the upwind scheme. Additionally, the FCT scheme (Section 32.6) is useful for reducing digging since it effectively reduces to the positive definite upwind scheme near the bottom in order to eliminate the dispersion errors from centered advection. Unfortunately, FCT is quite expensive.

In light of these ideas, it might be sensible to employ upwind tracer advection over the level just above the bottom topography. This approach can be used in combination with another advection scheme in the interior rather than the expensive FCT scheme. Option *bottom_upwind* enables this scheme. The implementation of option *bottom_upwind* is quite simple. All that is done is to over-write the advective flux predicted from any of the other advection schemes with the flux computed from upwind. The default implementation is to employ upwind to the lowest level next to the bottom topography. Each of the three advective fluxes entering this box are computed via upwind. Modifications to this approach are straightforward.

Chapter 33

Vertical SGS options

The parameterization of processes which are not explicitly resolved by an ocean model constitute *subgridscale parameterizations*. In many ways, the area of subgridscale (SGS) parameterizations is the most challenging, and frustrating, endeavor in oceanography. Many oceanographers would wish nothing more than to ignore the issue, while others are obsessed with it. Its importance from a pragmatic modeling perspective rests largely on the crucial part such parameterizations play in determining the physical integrity of the numerical solutions. Fundamentally, "good" parameterizations often (but not always) come only after understanding the process which is to be parameterized. For a very educational and entertaining summary of certain SGS parameterization issues as of 1989, the reader is encouraged to read Holloway's lectures from the Les Houches School (Holloway, 1989). These lectures place some perspective on the SGS affair.

MOM provides a large number of SGS parameterizations. Fundamentally, the large number reflects the broad spectrum of processes present in the ocean requiring parameterization in a finite sized ocean model. Additionally, it manifests the diverse requirements within the ocean modeling community whose goals are far from uniform. Unfortunately, this state of affairs can be intimidating and confusing to many researchers wishing to use MOM. The purpose of this and the following chapters is to provide an overview of the options. The hope is to reduce the intimidation and thence to allow for a more informed choice of the options suitable for each researcher's particular needs.

The present chapter focuses on the options in MOM associated with vertical processes. Most notably, options for representing convection and parameterizing mixed layer processes are described.

33.1 Vertical convection

The hydrostatic approximation necessitates the use of a parameterization of vertical overturning processes. The original parameterization used by Bryan in the 1960's was motivated largely from ideas then used for modeling convection in stars. Recent work by Marshall and collaborators (Klinger *et al.* 1996, Marshall *et al.* 1997) have largely indicated that the basic ideas of vertical adjustment are useful for purposes of large-scale ocean circulation. As disscussed below, the Cox (1984) implementation of convective adjustment had the possibility of leaving columns unstable after completing the code's adjustment loop. Various full convective schemes have come on-line, with that from Rahmstorf implemented in MOM. An alternative to the traditional form of convective adjustment is to increase the vertical diffusion coefficient

to some large value (say $\geq 50cm^2/sec$) in order to quickly diffuse vertically unstable water columns. Indeed, it is this form which is recommended from the study of Klinger *et al.* (1996). The recently implemented KPP vertical mixing scheme effectively uses the large vertical diffusion coefficient approach in the context of a local and non-local vertical mixing scheme. This scheme computes the vertical diffusivity based on a series of physical and heuristic arguments (option discussed in Section 33.2.3).

33.1.1 Summary of the vertical convection options

In short, the handling of vertically unstable water columns in MOM can happen in one of three basic ways:

1. Implicit A: By enabling option *implicitvmix*, a large vertical diffusion coefficient (*diff_cbt_limit* set in namelist) is employed between ocean cells that are gravitationally unstable. This large coefficient severely limits the model time step, and so vertical diffusion is solved implicitly to relax the restriction (see Section 38.5 for a discussion of the implicit solution to the vertical diffusion equation).

2. Implicit B: Enabling the option *kppvmix* (discussed in Section 33.2.3) automatically employs implicit vertical diffusion, with vertical diffusivities prescribed from the KPP boundary layer scheme.

3. Explicit: Explicit convection (Section 33.1.2) is MOM's default if not enabling option *implicitvmix* or option *kppvmix*. There are two choices for explicit convection, and both explicit convective schemes can be tested in a one dimensional model (refer to Section 15.1.1 for details). The two choices are the following:

   - The recommended scheme is that from Rahmstorf (see Section 33.1.2.3). In MOM 2, this scheme was implemented with option *fullconvect*. In MOM 3, nothing needs to be enabled since this scheme is the default. All instabilities in the water column are removed on every time step.
   - Option *oldconvect* is the old style convective adjustment (Cox 1984) which takes multiple passes through the water column, alternately looking for instability on odd and even model levels (Section 33.1.2.1). When an instability is found, tracers are mixed, with their means (weighted by cell thickness) preserved. This process may induce further instability and therefore more than one pass through the water column may be needed to remove all instability. The number of passes through the water column is controlled by variable "ncon" which is input through namelist. Refer to Section 14.4 for information on namelist variables. This scheme is not recommended, and is maintained for historical reasons.

The original Cox (1984) "NCON convection scheme" has come under a lot of scrutiny. The discussion in Section 33.1.2 from Rahmstorf, and the articles by Killworth (1989) and Marotzke (1991), provide some elaboration and motivation to *not* employ the NCON scheme. It is for these reasons that all releases of MOM, starting from MOM 3, use the Rahmstorf scheme for its default explicit convection sheme. The old convection remains in MOM due to historical reasons and for purposes of comparison.

It should be noted that upon encountering a vertically unstable water column, explicit convection and the 1998 MOM 3 implementation of KPP rapidly mix only the tracers, whereas

option *implicitvmix* mixes both tracers and momentum. When momentum is not mixed, it is thought that it is simply carried along through the effects on the density field. Killworth (1989) supports this idea, so long as the purpose is large-scale ocean modeling. Basically, through the geostrophic relation, affecting density appears sufficient. Also, the vertical thermal wind shears in simulated convection regions were found by Killworth to not be too strong. Hence, mixing momentum along with density did little to affect the overall solution. These ideas, however, appear less sound for equatorial oceanography, and so the mixing of both momentum and tracers, as afforded by option *implicitvmix*, might be more important in this region.

33.1.2 Explicit convection

Explicit convection happens in one of two ways. The default is the scheme of Rahmstorf (the *fullconvect* scheme from MOM 2, now the default). Option *oldconvect* employs the older style explicit convection. In either case, when option *save_convection* is enabled, the results of explicit convection can be subsequently analyzed. Both explicit convection schemes are explained below. The following discussion is taken from Stefan Rahmstorf (rahmstorf@pik-potsdam.de), which is largely taken from "A fast and complete convection scheme for ocean models" Ocean Modeling, volume 101.

Imagine having three half-filled glasses of wine lined up in front of you. On the left a German Riesling, in the middle a French Burgundy and on the right a Chardonnay from New Zealand. Imagine further that you're not much of a connoisseur, so you want to mix the three together to a refreshing drink, with exactly the same mixture in each glass. The trouble is, you can only mix the contents of two adjacent glasses at a time. So you start off by mixing the Riesling with the Burgundy, then you mix this mixture with the Chardonnay, then... How often do you need to repeat this process until you get an identical mix in all glasses?

Incidentally, putting this question to a friend is a good test to see whether she (or he) is a mathematician or a physicist. A mathematician would answer "an infinite number of times", while a physicist would be well aware that there is only a finite number of molecules involved, so you can get your perfect drink with a finite mixing effort (only you would have no way to tell whether you've got it or not).

In any case, the number of times you need to mix is very large, and this is the problem of the standard convection scheme of the GFDL ocean model (Cox 1984), which mixes two adjacent levels of the water column if they are statically unstable. The model includes the option to repeat this mixing process a number of times at each time step, as an iteration process towards complete removal of static instabilities. The minimum number of iterations needed to mix some of the information from layer 1 down to layer n is n-1.

To avoid this problem, one needs to relax the condition that only two levels may be mixed at a time. To achieve complete mixing, a convection scheme is required that can mix the whole unstable part of the water column in one go. I have been using such a scheme back in 1983 in a one-dimensional mixing model for the Irish Sea, and I'm sure many other people have been using similar ones. Marotzke (1991) introduced such a scheme into the GFDL ocean model. It appears that it hasn't been taken up as enthusiastically as it might have been, and an implicit convection scheme (which increases the vertical diffusivity at unstable parts of the water column) has been preferred because of lower computational cost (e.g. Weaver et al, 1993). However, it is not difficult to set up a complete convection scheme which uses less computer time than the implicit scheme.

33.1.2.1 The standard Cox 1984 scheme: *oldconvect*

Since the GFDL model works the grid row by row, we'll only discuss how one grid-row is treated. Here's how:

1. Compute the densities for all grid cells in the row. Two adjacent levels are always referenced to the same pressure in order to get the static stability of this pair of levels.
2. Mix all unstable pairs.
3. Since we have now only compared and mixed "even" pairs (i.e. levels 1 & 2; levels 3 & 4; etc), repeat steps (1) and (2) for "odd" pairs (i.e. levels 2 & 3; levels 4 & 5; etc).
4. Repeat steps (1)-(3) a predetermined number of times.

There are a couple of problems here. We've already said that strictly speaking this never leads to complete mixing of an unstable water column. So the process is repeated several times at each time step to approximate complete mixing. But each time all grid cells are checked for instabilities again, even those we already found to be stable. Each density calculation requires evaluation of a third order polynomial (Cox 1972) in T and S. This is where the cpu time is eaten up.

33.1.2.2 Marotzke's scheme

1. Same as step (1) above, except that the stability of all pairs of grid cells is checked, odd and even pairs (so that the density of interior levels is computed twice, for two different reference pressures).
2. Don't mix yet: just mark all unstable pairs and find continuous regions of the water column which are unstable (neutral stability is treated as unstable).
3. Mix the unstable regions.
4. If there was instability in any column, repeat steps (1) to (3). Those columns which were completely stable in the previous round are not dealt with again in (2) and (3), but the densities are still recomputed for the entire grid row. Repeat until no more instabilities are found.

So Marotzke relaxed the condition that only two levels are mixed at a time, and complete mixing will be achieved with at most k-1 passes through the water column, if k is the number of model levels. However, if only one grid point of a row requires n iterations, the densities for the entire grid row will be recomputed n times, so it still doesn't look too good in terms of cpu efficiency.

33.1.2.3 The fast way: MOM default explicit convection

1. Compute all densities like in (1) of Marotzke.
2. Compare all density pairs to find instabilities.

   From here on, deal column by column with those grid points where an instability was found, performing the following steps:

3. Mix the uppermost unstable pair.

4. Check the next level below. If it is less dense than the mixture, mix all three. Continue incorporating more levels in this way, until a statically stable level is reached.

5. Then check the level above the newly mixed part of the water column, to see whether this has become unstable now. If so, include it in the mixed part and go back to (3). If not, search for more unstable regions below the one we just mixed, by working your way down the water column comparing pairs of levels; if you find another unstable pair, go to (3).

Note that levels which have been mixed are from then on treated as a unit. This scheme has a slightly more complicated logical structure; it needs a few more integer variables and if statements to keep track of which part of the water column we have already dealt with. The advantage is that we only recompute the densities of those levels we need; levels which are not affected by the convection process are only checked once. The scheme includes diagnostics which allow to plot the convection depth at each grid point.

33.1.2.4 Discussion

Perhaps these schemes are best discussed with an example. Imagine a model with five levels. At one grid point levels 2 & 3 and levels 3 & 4 are statically unstable. The standard scheme will, at the first pass, mix 3 & 4 and then 2 & 3. It will repeat this ncon times. Marotzke's scheme will mark the unstable pairs and then mix 2-4 in one go. It will then return to this column for a second pass and check all levels once more. My scheme will mix 2 & 3; then compare the densities of 3 & 4 and (if unstable) mix 2-4 like Marotzke's scheme. It will then recompute the density of level 4, compare levels 4 & 5 and mix 2-5 if unstable. Finally it will compare 1 & 2 again, since the density of 2 has changed in the mixing process, so level 1 might have become statically unstable. Only the density of 2 is recalculated for this.

Note that Marotzke's scheme handles the initial mixing of levels 2-4 more efficiently. Probably my scheme could be made slightly faster still by including the "marking" feature from Marotzke's scheme (the schemes were developed independently). However, in the typical convection situation only levels 1 & 2 are initially unstable, due to surface cooling. In this situation marking doesn't help. My scheme saves time by "remembering" which parts of the water column we already know to be stable, and rechecking only those levels necessary.

There is a subtlety that should be mentioned: due to the non-linear equation of state the task of removing all static instability from the water column may not have a unique solution. In the example above, mixing 2 & 3 could yield a mixture with a lower density than level 4, in spite of 3 being denser than 4, and 2 being denser than 3 originally. In this case, my scheme would only mix 2 & 3, while Marotzke's scheme would still mix 2-4. So both schemes are not strictly equivalent, though for all practical purposes they almost certainly are.

I performed some test runs with the GFDL modular ocean model (MOM) in a two-basin configuration (the same as used by Marotzke and Willebrand 1991). The model has ca. 1000 horizontal grid points and 15 levels, and was integrated for 1 year (time step 1.5 h) on a Cray YMP. Three different model states were used: (A) a state with almost no static instability, achieved by strong uniform surface heating; (B) a state with convection occurring at about 15% of all grid points; (C) a state with convection at 30% of all grid points. The latter two were near equilibrium, with permanent convection. I compared the overall cpu time consumed by these runs with different convection schemes. The standard scheme was tried for three

different numbers of iterations ncon. The results are summarized in the table; the overall cpu time is given relative to a run with no convection scheme.

Convection scheme	relative cpu time		
	A	B	C
No convection scheme	1	1	1
standard, ncon=1	1.13	1.13	1.13
standard, ncon=7	1.88	1.89	1.92
standard, ncon=10	2.25	2.27	2.32
implicit	1.52	1.52	1.52
complete	1.12	1.20	1.36

It was surprising to find that the few innocent-looking lines of model code that handle the convection consume a large percentage of the overall processing time. The numbers are probably an upper limit; a model with realistic topography and time-dependent forcing will use a bigger chunk of the cpu time for iterations in the relaxation routine for the stream function, so that the relative amount spent on convection will be lower. In my test runs, the standard scheme adds 13% cpu time per pass. My complete convection scheme used as much time as 1-3 iterations of the standard scheme, depending on the amount of convection. For zero convection it is as fast as one pass of the standard scheme, because it does the same job in this case. Additional cpu time is only used at those grid points where convection actually occurs. My scheme is considerably faster than the implicit scheme, especially for models where convection happens only at a few grid points, or only part of the time. I did not have Marotzke's scheme available for the test, but in his 1991 paper he mentions a comparison where the computation time with the implicit scheme was 60% of that with his scheme. This would give Marotzke's scheme a relative cpu time of about 2.5 in the table, with strong dependence on the amount of convective activity.

Surface heat fluxes looked identical in the runs with the implicit and complete schemes. The standard scheme showed significant deviations, however, in the surface flux as well as the convective heat flux at different depths. This is not surprising, since the rate at which heat is brought up by convection will be reduced if mixing is incomplete. The runs with ncon=7 and ncon=10 still differed noticeably from each other, and from the complete mixing case. It is possible that this could affect the deep circulation, which is driven by convective heat loss, but I didn't do long integrations to test this. The problem gets worse for longer time steps; with the standard scheme the rate of vertical mixing depends on the time step length. If acceleration techniques are used ("split time stepping", Bryan 1984), the final equilibrium could differ from one without acceleration due to this unwanted time-step dependence. Marotzke (1991) reports a case where the choice of convective scheme had a decisive influence on the deep circulation. The intention of this note is not to examine these problems any further; it is to provide an efficient alternative.

Conclusion

A convection scheme which completely removes static instability from the water column in one pass has been described, and which is much faster than the implicit scheme of the GFDL model. This scheme avoids possible problems resulting from the incomplete mixing in the standard scheme, while only using as much computer time as 1-3 iterations of the standard scheme.

33.2 Vertical SGS mixing schemes

The following options parameterize the way in which momentum and tracers are mixed vertically through parameterized subgridscale processes. One and only one of these options must be enabled.

33.2.1 constvmix

This is a basic mixing scheme that uses constant values for vertical mixing coefficients κ_m and κ_h in Equations (4.1), (4.2), (4.5), and (4.6) which amounts to using the following form for mixing coefficients at the bottom of U and T cells

$$diff_cbu_{i,k,j} = \kappa_m \tag{33.1}$$

$$diff_cbt_{i,k,j} = \kappa_h \tag{33.2}$$

If implicit vertical mixing is used by enabling option *implicitvmix* then mixing coefficients in regions of gravitational instability are set to their maximum values using

$$diff_cbt_{i,k,j} \quad = \quad diff_cbt_limit \tag{33.3}$$

Typically, the value of $diff_cbt_limit$ is set to $10^6 cm^2/sec$. Note that *visc_cbu_limit* is not used to limit $diff_cbu_{i,k,j}$ but could be. What value to use for *visc_cbu_limit* in convective regions needs further study. Diffusive fluxes at the bottom of cells take the form

$$diff_fb_{i,k,j} \quad = \quad diff_cbu_{i,k,j}(u_{i,k,j,n,\tau-1} - u_{i,k+1,j,n,\tau-1})/dzw_k \tag{33.4}$$

for momentum and

$$diff_fb_{i,k,j} \quad = \quad diff_cbt_{i,k,j}(t_{i,k,j,n,\tau-1} - t_{i,k+1,j,n,\tau-1})/dzw_k \tag{33.5}$$

for tracers. The mixing coefficients κ_m and κ_h are independent of time and are input through namelist. Refer to Section 14.4 for information on namelist variables.

33.2.2 bryan_lewis_vertical

This is a hybrid mixing scheme in the sense that it affects only tracers. It was introduced by (Bryan and Lewis 1979) and is used in many climate models as a background diffusivity. It specifies the vertical tracer diffusivity κ_h as a time-independent depth dependent function given by

$$diff_cbt_{i,k,j} = afkph + \frac{dfkph}{\pi}\arctan(sfkph \cdot (zw_k - zfkph)). \tag{33.6}$$

Taking $afkph = 0.8\ cm^2/sec$, $dfkph = 1.05\ cm^2/sec$, $sfkph = 4.5\text{x}10^{-5}\ cm^{-1}$, and $zfkph = 2500.0\text{x}10^2\ cm$ implies a diffusivity coefficient ranging from 0.3 cm^2/sec near the surface of the ocean to 1.3 cm^2/sec near the bottom. Alternatively, taking $afkph = 0.69\ cm^2/sec$, $dfkph = 1.25\ cm^2/sec$, $sfkph = 4.5\text{x}10^{-5}\ cm^{-1}$, and $zfkph = 2500.0\text{x}10^2\ cm$ implies a diffusivity coefficient ranging from 0.1 cm^2/sec near the surface of the ocean to 1.3 cm^2/sec near the bottom. Values required for this option can be changed through namelist. Refer to Section 14.4 for information on namelist variables.

33.2.3 kppvmix

This section was contributed from the NCAR documentation by William Large (*wily@ncar.ucar.edu*) and transcribed to LaTeX(with apologies from me to Wily). Equation numbers are retained to match the original but some option names have been changed to match those in MOM. When time permits, the names of some variables and indices will also be changed to be consistent with their use throughout the rest of this manual. Here are some considerations to keep in mind. The Richardson number is computed as described in Section 33.2.4.1. All vertical mixing schemes use this discretization for the Richardson number. If the Prandtl number is not one and more than two tracers are used (i.e. $nt > 2$) then the vertical diffusion coefficient for salinity is used for all subsequent passive tracers. This scheme may not work well without option *gent_mcwilliams* due to excessive deepening of isopycnals when penetrative convection is active. This scheme may also be exercised in a 1-D framework. Refer to Section 15.1.6. What follows is the transcribed NCAR documentation.

Option *kppvmix* enables the KPP Boundary Layer Mixing Scheme of Large, McWilliams, and Doney (1994) which is based on an adaptation of the nonlocal K-profile parameterization of Troen and Mahrt (1986) for use as an oceanic boundary layer model. Important characteristics of both applications are that they are consistent with similarity theory in the surface layer, the boundary layer is capable of penetrating the interior stratification, and turbulent transport vanishes at the surface. The OBL (ocean boundary layer) model also has additional desirable features. For example, turbulent shear contributes to the diagnosed boundary layer depth, so as to make the entrainment of buoyancy at the base of the OBL independent of the interior stratification. Interior mixing at the base of the boundary layer ($d = h$) influences the turbulence throughout the boundary layer. Also, in the convective limit the turbulent velocity scales for both momentum and scalars become directly proportional to w^*. The problem of determining the vertical turbulent fluxes of momentum and both active and passive scalars throughout the OBL is closed by adding a nonlocal transport term γ_x:

$$\overline{wx}(d) = -K_x(\partial_z X - \gamma_x) \; . \tag{G1}$$

In practice the external forcing is first prescribed, then the boundary layer depth, h, is determined, and finally profiles of the diffusivity and nonlocal transport are computed. A complete description of this model can be found in Large et al. (1994), and what follows is based on Appendix D of that paper. The KPP scheme is activated in the model by specifying the ifdef option *kppvmix*. The ifdef option *implicitvmix* must also be specified, because the vertical mixing is done implicitly. Specifying this ifdef option also eliminates the explicit convective adjustment scheme. Other vertical mixing paramaterizations such as option *constvmix* or *ppvmix* must not be specified when using *kppvmix*. There are an additional three sub-options that can be specified with *kppvmix*: option *kmixcheckekmo* does an additional check against the Ekman and Monin-Obukhov length scales; option *kmixnori* sets the vertical viscosity and diffusivity below the boundary layer to the constant background values, *fkpm* and *fkph* which are not dependent on the local Richardson number; and option *kmixdd* adds a double diffusion contribution is added to the vertical diffusivity, see Large et al. (1994).

33.2.3.1 Vertical discretization

Layer grid points are denoted by whole number indices, whereas layer interfaces are denoted by half number indices. The layer thicknesses are $\Delta_n = d_{n+0.5} - d_{n-0.5}$, and the distances between grid levels are $\Delta_{n+0.5} = d_{n+1} - d_n$. To define the latter for $n = 0$ and $n = M$, where M is

the total number of layers, d_0 is taken to be zero and a fictitious layer, $M+1$, of zero thickness is added at the bottom. An index k can always be found such that $d_{k-1} \leq h < d_k$. A useful variable that varies from 0 to 1 over this grid interval is

$$\delta = (h - d_{k-1})/\Delta_{k-.5} \tag{G2}$$

Property values are defined at the grid levels, d_n, for $1 \leq n \leq M+1$, where the $M+1$ values are prescribed bottom boundary conditions. In computing near surface reference values, X_r, as the average value between the surface and ϵh, the following continuous profile is assumed:

$$\begin{aligned} X(d) &= X_1 \ , \qquad d < d_1 \\ &= X_{n-1} + \frac{X_n - X_{n-1}}{d_n - d_{n-1}}(d - d_{n-1}) \ , \ \ d_1 \leq d_{n-1} \leq d < d_n \leq D \end{aligned} \tag{G3}$$

Property gradients are required both at interfaces and grid levels, and in the model where partial derivatives with respect to d equal $-\partial_z$ these are computed, respectively, as

$$\begin{aligned} [\partial_z X]_{n+.5} &= \frac{X_n - X_{n+1}}{\Delta_{n+.5}}, \ 1 \leq n \leq M \\ [\partial_z X]_n &= \frac{X_{n-1} - X_{n+1}}{\Delta_{n-.5} + \Delta_{n+.5}} \ , \ 1 < n \leq M \\ &= \frac{X_1 - X_2}{2\Delta_{1.5}} \ , \ n = 1 \end{aligned} \tag{G4}$$

Interior—boundary layer matching

The convective and wind deepening cases are handled well, but not ideally by a low vertical resolution KPP model. Several techniques were explored in an attempt to dampen oscillations and reduce the bias in h. One approach is to use nonlinear interpolation. Several schemes were tested and they could be made to work very well with idealized profiles. However, in general the property profiles, and especially the bulk Richardson number profile, are highly variable and no one scheme could be found to work well over a wide variety of naturally occurring conditions.

The following practical scheme is used to remove the bias and dampen oscillations. It involves enhancing the diffusivity at the $k-.5$ interface. Figure G1 shows the diffusivities produced by the model's parameterizations in the vicinity of layer k for two situations: (a) $d_{k-1} < h < d_{k-.5}$ (left panel) and (b) $d_{k-.5} < h < d_k$ (right panel). The first step in the matching process is to determine the interior diffusivities (triangles). These are interpolated (dotted line) to give $\nu_x(h)$ and the gradient, $\partial_z \nu_x(h)$, at h. The only important requirement here is that there not be discontinuities in either quantity as h progresses through the grid. This is true of the simple scheme used in the model:

$$\begin{aligned} \nu_x(h) &= \nu_x(d_{n+.5}) + \partial_z \nu_x(h)(d_{n+.5} - h) \ , \ d_{n-.5} < h \leq d_{n+.5} \\ \partial_z \nu_x(h) &= (1-R) \ \left[\nu_x(d_{n-.5}) - \nu_x(d_{n+.5})\overline{\Delta_n}\right] \\ &+ \ R \ \left[\nu_x(d_{n+.5}) - \nu_x(d_{n+1.5})\overline{\Delta_{n+1}}\right] \ , \end{aligned} \tag{G5}$$

with $n = k - 1$ in situation (a) and $n = k$ in situation (b). The interpolated gradient is just a weighted average of two discrete gradients, with the weight $R = (h - d_{n-.5})\Delta_n^{-1}$. The continuous boundary layer diffusivity profile is the solid line in Fig. G1. The dotted line in Fig. G1 follows the $\nu_x(h)$ that would be computed from (G5) for different values of h between $d_{k-1.5}$ and $d_{k+.5}$.

The next step is to compute a modified diffusivity, Λ_x (cross in Fig. G1), which is to be applied at the $k - .5$ interface. It is a weighted average of $\nu_x(d_{k-.5})$ (triangle in Fig. G1) and an enhanced boundary layer diffusivity, K_x^*, such that for case (b) situations

$$\begin{aligned} \Lambda_x &= (1-\delta)\ \nu_x(d_{k-.5}) + \delta\ K_x^* \ , \\ K_x^* &= (1-\delta)^2\ K(d_{k-1}) + \delta^2\ K(d_{k-.5}) \ , \end{aligned} \tag{G6}$$

where δ, as defined in (G2), varies from 0 to 1. For case (a), $\nu_x(d_{k-.5})$ replaces $K(d_{k-.5})$ in (G6), because the latter is not defined for $d > h$. The important feature is that the dependency on $K(d_{k-1})$ (square in Fig. G1) leads to an enhanced diffusivity at the $k - .5$ interface as soon as h becomes greater than d_{k-1}. The increased deepening that results greatly reduces the boundary layer depth bias of low, relative to high, resolution simulations.

Figure G2 compares the diffusivity, Λ_x, used by the model to the boundary layer diffusivity at $d_{k-.5} = 15m$, as δ varies from 0 to 1. The latter are kept small by using a small $u^* = 0.006ms^{-1}$ and by using the grid of Fig. G1 with $k = 4$, so that h does not get very large. Two extreme cases are shown: no interior mixing (dashed versus dot-dashed lines) and substantial interior mixing (solid versus dotted lines). In the latter case, the interior diffusivities are $1, 2, 4$, and $8 \times 10^{-4} m^2/s$ at $25, 20, 15$, and $10m$ depth, respectively. The boundary layer diffusivities at the $k - .5$ interface in the range $0 \leq \delta \leq 0.5$ are constant at the interior value. The importance of using Λ_x is to enhance these diffusivities in this range of δ, when the interior diffusivity is small. When there is substantial interior mixing as in the cases shown in Figs. G1 and G2, the enhancement is not necessary and is much reduced. As δ approaches 1, the form of (G6) makes Λ_x less than $K_x(d_{k-.5})$ in an attempt to reduce biases.

33.2.3.2 Semi-implicit time integration

The prognostic equations of the model are solved by a semi–implicit integration scheme whose general matrix form is:

$$\mathbf{A}_x^i\ \mathbf{X}_{t+1}^{i+1} = \mathbf{X}_t + \mathbf{H}_x^i \ , \tag{G7}$$

where $\mathbf{A}$ is an M by M tridiagonal matrix, and $\mathbf{X}$ and $\mathbf{H}$ are vectors of length M. Integration over a time step, Δt, is accomplished by inversion of $\mathbf{A}$, which allows the properties at the new time $t + 1$ to be computed from past values at t. The integration can only be semi-implicit, because $\mathbf{A}$ and $\mathbf{H}$ depend on quantities like the diffusivity that depend on h, which in turn depend on the profiles $\mathbf{X}_{t+1}$ that are themselves computed from $\mathbf{A}$ and $\mathbf{H}$. The superscripts in (G7) denote various choices of when and how $\mathbf{A}$ and $\mathbf{H}$ are calculated. The simplest method, denoted by $i = 0$, would be to compute all the required quantities, including the forcing, at time t using $\mathbf{X}_t$ values. In some numerical schemes the prognostic variables are updated by advection prior to vertical diffusion, so that these updated $\mathbf{X}_{t+1}^0$ values could also be used.

Equation (G7) has been iterated until the new boundary layer depth from the ith iteration, h^{i+1}, differs from the previous h^i by less than a specified tolerance, η_h :

$$\frac{|h^i - h^{i+1}|}{\Delta_k} < \eta_h \ , \tag{G8}$$

where again $d_{k-1} \leq h^{i+1} \leq d_k$, defines the vertical index, k, and Δ_k is the local vertical resolution. Iteration allows the $\mathbf{X}^i_{t+1}$ values used to determine $\mathbf{A}^i_x$ and $\mathbf{H}^i_x$ to be close to the final $\mathbf{X}^{i+1}_{t+1}$ values. Either $\mathbf{X}_t$ or $\mathbf{X}^0_{t+1}$ from above could be used for the first iteration, $i = 1$. Alternatively, linear extrapolations of $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$ can be used to give the first iteration $\mathbf{X}^1_{t+1}$ values. Since it is undesirable to use the extrapolated values in the final iteration, at least two iterations are always performed. Over an annual cycle at OWS Papa less than 1% of all time integrations required more than two iterations. At most 12 iterations were needed, but 0.3 percent of all iterations (when the stratification was weak) failed to converge and the results of the twentieth iteration were used.

The elements of $\mathbf{A}$ have the same form for all properties, so the subscript x can be dropped for convenience. The elements of $\mathbf{A}$ then become $A_{n,m}$, where n is the row index and m the column. The non–zero elements are:

$$\begin{aligned}
A_{1,1} &= (1 + \Omega^+_1) \\
A_{n,n-1} &= -\Omega^-_n & 2 \leq n \leq M \\
A_{n,n} &= (1 + \Omega^-_n + \Omega^+_n) & 2 \leq n \leq M \\
A_{n,n+1} &= -\Omega^+_n & 1 \leq n \leq M-1 \ ,
\end{aligned} \tag{G9}$$

where

$$\Omega^-_n = \frac{\Delta t}{\Delta_n} \frac{K_x(d_{n-.5})}{\Delta_{n-.5}} \text{ and } \Omega^+_n = \frac{\Delta t}{\Delta_n} \frac{K_x(d_{n+.5})}{\Delta_{n+.5}}. \tag{G10}$$

The vector $\mathbf{H}$ is very different for scalars than for velocity. In general, the scalar $\mathbf{H}$ includes the boundary conditions, countergradient terms, and non-turbulent forcing. Let J be the non-turbulent forcing at the interfaces, $Q_n(d)$ for temperature and $-F_n(d)$ for salinity, with J_o the surface value. For a surface turbulent flux of $\overline{ws}_o$, the elements of $\mathbf{H}_s$ are :

$$\begin{aligned}
H_1 &= \frac{\Delta t}{\Delta_1} \left((K_s \, \gamma_s)_{1.5} - \overline{ws}_o + J_{1.5} - J_o \right) \\
H_n &= \frac{\Delta t}{\Delta_n} \left((K_s \, \gamma_s)_{n+.5} - (K_s \, \gamma_s)_{n-.5} + J_{n+.5} - J_{n-.5} \right) \qquad 2 \leq n \leq M-1 \\
H_M &= \frac{\Delta t}{\Delta_M} \left((K_s \, \gamma_s)_{M+.5} - (K_s \, \gamma_s)_{M-.5} + J_{M+.5} \right. \\
&- \left. J_{M-.5} + S_{M+1} \frac{K_s(d_{M+.5})}{\Delta_{M+.5}} \right) ,
\end{aligned} \tag{G11}$$

where S_{M+1} is the prescribed bottom boundary value of the scalar and subscripts $n+.5$ and $n-.5$ denote evaluation at $d_{n-.5}$ and $d_{n+.5}$, respectively. For velocity components, **H** includes only boundary conditions and Coriolis terms. For the zonal velocity component the elements of $\mathbf{H}_u$ are:

$$\begin{aligned} H_1 &= \Delta t\, f\, V_1^{t+.5} - \frac{\Delta t}{\Delta_1}\,\overline{wu}_o \\ H_n &= \Delta t\, f\, V_n^{t+.5} \qquad\qquad 2 \le n \le M-1 \\ H_M &= \Delta t\, f\, V_M^{t+.5} + \frac{\Delta t}{\Delta_M}\frac{U_{M+1}\, K_m(d_{M+.5})}{\Delta_{M+.5}}\,. \end{aligned} \tag{G12}$$

For the meridional velocity component the elements of $\mathbf{H}_v$ are:

$$\begin{aligned} H_1 &= -\Delta t\, f\, U_1^{t+.5} - \frac{\Delta t}{\Delta_1}\,\overline{wv}_o \\ H_n &= -\Delta t\, f\, U_n^{t+.5} \qquad\qquad 2 \le n \le M-1 \\ H_M &= -\Delta t\, f\, U_M^{t+.5} + \frac{\Delta t}{\Delta_M}\frac{V_{M+1}\, K_m(d_{M+.5})}{\Delta_{M+.5}}\,. \end{aligned} \tag{G13}$$

In (G12) and (G13) the fixed bottom boundary conditions are U_{M+1} and V_{M+1}. The Coriolis terms are estimates of their average value over the time step:

$$\begin{aligned} U_n^{t+.5} &= 0.5\,(U_n^t + U_n^i) \\ V_n^{t+.5} &= 0.5\,(V_n^t + V_n^i)\,, \end{aligned} \tag{G14}$$

where the layer n velocity components at time t are U_n^t and V_n^t.

Note: the following figures have not been received yet from NCAR.

Figure G1. Schematic of the diffusivities required to match the interior and boundary layer mixing for two cases: (a) h between d_{k-1} and $d_{k-1/2}$, and (b) h between $d_{k-1/2}$ and d_k. Shown are the discrete interior diffusivities (triangles) and their interpolation (dashed curve) from (G6); the continuous boundary layer diffusivity profile $K_x(d)$ (solid trace), including its value at d_{k-1} (square); and finally the modified diffusivity Λ_x at $d_{k-1/2}$, which is used by the model (cross).

Figure G2. Comparison of Λ_x used by the model and $K_x(d_{k-0.5})$ as δ varies from 0 to 1. Shown are a case of no interior mixing (Λ_x, dashed trace; K_x, dot-dashed trace) and a case with substantial interior mixing (Λ_x, solid trace; K_x, dotted trace).

33.2.3.3 Diagnostic output

The boundary layer depth "hbl" is added to the snapshots file when option -Dnetcdf is used.

33.2.4 ppvmix

This is a basic vertical mixing scheme which calculates Richardson dependent values of mixing coefficients κ_m and κ_h based on the formulation given in Pacanowski and Philander (1981). This option may be exercised in a 1-D framework. Refer to Section 15.1.6.

33.2.4.1 Richardson number

In previous discretizations of this scheme, the Richardson number was first computed at the base of U-cells, then averaged onto T-cells after which the three dimensional viscosity coefficient $visc_cbu_{i,k,j}$ was computed using Richardson numbers at the base of U-cells and the three dimensional diffusivity coefficient $diff_cbt_{i,k,j}$ was computed using Richardson numbers at the base of T-cells.

Actually, the above is only one of many possible ways to discretize the scheme. Instead of averaging Richardson numbers, another way is to compute Richardson numbers, viscosity, and diffusivity coefficients at the base of U-cells and then average the diffusivity coefficients onto T-cells. A third way is to compute separate Richardson numbers at the base of T-cells and U-cells separately after which the mixing coefficients are then computed.

Two other approaches are as follows: The first involves computing Richardson numbers at the base of T-cells, averaging to the base of U-cells, then computing mixing coefficients at the base of T-cells and U-cells. The second involves computing Richardson numbers at the base of T-cells, computing both mixing coefficients at the base of T-cells, and then averaging the viscosity coefficients onto U-cells.

Based on a combination of analytic and numerical explorations at GFDL by Anand Gnanadesikan, the last way turns out to be the most accurate of the above five schemes. This last scheme was used in an intermediate version[1] of MOM 2 for about 8 months. However, a variety of three dimensional simulations indicated that a weak instability exists which under certain conditions manifests as a two grid point structure (numerical noise) in the tracer fields. Based on these results, a change was made back to the original scheme which is currently being used and is described below. All vertical mixing schemes compute the Richardson number the following way.

The Richardson number is first computed at the base of U-cells using

$$riu_{i,k,j} = \frac{-grav \cdot \overline{\delta_z(\rho_{i,k,j,\tau-1})}^{\lambda\phi}}{(\delta_z(u_{i,k,j,1,\tau-1}))^2 + (\delta_z(u_{i,k,j,2,\tau-1}))^2} \qquad (k < kmu_{i,jrow}) \tag{33.7}$$

$$= 0 \qquad (k \geq kmu_{i,jrow}) \tag{33.8}$$

where $grav = 980.6 cm/sec^2$ and the local index j is related to the global index $jrow$ by the memory window offset given in Equation 11.4. Note that, because of the averaging in latitude and longitude, $riu_{i,k,j}$ is defined at the base of U-cells and can only be computed for rows 1 through $jmw - 1$ in the memory window (Refer to Section 22.2.4 for a discussion). Note also that $riu_{i,k,j}$ is zero on the top and bottom faces of land cells. The Richardson number at the base of U-cells is then averaged onto the base of T-cells by

$$rit_{i,k,j} = \overline{riu_{i-1,k,j-1}}^{\lambda\phi} \tag{33.9}$$

[1] It was in MOM 2.0 version 2.0 (beta).

However, with the aid of masking (not shown) only non-zero values of *riu* are considered in the average. This prevents generating low values of *rit* next to topography for the wrong reason[2]. Again, because of the averaging in latitude and longitude, $rit_{i,k,j}$ can only be computed for rows 2 through $jmw - 1$ in the memory window.

33.2.4.2 Vertical mixing coefficients

After computing the Richardson number, both vertical mixing coefficients are computed at the bases of their respective cells for rows 2 through $jmw - 1$ in the memory window

$$diff_cbt_{i,k,j} = \frac{fricmax}{(1 + 5 \cdot rit_{i,k,j})^3} + diff_cbt_back \tag{33.10}$$

$$visc_cbt_{i,k,j} = \frac{fricmax}{(1 + 5 \cdot riu_{i,k,j})^2} + visc_cbu_back \tag{33.11}$$

In regions of vertical instability, the mixing coefficients are set to their limiting values.

$$visc_cbt_{i,k,j} = visc_cbu_limit \;\; (for \;\; rit_{i,k,j} < 0) \tag{33.12}$$

$$diff_cbt_{i,k,j} = diff_cbt_limit \;\; (for \;\; riu_{i,k,j} < 0) \tag{33.13}$$

Typically, $diff_cbt_limit$ is set to $10^6 cm^2/sec$ when option *implicitvmix* is enabled. Otherwise it is set to $fricmax$. The value of $visc_cbu_limit$ is set to $fricmax$ regardless of whether option *implicitvmix* is enabled or not.
To account for the effect of high frequency wind mixing near the surface (which is absent in climatological monthly mean wind stress), the mixing coefficients at the base of level one are taken as the maximum of the predicted mixing coefficients and parameter *windmix*.

$$diff_cbt_{i,k=1,j} = max(windmix, diff_cbt_{i,k=1,j}) \tag{33.14}$$

$$visc_cbu_{i,k=1,j} = max(windmix, visc_cbu_{i,k=1,j}) \tag{33.15}$$

33.2.4.3 Adjustable parameters

A reasonable range for the background diffusion coefficient *diff_cbt_back* is from molecular values of 0.00134 cm^2/sec to bulk values of about 0.1 cm^2/sec. For background viscosity coefficient *visc_cbu_back*, a reasonable range is from molecular values of 0.0134 cm^2/sec to bulk values of about 1.0 cm^2/sec. Based on Pacanowski and Philander (1981), reasonable values for the maximum mixing coefficient *fricmax* range from about 50 cm^2/sec to 100 cm^2/sec. The Prandtl number is about 10 for stable regions and 1 for regions of strong mixing.

Choosing *diff_cbt_back* too large tends to erode the thermocline away. Values of *visc_cbu_back* occur in regions of high Richardson number and have an affect on limiting the speed of the Equatorial Undercurrent: lower (higher) values resulting in faster (slower) speeds. To work

[2] If a simple four point average is used, cases arise where values of *rit* are generated near the bottom which are much lower than any surrounding *riu* in the area. This is an artificial result which can lead to a situation with high vertical diffusion amidst low vertical viscosity. Martin Schmidt (personal communication) has observed the vertical density profile being unrealically eroded away in shallow regions when zero values of *riu* are used in the average.

reasonably well, this scheme needs about 10 meter or finer resolution in the vertical. It may not do well off the equator if shortwave penetration isn't taken into account because vertical shear may not be enough to overcome buoyancy when all the heat flux is absorbed within the first vertical cell. In simulations, maximum values for mixing coefficients occur in regions of low Richardson number such as surface mixed layers. Philander speculates that *fricmax* should be a function of windspeed and this is being explored. The *windmix* parameter is arbitrary and is meant to simulate the high frequency wind bursts that are absent (due to the averaging process) from climatological forcing. Typically, *windmix* has been set at 15 cm^2/sec.

Various parameters for this option may be changed through namelist. Refer to Section 14.4 for information on namelist variables.

33.2.5 tcvmix

This section was contributed by Tony Rosati (*ar@gfdl.gov*). This is a basic scheme which supplies values for vertical mixing coefficients κ_m and κ_h based on the second order turbulence closure scheme of Mellor and Yamada level 2.5 as given in Rosati and Miyakoda (1988). The scheme is being implemented by Tony Rosati but is not yet ready. Questions should be directed to Tony (ar@gfdl.gov).

Chapter 34

Horizontal SGS options

The purpose of this chapter is to document the options in MOM which parameterize horizontal mixing of momentum and tracers. The mixing tensor in this chapter is aligned parallel to the geopotential. Chapter 35 discusses options which orient the tracer mixing tensor according to locally defined neutral directions.

34.1 Summary of the options

The options for horizontal mixing can be broken into two main categories: Laplacian and biharmonic. Laplacian mixing is the traditional approach which involves adding a second order operator to the momentum and/or tracer equations. The biharmonic options replace the second order operator with a fourth order operator. The central motivation of this replacement is to enhance the scale selectivity of the mixing.

Each of the Laplacian and biharmonic options can be applied either together or separately to the velocity and tracers, and each can employ a constant or nonconstant viscosity/diffusivity. Because there are a number of options and sub-options, the names have been designed to provide a clear description of what each affects. The options are summarized in the following.

34.1.1 Horizontal tracer mixing options

There are two main options for specifying the horizontal diffusion of the tracer fields: options *tracer_horz_laplacian* and *tracer_horz_biharmonic*. One and only one must be specified. Each has various sub-options as well.

1. Option *tracer_horz_laplacian* enables the horizontal Laplacian mixing of tracers using the following mixing operator

$$R_L(T) = \nabla_h \cdot (A_h \nabla_h T), \tag{34.1}$$

   where A_h is the horizontal Laplacian diffusivity. There are three ways in which A_h can be specified. Only one can be specified at a time, and one *must* be specified.

   (a) Option *tracer_horz_mix_const* sets A_h to a spatial and temporal constant value determined from namelist.

(b) Option *bryan_lewis_horizontal* (Section 34.5) specifies A_h according to Bryan and Lewis (1979). In this case, the horizontal diffusivity is a function of depth.

(c) Option *tracer_horz_mix_var* allows for A_h to be an arbitrary three-dimensional function. There are one and a "half" sub-options which are available with options *tracer_horz_laplacian* and *tracer_horz_mix_var*, only one of which need be specified:

i. Option *tracer_horz_mix_smag* (Section 34.7) specifies B_h as a space-time dependent function according to the Smagorinsky (1963) scheme. If option *tracer_horz_mix_smag* is enabled, then option *tracer_horz_mix_var* is also automatically enabled. Since most ocean model applications of the Smagorinsky scheme are for determining the horizontal viscosity, not the horizontal diffusivity, then option *velocity_horz_mix_smag* (see below) must also be enabled if one wishes to use option *tracer_horz_mix_smag*. Additionally, if option *isoneutralmix* is enabled (Chapter 35), then the Smagorinsky diffusivity is not used for the tracers, and so option *tracer_horz_mix_smag* is ignored. The reason is that with option *isoneutralmix*, it is assumed that one does not wish to use horizontal diffusion, or at most one wishes to use some constant background horizontal diffusivity which is entered through namelist.

ii. The "half" sub-option is to simply provide a profile in the USER INPUT section of the code where Smagorinsky is computed.

Again, one and only one of the three options *tracer_horz_mix_const*, *tracer_horz_mix_var*, or *bryan_lewis_horizontal* for A_h needs to be specified in order for the model to run. Even if the experiment involves one of the isoneutral mixing schemes (Chapter 35), with $A_h = 0$, then option *tracer_horz_mix_const* needs to be enabled.

2. Option *tracer_horz_biharmonic* enables the horizontal biharmonic mixing of tracers using the following iterated mixing operator

$$R_B(T) = -\nabla_h \cdot (\nabla_h R_L(T)) \tag{34.2}$$

$$R_L(T) = \nabla_h \cdot (B_h \nabla_h T) \tag{34.3}$$

where $B_h > 0$ is the horizontal biharmonic diffusivity. There are two ways in which B_h can be specified:

(a) Option *tracer_horz_mix_const* sets B_h to a spatial and temporal constant determined from namelist.

(b) Option *tracer_horz_mix_var* (Section 34.6) allows for B_h to be an arbitrary three-dimensional function. There are one and a "half" suboptions which are available with *tracer_horz_biharmonic* and *tracer_horz_mix_var*:

i. Option *tracer_horz_mix_smag* (Section 34.7) specifies B_h as a space-time dependent function according to the Smagorinsky (1963) scheme. If option *tracer_horz_mix_smag* is enabled, then option *tracer_horz_mix_var* is also automatically enabled. Since most ocean model applications of the Smagorinsky scheme are for determining the horizontal viscosity, not the horizontal diffusivity, then option *velocity_horz_mix_smag* (see below) must also be enabled if one wishes to use option *tracer_horz_mix_smag*. Additionally, if option *isoneutralmix* is enabled (Chapter 35), then the Smagorinsky diffusivity is not used for the tracers, and so option *tracer_horz_mix_smag* is ignored. The reason is that with option *isoneutralmix*, it is assumed that one does not wish to use horizontal diffusion, or at

most one wishes to use some constant background horizontal diffusivity which is entered through namelist.

ii. The "half" sub-option is to simply provide a favorite profile in the USER INPUT section of the code where Smagorinsky is computed.

34.1.2 Horizontal velocity mixing options

There are two main options for specifying the horizontal viscous dissipation of the zonal and meridional velocity fields: options *velocity_horz_laplacian* and *velocity_horz_biharmonic*. One and only one must be specified. Each has various sub-options as well.

1. Option *velocity_horz_laplacian* enables the horizontal Laplacian mixing of velocity using the following friction vector

$$F^a = \tau^{ab}_{;b} \tag{34.4}$$

where $a, b = 1, 2$, the semi-colon symbolizes a covariant derivative, and τ^{ab} is a symmetric and trace-free stress-tensor. This method is the MOM default, and it is automatically enabled unless *velocity_horz_biharmonic* is enabled. Chapter 9 provides complete details regarding the form of the friction vector relevant for arbitrary coordinates and how it is discretized with MOM's spherical coordinates. For the present purposes, it is sufficient to quote the Cartesian form, in which the stress tensor is

$$\tau^{ab} = A_m \begin{pmatrix} u_x - v_y & u_y + v_x \\ u_y + v_x & v_y - u_x \end{pmatrix}, \tag{34.5}$$

where A_m is a Laplacian momentum viscosity. The Cartesian form of the covariant derivative is just the familiar partial derivative, and so the zonal and meridional friction components are

$$F^u = \nabla_h \cdot (A_m \nabla_h u) + \hat{z} \cdot \nabla_h v \wedge \nabla_h A_m \tag{34.6}$$

$$F^v = \nabla_h \cdot (A_m \nabla_h v) - \hat{z} \cdot \nabla_h u \wedge \nabla_h A_m. \tag{34.7}$$

The cross-product terms are necessary to provide a proper angular momentum budget when using nonconstant viscosities (Section 9.3.9).

There are two ways in which the viscosity A_m can be specified:

(a) Option *velocity_horz_mix_const* sets A_m to a spatial and temporal constant value determined from namelist.

(b) Option *velocity_horz_mix_var* (Section 34.6) allows for A_m to be an arbitrary spatially nonconstant function. There are four and a "half" sub-options for *velocity_horz_mix_var*:

i. Option *velocity_horz_mix_smag* (Section 34.7) specifies A_m as a space-time dependent function according to the Smagorinsky (1963) scheme. If option *velocity_horz_mix_smag* is enabled, then option *velocity_horz_mix_var* is automatically enabled.

ii. Option *am_cosine* sets $A_m = A_o \cos\phi$, where A_o is the value taken from namelist.

iii. Option *am_cosinep1* sets $A_m = A_o(\cos\phi + 1)$, where A_o is the value taken from namelist. This option has been used in the Hadley Centre's Unified Model.

iv. Option *am_taper_highlats* tapers the viscosity to a small values in the high latitudes.

v. The "half" sub-option is to provide one's favorite profiled in the USER INPUT section of the code where the Smagorinsky coefficients are computed.

One of the two options *velocity_horz_mix_const* and *velocity_horz_mix_var* needs to be specified in order for the model to run with option *velocity_horz_laplacian*.

2. Option *velocity_horz_biharmonic* enables the horizontal biharmonic dissipation of the zonal and meridional velocity components using an iterative approach. This approach is discussed in Section 9.5. As with the *velocity_horz_laplacian* option, the biharmonic viscosity B_m can be determined through either one of the following ways:

   (a) Option *velocity_horz_mix_const* sets B_m to a spatial and temporal constant value determined from namelist.

   (b) Option *velocity_horz_mix_var* (Section 34.6) allows for B_m to be an arbitrary function. There are three and a "half" sub-options for *velocity_horz_mix_var*:

      i. Option *velocity_horz_mix_smag* (Section 34.7) specifies B_m as a space-time dependent function according to the Smagorinsky (1963) scheme. If option *velocity_horz_mix_smag*, then option *velocity_horz_mix_var* is automatically enabled.

      ii. Option *am_cosine* sets $B_m = B_o \cos^3 \phi$, where B_o is the value taken from namelist.

      iii. Option *am_cosinep1* sets $B_m = B_o (\cos \phi + 1)^3$, where B_o is the value taken from namelist.

      iv. Option *am_taper_highlats* tapers the viscosity to a small values in the high latitudes.

      v. The "half" sub-option is to provide one's favorite profiled in the USER INPUT section of the code where the Smagorinsky coefficients are computed.

   One of the two options *velocity_horz_mix_const* and *velocity_horz_mix_var* needs to be specified in order for the model to run with option *velocity_horz_biharmonic*.

The remaining sections of this chapter provide some guidance for choosing a particular diffusivity and/or viscosity scheme. Details of the discretizations are provided thereafter.

34.2 Some numerical constraints

When deciding on values to be used for the various mixing coefficients, it is useful to know the numerical constraints which bound their sizes. The purpose of this section is to review some of the basic considerations.

34.2.1 Balance between advection and diffusion

The representation of advection in a numerical model is nontrivial. In MOM, most concern has gone into providing some options for advecting tracers (Chapter 32). Advection of momentum is still done with centered difference scheme used by Bryan (1969). As discussed there, centered advection of momentum provides for an energetically conservative scheme, which in turn eliminates Phillip's (1959) nonlinear instability. The introduction of viscous friction eliminates the energetically conservative nature of the model. However, having the underlying

dynamical core energetically conservative has been a very important element in diagnosing problems with various new physical parameterizations.

The purpose of this section is to summarize the issues concerning the use of centered differences. Most notably, it will be shown that some amount of mixing is needed in order to eliminate a computational mode. The arguments are taken from Appendix B of Bryan, Manabe, and Pacanowski (1975).

Consider the steady state balance between advection and diffusion

$$U\,\psi_x = A\,\psi_{xx}, \tag{34.8}$$

where U is a constant advection velocity, ψ is any field which is advected, including velocity and tracers, and A is a Laplacian viscosity or diffusivity. Using centered differences in space on a constant grid of size Δ, the finite difference counterpart to this equation takes the form

$$(R-2)\,\psi_{i+1} + 4\,\psi_i - (R+2)\,\psi_{i-1} = 0, \tag{34.9}$$

where

$$R = \frac{U\,\Delta}{A}. \tag{34.10}$$

When ψ is one of the velocity components, R is the grid Reynolds number, where the qualifier "grid" is used since the length scale is the grid scale. When ψ is a tracer, R is the grid Peclet number. In either case, R measures the ratio of advection to diffusion.

A constant solves the finite difference equation (34.9). In addition, a power solves it

$$\psi_i = C\,\xi^i, \tag{34.11}$$

where C is a constant and the i superscript on the right-hand side represents a power. Substituting this function into the finite difference equation yields the quadratic equation

$$(R-2)\,\xi^2 + 4\,\xi - (R+2) = 0. \tag{34.12}$$

The two real roots to this equation are

$$\xi = 1 \tag{34.13}$$

$$\xi = \frac{2+R}{2-R}. \tag{34.14}$$

The first root is the constant which has already been mentioned. The second root is the most relevant. With $\psi_i = C\,\xi^i$, if ξ^i is negative, then ψ_i will oscillate in space with a wave length given by the grid size. Such behaviour is not physical and so should be avoided. Ensuring that $R < 2$, or

$$\frac{U\,\Delta}{A} < 2 \tag{34.15}$$

keeps the second root positive and so eliminates the unphysical behaviour. Therefore, with centered difference discretization of advection, it is necessary to also include a nonzero amount of viscosity or diffusivity.

For the case of tracers, various discretizations other than centered differences can either reduce the need to include diffusivity, or will introduce diffusivity in regions where the Peclet number does not satisfy the above constraint. For momentum, this constraint is always relevant since MOM only uses centered differences for momentum advection.

Experience has shown that if the Reynolds/Peclet number constraint is not satisfied, then the model will not necessarily blow-up. Instead, the solution may have a tendency to slowly degrade over the length of the integration.

34.2.2 Linear stability of the diffusion equation

Convergence of the meridions makes it possible that a horizontal mixing coefficient appropriate for the mid-latitudes will be too large in the high latitudes. In order to derive conservative constraints, it is sufficient to consider the situation in Cartesian coordinates.

34.2.2.1 Laplacian mixing

Consider two dimensional Laplacian mixing in Cartesian coordinates

$$\psi_t = A_l\,(\psi_{xx} + \psi_{yy}), \tag{34.16}$$

where A_l is the Laplacian viscosity. With a uniform grid, the discrete form of this equation is given by

$$\psi_{i,j}^{n+1} = (1 - 2\sigma_x - 2\sigma_y)\,\psi_{i,j}^{n-1} + \sigma_x\,(\psi_{i+1,j}^{n-1} + \psi_{i-1,j}^{n-1}) + \sigma_y\,(\psi_{i,j+1}^{n-1} + \psi_{i,j-1}^{n-1}) \tag{34.17}$$

where

$$\sigma_x = A_l\left(\frac{2\,\Delta t}{(\Delta x)^2}\right) \tag{34.18}$$

$$\sigma_y = A_l\left(\frac{2\,\Delta t}{(\Delta y)^2}\right). \tag{34.19}$$

In time, this equation has the form of a *forward discretization* with a time step $2\Delta t$. As shown in Section (5-9) of Haltiner and Williams (1980), such forward time stepping is necessary for stability of the diffusion equation. Note that in MOM, the splitting encured by the leapfrog scheme is removed by either a Robert time filter or a periodic Euler forward or Euler backward step (see Section 21.4 for a discussion of time stepping schemes).

A von Neumann stability analysis is sufficient for the present purposes. This method determines the constraints necessary for numerical stability of an arbitrary grid wave which takes the form

$$\psi_{i,j}^{n} = B^n\,e^{i\,\mu\,x_i}\,e^{i\,\nu\,y_j}, \tag{34.20}$$

where $x_i = i\,\Delta x$, $y_j = j\,\Delta y$, and μ and ν are arbitrary wavenumbers. With this wave ansatz, the finite difference form of the diffusion equation becomes

$$B^{n+1} = \Omega\,B^{n-1} \tag{34.21}$$

where the amplification factor is

$$\Omega = 1 - 4\,\sigma_x\,\sin^2(\mu\,x_i/2) - 4\,\sigma_y\,\sin^2(\nu\,y_j/2). \tag{34.22}$$

To ensure stability, $-1 < \Omega < 1$ is required. So long as σ_x and σ_y are positive, $\Omega < 1$ is trivial to satisfy. The opposite inequality requires

$$\sigma_x\,\sin^2(\mu\,x_i/2) + \sigma_y\,\sin^2(\nu\,y_j/2) < 1/2. \tag{34.23}$$

The most conservative form of this constraint occurs when the sine terms are unity, in which case one finds

$$A_l < \frac{1}{4\Delta t}\,\left((\Delta x)^{-2} + (\Delta y)^{-2}\right)^{-1}. \tag{34.24}$$

For one spatial dimension, this constraint implies

$$A_l^{one-dim} < \frac{(\Delta s)^2}{4\,\Delta t}, \tag{34.25}$$

where again $2\Delta t$ is the leap frog time step.

For two dimensions, again being conservative and so choosing $\Delta x = \Delta y = \Delta s$, yields the constraint

$$A_l^{two-dim} < \frac{(\Delta s)^2}{8\,\Delta t}. \tag{34.26}$$

A final conservative approximation is to take Δs as the minimum of Δx and Δy within this equation.

The constraint discussed here can be likened to the CFL constraint placed on the time step due to wave propagation. That is, a diffusive or viscous flux represents a transfer of information across the grid. If this transfer occurs too fast relative to the model's time step, then the numerical scheme will go unstable. Correspondingly, in one dimension, the constraint is less strong by a factor of $1/2$. The increased restriction placed on the two-dimensional problem is similar to the two dimensional CFL constraint, as discussed in Section (5-6-7) of Haltiner and Williams (1980). Namely, the effective size of the smallest grid wave is reduced through the addition of an extra dimension, and so the constraint is stronger.

As with the CFL condition, experience has shown that if the linear diffusion equation constraint is not satisfied, even mildly, then the model will soon become wildly unstable. The instability can be removed by either reducing the time steps or reducing the mixing coefficient.

34.2.2.2 Biharmonic mixing

The same type of analysis for the two-dimensional Cartesian biharmonic equation

$$\psi_t = -A_b\,(\psi_{xxxx} + 2\,\psi_{xxyy} + \psi_{yyyy}), \tag{34.27}$$

where $A_b > 0$ is the biharmonic viscosity, leads to the biharmonic amplification factor

$$\Omega = 1 - 16\,[\sqrt{\sigma_x}\,\sin^2(\mu\,\Delta x/2) + \sqrt{\sigma_y}\,\sin^2(\nu\,\Delta y/2)]^2, \tag{34.28}$$

where now

$$\sigma_x = A_b\left(\frac{2\,\Delta t}{(\Delta x)^4}\right) \tag{34.29}$$

$$\sigma_y = A_b\left(\frac{2\,\Delta t}{(\Delta y)^4}\right). \tag{34.30}$$

Restricting $-1 < \Omega < 1$ implies

$$[\sqrt{\sigma_x}\,\sin^2(\mu\,\Delta x/2) + \sqrt{\sigma_y}\,\sin^2(\nu\,\Delta y/2)]^2 < 1/8. \tag{34.31}$$

This constraint is satisfied if the following more conservative constraint is satisfied

$$(\sqrt{\sigma_x} + \sqrt{\sigma_y})^2 < 1/8, \tag{34.32}$$

or

$$A_b < \frac{1}{16\,\Delta t}\left((\Delta x)^{-2} + (\Delta y)^{-2}\right)^{-2}. \tag{34.33}$$

The most conservative constraint is to set

$$A_b < \frac{(\Delta s)^4}{64\,\Delta t} \tag{34.34}$$

where Δs is the minimum grid spacing.

34.2.3 Western boundary currents

In models with meridional boundaries, there will be boundary currents. The Munk layer (Munk 1950, Gill 1982) is relevant for determining the width of the boundary layer in MOM. As discussed by Bryan, Manabe, and Pacanowski 1975), the model must resolve this layer with at least one grid point (optimally more than one grid point) in order to maintain numerical stability. Under-resolution of the Munk layer shows up most visibly in the vertically integrated velocity field (i.e., the barotropic streamfunction when using the rigid lid, or the free surface height with the free surface). In addition, the work of Griffies, Pacanowski, and Hallberg (1998) emphasized the importance of having at least two grid points in the Munk layer in order to minimize the level of spurious diapycnal mixing associated with tracer advection.

With n grid points per Munk layer, the viscosity must satisfy

$$A > \beta(n\,\Delta s\sqrt{3}/\pi)^3, \tag{34.35}$$

where $\beta = 2.28 \times 10^{-13}(cm\,sec)^{-1}\cos\phi$, which implies

$$A(cm^2/sec) > 3.82 \times 10^{-14}(n\,\Delta s)^3\cos\phi. \tag{34.36}$$

For example, with $\Delta s = 100km$ at the equator, having one grid cell within the Munk boundary layer requires a viscosity of $2.7 \times 10^7 cm^2/sec$, whereas with a $10km$ resolution, the viscosity must be larger than only $2.7 \times 10^4 cm^2/sec$.

Experience has shown that if the Munk layer constraint is mildly not satisfied, the model will not tend to blow-up. Instead, the barotropic mode will steadily become more noisy, and the solution less numerically and physically satisfying as the model is integrated further in time.

34.2.4 Summary: viscosity on the sphere

On a sphere using spherical coordinates, the grid spacing in the zonal direction changes according to $\cos\phi$. From many numerical and physical perspectives, it is useful to employ isotropic grids in which the latitudinal resolution is kept abreast with the converging meridions

$$\Delta\phi = \cos\phi\,\Delta\lambda. \tag{34.37}$$

In this way, a grid cell is roughly square, or isotropic, with squares becoming smaller as one moves poleward. As discussed in Section 16.1.3, MOM provides an option which constructs grids satisfying this equation.

Consistent with the desire to employ isotropic grids, it might be useful to prescribe a momentum friction which damps a particular grid scale anomaly with the same time scale regardless of the position on the sphere. As shown in Section 34.4, the damping times for a constant viscosity used for Laplacian and biharmonic friction in one Cartesian dimension is given by

$$\tau_{Lap} = (\Delta/2)^2/A \tag{34.38}$$

$$\tau_{Bih} = (\Delta/2)^4/B, \tag{34.39}$$

where Δ is the grid spacing and B is the biharmonic viscosity. Preserving the damping time as Δ changes on the sphere suggests letting A have a $\cos^2\phi$ dependence and B have a $\cos^4\phi$ dependence.

Besides providing for a constant damping time, a latitudinally dependent friction can be prescribed that relieves the time step constraint given in Section 34.2.2 which ensues when employing a constant viscosity over the extent of the sphere. That constraint becomes more restrictive on the size of $A\,\Delta t$ when moving towards the poles. Again, letting A have a $\cos^2\phi$ dependence relieves this constraint. Similar stability considerations with biharmonic friction leads to a biharmonic coefficient with $\cos^4\phi$ dependence.

The above considerations neglect the lower bound considerations given in Sections 34.2.1 and 34.2.3. Notably, if the viscosity gets too small, the flow will become numerically unstable. Therefore, as a compromise, instead of a $\cos^2\phi$ dependence, the Laplacian viscosity is typically given a $\cos\phi$ dependence

$$A = A_o \cos\phi, \tag{34.40}$$

where A_o has no latitudinal dependence. This form for the viscosity is furthermore not carried all the way to the pole. In practice, the $\cos\phi$ dependence appears sufficient to alleviate the time step restrictions arising from friction. This form for the viscosity is enabled through option *varhmix* and the suboption *am_cosine* as discussed in Section 34.6.2. For biharmonic friction, the analogous viscosity is given by

$$B = B_o \cos^3\phi, \tag{34.41}$$

where B_o has no latitudinal dependence.

In summary, there are three main constraints which are placed on the viscosity. The Reynolds number and Munk boundary layer constraint provide a lower bound on viscosity, whereas the linear diffusion equation constraint provides an upper bound. In general, modelers hope to reduce the viscosity to its lowest value consistent with these constraints. The reason is that it will allow the flow to become more advectively dominant, which is more realistic. Unfortunately, that effort is often difficult to achieve, largely due to the Reynolds number and Munk constraints.

As proposed in Section 34.7, the Smagorinsky scheme provides the most general means of satisfying each of the above three constraints with only one adjustable constant. This scheme, originally used in the GFDL model by Rosati and Miyakoda (1988), is currently being employed more frequently in experiments at GFDL.

34.3 A comment on mixing and finite impulse filtering

The simplest one-dimensional spatial finite impulse filter (Section 27.1.2) is a 1-2-1 filter given by

$$\psi_i^{n*} = (\psi_{i-1}^n + 2\,\psi_i^n + \psi_{i+1}^n)/4, \tag{34.42}$$

where ψ_i^{n*} is the filtered version of ψ_i^n. This filter will identically eliminate a $2\Delta x$ grid wave defined by the condition

$$\psi_{i-1}^n = \psi_{i+1}^n \tag{34.43}$$

$$\psi_i^n = -\psi_{i+1}^n. \tag{34.44}$$

The above filtering can be written in the form of a Laplacian mixing by subtracting ψ_i^n from both sides

$$(\psi_i^{n*} - \psi_i^n) = (\psi_{i-1}^n - 2\,\psi_i^n + \psi_{i+1}^n)/4. \tag{34.45}$$

Now consider the discrete one-dimensional Laplacian mixing equation centered in space using a leapfrog timestep of $2\Delta t$

$$\psi_i^{n+1} = \psi_i^{n-1} + \frac{2\,\Delta t\,A_l}{(\Delta x)^2}\,(\psi_{i+1}^{n-1} - 2\psi_i^{n-1} + \psi_{i-1}^{n-1}). \tag{34.46}$$

The maximum mixing coefficient compatible with linear stability constraints (Section 34.2.2) is

$$A_l^{max} = \frac{(\Delta x)^2}{4\Delta t} \tag{34.47}$$

Setting the mixing coefficient to

$$A_l = \frac{1}{2} A_l^{max} \tag{34.48}$$

results in the discrete diffusion equation

$$\psi_i^{n+1} = (\psi_{i-1}^{n-1} + 2\,\psi_i^{n-1} + \psi_{i+1}^{n-1})/4. \tag{34.49}$$

Equation (34.49) should be compared with the 1-2-1 filter given in equation (34.42). If one identifies ψ^{n+1} with ψ^{n*}, then the equations agree exactly. Multiple applications of equation (34.49) with resetting of $\psi = \psi^{n*}$ after each application is identical with integration of equation (34.42) in time.

The comparison between finite impulse filtering and mixing prompts the following comments. First, generalize the diffusion equation to include other processes which are discretized at the central time n,

$$\psi_i^{n+1} = \psi_i^{n-1} + (2\,\Delta t)\,\mathcal{A}[\psi^n] + \frac{2\,\Delta t\,A_l}{(\Delta x)^2}\,(\psi_{i+1}^{n-1} - 2\psi_i^{n-1} + \psi_{i-1}^{n-1}), \tag{34.50}$$

where $\mathcal{A}[\psi^n]$ is a functional of ψ^n. In particular, $\mathcal{A}[\psi^n]$ could represent the advection operator. Now consider the case in which $\mathcal{A}[\psi^n]$ introduces grid noise into ψ_i^{n+1} such as may occur with dispersion errors arising from numerical advection. Because the diffusion operation is always

lagged in time, it will never be able to completely remove grid noise added through $\mathcal{A}[\psi^n]$. It is for this reason that solutions of the advection-diffusion equation may contain grid noise. In contrast, if the advection-diffusion equation is solved first using

$$\psi_i^{n*+1} = \psi_i^{n-1} + (2\,\Delta t)\,\mathcal{A}[\psi^n] \tag{34.51}$$

and the diffusion component is added afterwards using

$$\psi_i^{n+1} = \psi_i^{n*+1} + \frac{2\,\Delta t\,A_l}{(\Delta x)^2}\,(\psi_{i+1}^{n-1} - 2\psi_i^{n-1} + \psi_{i-1}^{n-1}), \tag{34.52}$$

then $2\Delta x$ grid noise will be eliminated identically. The downside is that breaking the equation into two parts reduces the order of accuracy in time from second to first order. The consequence of this reduction of order of accuracy has yet to be investigated.

34.4 Comparing Laplacian and biharmonic mixing

To understand the differences and similarities between Laplacian and biharmonic dissipation, it is sufficient to consider the following linear evolution equations in one Cartesian space dimension (Semtner and Mintz 1977)

$$\psi_t = A_l\,\psi_{xx} \tag{34.53}$$

$$\psi_t = -A_b\,\psi_{xxxx}, \tag{34.54}$$

where A_l is the Laplacian eddy coefficient and A_b is the biharmonic eddy coefficient. Discretizing the Laplacian operator on a Cartesian grid of constant size Δ yields

$$\begin{aligned}\psi_t &= (A_l/\Delta^2)\,(\psi_{i+1} - 2\psi_i + \psi_{i-1}) \\ &= (A_l/\Delta^2)\left(D^{1/2} - D^{-1/2}\right)^2\psi_i\end{aligned} \tag{34.55}$$

where

$$D^m\psi_i = \psi_{i+m} \tag{34.56}$$

is a linear shift operator. A similar discretization of the biharmonic operator leads to

$$\begin{aligned}\psi_t &= -(A_b/\Delta^4)\,(\psi_{i+2} - 4\psi_{i+1} + 6\psi_i - 4\psi_{i-1} + \psi_{i-2}) \\ &= -(A_b/\Delta^4)\left(D^{1/2} - D^{-1/2}\right)^4\psi_i\end{aligned} \tag{34.57}$$

To garner a sense of how the two forms of dissipation compare, consider a monochromatic grid wave

$$\psi = c(t)\,e^{j\,k\,x_i}, \tag{34.58}$$

where k is a wavenumber, $j = \sqrt{-1}$, and $x_i = i\,\Delta$. For such a wave,

$$(D^{1/2} - D^{-1/2})e^{j\,k\,x_i} = 2\,j\,\sin(k\,\Delta/2)\,e^{j\,k\,x_i} \tag{34.59}$$

$$(D^{1/2} - D^{-1/2})^2 e^{j\,k\,x_i} = -4\,\sin^2(k\,\Delta/2)\,e^{j\,k\,x_i}. \tag{34.60}$$

$$(D^{1/2} - D^{-1/2})^4 e^{j\,k\,x_i} = 16\,\sin^4(k\,\Delta/2)\,e^{j\,k\,x_i}. \tag{34.61}$$

As such, the evolution equations for this wave under the two forms of dissipation are given by

$$\frac{d\ln c(t)}{dt} = -A_l \left(\frac{\sin(k\Delta/2)}{(\Delta/2)}\right)^2 \tag{34.62}$$

$$\frac{d\ln c(t)}{dt} = -A_b \left(\frac{\sin(k\Delta/2)}{(\Delta/2)}\right)^4. \tag{34.63}$$

The solution to these equations is an exponential damping $c(t) = c(0)\,e^{-t/\tau}$, where the inverse damping times are given by

$$\tau_{Lap}^{-1}(k) = A_l \left(\frac{\sin(k\Delta/2)}{\Delta/2}\right)^2 \tag{34.64}$$

$$\tau_{Bih}^{-1}(k) = A_b \left(\frac{\sin(k\Delta/2)}{\Delta/2}\right)^4. \tag{34.65}$$

The damping times are equal whenever

$$A_l = A_b \left(\frac{\sin(k\Delta/2)}{\Delta/2}\right)^2. \tag{34.66}$$

For example, the smallest grid wave that can live on a discrete grid has size 2Δ, which means the wavenumber for this wave is $k = \pi/\Delta$. In this case, $\sin(k\Delta/2) = 1$, and so the damping times for this wave are given by

$$\tau_{Lap} = (\Delta/2)^2/A_l \tag{34.67}$$

$$\tau_{Bih} = (\Delta/2)^4/A_b. \tag{34.68}$$

This is the strongest damping available from either form of dissipation; waves of smaller wavenumber, or longer wavelength, are less damped and so have larger τ. For the $k = \pi/\Delta$ grid wave, if the dissipation coefficients satisfy

$$A_b = (\Delta/2)^2\, A_l, \tag{34.69}$$

then the damping times are the same. For a typical mid-latitude eddy permitting model, let $\Delta = .25 \times \cos(\pi/6) \times 111 \times 1000 \times 100 = 2.4 \times 10^6 cm$. A biharmonic coefficient $A_b = 10^{19} cm^4/sec$ leads to a damping time of $2.4 days$ for the smallest grid scale waves. Relation (34.69) says that if $A_l = 7 \times 10^6 cm^2/sec$, the Laplacian dissipation will lead to the same damping time.

34.5 bryan_lewis_horizontal

This is a hybrid mixing scheme in the sense that it affects only tracers and not velocity. It specifies the horizontal tracer diffusivity A_h to be a function of depth given by the Bryan and Lewis (1979) form

$$diff_cet_k = diff_cnt_k = (ahb + (ahs - ahb)\exp(-\frac{zt_k}{50000.0})) \cdot 1.0\text{x}10^4, \tag{34.70}$$

with $ahs = 5.0\text{x}10^3\ m^2/sec$ and $ahb = 1.0\text{x}10^3\ m^2/sec$. The diffusivities $diff_cet_k$ and $diff_cnt_k$ are functions of k. It should be noted that the functional relationship can easily be changed to whatever vertical dependence is desired. Values required for this option can be changed through namelist. Refer to Section 14.4 for information on namelist variables.

34.6 Variable horizontal mixing coefficients

As discussed in Sections 34.2.4, care must be taken when formulating the form of momentum friction when employing spatially variable viscosities. This section discusses the numerical discretization of the formulation provided in Section 9.8. For the option *velocity_horz_mix_var*, viscosity may be specified *a priori* to be any function of latitude, longitude, and depth. Additionally, as discussed in Section 34.7, it may be specified through the Smagorinsky scheme. The option *tracer_horz_mix_var* allows for a similar specification of the horizontal tracer diffusivity.

34.6.1 Discretization of the new metric terms

Once the viscosity coefficients have been computed, either from some *a priori* profile or using a scheme such as that of Smagorinsky (Section 34.7), the new metric terms (Section 9.8) arising from the non-constant viscosity coefficient must be computed. The discretization of the new metric terms, which are located at the center of the U-cell, is given schematically by

$$new_metric^{(1)} = -\frac{\overline{\delta_\lambda A}^{\phi}}{\cos\phi}(\overline{\delta_\phi v}^{\phi} + a^{-1}\, v \tan\phi) + \frac{\overline{\delta_\phi A}^{\lambda}}{\cos\phi}(\overline{\delta_\lambda v}^{\lambda} + a^{-1}\, u \sin\phi) \tag{34.71}$$

$$new_metric^{(2)} = \frac{\overline{\delta_\lambda A}^{\phi}}{\cos\phi}(\overline{\delta_\phi u}^{\phi} + a^{-1}\, u \tan\phi) + \frac{\overline{\delta_\phi A}^{\lambda}}{\cos\phi}(-\overline{\delta_\lambda u}^{\lambda} + a^{-1}\, v \sin\phi). \tag{34.72}$$

Recall that the finite difference operators δ_λ and δ_ϕ absorb one factor of the earth's radius a. The averaging performed on the viscosity coefficients assumes they are defined on the corners of the U-cell. This is the natural positioning of these coefficients when employing the Smagorinsky scheme (see Section 34.7), and so the same positioning is employed in the general case described here. Note the presence of computational modes in the discretization of the velocity gradients. Again, they are of no consequence since the discretization of the corresponding Laplacian acts on all waves, thus eliminating the potential for this splitting in the metric terms to be harmful.

34.6.2 am_cosine

One sub-option for *varhmix* is option *am_cosine* which applies a cosine function to the constant value of *am* to counteract the convergence of meridians. When implemented with the option *laplacian_velocity*, the Laplacian viscosity is given by

$$A_l = am \cdot csuj. \tag{34.73}$$

When implemented with option *biharmonic_velocity*, the biharmonic viscosity is given by

$$A_b = ambi \cdot (csuj)^3. \tag{34.74}$$

34.6.3 am_taper_highlats

Another sub-option for *varhmix* is option *am_taper_highlats*. This option keeps the viscosity constant until reaching a specified latitude, defaulted to 60°, and then will taper the viscosity quadratically poleward until it reaches a fraction of the namelist viscosity. Details of the function used to taper are easily altered, and they are specified in the Smagorinsky fortran routine, along with option *am_cosine*

34.7 The Smagorinsky scheme

The Smagorinsky scheme in MOM provides expressions for horizontal momentum viscosity coefficients and horizontal tracer diffusivity coefficients. It does not provide expressions for vertical dissipation. The framework established by Smagorinsky dissipation allows for the general introduction of a non-constant viscosity coefficient in a physically consistent fashion, and the details for general non-constant viscosity are provided in Section 34.6. The values of the dissipation coefficients are based on a closure scheme which dissipates energy and tracer variance where the maximum dissipation acts on grid scale features. The fundamental ideas and history behind the method are nicely summarized in Smagorinsky (1993). For an alternative perspective, Leith (1996) has provided a critique of the Smagorinsky closure and argues for a form of dissipation based on ideas from two-dimensional turbulence.

34.7.1 General ideas

The Smagorinsky scheme is based on some physical assumptions. The first is the relevance of momentum dissipation as motivated by elasticity theory. Within that framework, Smagorinsky produced expressions for the friction relevant to a horizontally isotropic and hydrostatic fluid on a sphere (see Chapter 9 for complete details). This form for the friction is employed by MOM for the general case of a non-constant horizontal viscosity (option *varhmix*), and is described in Section 34.6. In addition to elasticity theory, Smagorinsky applied certain ideas from Kolmogorov and Heisenberg, which are only justified strictly to 3-dimensional homogeneous and isotropic turbulence. These ideas were used to determine energy dissipation rates and the dissipation or viscosity coefficient. This section provides a discussion of the Smagorinsky scheme and its implementation in MOM.

In the closure scheme, Smagorinsky introduced an adjustable parameter which is basically tuned for each model situation. This parameter sets the overall magnitude of the viscosity coefficient computed from the scheme. In an attempt to bring the definition of this parameter into line with that used by others, it is useful to present the following discussion, based largely on that given in Smagorinsky (1993), Section 1.9.

First, consider the dissipation of kinetic energy, which is written

$$\epsilon = A\, D^2, \tag{34.75}$$

where $|D| = \sqrt{D_T^2 + D_S^2}$ is the total deformation rate,

$$D_T = \frac{1}{a\cos\phi}\frac{\partial u}{\partial\lambda} - \cos\phi\,\frac{\partial(v\sec\phi)}{\partial(a\phi)} \tag{34.76}$$

$$D_S = \frac{1}{a\cos\phi}\frac{\partial v}{\partial\lambda} + \cos\phi\,\frac{\partial(u\sec\phi)}{\partial(a\phi)}, \tag{34.77}$$

are the horizontal tension and shearing rate of strain (see Chapter 9 for more discussion), and the viscosity coefficient A is written as

$$A = (k_o/k_m)^2|D|. \tag{34.78}$$

The viscosity depends in a nonlinear manner on the flow, and so is often termed a *nonlinear viscosity.* Note that the dimension of the deformation rates are $time^{-1}$. k_o is an adjustable dimensionless parameter, and has been predicted by theories and determined by experiments. In

MOM, $k_o \equiv ksmag$ is used as an adjustable parameter. Originally, Smagorinsky (1963) was motivated to set $ksmag = 0.4$, which is the value expected from wall boundary layer turbulence, where 0.4 is the "von Karman constant." This is the default value in the model. However, k_o depends on the particular geometry and symmetry of the turbulent flow. Consequently, there is little theoretical justification for choosing one particular value of 0.4 in an ocean model, and so the researcher should consider tuning this parameter accordingly, much as one tunes values for the constant viscosity coefficient.

For the purpose of directly comparing the MOM implementation of Smagorinsky dissipation with other model implementations, it is important to be explicit about how the horizontal wavenumber k_m is defined. In MOM, k_m is defined to be the largest resolvable wavenumber local to the grid cell of interest. That is, for a given horizontal grid scale Δs, $2\Delta s$ represents the smallest explicitly resolvable wave length. The corresponding largest resolvable wavenumber is $k_m = (2\pi)/(2\Delta s) = \pi/\Delta s$. Hence, the viscosity is determined by

$$A = (k_o \Delta s/\pi)^2 |D|. \tag{34.79}$$

This result differs by a factor of π from that employed by Smagorinsky (1963), as well as that used in MOM 1. More will be said about differences between the MOM 1 and MOM coefficients in Section 34.7.3. On an anisotropic grid,

$$\Delta s \equiv min(\Delta x, \Delta y). \tag{34.80}$$

Note that it is important to define a single viscosity, not a separate one for each of the two horizontal directions as suggested by Rosati and Miyakoda (1988). The reason is that if two viscosities are introduced, then that would generally break the trace-free and symmetric properties of the stress tensor. As discussed in Chapter 9, these properties are fundamental to ensuring the friction is truly dissipative and will not introduce unphysical torques.

For the dissipation of tracer variance through horizontal diffusion, Smagorinsky assumed that the turbulent tracer transport is purely forced, unlike the geostrophic mixing of tracers occurring with baroclinic eddies. This assumption implies a unit value for the horizontal turbulent Prandtl number[1], thus equating the horizontal viscosity to diffusivity. In MOM, this unit Prandtl number assumption is the default. Yet the researcher can change this value through the namelist parameter *prandtl*. For coarse models (resolution greater than 2 degrees), a horizontal Prandtl number close to 100 is typical.

A necessary condition for 3D turbulence theories to be relevant for a GFD model is that the resolved scales of motion should cover at least a portion of the geostrophic turbulence spectrum. The reason is that 3D turbulence ideas, which incorporate a cascade of energy to small scales, are not dominant at length scales within the mostly two-dimensional planetary geostrophic range, where energy cascades to large scales. Therefore, the Smagorinsky dissipation is not meant to parameterize the effects from synoptic-scale eddies. This restriction is not so strong for atmospheric models in which the synoptic scale geostrophic eddies are typically well resolved (although Leith 1996 argues differently). For the ocean, the synoptic eddies are much smaller in scale and thus more difficult to numerically resolve. Therefore, the use of the Smagorinsky scheme may not strictly appear to be relevant for models which do not at least partially resolve the geostrophic eddies. However, the amount of the geostrophic turbulent scale that should be resolved remains unclear. On a less theoretical note, the state of the art in most large-scale GFD parameterizations of subgrid energy dissipation is approached

[1]Recall that the Prandtl number is the ratio of the viscosity to diffusivity.

in a mostly empirical manner. For example, in ocean models containing a good bit of the geostrophic eddy spectrum, biharmonic dissipation (see Section 34.4) is the choice most commonly employed. Such dissipation acts less on the large scales of motion than Laplacian dissipation, and so allows for more energetic geostrophic eddies (Semtner and Mintz 1977). The Smagorinsky scheme is an alternative to biharmonic dissipation, and it provides a means, within a second order scheme, for selectively dissipating only those scales with the largest energy. Furthermore, as shown by Rosati and Miyakoda (1988), the Smagorinsky scheme provides the ocean modeler with the means to employ relatively small dissipation, even in a modestly resolved global model, while maintaining numerical integrity. When using the Smagorinsky scheme in MOM, one is treading on mostly new ground with little z-coordinate experience, other than that of Rosati and collaborators.

34.7.2 Choosing the scaling coefficient

As discussed in Section 34.2, there are at least three things which must be considered when deciding what value to use for the scaling coefficient k_o. Firstly, the viscosity used in a centered differenced advection scheme must be large enough so that the cell Reynolds number satisfies (Bryan *et al* 1975)

$$Re_{cell} = U\Delta s/A < 2 \Rightarrow A > U\Delta s/2, \tag{34.81}$$

where U is a velocity scale and Δs the corresponding grid spacing. In order to get a rough value for k_o based on this constraint, let $|D| \approx U/\Delta s$, which means that the Smagorinsky viscosity is roughly $A \approx (k_o/\pi)^2\, U\,\Delta s$. Forcing $Re_{cell} < 1/2$ implies that

$$k_o > \pi/\sqrt{2} \approx 2.2. \tag{34.82}$$

Actually, this estimate provides an upper bound on the viscosity realized with the scheme, since $\Delta U \approx U$ only in regions of strong horizontal shear. In other regions, the nonlinear viscosity will be significantly smaller.

The second consideration involves the western boundary layer (Section 34.2.3). With n grid points within the Munk layer, the viscosity must satisfy

$$A > \beta(n\,\Delta s\sqrt{3}/\pi)^3, \tag{34.83}$$

where $\beta = 2.28 \times 10^{-13}(cm\,sec)^{-1}\cos\phi$, which implies

$$A(cm^2/sec) > 3.82 \times 10^{-14}(n\,\Delta s)^3\cos\phi. \tag{34.84}$$

For example, with a $100km$ model resolution Δs, having one grid cell within the Munk boundary layer requires a viscosity of $2.7 \times 10^7 cm^2/sec$, whereas with a $10km$ resolution, the viscosity must be larger than only $2.7 \times 10^4 cm^2/sec$. In a $100km$ resolution model, the typical velocities are $10cm/sec$, and so the maximum Smagorinsky nonlinear viscosities will be on the order of $k_o^2 \times 10^7 cm^2/sec$. With $k_o = 2.2$, the model might be safely resolving the Munk layer since the regions where the layer is formed is also where there are strong shears. A larger value of k_o, however, might be necessary.

The considerations concerning the Munk boundary layer can be computed prior to any model computations since they do not involve flow properties. If the result of this calculation is that the needed Munk viscosity far excedes a reasonable estimate based on a chosen k_o, such as the 2.2 based on the cell Reynolds number, then there are two choices that can be

made. The first is to run with a constant background viscosity set large enough to satisfy the Munk boundary layer condition. If this choice is made, there is little reason to employ the Smagorinsky scheme since the viscosities that it will compute will be swamped by the background. The second choice is to increase k_o so that the nonlinear viscosity will satisfy the Munk boundary layer condition, at least for the cells within the boundary layer.

A final consideration concerns the linear stability constraint discussed in Section 34.2.2. For constant viscosity, the main region in which this constraint becomes important is in the high latitudes due to the convergence of the meridions. Because the Smagorinsky scheme "knows about the sphere", which means it has the proper $\cos\phi$ factors, it naturally produces small viscosities in the high latitudes. Hence, the convergence of the meridions is of no issue. That is, k_o need not have any latitudinal dependence. This property is quite convenient for global models.

34.7.3 Scaling coefficient conventions

There are a few factors in the viscosity coefficients that differ in these expressions from those defined in Rosati and Miyakoda (1988). First, the factor of π is missing since their work was based on the earlier work of Smagorinsky (1963) which omitted this factor. Secondly, there are some factors of 2 which differ. The implementation of the scheme in MOM 1 effectively multiplied the definition of Rosati and Miyakoda's c^λ and c^ϕ by $\sqrt{2}$. Additionally, the identification of a different viscosity in the respective horizontal directions is not followed in the MOM implementation, since to do so would generally sacrifice the trace-free and symmetric properties of the stress tensor. This latter change in approach precludes a clean comparison between the MOM 1 and MOM implementations of the scheme. However, the lack of direct compatibility was deemed acceptable since preserving the continuum properties of the stress tensor are fundamental to the method. Nonethless, it is useful to provide the relevant expressions for the viscosity coefficients employed in the two implementations.

$$A^\lambda_{mom1} = |D|(c\, a\Delta\lambda\cos\phi)^2/\sqrt{2} \tag{34.85}$$

$$A^\phi_{mom1} = |D|(c\, a\Delta\phi)^2/\sqrt{2}, \tag{34.86}$$

whereas MOM uses

$$A = |D|(ksmag\ \Delta s/\pi)^2, \tag{34.87}$$

where $\Delta s = min(a\cos\phi\Delta\lambda, a\Delta\phi)$. If $\Delta\lambda\cos\phi = \Delta\phi$, then a $ksmag = 0.4$ in MOM corresponds to a constant $c = 0.15$ in MOM 1. The default value in MOM 1 was 0.14.

34.7.4 Smagorinsky and isoneutral mixing together

If one of the isoneutral mixing schemes (Section 35.1) is enabled, it is likely that the researcher does not wish the Smagorinsky scheme to specify horizontal or lateral tracer diffusivities. Rather, either a constant isoneutral diffusivity or one of the nonconstant schemes discussed in Section 35.2 are typical. Consequently, when both options *smagnlmix* and one of the isoneutral mixing schemes are enabled, the tracer diffusivity computed from the Smagorinsky scheme is set to zero inside of the routine *smagnl.F*. In this case, the Smagorinsky scheme will only specify the horizontal viscosities.

34.7.5 Biharmonic Smagorinsky

Section 9.5 provided the theoretical framework for implementing nonconstant viscosities with the biharmonic scheme, and Section 34.11 described how MOM implements such a scheme. If the Smagorinsky scheme is employed along with the laplacian mixing, then the discussion in Section 34.2.2 indicates that the Smagorinsky viscosity should satisfy

$$A_l < \frac{(\Delta s)^2}{8\,\Delta t}, \tag{34.88}$$

where $\Delta s = min(\Delta x, \Delta y)$. A biharmonic viscosity should likewise satisfy

$$A_b < \frac{(\Delta s)^4}{64\,\Delta t}. \tag{34.89}$$

These results suggest setting the biharmonic viscosity from the Smagorinsky scheme according to

$$A_b = A_l \left(\frac{min[(\Delta x)^2, (\Delta y)^2]}{8} \right), \tag{34.90}$$

where A_l is the Laplacian Smagorinsky viscosity. This scheme has the advantages of the Laplacian Smagorinsky scheme while providing the enhanced scale selectivity of a biharmonic operator.

34.7.6 Discretization of the Smagorinsky viscosity coefficient

The nonlinear viscosity coefficient is determined in terms of the deformation rate and grid spacing. Since D_T and D_S involve terms with derivatives in both horizontal directions, there needs to be some sort of averaging to place them at a common grid position. What was done in MOM 1 was to define both at the north face of the U-cell (see Figure 16.2). This is the natural position for the meridional derivative terms $\partial u/\partial\phi$ and $\partial v/\partial\phi$. To get the zonal derivative terms $\partial u/\partial\lambda$ and $\partial v/\partial\lambda$ defined at the north face, it was necessary to average over the four zonal derivatives surrounding the north face. The problem with such "four point" averages on the B-grid, as discussed in Appendices C and E, is that they can introduce computational modes, and such modes were present in MOM 1. Computational modes are not always problematical if there are other processes which can render the possible growth of the modes harmless. The problem with these particular modes in the Smagorinsky scheme is that they are field configurations which will yield a zero deformation rate, and so they will not be dissipated. Furthermore, these modes are waves at the grid scale, which is the scale for which the Smagorinsky scheme should provide the largest amount of dissipation. Therefore, the modes are not acceptable for the implementation of the scheme, and so another approach is neccesary.

The following discretization eliminates the computational modes present in MOM 1 by performing a symmetric discretization of the shearing strains to define them at the northeast corner of a U-cell (point $T_{i+1,j+1}$). The strains are later averaged to define the viscosity coefficient where needed. In this average, since $|D| = \sqrt{D_T^2 + D_S^2}$ is nonlinear, there will be no computational mode in the velocity field arising from such averaging operations.

The discretized D_T and D_S, located at the northeast corner of the U-cell (i.e., at the T-point $T_{i+1,k,j+1}$), are given by

$$D_T \;=\; \frac{1}{\cos\phi}\,\overline{\delta_\lambda u}^{\phi} - \cos\phi\,\overline{\delta_\phi(v/\cos\phi)}^{\lambda} \tag{34.91}$$

$$D_S = \frac{1}{\cos\phi}\,\overline{\delta_\lambda v}^{\phi} + \cos\phi\,\overline{\delta_\phi(u/\cos\phi)}^{\lambda} \tag{34.92}$$

Refer to Fig 16.2 for the layout of grid cells in the horizontal. Exposing grid indices to define quantities on the northeast corner of a U-cell yields

$$\begin{aligned}
[D_T]_{i,k,j} = & \frac{1}{2dxt_{i+1}\cos\phi^T_{jrow+1}}[(u_{i+1,k,j,1,\tau-1} - u_{i,k,j,1,\tau-1}) + (u_{i+1,k,j+1,1,\tau-1} - u_{i,k,j+1,1,\tau-1})] \\
- & \frac{\cos\phi^T_{jrow+1}}{2dyt_{jrow+1}}[(u_{i,k,j+1,2,\tau-1} + u_{i+1,k,j+1,2,\tau-1})/\cos\phi^U_{jrow+1} \\
& -(u_{i,k,j,2,\tau-1} + u_{i+1,k,j,2,\tau-1})/\cos\phi^U_{jrow}]
\end{aligned} \tag{34.93}$$

$$\begin{aligned}
[D_S]_{i,k,j} = & \frac{1}{2dxt_{i+1}\cos\phi^T_{jrow+1}}[(u_{i+1,k,j,2,\tau-1} - u_{i,k,j,2,\tau-1}) + (u_{i+1,k,j+1,2,\tau-1} - u_{i,k,j+1,2,\tau-1})] \\
+ & \frac{\cos\phi^T_{jrow+1}}{2dyt_{jrow+1}}[(u_{i,k,j+1,1,\tau-1} + u_{i+1,k,j+1,1,\tau-1})/\cos\phi^U_{jrow+1} \\
& -(u_{i,k,j,1,\tau-1} + u_{i+1,k,j,1,\tau-1})/\cos\phi^U_{jrow}]
\end{aligned} \tag{34.94}$$

The square brackets on D_T and D_S are intended only to highlight the indices. Since D_T and D_S are needed only for computing the nonlinear viscosity coefficient A, there is actually no need to save them in storage. The viscosity coefficient A is defined at the same point as the strains, thus yielding

$$A_{i,k,j} = (ksmag\,\Delta s/\pi)^2|D|_{i,k,j} \tag{34.95}$$

where

$$\Delta s = min(dxt_{i+1}\cos\phi^T_{jrow+1}, dyt_{jrow+1}), \tag{34.96}$$

with

$$|D|_{i,k,j} = \sqrt{[D_T]^2_{i,k,j} + [D_S]^2_{i,k,j}}. \tag{34.97}$$

Again, since $|D|$ is only needed for computing the viscosity, it is not stored in memory. Note that the grid spacing used for determining Δs corresponds to the T-cell since the viscosity coefficient is centered on a T-point.

For constructing the viscosity on the east and north side of the U-cell, which is where they are required for constructing momentum fluxes, a spatial average must be performed

$$visc_ceu_{i,k,j} = visc_c_back + \overline{A_{i,k,j}}^{\phi} \tag{34.98}$$

$$visc_cnu_{i,k,j} = visc_c_back + \overline{A_{i,k,j}}^{\lambda}. \tag{34.99}$$

For those viscosity coefficients needed at that center of the U-cell, which is required for constructing the old metric terms, a four point average yields

$$\overline{A_{i,k,j}}^{\lambda\phi}. \tag{34.100}$$

Spatial averages of the viscosity coefficients can be performed with impunity since this will not introduce computational modes into the velocity field. Again, the reason is that viscosity is a nonlinear function of the velocity gradients through the deformation rate $|D|$, and so there will be no splitting of the grid upon averaging. Even if there was such a splitting, it would be of no consequence since the formulation of the non-constant viscosity friction exposes a Laplacian operator, which has no computational mode and so renders harmless any such modes which may occur in other portions of the friction (see Section 34.6 for further discussion).

In the MOM Smagorinsky code, there are three parameters which are input to the scheme: *ksmag*, *visc_c_back*, and *diff_c_back*. Each can be changed through the namelist. Refer to Section 14.4 for information on namelist variables. The MOM default for *ksmag* is $ksmag = 0.4$. With *ksmag* set large enough, the Smagorinsky scheme should in principle produce enough dissipation to ensure satisfaction of the relevant grid stability criteria. Nevertheless, the researcher may wish to add additional constant background viscosity coefficient and diffusivity through the parameters *visc_c_back* and *diff_c_back*, both of which are defaulted to zero but can be changed through namelist.

34.7.7 Diffusive terms for the tracer equation

Since metric terms do not appear for diffusion of scalars (tracers), it is only necessary to formulate diffusion coefficients rather than horizontal flux and metric terms as in the case of velocity. The diffusive flux is computed inside routine *tracer*. The diffusion coefficients at the east and north face of a T-cell are given by

$$diff_cet_{i,k,j} = diff_c_back + prandtl^{-1}\,\overline{A_{i-1,k,j-1}}^{\lambda} \tag{34.101}$$

$$diff_cnt_{i,k,j} = diff_c_back + prandtl^{-1}\,\overline{A_{i-1,k,j-1}}^{\phi} \tag{34.102}$$

where the Prandtl number *prandtl* has been exposed. It is set to unity in the Smagorinsky implementation.

For diagnostic purposes, the viscosity coefficient and diffusivity coefficients are written out when option *matrix_sections* or option *save_mixing_coeff* are enabled.

34.8 tracer_horz_laplacian

With option *tracer_horz_laplacian*, the general form of the horizontal piece of the tracer tracer mixing operator is given by (see also Section 21.3.2)

$$R(t_{i,k,j,n}) = DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} \tag{34.103}$$

$$DIFF_Tx_{i,k,j} = \frac{1}{\cos\phi^{T}_{jrow}}\delta_{\lambda}(diff_fe_{i-1,k,j}) \tag{34.104}$$

$$DIFF_Ty_{i,k,j} = \frac{1}{\cos\phi^{T}_{jrow}}\delta_{\phi}(diff_fn_{i,k,j-1}). \tag{34.105}$$

The diffusive fluxes *diff_fe* and *diff_fn* are situated on the east and north sides of the tracer cells, respectively, and are given by

$$diff_fe_{i,k,j} = \frac{diff_cet_{i,k,j}}{\cos\phi^{T}_{jrow}}\delta_{\lambda}(t_{i,k,j,n,\tau-1}) \tag{34.106}$$

$$diff_fn_{i,k,j} = diff_cnt_{i,k,j}\cdot\cos\phi^{U}_{jrow}\delta_{\phi}(t_{i,k,j,n,\tau-1}). \tag{34.107}$$

This piece of the mixing operator is combined with the vertical piece discussed in Chapter 33 to form the full tracer mixing operator. Note the time lag placed on the diffusive fluxes, consistent with the linear stability discussion in Section 34.2.2.

When using option *velocity_horz_mix_const*, the tracer diffusivities at the eastern and northern faces of tracer grid cells take the constant values

$$diff_cet = A_h \tag{34.108}$$

$$diff_cnt = A_h. \tag{34.109}$$

When using option *velocity_horz_mix_var*, *diff_cet* and *diff_cnt* become arrays. Note that this use of two conventions slightly complicates the model code, but it is important since constant coefficient experiments remain quite common and it is useful to keep memory requirements to a minimum.

For reasons of computational speed, the mixing coefficients are combined with grid factors when computing diffusive fluxes. For example, with constant diffusivities, the diffusive flux at the east side of a tracer cell takes the form

$$diff_fe_{i,k,j} = ah_cstdxur_{i,j}(t_{i+1,k,j,n,\tau-1} - t_{i,k,j,n,\tau-1}), \tag{34.110}$$

where

$$ah_cstdxur_{i,j} = \frac{diff_cet}{\cos\phi^T_{jrow} \cdot dxu_i} \tag{34.111}$$

There is a similar absorption of the mixing coefficient *diff_cnt* for computing the northward diffusive tracer flux. The coefficient A_h is input through namelist. Refer to Section 14.4 for information on namelist variables.

34.9 tracer_horz_biharmonic

With option *tracer_horz_biharmonic*, the general form of the horizontal piece of the tracer mixing operator is determined by iterating the Laplacian mixing. That is, first one computes minus the Laplacian mixing operator

$$del2_tracer_{i,k,j,n} = -DIFF_Tx_{i,k,j} - DIFF_Ty_{i,k,j}, \tag{34.112}$$

where the fluxes used to construct $DIFF_Tx_{i,k,j}$ and $DIFF_Ty_{i,k,j}$ are the usual diffusive fluxes described in Section 34.8. Note that the full biharmonic diffusivity is used to compute these diffusive fluxes. The n label is for the tracers, with $n = 1, nt$. The biharmonic diffusivity will be a constant if option *tracer_horz_mix_const* is enabled, and an array if option *tracer_horz_mix_var* is enabled.

Next, fluxes of the scalar quantity $del2_tracer_{i,k,j,n}$ on the eastern and northern sides of T cells are computed using

$$\begin{aligned} diff_fe_{i,k,j} &= \frac{\delta_\lambda(del2_tracer_{i,k,j,n})}{\cos\phi^T_{jrow}} \\ &= \frac{del2_tracer_{i+1,k,j,n} - del2_tracer_{i,k,j,n}}{\cos\phi^T_{jrow} dxu_i}, \; j = 2, jmw-1 \end{aligned} \tag{34.113}$$

$$\begin{aligned} diff_fn_{i,k,j} &= \cos\phi^U_{jrow}\delta_\phi(del2_tracer_{i,k,j,n}) \\ &= \cos\phi^U_{jrow}\frac{del2_tracer_{i,k,j+1,n} - del2_tracer_{i,k,j,n}}{dyu_{jrow}}\ , j = 1, jmw-1. \end{aligned} \tag{34.114}$$

These fluxes are of the same form as fluxes of the tracers, with the important difference that fluxes of *del2_tracer* require no diffusivity. The reason is that the full biharmonic diffusivity has already been used to compute the scalar *del2_tracer*. The fluxes of *del2_tracer* are then used in the operators $DIFF_Tx_{i,k,j}$ and $DIFF_Ty_{i,k,j}$. The result is the tracer biharmonic mixing operator. Two extra boundary conditions are required beyond the usual Laplacian operator, and they are prescribed to be a vanishing of fluxes of *del2* at land boundaries.

Since option *tracer_horz_biharmonic* is a fourth order scheme, the minimum size of the memory window is $jmw = 4$ instead of $jmw = 3$ required for second order differences. As with all fourth order schemes, option *fourth_order_memory_window* must be enabled to handle this option. This is automatically done when either option *tracer_horz_biharmonic* or option *velocity_horz_biharmonic* is enabled.

34.10 velocity_horz_laplacian

As derived in Section 9.8, the general form of the zonal ($n = 1$) and meridional ($n = 2$) friction as implemented with MOM's spherical coordinates is given by (see also Section 21.3.1)

$$F^U(u_{i,k,j,n,\tau-1}) = DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_metric_{i,k,j,n} \tag{34.115}$$

$$DIFF_Ux_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(diff_fe_{i-1,k,j}) \tag{34.116}$$

$$DIFF_Uy_{i,k,j} = \frac{1}{\cos\phi^U_{jrow}}\delta_\phi(diff_fn_{i,k,j-1}) \tag{34.117}$$

$$\begin{aligned} DIFF_metric_{i,k,j,n} &= A_m\frac{1-\tan^2\phi^U_{jrow}}{radius^2}\ u_{i,k,j,n,\tau-1} \\ &\mp A_m\frac{2\sin\phi^U_{jrow}}{radius^2\cdot\cos^2\phi^U_{jrow}}\delta_\lambda(\overline{u_{i-1,k,j,3-n,\tau-1}}^{\lambda}) \\ &+ new_metric_{i,k,j,n,\tau-1} \end{aligned} \tag{34.118}$$

Again note the time lag placed on each of the friction terms, which is required for linear stability. The form of term *new_metric* is given in Section 34.6.1. This term vanishes when using a constant viscosity, but generally is nonzero. The horizontal velocity fluxes are given by

$$diff_fe_{i,k,j} = \frac{visc_ceu_{i,k,j}}{\cos\phi^U_{jrow}}\delta_\lambda(u_{i,k,j,n,\tau-1}) \tag{34.119}$$

$$diff_fn_{i,k,j} = visc_cnu_{i,k,j}\cdot\cos\phi^T_{jrow+1}\delta_\phi(u_{i,k,j,n,\tau-1}), \tag{34.120}$$

where *visc_ceu* and *visc_cnu* are arrays if option *velocity_horz_mix_var* is enabled, and a constant scalar if using option *velocity_horz_mix_const*.

34.11 velocity_horz_biharmonic

As derived in Section 9.5, the biharmonic operator acting on the horizontal velocity field can be implemented as an iterative algorithm in a similar manner to when biharmonic acts on tracers. The central difference is the addition of the metric terms.

First, the *del*2_*vel* operator is constructed from minus the Laplacian friction

$$del2_{i,k,j,n} = -DIFF_Ux_{i,k,j} - DIFF_Uy_{i,k,j} - DIFF_metric_{i,k,j,n}. \tag{34.121}$$

The northern and eastern fluxes of velocity have been computed as in Section 34.10. The metric term is computed with the spatial dependence of the viscosity taken into account, just as for Laplacian friction. Note that because it is necessary to take the spatial derivative of the viscosities to construct the new metric term (e.g., Section 34.6), it is important to use the full biharmonic viscosity when constructing *del*2_*vel*, and not to use the square-root of the viscosity as was done in earlier versions of MOM.

Next, fluxes of $del2_vel_{i,k,j,n}$ on the eastern and northern sides of U cells along with the metric term are computed using

$$\begin{aligned} diff_fe_{i,k,j} &= \frac{\delta_\lambda(del2_vel_{i,k,j,n})}{\cos\phi^U_{jrow}} \\ &= \frac{del2_vel_{i+1,k,j,n} - del2_vel_{i,k,j,n}}{\cos\phi^U_{jrow} dxt_{i+1}}\ , j = 2, jmw-1 \end{aligned} \tag{34.122}$$

$$\begin{aligned} diff_fn_{i,k,j} &= \cos\phi^T_{jrow+1}\delta_\phi(del2_vel_{i,k,j,n}) \\ &= \cos\phi^T_{jrow+1}\frac{del2_vel_{i,k,j+1,n} - del2_vel_{i,k,j,n}}{dyt_{jrow+1}}\ , j = 1, jmw-1 \\ DIFF_metric_{i,k,j,n} &= \frac{1-\tan^2\phi^U_{jrow}}{radius^2} del2_vel_{i,k,j,n} \\ &\quad \mp \frac{2\sin\phi^U_{jrow}}{radius\cdot\cos^2\phi^U_{jrow}} \frac{del2_vel_{i+1,k,j,3-n} - del2_vel_{i-1,k,j,3-n}}{dxt_i + dxt_{i+1}}. \end{aligned} \tag{34.123}$$

Note the absence of a viscosity in these fluxes. The reason is that the biharmonic viscosity has already been used in the *del*2_*vel* operator. Hence, the metric term here does not include the new piece arising from a nonconstant viscosity. Again, this result holds whether the biharmonic viscosity is a constant or not. Two extra boundary conditions are required, and they are prescribed to be a vanishing of fluxes of *del*2_*vel* at land boundaries.

Finally, the fluxes of $del2_vel_{i,k,j,n}$ are used to define the biharmonic mixing operator

$$bih_{i,k,j,n} = DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_metric_{i,k,j,n}, \tag{34.124}$$

where the operators $DIFF_Ux$ and $DIFF_Uy$ are defined using the *del*2_*vel* fluxes.

Since option *biharmonic_velocity* is a fourth order scheme, the minimum size of the memory window is $jmw = 4$ instead of $jmw = 3$ required for second order differences. As with all fourth order schemes, option *fourth_order_memory_window* must be enabled to handle this option. This is automatically done when either option *biharmonic_velocity* is enabled.

34.12 velocity_horz_friction_operator

This option enables the discretization of friction according to the development given in Sections 9.7 and Appendix D. Either Laplacian or biharmonic friction are available, with constant or variable viscosity. Refer to those sections for details. Note that this code is available *only* with option *max_window* also enabled. This form of the friction discretization will be the default in future versions of MOM.

Chapter 35

Isoneutral SGS options

This chapter is devoted to options whose main focus is on the isoneutral mixing and stirring of tracers and thickness. For horizonal and vertical tracer mixing options, refer to Chapters 33 and 34.

35.1 Basic isoneutral schemes

The goal of this section is to help guide the researcher through the various model options for mixing tracers with isoneutral diffusion and/or one of the various skew-diffusion processes. These schemes explicitly mix only tracers. These schemes should be enabled concurrently with one basic mixing option discussed in Chapters 34 and 33. This section describes only those schemes which are implemented with constant diffusivities. Generalizations to nonconstant diffusivities are described in Section 35.2.

35.1.1 A note about MOM3 updates

The code for the isoneutral mixing schemes have undergone some restructuring in MOM3 to account for the additional options of *biharmonic_rm* (Section 35.1.8), and the possibility of using nonconstant diffusivities (Section 35.2). Additional streamlining of the code to help speed up the schemes have been implemented as well. In summary, although there are major changes in the MOM2 versus MOM3 code for isoneutral mixing, these changes do not reflect any bugs in the MOM2 code.

35.1.2 Summary of the isoneutral mixing schemes

There are two common aspects of each mixing scheme. Firstly, each involves the computation of neutral directions, or more precisely, the slopes at each grid point of the tangent to the locally defined potential density surface. These *isoneutral slopes* provide the fundamental orientation for the tracer mixing schemes described here. As such, all schemes which compute the slopes are referred to in the following as *isoneutral mixing* schemes, and the central model option associated with these schemes is *isoneutralmix*. In MOM, the isoneutral slopes are computed using the best available (within the chosen grid stencil) discrete approximation to the locally referenced potential density. A great deal of effort has gone into refining the discretization of these slopes. The reason for such effort is that inconsistently discretized slopes can lead to instabilities in the isoneutral diffusion scheme (Griffies *et al.* 1998). The isoneutral slopes used

in all of the following mixing schemes are computed as discussed in Griffies *et al.* (see also Appendix C).

Secondly, Griffies *et al.* motivate a discretization of the diffusive flux components consisting of groups of four "triads" of slopes. These triads build up each of the isoneutral diffusion flux components on the side of the tracer cells. The use of this discretization ensures that isoneutral diffusion will reduce the tracer variance. Section 35.1.5 provides details of the isoneutral diffusion scheme in MOM.

In addition to using the slope computation and triad structure for isoneutral diffusion, Griffies (1998) showed how this technology can be exploited for the discretization of the Gent and McWilliams (1990; GM90) eddy stirring process. The results provide strong motivation for using a skew-diffusion, or skew-flux, approach for discretizing parameterized adiabatic eddy stirring. As a result, this approach is the default option in MOM for the adiabatic stirring schemes. The alternative approach, which is more cumbersome and inefficient computationally, involves the use of an extra term in the tracer advection velocity. This "eddy-induced advection velocity" approach is retained for comparative and diagnostic purposes.

Roberts and D. Marshall (1998; RM98) provide motivation for employing an adiabatic biharmonic operator. RM98 show how such an operator can provide an adiabatic means to damp grid-scale structures. Their scheme is implemented in MOM using the skew-flux approach. Section 35.1.8 provides details of the RM98 scheme in MOM.

Note that the continous versions of the GM90 and RM98 schemes are adiabatic in the sense that they preserve all moments of a tracer. On the discrete lattice, however, this property can at best be emulated by conserving the first (mean) and second (variance) moments. The skew flux approach, when implemented in terms of the triads of Griffies *et al.* (1998), conserves the first and second tracer moments for the GM90 and RM98 schemes.

In a level model, the isoneutral slopes can become steep, if not vertical. For example, isoneutral slopes are effectively vertical in regions of free convection. The relative steepness of a slope is determined by the model's grid aspect ratio, the time step, and the diffusivity. For slopes that are greater in magnitude than some maximum slope, it is necessary to implement a numerical stabilization for each of the isoneutral mixing schemes. Otherwise, the schemes will become linearly unstable (Cox 1987, Griffies *et al.* 1998). The form for stabilization in the past was "slope clipping." As discussed by Griffies *et al.*, slope clipping can introduce a substantial amount of unphysical dianeutral fluxes. Consequently, slope clipping *is not* available in MOM. Rather, either one of two means of *diffusivity tapering* is now employed. This approach ensures numerical stability without introducing spurious dianeutral fluxes (see Gerdes *et al.* 1991, Danabasoglu and McWilliams 1996, or Griffies *et al.* 1998). Note that when tapering, *all* of the isoneutral mixing processes are turned down to zero as the slope increases. The only nonzero mixing, besides that from advection or convection, arises from any nonzero vertical diffusivity. For various reasons, such as those discussed by Treguier *et al.* (1997), it might be desirable to maintain an additional nonzero horizontal diffusivity in these steep sloped regions. Such diffusion will be the only dissipative source of density mixing in regions with vertical density isolines. This option is available in MOM. Section 35.1.9 provides details of steep slope options.

Within the framework of any of the above mixing schemes, particular values for the eddy diffusivities must be chosen. Currently, most large-scale modelers use constant diffusivities. However, recent papers (e.g., Held and Larichev 1996, Treguier *et al.* 1997, Visbeck *et al.* 1997, Killworth 1997, and others) have criticized this approach from various perspectives. In MOM, the schemes of Held and Larichev (1996) and Visbeck *et al.* (1997) have been implemented. Section 35.2 provides details of these nonconstant diffusivity schemes. The remainder of this

section is devoted to the constant diffusivity schemes.

35.1.3 Summary of the options and namelist parameters

Option *isoneutralmix* is referred to as a *hybrid* mixing scheme since it only mixes tracers, not momentum. One basic horizontal mixing scheme (e.g. option *consthmix*) and one basic vertical mixing scheme (e.g. option *constvmix*) must also be enabled for use with option *isoneutralmix*. Options*redi_diffusion*, *gent_mcwilliams*, and *biharmonic_rm* are the three main sub-options to option *isoneutralmix*. Enabling either of these three sub-options will automatically enable *isoneutralmix*. There is one other sub-option, *diff_nonconstant*, which is described in Section 35.2.

For conceptual orientation, the combined mixing tensor which defines the default tracer fluxes arising from *redi_diffusion*, *gent_mcwilliams*, and *biharmonic_rm* takes the form

$$J^{mn} = \begin{pmatrix} A_I & 0 & (A_I - \kappa + B\,\nabla_h^2)\,S_x \\ 0 & A_I & (A_I - \kappa + B\,\nabla_h^2)\,S_y \\ (A_I + \kappa - B\,\nabla_h^2)\,S_x & (A_I + \kappa - B\,\nabla_h^2)\,S_y & (A_I\,S^2 + A_D) \end{pmatrix}, \tag{35.1}$$

where A_I is the along isoneutral diffusivity, κ is the GM90 diffusivity, B is the RM98 biharmonic diffusivity, S_x and S_y are the isoneutral slopes in the x and y directions, respectively, $S^2 = S_x^2 + S_y^2$, A_D is the dianeutral diffusivity, approximated here as vertical diffusion, and ∇_h^2 is the horizontal Laplacian operator. This tensor represents the sum of the symmetric small angle isoneutral diffusion tensor, the anti-symmetric GM90 tensor, and the anti-symmetric RM98 tensor. The central goals of the numerical algorithms are to discretize the tracer fluxes arsing from this tensor in a numerically stable and physically consistent fashion. It is believed that the current algorithms in MOM provide a workable approximation to these two goals.

1. Option *redi_diffusion* enables the Redi isoneutral diffusion process. There are two sub-options for *redi_diffusion*.

   - Option *small_tensor* (Section 35.1.5) is the default version of *redi_diffusion*. It employs the small angle approximated tensor first written down by Gent and McWilliams (1990). This approximation is discussed further in Griffies *et al.* (1998).
   - Option *full_tensor* employs the full, unapproximated isoneutral diffusion tensor of Redi (1982). This scheme requires roughly 5-10 times more computational time than *small_tensor*; it is not compatible with the default option *gm_skew*; it is not compatible with option *diff_nonconstant*, and it has questionable physical relevance in mixed layer regions (see Section 35.1.9). Therefore, option *full_tensor* is *not* recommended. The code for the full tensor remains frozen and so will not be made compatible with future options, unless trivially done. It remains in MOM, nevertheless, since the full tensor code is cleanly isolated, and it is of potential interest to those comparing the results from different ways to handle the steep sloped regions.

   For the small tensor, and for most parameter ranges of the full tensor, some form of tapering is required in order to scale to zero the along isoneutral diffusivity as the slope increases. There are two options for implementing this tapering.

   - Option *dm_taper* (section 35.1.9.1) uses the scheme of Danabasoglu and McWilliams (1996).
   - Option *gkw_taper* (Section 35.1.9.2) uses the scheme of Gerdes *et al.* (1991). This is the default option.

If one uses a mixed layer scheme such as option *kppvmix* (Section 33.2.3), it might be useful to compute horizontal tracer fluxes within the boundary layer using the same horizontal diffusivity as used to compute vertical tracer fluxes. Option *isotropic_mixed* (Section 35.1.9.3) enables this choice to be made in MOM.

The namelist parameters (refer to Section 14.4 for information on namelist variables) for *redi_diffusion* are the following:

- *ahisop* is the along isoneutral diffusion coefficient. The default value is $10^7 cm^2/sec$. *ahisop* is identical to A_I in the mixing tensor shown in equation (35.1).
- *slmx* is the maximum isoneutral slope. The default value is 0.01. For slopes with magnitudes larger than $slmx$, one of the tapering schemes *dm_taper* or *gkw_taper* is used.
- *ahsteep* is the horizontal diffusion coefficient which kicks in whenever the isoneutral slopes are larger than *slmx*. The default value is $ahsteep = ahthk$. When one of the nonconstant diffusivity schemes described in Section 35.2 is used, *ahsteep* defaults to the value of the non-constant skew-diffusivity. This default for the horizontal diffusion in steep regions is based on the ideas in Treguier *et al.* (1997).
- *del_dm* is the transition for scaling isoneutral diffusivities with option *dm_taper*. The default value is 0.004.
- *s_dm* is the half width scaling for diffusivity with option *dm_taper*. The default is 0.001.

In order to obtain the standard small slope tensor using the Gerdes *et al.* (1991) form for tapering, the option *redi_diffusion* is all that is needed. Default options *small_tensor* and *gkw_taper* are automatically enabled. Setting the alternatives will override these defaults.

2. Option *gent_mcwilliams* enables the GM90 adiabatic stirring process. There are two sub-options for *gent_mcwilliams*.

   - Option *gm_skew* (Section 35.1.6.1) implements GM90 in terms of the skew-flux approach described in Griffies (1998). This is the default option.
   - Option *gm_advect* (Section 35.1.6.2) implements GM90 in terms of the advective-flux approach discussed in Danabasoglu and McWilliams (1996).

   The namelist parameter for *gent_mcwilliams* is the following:

   - *athkdf* is the GM90 diffusion coefficient. The default value is $10^7 cm^2/sec$. *athkdf* is identical to κ in the mixing tensor shown in equation (35.1).

   Just as for the isoneutral diffusion scheme, it is necessary to scale the GM90 diffusivity to zero as the isoneutral slope increases. The same options, with the same maximum slope *slmx*, are used here as for isoneutral diffusion.

   In order to obtain the standard *gent_mcwilliams* scheme, employing skew-fluxes and the Gerdes *et al.* (1991) form for tapering, the option *gent_mcwilliams* is all that is needed. Default options *gm_skew* and *gkw_taper* are automatically enabled. Setting the alternatives will override these defaults.

3. Option *biharmonic_rm* (Section 35.1.8) enables the RM98 adiabatic biharmonic operator in terms of skew-fluxes. There are no sub-options for *biharmonic_rm*. The namelist parameter is

- *abihrm* is the RM98 diffusion coefficient. The default value is $10^{19} cm^4/sec$. *abihrm* is identical to B in the mixing tensor shown in equation (35.1).

Just as for the isoneutral diffusion scheme, it is necessary to scale the RM98 diffusivity to zero as the isoneutral slope increases. The same options, with the same maximum slope *slmx*, are used here as for isoneutral diffusion.

4. Option *diff_nonconstant* enables one of the mesoscale eddy closures for use in producing an eddy diffusivity which is a function of space and time. This option is described in Section 35.2.

35.1.4 Some caveats and comments

Each of the above mentioned schemes has been implemented in MOM so that zero horizontal background diffusion is required; i.e., the schemes are stable. This situation is in contrast to the Cox (1987) isoneutral diffusion scheme which required roughly $10^6 - 10^7 cm^2/sec$ horizontal background diffusion; otherwise, the solution blew-up (Griffies *et al.* 1998). In summary, the new diffusion scheme does the following:

1. It produces a zero flux of locally referenced potential density. The Cox (1987) scheme did not respect this property, and this led to the instability of that scheme.

2. It reduces tracer variance, and produces downgradient oriented tracer fluxes when considering a particular finite volume.

3. It computes the best approximation to the neutral directions within the limitations of the discrete lattice.

4. It requires zero background diffusion to remain stable; the Cox (1987) scheme blew-up without this diffusion.

Unfortunately, there is no constraint with the new diffusion or skew-diffusion schemes equivalent to the positive definiteness possessed by certain advection schemes, such as FCT (Section 32.6). As such, it is possible to realize unphysical tracer values even though the schemes are numerically stable. Experience has shown that most problems occur with passive tracers when employing isoneutral diffusion without skew-diffusion. Additionally, the problems occur most readily in regions of steep isoneutral slopes. The reason that steep sloped regions (say slopes greater than 1/100 or so) are most problematical relates to the inability to resolve these regions with grid aspect ratios typically on the order of 1/1000. Namely, the three components to the tracer flux may not properly balance to provide a strictly downgradient diffusion and cross-gradient skew-diffusion. The active tracers, as they are constrained to preserve the locally defined potential density, appear less problematic than the passive tracers. Also, depending on the value of the thickness diffusivity, GM90 can reduce the problems since it acts to reduce slopes. At this time, there are no feasible alternatives to the current methods of implementing these schemes in MOM which may overcome this difficulty with positive definiteness. The note by Beckers *et al.* (1998) provides some useful speculation.

The numerical problems with isoneutral mixing in steep sloped regions prompts one to assess how far the physics indicates that steep sloped regions need to preserve the along isoneutral nature of the diffusion. Measurements clearly indicate that the use of $10^6 - 10^7 cm^2/sec$ globally applied horizontal background diffusion is not justified due to its huge dianeutral

nature in regions of even modest slopes. However, in regions of steep slopes, it is arguable (e.g., Treguier *et al.* 1997) that the mesoscale eddy processes are transporting tracers across the steeply sloped mean density surfaces. Additionally, such slopes are typically associated with boundary layer processes, which are far from adiabatic. These two points indicate that a fair amount of horizontally aligned background diffusion might be warranted in steep sloped regions. As mentioned above, MOM allows for the use of such mixing in steep sloped regions (Section 35.1.9 provides further details). As might be expected, adding some form of horizontal diffusion in steep sloped regions ameliorates many of the numerical problems.

Another caveat concerns the ubiquitous problems with dispersive advection schemes. Upon stabilizing the isoneutral diffusion scheme, it is now possible to remove the horizontal background diffusion previously required with the Cox (1987) scheme. In so doing, however, one is exposed to problems with dispersive advection schemes. These problems are most apparent next to topography in regions where the tracers are aligned with isoneutrals, and so diffusive fluxes are weak. This "Peclet grid noise" problem is fundamental to dispersive advection. It is *not* a problem with the rotated diffusion.

35.1.5 redi_diffusion

Now that the various isoneutral mixing schemes and model options have been summarized, the following sections provide more details and discussions. First, this section quotes the discretized isoneutral diffusion flux in the special case of the small angle approximation (the default option *small_tensor*). Complete details for the derivation and explanation of the labels are presented in Griffies *et al.* (1998) and Appendix C. The results for the full tensor are provided there as well.

Note that the discretizations given below for all the isoneutral mixing schemes assume the use of MOM's default grid construction. The desired numerical properties are rigorously not maintained for the grid option *centered_t* which always centers grid points within T-cells as in MOM 1. However, preliminary idealized tests (flat bottom) indicate no problem using option *centered_t*. Additionally, the isoneutral mixing schemes have been coded for either the full or partial vertical cells (see Chapter 26 for details of the partial vertical cells).

35.1.5.1 Zonal isoneutral diffusion flux

The component of the isoneutral diffusive flux through the eastern face of a T cell $diff_fet^{iso}_{i,k,j}$ for the small angle Redi tensor is given by

$$\begin{aligned} -F^{x}_{i,k,j} &= diff_fet^{iso}_{i,k,j} = K^{11\,small}_{i,k,j}\ \delta_x T_{i,k} \\ + \quad \frac{1}{4}\sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\ Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)} \delta_z T_{i+ip,k-1+kr}, \end{aligned} \tag{35.2}$$

where the non-negative diagonal component to the small angle isoneutral diffusion tensor is given by

$$K^{11\,small}_{i,k,j} = \frac{1}{4}\sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}. \tag{35.3}$$

The contribution of this flux to the diffusion operator is given by

$$R^x(T)_{i,k,j} = \left(\frac{1}{dht_{i,k,j}}\right)\left(\frac{\delta_x(diff_fet^{iso}_{i-1,k,j})}{\cos\phi^T_{jrow}}\right). \tag{35.4}$$

Note that certain irrelevant latitudinal j labels were omitted for brevity.

As written, it appears necessary to dimension two arrays: $Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ and $Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$. However, it turns out that it is possible code this flux so that only one array, $(Ax * Sx)^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ is dimensioned. This trick is also employed for discretizing the other flux components, as well as for the *gent_mcwilliams* and *biharmonic_rm* schemes. However, it cannot be employed for *full_tensor*. In the following, this approach will be implicit in the expressions written for the fluxes. This approach was not used in the original MOM 2 implementation of isoneutral diffusion. Instead, the slopes were computed both in the subroutine which determined the diffusivities and in the subroutine which determined the flux components. This double computation of the slopes was chosen instead of saving the slopes inside of an array dimensioned as the diffusivity and hence incurring an increase in memory cost. The new approach uses the same memory, but eliminates the second calculation of the slope. A test with the flat bottom sector model used in Griffies *et al.* (1998), using two active tracers and the small angle isoneutral diffusion process, realized a 15% reduction in total model runtime with the new code over the MOM 2 code. The relative savings will clearly increase upon using more tracers.

35.1.5.2 Meridional isoneutral diffusion flux

The component of the isoneutral diffusive flux through the northern face of a T cell $diff_fnt^{iso}_{i,k,j}$ for the full Redi tensor is given by

$$\begin{aligned} -F^y_{i,k,j} = diff_fnt^{iso}_{i,k,j} &= \cos\phi^U_j\, K^{22\,small}\, \delta_y T_{k,j} \\ + \quad \frac{\cos\phi^U_j}{4\cos\phi^T_j} \sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{jq=0}^{1} Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)} &\times \\ \left(Sy^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\, \delta_z T_{k-1+kr,j+jq}\right). \end{aligned} \tag{35.5}$$

where the diagonal component to the small angle isoneutral diffusion tensor is given by

$$K^{22\,small}_{i,k,j} = \frac{1}{4\cos\phi^T_j} \sum_{kr=0}^{1} \sum_{jq=0}^{1} \left(\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\, Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\right). \tag{35.6}$$

The contribution of this flux to the diffusion operator is given by

$$R^y(T)_{i,k,j} = \left(\frac{1}{dht_{i,k,j}}\right)\left(\frac{\delta_y(diff_fnt^{iso}_{i,k,j-1})}{\cos\phi^T_{jrow}}\right) \tag{35.7}$$

Note that certain irrelevant longitudinal i labels were omitted for brevity.

35.1.5.3 Vertical isoneutral diffusion flux

The component of the isoneutral diffusive flux through the bottom face of a T cell $diff_fbt^{iso}_{i,k,j}$ is broken into two parts: the $K^{33}_{i,k,j}$ component and the off diagonal component $diff_fbiso_{i,k,j}$ which contains $K^{31}_{i,k,j}$ and $K^{32}_{i,k,j}$ pieces. The vertical diffusion operator term for tracers is also broken into two parts for isoneutral diffusion. First, the part containing $diff_fbiso_{i,k,j}$ is solved explicitly with all other explicit components. Second, the part containing the $K^{33}_{i,k,j}$ component is solved implicitly along with any other vertical diffusivity piece arising from dianeutral diffusion. The use of an implicit solver for the $K^{33}_{i,k,j}$ term allows for steeper neutral direction slopes to be handled within the constraints of the linear diffusion equation stability (see Cox 1987 and Griffies *et al.* 1997 for details). In the model, the vertical component is written

$$
\begin{aligned}
& -F^z_{i,k,j} = K^{33}_{i,k,j}\delta_z T_{i,k,j} + diff_fbiso_{i,k,j} = K^{33}_{i,k,j}\delta_z T_{i,k,j} \\
+ \quad & \frac{1}{4dxt_i\, dht_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}\, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \\
& \times Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}\, \delta_x T_{i-1+ip,k+kr} \\
+ \quad & \frac{1}{4\cos\phi^T_j dyt_j\, dht_{i,k,j}} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)}\, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
& \times Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}\, \delta_y T_{k+kr,j-1+jq}. \qquad (35.8)
\end{aligned}
$$

where the non-negative diagonal component to the small angle isoneutral diffusion tensor is given by

$$
\begin{aligned}
K^{33\,small}_{i,k,j} \quad = \quad & \frac{1}{4dxt_i\, dht_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \times \\
& \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}\, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}\, (Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)})^2 \\
+ \quad & \frac{1}{4\cos\phi^T_j dyt_j\, dht_{i,k,j}} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \times \\
& \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)}\, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}\, (Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)})^2, \qquad (35.9)
\end{aligned}
$$

The contribution of this flux to the diffusion operator is given by

$$
R^z(T)_{i,k,j} = \delta_z(diff_fbt^{iso}_{i,k-1,j}). \qquad (35.10)
$$

Note that certain irrelevant longitudinal i and latitudinal j labels were omitted for brevity.

35.1.6 gent_mcwilliams

There are two methods for implementing the GM90 scheme: a skew-flux approach, which is the default, and an advective flux approach. This section provides a summary of the basic issues.

35.1.6.1 gm_skew

The addition of a divergence-free eddy-induced velocity to the tracer equation is equivalent to adding an anti-symmetric component to the tracer mixing tensor. This equivalence is well known. Griffies (1998) argued that for the purpose of numerically implementing GM90, the use of its anti-symmetric tensor is preferred to the use of eddy-induced velocities. The advantages over the use of the advective formulation are basically the following:

1. One less spatial differentiation of the slope and diffusivity is required with the skew-flux approach. Spatial differentiation of these objects can be numerically problematical, especially in regions of steep isoneutral slopes where both the slope, and the possibly spatially dependent diffusivity, are rapidly changing.

2. It exploits the work of Griffies *et al* (1998) for implementing isoneutral diffusion. That framework makes for a trivial implementation of GM90 in which the tracer variance is conserved.

3. The skew-flux is faster computationally than the advective flux.

4. When setting the diffusivities A_I and κ equal, the off-diagonal terms in the horizontal flux drop out identically, unless option *biharmonic_rm* is also enabled. This relative setting of the diffusivities is commonly employed in the literature (e.g., Danabasoglu and McWilliams 1996, Hirst and McDougall 1996). It should be noted that researchers are currently suggesting that the GM90 diffusivity be set to roughly 1/2 to 1/4 that of the along isoneutral diffusivity. Regardless of the relative setting of the diffusivities, the skew-flux approach is preferable.

The implementation of option *gm_skew* is trivial once the implementation of the small slope isoneutral diffusion process has been done. All that is necessary is to modify the diffusivities appearing in the off-diagonal terms already used in the isoneutral diffusion scheme. Note that all skew-flux components are integrated explicitly in time, since there is no diagonal contribution to the vertical tracer flux component.

Regardless of the numerical preference for the skew-flux approach, it is useful diagnostically to map the GM90 eddy induced velocities. For this purpose, anytime option *snapshots* is enabled, these GM90 velocities are computed using the algorithm from option *gm_advect* and placed in the snapshots file.

35.1.6.2 gm_advect

In this approach, the GM90 parameterization is implemented in terms of an eddy-induced transport velocity. The implementation of the eddy-induced velocity in MOM is different than what is described in Danabasoglu and McWilliams (1996). Most notably, a computational mode (see Appendix E for a disucssion of computational modes), which was related to the original Cox (1987) implementation of the isoneutral diffusion, has been removed. Additionally, MOM employs a reference level for every depth level, rather than the reduced number of reference levels originally employed in the Danabasoglu and McWilliams code. As described in Section 35.1.6.1, the advection velocity approach is not as computationally efficient as the skew-flux approach. Therefore, *gm_advect* is retained solely for the comparative and diagnostic purposes. Consequently, the code for *gm_advect* is basically frozen, and future implementations of eddy stirring processes (e.g., *biharmonic_rm*) will be made using the skew-flux approach.

The eddy-induced velocities, as with the regular advection velocities in MOM, are computed at the centers of the eastern, northern, and bottom faces of the cells. The velocities are given by $adv_vetiso_{i,k,j}$, $adv_vntiso_{i,k,j}$, and $adv_vbtiso_{i,k,j}$ respectively. In MOM 2 version 1, the eddy induced transport velocities were discretized based on the notes of Gokhan Danabasoglu as

$$adv_vetiso_{i,k,j} = -\delta_z(\overline{\frac{\kappa}{A_I}[K^{13}]_{i,k-1,j}}^z) \tag{35.11}$$

$$adv_vntiso_{i,k,j} = -\delta_z(\overline{\frac{\kappa}{A_I}[K^{23}]_{i,k-1,j}}^z) \tag{35.12}$$

However, the above form contains a null mode and has been replaced by the following

$$adv_vetiso_{i,k,j} = -\delta_z(\kappa \cdot S^{xb}_{i,k-1,j}) \tag{35.13}$$

$$adv_vntiso_{i,k,j} = -\delta_z(\kappa \cdot S^{yb}_{i,k-1,j}) \tag{35.14}$$

where the isoneutral slope in the zonal direction at the bottom of the eastern face of a T grid cell is given by

$$S^{xb}_{i,k,j} = -\frac{\overline{\alpha_{i,k,j}}^{\lambda,z}\overline{\delta_\lambda(t_{i,k,j,1,\tau-1})}^z + \overline{\beta_{i,k,j}}^{\lambda,z}\overline{\delta_\lambda(t_{i,k,j,2,\tau-1})}^z}{\overline{\alpha_{i,k,j}}^{\lambda,z}\overline{\delta_z(t_{i,k,j,1,\tau-1})}^\lambda + \overline{\beta_{i,k,j}}^{\lambda,z}\overline{\delta_z(t_{i,k,j,2,\tau-1})}^\lambda} \tag{35.15}$$

and the neutral slope in the meridional direction at the bottom of the northern face of a T grid cell is given by

$$S^{yb}_{i,k,j} = -\frac{\overline{\alpha_{i,k,j}}^{\phi,z}\overline{\delta_\phi(t_{i,k,j,1,\tau-1})}^z + \overline{\beta_{i,k,j}}^{\phi,z}\overline{\delta_\phi(t_{i,k,j,2,\tau-1})}^z}{\overline{\alpha_{i,k,j}}^{\phi,z}\overline{\delta_z(t_{i,k,j,1,\tau-1})}^\phi + \overline{\beta_{i,k,j}}^{\phi,z}\overline{\delta_z(t_{i,k,j,2,\tau-1})}^\phi} \tag{35.16}$$

where the $\alpha_{i,k,j}$ and $\beta_{i,k,j}$ are defined as in Section C.2.7.

The vertical component of the eddy-advection velocity is obtained by vertically integrating the divergence of the horizontal eddy-advection velocities as is done in the notes of Gokhan Danabasoglu.

$$adv_vbtiso_{i,k,j} = \frac{1}{\cos\phi^T_{jrow}}\sum_{m=1}^{k}\Big[\delta_\lambda(adv_vetiso_{i-1,m,j}) + \delta_\phi(adv_vntiso_{i,m,j-1})\Big]\cdot dzt_m \tag{35.17}$$

Note that traditionally there is a zero vertical eddy-advection velocity at the top face of $cell_{i,k=1,j}$ and bottom face of $cell_{i,k=bottom,j}$. This boundary condition on the velocity effectively places a boundary condition on the diffusivity κ (e.g., see discussion in Treguier *et al* 1997).

noindent The eddy-induced advection terms are discretized as:

$$\begin{aligned}\mathcal{L}^{gm}(\gamma_{i,k,j}) = \frac{1}{\cos\phi^T_{jrow}}\Big[\ & \delta_\lambda(adv_vetiso_{i-1,k,j}\overline{\gamma_{i-1,k,j}}^\lambda) + \delta_\phi(adv_vntiso_{i,k,j-1}\overline{\gamma_{i,k,j-1}}^\phi)\Big] \\ - \quad & \delta_z(adv_vbtiso_{i,k-1,j}\overline{\gamma_{i,k-1,j}}^z)\end{aligned} \tag{35.18}$$

where $adv_vntiso_{i,k,j-1}$ contains an embedded cosine factor as does $adv_vnt_{i,k,j}$. Refer to Section 22.8.7 for a definition of the advective operator.

35.1.7 Linear numerical stability for Redi and GM

Numerically realizing isoneutral diffusion along steep isoneutral slopes is difficult partly because of the small vertical to horizontal aspect ratio in the ocean and hence the ocean model grid. As isoneutral slopes steepen, the projection of diffusive fluxes onto the vertical become stronger, pushing up against the limits of the linear stability criteria for the diffusion equation. This issue is relevant for discretizing both the small and full diffusion tensors. In particular, the linear numerical constraint from the diffusion equation, as discussed in Cox (1987) and Griffies *et al.* (1998), indicates that an explicit numerical scheme with a leap-frog time step will be stable if the grid CFL number satisfies

$$\frac{|K^{mn}|\Delta t}{\Delta x_m \Delta x_n} \leq \frac{1}{4}, \tag{35.19}$$

where K^{mn} are the components of the diffusion tensor $\mathbf{K}$, and there is no sum implied in this expression. This stability constraint is also relevant for the Gent-McWilliams scheme when implemented according to the skew-flux approach. Similar stability constraints hold for the advective-flux approach.

Assuming a geophysically relevant vertical to horizontal aspect ratio for the grid ($\Delta x/\Delta z \leq 1/1000$), the two dimensional horizontal sub-matrix is stable when the diffusion equation in the horizontal is stable. In general, satisfying this stability constraint in the horizontal is trivial and so will not be considered further. Solving the vertical K^{zz} diagonal piece implicitly, as done by Cox (1987), points to the K^{xz} and K^{yz} cross terms as setting the most restrictive constraint. From these terms, the diffusion equation using the fluxes from the full tensor will be linearly stable when, for each grid cell,

$$\frac{|S_a|}{1 + S_x^2 + S_y^2} \leq \frac{\Delta a \, \Delta z}{4 A_I \Delta t} \equiv \delta, \tag{35.20}$$

where a is either x or y. The small tensor's stability is determined with the $1 + S_x^2 + S_y^2$ denominator set to unity. For the present discussion, the small slope approximation is made. Linear stability for the full tensor is further discussed in Griffies *et al.* (1998). For the small angle tensor, δ represents the maximum allowable slope which can be used before some prescription must be employed to ensure numerical stability. For many large-scale ocean model configurations, this slope check parameter is roughly 1/100, thus providing for the self-consistency of the discretization of the small slope tensor. The prescriptions for maintaining stability are discussed in Section 35.1.9.

This analysis is based on the conservative assumption that if all components to the diffusion tensor produce linearly stable diffusion, then the scheme is linearly stable. Although conservative, experience has shown that violation of these constraints can result in unacceptably large numerical inaccuracies. These inaccuracies are of special concern since they make it more difficult to realize the balance $\alpha F_I^z(\theta) = \beta F_I^z(s)$, thus exposing the solution to the nonlinear instability discussed in Griffies *et al.* (1998).

35.1.8 biharmonic_rm

Roberts and D. Marshall (1998) (RM98) proposed a new biharmonic operator for use in z-level models. Ideally, this operator adiabatically dissipates structures at the grid scale. This property is useful for both eddy permitting models, where enstrophy cascades to the small scales

and so must be dissipated, and coarse models, where problems with advection schemes can pollute the solution with spurious noise. This operator is motivated for numerical reasons, not from any fundamental arguments. Stated quite simply, the goal is to enhance the advective nature of the simulation without sacrificing adiabaticity. Without some way to suppress the grid noise, an advectively dominant solution can become a sea of noise. If the manner to suppress the grid noise does not take into account the needs of adiabaticity, such as the traditional horizontal biharmonic operator or dissipative advection schemes, then the solution may also be of low value due to the loss of water mass integrity. These competing needs are potentially addressed by this new operator.

RM98 termed their operator a "biharmonic GM" operator since it represents a straightforward generalization of the usual "Laplacian GM" operator. Yet, as shown by RM98 and in the following, their biharmonic operator does not generally dissipate APE, whereas the usual GM operator does. Therefore, the RM98 operator is perhaps better considered one of the many possible adiabatic biharmonic operators. Indeed, an alternative adiabatic operator, which always dissipates APE, is mentioned in the following. It turns out, however, that the RM98 operator is more readily discretized using the triad approach already used for the Redi diffusion (Section 35.1) and GM skew-diffusion (Section 35.1.6) processes.

35.1.8.1 The RM98 operator

The additional "eddy-induced" velocity proposed by RM98 has the components

$$\mathbf{u}_h^* = \partial_z \left(B \, \nabla_h^2 \mathbf{S} \right) \tag{35.21}$$

$$w^* = -\nabla_h \cdot \left(B \, \nabla_h^2 \mathbf{S} \right), \tag{35.22}$$

where

$$\mathbf{S} = -\left(\frac{\nabla_h \rho}{\rho_z} \right) \tag{35.23}$$

is the isoneutral slope vector, ρ is the locally referenced potential density, and $B \geq 0$ is the biharmonic dissipation coefficient with units of $length^4/time$. The realization of this dissipation can readily be made through the skew-flux approach of Griffies (1998), as discussed in Appendix B and Section 35.1.6.1. The anti-symmetric tensor corresponding to the RM98 advection velocity takes the form

$$\mathbf{A} = [A^{mn}] = \begin{pmatrix} 0 & 0 & B \, \nabla_h^2 S_x \\ 0 & 0 & B \, \nabla_h^2 S_y \\ -B \, \nabla_h^2 S_x & -B \, \nabla_h^2 S_y & 0 \end{pmatrix}, \tag{35.24}$$

and the components to the corresponding skew-flux components for an arbitrary tracer are

$$\mathbf{F}_h = -\left(B \, \nabla_h^2 \mathbf{S} \right) T_z \tag{35.25}$$

$$F^z = \left(B \, \nabla_h^2 \mathbf{S} \right) \cdot \nabla_h T. \tag{35.26}$$

In general, the skew-flux for a particular tracer is directed normal to the gradient of that tracer

$$\nabla T \cdot \mathbf{F}(T) = 0. \tag{35.27}$$

Notably, this result holds when T is the locally referenced potential density ρ, which reflects the adiabatic nature of the scheme. The manner in which this continuum result is implemented numerically is discussed in Griffies (1998). That approach ensures that the numerical scheme does not alter the tracer mean and variance. It is therefore conservative in this sense.

35.1.8.2 RM98 for a special vertical profile

Consider the GM90 skew-flux discussed in Griffies (1998)

$$\mathbf{F}_h = \kappa \, \mathbf{S} \, T_z \tag{35.28}$$
$$F^z = -\kappa \, \mathbf{S} \cdot \nabla_h T. \tag{35.29}$$

Recall that in the special case of a linear equation of state, the GM90 density skew-flux takes the especially simple form

$$\mathbf{F}_h(\rho) = -\kappa \, \nabla_h \rho \tag{35.30}$$
$$F^z(\rho) = \kappa \, S^2 \rho_z. \tag{35.31}$$

With a stable density profile, $\rho_z < 0$, which means that the vertical skew-flux component is always negative. In general, the horizontal GM90 skew-flux components are directed down the density gradient, and the vertical component is upgradient.

With an always upgradient flux of density, the GM90 scheme always decreases the potential energy in the stably stratified fluid. This property is not generally respected by RM98, as discussed in Section 35.1.8.3. However, it is useful to consider a case in which these properties are shared for the purpose of illustrating the biharmonic nature of the RM98 scheme. One such profile is given by

$$\rho = \rho_o(z) + \rho_1 \cos(\mathbf{p} \cdot \mathbf{x}), \tag{35.32}$$

where $\rho_o(z)$ is some stable mean vertical profile, ρ_1 is a (possibly time dependent) amplitude function, and $\mathbf{p} = (p_x, p_y, 0)$ is a horizontal wave-vector. The slope vector for this profile is given by

$$\mathbf{S} = \frac{\rho_1 \, \mathbf{p} \, \sin(\mathbf{p} \cdot \mathbf{x})}{\rho_o'(z)}, \tag{35.33}$$

and the horizontal Laplacian is

$$\nabla_h^2 \mathbf{S} = -p^2 \, \mathbf{S}, \tag{35.34}$$

where $p^2 = \mathbf{p} \cdot \mathbf{p}$. The RM98 skew-flux therefore takes the form

$$\mathbf{F}_h = (B \, p^2) \, \mathbf{S} \, T_z \tag{35.35}$$
$$F^z = -(B \, p^2) \, \mathbf{S} \cdot \nabla_h T. \tag{35.36}$$

The RM98 skew-flux of density, linearly dependent on temperature, is given by

$$\mathbf{F}_h(\rho) = -(B \, p^2) \, \nabla_h \rho \tag{35.37}$$
$$F^z(\rho) = (B \, p^2) \, S^2 \, \rho_z. \tag{35.38}$$

As such, just as for the GM90 case which holds in general, the horizontal RM98 skew-flux components for density are down the horizontal density gradient, and the vertical skew-flux component is up the vertical density gradient. The effective diffusivity, however, is scale-dependent in the RM98 case, with small scales, or large p^2, acted on with the largest effective diffusivity. It is this sort of behaviour which is characteristic of a biharmonic mixing scheme.

35.1.8.3 Effects on potential energy of the RM98 operator

Now consider how the RM98 operator affects the potential energy for the case when density is a linear function of potential temperature. A similar discussion was given in the RM98 paper, where they employ the advective flux formulation rather than the skew-flux formulation. A few speculative remarks are added here as well. Focusing just on the biharmonic operator, the time tendency for potential energy is given by

$$\begin{aligned} P_t &= g \int d\mathbf{x}\, z\, \rho_t \\ &= -g \int d\mathbf{x}\, z\, \nabla \cdot \mathbf{F} \\ &= g \int d\mathbf{x}\, F^z, \end{aligned} \tag{35.39}$$

where the no-normal flux condition at the sides was assumed. Note that for simplicity, a rigid lid was also assumed, although this assumption has no bearing on the effects the $g \int d\mathbf{x}\, F^z$ term has on the evolution of the total potential energy. Using the RM98 skew-diffusion leads to

$$\begin{aligned} \int d\mathbf{x}\, F^z &= -\int d\mathbf{x}\, B\, \rho_z\, \mathbf{S} \cdot \nabla_h^2 \mathbf{S} \\ &= -\int d\mathbf{x}\, \nabla_h \cdot (B\, \rho_z\, S_i\, \nabla_h S_i) + \int d\mathbf{x}\, \nabla_h (B\, \rho_z\, S_i) \cdot \nabla_h S_i, \end{aligned} \tag{35.40}$$

where $i = 1, 2$ is summed. Assuming the isopycnal slopes vanish next to the lateral ocean boundaries allows for the total derivative term to be dropped. The result is

$$\int d\mathbf{x}\, F^z = (1/2) \int d\mathbf{x}\, \nabla_h (B\, \rho_z) \cdot \nabla_h S^2 + \int d\mathbf{x}\, B\, \rho_z\, (\partial_j \mathbf{S} \cdot \partial_j \mathbf{S}), \tag{35.41}$$

where $S^2 = \mathbf{S} \cdot \mathbf{S}$, and $j = 1, 2$ is summed in the second term over the horizontal spatial dimensions. The second term is non-positive in stably stratified fluids, for which $\rho_z \leq 0$. It therefore represents a potential energy sink. The first term, however, is sign indefinite even when B is a constant. For the special case of $B\, \rho_z$ independent of horizontal position, the first term vanishes and so potential energy is reduced. For example, the special density profile considered previously $\rho = \rho_o(z) + \rho_1 \cos(\mathbf{p} \cdot \mathbf{x})$ has $B\, \rho_z$ horizontally constant if B is constant, and so the potential energy is reduced. In the slightly more general case of constant B and with $\rho_z = \rho_z^0(z) + \rho_z^1(\mathbf{x})$, where $|\rho_z^0| >> |\rho_z^1|$, the first term in the expression for potential energy is nonzero, but subdominant to the second term. So potential energy is again reduced in this case. It is unclear what happens in the more general case.

It might be speculated that the inability to prove that the potential energy is generally reduced may indicate that the RM98 operator is unstable. However, if numerically implemented according to the triad approach of Griffies *et al.* (1998), the discretized RM98 skew-fluxes will preserve the density variance. As variance growth is typically associated with linearly unstable numerical schemes, any potential instability of the RM98 scheme will likely be nonlinear. So far, no such instabilities have been encountered. Rather, the operator appears to be quite stable in a wide suite of both coarse and fine models.

35.1.8.4 Effects on potential energy of an operator suggested by Gent

An alternative operator suggested by Peter Gent (personal communication; reported in RM98) is derived from the velocities

$$\mathbf{U}_h^* = -\partial_z \left[\nabla_h \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) \right] \tag{35.42}$$

$$W_* = \nabla_h^2 \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right). \tag{35.43}$$

Since the operator does not maintain the integrity of the slope vector $\mathbf{S} = -\nabla_h \rho / \rho_z$, its implementation in MOM is not as simple as the RM98 operator.

The anti-symmtric tensor corresponding to the Gent biharmonic operator is given by

$$\mathbf{A} = [A^{mn}] = \begin{pmatrix} 0 & 0 & -\partial_x \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) \\ 0 & 0 & -\partial_y \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) \\ \partial_x \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) & \partial_y \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) & 0 \end{pmatrix}, \tag{35.44}$$

and the components to the skew-flux for an arbitrary tracer takes the form

$$\mathbf{F}_h = \nabla_h \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) T_z \tag{35.45}$$

$$F^z = -\nabla_h \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) \cdot \nabla_h T. \tag{35.46}$$

The effects on the time tendency of potential energy arising from this skew-flux take the form

$$\begin{aligned} \int d\mathbf{x}\, F^z &= -\int d\mathbf{x}\, \nabla_h \left(B \frac{\nabla_h^2 \rho}{\rho_z} \right) \cdot \nabla_h \rho \\ &= \int d\mathbf{x}\, \nabla_h \cdot (B\, \mathbf{S}\, \nabla_h^2 \rho) + \int d\mathbf{x}\, (B/\rho_z) \left(\nabla_h^2 \rho \right)^2. \end{aligned} \tag{35.47}$$

The first term can be dropped upon assuming the isoneutral slopes vanish at the horizontal boundaries. The second term is non-positive in stably stratified water, and so represents a sink for potential energy for any density profile.

The Gent operator cannot naively be discretized using the triad technology employed by Griffies *et al.* (1998) or Griffies (1998). The reason is that the slope vector is not a fundamental piece of the Gent operator, whereas it fundamental for isoneutral diffusion, and GM90 and RM98 skew-diffusion. Indeed, results from a straightforward discretization of the Gent operator indicate the presence of numerical instabilities (RM98 and personal communication). Further research is necessary to determine the relative merits of the RM98 and Gent operators amongst other possible adiabatic biharmonic operators.

35.1.8.5 A note about spherical coordinates and extra metric terms

In the formulation of Redi diffusion and GM90 skew-diffusion, there is no need to worry about spherical versus Cartesian coordinates. The Cartesian form for the expressions transform trivially to spherical. RM98, however, prescribe a Laplacian acting on the slope vector. On the

sphere, the unit vectors $\hat{\lambda}, \hat{\phi}$ are spatially dependent and so the Laplacian will pick up extra terms[1]. These metric terms are related, though not identical, to the metric terms arising in the dissipation of momentum (a vector) as described in Section 9.8. For the purpose of completeness, it is worth presenting these metric terms, and then discussing why it may make sense to ignore them.

The two-dimensional slope vector can be written in the form

$$\mathbf{S} = S_\lambda \hat{\lambda} + S_\phi \hat{\phi}, \tag{35.48}$$

where the components to the slope are given by

$$S_\lambda = -\frac{1}{a \cos\phi}\left(\frac{\rho_\lambda}{\rho_z}\right) \tag{35.49}$$

$$S_\phi = -\frac{1}{a}\left(\frac{\rho_\phi}{\rho_z}\right). \tag{35.50}$$

The following expression for the horizontal Laplacian acting on a spherical vector can be obtained from Appendix 2 in Batchelor (1967)

$$\begin{aligned} \nabla_h^2 \mathbf{S} &= \hat{\lambda}\left[\left(\nabla_h^2 - \frac{1}{a^2\cos^2\phi}\right) S_\lambda - \left(\frac{2\sin\phi}{a^2\cos^2\phi}\right)\frac{\partial S_\phi}{\partial\lambda}\right] \\ &+ \hat{\phi}\left[\left(\nabla_h^2 - \frac{1}{a^2\cos^2\phi}\right) S_\phi + \left(\frac{2\sin\phi}{a^2\cos^2\phi}\right)\frac{\partial S_\lambda}{\partial\lambda}\right], \end{aligned} \tag{35.51}$$

where

$$\nabla_h^2 \alpha = \left(\frac{1}{a^2\cos\phi}\right)\left(\frac{1}{\cos\phi}\alpha_{\lambda\lambda} + (\cos\phi\,\alpha_\phi)_\phi\right) \tag{35.52}$$

is the horizontal Laplacian acting on a scalar which lives on the sphere. The terms appearing in equation (35.51) in addition to $\nabla_h^2 S_\lambda$ and $\nabla_h^2 S_\phi$ constitute the "metric terms." To see what the metric terms do, it is useful to write the tracer flux with the GM90 scheme included as well

$$\mathbf{F}_h = T_z\left(\kappa + \frac{B}{a^2\cos^2\phi} - B\nabla_h^2 + \left(\frac{2B\sin\phi}{a^2\cos^2\phi}\right)\hat{z}\partial_\lambda \wedge\right)\mathbf{S} \tag{35.53}$$

$$F^z = -\nabla_h T \cdot \left(\kappa + \frac{B}{a^2\cos^2\phi} - B\nabla_h^2 + \left(\frac{2B\sin\phi}{a^2\cos^2\phi}\right)\hat{z}\partial_\lambda \wedge\right)\mathbf{S}. \tag{35.54}$$

In general, the metric terms are smaller than the Laplacian in those cases when the power is concentrated at the grid scale. This is the situation for which the RM98 biharmonic operator is designed. One therefore finds little motivation to include the metric terms. Even so, it is useful to look a bit more closely at how the metric terms contribute to the properties of the operator.

The first metric term, which is proportional to the slope, acts in a manner just like the κ term from GM90. As such, this metric term provides a sign definite sink of potential energy. To gauge the strength of this sink, consider a very high latitude point $\phi = 89°$ and a relatively large diffusivity $B = 10^{20} cm^4/sec$. In this case,

$$\frac{B}{a^2\cos^2\phi} = 8 \times 10^5 cm^2/sec. \tag{35.55}$$

[1]This point was emphasized by Bob Hallberg, 1997.

For the more reasonable $\phi = 45^\circ$ and $B = 10^{19} cm^4/sec$,

$$\frac{B}{a^2 \cos^2 \phi} = 50 cm^2/sec. \tag{35.56}$$

Both of these values should be compared to the usual $\kappa \approx 10^7 cm^2/sec$ GM90 diffusivity. As such, the sink is quite small.

The second metric term, proportional to the zonal derivative of the slope, adds a term to the vertical density flux of the form

$$\begin{aligned} -\left(\frac{2\,B\,\sin\phi}{a^2\,\cos^2\phi}\right) \nabla_h \rho \cdot (\hat{z} \wedge \partial_\lambda \mathbf{S}) &= \left(\frac{2\,B\,\sin\phi}{a^2\,\cos^2\phi}\right) \rho_z\, \mathbf{S} \cdot (\hat{z} \wedge \partial_\lambda \mathbf{S}) \\ &= \left(\frac{2\,B\,\sin\phi}{a^2\,\cos^2\phi}\right) \rho_z\, \hat{z} \cdot (\partial_\lambda \mathbf{S} \wedge \mathbf{S}). \end{aligned} \tag{35.57}$$

This term has no definite sign, and so its effects on potential energy cannot be established in general. As with the constant slope metric term, this term is largest at the high latitudes. To consider its strength, let the slopes have a scale $S_\lambda \approx S_\phi \approx S$, where $|S| \leq 1/100$. Also, let the contributions to the Laplacian due to zonal variations be about the same as the meridional variations: $\partial_{\lambda\lambda} S \approx \cos\phi\, \partial_\phi(\cos\phi\, \partial_\phi S)$. As such, the second metric term is large whenever

$$\left(\frac{\partial^2 S}{\partial \lambda^2}\right) \times \left(2\,\sin\phi\,\frac{\partial S}{\partial \lambda}\right)^{-1} \tag{35.58}$$

is small. Let $\partial_\lambda S \approx S/\Delta\lambda$, and $\partial_\lambda^2 S \approx S/(\Delta\lambda)^2$, where $\Delta\lambda$ is the zonal grid spacing in radians. The question then becomes whether $2\,\sin\phi\,\Delta\lambda$ is larger than one. If it is, then the second metric term is non-negligible. For a $3^\circ = 0.0524$ radian zonal resolution and $\phi \approx 90^\circ$, $2\,\sin\phi\,\Delta\lambda \approx 1/10$. For mid-latitudes and $\Delta\lambda = 1/4^\circ$, $2\,\sin\phi\,\Delta\lambda \approx 1/162$. Both of these results suggests that the second metric term is no more than 10% as large as the Laplacian term, for the cases when the scaling is relavent. Of course, when there is zero curvature in the slope field, then Laplacian vanishes when the metric term may not. But again, such a slope field is perhaps not the kind for which the RM98 scheme is designed to attack.

In summary, the added metric terms do the following:

- The first metric term is proportional to the slope, and it acts in a manner just like GM90. The latitudinally dependent diffusivity setting the scale of this term increases with increasing latitude, with largest values no larger than $10^4 - 10^5 cm^2/sec$ next to the pole. This term is trivial to incorporate into the existing numerical framework from Griffies *et al.* (1998).

- The second term is proportional to the zonal derivative of the slope. This term adds a sign indefinite contribution to the potential energy. It is at most 1/10 the size of the Laplacian term for fields in which the slope curvature is nonzero, with 1/100 being the a more realistic size. This term cannot be discretized using the Griffies *et al.* (1998) numerical framework.

In conclusion, MOM currently ignores the metric terms, and this was also the approach taken by Roberts and Marshall (1998) (M. Roberts, personal communication, 1998). As seen from the above arguments, ignoring these terms is consistent with a desire to act on noise at the grid scale, and to leave the larger scales untouched. Recall that neglecting the analogous (not

identical) metric terms in the second order momentum friction operator leads to unphysical consequences, as discussed in Section 9.3.9 and Wajsowicz (1993). Nevertheless, the angular momentum arguments which guide the form of second order momentum friction are absent for the tracer mixing operators. In the absence of other arguments, there appears little to motivate retaining the metric terms for RM98.

35.1.8.6 Linear numerical stability for the RM98 operator

For linear stability, it is sufficient, and conservative, to consider the numerical stability issues raised by Cox (1987) and Griffies *et al.* (1998) isoneutral diffusion papers. In particular, one requires

$$\frac{(B\,|\nabla_h^2 S_x|)\,\Delta t}{\Delta x\,\Delta z} \leq \frac{1}{4} \tag{35.59}$$

$$\frac{(B\,|\nabla_h^2 S_y|)\,\Delta t}{\Delta y\,\Delta z} \leq \frac{1}{4}. \tag{35.60}$$

For a conservative estimate of the stability, introduce the dimensionless grid factor

$$\delta = \frac{\Delta z\,(\Delta a)^3}{4\,B\,\Delta t}, \tag{35.61}$$

where Δa is the minimum horizontal grid spacing over the extent of the model domain, and Δz is the minimum vertical spacing. Note that Δa should take into account the convergence of the meridions. With this notation, the stability constraint takes the form

$$|S_x| \leq \delta \tag{35.62}$$

$$|S_y| \leq \delta. \tag{35.63}$$

As with the isoneutral diffusion scheme, these constraints place limits on the maximum isoneutral slope that can be realized without introducing some form of tapering to the biharmonic coefficient B. When either component of the slope vector has a magnitude larger than δ, then the tapering of *dm_taper* (section 35.1.9.1) or *gkw_taper* (Section 35.1.9.2; the default) is employed.

An example is useful. Consider a typical mid-latitude channel model with

$$\Delta a = \cos(\pi/4)\,(.25^\circ) \tag{35.64}$$

$$\Delta z = 10m \tag{35.65}$$

$$\Delta t = 1000secs \tag{35.66}$$

$$B = 10^{19} cm^4/sec. \tag{35.67}$$

Linear stability says that tapering of the biharmonic coefficient must occur when the slope is larger than

$$\delta = 75/100. \tag{35.68}$$

In the ocean, this is a rather steep slope, and so should not place a serious constraint on the regions for which the biharmonic dissipation acts with the full diffusivity. Note that the use of $B = 10^{19} cm^4/sec$ implies a dissipation time scale of grid sized tracer anomalies (see Section 34.4 for details of how this damping time is derived)

$$\tau = (\Delta a/2)^4/B \tag{35.69}$$

equal to 1day. Such a time scale corresponds to the time needed to damp out a wave with wavenumber $k = \pi/\Delta a$, which is the highest wavenumber available to the grid.

Note that for most model grids, the huge maximum slope of 75/100 is indistinguishable from the more modest 1/100 value commonly used with *redi_diffusion* or *gent_mcwilliams*. The reason is that the grid aspect ratio is typically on the order of 1/1000 in the models. Therefore, the model grid essentially equates a slope $\geq 1/100$ to an infinite slope. Hence, it makes little difference whether one uses the same maximum slope for the *biharmonic_rm* option as for the *redi_diffusion* or *gent_mcwilliams* options, or if one allows the maximum slope for the *biharmonic_rm* option to be larger. For coding simplicity and consistency between the different schemes, MOM uses the *same* maximum slope $slmx$ for all of the isoneutral mixing schemes.

35.1.8.7 Choosing the biharmonic coefficient

The choice of the biharmonic diffusivity is based on the need to maintain linear stability in steep sloped regions, and the interest to damp grid scale features within a reasonable time. The previous section mentioned the slope constraints. The smallest "resolvable" wave on a grid has size 2Δ and wavenumber $k = \pi/\Delta$. For such a wave, as mentioned above, the biharmonic scheme will act to damp it within a time scale $\tau = (\Delta/2)^4/B$. Again, this relation is derived in Section 34.4 for the analysis of the traditional biharmonic scheme. Its application to the present scheme is approximate.

35.1.8.8 Discretization details for the RM98 operator

Discretization of the RM98 skew-fluxes directly parallels that of the GM skew-fluxes discussed in Section 35.1.6.1. Just as for the Laplacian GM scheme, the skew-flux approach using the Griffies *et al.* (1998) algorithm guarantees that the biharmonic scheme will not alter the tracer variance. Additionally, the biharmonic tracer fluxes are much less noisy than the fluxes resulting from computing the eddy-velocity (see discussion in Section 35.1.6.1). However, when applied to the biharmonic scheme, the Griffies *et al.* (1998) algorithm brings a relatively heavy computational load. The algorithm described by RM98 is likely cheaper, but it computes eddy-velocities and so has not been implemented at this time.

The main difference between the Laplacian GM scheme and the biharmonic scheme is the extension of the stencil due to the Laplacian operator acting on the slopes. In order to maintain numerical stability in steep sloped regions, each slope comprising the elements of the Laplacian is weighted by its corresponding diffusivity. In the case of constant diffusivity and small slopes, this discretization reduces to that prescribed in the continuum. All flux components are integrated explicitly in time, since there is no diagonal contribution to the vertical tracer flux component.

To get started, recall that the off-diagonal term in the discretized small angle x-component of the Redi diffusive flux is given by

$$-F^{off-diag}_{i,k,j} = \frac{1}{4}\sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\, Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)} \delta_z T_{i+ip,k-1+kr}, \tag{35.70}$$

where the terms in this equation are defined in Section 35.1.5.1. This form for the flux is taken for the discretization of the RM98 operator, with the diffusivity times the slope becoming the

horizontal Laplacian of the biharmonic diffusivity times the slope

$$-F^{x-biharm}_{i,k,j} = \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}$$
$$\sum_{ip=0}^{1}\nabla_h^2\left(Bx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\,Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\right)\delta_z T_{i+ip,k-1+kr}. \tag{35.71}$$

Again, the inclusion of $Bx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ inside the Laplacian is necessary for maintaining numerical stability in regions of steep slopes. Otherwise, for example, $\nabla_h^2 Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ could be large due to the presence of a steep slope, say $Sx^{(i+1+ip,k)}_{(i+1,k|i+1+ip,k-1+kr)}$. With $Bx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ outside the Laplacian, the product $Bx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\nabla_h^2 Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}$ will then be larger than allowed by numerical stability.

The meridional flux is given likewise by

$$-F^{y-biharm}_{i,k,j} = \frac{1}{4\cos\phi_j^T}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}$$
$$\sum_{jq=0}^{1}\nabla_h^2\left(By^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\,Sy^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\right)\delta_z T_{k-1+kr,j+jq}. \tag{35.72}$$

The vertical flux is given by

$$\begin{aligned}
-F^{z-biharm}_{i,k,j} = & -\frac{1}{4dxt_i\,dht_{i,k,j}}\sum_{ip=0}^{1}dxu_{i-1+ip} \\
& \sum_{kr=0}^{1}\Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)} \\
& \times\nabla_h^2\left(Bx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}\right)\delta_x T_{i-1+ip,k+kr} \\
- & \frac{1}{4\cos\phi_j^T dyt_j\,dht_{i,k,j}}\sum_{jq=0}^{1}\cos\phi^U_{j-1+jq}dyu_{j-1+jq}\sum_{kr=0}^{1}\Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)} \\
& \times\nabla_h^2\left(By^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}\right)\delta_y T_{k+kr,j-1+jq}.
\end{aligned} \tag{35.73}$$

In each of the above expressions, the horizontal Laplacian operator is given by

$$\begin{aligned}
\nabla_h^2\alpha &= \frac{1}{dxt_i(\cos\phi_j^T)^2}\left(\frac{\alpha_{i+1}-\alpha_i}{dxu_i}-\frac{\alpha_i-\alpha_{i-1}}{dxu_{i-1}}\right) \\
&+ \frac{1}{dyt_j\cos\phi_j^T}\left(\frac{\cos\phi_j^U(\alpha_{j+1}-\alpha_j)}{dyu_j}-\frac{\cos\phi_{j-1}^U(\alpha_j-\alpha_{j-1})}{dyu_{j-1}}\right).
\end{aligned} \tag{35.74}$$

35.1.9 Isoneutral mixing and steep sloped regions

Regions of extremely steep isoneutral slopes are typically associated with a strong amount of dianeutral mixing. These are the ocean's mixed layers. To fully resolve them requires non-hydrostatic models with nearly isotropic grid aspect ratios (e.g., Marshall *et al.* 1997). MOM

is hydrostatic and so must parameterize the physics of these boundary layer regions through one of its mixed layer schemes. In these regions, the adiabatic stirring and isoneutral diffusion schemes can be eliminated in favor of a mixed layer scheme. This argument prompts an approach which says that when isoneutral slopes steepen, the fluxes from *redi_diffusion*, *gent_mcwilliams*, and *biharmonic_rm* schemes are systematically reduced in favor of the fluxes arising from the mixed layer scheme. This is the central argument motivating one to *not* bother using the full Redi diffusion tensor. This argument is also coincident with the numerical need to turn down the fluxes from the isoneutral schemes as the slopes steepen in order to maintain linear stability (Cox 1987, Griffies *et al.* 1998). The way to turn off these schemes is simple: scale or taper the diffusivities A_I, κ, and B to zero as the magnitude of the slope increases. There are two ways in MOM to taper these coefficients to zero: in a quadratic manner as implemented in option *gkw_taper* (Gerdes, Köberle, and Willebrand 1991) or exponentially as with option *dm_taper*. The option *gkw_taper* is the default approach used in MOM.

When tapering the fluxes from *redi_diffusion*, *gent_mcwilliams*, and *biharmonic_rm* in steep sloped regions, it has traditionally been assumed that the only remaining nonzero fluxes arise from a nonzero vertical diffusivity. Namely, all horizontal fluxes are zeroed out in the steep sloped regions. However, one might suspect that horizontal mixing occurs by the unresolved eddies in the mixed layers, as argued by Treguier, Held, and Larichev (1997). Otherwise, there is no way to dissipate density structures using only vertical diffusion, since the isoneutral slopes are nearly vertical. Consequently, Treguier *et al.* suggest the retention of a nonzero horizontal diffusivity in the mixed layers, where the strength of this diffusivity is the same as the skew-diffusivity used in the interior. The diffusivity *ahsteep* serves this purpose. As the slopes steepen, the diagonal component to the two horizontal tracer fluxes employ a diffusivity which is no smaller than *ahsteep*. For the case of constant diffusivities, setting *ahsteep* to zero recovers the traditional approach with zero horizontal fluxes. When the diffusivities are nonconstant, as described in Section 35.2, the default is to have *ahsteep* equal the nonconstant skew-diffusivity used in the interior.

The use of a nontrivial *ahsteep* in steep sloped regions is restricted to models which employ the default *redi_diffusion* option *small_tensor*. For example, models without option *small_tensor*, but with *biharmonic_rm*, are meant to be eddy-permitting models, and so the eddies should explicitly provide the horizontal stirring parameterized by *ahsteep*. The most simple choice for coarse models is to set $ahisop = athkdf = ahsteep$. In this case, the algorithm becomes particularly simple in that the 13 and 23 components to the mixing tensor are identically eliminated, unless option *biharmonic_rm* is also enabled.

Another approach is to employ a horizontal diffusivity equal to the value of the vertical diffusivity arising from some prognostic mixed layer scheme. In this way, the steep sloped regions maintain a 3D isotropic symmetry, as appropriate for a 3D isotropic turbulent mixed later. Option *isotropic_mixed* implements this parameterization so long as option *redi_diffusion* is enabled.

35.1.9.1 dm_taper

Option *dm_taper* enables the hyperbolic tangent re-scaling for steep slopes by Danabasoglu and McWilliams (1996). The tapering is given by

$$taper = 0.5 \left[1 - \tanh \left(\frac{|S| - del^{dm}}{S^{dm}} \right) \right], \tag{35.75}$$

where S is the neutral direction slope, and del^{dm} and S^{dm} are namelist input parameters. These three tunable parameters determine the details of this tapering scheme. One should exercise some care in choosing these coefficients in order to maintain model stability.

Since the tanh function is so quickly changing, there is no need to provide a slope check if-test with *dm_taper*. Instead, the diffusivities are always defined according to

$$(A_I, \kappa, B)_{effective} = taper * (A_I, \kappa, B)_{un-scaled}. \tag{35.76}$$

With the GFDL Cray T90, a strange code bug is associated with the tanh function. Whenever tanh is called with an argument larger than roughly 19, an overflow results and the model is brought down. The tanh of numbers larger than 19 are quite close to unity, and so a fix for this bug has been implemented so that the argument never gets too large. Those wishing to use option *dm_taper* who do not share such silly problems are certainly encouraged to remove this fix in order to save in computation time. With the current code, the option *dm_taper* is slower than *gkw_taper*.

Note that neither *gkw_taper* nor *dm_taper* apply to option *full_tensor*, which is re-scaled as given in Section C.2.9.

35.1.9.2 gkw_taper

If option *gkw_taper* is enabled, the algebraic tapering of Gerdes *et al.* (1991) is used

$$taper = \left(\frac{slmx}{|S|}\right)^2. \tag{35.77}$$

This is the MOM3 default tapering option. In this case, when the slope is larger in absolute value than $slmx$, then the effective diffusivities take the values

$$(A_I, \kappa, B)_{effective} = taper * (A_I, \kappa, B)_{un-scaled}. \tag{35.78}$$

To determine when to taper, an if-test is required. This taper is much weaker than that used with option *dm_taper*. However, preliminary tests indicate the absence of much difference in the large scale features in coarse models between the different tapers.

Note that neither *gkw_taper* nor *dm_taper* apply to option *full_tensor*, which is re-scaled as given in Section C.2.9.

35.1.9.3 isotropic_mixed

In planetary boundary layers, it might be of interest to compute diagonal horizontal tracer diffusion fluxes with the same diffusivity as the vertical tracer diffusion flux. In this case, the parameterized turbulence is represented by isotropic horizontal/vertical diffusion. The physical picture is of large eddies moving parcels both horizontally and vertically in the mixed layer. If one is using some mixed layer scheme such as option *kppvmix* (Section 33.2.3), then the vertical diffusivities can sometimes reach up $10^3 - 10^5 cm^2/sec$, especially in mixed layers associated with deep convection. In these regions, the use of equal horizontal and vertical diffusivities can provide a nontrivial amount of horizontal fluxes in addition to the vertical fluxes. Option *isotropic_mixed* implements this idea in MOM.

For use in models with isoneutral mixing, the above ideas are implemenated inside the small angle redi diffusion scheme (the default option for *redi_diffusion*). In this case, the continuum horizontal diffusion fluxes take the form

$$-F^x = (A_I + A_D)\, T_x + A_I\, S_x\, T_z \tag{35.79}$$
$$-F^y = (A_I + A_D)\, T_y + A_I\, S_y\, T_z. \tag{35.80}$$

Only in regions where A_D is on the same order as A_I will its presence be significant. For regions outside the boundary layer, A_D is generally 7-8 orders of magnitude smaller than A_I, and so it is completely negligible. Inside the boundary layer in regions with steep isoneutral slopes, the effective A_I is tapered to a small value using either *gkw_taper* or *dm_taper*. The effective A_D, computed from the boundary layer scheme, is also much larger than in the interior, and so it can contribute a relatively larger amount to the horizontal fluxes.

In the code, the vertical diffusivity $diff_cbt$ is the sum $A_D + K33$, as this coefficient is that which is used for the implicit solution of the vertical diffusion equation. Using a mixed continuous/numerical notation, option *isotropic_mixed* uses the algorithm

$$-F^x = [A_I + \left(\overline{diff_cbt - K33}\right)^{z,x}]\, T_x + A_I\, S_x\, T_z \tag{35.81}$$
$$-F^y = [A_I + \left(\overline{diff_cbt - K33}\right)^{z,y}]\, T_y + A_I\, S_y\, T_z, \tag{35.82}$$

where the $\overline{()}^{z,x}$ and $\overline{()}^{z,y}$ represent four point averages used to bring the vertical diffusivities, defined on the bottom of a T-cell, to the relevant zonal or meridional sides.

There are no namelist parameters to set with option *isotropic_mixed*. Turning it on, however, requires the use of option *fourth_order_memory_window*. The reason is that the vertical diffusivity $A_D = diff_cbt$ is computed on the bottom face of the tracer cell. To add it to the diagonal piece of the horizontal tracer fluxes, however, it must be averaged using a four point horizontal and vertical average. Such an average in the $y - z$ plane requires option *fourth_order_memory_window*. Option *fourth_order_memory_window* is automatically enabled when option *isotropic_mixed* is used. Note that option *isotropic_mixed* can be run in parallel to setting *ahsteep* to some nonzero value (see discussion at beginning of Section 35.1.9).

35.2 Schemes with nonconstant diffusivities

The isoneutral diffusion and skew-diffusion eddy coefficients have traditionally been assumed constant, with the exception of tapering discussed in Section 35.1.9 and the zero eddy flux condition at the ocean surface (Gent *et al* 1995). Recent studies, however, indicate the importance of moving away from a constant diffusivity towards the use of flow dependent diffusivities (e.g., Held and Larichev 1996, Visbeck *et al.* 1997, Treguier *et al.* 1997, Killworth 1997, Stammer 1998, Bryan *et al.* 1999).

At present, there are two schemes implemented in MOM for prescribing the isoneutral diffusion and skew-diffusion eddy coefficients according to the large-scale geostrophic flow field: the theories of Held and Larichev (1996) and Visbeck *et al.* (1997). The resulting diffusivities have no more depth dependence than that already implied by the discussion in Section 35.1. As such, either the eddy-induced velocity of Gent *et al.* (1995), or the skew-flux formulation of Griffies (1998), are used with a diffusivity that is a function of (λ, ϕ, t). The more general ideas from Treguier *et al.* (1997) and Killworth (1997), which include depth dependence in the diffusivity and extra terms in the closure associated with the β-effect, have not been implemented.

Both the skew-flux (Section 35.1.6.1) and advective flux (Section 35.1.6.2) formulation of GM have been extended to the nonconstant diffusivities. Again, the recommended approach is the skew-flux approach.

Held and Larichev and Visbeck *et al.* develop expressions for a time scale T and length scale L according to properties of the model's resolved flow field. Thereafter, a diffusivity is given by

$$\kappa = L^2/T. \tag{35.83}$$

The two schemes prescribe the same time scale, but different length scales.

In general, the quasi-geostrophic approximation is fundamental to all the theories used to prescribe the mixing coefficients. Since this approximation breaks down near the equator, care should be exercised when interpreting global model results using the given parameterizations.

For MOM, there are three main cases available for setting the relative values of the isoneutral diffusivity and skew-diffusivity:

1. For the constant diffusivity case discussed in Section 35.1, the namelist parameters *ahisop* and *athkdf* set the isoneutral and skew diffusivities. When *ahisop* = *athkdf*, the code exploits the simplifications in the tracer flux discussed in Section 35.1.6.1 so long as the default option *gm_skew* is employed.

2. For the nonconstant diffusivity case, setting *ahisop* = *athkdf* will also provide for equal nonconstant diffusivities, and the code will exploit the simplifications in the tracer flux discussed in Section 35.1.6.1 so long as the default option *gm_skew* is employed.

3. If either of the nonconstant diffusivity options is enabled, and if the namelist parameters *ahisop* $\neq$ *athkdf*, then the isoneutral diffusive flux will be computed with the constant diffusivity *ahisop*, whereas the skew-flux or eddy-induced velocity will be computed with a nonconstant diffusivity.

35.2.1 hl_diffusivity

Held and Larichev (1996) specified a time and length scale given by

$$T^{-2} = \frac{f^2}{D}\int_{-D_b}^{-D_t} Ri^{-1}\,dz \tag{35.84}$$

$$L_{hl}^{-1} = \beta_{eff}\,T, \tag{35.85}$$

where $D = D_b - D_t$. The resulting diffusivity is given by

$$\begin{aligned}\kappa &= L_{hl}^2\,T^{-1}\\ &= (\beta_{eff}^2\,T^3)^{-1}.\end{aligned} \tag{35.86}$$

The time scale is related to the growth rate of a baroclinic wave (an Eady growth rate; on the order of days), and the length scale is related to the Rhines' scale.

35.2.1.1 The thermal wind Richardson number and the depth range

Fundamental to the mesoscale eddy closure theories (e.g., Held and Larichev, Treguier *et al.*, Visbeck *et al.*, Killworth) is the assumption that the mesoscale eddy field is quasi-geostrophic.

As such, the Richardson number, Ri, is a large-scale Richardson number based on vertical shears under thermal-wind balance with the buoyancy field. This assumption renders

$$\begin{aligned} Ri &= \frac{N^2}{U_z^2 + V_z^2} \\ &= \frac{-g\,\rho_z/\rho_o}{(g/f\rho_o)^2(\rho_x^2+\rho_y^2)} \\ &= -\left(\frac{f^2\,\rho_o}{g\,\rho_z}\right) S^{-2} \\ &= (f/NS)^2 \end{aligned} \tag{35.87}$$

where $S^2 = (\rho_x^2 + \rho_y^2)/\rho_z^2$ is the squared isoneutral slope vector and $N^2 = -g\,\rho_z/\rho_o$ is the squared buoyancy frequency based on the vertical gradient of locally referenced potential density.

The integration depth range, $D = D_b - D_t$, corresponds to the depth over which the baroclinic eddies predominanty occur. Treguier *et. al.* (1996) use the values $D_t = 100m$ and $D_b = 2000m$. This depth range is also taken in the Hadley Centre ocean model in which they implement the Visbeck *et al.* scheme (Section 35.2.2), and it is also the default for MOM. This depth range is not fundamental, and sensitivity of the results to the details of this range has not been documented. In order to avoid problems with unstratified lowest model levels, as might occur with bottom boundary layers, the bottom level of the depth range is set to at least two depth levels above the ocean bottom. In this way, the computed Eady growth rate is taken over interior model levels. Pragmatically, this restriction also avoids the distinction between partial and full bottom cells (Chapter 26). In regions where the ocean is shallower than D_t, the diffusivities default to the background values $A_I = ahisop$ and $\kappa = ahthk$ used in the constant diffusivity case, which are set in the namelist.

Using the thermal wind Richardson number (35.87) brings the squared inverse time scale to the form

$$\begin{aligned} T^{-2} &= \frac{1}{D}\int_{-D_b}^{-D_t} N^2\,S^2\,dz \\ &= \frac{g}{\rho_o\,D}\int_{-D_b}^{-D_t} |\rho_z|\,S^2\,dz. \end{aligned} \tag{35.88}$$

Note how the explicit dependence on the Coriolis parameter f has cancelled. Again, the source of this cancellation is the use of thermal wind balance for computing the Richardson number. Consequently, the time scale is an explicit function only of the vertically averaged horizontal and vertical stratification. Notably, the inverse time scale, or the growth rate, vanishes when the vertically integrated horizontal stratification vanishes; i.e., when there is zero baroclinicity. As such, the diffusivity vanishes when the neutral directions are flat, as one would expect from theories of baroclinic instability. Relatedly, the explicit cancelation of the Coriolis parameter allows for the time scale calculation to be naively applied globally, including at the equator, where geostrophy is irrelevant.

35.2.1.2 The effective β parameter

The parameter β_{eff} incorporates both planetary vorticity gradients and gradients in the large-scale topography

$$\beta_{eff} = H|\nabla_h(f/H)|$$

$$= \left((\beta - f\, H_y/H)^2 + (f\, H_x/H)^2 \right)^{1/2} \tag{35.89}$$

where $\beta = \partial_y f$ is the planetary vorticity gradient, $H = H(\lambda, \phi) \geq 0$ is the total depth of the ocean, and

$$H_x = \frac{H_\lambda}{a \cos\phi} \tag{35.90}$$

$$H_y = \frac{H_\phi}{a}. \tag{35.91}$$

35.2.1.3 Smoothing and temporal frequency of computation

As the Held and Larichev mixing coefficients are derived under the assumptions of quasi-geostrophy, it is sensible to impose on the diagnosed diffusivity some smoothness whose scales reflect the large-scale geostrophic flow. To ensure this smoothness, the following filtering is performed. The researcher can impose more or less filtering as desired.

For evaluating β_{eff}, the model's stepped bottom topography is first smoothed with a 2-dimensional finite impulse response filter

$$\tilde{H}_{i,j} = \sum_{ip=-1}^{1} \sum_{jq=-1}^{1} H_{i+ip,j+jq}\, M_{ip,jq} \tag{35.92}$$

where the smoothing matrix has components

$$M_{ip,jq} = \begin{pmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{pmatrix}. \tag{35.93}$$

Multiple passes through this filter will further smooth the topography. The researcher can choose the number of passes through changing the parameter *numfltrtopog* inside of the fortran routine *topog.F*. The default is one pass. The spatial derivatives of the topography are then computed using the filtered topography field.

Implicit in the formalism is some smoothing in both time and space due to the use of the thermal wind relation when computing the time scale in equation (35.88). A further smoothing can be performed on this time scale through the use of the two dimensional finite impulse response filter. The parameter *numfltrgrth*, set inside of *nonconstdiff.F*, determines the number of passes through the FIR for the growth rate. The default is $numfltrgrth = 0$ for zero filtering. To avoid problems with overly huge growth rates computed in regions of very low vertical stratification, a minimum time scale for the growth is taken to be $1/4day$, which leads to a maximum squared growth rate of $2.14 \times 10^{-9} sec^{-1}$. This limit can be changed through altering the parameter *growth2max* inside of *nonconstdiff.F*. Temporal smoothing of the growth rate in the form of a Robert filter has been suggested by Visbeck *et al.* (1997). This smoothing has not been implemented in MOM.

The time scale T is computed in MOM based on density fields one time step previous to the present time step. T is accumulated as a vertical average within the *isopyc.F* routine using the same code that computes the isoneutral slopes (inside subroutines *ai_east*, *ai_north*, and *ai_bottom*), where the slopes are computed as described in Griffies *et al.* (1998), Section 35.1.5, and Appendix C. Conveniently, no extra slope computations are needed beyond that

already required by the constant diffusivity isoneutral schemes. A minimum time scale of $T = 1day$ is imposed on the computation; this value is set by the parameter *growthmax* inside of $nonconstdiff.F$.

When finished computing the vertically averaged Richardson number, a two-dimensional time scale field $T(\lambda, \phi)$ is retained and then used to define the Rhines' length and diffusivity. The time scale field is saved inside of a restart file as well as the nonconstant diffusivity, thus allowing for a smooth evolution of the nonconstant diffusivity across model restarts.

To facilitate those cases in which one wishes to turn on a nonconstant diffusivity after running for some time with constant diffusivities, the option $nonconst_diffusivity_initial$ will initialize $T(\lambda, \phi)$ to zero and it will override the attempt to read in $T(\lambda, \phi)$ from the restart file. Conversely, for those wishing to change from nonconstant to constant diffusivity in the middle of an ongoing experiment, turning on $nonconst_diffusivity_final$ will mean that $T(\lambda, \phi)$ will not be written to the restart file.

The namelist parameter $diffint$ sets the temporal frequency used to update the diffusivities. The idea is that the diffusivities should change only over time scales determined by the eddy time scale T, which is a few days. Indeed, for the extreme example of a static model exhibiting no internal or forced varibility, the diffusivities are unchanging. For more dynamic models, for example with a seasonal cycle and/or realistic atmospheric forcing, more frequent diffusivity calculations are prudent. The parameter $diffint$ determines the number of time steps skipped before computing a new value for the diffusivity. Hence, the frequency of computing the diffusivity is dependent on the model time step. The current implementation was found useful in order to ensure results agree across restart files. The MOM default is $diffint = 5$, but this should be adjusted according to the time steps used for the particular experiment.

35.2.1.4 Summary of namelist parameters

In summary, the namelist parameters for the Held and Larichev scheme, which are set in the namelist $/ncdiff/$, are the following:

1. $D_b = depthbot$ is the maximum depth over which the vertically averaged Richardson number is computed. The default is $depthbot = 1000m$. In regions shallower than $depthbot$, the lower limit on the vertical average defaults to the local model depth.

2. $D_t = depthtop$ is the minimum depth over which the vertically averaged Richardson number is computed. The default is $depthtop = 100m$. In regions where the depth is shallower than $depthtop$, the scheme returns a diffusivity equal to the value $athkdf$ input to the namelist.

3. $diffmin$ is the minimum diffusivity returned from the scheme. The default is $diffmin = 10^6 cm^2/sec$.

4. $diffmax$ is the maximum diffusivity returned from the scheme. The default is $diffmax = 2 \times 10^7 cm^2/sec$. Note that the effective value of the maximum diffusivity employed is constrained by the steep slope conditions discussed in Section 35.1.9.

5. $diffint$ sets the frequency, in model time steps, on which the nonconstant diffusivity will be updated. The default is $diffint = 5$.

35.2.2 vmhs_diffusivity

The MOM implementation of the Visbeck, Marshall, Haine, and Spall (1997) (denoted VMHS for the duration of this section) scheme follows closely to the Hadley Centre's implementation (Wright 1997, Malcolm Roberts personal communication, 1998). Many thanks go to Malcolm Roberts for communicating the details of their implementation, especially for sharing the length scale algorithm described below.

35.2.2.1 Time scale same as Held and Larichev

For the VMHS scheme, the time scale T is written

$$T_{vmhs}^{-1} = \frac{1}{D}\int_{-D_b}^{-D_t} (f^2/Ri)^{1/2}\, dz. \tag{35.94}$$

The square of this expression is not identical to equation (35.84) from the Held and Larichev scheme

$$T_{hl}^{-2} = \frac{1}{D}\int_{-D_b}^{-D_t} (f^2/Ri)\, dz. \tag{35.95}$$

However, in both approaches the time scale is determined by scaling arguments rather than from a fundamental theory. Therefore, consistency and simplicity motivate using an identical expression in MOM. Note that the expression from Held and Larichev was also used by Wright (1997) in the VMHS scheme implemented in the Hadley Centre ocean model. Hence, due partly to historical reasons (the Held and Larichev scheme was implemented first), and the desire to be consistent with the Hadley Centre implementation, the Held and Larichev expression for the time scale is implemented in MOM for both the *hl_diffusivity* and *vmhs_diffusivity* schemes.

As mentioned in Section 35.2.1.1, the thermal wind relation is used to compute the above Richardson number (equation (35.87)). The reason is that the theories used to define the diffusivities are based on quasi-geostrophic scaling. For computational reasons, the Hadley Centre uses the vertical shears of the full velocity field, rather than the thermal wind shears, in their Richardson number computation. This difference in Richardson number calculation represents the central difference between the MOM implementation of VMHS and that of the Hadley Centre. It is unclear how much difference this approach will make for the resulting time scale.

35.2.2.2 Length scale based on baroclinic zone width

The length scale L in the scheme of VMHS is determined by the regional structure of the baroclinicity. Figure 35.1 illustrates the basics of the algorithm used to determine L (algorithm based on that used at the Hadley Centre).

- VMHS used a minimum L as the maximum of either the grid spacing Δs or the first baroclinic Rossby radius (NH/f), where $\Delta s = max(\Delta x, \Delta y)$. For most global models employing this scheme, Δs will be larger than the first baroclinic Rossby radius. Only for those cases in which the mesoscale eddies are partially resolved will there be overlap. The implementation in MOM simply sets the minimum length scale to $L_{min} = \Delta s = max(\Delta x, \Delta y)$.

- Generally, L will be larger than its minimum value. To determine its value, the spatial variability of the time scale T is used to determine L. L is proportional to the distance it takes to have the growth rate T^{-1} become smaller than some rate *vmhs_rate_limit*. The MOM default value is $vmhs_rate_limit = 1.4 \times 10^{-6} sec^{-1}$. This value yields reasonable diffusivities using the Levitus (1982) dataset (Malcolm Roberts, personal communication). In words, the algorithm proceeds as follows (see Figure 35.1):

  1. If the growth rate at a grid point (x_i, y_j) is less than *vmhs_rate_limit*, then the length scale at this point is set to $L_{i,j} = max(\Delta x_i, \Delta y_j)$. Point P1 in Figure 35.1 is an example of such a point.
  2. If the growth rate at a grid point (i, j) is greater than *vmhs_rate_limit*, then the "width of the baroclinic zone" is determined. Using point P2 in Figure 35.1 as an example, the distances

$$distns = distn + dists \tag{35.96}$$
$$distew = diste + distw \tag{35.97}$$
$$zone = \min(distns, distew) \tag{35.98}$$

     are found using a search algorithm. The distance *zone* defines the "width of the baroclinic zone" so far as point point P2 is concerned. Additionally, the four "dist" distances $(distn, dists, diste, distw)$ have a maximum grid spacing of *ijvmhs_maxlength*. $ijvmhs_maxlength = 10$ is the default used in MOM (based on the same value used at the Hadley Centre).
  3. For point P2 in Figure 35.1, $zone = distns$. In this case, define the fraction

$$frac = \frac{\min(distn, dists)}{\max(distn, dists)}. \tag{35.99}$$

     For point P2, $frac = distn/dists$. The length scale for point P2 is then defined by

$$\begin{aligned} L_{P2} &= frac * zone \\ &= distn \left(1 + \frac{distn}{dists}\right). \end{aligned} \tag{35.100}$$

     Note that the length scale L_{P2} is always larger than the grid scale Δy. If $frac$ were instead defined by $frac = \min(distn, dists)/distns$, then L_{P2} would simply equal $\min(distn, dists)$, which has a minimum value equal to the grid scale. Instead, the chosen algorithm gives a bit more weight to any of the points within a baroclinic zone, even if they are just a single point within the zone.

35.2.2.3 Diffusivity and the basic tunable parameter

Once the length and time scales are computed, the diffusivity is given by

$$\kappa = \alpha L_{vmhs}^2 T^{-1} \tag{35.101}$$

where α is a tuning parameter. $\alpha = 0.015$ was found by VMHS to be optimal for fitting their model experiments. This value is also used in the Hadley Centre ocean model. It is the default value in MOM, and can be changed by setting the namelist parameter *vmhs_alpha*.

35.2.2.4 Smoothing and temporal frequency of computation

The issues of smoothness and frequency of computation for the diffusivities were discussed in Section 35.2.1.3 for the Held and Larichev scheme. That discussion is relevant for the VMHS scheme as well.

35.2.2.5 Summary of namelist parameters

In summary, the namelist parameters for the VMHS scheme, which are set in the namelist $/ncdiff/$, are the following:

1. $D_b = depthbot$ is the maximum depth over which the vertically averaged Richardson number is computed. The default is $depthbot = 1000m$. In regions shallower than $depthbot$, the lower limit on the vertical average defaults to the local model depth.

2. $D_t = depthtop$ is the minimum depth over which the vertically averaged Richardson number is computed. The default is $depthtop = 100m$. In regions where the depth is shallower than $depthtop$, the scheme returns a diffusivity equal to the value $athkdf$ input to the namelist.

3. $diffmin$ is the minimum diffusivity returned from the scheme. The default is $diffmin = 10^6 cm^2/sec$.

4. $diffmax$ is the maximum diffisivity returned from the scheme. The default is $diffmax = 2 \times 10^7 cm^2/sec$. Note that the effective value of the maximum diffusivity employed is constrained by the steep slope conditions discussed in Section 35.1.9.

5. $diffint$ sets the frequency, in model time steps, on which the nonconstant diffusivity will be updated. The default is $diffint = 5$.

6. *vmhs_alpha* is a tuning parameter which scales the diffusivity. The default is $vmhs_alpha = 0.015$.

7. *vmhs_rate_limit* defines what is meant by a large growth rate when defining the length scale. The default is $vmhs_rate_limit = 1.4 \times 10^{-6} sec^{-1}$

8. *ijvmhs_maxlength* is the maximum value of the search which is conducted in the north, south, east, and west directions for determining the width of the baroclinic zone. The default is 10 latitude and longitude grid boxes.

The last three namelist parameters are the only ones not also used by the Held and Larichev scheme (Section 35.2.1).

35.2.3 Held and Larichev combined with Visbeck *et al.*

Both the Held and Larichev and Visbeck *et al.* schemes employ the same time scale. They differ only in their choice of length scale. Held and Larichev (1996) use the Rhines' length, whereas Visbeck *et al.* use a length determined by the width of the baroclinic zone. Treguier *et al.* suggest that the Larichev and Held (1995) and Held and Larichev (1996) studies indicate that an appropriate choice is the *smaller* of the two length scales. In MOM, if both options *hl_diffusivity* and *vmhs_diffusivity* are employed, then the smaller of the two length scales will be used.

35.2.4 Netcdf information for nonconstant diffusivities

The nonconstant diffusivity schemes provide an added amount of structure to the tracer diffusivities. This structre is of interest when studying the details of the schemes. For this purpose, a number of two dimensional fields are mapped to the netcdf file *ncdiff.dta.nc* whenever the diagnostic option *ncdiff_netcdf* is enabled. The interval of output is determined by the namelist parameter $diffsnapint$, which is set along with the other nonconstant diffusivity namelist parameters. The following fields are plotted: T^{-2}, β_{eff}, L_{hl}, κ_{hl}, L_{vmhs}, κ_{vmhs}, and A_{gm}. $A_{gm} = min(\kappa_{hl}, \kappa_{vmhs})$ is the nonconstant GM diffusivity. Note that $A_{redi} = A_{gm}$ if the namelist parameters $ahisop = athkdf$, whereas $A_{redi} = ahisop$ if $ahisop \neq athkdf$. Again, the above fields depend only on latitude and longitude.

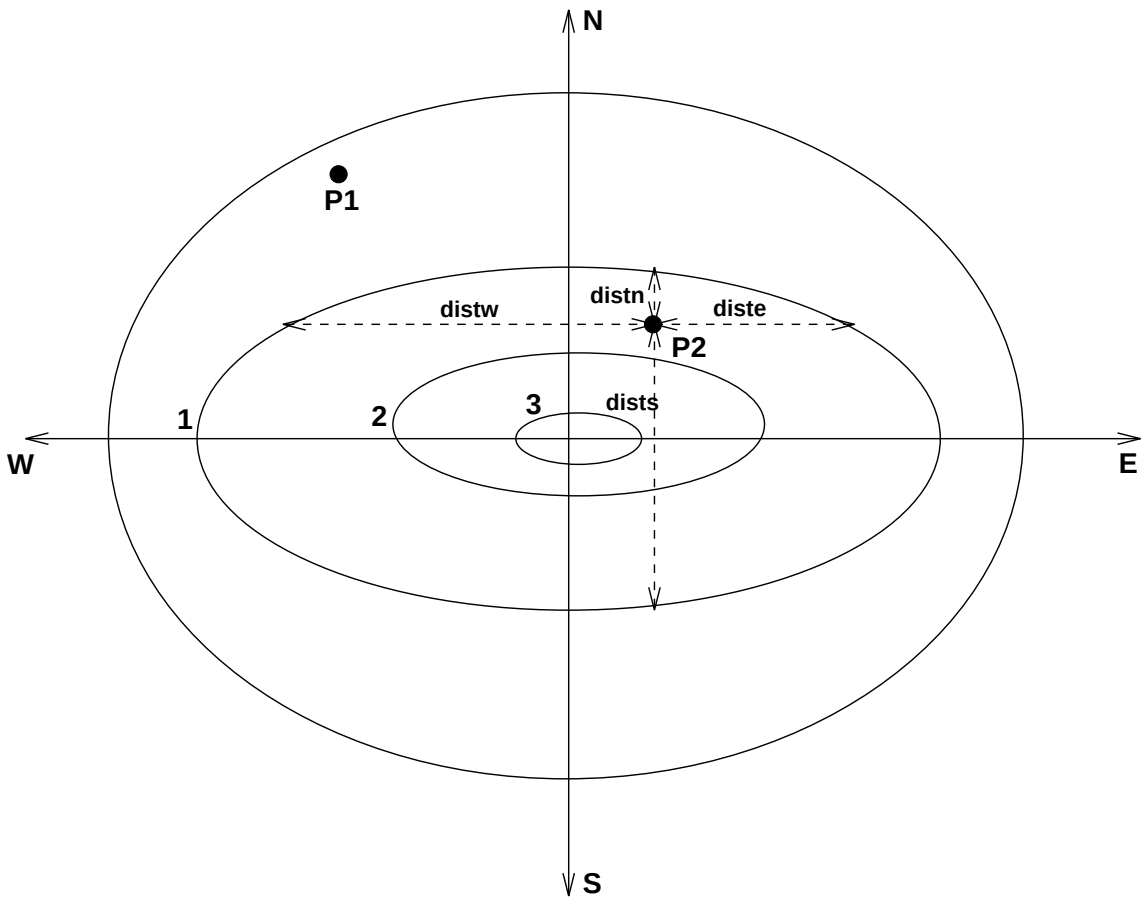


Figure 35.1: *An illustration of the ideas used to generate the algorithm for determining the width of the baroclinic zone. Shown are contours of the growth rate* T^{-1} *in units of* *vmhs_rate_limit*. *Points outside the unit contour are outside the "baroclinic zone," and those inside the contour are inside the zone. For example, point P1 is outside the zone, and so the length scale at P1 takes the value* $L_{P1} = max(\Delta x, \Delta y)$. *Point P2 is inside the baroclinic zone. In order to determine its length scale* L_{P2}, *the distances* *distn*, *dists*, *diste*, *and* *distw* *are determined as the distance from P2 to the unit contour in the north, south, east, and west directions, respectively. These distances are then employed using the algorithm described in the text to find* L_{P2}.

Chapter 36

Miscellaneous SGS options

This chapter is summarizes some miscellaneous SGS options.

36.1 Eddy-topography interactions and neptune

Option *neptune* implements the following. Based on statistical mechanics arguments, Holloway (1992) proposed that interaction between mesoscale eddies and topography results in a stress on the ocean with two important consequences: first, the ocean is not driven towards a state of rest and secondly, the resulting motion may have scales much larger than the scale of the eddies. Somewhat suprisingly, this interaction[1] can generate coherent mean flows on the scale of the topography. The magnitude of this topographic stress is dependent on the correlation between pressure p and topographic gradients ∇H which is largely unknown but even if the correlation is 0.1, the resulting topographic stress would be comparable in magnitude to that of the surface wind.

If the view is taken that equations of motion are solved for moments of probable flow (because of imperfect resolution) then those moments are forced in part by derivatives of the distribution entropy with respect to the realized moments. The entropy gradient is estimated as begin proportional to a departure of the realized moments from a state in which the entropy gradient is weak. This latter state is approximated by a transport streamfunction $\psi^\star$ and maximum entropy velocity $u^\star$ given by

$$\psi^\star = -fL'H \tag{36.1}$$

$$u^\star = \hat{z} \times \nabla\psi^\star \tag{36.2}$$

where f is the Coriolis term, H is depth, and L' is O (10 km). If model resolution is coarse relative to the first deformation radius, $u^\star$ is independent of depth. Instead of eddy viscosity driving flow towards rest, flow is driven towards $u^\star$ using an eddy viscosity of the form $A\nabla^2(u^\star - u)$. Note that topographic influence on flow[2] is not strongest near bottom topography. Instead, the flow implied by $\psi^\star$ only approximates a maximum entropy system given eddies and topography. Since this approximation is admittedly crude, further refinements are open to researchers.

[1] Which is missing or at best poorly represented in numerical models at any resolution.

[2] This is referred to as the Neptune effect because when Greg Holloway described coastal currents that persistantly flow againgst both wind forcing and pressure gradient, the response was that it must be due to King Neptune. Who else?

There is legitimate concern about the stepwise resolution of bottom topography in level models such as MOM and its predecessors. Option *neptune* is an attempt to instruct the model about physical consequences due to topography and eddies which are nearly unachievable even at the most ambitious resolutions. The hope is that if the model can be suitably informed about the effect of topography, it matters little if that topography is only "approximately" represented.

36.2 xlandmix

This section was contributed by Keith Dixon (*kd@gfdl.gov*), with modifications made for the free surface height by Eli Tziperman (*ett@gfdl.gov*) and Stephen Griffies (*smg@gfdl.gov*).

It is sometimes useful to allow water masses which are separated by land to exchange tracer properties. A case in point is when model grid resolution is too coarse to resolve narrow passageways which in reality connect water masses. For example, insufficient resolution may close off the Mediterranean from the Atlantic at Gibraltar, or the Arctic from the Pacific in the Bering Strait region.

Option *xlandmix* allows researchers to establish communication between bodies of water separated by land. The communication consists of mixing tracers between non-adjacent water columns. Momentum is not mixed. If the explicit free surface is enabled, option *xlandmix_eta* allows for the mixing of free surface height.

The scheme conserves tracers and volume. Its influence on tracers is tracked in the MOM diagnostics as part of the tracer source terms.

36.2.1 Formulation

Let *nxland* be the number of water column pairs to be mixed. For each pair "nxl", the "i,jrow" coordinates are given by "$ixland(nxl,1)$, $jxland(nxl,1)$" and "$ixland(nxl,2)$, $jxland(nxl,2)$". Between each pair, mixing takes place from level "*kxland(nxl,1)*" to level "*kxland(nxl,2)*". When one of these "i,jrow" coordinates is encountered in the memory window, the source term for each tracer is modified as follows:

For all k-levels *k = kxland(nxl,1)* through *kxland(nxl,2)*

$$source_{i,k,j} = source_{i,k,j} + bxland_{nxl} \cdot (t_{i_{there},k,j_{there},n,\tau-1} - t_{i,k,j,n,\tau-1}) \tag{36.3}$$

$$bxland_{nxl} = \frac{vxland(nxl)}{\sum_{k=kxland(nxl,1)}^{kxland(nxl,2)} \cos\phi_{jrow}^{T}\, dxt_i\, dyt_{jrow}\, dzt_k} \tag{36.4}$$

where i_{there}, j_{there} represents the "i,j" coordinates of the other point in the nxl^{th} mixed pair within the memory window.

36.2.2 Considerations

- After identifying the "i,jrow" locations of the water column pairs to be mixed, an appropriate volume rate of mixing and the depth range for mixing must be specified. Refer to physical oceanography resources for guidance.

- Specify the following information by editing the USER INPUT section of subroutine *xland1st*: the number of crossland mixing point pairs (parameter *nxland*), their "i,jrow"

locations (*ixland* and *jxland*), the range of model k-levels to be mixed (*kxland*), and the time invariant volume rate of mixing (*vxland*) in units of cm^3/sec.

- An upper limit is placed on the value of *vxland(nxl)*. Within a leap frog timestep (including any tracer timestep acceleration effects), no more than one half of a water column may be transported into its paired column. During initialization, a consistancy check is made to insure that the mixing rate specified by*vxland(nxl)* is not too large. If the researcher implements*vxland(nxl)* as a function of time, then this check must be made each time step.

- The size of the model's memory window (parameter *jmw*) may need to be adjusted, depending upon the j-row separation of pairs of crossland mixing points. The following condition must be met:

$$|(jxland(nxl,2) - jxland(nxl,1)| \leq (jmw-1)/2 \quad for\ nxl = 1,\ nxland \tag{36.5}$$

36.2.3 xlandmix_eta

Consider a model in which the Mediterranean Sea is artificially land-locked due to the use of coarse resolution. When using MOM's explicit free surface with explicit fresh water fluxes, the net evaporation over the Mediterranean Sea will cause the free surface height to decrease without bound. In a model resolving the Straits of Gilbraltor, there will be a transfer of volume across the Strait from the Atlantic. This volume transfer will create a change in the height of the free surface, and the transfer time will be determined by the speed of external gravity waves. The purpose of this section is to describe a means of parameterizing this effect in MOM when employing the explicit free surface and explicit fresh water fluxes.

As discussed in Section 7.3.3, MOM's free surface tendency is given by

$$\eta_t^0 \quad = \quad w(z=0) + q_w, \tag{36.6}$$

where η^0 is the linearized free surface height, $w(z=0) = -\nabla_h \cdot \mathbf{U}_0$ is the vertical velocity at the ocean surface, as determined by the convergence of the vertically integrated flow, and q_w is the fresh water input through evaporation, precipitation, and river runoff. This equation results from neglecting the advection of free surface height by the surface currents.

In the model, if η lives at a grid point identified as one of a cross-land mixing pair, and option *xlandmix_eta* is enabled, the time tendency for these points takes the modified form

$$\eta_t^{here} \quad = \quad w(0) + q_w + \mathcal{S}^{here}/A^{here}, \tag{36.7}$$

where $\mathcal{S}^{here}$ is the volume mixing term

$$\mathcal{S}^{here} = \left(\frac{\eta^{there} - \eta^{here}}{\tau}\right)\left(\frac{A^{here} + A^{there}}{2}\right). \tag{36.8}$$

The mixing term $\mathcal{S}^{here}$ nudges the free surface height η^{here} towards η^{there}. Note that the straightforward volume difference form $\mathcal{S}^{here} = (A^{there}\eta^{there} - A^{here}\eta^{here})/\tau$ will lead to

a nonzero source when $\eta^{there} = \eta^{here}$ but if $A^{there} \neq A^{here}$. Such mixing is not desirable, hence motivating the chosen form.

Originally, the time scale τ for the mixing was determined by an estimate of the time it takes an external gravity wave to cross between the two points:

$$\tau = L\,(g\,H)^{-1/2}, \tag{36.9}$$

where H is set to the averaged depth of the two cross-land points, and L is the horizontal distance between them. However, this time scale gave very noisy results. Consequently, the model currently has the value $\tau = 3$days hardwired. Changes to this time scale may be necessary depending on grid resolution.

It is useful to discuss what cross-land η mixing implies about tracers. For this purpose, consider the case in which one starts with two basins connected by a strait, where the basins have differing surface heights, yet let the temperature and salinity be the same uniform values. Also, remove all surface heat and water fluxes. In a model with strait resolved by at least one velocity point (which, for a B-grid, means at least two tracer points), there will be an adjustment process in which the free surface height equilibrates to the same value across the two basins. In contrast, the tracer concentrations, as there are no surface fluxes, remain unchanged throughout the adjustment. In a model with an unresolved strait, one should achieve the same equilibrated solution. Cross-land mixing of η clearly yields a uniform free surface height at equilibrium. In order to preclude affecting the tracer concentrations through the η mixing, it is necessary to *not* include the cross-land mixing term $\mathcal{S}^{here}$ in the tracer budget. Similar considerations apply to the baroclinic velocity tendency. Note that this result does not depend on having made the linearized free surface approximation. Rather, it is simply a result of tracer conservation. The result is convenient, since it means that no model changes are required for the tracer or baroclinic equations when adding cross-land η mixing.

Chapter 37

Bottom Boundary Layer

Dense water flowing down a slope is at best only poorly represented in z-level coordinate models. However, this situation is remedied within MOM by enabling option *bbl* which parameterizes the bottom boundary layer (hereafter referred to as BBL) by assuming frictional dynamics within a thin layer that effectively "hangs" underneath bottom-most ocean cells. Although the surface under which the BBL cells are "hung" is disconnected due to changes in the number of vertical levels given by $kmt_{i,jrow}$, the BBL cells are mapped onto a continuous two dimensional surface so that each BBL cell is in communication with its four nearest neighbors. For practical reasons, this two dimensional surface is defined to be at constant depth within level $k = km$. For the interior model, level $k = km$ is treated as if it were always land. Within the BBL model (level $k = km$), there are no land cells ... only a two dimensional continuously connected lattice of BBL cells. The BBL cells communicate vertically through advection and diffusion with the bottom-most interior ocean cells above them. Note that land cells may exist between the bottom-most ocean cell and the BBL cell beneath it.

This configuration works well for steeply sloping topography but is inadequate in regions where the topographic slope is less than the grid aspect ratio. Refer to Chapter 26. Enabling option *partial_cell* resolves these mildly sloping regions and the BBL cells are then "hung" underneath partial-cells which allows the BBL to feel shallow as well as steep topographic slopes. As without option *partial_cell*, the BBL cells populate level $k = km$.

Just as with option *partial_cell*, the *bbl* option must be used right from the beginning of an experiment. It is not intended as an option which can be enabled in the middle of an experiment because it will wipe out all information in level $k = km$. For further information, refer to http://www.gfdl.gov/ a1g/bbl.html.

Chapter 38

Miscellaneous options

This chapter contains various options which have not been placed into other catagories.

38.1 max_window

The memory window supports second order, fourth order, and sixth order numerics. For each order numerics, the memory window is configured to contain the minimum number of latitude rows to solve the equations. For second order numerics, three rows are needed, five rows for fourth order, and seven rows for sixth order. The extra rows are used as buffers to hold data needed to solve equations on the central row within the window. If option *max_window* is enabled, the memory window is opened up all the way to hold all rows. For more detail, refer to Section 11.3.4.

38.2 knudsen

Option *knudsen* computes density coefficients according to the Knudsen formulation. If this option is not enabled, then density coefficients are computed according to the UNESCO formulation. Refer to Section 15.1.2 for details.

38.3 pressure_gradient_average

Option *pressure_gradient_average* implements the pressure gradient averaging technique of Brown and Campana (1978) which can allow the time step to be increased by up to a factor of two in certain circumstances! This applies when the time step is limited by internal gravity waves. The actual time step, which should always be determined empirically, will typically be somewhat less than the theoretical factor of two limit. It should be noted that this is not a damping scheme. The amplification factor $|\lambda|$ in the stability analysis given by Brown and Campana (1978) is unity within the region of stable solutions.

Basically, the way it works is that instead of using ρ^τ in the hydrostatic pressure gradient, a semi-implicit density given by

$$\tilde{\rho} = alpha'(\rho^{tau+1} + \rho^{tau-1}) + (1 - 2\alpha')\rho^\tau \tag{38.1}$$

is used with $alpha' = 1/4$. Brown and Campana (1978) also discuss three computational modes which are introduced by this technique. They are handled by either reducing $alpha'$ slightly

or applying additional time averaging to other prognostic variables. Both methods sharply reduce the maximum allowable time step. In MOM 2, the Euler backward mixing time step damps the computational modes. For a discussion on when a semi-implicit pressure gradient is applicable, refer to Killworth, Smith, and Gill (1984). Their analysis indicates that a semi-implicit pressure gradient is applicable for coarse and medium resolution ($\geq$ 1 deg) studies but may not be applicable for high resolution ($\leq$ 30 km) ones.

In order to apply this scheme for one or more rows within the memory window, there must be one additional row of tracers calculated before the internal modes of velocity can be calculated. This is because the pressure gradient, which is defined on U cell latitude rows, requires an average of four surrounding densities which are defined on adjacent T cell latitude rows. Although, strictly not a fourth order option, this extra computed tracer row requires that option *fourth_order_window* also be enabled. This is automatically done when option *pressure_gradient_average* is enabled.

For the minimum fourth order window configuration with jmw=5, tracers are computed for rows 3 and 4 while velocities are only computed on row 3. To accommodate this, starting and ending rows for tracer calculations are given by

$$jstrac = jbuf + 2 - 1/mw \tag{38.2}$$

$$jetrac = min(jmw - jbuf + 1, jmt - 1 - joff) \tag{38.3}$$

where the number of buffer rows is typically $jbuf = 2$, the memory window counter is mw, and the function "min" limits $jetrac$ to memory window rows corresponding to latitude rows less than jmt. Refer to the formal treatment of dataflow in the memory window given in Section 11.3.2 where the starting and ending rows are given. The situation is further complicated when this option is used in conjunction with a fourth order option such as option *biharmonic*. The minimum configuration for the memory window is then jmw=5. This time, tracers are again computed for rows 2 and 3, while velocities are computed for row 2 but an extra fifth row is needed for the biharmonic computation for tracers on row 3. Yes, it works!

38.4 fourth_order_memory_window

The memory window typically has a minimum size of three latitude rows (jmw=3) which is appropriate for second order accurate numerics. Some options use fourth order numerics or, in some cases, averaging operators which require information from two cells away. All fourth order schemes require option *fourth_order_memory_window* which is automatically enabled in file *size.h* when any of the existing fourth order schemes are enabled. These schemes require the minimum size of the memory window to be four latitude rows (jwm=4). Some combination of schemes require the minimum memory window to be of size $jmw = 5$. However, more is required than simply opening up the window to jmw=4 rows. In the minimum size fourth order window, prognostic equations are solved only on row 2. In a second order window with $jmw = 4$, they are solved for rows 2 and 3. For a further description of how the window works with higher order schemes, refer to Section 11.3.2.

Calculations always proceed up to latitude row $jrow = jmt - 1$ even with higher order schemes!. There are no out of bounds references because meridional indexing is limited to a maximum at latitude $jrow = jmt$ and a maximum corresponding memory window row given by $j = min(j + joff, jmt) - joff$. To accommodate higher order schemes when a fully open

memory window $jmw = jmt$ is used, meridional fluxes are set to zero at latitude $jrow = jmt$ which allows calculations to proceed through latitude row $jrow = jmt - 1$.

38.5 implicitvmix

Option *implicitvmix* allows the vertical diffusion of momentum and tracers to be solved implicitly under control of an implicit vertical diffusion factor *aidif* which is input through namelist. Refer to Section 14.4 for information on namelist variables. Setting *aidif = 1.0* gives full implicit treatment and setting *aidif = 0* gives full explicit treatment.

Pages 42 and 43 of Numerical Recipes (1992) recommend a setting of *aidif = 0.5* which is the Crank-Nicholson scheme followed by a few fully implicit steps using *aidif = 1.0*. The reason, according to Anand Gnanadesikan, is because when $R = K_v \delta t / \Delta z^2 > 1$, the explicit treatment of diffusion produces wiggles in the solution, which are then smoothed out by the implicit treatment. Essentially, using the fully implicit treatment gives more smoothing than may be realistic for $R > 1$. However, using a few fully implicit steps at the end of the integration allows for proper averaging in cases where $R >> 1$. In MOM however, the interval of interest is each timestep and without a fully implicit treatement, the wiggles produced by the explicit treatment are left. In order for the most accurate solution of the diffusion term it is necessary to set $aidif = 1.0$ for all cases where $R > 1$.

Option *implicitvmix* solves both the vertical diffusion of tracers and vertical diffusion of horizontal velocity components implicitly. When option *implicitvmix* is <u>not</u> enabled, the vertical diffusion of horizontal velocity components is always solved explicitly. However, in the case where option *implicitvmix* is <u>not</u> enabled, vertical diffusion of tracers is still solved implicitly if option *isoneutralmix* is enabled.

When option *isoneutralmix* is enabled, the vertical diffusion of tracers is solved implicitly to handle stability problems brought about by large vertical mixing coefficients. Recall that mixing is along isoneutrals, and the maximum slope is constrained by the small slope approximation. However, explicit convection operates when *isoneutralmix* is enabled to handle convective regions which cannot be dealt with by *isoneutralmix*. Note that when option *full_tensor* is used with *isoneutralmix*, the isoneutral may wrap over on itself vertically but mixing is still along the isoneutral. Explicit convection of tracers will break through the isoneutral and cause it to align with the vertical.

Therefore, explicit convection of tracers as described in Section 33.1 operates only when option *implicitvmix* is <u>not</u> enabled or option *isoneutralmix* <u>is</u> enabled.

Assume that the one dimensional vertical diffusion equation to be solved implicitly is given by

$$\xi_k = \xi_k^\star + aidif \cdot 2\Delta\tau\delta_z(dcb_k \cdot \delta_z(\xi_k)) \tag{38.4}$$

for levels $k = 1$ to $k = km$ where ξ_k is the vertical profile of a tracer or a horizontal velocity component at time level $\tau + 1$. The given quantities are dcb_k which is the diffusion coefficient at the bottom of T or U cells, $\Delta\tau$ which is the time step, $aidif$ which is the implicit factor, and $\xi_k^\star$. If ξ is a tracer , $\xi_k^\star$ is known from the solution of Equation (22.77). If ξ is a horizontal component of velocity, $\xi_k^\star$ is the solution from Equation (22.116).

Equtation (38.4) is solved subject to flux boundary conditions at the top of the first cell at $k = 1$ (given as $sflux$) and base of the bottom cell at $k = kz$ (given as $bflux$). The solution follows from pages 42 and 43 of Numerical Recipes (1992). Note that when ξ is a horizontal

component of velocity, $sflux = smf$ and $bflux = bmf$. When ξ is a tracer, $sflux = stf$ and $bflux = btf$. Equation (38.4) can be written as

$$\xi_k = \xi_k^\star + aidif \cdot 2\Delta\tau(dcb_{k-1}\frac{\xi_{k-1} - \xi_k}{dzt_k \cdot dzw_{k-1}} - dcb_k\frac{\xi_k - \xi_{k+1}}{dzt_k \cdot dzw_k}) \tag{38.5}$$

which can be re-arranged to

$$A_k \cdot \xi_{k-1} + B_k \cdot \xi_k + C_k \cdot \xi_{k+1} = \xi_k^\star \tag{38.6}$$

where

$$A_k = -\frac{aidif \cdot 2\Delta\tau \cdot dcb_{k-1}}{dzt_k \cdot dzw_{k-1}} \tag{38.7}$$

$$C_k = -\frac{aidif \cdot 2\Delta\tau \cdot dcb_k}{dzt_k \cdot dzw_k} \tag{38.8}$$

$$B_k = 1 - A_k - C_k \tag{38.9}$$

At $k = 1$, $A_k = 0$ and at $k = kz$, $C_k = 0$. Note that the last ocean level kz may be less than bottom level km to accommodate bottom topography. The boundary conditions at the top $k = 1$ and bottom $k = kz$ are imposed by setting

$$F_{k=1} = \xi_{k=1}^\star + \frac{sflux \cdot aidif \cdot 2\Delta\tau}{dzt_{k=1} \cdot dzw_{k=0}} \tag{38.10}$$

$$F_k = \xi_k^\star \tag{38.11}$$

$$F_{k=kz} = \xi_{k=kz}^\star - \frac{bflux \cdot aidif \cdot 2\Delta\tau}{dzt_{k=kz} \cdot dzw_{k=kz}} \tag{38.12}$$

The solution is arrived at by performing a decomposition and forward substitution using

$$bet = B_{k=1}$$
$$\xi_{k=1} = F_{k=1}/bet$$

do k = 2,kz

$$E_k = C_{k-1}/bet$$
$$bet = B_k - A_k \cdot E_k$$
$$\xi_k = (F_k - A_k \cdot \xi_{k-1})/bet$$

enddo

then a back substitution using

do k = kz-1,1,-1

$$\xi_k = \xi_k - E_{k+1} \cdot \xi_{k+1}$$

enddo

38.6 beta_plane

Normally, the equations in MOM are formulated in spherical coordinates. This option turns the model into a beta plane: $f = f_\circ + \beta \cdot y$ where $\beta = \partial f / \partial_y$ and $f_\circ$ is taken at the latitude given by $\phi^U_{jrow=1}$.

38.7 f_plane

Normally, the equations in MOM are formulated in spherical coordinates. This option turns the model into a f plane as in option *beta_plane* with $\beta = 0$. Choosing $f_\circ$ as the equator sets a flat space cartesian grid.

38.8 source_term

This option allows adding source terms to the momentum and tracer equations as indicated in Sections 22.9.4 and 22.8.4.

38.9 readrmsk

This option allows importing region masks $mskhr_{i,jrow}$and $mskvr_k$into MOM for use with certain diagnostics.

38.10 show_details

When enabled, this option allows details from various parts of the setup calculations to be printed to file *stdout*. When MOM executed, the printout indicates where this option will give more information if enabled. When enabled, it leads to lots of printout and is left disabled unless the missing details are needed.

38.11 timing

This option allows any Fortran source code or portions of the code to be timed using a simple set of timing routines. Executing script *run_timer* will exercise these routines by solving a tracer equation in various ways. This is useful when trying to optimize speed for a particular computer platform. Refer to Section 15.1.8 for details.

38.12 equivalence_mw

This option hides the nine two dimension fields of the coefficient matrix for inverting the external mode elliptic equation over the memory window space. This can be done since the external mode and internal mode are essentially orthogonal calculations and the space required by one can be used by the other.

Part VIII

Diagnostic options

Chapter 39

Design of diagnostic options

MOM is instrumented with a variety of diagnostic options. Some are useful for diagnosing model problems while others are aimed towards providing information to help resolve questions of a more scientific nature. All are independent of each other and each is activated with its own option at compile time. For added flexibility, each diagnostic has an associated interval, control, and possibly an averaging period variable which are input through namelist. To see all namelist variables, refer to Section 14.4.

When not enabled, a diagnostic requires neither memory nor computational time. When enabled, some require large amounts of memory, disk, or cpu time so it is important to use them cautiously with specific goals in mind. The amount of time generally depends on the particular configuration of MOM, which diagnostics are enabled, the interval between diagnostic output, and the averaging period (if applicable). An assessment of the computational time[1] can easily be made by enabling the required diagnostics along with option *timing*[2] in a short model execution.

The purpose of this chapter is to introduce some general ideas about the design of diagnostics in MOM. Subsequent chapters describe the different physical and numerical analysis options.

39.1 Ferret

MOM can produce many types of diagnostics and all are described within this chapter. How to organize and view results from these diagnostics has been a major problem in the past. With the adoption of a NetCDF standard, results can now be viewed almost without effort. A good way to visualize and further analyze results is to use Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (URL: http://ferret.wrc.noaa.gov/Ferret/ email: ferret@pmel.noaa.gov). This package is highly recommended because it works well with large datasets, supports non-uniform staggered grids, allows dynamic regridding of data, abstract variables, modulo axes, and contains many useful transformations and built in functions.

[1]Note that when enabling lots of diagnostics, substantial reduction in computational time can be realized by opening up the memory window even on a single processor. As an example, execute the test case script *run_mom* and open the memory window wider than the minimum (jmw exceeds 3 in file *size.h*). Of course, the price to be paid is an increase in the required memory.

[2]After timing a particular configuration, option *timing* should always be disabled because the act of timing takes non-negligible time!

39.2 Naming Diagnostic files

Many diagnostic options may be enabled during a single run of MOM. Each one writes data to a filename which is unique to the particular diagnostic. For instance, data from diagnostic option *snapshots* will be written to a filename containing the keyword "snapshots" while data from option *tracer_averages* will be written to a filename containing the keyword "tracer_avg".

Within a given diagnostic, if data is written as unformatted 32 bit IEEE, then the output filename is given the suffix *yyyyyy.mm.dd.dta*. If data is written in NetCDF format, then the output filename is given the suffix *yyyyyy.mm.dd.dta.nc*. The "yyyyyy.mm.dd" is a place holder for the year, month, and day information. The purpose of this infomation is to allow for easy file management using UNIX.

If option *separate_diag_files* is enabled, the year, month, and day correspond to the timestep during which the diagnostic data was written. If the interval for saving data results in multiple datasets being written during one run, then each dataset will have a distinct filename (i.e. same keyword name but different date information). If option *separate_diag_files* is not enabled, the year, month, and day information in the filename will correspond to the last timestep of the run. So multiple datasets written out for a particular diagnostic during a single run will all be appended to one filename.

Since data from options *time_step_monitor* and *trajectories* are typically written out very often within a model run, both options write to a filenames which corresponds to the last timestep of the run.

39.3 Format of diagnostic data files

In general, diagnostic output can be written in either 32bit unformatted IEEE or NetCDF format. In both cases, the actual writing of data is done through the I/O interface described in http://www.gfdl.gov/ vb. The option for NetCDF format is *use_netCDF*. If not enabled, then data will be written in 32bit unformatted IEEE.

Saving diagnostic data in NetCDF format is the preferred approach since diagnostic output is immediately accessible to visualization packages which support NetCDF. This means that there is no need to write and maintain code for manipulating diagnostic data in order to visualize it. NetCDF format also allows data to be passed between various computer platforms easily and is therefore the preferred method for sharing data.

Two options affect the efficiency of writing data. Option *diagnostics_himem* uses large buffers to speed up writing of data at the expense of using extra memory. Option *diagnostics_lomem* uses small buffers and increases computer time but uses much less memory than the former option. Option *diagnostics_himem* is the default.

39.4 Sampling data

Depending on the particular diagnostic, output may consist of instantaneous or time averaged data. Instantaneous data is written out periodically on those time steps that fall nearest to the end of a specified interval. For instance, specifying a 30 day interval means that results are written out every 30 days from some specified reference time. The reference time is also specified through namelist and is the same for all diagnostics.

Averaged data starts out being accumulated over all time steps within a specified interval. It is then averaged and written out periodically at the end of the interval. If the interval were

30 days, results would be output every 30 days from the reference time and would represent 30 day averages. It is possible to produce sub sampled averages by specifying an averaging period as less than an interval. The averaging period is also input through namelist. For example, if the interval was set to 30 days and the averaging period was set to 2.0 days, then 2 day averages would be written out every 30 days and the averaging would be over days 29 and 30 of each interval. How about producing monthly averages when months vary in length? Enabling option *monthly_averages* will over-ride values specified by "interval" and "period" and make it so. However, this option is global and applies to all time averaged diagnostics that are enabled. Refer to Section 40.11 for more details.

How often is it necessary to sample model generated data? It should be sampled often enough to resolve the shortest time scale which is of interest. Of course, it follows that longer time scales will also be resolved. For example, if data is sampled at an interval of once per month, the implicit assumption is that time scales with periods of four months and longer are being resolved.

What happens when the data has energy at periods of one week? If data is sample instantaneously once per month, then sampled data will be aliased by the shorter period. One way to remove this alias is to produce a monthly climatology. This is done by saving data for many years and averaging all Januaries together, all Februaries together and so forth to produce twelve climatological or "mean" months. The alias error reduces as more and more years are included in the climatology.

A better way is to filter out energy at periods of one week by constructing monthly averages which are then saved once per month. A dataset of monthly averages represents estimates that are statistically more stable than a dataset of instantaneous values. If there is little energy at periods shorter than the period of interest, then saving instantaneous samples is essentially the same as saving monthly averages.

The averaging period need not always equal the interval at which results are saved. This applies when frequencies are widely separated. For instance, suppose that the amplitude of a diurnal period (one cycle per day) is not negligible compared to the amplitude of an annual period (one cycle per year). The diurnal period can be effectively removed and the annual period resolved by saving a three or four day average at the end of every month.

39.5 Regional masks

For use with certain diagnostics calculations, the model domain may be sub-divided into a number of regions over which calculations are averaged. An arbitrary number of non-overlapping regions of areal extent are defined by setting a horizontal region mask number $mskhr_{i,jrow} = m$ where $(i, jrow)$ is in region m for $m = 1 \cdots nhreg$ and $nhreg$ is the number of regions. One way to do this is to use the subroutine *sethr* which assigns $mskhr_{i,jrow}$ to a region number within a rectangular region. Regions may be built up from calls to *sethr* but in general need not be simply rectangular.

In a similar fashion, the vertical region mask number $mskvr_k = \ell$ where k is in vertical region ℓ for $\ell = 1 \cdots nvreg$ and $nvreg$ is the number of vertical regions. Unlike $mskhr_{i,jrow}$ which may contain a region which is multiply connected, the k indices within a vertical region ℓ must be contiguous.

Regional volumes are constructed in subroutine *setocn* by the union of $mskhr_{i,jrow}$ and $mskvr_k$ such that the regional volume numbers are given by

$$nreg = nhreg \cdot (mskvr_k - 1) + mskhr_{i,jrow} \tag{39.1}$$

39.6 A note about areas on the sphere

The area of a grid box on the sphere is given by

$$A = a^2 \int_{\lambda_1}^{\lambda_2} d\lambda \int_{\phi_2}^{\phi_1} \cos\phi \, d\phi. \tag{39.2}$$

In a grid-point model with uniform resolution, this area is often approximated with

$$A_{approx} = a^2 \, \Delta\lambda \, \Delta\phi \, \cos\overline{\phi}, \tag{39.3}$$

where

$$\overline{\phi} = \frac{\phi_1 + \phi_2}{2} \tag{39.4}$$

represents the midpoint latitude. The exact area of the finite sized grid box is given by

$$\begin{aligned} A_{exact} &= a^2 \, \Delta\lambda \int_{\phi_2}^{\phi_1} d(\sin\phi) \\ &= a^2 \, \Delta\lambda \, \Delta(\sin\phi). \end{aligned} \tag{39.5}$$

It is useful to determine the error made in the approximate expression through the use of some trigonometry:

$$\begin{aligned} \Delta(\sin\phi) &= \sin\phi_2 - \sin\phi_1 \\ &= 2\cos\overline{\phi}\,\sin(\Delta\phi/2) \\ &\approx \Delta\phi\,\cos\overline{\phi}\,(1 - (\Delta\phi)^2/24) \end{aligned} \tag{39.6}$$

Hence,

$$\begin{aligned} A_{exact} &= a^2 \, \Delta\lambda \, \Delta(\sin\phi) \\ &\approx A_{approx}\,(1 - (\Delta\phi)^2/24). \end{aligned} \tag{39.7}$$

As a result, the area of the grid box is overestimated by the amount $(\Delta\phi)^2/24$ when using the approximate expression. For a model with $\Delta\phi = 4^\circ = 0.0698 rad$, the leading error is

$$(\Delta\phi)^2/24 = 0.0002 = 0.02\% \tag{39.8}$$

It is important to note that the above error estimate is a lower bound for the case of a grid where the cosine factor does not represent the cosine of the latitude at the center of the grid box (as in a Gaussian grid). Atmospheric models at GFDL use the exact areas, not the approximate values. Hence, care must be taken to use the same area weights between an atmospheric model and MOM. Unless the weights are identical, flux conservation is not possible. Currently, the approximate area method is used for computing diagnostics in MOM. It is clear that for the purpose of ocean-only diagnostics, the differences are quite minor and can be safely ignored. When time permits, these approximate area calculations will be replaced by the correct ones.

Chapter 40

Diagnostics for physical analysis

Physical science is involved with MOM at two central stages. First, there is the formulation of the basic equations of the model, and the physical assumptions inherent in that formulation. The next stage is in the analysis of solutions obtained from the model. It is for the second stage that the analysis tools described in this chapter are designed. The options are presented here in alphabetical order.

40.1 cross_flow_netcdf

How much of the velocity field is along the neutral directions, and how much is in the dianeutral direction? Option *cross_flow_netcdf* computes the projection of the velocity field into a component parallel to the neutral direction and a component in the dianeutral direction. When combined with the diagnostic option *local_potential_density_terms*, such a partitioning of the velocity vector is useful for diagnosing the mechanisms of water mass transformation.

Note that in order to compute the neutral directions, it is necessary to compute the derivatives ρ_θ and ρ_s (see below). These fields are computed in the model only when option *isoneutralmix* is enabled. If one wishes to employ the diagnostic option *cross_flow_netcdf* yet does not wish to diffuse the tracers with option *isoneutralmix*, it is necessary to run with option *isoneutralmix* on, yet with the isoneutral diffusivity *ahiso* set to zero.

40.1.1 Continuous formulation

The partitioning of the velocity vector into a neutral and dianeutral component is straightforward. Consider now the relevant kinematics in the continuum. Let the unit vector $\hat{\gamma}$ define the dianeutral direction. This vector is given by

$$\hat{\gamma} = \frac{\rho_\theta \, \nabla\theta + \rho_s \, \nabla s}{|\rho_\theta \, \nabla\theta + \rho_s \, \nabla s|}, \tag{40.1}$$

where

$$\rho_\theta = \frac{\partial \rho}{\partial \theta} \tag{40.2}$$

$$\rho_s = \frac{\partial \rho}{\partial s} \tag{40.3}$$

are the partial derivatives of *in situ* density with respect to the active tracers potential temperature and salinity. For more discussion, see Appendix C or Griffies *et al.* (1998) in which the discretization of isoneutral diffusion is discussed. Note that in the case of a single active tracer, the dianeutral direction is simply in the direction of the active tracer gradient. Also note that in many regions of the ocean, the denominator is dominated by the vertical tracer gradients. This result may motivate approximating the denominator by $\rho_\theta \, \partial_z \theta + \rho_s \, \partial_z s$. However, such an approximation introduces a spurious singularity in vertically unstratified regions. Therefore, it is useful to compute the full denominator for the purposes of this diagnostic.

Given the expression (40.1) for the dianeutral direction, it is possible to split the velocity field into the dianeutral and neutral components

$$\begin{aligned} \vec{u} &= (\vec{u} - \vec{D}) + \vec{D} \\ &\equiv \vec{N} + \vec{D}, \end{aligned} \tag{40.4}$$

where

$$\vec{N} = \vec{u} - \vec{D} \tag{40.5}$$

defines the component of the velocity field parallel to the neutral direction, and

$$\vec{D} = (\vec{u} \cdot \hat{\gamma}) \, \hat{\gamma} \tag{40.6}$$

defines the dianeutral velocity vector. These are the two vectors which are provided by this diagnostic. Mapping these vectors on top of potential density isolines gives an approximate view of how much of the currents are along and how much are across the neutral directions. The approximation occurs when noting that neutral directions are not quite parallel to potential density isolines. But for regions not extending too much in the vertical direction (e.g., a few hundred to a 1000 meters), the approximation is rather good (e.g., McDougall 1987).

40.1.2 Discretization

When discretizing, there will be some averaging that must be made in order to provide a consistent computation of the neutral and dianeutral velocity vectors. First, it is necessary to assume a position on the lattice at which the different components will be computed. For the present diagnostic purposes, it is not crucially important how the averaging is done, only that it is done in a consistent manner. One simple choice is to average all quantities so that they lie at the tracer point. This choice is motivated by having the density derivatives ρ_θ and ρ_s already available at the tracer points in the computation of isoneutral diffusion (Appendix C). This choice leads to the discretization of the numerator of the dianeutral direction

$$\begin{aligned} &\rho_\theta \, \nabla\theta + \rho_s \, \nabla s \approx \\ &\quad (\rho_\theta)_{i,k,j} \left(\frac{1}{\cos\phi^T_{jrow}} \overline{\delta_\lambda(t_{i-1,k,j,1,tau})}^{\lambda} \cdot \hat{x} \; + \; \overline{\delta_\phi(t_{i,k,j-1,1,tau})}^{\phi} \cdot \hat{y} \; + \; \overline{\delta_z(t_{i,k-1,j,1,tau})}^{z} \cdot \hat{z} \right) \\ + \; &(\rho_s)_{i,k,j} \left(\frac{1}{\cos\phi^T_{jrow}} \overline{\delta_\lambda(t_{i-1,k,j,2,tau})}^{\lambda} \cdot \hat{x} \; + \; \overline{\delta_\phi(t_{i,k,j-1,2,tau})}^{\phi} \cdot \hat{y} \; + \; \overline{\delta_z(t_{i,k-1,j,2,tau})}^{z} \cdot \hat{z} \right) \\ = \; &\frac{1}{\cos\phi^T_{jrow}} \left((\rho_\theta)_{i,k,j} \, \overline{\delta_\lambda(t_{i-1,k,j,1,tau})}^{\lambda} + (\rho_s)_{i,k,j} \, \overline{\delta_\lambda(t_{i-1,k,j,2,tau})}^{\lambda} \right) \hat{x} \\ + \; &\left((\rho_\theta)_{i,k,j} \, \overline{\delta_\phi(t_{i,k,j-1,1,tau})}^{\phi} + (\rho_s)_{i,k,j} \, \overline{\delta_\phi(t_{i,k,j-1,2,tau})}^{\phi} \right) \hat{y} \\ + \; &\left((\rho_\theta)_{i,k,j} \, \overline{\delta_z(t_{i,k-1,j,1,tau})}^{z} + (\rho_s)_{i,k,j} \, \overline{\delta_z(t_{i,k-1,j,2,tau})}^{z} \right) \hat{z}, \end{aligned} \tag{40.7}$$

where $\hat{x}$, $\hat{y}$, and $\hat{z}$ are local unit vectors in longitude (eastward), latitude (northward), and vertical (upward). The velocity field must be averaged in order to provide an approximation for its value at the cell $T_{i,k,j}$. The following averaging is employed

$$\tilde{V}_{i,k,j} = \overline{u_{i-1,k,j-1,1,\tau}}^{\lambda\phi} \cdot \hat{x} \; + \; \overline{u_{i-1,k,j-1,2,\tau}}^{\lambda\phi} \cdot \hat{y} \; + \; \overline{adv_vbt_{i,k-1,j}}^{z} \cdot \hat{z}. \tag{40.8}$$

Therefore, the instantaneous discretized dianeutral $\vec{D}$ and neutral $\vec{N}$ velocity fields are given by

$$\vec{D}_{i,k,j} = \left(\tilde{V}_{i,k,j} \cdot \hat{\gamma}_{i,k,j}\right) \; \hat{\gamma}_{i,k,j} \tag{40.9}$$

$$\vec{N}_{i,k,j} = \tilde{V}_{i,k,j} - \vec{D}_{i,k,j} \tag{40.10}$$

Output in 32 bit IEEE unformatted data is not an option for this diagnostic. The three velocity components (zonal, meridional, and vertical) of $\vec{D}$ and $\vec{N}$ are output only in NetCDF format to file *cross.yyyyyy.mm.dd.dta.nc* and the interval between output is specified by namelist variable *crossint* in units of days. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. Refer to Section 14.4 for information on namelist variables.

40.2 density_netcdf

It is useful to construct both a locally referenced potential density $\rho^{full}_{i,k,j}$ and potential density $\sigma_{i,k,j}$ referenced to a specific depth. The superscript *in* is used to differentiate between potential density anomaly (used throughout MOM) and the full potential density. To construct the full density $\rho^{full}_{i,k,j}$ in units of gm/cm^3 at each T cell, the mean reference density must be added to the deviation

$$\rho^{full}_{i,k,j} = \rho_{i,k,j} + \rho^{ref}_{k} \tag{40.11}$$

where ρ^{ref}_{k} is described in Section 15.1.2. When option *potential_density* is enabled, σ_0 (referenced to the surface), σ_1 (referenced to 1000m), σ_2 (referenced to 2000m), and σ_3 (referenced to 3000m) are also constructed and saved.

Using, $\rho_{i,k,j}$, the hydrostatic pressure is given by integrating Equation 21.42 to yield

$$p_{i,k,j} = grav \cdot dzw_0 \cdot \rho_{i,1,j} + \sum_{m=2}^{k} grav \cdot dzw_{m-1} \overline{\rho_{i,m-1,j}}^{z} \tag{40.12}$$

The internal mode pressures in units of $gm/cm/sec^2$ are given by removing the vertical mean

$$p^{int}_{i,k,j} = p_{i,k,j} - \frac{1}{zw_{k=kmt(i,jrow)}} \sum_{k=1}^{kmt(i,jrow)} p_{i,k,j} \cdot dzt_k \tag{40.13}$$

The external mode pressure is available from diagnostic *diagnostic_surf_height*. Note that output from this diagnostic is only available in NetCDF format in file *density.yyyyyy.mm.dd.dta.nc* and the interval between output is specified by namelist variable *densityint* in units of days. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. Refer to Section 14.4 for information on namelist variables.

40.3 diagnostic_surf_height

Option *diagnostic_surf_height* constructs an average sea surface elevation from the prognostic stream function. It does this by accumulating a forcing term in time, then averaging over a specified interval (because this is a linear problem) to produce an average forcing X and solving an elliptic equation of the form

$$\nabla \cdot H \nabla \overline{p^s} = X \tag{40.14}$$

for average surface pressure $\overline{p^s}_{i,jrow}$ which is then converted into height using

$$dsp_{i,jrow} = \frac{\overline{p^s}_{i,jrow}}{\rho_\circ \cdot grav} \tag{40.15}$$

where the height $dsp_{i,jrow}$ is in units of cm.

The process begins after solving for the external mode stream function. The surface pressure gradient terms can be reconstructed from Equations (29.3) and (29.4) by replacing the vertically integrated velocities by ψ with the aid of Equations (29.6) through (29.10) to yield

$$\begin{aligned} \frac{1}{\rho_\circ \cos\phi^U_{jrow}} \delta_\lambda(\overline{p^s_{i,jrow}}^\phi) &= \frac{1}{2\Delta\tau}\left(\frac{1}{H_{i,jrow}} \delta_\phi(\overline{\Delta\psi_{i,jrow}}^\lambda)\right) + \frac{\tilde{f}_{jrow}}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\Delta\psi_{i,jrow}}^\phi) \\ &\quad + zu_{i,jrow,1} \end{aligned} \tag{40.16}$$

$$\begin{aligned} \frac{1}{\rho_\circ} \delta_\phi(\overline{p^s_{i,jrow}}^\lambda) &= \frac{1}{2\Delta\tau}\left(-\frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\Delta\psi_{i,jrow}}^\phi)\right) + \frac{\tilde{f}_{jrow}}{H_{i,jrow}} \delta_\phi(\overline{\Delta\psi_{i,jrow}}^\lambda) \\ &\quad + zu_{i,jrow,2} \end{aligned} \tag{40.17}$$

where $\tilde{f}_{jrow}$ is given by Equation (29.5). Since the coefficient matrix required for solving Equation (40.14) is generated within MOM for purposes of solving the prognostic surface pressure and implicit free surface methods, this coefficient matrix can be easily generated. It is done by essentially replacing $1/H$ with H in the method given within Section 31.2. The average forcing X is computed as

$$X = \frac{1}{L} \sum_{\ell=1}^{L} \delta_\lambda \left(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}} \cdot \delta_\lambda(\overline{p^s_{i-1,jrow-1}}^\phi)}^\phi \right) + \delta_\phi \left(\overline{H_{i-1,jrow-1} \cdot \cos\phi^U_{jrow-1} \cdot \delta_\phi(\overline{p^s_{i-1,jrow-1}}^\lambda)}^\lambda \right) \tag{40.18}$$

with the aid of Equations (40.16) and (40.17) where L is the number of timesteps in the averaging period. Equation (40.14) is then inverted for $\overline{p^s}_{i,jrow}$ by the method of conjugate gradients.

Equations (40.16) and (40.17) may be solved directly by line integrals to construct $\overline{p^s}_{i,jrow}$. However, the integration paths must be chosen carefully so as not to propagate errors introduced by solving the stream function equation exactly (hy using a non-zero tolerance "tolrsf"). Choosing nine point numerics by enabling option *sf_9_point* minimizes these errors compared to using option *sf_5_point*.

The solution $\overline{p^s}_{i,jrow}$ is thus an average and may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *diag_surf.yyyyyy.mm.dd.dta*. If option *netcdf* or *diagnostic_surf_height_netcdf* is enabled, data is written in NetCDF format to file *diag_surf.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *dspint* and the control is specified by variable *iodsp*. The averaging period *dspper* is typically set the same as the interval but may be specified shorter. How about producing monthly averages when months vary in length? Enabling option *monthly_averages* will over-ride values in *dspper* and *dspint* and make it so. However, this option is global and applies to all time averaged diagnostics that are enabled. Refer to Section 40.11 for more details. Variables *dspper* and *dspint* are input through namelist. Refer to Section 14.4 for information on namelist variables.

40.4 energy_analysis

Option *energy_analysis* computes the instantaneous scalar product of $\vec{u}$ and the momentum Equations (4.1) and (4.2) integrated over the entire domain to verify energy conservation as demonstrated for the continuous equations in Section A.1. The finite discrete equivalent is demonstrated in Chapter A. From Equations (4.1) and (4.2) the total velocity $\vec{u}$ is divided into internal mode $\hat{u}$ and external mode $\overline{u}$ components. The work done by each term in Equations (4.1) and (4.2) is broken into internal and external mode contributions. Note that this approach is different than that given in Cox (1984). Units are in $gm/cm/sec^3$.

Define two operators for integrating the scalar product of $\vec{u}$ and any terms in Equations (4.1) and (4.2). In finite difference form, they are

$$< \overline{\alpha_{i,k,j,n}}^{ext} >= \frac{1}{Vol^U} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \overline{u}_{i,j,n,\tau} \cdot (\alpha_{i,k,j,n}) \Delta^U_{i,k,j} \tag{40.19}$$

$$< \overline{\alpha_{i,k,j,n}}^{int} >= \frac{1}{Vol^U} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \hat{u}_{i,k,j,n,\tau} \cdot (\alpha_{i,k,j,n}) \Delta^U_{i,k,j} \tag{40.20}$$

$$\tag{40.21}$$

$< \overline{\alpha_{i,k,j,n}}^{ext} >$ represents the external mode component of the work done by $\alpha_{i,k,j,n}$, $< \overline{\alpha_{i,k,j,n}}^{int} >$ represents the internal mode component of the work done by $\alpha_{i,k,j,n}$, and the relation between j and *jrow* is as described in Section 14.2. The volume element and total volume are given by

$$\Delta^U_{i,k,j} = umask_{i,k,j} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{40.22}$$

$$Vol^U = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^U_{i,k,j} \tag{40.23}$$

Work done by each term in the momentum equations is computed using operators described in Sections 22.9.5.

Total change in kinetic energy

$$E = < \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{ext} > + < \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{int} > \tag{40.24}$$

Non-linear terms

The work done by non-linear terms on the velocity is broken into two parts: G1 is the horizontal part including the metric term and G2 is the vertical part. G1 + G2 should be zero. The "nonlinear error" = G1 + G2 which should be roundoff to within machine precision. On a Cray YMP with 64 bit words, this amounts to the error being about 1x10^{-12} smaller than the leading term in the computation of G1 and G2.

$$\begin{aligned} G1 &= < \overline{ADV_Ux_{i,k,j} + ADV_Uy_{i,k,j} + ADV_metric_{i,k,j,n}}^{ext} > \\ &\quad + < \overline{ADV_Ux_{i,k,j} + ADV_Uy_{i,k,j} + ADV_metric_{i,k,j,n}}^{int} > \qquad (40.25) \\ G2 &= < \overline{ADV_Uz_{i,k,j}}^{ext} > \\ &\quad + < \overline{ADV_Uz_{i,k,j}}^{int} > \qquad (40.26) \end{aligned}$$

Buoyancy

$$B = < \overline{grav \cdot adv_vbt_{i,k,j} \cdot \overline{\rho_{i,k,j}}^{z}}^{ext} > + < \overline{grav \cdot adv_vbt_{i,k,j} \cdot \overline{\rho_{i,k,j}}^{z}}^{int} > \tag{40.27}$$

Horizontal pressure gradients

$$G = < \overline{grad_p_{i,k,j,n}}^{ext} > + < \overline{grad_p_{i,k,j,n}}^{int} > \tag{40.28}$$

The work done by the horizontal pressure gradients should equal the work done by buoyancy. G − B should be zero. The imbalance G − B is the "energy conversion error" which should be roundoff within machine precision. On a Cray YMP with 64 bit words, this amounts to the error being about 1x10^{-12} smaller than either term.

Mixing

The work done by viscous mixing terms on the velocity is broken into two parts

$$\begin{aligned} D1 &= < \overline{DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_metric_{i,k,j,n}}^{ext} > \\ &\quad + < \overline{DIFF_Ux_{i,k,j} + DIFF_Uy_{i,k,j} + DIFF_metric_{i,k,j,n}}^{int} > \qquad (40.29) \\ D2 &= < \overline{DIFF_Uz_{i,k,j}}^{ext} > + < \overline{DIFF_Uz_{i,k,j}}^{int} > \qquad (40.30) \end{aligned}$$

Wind and Bottom drag

$$W1 = < \overline{smf_{i,j,n}}^{ext} > + < \overline{smf_{i,j,n}}^{int} > \tag{40.31}$$

$$W2 = < \overline{bmf_{i,j,n}}^{ext} > + < \overline{bmf_{i,j,n}}^{int} > \tag{40.32}$$

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *energy_int.yyyyyy.mm.dd.dta*. If option *netcdf* or *energy_analysis_netcdf* is enabled, data is written in NetCDF format to file *energy_int.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *glenint* and the control is specified by variable *ioglen*.

40.5 fct_netcdf

When using the FCT advection scheme, it is common to wonder how much the upwind part of FCT is active for any particular flow regime. A simple way to judge this property is to subtract from the FCT advective flux the centered differenced advective flux:

$$\Delta\vec{F} = \vec{F}^{(fct)} - \vec{F}^{(ctr)}. \tag{40.33}$$

The option *fct_netcdf* maps the three components of $\Delta\vec{F}$ for each tracer. In addition, this option maps the convergence

$$-\nabla \cdot (\Delta\vec{F}), \tag{40.34}$$

which tells one how FCT and centered differ as they act in the tracer equation.

When the Gent-McWilliams scheme is implemented through the eddy-velocity formulation (see Section 35.1.6.2) and FCT is employed, the FCT advective flux incorporates both the resolved velocity and the GM velocity. The diagnostic option *fct_netcdf* takes this fact into account.

Output from diagnostic option *fct_netcdf* is only available in NetCDF format. If option *fct_netcdf* is enabled, data is written in NetCDF format to file *fct.yyyyyy.mm.dd.dta.nc*. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *fctint* and the data is instantaneous.

40.6 gyre_components

Option *gyre_components* computes instantaneous values of various components of northward tracer transport. The longitudinally averaged temperature and meridional velocity are constructed for the latitude of ϕ^U_{jrow}as a function of depth. Also, the vertically averaged temperature and meridional velocity are constructed for the latitude of ϕ^U_{jrow}as a function of longitude

$$< \overline{T}^{\phi}_{k,j} >^{\lambda} = \frac{1}{Volx^T_{k,j}} \sum_{i=2}^{imt-1} \overline{t_{i,k,j,n,\tau}}^{\phi} \Delta^T_i \tag{40.35}$$

$$< V_{k,j} >^{\lambda} = \frac{1}{Volx^U_{k,j}} \sum_{i=2}^{imt-1} u_{i,k,j,2,\tau} \Delta^U_i \tag{40.36}$$

$$< \overline{T}^{\phi}_{i,j} >^{z} = \frac{1}{Volz^{T}_{i,j}} \sum_{k=1}^{km} \overline{t_{i,k,j,n,\tau}}^{\phi} \Delta^{T}_{k} \tag{40.37}$$

$$< V_{i,j} >^{z} = \frac{1}{Volz^{U}_{i,j}} \sum_{k=1}^{km} adv_vnt_{i,k,j} \Delta^{U}_{k} \tag{40.38}$$

Note that a factor of $\cos \phi^{U}_{jrow}$is built into $adv_vnt_{i,k,j}$. The volume elements and volume of the latitude strip as a function of depth are

$$\Delta^{T}_{i} = tmask_{i,k,j} \cdot tmask_{i,k,j+1} \cdot dxt_{i} \tag{40.39}$$

$$\Delta^{U}_{i} = dxui \cdot \cos \phi^{U}_{jrow} \tag{40.40}$$

$$\Delta^{T}_{k} = tmask_{i,k,j} \cdot tmask_{i,k,j+1} \cdot dzt_{k} \tag{40.41}$$

$$\Delta^{U}_{k} = dzt_{k} \tag{40.42}$$

$$Volx^{T}_{k,j} = \sum_{i=2}^{imt-1} \Delta^{T}_{i} \tag{40.43}$$

$$Volx^{U}_{k,j} = \sum_{i=2}^{imt-1} \Delta^{U}_{i} \tag{40.44}$$

$$Volz^{T}_{i,j} = \sum_{k=1}^{km} \Delta^{T}_{k} \tag{40.45}$$

$$Volz^{U}_{i,j} = \sum_{k=1}^{km} \Delta^{U}_{k} \tag{40.46}$$

$$\tag{40.47}$$

The canonical form of the northward components of tracer transport by various means and deviations is given as

$$ttn_{1,jrow,n} = \sum_{k=1}^{km} < \overline{T}^{\phi}_{k,j} >^{\lambda} \cdot < V_{k,j} >^{\lambda} \cdot dzt_{k} \tag{40.48}$$

$$ttn_{2,jrow,n} = ttn_{6,jrow,n} - ttn_{1,jrow,n} \tag{40.49}$$

$$ttn_{3,jrow,n} = \sum_{i=2}^{imt-1} < \overline{T}^{\phi}_{i,j} >^{z} \cdot < V_{i,j} >^{z} \cdot dxt_{i} \tag{40.50}$$

$$ttn_{4,jrow,n} = ttn_{6,jrow,n} - ttn_{3,jrow,n} - ttn_{5,jrow,n} \tag{40.51}$$

$$ttn_{5,jrow,n} = \sum_{i=2}^{imt-1} -(\overline{smf_{i-1,j,1} \cdot dxu_{i-1}}^{\lambda}) \cdot (\overline{t_{i,1,j,n,\tau}}^{\phi} - < \overline{T}^{\phi}_{i,j} >^{z}) \cdot \frac{\cos \phi^{U}_{jrow}}{f_{jrow}} \tag{40.52}$$

$$ttn_{6,jrow,n} = \sum_{k=1}^{km} \sum_{i=2}^{imt-1} 0.5 \cdot adv_fn_{i,k,j} \cdot \Delta^{T}_{i} \cdot dzt_{k} \tag{40.53}$$

$$ttn_{7,jrow,n} = \sum_{k=1}^{km} \sum_{i=2}^{imt-1} diff_fn_{i,k,j} \cdot \Delta^{T}_{i} \cdot dzt_{k} \tag{40.54}$$

$$ttn_{8,jrow,n} = ttn_{6,jrow,n} + ttn_{7,jrow,n} \tag{40.55}$$

Note the factor of 0.5 which is needed to correct the advective flux of tracer as described in Section 22.8.2. These terms may also be broken down as a function of latitude within $mskhr_{i,jrow}$.

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *gyre_comp.yyyyyy.mm.dd.dta*. If option *netcdf* or *gyre_components_netcdf* is enabled, data is written in NetCDF format to file *gyre_comp.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *gyreint* and the control is specified by variable *iogyre*.

40.7 local_potential_density_terms

Option *local_potential_density_terms* diagnoses processes affecting the time evolution of locally referenced potential density. These processes include

1. Advection
2. Vertical diffusion
3. Laplacian horizontal diffusion
4. Laplacian skew-diffusion
5. Biharmonic skew-diffusion
6. Cabbeling
7. Thermobaricity
8. Halobaricity.

Such a diagnosis can be useful for understanding processes affecting water mass formation and transformation. Note that the added important effects from convective mixing can be deduced from the option *save_convection* (section 40.12) and so are not built into option *local_potential_density_terms*. In order to diagnose cabbeling, thermobaricity, and halobaricity, the code assumes that the isoneutral diffusion scheme is employed using the small angle approximation (option *isoneutralmix* described in Section 35.1). If Gent-McWilliams is used, then its effects on locally referenced potential density is computed in terms of skew-diffusion (see Section 35.1.6).

The first part of this section describes the breakdown of terms in the locally referenced potential density equation, the second part describes the discretization of these terms, and the third part describes the model output.

In coding this diagnostic, it was decided that a straightforward approach should be taken, rather than one which involves tricks to save computation time and memory. Consequently, the code should be rather easy to read. Yet, due to the rather large number of terms to compute, this diagnostic is currently expensive.

40.7.1 Locally referenced potential density equation

Some of the discussion in this section can be found in McDougall (1991) and Appendix B in Griffies *et al.*, (1998). What is of interest is the time tendency of locally referenced potential density

$$\partial_t \rho = \rho_\theta \, \partial_t \theta + \rho_s \, \partial_t s, \tag{40.56}$$

where

$$\begin{aligned} \rho_\theta &= \frac{\partial \rho}{\partial \theta} = -\alpha\rho \\ \rho_s &= \frac{\partial \rho}{\partial s} = \beta\rho \end{aligned} \tag{40.57}$$

are the partial derivatives of density with respect to the active tracers potential temperature θ and salinity s. These derivatives are evaluated at the local temperature, salinity, and pressure. The reason there is no pressure time tendency term in equation (40.56) is due to the local referencing used for locally referenced potential density. In other words, locally referenced potential density is a *local* water mass variable in the sense that it changes only when water mass properties (temperature and salinity) change. Jackett and McDougall (1997) discuss an approximate *global* water mass variable called *neutral density*.

Now split the right hand side of equation (40.56) into various processes using the prognostic equations for temperature and salinity

$$\begin{aligned} \partial_t \rho &= \rho_\theta\, \partial_t \theta + \rho_s\, \partial_t s \\ &= -\vec{u} \cdot (\rho_\theta\, \nabla\theta + \rho_s \nabla s) - (\rho_\theta\, \nabla \cdot \vec{F}(\theta) + \rho_s\, \nabla \cdot \vec{F}(s)) \\ &= -[\rho_\theta\, \nabla(\vec{u}\,\theta) + \rho_s \nabla(\vec{u}\, s)] - [\rho_\theta\, \nabla \cdot \vec{F}(\theta) + \rho_s\, \nabla \cdot \vec{F}(s)] \end{aligned} \tag{40.58}$$

where $\vec{u}$ is the divergence-free current vector. The non-advective tracer flux takes the form

$$\vec{F} = \vec{F}_I + \vec{F}_{skew-lap} + \vec{F}_{skew-bih} + \vec{F}_V + \vec{F}_H, \tag{40.59}$$

where

- $\vec{F}_I$ is the diffusive flux of tracer along the neutral directions.
- $\vec{F}_{skew-lap}$ is the skew-diffusive flux arising from Laplacian skew-diffusion (e.g., option *gent_mcwilliams* discussed in Section35.1.6).
- $\vec{F}_{skew-bih}$ is the skew-diffusive flux arising from biharmonic skew-diffusion (e.g., option *biharmonic_rm* discussed in Section 35.1.8).
- $\vec{F}_V$ is the vertical diffusive flux.
- $\vec{F}_H$ is the Laplacian horizontal diffusive flux.

Note again that convection is absent in this analysis, as its effects on density are readily diagnosed using the option *save_convection*. Additionally, the effects from a biharmonic horizontal diffusive flux currently has not been implemented in this diagnostic.

40.7.1.1 Cabbeling, thermobaricity, and halobaricity

To isolate the mathematical expressions for cabbeling, thermobaricity, and halobaricity, write the convergence of the isoneutral diffusive fluxes in the form

$$-\rho_\theta\, \nabla \cdot \vec{F}_I(\theta) - \rho_s\, \nabla \cdot \vec{F}_I(s) = \nabla\rho_\theta \cdot \vec{F}_I(\theta) + \nabla\rho_s \cdot \vec{F}_I(s) \tag{40.60}$$

where the identity

$$\rho_\theta\, \vec{F}_I(\theta) + \rho_s\, \vec{F}_I(s) = 0 \tag{40.61}$$

was used. This identity represents an important balance between the isoneutral diffusive flux of the two active tracers. It is a manifestation of the absence of a neutral direction diffusive flux of locally referenced potential density. More simply, isoneutral diffusion does not act on buoyancy, and equation (40.61) is a mathematical statement of this physical property. Griffies *et al* 1998 provide further discussion of this balance, and its importance when discretizing isoneutral diffusion. To procede, use the identities

$$\nabla s \cdot \vec{F}_I(\theta) = \nabla\theta \cdot \vec{F}_I(s) \tag{40.62}$$

$$\nabla \rho_\theta = \rho_{\theta\theta}\nabla\theta + \rho_{\theta s}\nabla s + \rho_{\theta\, p}\nabla p \tag{40.63}$$

$$\nabla \rho_s = \rho_{s\theta}\nabla\theta + \rho_{ss}\nabla s + \rho_{s\, p}\nabla p. \tag{40.64}$$

The first identity is most easy to verify when writing the diffusive flux in terms of a symmetric diffusion tensor

$$\begin{aligned} \nabla s \cdot \vec{F}_I(\theta) &= -\partial_i s\, K^{ij}\, \partial_j \theta \\ &= -\partial_j \theta\, K^{ji}\, \partial_i s \\ &= \nabla\theta \cdot \vec{F}_I(s). \end{aligned} \tag{40.65}$$

Identities (40.63) and (40.64) follow from the chain rule. The pressure gradient terms represent the effects of probing different pressure surfaces, and hence different potential density surfaces. Such is necessary for computing the spatial gradients of the thermal and saline expansion coefficients, and they lead to the thermobaric and halobaric effects. When evaluating the horizontal pressure gradient for this diagnostic, the gradient in the lid or surface pressure is ignored. These three identities, along with the balance of temperature and salinity fluxes given by equation (40.61), render

$$\begin{aligned} -\rho_\theta\, \nabla \cdot \vec{F}_I(\theta) - \rho_s\, \nabla \cdot \vec{F}_I(s) &= \\ \rho_{\theta\theta}\nabla\theta \cdot \vec{F}_I(\theta) + \rho_{ss}\nabla s \cdot \vec{F}_I(s) + 2\rho_{\theta s}\nabla s \cdot \vec{F}_I(\theta) + \nabla p \cdot [\rho_{\theta\, p}\vec{F}_I(\theta) + \rho_{s\, p}\vec{F}_I(s)] \\ = \nabla\theta \cdot \vec{F}_I(\theta)[\rho_{\theta\theta} - 2\rho_{\theta s}(\rho_\theta/\rho_s) + \rho_{ss}(\rho_\theta/\rho_s)^2] + \nabla p \cdot [\rho_{\theta\, p}\vec{F}_I(\theta) + \rho_{s\, p}\vec{F}_I(s)]. \end{aligned} \tag{40.66}$$

The terms in equation (40.66) proportional to the second derivatives of density with respect to temperature and salinity are identified as *cabbeling*. There is a useful way to make the form for cabbeling appear in a slightly more tidy and intuitive manner. To do so, introduce the vector

$$\begin{aligned} \vec{V} &= (1, \alpha/\beta) \\ &= (1, -\rho_\theta/\rho_s) \end{aligned} \tag{40.67}$$

and the symmetric tensor

$$\rho_{ab} = \begin{pmatrix} \rho_{\theta\theta} & \rho_{\theta s} \\ \rho_{s\theta} & \rho_{ss} \end{pmatrix}, \tag{40.68}$$

which yields

$$\rho_{\theta\theta}\nabla\theta \cdot \vec{F}_I(\theta) + \rho_{ss}\nabla s \cdot \vec{F}_I(s) + 2\rho_{\theta s}\nabla s \cdot \vec{F}_I(\theta) = \nabla\theta \cdot \vec{F}_I(\theta)\, \rho_{ab} V^a V^b. \tag{40.69}$$

The quadratic form

$$\begin{aligned} \rho_{ab} V^a V^b &= \rho_{\theta\theta} - 2\rho_{s\theta}\,(\rho_\theta/\rho_s) + \rho_{ss}\,(\rho_\theta/\rho_s)^2 \\ &= -\alpha_\theta - 2(\alpha/\beta)\alpha_s + (\alpha/\beta)^2\beta_s \end{aligned} \tag{40.70}$$

can be thought of as the squared length of the vector $\vec{V}$ on the curved potential density surface characterized locally by the metric tensor ρ_{ab}. It can be easily computed in MOM through tabulating the first and second partial derivatives of the density (see Section 40.7.2.1). A fundamental property of seawater is that the total or Gaussian curvature (Aris, 1962) of a potential density surface

$$\begin{aligned} \kappa_{gauss} &= \det(\rho_{ab})(1+\rho_\theta^2+\rho_s^2)^{-1} \\ &= (\rho_{\theta\theta}\,\rho_{ss}-\rho_{\theta s}^2)(1+\rho_\theta^2+\rho_s^2)^{-1} \end{aligned} \tag{40.71}$$

is negative. Consequently,

$$\rho_{ab}V^aV^b \le 0. \tag{40.72}$$

In summary, cabbeling

$$\text{cabbeling} = \nabla\theta\cdot\vec{F}_I(\theta)\;\rho_{ab}V^aV^b \tag{40.73}$$

is written as the product of two separate quadratic forms whose physical content can be individually identified. The first

$$\nabla\theta\cdot\vec{F}_I(\theta) = -\partial_i\theta\,K^{ij}\,\partial_j\theta \tag{40.74}$$

is the projection of the temperature gradient onto the isoneutral diffusive flux of temperature. This term is negative semi-definite for a downgradient isoneutral diffusive temperature flux. In this case, the tensor K^{ij} is the symmetric and positive semi-definite Redi isoneutral diffusion tensor. The second term, $\rho_{ab}V^aV^b$, as just discussed, summarizes certain intrinsic properties of the potential density surface. The combined effects of a negatively curved potential density surface and a downgradient isoneutral diffusive flux of temperature render

$$\text{cabbeling} \ge 0. \tag{40.75}$$

As a result, cabbeling increases the value of the locally referenced potential density ρ, thus moving a water parcel downward.

The terms proportional to the pressure gradient

$$\begin{aligned} \text{thermobaricity + halobaricity} &= \rho_{\theta\,p}\vec{F}_I(\theta)\cdot\nabla p + \rho_{s\,p}\vec{F}_I(s)\cdot\nabla p \\ &= \rho_{\theta\,p}\vec{F}_I(\theta)\cdot\nabla p - \rho_{s\,p}\left(\frac{\rho_\theta}{\rho_s}\right)\vec{F}_I(\theta)\cdot\nabla p \end{aligned} \tag{40.76}$$

represent the effects from thermobaricity and halobaricity. The balance (40.61) was used to obtain the second equality. These processes arise from the pressure dependence of the equation of state for seawater. The ratio

$$\frac{\text{thermobaricity}}{\text{halobaricity}} = -\left(\frac{\rho_{\theta p}\,\rho_s}{\rho_{sp}\,\rho_\theta}\right) \tag{40.77}$$

is quite large in absolute value, indicating the dominance of thermobaricity over halobaricity (McDougall 1987). In contrast to the cabbeling term, both the thermobaric and halobaric terms are sign-indefinite.

40.7.1.2 Summary of the terms forcing locally referenced potential density

In summary, the time tendency of locally referenced potential density can be written

$$\begin{aligned}\partial_t \rho &= \text{advection} + \text{vertical diffusion} + \text{Laplacian horizontal diffusion} \\ &+ \text{Laplacian skew-diffusion} + \text{biharmonic skew-diffusion} \\ &+ \text{cabbeling} + \text{thermobaricity} + \text{halobaricity}, \end{aligned} \tag{40.78}$$

where

$$\text{advection} = -[\rho_\theta \nabla \cdot (\vec{u}\,\theta) + \rho_s \nabla \cdot (\vec{u}\,s)] \tag{40.79}$$

$$\text{vertical diffusion} = -[\rho_\theta \nabla \cdot \vec{F}_V(\theta) + \rho_s \nabla \cdot \vec{F}_V(s)] \tag{40.80}$$

$$\text{Laplacian horizontal diffusion} = -[\rho_\theta \nabla \cdot \vec{F}_H(\theta) + \rho_s \nabla \cdot \vec{F}_H(s)] \tag{40.81}$$

$$\text{Laplacian skew-diffusion} = -[\rho_\theta \nabla \cdot \vec{F}_{skew-lap}(\theta) + \rho_s \nabla \cdot \vec{F}_{skew-lap}(s)] \tag{40.82}$$

$$\text{Biharmonic skew-diffusion} = -[\rho_\theta \nabla \cdot \vec{F}_{skew-bih}(\theta) + \rho_s \nabla \cdot \vec{F}_{skew-bih}(s)] \tag{40.83}$$

$$\text{cabbeling} = \nabla\theta \cdot \vec{F}_I(\theta)\ \rho_{ab} V^a V^b \tag{40.84}$$

$$\text{thermobaricity} = \rho_{\theta\,p} \vec{F}_I(\theta) \cdot \nabla p \tag{40.85}$$

$$\text{halobaricity} = \rho_{s\,p} \vec{F}_I(s) \cdot \nabla p. \tag{40.86}$$

Notice that each term has the dimensions of density per time.

A steady state ocean represents a balance between the terms on the right hand side of equation (40.78). For the simplest case, there is only advection by the *resolved* current $\vec{u}$. In this case, the steady state current is aligned parallel to the neutral directions: $\vec{u} \cdot (\rho_\theta\ \nabla\theta + \rho_s \nabla s) = \vec{u} \cdot \nabla\rho = 0$. Such a flow field is typically associated with steady state adiabatic fluid flow. When there is time dependence, and/or when cabbeling, thermobaricity, halobaricity, diffusion, or skew-diffusion are present, there will be a nonzero dianeutral component to the velocity field: $\vec{u} \cdot \nabla\rho \neq 0$. Note that since the cabbeling term is sign-definite, the balance between advection and cabbeling,

$$\vec{u}_{cab} \cdot \nabla\rho = \nabla\theta \cdot \vec{F}_I(\theta)\ \rho_{ab} V^a V^b, \tag{40.87}$$

results in a dianeutral component to the velocity which is always directed towards values of increasing locally referenced potential density

$$\vec{u}_{cab} \cdot \nabla\rho \geq 0. \tag{40.88}$$

Namely, for a stably stratified fluid, the velocity field which balances cabbeling acts in a downward dianeutral direction (McDougall 1991). The other processes, as they are not sign-definite, do not necessarily lead to a particular flow direction for their associated dianeutral velocity.

The diagnostic option *cross_flow_netcdf* (Section 40.1) allows one to diagnose the component of velocity along and across the neutral directions. In many regions of the ocean, dianeutral advection is presumed to be small. The option *local_potential_density_terms* allows one to diagnose terms in the locally referenced potential density equation and thus to associate a cause for any dianeutral flow in the model.

40.7.2 Discretization

The diagnostic option *local_potential_density_terms* provides a discretization of the terms on the right hand side of equations (40.79)–(40.86). Since they represent source/sink terms for

the time tendency of locally referenced potential density, it is useful to diagnose them at a model tracer point. All terms, except advection, are computed inside the fortran program *ptlrhoterms.F*. The need to separate all of the various processes implies that none of the fluxes computed in *isopyc.F* can be used, since that code, for optimization reasons, combines the various processes to form a single flux vector. As such, it is very important to maintain consistency between the algorithms used in *isopyc.F* and *ptlrhoterms.F*.

Whenever discretizing continuous equations which possess certain local properties, it is not always possible to maintain those local properties on the lattice. Griffies *et al.* (1998) discuss how their discretization of the isoneutral diffusive flux maintains a downgradient property only over an extended "finite volume" rather than on each grid cell. In particular, $\vec{F}_I(\theta) \cdot \nabla\theta$ on the lattice is not constrained to be negative. This result means that the lattice form of cabbeling, $\nabla\theta \cdot \vec{F}_I(\theta)\, \rho_{ab} V^a V^b$, will not generally maintain non-negative values as it does in the continuum. The negative values for cabbeling on the lattice, however, should be minimal and localized.

Note that in the model, the computation of tracer flux components employs the opposite sign to that employed in the continuum formulation given above. Hence, the discrete equations will have a minus sign inserted to account for this convention.

40.7.2.1 Equation of state considerations

As described in Section 15.1.2, the density is evaluated as a cubic approximation (Bryan and Cox, 1972) to the UNESCO equation of state

$$\begin{aligned}
\rho(\tilde{t}, \tilde{s}, k) &= (c_{k,1} + (c_{k,4} + c_{k,7} * \tilde{s}) * \tilde{s} + \\
&\quad (c_{k,3} + c_{k,8} * \tilde{s} + c_{k,6} * \tilde{t}) * \tilde{t}) * \tilde{t} + \\
&\quad (c_{k,2} + (c_{k,5} + c_{k,9} * \tilde{s}) * \tilde{s}) * \tilde{s} \\
&= c_{k,1}\, \tilde{t} + c_{k,2}\, \tilde{s} + c_{k,3}\, \tilde{t}^2 + c_{k,4}\, \tilde{s}\, \tilde{t} + c_{k,5}\, \tilde{s}^2 \\
&+ c_{k,6}\, \tilde{t}^3 + c_{k,7}\, \tilde{s}^2\, \tilde{t} + c_{k,8}\, \tilde{s}\, \tilde{t}^2 + c_{k,9}\, \tilde{s}^3 \qquad (40.89)
\end{aligned}$$

where

$$\tilde{t}_{i,k,j} = t_{i,k,j,1,\tau} - T_k^{ref} \qquad (40.90)$$

$$\tilde{s}_{i,k,j} = t_{i,k,j,2,\tau} - S_k^{ref} \qquad (40.91)$$

are tracer anomalies with respect to some pre-specified reference values. The choice of computing density anomalies is based on the increased numerical accuracy inherent in this approach (see Section 15.1.2 for further discussion). The nine polynomial coefficients $c_{k,1-9}$ in this equation depend on the depth level k. They are defined at the tracer points, which means that the density is defined there as well. The first partial derivative of the density with respect to the active tracers are given in MOM by the quadratic expressions

$$(\rho_\theta)_{i,k,j} = \partial_{\tilde{t}}\, \rho(\tilde{t}, \tilde{s}, k) = c_{k,1} + 2\, c_{k,3}\, \tilde{t} + c_{k,4}\, \tilde{s} + 3\, c_{k,6}\, \tilde{t}^2 + c_{k,7}\, \tilde{s}^2 + 2\, c_{k,8}\, \tilde{s}\, \tilde{t} \qquad (40.92)$$

$$(\rho_s)_{i,k,j} = \partial_{\tilde{s}}\, \rho(\tilde{t}, \tilde{s}, k) = c_{k,2} + c_{k,4}\, \tilde{t} + 2\, c_{k,5}\, \tilde{s} + 2\, c_{k,7}\, \tilde{t}\, \tilde{s} + c_{k,8}\, \tilde{t}^2 + 3\, c_{k,9}\, \tilde{s}^2 \qquad (40.93)$$

The second partial derivatives of density with respect to the active tracers are given by the linear expressions

$$(\rho_{\theta\theta})_{i,k,j} = \partial_{\tilde{t}\,\tilde{t}}\, \rho(\tilde{t}, \tilde{s}, k) = 2\, c_{k,3} + 6\, c_{k,6}\, \tilde{t} + 2\, c_{k,8}\, \tilde{s} \qquad (40.94)$$

$$(\rho_{\theta s})_{i,k,j} = \partial_{\tilde{t}\,\tilde{s}}\, \rho(\tilde{t}, \tilde{s}, k) = c_{k,4} + 2\, c_{k,7}\, \tilde{s} + 2\, c_{k,8}\, \tilde{t} \qquad (40.95)$$

$$(\rho_{ss})_{i,k,j} = \partial_{\tilde{s}\,\tilde{s}}\, \rho(\tilde{t}, \tilde{s}, k) = 2\, c_{k,5} + 2\, c_{k,7}\, \tilde{t} + 6\, c_{k,9}\, \tilde{s} \qquad (40.96)$$

These expressions are tabulated in the routine *dens.h*. The second partial derivatives of the density with respect to an active tracer and pressure are given by

$$\begin{aligned} \rho_{\theta\, p} &= z_p \frac{\partial \rho_\theta}{\partial z} \\ &= -\left(\frac{1}{\rho\, g}\right) \partial_z \rho_\theta \qquad (40.97) \\ \rho_{s\, p} &= z_p \frac{\partial \rho_s}{\partial z} \\ &= -\left(\frac{1}{\rho\, g}\right) \partial_z \rho_s, \qquad (40.98) \end{aligned}$$

where the hydrostatic approximation has been used. Consistent with the Boussinesq approximation used in MOM, the ρg term will be evaluated as $\rho_o g$, where $\rho_o = 1.035 g/cm^3$ and $g = 980.6 cm/sec^2$. The vertical derivatives are discretized in a centered fashion:

$$(\rho_{\theta p})_{i,k,j} = (\partial_{\bar{t}\, p}\, \rho)_{i,k,j} = -\frac{1}{\rho_o\, g} \left(\frac{(\rho_\theta)_{i,k-1,j} - (\rho_\theta)_{i,k+1,j}}{dhwt_{i.k-1,j} + dhwt_{i,k,j}} \right) \qquad (40.99)$$

$$(\rho_{sp})_{i,k,j} = (\partial_{\bar{s}\, p}\, \rho)_{i,k,j} = -\frac{1}{\rho_o\, g} \left(\frac{(\rho_s)_{i,k-1,j} - (\rho_s)_{i,k+1,j}}{dhwt_{i.k-1,j} + dhwt_{i,k,j}} \right). \qquad (40.100)$$

Next to surface and bottom boundaries, the values of $(\rho_{\theta p})_{i,k,j}$ and $(\rho_{sp})_{i,k,j}$ are equated to the value one level away. Note that these vertical derivatives compute differences in *in situ* density; i.e., the two densities are not referenced to the same temperature, salinity, and pressure. The reason is that one is interested in gradients between different potential density surfaces when computing thermobaricity and halobaricity.

40.7.2.2 Advection

Advection is diagnosed using the same advection operators used in the model for the chosen advection scheme. The contribution to locally referenced potential density takes the form

$$advrho_{i,k,j} = -(\rho_\theta)_{i,k,j}\, ADV_{i,k,j,1} - (\rho_s)_{i,k,j}\, ADV_{i,k,j,2}. \qquad (40.101)$$

This term is computed inside of *tracer.F*. Note that the contribution from advection may include some dianeutral effects if using a dissipative advection scheme such as QUICKER or FCT.

40.7.2.3 Vertical diffusion

The contribution from vertical diffusion is given by

$$diffvert_{i,k,j} = (\rho_\theta)_{i,k,j} (diffvert_\theta)_{i,k,j} + (\rho_s)_{i,k,j} (diffvert_s)_{i,k,j}, \qquad (40.102)$$

where the contribution from each tracer takes the form

$$(diffvert_\theta)_{i,k,j} = dz_tr_{i,k,j} \left(diff_fb_{i,k-1,j} - diff_fb_{i,k,j} \right), \qquad (40.103)$$

with

$$diff_fb_{i,k,j} = diff_cbt_{i,k,j}\left(t_{i,k,j,1} - t_{i,k+1,j,1}\right) dz_wtr_{i,k,j}. \tag{40.104}$$

Note that if the double-diffusive aspect of KPP vertical mixing (Section 33.2.3) is enabled, then the vertical diffusion coefficient $diff_cbt_{i,k,j}$ is generally different for temperature and salinity. As inside of *tracer.F*, the surface and bottom tracer fluxes are incorporated into the vertical tracer flux at the top and bottom model levels, respectively. In the integration of the tracer equation in *tracer.F*, vertical diffusion is usually computed implicitly in time. For the purposes of this diagnostic, however, it will be evaluated explicitly.

40.7.2.4 Laplacian horizontal diffusion

The contribution from horizontal diffusion consists of three model processes:

- Nonzero horizontal diffusivity Ah
- Use of option *isotropic_mixed* (Section 35.1.9.3)
- Use of *ahsteep* in regions where the scaled isoneutral diffusivity is less than *ahsteep* (Section 35.1.9).

It is generally written

$$diffhorz_{i,k,j} = (\rho_\theta)_{i,k,j}(diffhorz_\theta)_{i,k,j} + (\rho_s)_{i,k,j}(diffhorz_s)_{i,k,j}. \tag{40.105}$$

The contribution from horizontal diffusion for each tracer takes the form

$$\begin{aligned}(diffhorz)_{i,k,j} &= dx_tr_{i,k,j}\left(diff_fe_{i,k,j} - diff_fe_{i-1,k,j}\right) \\ &+ dy_tr_{i,k,j}\left(diff_fn_{i,k,j} - diff_fn_{i,k,j-1}\right)\end{aligned} \tag{40.106}$$

where

$$diff_fe_{i,k,j} = diff_cet(i,k,j)\,(t_{i+1,k,j,n} - t_{i,k,j,n})\,cstdxur_{i,j} \tag{40.107}$$

$$diff_fn_{i,k,j} = diff_cnt(i,k,j)\,(t_{i,k,j+1,n} - t_{i,k,j,n})\,csu_dyur_j. \tag{40.108}$$

The *isotropic_mixed* and *ahsteep* contributions are added where appropriate.

40.7.2.5 Laplacian skew-diffusion

If the Gent-McWilliams transport is enabled (Section 35.1.6), then the convergence of the GM skew-flux of temperature and salinity are diagnosed. Note that the contribution from GM is computed in terms of the GM skew-flux regardless of whether the GM scheme is actually implemented with the default option *gm_skew*, or with option *gm_advect*. The reason is that the computation of the skew flux is much easier than the alternative advective flux.

The contribution from the skew-flux is given by

$$diffgmskew_{i,k,j} = (\rho_\theta)_{i,k,j}(diffgmskew_\theta)_{i,k,j} + (\rho_s)_{i,k,j}(diffgmskew_s)_{i,k,j}. \tag{40.109}$$

The contribution from temperature and salinity each take the form

$$\begin{aligned}(diffgmskew_\theta)_{i,k,j} &= dx_tr_{i,k,j}\left(skew_fe_{i,k,j} - skew_fe_{i-1,k,j}\right) \\ &+ dy_tr_{i,k,j}\left(skew_fn_{i,k,j} - skew_fn_{i-1,k,j}\right) \\ &+ dz_tr_{i,k,j}\left(skew_fb_{i,k-1,j} - skew_fb_{i,k,j}\right). \end{aligned} \tag{40.110}$$

The skew-flux components are computed just as in the solution of the tracer equation (Section 35.1.6 and Appendix C).

40.7.2.6 Biharmonic skew-diffusion

If the option *biharmonic_rm* is enabled (Section 35.1.8), then the convergence of the Roberts and Marshall biharmonic skew-flux of temperature and salinity is diagnosed. The contribution from the biharmonic skew-flux is given by

$$diffbihskew_{i,k,j} = (\rho_\theta)_{i,k,j}(diffbihskew_\theta)_{i,k,j} + (\rho_s)_{i,k,j}(diffbihskew_s)_{i,k,j}. \tag{40.111}$$

The contribution from temperature and salinity each take the form

$$\begin{aligned}(diffbihskew_\theta)_{i,k,j} &= dx_tr_{i,k,j}\left(bihskew_fe_{i,k,j} - bihskew_fe_{i-1,k,j}\right) \\ &+ dy_tr_{i,k,j}\left(bihskew_fn_{i,k,j} - bihskew_fn_{i-1,k,j}\right) \\ &+ dz_tr_{i,k,j}\left(bihskew_fb_{i,k-1,j} - bihskew_fb_{i,k,j}\right). \end{aligned} \tag{40.112}$$

The biharmonic skew-flux components are computed just as in the solution of the tracer equation (Section 35.1.8.8).

40.7.2.7 Cabbeling, thermobaricity, halobaricity, and partial cells

For cabbeling, thermobaricity, and halobaricity, it is necessary to compute the isoneutral diffusive flux of temperature and salinity. There is a subtle point in the calculation of this flux in the case of partial vertical cells. As discussed in Appendix C, the physical dimension of the diffusive flux is different depending on whether full or partial cells are employed. The different dimensions of the flux are irrelevant for the tracer equation, since it is the convergence which affects the time tendency, and the convergence does have the same dimension. For diagnosing cabbeling, thermobaricity, and halobaricity, however, the different dimensions introduces a problem. It turns out that the full cell discretization has the same dimension as the continuum formulation given above. Hence, for this diagnostic, the partial cell option will be ignored when computing cabbeling, thermobaricity, and halobaricity. This shortcoming only affects the values at the bottom model level. Note that the diagnosis of all other terms affecting locally referenced potential density properly take into account the partial cell option.

40.7.2.8 Cabbeling

The isoneutral diffusion piece $\nabla\theta \cdot \vec{F}_I(\theta)$ of cabbeling is given by

$$ISO_{i,k,j} = \frac{ison_fe_{1,i,k,j}\,\delta_x t_{i,k,j,1,\tau-1} + ison_fe_{1,i-1,k,j}\,\delta_x t_{i-1,k,j,1,\tau-1}}{2}$$

$$
\begin{aligned}
&+ \frac{ison_fn_{1,i,k,j}\ \delta_y t_{i,k,j,1,\tau-1} + ison_fn_{1,i,k,j-1}\ \delta_y t_{i,k,j-1,1,\tau-1}}{2} \\
&+ \frac{ison_fb_{1,i,k,j}\ \delta_z t_{i,k,j,1,\tau-1} + ison_fb_{1,i,k-1,j}\ \delta_z t_{i,k-1,j,1,\tau-1}}{2}, \qquad (40.113)
\end{aligned}
$$

where the isoneutral diffusion flux components are those for temperature, and they are defined by

$$
\begin{aligned}
ison_fe_{n,i,k,j} &= diff_fe^{iso}_{i,k,j} && (40.114)\\
ison_fn_{n,i,k,j} &= diff_fn^{iso}_{i,k,j} && (40.115)\\
ison_fb_{n,i,k,j} &= diff_fb^{iso}_{i,k,j} + K33_{i,k,j}\delta_z t_{i,k,j,n,\tau-1}. && (40.116)
\end{aligned}
$$

The diffusive fluxes $diff_fe^{iso}, diff_fn^{iso}, diff_fb^{iso}$ and diagonal diffusion tensor component $K33$ are defined in Section 35.1. The nonlinear equation of state piece $\rho_{ab}V^aV^b$ is given by

$$
NONLIN_{i,k,j} = (\rho_{\theta\theta})_{i,k,j} - 2(\rho_{\theta s})_{i,k,j}\frac{(\rho_\theta)_{i,k,j}}{(\rho_s)_{i,k,j}} + (\rho_{ss})_{i,k,j}\left(\frac{(\rho_\theta)_{i,k,j}}{(\rho_s)_{i,k,j}}\right)^2. \qquad (40.117)
$$

The cabbeling term is given by the product of these two terms

$$
cabbel_{i,k,j} = -ISO_{i,k,j} * NONLIN_{i,k,j}, \qquad (40.118)
$$

where the minus sign accounts for the minus sign used for computing the discretized flux components.

40.7.2.9 Thermobaricity and halobaricity

In the model, the horizontal hydrostatic pressure gradients are stored in an array $grad_p_{i,k,j}$ (Section 22.9.1). These gradients are defined at the U-cell point, which is the northeast corner of a T-cell in the x-y plane. Since the diffusive flux components are defined at the T-cell faces, it is necessary to average the pressure gradients before multiplying by the tracer flux. Thereafter, an average over the faces of the T-cell is prescribed in order to get the thermobaric and halobaric contributions at the tracer point. For the vertical pressure gradient, the hydrostatic approximation yields $\partial_z p_{i,k,j} = -g\rho_{i,k,j}$. Within the Bousinessq limit, this pressure gradient is evaluated as $-g\rho_o$, where $\rho_o = 1.035 g/cm^3$. In symbols, this prescription yields for the contribution to thermobaricity and halobaricity

$$
\begin{aligned}
thermo_{i,k,j} = -(\rho_{p\theta})_{i,k,j}\Bigg(&\frac{(grad_p_{i,k,j,1} + grad_p_{i,k,j-1,1})\ ison_fe_{1,i,k,j}}{4} \\
&+ \frac{(grad_p_{i-1,k,j,1} + grad_p_{i-1,k,j-1,1})\ ison_fe_{1,i-1,k,j}}{4} \\
&+ \frac{(grad_p_{i,k,j,2} + grad_p_{i-1,k,j,2})\ ison_fn_{1,i,k,j}}{4} \\
&+ \frac{(grad_p_{i,k,j-1,2} + grad_p_{i-1,k,j-1,2})\ ison_fn_{1,i,k,j-1}}{4} \\
&- g\ \rho_o\left[\frac{(ison_fb_{1,i,k,j} + ison_fb_{1,i,k-1,j})}{2}\right]\Bigg), \qquad (40.119)
\end{aligned}
$$

where the overall minus sign accounts for the minus sign used for computing the discretized flux components, relative to the fluxes in the continuum. The pressure derivatives of ρ_θ and ρ_s are evaluated as

$$\partial_p \rho_\theta = -(\rho_o\, g)^{-1}\, \partial_z \rho_\theta \tag{40.120}$$

$$\partial_p \rho_s = -(\rho_o\, g)^{-1}\, \partial_z \rho_s. \tag{40.121}$$

Halobaricity is diagnosed from thermobaricity as

$$halob_{i,k,j} = -\left(\frac{(\rho_\theta)_{i,k,j}\, (\rho_{sp})_{i,k,j}}{(\rho_s)_{i,k,j}\, (\rho_{\theta p})_{i,k,j}} \right) thermo_{i,k,j}. \tag{40.122}$$

40.7.3 Output

Output in 32 bit IEEE unformatted data is not an option for this diagnostic. The three dimensional fields *advrho*, *diffdvert*, *diffhorz*, *diffgmskew*, *diffbihskew*, *cabbel*, *thermob*, *halob* are output only in NetCDF format to file *ptlrho_terms.nc*. The interval between output is specified by namelist variable *ptlrhoint* in units of days. Refer to Section 14.4 for information on namelist variables.

40.8 matrix_sections

The purpose of option *matrix_sections* is mainly for debugging. It is useful when trying to look at numbers as the model integrates. Various quantities are printed in matrix form as a function of longitude x and depth z for specific latitudes. The latitudes and ranges of longitudes and depths for limiting the printout are input through namelist. Refer to Section 14.4 for information on namelist variables. The output from this diagnostic is written as ascii to the model *printout* file and there is no NetCDF capability. The interval between output is specified by variable *prxzint*.

40.9 meridional_overturning

This section resulted from the combined efforts of Stephen Griffies and Young-Gyu Park (*ygp@gfdl.gov*).

The large-scale circulation in numerical models is often abbreviated in terms of its meridional-vertical plane circulation. Under certain assumptions, this circulation can be rendered by an *overturning streamfunction*. The streamfunction is constructed by integrating over the zonal direction between either two fixed meridional boundaries, on which the zonal velocity is assumed to vanish, or over a zonally periodic domain such as the World Ocean. Although useful, this abbreviated picture is potentially misleading due to the integration over sometimes important details of longitudinal variation. To alleviate some of the misleading characters, it is often useful to employ various vertical coordinates to construct different versions of the streamfunction. The purpose of this section is to provide a general formulation of the overturning streamfunction, and then to discuss the manners of computing this streamfunction in MOM.

40.9.1 Thickness equation

For a stably stratified fluid, it is possible to introduce a general monotonic vertical coordinate

$$s = s(\lambda, \phi, z, t). \tag{40.123}$$

Conservation of volume (incompressible fluid) for a fluid parcel implies

$$\frac{D}{Dt}\,(a^2 \cos\phi\, d\lambda\, d\phi\, dz) = \frac{D}{Dt}\,(a^2 \cos\phi\, d\lambda\, d\phi\, z_s\, ds) = 0, \tag{40.124}$$

where D/Dt is the material time derivative. This relation leads to the *thickness equation*

$$\partial_t z_s + \nabla_s \cdot (\vec{u}_h\, z_s) + \partial_s(z_s\, \dot{s}) = 0, \tag{40.125}$$

where ∇_s is the lateral gradient operator taken with the generalized vertical coordinate s held fixed, and the dot represents a Lagrangian time derivative. $\vec{u}_h$ is the horizontal velocity vector

$$\vec{u}_h = (u, v) = (\dot{x}, \dot{y}), \tag{40.126}$$

whereas $\dot{s}$ is the component of the velocity in the generalized vertical direction. In the direction perpendicular to the geopotential, the vertical velocity is given by

$$w = \dot{z} = (\partial_t + \vec{u}_h \cdot \nabla_s + \dot{s}\, \partial_s)\, z. \tag{40.127}$$

In this expression, z_t is the time tendency for the depth of a constant s surface.

$$\nabla_s z = -\left(\frac{\nabla_h s}{s_z}\right) \tag{40.128}$$

is the horizontal slope vector of constant s surfaces, where $\nabla_h s$ is the horizontal gradient of s taken with depth z held fixed, and s_z is the vertical gradient of the s surfaces. In general, if there is zero flow across the s surfaces, then $\dot{s} = 0$. Mathematically, $z_s = (s_z)^{-1}$ is the Jacobian of transformation between the (x, y, z, t) and the (x, y, s, t) coordinate systems. Often z_s is called the *specific thickness*. For a smooth and monotonic vertical coordinate, z_s is single signed and does not vanish. A trivial example for a vertical coordinate is $s = z$, which leads to $z_s = 1$, $\dot{s} = \dot{z} = w$, and the thickness equation reduces to $\nabla \cdot \vec{u} = 0$, which is the continuity equation used in MOM. In the following, it will prove useful to write the thickness equation in one of the two following equivalent forms:

$$\partial_s(z_t + z_s\, \dot{s}) + \nabla_s \cdot (\vec{u}_h\, z_s) = 0, \tag{40.129}$$

or, upon insertion of the vertical velocity w from equation (40.127),

$$\partial_s(w - \vec{u}_h \cdot \nabla_s z) + \nabla_s \cdot (\vec{u}_h\, z_s) = 0. \tag{40.130}$$

40.9.2 Zonally integrated circulation and its streamfunction

Now consider the zonally integrated water transport

$$\mathcal{V}(\phi, s, t) = a \cos\phi \int d\lambda\, z_s\, v \tag{40.131}$$

$$\mathcal{W}(\phi, s, t) = a \cos\phi \int d\lambda\, (z_t + z_s\, \dot{s}) = a \cos\phi \int d\lambda\, (w - \vec{u}_h \cdot \nabla_s z). \tag{40.132}$$

In these expressions, the zonal integration is taken with *both* the latitude ϕ and generalized vertical coordinate s held fixed. The component $\mathcal{V}$ represents the zonally integrated water transport moving in the meridional direction. The weighting with the specific thickness can best be understood in a finite difference context, for which a weighting $z_s\,\delta s$ implies that $\mathcal{V}$ is the meridional transport (in units of $length^2/time$) between the surfaces s and $s+\delta s$. In the continuum, the units of $\mathcal{V}$ are $length^2/time \times length/s$. The component $\mathcal{W}$ represents the zonally integrated time tendency for the changes in height of a constant s surface, plus the specific thickness weighted velocity of water moving across a constant s surface. The units for $\mathcal{W}$ are $length^2/time$.

In the following, the zonal integration will be taken either within a particular basin enclosed by meridional boundaries on which the zonal velocity is assumed to vanish, or for a zonally symmetric domain such as the World Ocean for which the zonal velocity is periodic. As such, the two components of the zonally integrated transport are related by

$$\begin{aligned}
\partial_s \mathcal{W} &= a\cos\phi \int d\lambda\, \partial_s\,(z_t + z_s\,\dot{s}) \\
&= -a\cos\phi \int d\lambda\, \nabla_s \cdot (\vec{u}_h\, z_s) \\
&= -a\cos\phi \int d\lambda\, \frac{1}{a\cos\phi}(z_s\, v\cos\phi)_\phi \\
&= -a^{-1}\,\partial_\phi\,\mathcal{V}, \qquad (40.133)
\end{aligned}$$

where the thickness equation (40.129) has been used. The assumptions regarding the zonal velocity allow for the elimination of the $\int d\lambda\,(z_s\,u)_\lambda$ term. Hence, the two dimensional zonally integrated circulation is divergent-free.

The assumption of zero zonal velocity on the meridional boundaries prompts some further comments. In closed basins, a no-normal flow boundary condition does not imply a zero zonal velocity next to the meridional water-land boundaries. In MOM, there is an unambiguous distinction made between bottom and side boundaries, which is allowed by the use of step-topography. Bottom boundaries generally employ the no-normal flow condition relevant for an inviscid fluid, with an option available to add a bottom boundary layer. Side boundaries, however, always use the no-slip condition which means that all components of the velocity vanish next to side boundaries. Therefore, the elimination of the $\int d\lambda\,(z_s\,u)_\lambda$ term is always valid in MOM.

Since the two components of the zonally integrated water transport satisfy the zero divergence condition

$$\partial_s\,\mathcal{W} + a^{-1}\,\partial_\phi\,\mathcal{V} = 0, \qquad (40.134)$$

the zonally integrated transport can be determined by an overturning streamfunction

$$\begin{aligned}
-\partial_s\psi &= \mathcal{V} \qquad (40.135)\\
a^{-1}\,\partial_\phi\psi &= \mathcal{W}. \qquad (40.136)
\end{aligned}$$

It is important to remember that the latitudinal derivative ∂_ϕ is taken at fixed generalized vertical coordinate s, and for the generalized vertical derivative ∂_s, the latitude is held fixed. In words, the vertical convergence of the streamfunction at constant latitude gives the zonally integrated transport across that latitude. Likewise, the latitudinal divergence on constant s surfaces yields the zonally integrated transport across the s surface plus the zonally integrated

time tendency for the height of the surface. It is useful to state these results another way, in terms of finite differences. The difference between the streamfunction at two points at the same latitude, yet on different s surfaces, represents the total meridional transport of fluid (in units of volume/time) crossing this latitude in between the two s surfaces. Likewise, the difference between the value of the streamfunction at two points at the same s surface, but at different latitudes, equals the total transport of fluid (in units of volume/time) crossing this particular s surface, plus a contribution due to the time tendency for the height of the s surface. An alternative interpretation of the latitudinal divergence on constant s surfaces follows from the equivalence $z_t + z_s \dot{s} = w - \vec{u}_h \cdot \nabla_s z$.

Integration of the equation (40.135) at a fixed latitude yields

$$\psi(\phi, s, t) = \psi(\phi, s_o, t) - \int_{s_o}^{s} ds' \; \mathcal{V}(\phi, s', t), \tag{40.137}$$

where s_o is a reference value for the generalized vertical coordinate. Likewise, a latitudinal integral of equation (40.136) along a fixed surface s yields

$$\psi(\phi, s, t) = \psi(\phi_o, s, t) + a \int_{\phi_o}^{\phi} d\phi' \; \mathcal{W}(\phi', s, t), \tag{40.138}$$

where ϕ_o is a reference latitude. It is now possible to develop two equivalent expressions for the overturning streamfunction. The first is found through substituting the expression (40.137) into (40.138)

$$\psi(\phi, s, t) = \psi(\phi_o, s_o, t) - \int_{s_o}^{s} ds' \; \mathcal{V}(\phi_o, s', t) + a \int_{\phi_o}^{\phi} d\phi' \; \mathcal{W}(\phi', s, t). \tag{40.139}$$

The second expression is determined through substituting equation (40.138) into (40.137)

$$\psi(\phi, s, t) = \psi(\phi_o, s_o, t) - \int_{s_o}^{s} ds' \; \mathcal{V}(\phi, s', t) + a \int_{\phi_o}^{\phi} d\phi' \; \mathcal{W}(\phi', s_o, t). \tag{40.140}$$

Note that the streamfunction has dimensions *volume/time*, and so it represents the volume transport of water in the (ϕ, s) plane.

40.9.3 Overturning streamfunction

The question now is how to choose the reference values ϕ_o and s_o to simplify the computation of the overturning streamfunction. In general, evaluation of the vertical transport term is more difficult than the meridional transport term. One is therefore motivated to focus on equation (40.140) when computing the streamfunction, rather than equation (40.139). For choosing the reference value s_o of the generalized vertical coordinate, it is useful to consider the water budget and how it closes. In the rigid lid, the water budget is closed within the ocean domain. As such, so long as the value of s_o corresponds to a value anywhere completely outside the ocean domain, the vertical transport term $\int_{\phi_o}^{\phi} d\phi' \; \mathcal{W}(\phi', s_o, t)$ will vanish. For the free surface, however, the possibility of surface water fluxes allows for an open water budget above the ocean surface. Since there is no attempt here to account for water cycling through the rock beneath the ocean, one can assume all water transport in rock vanishes. Hence, by taking s_o to be some value completely beneath the ocean bottom, the vertical transport term can again be dropped with the free surface. As a consequence, a general expression for the overturning

streamfunction, valid for both the free surface and rigid lid, is given just by the meridional transport term

$$\psi(\phi, s, t) = -\int_{s_o}^{s} ds' \; \mathcal{V}(\phi, s', t) = -a \cos\phi \int_{s_o}^{s} ds' \int d\lambda \; z_s \, v. \tag{40.141}$$

In practice, of course, it is not necessary to evaluate the integral anyplace beneath the ocean bottom, since the water velocity vanishes there. On the bottom, the generalized vertical coordinate takes on the non-constant value $s(\lambda, \phi, z = -H, t)$. As such, when integrating just to the ocean bottom, it is necessary to perform the vertical integral first, and then the zonal integral

$$\psi(\phi, s, t) = -a \cos\phi \int d\lambda \int_{s(\lambda,\phi,z=-H,t)}^{s} ds' \; z_s \, v = -a \cos\phi \int d\lambda \int_{-H(\lambda,\phi)}^{z(\lambda,\phi,s,t)} dz' \; v, \tag{40.142}$$

where $z_s \, ds = dz$ was used to reach the final expression. Note that $z(\lambda, \phi, s, t)$ is the depth of the smooth surface whose generalized vertical coordinate has the value s.

For the rigid lid, the expression (40.142) for the streamfunction can be brought to a more familiar form by noting that the volume of water passing northward across any latitude must balance the volume flowing southward. Therefore,

$$\int d\lambda \int_{-H(\lambda,\phi)}^{0} dz' \; v(\lambda, \phi, z', t) = 0. \tag{40.143}$$

The result leads to the familiar expression for the rigid lid overturning streamfunction

$$\psi_{rl}(\phi, s, t) = a \cos\phi \int d\lambda \int_{s}^{s(\lambda,\phi,z=0,t)} ds' \; (z_{s'} \, v) = a \cos\phi \int d\lambda \int_{z(\lambda,\phi,s,t)}^{0} dz' \; v. \tag{40.144}$$

Again, this expression is valid only for the rigid lid, since the balance given by equation (40.143) is only valid in this case. For the free surface, the more general expression (40.142) must be used. The differences between these two expressions will be further discussed in the next section.

In the ocean interior, the no-normal flow condition implies that the overturning streamfunction is a constant along the side and bottom boundaries (for the relevant arguments, see Section 6.5 in which the boundary conditions for the barotropic streamfunction are derived). For the rigid lid, the absence of fresh water input to the ocean surface also implies that its overturning streamfunction is a constant at the ocean surface. The choice $\psi(\phi_o, s_o, t) = 0$ means that the rid lid overturning streamfunction is zero along all the boundaries. For the free surface, however, the overturning streamfunction need not be a constant on the ocean surface, due to the presence of surface water fluxes, whereas it remains zero on the sides/bottom just as for the rigid lid.

40.9.4 Comments on the free surface overturning streamfunction

Consider now an alternative method for computing the free surface streamfunction. It is less concise than equation (40.142), and requires some approximations whereas equation (40.142) is exact. However, it has the virtue of exposing the fresh water contribution to the streamfunction. It is discussed here for pedagogical reasons.

The central difference from the previous result is that the reference value of s_o is taken someplace above the ocean surface rather than beneath the bottom. Now care must be taken

for how to handle the upper surface boundary condition where the fresh water enters the ocean. To do so, some approximations will be made. First, assume that the region for which the water flux enters the ocean is a completely mixed region, and use a z-coordinate as the generalized vertical coordinate for which $z_t = 0$ and $z_s \dot{s} = w$. Next, recall from Section 4.3.2 the surface boundary condition for the free surface

$$\eta_t = w + q_w - \vec{u}_h \cdot \nabla_h \eta \qquad z = \eta, \tag{40.145}$$

where $q_w(\lambda, \phi, t) > 0$ means that water enters the ocean through the free surface. In order to account for the large-scale effects of fresh water in the overturning streamfunction, it is sufficient to assume that $\eta = 0$, and in turn to employ

$$w \approx -q_w \qquad z = \eta. \tag{40.146}$$

This approximate expression is the "Natural Boundary Condition" of Huang (1993). Taking $s_o = z_o = 0$, this assumption brings the zonally integrated vertical transport at the ocean surface to the form

$$\mathcal{W}(\phi, z = 0, t) = -a \cos\phi \int d\lambda \, q_w. \tag{40.147}$$

Note that a net zonally integrated fresh water input leads to transport into the ocean, and so $\mathcal{W} < 0$. The corresponding overturning streamfunction then takes the form

$$\psi(\phi, s, t) = a \cos\phi \int d\lambda \left(\int_{z(\lambda,\phi,s,t)}^{z=0} dz' \, v - \int_{\phi_o}^{\phi} a \, d\phi' \, q_w \right). \tag{40.148}$$

Setting the fresh water term to zero in equation (40.148) recovers the rigid lid result (40.144). Note that the small transport in the region between $z = \eta$ and $z = 0$ has been neglected. Using

$$\int_{z(\lambda,\phi,s,t)}^{0} dz' \, v \;=\; \int_{-H}^{0} dz' \, v - \int_{-H}^{z(\lambda,\phi,s,t)} dz' \, v \tag{40.149}$$

in equation (40.148), and recalling the general expression (40.142) for the overturning streamfunction, results in the identity

$$\int d\lambda \int_{-H(\lambda,\phi)}^{0} dz' \, v = \int_{\phi_o}^{\phi} a \, d\phi' \int d\lambda \, q_w. \tag{40.150}$$

Taking the reference latitude $\phi_o = \phi_{south}$ leads to a simple interpretation for this result. For example, if there is a net water input through the surface in the region to the south of a particular latitude, then in the steady state, there is a net northward meridional transport in the ocean (see Figure 40.1).

40.9.5 Overturning streamfunction in the (ϕ, z) plane

For those running z-level models such as MOM, the simplest choice for vertical coordinate is $s = z$, where z is the time independent height of the constant depth surfaces. Derivatives of the streamfunction yield the zonally integrated Eulerian meridional and vertical transport

$$-\partial_z \psi(\phi, z, t) \;=\; a \cos\phi \int d\lambda \, v(\lambda, \phi, z, t) \tag{40.151}$$

$$a^{-1} \, \partial_\phi \psi(\phi, z, t) \;=\; a \cos\phi \int d\lambda \, w(\lambda, \phi, z, t), \tag{40.152}$$

and the streamfunction from equation (40.142) is given by

$$\psi(\phi, z, t) \;=\; -a \cos\phi \int d\lambda \int_{-H(\phi,\lambda)}^{z} dz' \, v. \tag{40.153}$$

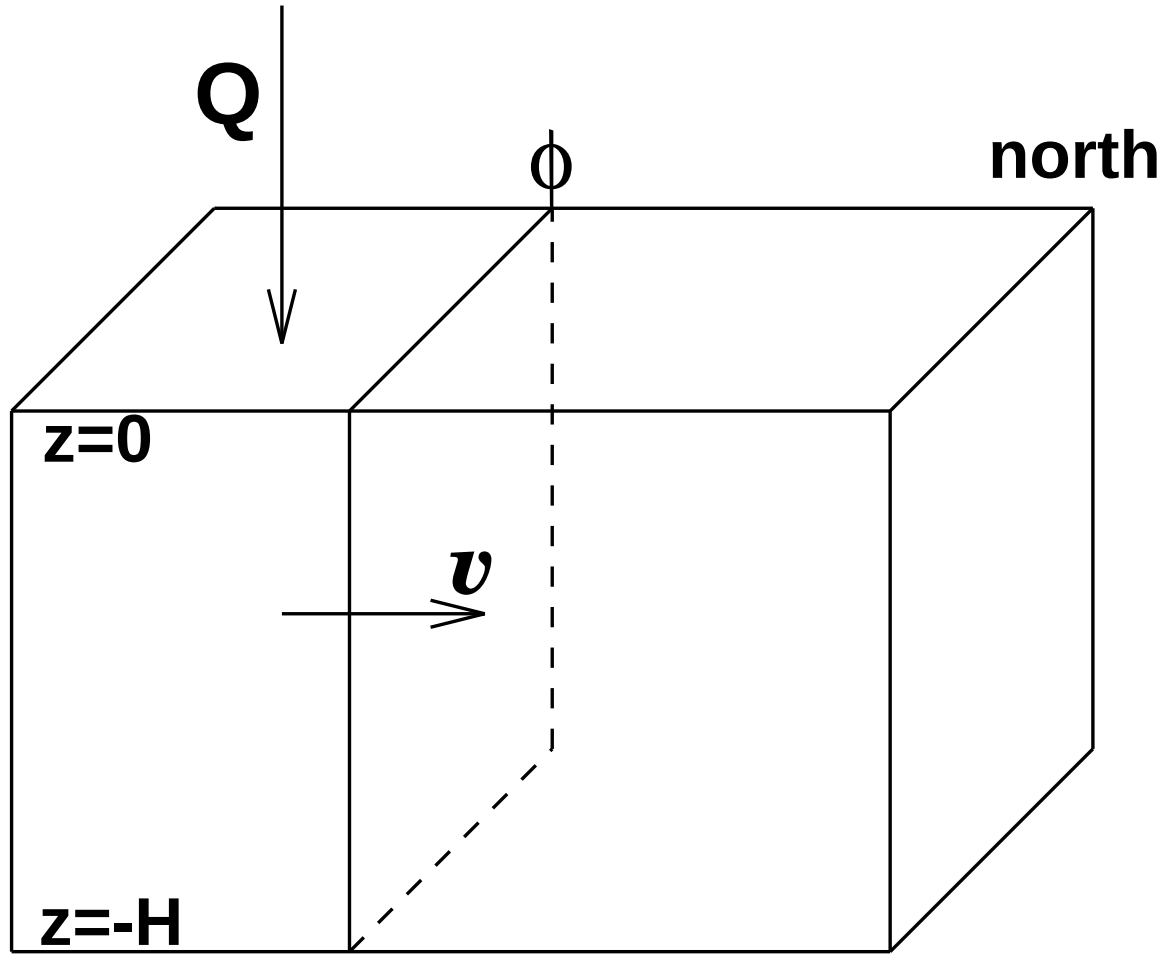


Figure 40.1: Sketch of an ocean domain in which there is a net input of surface fresh water through the free surface in the region to the south of a particular latitude. In the steady state, the result is a net northward meridional transport of water in the interior of the ocean at this latitude.

40.9.6 Overturning streamfunction in the (ϕ, θ) plane

Another choice for the vertical coordinate is $s = \theta$, with θ the potential temperature. This choice was discussed by Bryan and Sarmiento (1985). Taking $s = \theta$ makes sense only when θ surfaces are monotonic. Assuming such, the overturning streamfunction from equation (40.142) is given by

$$\psi(\phi, \theta, t) = -a \cos\phi \int d\lambda \int_{-H(\lambda,\phi)}^{z(\lambda,\phi,\theta,t)} dz' \, v(\lambda, \phi, z', t). \tag{40.154}$$

Derivatives of the streamfunction yield

$$-\partial_\theta \psi = a \cos\phi \int d\lambda \, z_\theta \, v \tag{40.155}$$

$$a^{-1} \, \partial_\phi \psi = a \cos\phi \int d\lambda \, (z_t + z_\theta \, \dot{\theta}), \tag{40.156}$$

where z_t is the time tendency for the height of a θ surface. Note that $z_\theta > 0$ for a stably stratified fluid dominated by temperature. For a fluid with θ the only active tracer, $\dot{\theta} = 0$ results when the flow is adiabatic. In this case, and assuming a steady state, the meridional streamfunction ψ is dependent only on the potential temperature: $\partial_\phi \psi = 0$.

40.9.7 Overturning streamfunction in the $(\phi, \rho^{(0)})$ plane

Another choice is to set $s = \rho^{(0)}$, where $\rho^{(0)}$ is the potential density using the ocean surface as the reference pressure. This choice, as with the choice $s = \theta$, is adequate only when the ocean stratification is monotonic in $\rho^{(0)}$. The discussion with $s = \theta$ follows through here, with the material derivative now given by

$$\frac{D\rho^{(0)}}{Dt} = \frac{\partial \rho^{(0)}}{\partial \theta}\,\dot{\theta} + \frac{\partial \rho^{(0)}}{\partial S}\,\dot{S}, \tag{40.157}$$

with S the salinity.

40.9.8 Overturning streamfunction in the $(\phi, \rho^{(p)})$ plane

Another choice is to set $s = \rho^{(p)}$, where $\rho^{(p)}$ is the density referenced to a particular pressure level. The only difference with the $s = \rho^{(0)}$ discussion is that the reference pressure for evaluating the thermal and saline expansion coefficients is altered.

40.9.9 Overturning streamfunction in the $(\phi, \rho^{neutral})$ plane

Another vertical coordinate which is quite useful is the neutral density coordinate of Jackett and McDougall (1997). The paper by Hirst, McDougall, and Jackett (1996) illustrates the utility of this coordinate for globally diagnosing mechanisms of water mass transformation. The $(\phi, \rho^{neutral})$ plane streamfunction requires the software of Jackett and McDougall (1997), which is not provided with MOM.

40.9.10 Discrete vertical-meridional streamfunction

For the (ϕ, z) plane, a straightforward discretization of equation (40.142)

$$\psi(\phi, z, t) = -a \cos\phi \int d\lambda \int_{-H(\lambda,\phi)}^{z} dz'\, v, \tag{40.158}$$

leads to the discretized overturning streamfunction valid for either the rigid lid or free surface

$$vmsf_{jrow,k} = -\cos\phi^{U}_{jrow} \sum_{i=2}^{imt-1} \sum_{m=k}^{kmt(i,jrow)} dxu_i\, dzt_m\, u_{i,m,j,2,\tau}. \tag{40.159}$$

Note that the switch in the limits on the sum, relative to the integral, accounts for k increasing downward, whereas z increases upwards.

The vertical integration procedes from the ocean bottom, which is the bottom of the bottom-most velocity cell, upwards to the top of the particular grid cell of interest. As such, the vertical placement of the meridional streamfunction is at the interface of the cells in which the velocity is located. These points are labeled by zw_k in the model, where $k = 1, km$. To aid in visualization, the top of the ocean at $z = 0$ is plotted as well. For the rigid lid, the streamfunction value is zero at $z = 0$; for the free surface, it represents the zonally averaged fresh water flux crossing the ocean-atmosphere interface.

40.9.11 Discrete density-meridional streamfunction

The streamfunction computed with potential density as the vertical coordinate requires a bit more work. The most important part of the process is how to partition the potential density layers. In general, how the layers are placed will largely determine the quality of the diagnosed overturning streamfunction. Currently, the density range is divided into *km* layers, since this is the maximum number of horizontally uniform density layers which can be resolved in a level model. Using fewer density layers will generally produce a smoother streamfunction, but the value of the streamfunction is typically reduced in magnitude. More layers typically produces a noisy solution, and will make the streamfunction have larger values for its extrema. More sophisticated schemes are possible, in which various forms of interpolation are applied. These have not been tried. If someone has a better scheme, please feel free to suggest its use. Currently, the simplest algorithm was chosen, in which zero interpolation is used. Furthermore, since each model solution likely will require its own particular partitioning, it is currently left to the researcher to determine the density layers. The placement of the density layers occurs inside the routine *diag.F*. Look for the USER INPUT section within the ifdef option *meridional_overturning*.

Given a partitioning of the density coordinate σ_m, for $m = 1, km$, the algorithm for computing the streamfunction is the following:

$$sigmsf_{jrow,m} = -\cos\phi^U_{jrow} \sum_{i=2}^{imt-1} \sum_{k=1}^{kmt(i,jrow)} dxu_i \, dzt_m \, \mathcal{H}(\sigma_{i,k,j} - \sigma_m) \, u_{i,m,j,2,\tau}, \tag{40.160}$$

where $\mathcal{H}(x)$ is a Heaviside step function (equal to zero for $x < 0$ and unity for $x > 0$.

As coded in MOM, there are two potential densities which are used for defining the streamfunction: potential density referenced to the surface, and potential density referenced to 2000m. The framework for using these potential densities can easily be extended.

40.9.12 Option *merid_by_basin*

For global models, it is often useful to plot the overturning streamfunction within particular basins in addition to the global ocean. With option *merid_by_basin*, the streamfunction can be computed for different ocean domains using masks set up by the researcher. See the code for details of how to set up the masks. Also note that MOM generically computes an ASCII file *kmt.dta* which is a useful place to start when constructing the masks. An objective way to construct masks is to use a "fill algorithm" such as that used in MOM for constructing the land-sea boundaries.

40.9.13 Output

The output for the vertical-meridional streamfunction may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *overturn.yyyyyy.mm.dd.dta*. If option *netcdf* or *meridional_overturning_netcdf* is enabled, the vertical-meridional streamfunction is written in NetCDF format to file *overturn.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. If the sub-option *meridional_overturning_potential* is also enabled, the sigma-meridional streamfunctions are written to files *overturn.sigAmsf.yyyyyy.mm.dd.dta.nc* for σ_0 and *overturn.sigBmsf.yyyyyy.mm.dd.dta.nc* for σ_{2000}. The interval between output for each of these

streamfunctions is specified by the single variable *vmsfint*, and the control is specified by variable *iovmsf*.

40.10 meridional_tracer_budget

What are the dominant meridional balances (if any) in the model and how do they depend on time? Option *meridional_tracer_budget* contracts the tracer equation into a one dimensional equation in latitude by averaging over longitude and depth cells. Using operators described in Section 22.8.7, this yields

$$\frac{1}{Vol_j^T}\sum_{i=2}^{imt-1}\sum_{k=1}^{km}\Delta^T_{i,k,j} \cdot \Big[\delta_t(T_{i,k,j,n,\tau}) + ADV_Tx_{i,k,j} + ADV_Ty_{i,k,j} + ADV_Tz_{i,k,j} = DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + DIFF_Tz_{i,k,j} + source_{i,k,j}\Big] \tag{40.161}$$

where n is the tracer and the relation between j and $jrow$ is as described in Section 14.2. The volume element and total volume as a function of latitude are given by

$$\Delta^T_{i,k,j} = tmask_{i,k,j} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{40.162}$$

$$Vol_j^T = \sum_{k=1}^{km}\sum_{i=2}^{imt-1}\Delta^T_{i,k,j} \tag{40.163}$$

The quantity $tmask_{i,k,j}$ is 1 for ocean cells and 0 for land cells. Integrating and applying boundary conditions in depth and longitude to Equation (40.161) yields

$$\sum_{i,k}\delta_t(T_{i,k,j,n,\tau})\Delta^T_{i,k,j} + \sum_{i,k}(ADV_Ty_{i,k,j})\Delta^T_{i,k,j} = \sum_{i,k}(stf_{i,j,n})\Delta^T_{i,k,j} + \sum_{i,k}(DIFF_Ty_{i,k,j})\Delta^T_{i,k,j} + \sum_{i,k}(source_{i,k,j})\Delta^T_{i,k,j} \tag{40.164}$$

where $\sum_{i,k}$ is shorthand for summing over all cells in longitude and depth. Each term in Equation (40.164) is then averaged in time to produce stable estimates which can indicate the dominant meridional balances as a function of time. The meridional tracer equation becomes

$$\frac{1}{L}\sum_{i,k,\ell=1}^{L}\Big[\delta_t(T_{i,k,j,n,\tau}) + ADV_Ty_{i,k,j} = stfi,j,n + DIFF_Ty_{i,k,j} + source_{i,k,j}\Big] \tag{40.165}$$

where ℓ is the time step counter and L is the number of time steps in the averaging period described below. The i,k in the sum indicates a sum over all longitude and depth cells. The individual terms in Equation (40.165) are given as

$$tstor_{jrow,n} = \frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol_j^T}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\frac{t_{i,k,j,n,\tau+1}-t_{i,k,j,n,\tau-1}}{2\Delta\tau}\Delta_{i,k,j}^T\Big] \quad (40.166)$$

$$tdiv_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol_j^T}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}ADV_Ty_{i,k,j}\cdot\Delta_{i,k,j}^T\Big] \quad (40.167)$$

$$tflux_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Area_j^T}\sum_{i=2}^{imt-1}stf_{i,j,n}\cdot A_{i,j}^T\Big] \quad (40.168)$$

$$tsorc_{jrow,n} = \frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol_j^T}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}source_{i,k,j}\cdot\Delta_{i,k,j}^T\Big] \quad (40.169)$$

$$tdif_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol_j^T}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}DIFF_Ty_{i,k,j}\cdot\Delta_{i,k,j}^T\Big] \quad (40.170)$$

where the area element and total area of a latitude are given by

$$A_{i,j}^T = tmask_{i,1,j}\cdot dxt_i\cdot\cos\phi_{jrow}^T\cdot dyt_{jrow} \quad (40.171)$$

$$Area_j^T = \sum_{i=2}^{imt-1}A_{i,j}^T \quad (40.172)$$

$$(40.173)$$

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *tracer_bud.yyyyyy.mm.dd.dta*. If option *netcdf* or *meridional_tracer_budget_netcdf* is enabled, data is written in NetCDF format to file *tracer_bud.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *tmbint* and the control is specified by variable *iotmb*. The averaging period *tmbper* is typically set equal to the interval but may be specified shorter. How about producing monthly averages when months vary in length? Enabling option *monthly_averages* will over-ride values in *tmbper* and *tmbint* and make it so. However, this option is global and applies to all time averaged diagnostics that are enabled. Refer to Section 40.11 for more details. Variables *tmbper* and *tmbint* are input through namelist. Refer to Section 14.4 for information on namelist variables. Note further that this dataset may be contracted in latitude and region space to yield a zero dimensional model which describes the total heat content of one tremendously gigantic cell as a function of time.

$$< tstor_{jrow,n} >^{\phi} = \frac{1}{Area^T}\sum_{jrow=2}^{jmt-1}\Big[tflux_{jrow,n}\Big]\Delta_j^T \quad (40.174)$$

where the area element is

$$Area^T = \sum_{jrow=2}^{jmt-1}\Delta_j^T \quad (40.175)$$

However, this can only be done if the averaging period equals the interval for writing the output. But, when the averaging period equals the sampling interval this diagnostic is a significant time burner. For most cases, it may be adequate to specify a short averaging period at the end of the sampling interval (e.g. in the limit, a one time step average at the end of every sampling interval.). The terms in the above analysis may be further decomposed into regions based on mask $msktmb_{i,jrow}$ along the lines of Section 39.5. However, $msktmb_{i,jrow}$ =1 and it is left to the researcher to define other regions if desired. Note that the regional mask $msktmb_{i,jrow}$ is also written to file *tracer_bud.yyyyyy.mm.dd.dta* when the initialization boolean *itmb* is true. It should be set true on the first run but false thereafter.

40.11 monthly_averages

It is useful to produce monthly averaged datasets for diagnostics which compute time averaged quantities. Specifying a periodic "interval" for output and an averaging "period" as discussed in Section 39.4 is not helpful when months vary in length. However, enabling option *monthly_averages* will over-ride values specified by "interval" and "period" and produce monthly averaged datasets. However, this option is global and applies to all time averaged diagnostics that are enabled. The only caveat is that if the model is started or stopped at any time other than a month boundary, the first or last average will not be over the full month.

40.12 save_convection

When convection is being computed explicitly (option *implicitvmix* is <u>not</u> enabled), the instantaneous value of the component of the time rate of change of temperature due to explicit convection can be saved three dimensionally. If cell $T_{i,k,j}$is a land cell, its value is set to a flag value $= 10^{-20}$ to denote land. If no convection has taken place in cell $T_{i,k,j}$, the convection for that cell is zero. Otherwise

$$convect_{i,k,j} = \frac{t^{after\ convection}_{i,k,j,1,\tau+1} - t^{before\ convection}_{i,k,j,1,\tau+1}}{2\Delta\tau} \tag{40.176}$$

The output from this diagnostic is written only as 32 bit IEEE unformatted data to file *cvct.yyyyyy.mm.dd.dta*. If option *netcdf* or *save_convection_netcdf* is enabled, data is written in NetCDF format to file *cvct.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *exconvint* and there is no control variable since only the unformatted data is written.

40.13 save_mixing_coeff

Much of the physics in MOM is distilled into mixing coefficients. Mixing coefficients can be computed in a variety of ways depending on the combinations of horizontal and vertical subgrid scale mixing parameterization options which have been enabled. In some cases, the mixing coefficients are not explicitly computed. However, the flux across faces of cells is always computed. Option *save_mixing_coeff* estimates mixing coefficients from flux across cell faces. For momentum, the coefficients on east, north, and bottom faces of U cells is estimated by

$$ce_{i,k,j,1} = \frac{diff_fe_{i,k,j}}{\frac{1}{\cos\phi^{u}_{jrow}}\delta_{\lambda}(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.177}$$

$$cn_{i,k,j,1} = \frac{diff_fn_{i,k,j}}{\delta_{\phi}(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.178}$$

$$cb_{i,k,j,1} = \frac{diff_fb_{i,k,j}}{\delta_{z}(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.179}$$

where $\epsilon = 10^{-20}$ to keep from dividing by zero where no gradient in velocity exists. For tracers, the coefficients on east, north, and bottom faces of T cells is estimated by

$$ce_{i,k,j,2} = \frac{diff_fe_{i,k,j}}{\frac{1}{\cos\phi^{u}_{jrow}}\delta_{\lambda}(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.180}$$

$$cn_{i,k,j,2} = \frac{diff_fn_{i,k,j}}{\delta_{\phi}(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.181}$$

$$cb_{i,k,j,2} = \frac{diff_fb_{i,k,j}}{\delta_{z}(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{40.182}$$

where ϵ plays the same role as for velocities. If option *isoneutralmix* is enabled, the $K11_{i,k,j}$, $K22_{i,k,j}$, $K33_{i,k,j}$ elements of the isoneutral mixing tensor along with the suitably averaged mixing coefficients $A^{ez}_{i,k,j}$, $A^{nz}_{i,k,j}$, $A^{bx}_{i,k,j}$, and $A^{by}_{i,k,j}$ are additionally output.

The output from this diagnostic is written as 32 bit IEEE unformatted data to file *cmix.yyyyyy.mm.dd.dta*. If option *netcdf* or *save_mixing_coeff_netcdf* is enabled, data is written in NetCDF format to file *cmix.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *cmixint* and there is no control variable since only the unformatted data is written.

40.14 show_external_mode

Option *show_external_mode* saves instantaneous values (at time level $\tau + 1$) of either the stream function, prognostic surface pressure, or implicit free surface depending on which option is enabled. Stream function is in units of cm^3/sec, surface pressure and implicit free surface is in units of $gram/cm/sec^2$. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *psi.yyyyyy.mm.dd.dta*, *surf_press.yyyyyy.mm.dd.dta*, or *ifree_surf.yyyyyy.mm.dd.dta*. If option *netcdf* or *show_external_mode_netcdf* is enabled, data is written in NetCDF format to file *psi.yyyyyy.mm.dd.dta.nc*, *surf_press.yyyyyy.mm.dd.dta.nc*, or *ifree_surf.yyyyyy* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *extint* and the control is specified by variable *ioext*.

40.15 show_zonal_mean_of_sbc

Option *show_zonal_mean_of_sbc* saves instantaneous zonal means of all surface boundary conditions as a function of latitude. This is useful to verify that the surface boundary conditions are reasonable, at least in the zonal mean sense.

$$zmsm_{jrow,n} = \frac{1}{Vol_j^U} \sum_{i=2}^{imt-1} u_{i,1,j,n,\tau} \cdot \Delta_{i,j}^U \tag{40.183}$$

$$zmsmf_{jrow,n} = \frac{1}{Vol_j^U} \sum_{i=2}^{imt-1} smf_{i,j,n} \cdot \Delta_{i,j}^U \tag{40.184}$$

$$zmst_{jrow,n} = \frac{1}{Vol_j^T} \sum_{i=2}^{imt-1} t_{i,1,j,n,\tau} \cdot \Delta_{i,j}^T \tag{40.185}$$

$$zmstf_{jrow,n} = \frac{1}{Vol_j^T} \sum_{i=2}^{imt-1} stf_{i,j,n} \cdot \Delta_{i,j}^T \tag{40.186}$$

where the volume elements and total volumes on U cells and T cells are

$$\Delta_{i,j}^U = umask_{i,1,j} \cdot dxu_i \cdot \cos\phi_{jrow}^U \cdot dyu_{jrow} \cdot dzt_k \tag{40.187}$$

$$\Delta_{i,j}^T = tmask_{i,1,j} \cdot dxt_i \cdot \cos\phi_{jrow}^T \cdot dyt_{jrow} \cdot dzt_k \tag{40.188}$$

$$Vol_j^U = \sum_{i=2}^{imt-1} \Delta_{i,j}^U \tag{40.189}$$

$$Vol_j^T = \sum_{i=2}^{imt-1} \Delta_{i,j}^T \tag{40.190}$$

and the relation between j and $jrow$ is as described in Section 14.2. Surface heat flux is converted to $watts/m^2$, windstress is in $dyne/cm^2$ precip minus evaporation is in mm/day, velocity is in cm/sec, temperature is in $degC$ and salinity is in $ppt - 35.0$. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *zmean_sbc.yyyyyy.mm.dd.dta*. If option *netcdf* or *show_zonal_mean_of_sbc_netcdf* is enabled, data is written in NetCDF format to file *zmean_sbc.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *zmbcint* and the control is specified by variable *iozmbc*.

40.16 snapshots

Option *snapshots* saves <u>instantaneous</u> values of prognostic and associated variables. The variables are tracers $t_{i,k,j,n,\tau}$ for $n = 1, nt$, horizontal velocities $u_{i,k,j,n,\tau}$ for $n = 1, 2$, vertical velocity at the base of T cells $adv_vbt_{i,k,j}$, surface tracer flux $stf_{i,j,n}$ for $n = 1, nt$, surface momentum flux $smf_{i,j,n}$ for $n = 1, 2$, and the external mode which is given by either[1] $psi_{i,jrow,\tau}$ or $ps_{i,jrow,\tau}$.

[1]Depending on whether option *stream_function*, *rigid_lid_surface_pressure* or *implicit_free_surface* is enabled.

This output data may be restricted to certain contiguous latitude and depth ranges using variables input through namelist. Refer to Section 14.4 for information on namelist variables. Note that on time step = itt (which corresponds to $\tau + 1$), data is written from time level τ rather than $\tau + 1$ because the $\tau + 1$ external mode is unknown at the time when data is written. Therefore, $u_{i,k,j,n,\tau}$ is a total velocity containing both internal and external modes. The output from this diagnostic is written as 32 bit IEEE unformatted data to file *snapshots.yyyyyy.mm.dd.dta*. If option *netcdf* or *snapshots_netcdf* is enabled, data is written in NetCDF format to file *snapshots.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2.

It should be noted that for use with Ferret, file *snapshots.yyyyyy.mm.dd.dta.nc* has the x, y, and z axes defined such that "edges" of cells along any axis on the T grid coincide with grid point coordinates on the U grid and visa versa. This is important when performing operations like integrals from within Ferret. For instance, if the dimensions of cell $T_{i,k,j}$are dxt_i, dyt_{jrow}, and dzt_k, then the cell edges are defined as follows: eastern edge of cell $T_{i,k,j}$is at $xt_i_edges_i = xu_i$ and the western edge is at $xt_i_edges_{i-1} = xu_{i-1}$. The northern edge is at $yt_j_edges_{jrow} = yu_{jrow}$ and the southern edge is at $yt_j_edges_{jrow-1} = yu_{jrow-1}$ and the bottom edge is at $zt_k_edges_k = zw_k$ and the top edge is at $zt_k_edges_{k-1} = zw_{k-1}$. Therefore, the cell dimensions can be computed from

$$dxt_i = xt_i_edges_i - xt_i_edges_{i-1} \tag{40.191}$$
$$dyt_{jrow} = yt_j_edges_{jrow} - yt_j_edges_{jrow-1} \tag{40.192}$$
$$dzt_k = zt_k_edges_k - zt_k_edges_{k-1} \tag{40.193}$$

and the volume is then given by

$$T_vol_{i,k,jrow} = dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{40.194}$$

For cell $U_{i,k,j}$, the corresponding cell dimensions are

$$dxu_i = xu_i_edges_i - xu_i_edges_{i-1} \tag{40.195}$$
$$dyu_{jrow} = yu_j_edges_{jrow} - yu_j_edges_{jrow-1} \tag{40.196}$$
$$dzt_k = zt_k_edges_k - zt_k_edges_{k-1} \tag{40.197}$$

and the volume is

$$U_vol_{i,k,jrow} = dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{40.198}$$

The structure of this NetCDF file is very similar to the one from option *time_averages*. The interval between output is specified by variable *snapint*.

40.17 term_balances

Option *term_balances* constructs <u>instantaneous</u> spatial averages of all terms[2] in the prognostic equations over arbitrary regional volumes defined by horizontal and vertical region masks

[2]Plus a few extra ones.

as described in Section 39.5. Regional volumes may be set up as indicated by the test case example in file *setocn.F*. They may not overlap one another. This diagnostic is useful when trying to understand the dominant balances within regional volumes of the model domain. Be aware that aliasing may occur because results are not averaged in time. This diagnostic is relatively slow and can appreciably degrade the speed of MOM. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *term_bal.yyyyyy.mm.dd.dta*. If option *netcdf* or *term_balances_netcdf* is enabled, data is written in NetCDF format to file *term_bal.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *trmbint* and the control is specified by variable *iotrmb*. Note that the regional masks $mskhr_{i,jrow}$and $mskvr_k$are also written to file *term_bal.yyyyyy.mm.dd.dta* when the initialization boolean *itrmb* is true. It should be set true on the first run but false thereafter.

Partial sums are taken over all *(i,k,j)* within the domain to contract quantities into regional volumes given as

$$nreg = (mskvr_k - 1) \cdot nhreg + mskhr_{i,jrow} \tag{40.199}$$

where masks $mskvr_k$and $mskhr_{i,jrow}$are defined as in Section 39.5 and the range of *nreg* is from 0 to $nhreg \cdot nvreg$. Two operators are defined to perform these contractions separately for quantities defined on T cells and U cells. They are given by

$$\overline{\alpha_{i,k,j}}^{\mathcal{U}}(nreg) = \frac{1}{Vol^{\mathcal{U}}(nreg)} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \alpha_{i,k,j} \cdot \Delta^U_{i,k,j} \tag{40.200}$$

$$\overline{\beta_{i,k,j}}^{\mathcal{T}}(nreg) = \frac{1}{Vol^{\mathcal{T}}(nreg)} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \beta_{i,k,j} \cdot \Delta^T_{i,k,j} \tag{40.201}$$

$$\tag{40.202}$$

where $\alpha_{i,k,j}$ is defined on U cells, $\beta_{i,k,j}$ is defined on T cells, and the respective volume elements and total volumes for each region are

$$\Delta^U_{i,k,j} = umask_{i,k,j} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{40.203}$$

$$Vol^{\mathcal{U}}(nreg) = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^U_{i,k,j} \tag{40.204}$$

$$Vol^{\mathcal{U}}(0) = \sum_{nreg=1}^{nhreg} Vol^{\mathcal{U}}(nreg) \tag{40.205}$$

$$\Delta^T_{i,k,j} = tmask_{i,k,j} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{40.206}$$

$$Vol^{\mathcal{T}}(nreg) = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^T_{i,k,j} \tag{40.207}$$

$$Vol^{\mathcal{T}}(0) = \sum_{nreg=1}^{nhreg} Vol^{\mathcal{T}}(nreg) \tag{40.208}$$

with the relation between *j* and *jrow* as described in Section 14.2.

40.17.1 Momentum Equations

Using arrays and operators described in Section 22.9.5, all components of the momentum Equation are contracted into regional volumes with the canonical forms given below. One way to think of this is that all cells within a regional volume $\mathcal{U}(nreg)$are replaced by one gigantic U cell and the value of all terms in the momentum equation are given for this one cell. Subscript $n = 1$ refers to the zonal velocity component and $n = 2$ refers to the meridional component.

$$termbm_{1,n,nreg} = \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{\mathcal{U}(nreg)} \tag{40.209}$$

$$termbm_{2,n,nreg} = \overline{-grad_p_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{40.210}$$

$$termbm_{3,n,nreg} = \overline{-ADV_Ux_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.211}$$

$$termbm_{4,n,nreg} = \overline{-ADV_Uy_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.212}$$

$$termbm_{5,n,nreg} = \overline{-ADV_Uz_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.213}$$

$$termbm_{6,n,nreg} = \overline{DIFF_Ux_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.214}$$

$$termbm_{7,n,nreg} = \overline{DIFF_Uy_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.215}$$

$$termbm_{8,n,nreg} = \overline{DIFF_Uz_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.216}$$

$$termbm_{9,n,nreg} = \overline{DIFF_metric_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{40.217}$$

$$termbm_{10,n,nreg} = \overline{CORIOLIS_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{40.218}$$

$$termbm_{11,n,nreg} = \overline{source_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.219}$$

$$termbm_{12,n,nreg} = \overline{-grad_p^s_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{40.220}$$

$$termbm_{13,n,nreg} = \overline{ADV_metric_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{40.221}$$

$$\tag{40.222}$$

where the surface pressure gradient terms $grad_p^s_{i,k,j,1}$ and $grad_p^s_{i,k,j,2}$ are reconstructed from Equations (29.3) and (29.4) by replacing the vertically integrated velocities by ψ with the aid of Equations (29.6) through (29.10) to yield

$$\begin{aligned} grad_p^s_{i,k,j,1} \equiv \frac{1}{\rho_\circ \cos\phi^U_{jrow}} \delta_\lambda(\overline{p^s_{i,jrow}}^\phi) &= \frac{1}{2\Delta\tau}\Big(\frac{1}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^\lambda)\Big) \\ &+ \frac{\tilde{f}_{jrow}}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^\phi) + zu_{i,jrow,1} \end{aligned} \tag{40.223}$$

$$\begin{aligned} grad_p^s_{i,k,j,2} \equiv \frac{1}{\rho_\circ}\delta_\phi(\overline{p^s_{i,jrow}}^\lambda) &= \frac{1}{2\Delta\tau}\Big(-\frac{1}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^\phi)\Big) \\ &+ \frac{\tilde{f}_{jrow}}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^\lambda) + zu_{i,jrow,2} \end{aligned} \tag{40.224}$$

where $\tilde{f}_{jrow}$ is given by Equation (29.5). The equation for the gigantic grid cell in each regional volume $\mathcal{U}(nreg)$is given by

$$termbm_{1,n,nreg} = \sum_{\ell=2}^{13} termbm_{\ell,n,nreg} \tag{40.225}$$

Note that terms 3,4, and 5 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the true advection in λ, ϕ, z because they contain divergent components. The true advection in λ, ϕ, z is given below by terms 14,15, and 16.

$$termbm_{14,n,nreg} = \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_veu_{i,k,j} - adv_veu_{i-1,k,j}}{dxu_i \cdot \cos\phi^U_{jrow}} - ADV_Ux_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.226}$$

$$termbm_{15,n,nreg} = \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_vnu_{i,k,j} - adv_vnu_{i,k,j-1}}{dyu_{jrow} \cdot \cos\phi^U_{jrow}} - ADV_Uy_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.227}$$

$$termbm_{16,n,nreg} = \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_vbu_{i,k-1,j} - adv_vnu_{i,k,j}}{dzt_k} - ADV_Uz_{i,k,j}}^{\mathcal{U}(nreg)} \tag{40.228}$$

$$termbm_{17,n,nreg} = \overline{u_{i,k,j,n,\tau}}^{\mathcal{U}(nreg)} \tag{40.229}$$

$$avgw_{nreg} = \overline{\frac{adv_vbu_{i,k-1,j} + adv_vbu_{i,k,j}}{2}}^{\mathcal{U}(nreg)} \tag{40.230}$$

$$smflx_{n,nreg} = \overline{smf_{i,j,n}}^{\mathcal{U}(nreg)} \tag{40.231}$$

Term 17 represents the average horizontal velocity components within the cell and *avgw* is the average vertical component. *smflx* represents the windstress acting on the top of the near surface cells and Equation (40.231) is only averaged over the regional volumes at the ocean surface.

40.17.2 Tracer Equations

Using operators and arrays described in Section 22.8.7, all components of the tracer equation are contracted into regional volumes with the canonical forms given below. As with the momentum contraction above, one way to think of this is that all cells within a regional volume $\mathcal{T}(nreg)$are replaced by one gigantic T cell and the value of all terms in the tracer equation are given for this one cell. Subscript $n = 1$ refers to the temperature component and $n = 2$ refers to the salinity.

$$termbt_{1,n,nreg} = \overline{\frac{t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{\mathcal{T}(nreg)} \tag{40.232}$$

$$termbt_{2,n,nreg} = \overline{-ADV_Tx_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.233}$$

$$termbt_{3,n,nreg} = \overline{-ADV_Ty_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.234}$$

$$termbt_{4,n,nreg} = \overline{-ADV_Tz_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.235}$$

$$termbt_{5,n,nreg} = \overline{DIFF_Tx_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.236}$$

$$termbt_{6,n,nreg} = \overline{DIFF_Ty_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.237}$$

$$termbt_{7,n,nreg} = \overline{DIFF_Tz_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.238}$$

$$termbt_{8,n,nreg} = \overline{source_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.239}$$

$$termbt_{9,n,nreg} = \overline{explicit\ \ convection}^{\mathcal{T}(nreg)} \tag{40.240}$$

$$\tag{40.241}$$

When options *gent_mcwilliams* and *gm_advect* are enabled, then Equations (40.233), (40.234), and (40.235) also include the flux form of the advection terms from option *gm_advect* given by $-ADV_Txiso_{i,k,j}$, $-ADV_Tyiso_{i,k,j}$, and $-ADV_Tziso_{i,k,j}$, respectively. Alternatively, when options and *gent_mcwilliams* and the default *gm_skew* are enabled, or if option *biharmonic_rm* is enabled, then Equations (40.236), (40.237), and (40.238) include the flux form of the skew-diffusive terms.

In Equation (40.240), the quantity *explicit convection* comes from solving Equation (40.232) before and after explicit convection. The equation for the gigantic grid cell in each regional volume $\mathcal{T}(nreg)$is given by

$$termbt_{1,n,nreg} = \sum_{\ell=2}^{9} termbt_{\ell,n,nreg} \tag{40.242}$$

Note that terms 2,3, and 4 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the true advection in λ, ϕ, z because they contain divergent components. The canonical form for the true advection in λ, ϕ, z is given below by terms 11,12, and 13.

$$termbt_{11,n,nreg} = \overline{t_{i,k,j,n,\tau} \cdot \frac{adv_vet_{i,k,j} - adv_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} - ADV_Tx_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.243}$$

$$termbt_{12,n,nreg} = \overline{t_{i,k,j,n,\tau} \cdot \frac{adv_vnt_{i,k,j} - adv_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} - ADV_Ty_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.244}$$

$$termbt_{13,n,nreg} = \overline{t_{i,k,j,n,\tau} \cdot \frac{adv_vbt_{i,k-1,j} - adv_vnt_{i,k,j}}{dzt_k} - ADV_Tz_{i,k,j}}^{\mathcal{T}(nreg)} \tag{40.245}$$

$$termbt_{15,n,nreg} = \overline{t_{i,k,j,n,\tau}}^{\mathcal{T}(nreg)} \tag{40.246}$$

$$stflx_{n,nreg} = \overline{stf_{i,j,n}}^{\mathcal{T}(nreg)} \tag{40.247}$$

When options and *gent_mcwilliams* and *gm_advect* are enabled, Equations (40.243), (40.244), and (40.245) also include the advective or transport velocities from option *gm_advect* given by $adv_vetiso_{i,k,j}$, $adv_vntiso_{i,k,j}$, and $adv_vbtiso_{i,k,j}$ respectively. Term 15 represents the average tracer within the cell and term *stflx* represents the surface tracer flux acting on the top of the surface cells. Equation (40.247) is only averaged over the regional volumes at the ocean surface.

40.18 time_averages

Option *time_averages* saves the same variables as does option *snapshots*. However, the variables are time averaged for stability and are defined on an averaging grid which can be the model grid or a sparse subset of the model grid. Often, for analysis purposes, it is not necessary to have data at every grid point or for the entire domain. When this is the case, the size of the archive disk space may be significantly reduced. The variables are tracers $t_{i,k,j,n,\tau}$ for $n = 1, nt$, horizontal velocities $u_{i,k,j,n,\tau}$ for $n = 1, 2$, vertical velocity at the base of T cells $adv_vbt_{i,k,j}$, surface tracer flux $stf_{i,j,n}$ for $n = 1, nt$, surface momentum flux $smf_{i,j,n}$ for $n = 1, 2$, and the external mode which is given by either[3] $psi_{i,jrow,\tau}$ or $ps_{i,jrow,\tau}$.

Before using this diagnostic, the averaging grid must be constructed by first executing script *run_timeavgs* with option *drive_timeavgs* enabled. This runs the averaging grid generator in a stand alone mode. To define the averaging grid, follow the example in the USER INPUT section of *timeavgs.F*. It is possible to arbitrarily pick which model grid points will be on the averaging grid. However, the code is set to specify a constant spacing between points on the averaging grid in longitude, latitude, and depth. Model grid points nearest to this spacing will be chosen as the averaging grid. To incorporate this averaging grid into MOM, follow the directions given when script *run_timeavgs* executes.

The output from this diagnostic is written as 32 bit IEEE unformatted data to file *time_mean.yyyyyy.mm.dd.dta*. If option *netcdf* or *time_averages_netcdf* is enabled, data is written in NetCDF format to file *time_mean.yyyyyy.mm.dd.dta.n* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2.

It should be noted that for use with Ferret, file *time_mean.yyyyyy.mm.dd.dta.nc* has the x, y, and z axes defined such that "edges" of cells along any axis on the T grid coincide with grid point coordinates on the U grid and visa versa. This is important when performing operations like integrals from within Ferret. For instance, if the dimensions of cell $T_{i,k,j}$are dxt_i, dyt_{jrow}, and dzt_k, then the cell edges are defined as follows: eastern edge of cell $T_{i,k,j}$is at $xtav_i_edges_i = xuav_i$ and the western edge is at $xtav_i_edges_{i-1} = xuav_{i-1}$. The northern edge is at $ytav_j_edges_{jrow} = yuav_{jrow}$ and the southern edge is at $ytav_j_edges_{jrow-1} = yuav_{jrow-1}$ and the bottom edge is at $ztav_k_edges_k = zwav_k$ and the top edge is at $ztav_k_edges_{k-1} = zwav_{k-1}$. Therefore, the cell dimensions can be computed from

$$dxt_i = xtav_i_edges_i - xtav_i_edges_{i-1} \tag{40.248}$$
$$dyt_{jrow} = ytav_j_edges_{jrow} - ytav_j_edges_{jrow-1} \tag{40.249}$$
$$dzt_k = ztav_k_edges_{k-1} - ztav_k_edges_k \tag{40.250}$$

and the volume is then given by

$$T_vol_{i,k,jrow} = dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{40.251}$$

For cell $U_{i,k,j}$, the corresponding cell dimensions are

$$dxu_i = xuav_i_edges_i - xuav_i_edges_{i-1} \tag{40.252}$$
$$dyu_{jrow} = yuav_j_edges_{jrow} - yuav_j_edges_{jrow-1} \tag{40.253}$$
$$dzt_k = ztav_k_edges_{k-1} - ztav_k_edges_k \tag{40.254}$$

[3]Depending on whether option *stream_function*, *rigid_lid_surface_pressure* or *implicit_free_surface* is enabled.

and the volume is

$$U_vol_{i,k,jrow} = dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{40.255}$$

Note that the above definitions are only true if the averaging grid is the same as the model grid. They should not be used when the averaging grid is a subset of the model grid.

The interval between output is specified by variable *timavgint*. The averaging period is controlled by variable *timavgper*. Typically the averaging period is set equal to the interval for output but it may be specified as less than the interval. How about producing monthly averages when months vary in length? Enabling option *monthly_averages* will over-ride values in *timavgper* and *timavgint* and make it so. However, this option is global and applies to all time averaged diagnostics that are enabled. Refer to Section 40.11 for more details. Variables *timavgper* and *timavgint* are input through namelist. Refer to Section 14.4 for information on namelist variables.Refer to Section 39.4 for a discussion of when this is appropriate.

If option *time_averages* can produce the same output as option *snapshots* then isn't option *snapshots* redundant? It would be except for the fact that option *time_averages* needs storage space to accumulate data to produce averages. This space is equal to the size of the data being averaged. Typically $5(imt \cdot jmt \cdot km + imt \cdot jmt)$ words are needed. If option *time_averages_disk* is enabled, disk space is used for the storage area, otherwise the storage space is in memory! If memory is insufficient or solid state disk space is insufficient, option *snapshots* is the only way to save the data.

40.19 time_step_monitor

Option *time_step_monitor* computes instantaneous values of total kinetic energy[4] per unit volume averaged over the entire domain volume in units of $gm/cm/sec^2$. It also computes first and second moments of tracer quantities. The mean tracer is the first moment and tracer variance[5] is the second moment about the mean in units of tracer squared. Additionally, the quantity $|\partial T/\partial t|$ in units of *degC/sec* (for n=1) and *(grams of salt per grams of water)/sec* (for n=2) averaged over the domain volume is computed. The explicit forms are given by

$$< ke > = \frac{\rho_\circ}{2}\frac{1}{Vol^U}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\sum_{n=1}^{2} u^2_{i,k,j,n,\tau} \cdot \Delta^U_{i,k,j} \tag{40.256}$$

$$< t_{i,k,j,n,\tau} > = \frac{1}{Vol^T}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} t_{i,k,j,n,\tau} \cdot \Delta^T_{i,k,j} \tag{40.257}$$

$$tracer_variance = < t^2_{i,k,j,n,\tau} > - < t_{i,k,j,n,\tau} >^2 \tag{40.258}$$

$$< |\delta_\tau(t_{i,k,j,n,\tau})| > = \frac{1}{Vol^T}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} \frac{|t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}|}{2\Delta\tau} \cdot \Delta^T_{i,k,j} \tag{40.259}$$

where

[4]Neglecting the vertical velocity component on the basis of scale analysis.

[5]Tracer variance is not conserved if explicit convection is active or there is a non zero surface tracer flux. It is also not conserved when diffusion is present. Refer to Appendix B for further discussion.

$$< t^2_{i,k,j,n,\tau} >= \frac{1}{Vol^T} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} t^2_{i,k,j,n,\tau} \cdot \Delta^T_{i,k,j} \tag{40.260}$$

and the volume elements for U cells and T cells are

$$\Delta^U_{i,k,j} = umask_{i,k,j} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{40.261}$$

$$\Delta^T_{i,k,j} = tmask_{i,k,j} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{40.262}$$

The masks $umask_{i,k,j}$ and $tmask_{i,k,j}$ are 1.0 on ocean U cells and T cells but 0.0 on land U cells and T cells. Also, the relation between j and $jrow$ is as described in Section 14.2. The total volumes are constructed as

$$Vol^U = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^U_{i,k,j} \tag{40.263}$$

$$Vol^T = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^T_{i,k,j} \tag{40.264}$$

In addition to the above quantities, the iteration count from the elliptic solver is saved. When option *netcdf* is enabled, the output from this diagnostic has a NetCDF format and is written to file *ts_integrals.yyyyyy.mm.dd.dta.nc*. Otherwise, the output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *ts_integrals.yyyyyy.mm.dd.dta*. If option *netcdf* or *time_step_monitor_netcdf* is enabled, data is written in NetCDF format rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *tsiint* and the control is specified by variable *iotsi*.

40.20 topog_diagnostic

The output from option *topog_diagnostic* is ocean depth at T cells given by

$$HT_{i,jrow} = zw_{kmt_{i,jrow}} \tag{40.265}$$

in units of cm and $f/HT_{i,jrow}$ in units of $cm^{-1}sec^{-1}$ where the Coriolis term f is defined at the latitude of T cells. When this option is enabled, output is written in a NetCDF format to file *topog.dta.nc*. There is no 32 bit IEEE unformatted data option. Therefore, the only way to get it is to enable option *topog_diagnostic* (using option *netcdf* will not do it). Although this option is intended to be used when executing the topography module in stand alone mode from script *run_topog*, it can also be enabled in a model run using script *run_mom*. There is no associated interval or control variable.

40.21 tracer_averages

Option *tracer_averages* constructs instantaneous spatial averages of tracers over regional areas described by $mskhr_{i,jrow}$in Section 39.5. This diagnostic is useful when trying to examine how the model equilibrates with area and depth. The average of tracer n as a function of depth level k and region number m is constructed as

$$\overline{T}_{k,m,n} = \frac{1}{Area^T_{k,m}} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} t_{i,k,j,n,\tau} \cdot A^T_{i,k,j} \tag{40.266}$$

$$m = mskhr_{i,jrow} \tag{40.267}$$

where the area element and area of each region at level k are given by

$$A^T_{i,k,j} = tmask_{i,k,j} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \tag{40.268}$$

$$Area^T_{k,m} = \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} A^T_{i,k,j} \tag{40.269}$$

$$\tag{40.270}$$

and the relation between j and $jrow$ are as described in Section 14.2. The average of tracer n over all regions as a function of depth is given by

$$\overline{T}^m_{k,n} = \frac{1}{Area^T_{k,0}} \sum_{m=1}^{nhreg} \overline{T}_{k,m,n} \cdot Area^T_{k,m} \tag{40.271}$$

$$Area^T_{k,0} = \sum_{m=1}^{nhreg} Area^T_{k,m} \tag{40.272}$$

where $Area^T_{k,0}$ is the total area of all regions at level k. In a likewise manner, the tracer flux through the ocean surface for tracer n as a function of region number m is given by

$$\overline{stf}_{m,n} = \frac{1}{Area^T_{1,m}} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} stf_{i,j,n} \cdot A^T_{i,1,j} \tag{40.273}$$

$$\overline{stf}^m_n = \frac{1}{Area^T_{1,0}} \sum_{m=1}^{nhreg} \overline{stf}_{m,n} \cdot Area^T_{1,m} \tag{40.274}$$

The output from this diagnostic is written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *tracer_avg.yyyyyy.mm.dd.dta*. If option *netcdf* or *tracer_averages_netcdf* is enabled, data is written in NetCDF format to file *tracer_avg.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *tavgint* and the control is specified by variable *iotavg*. Note that the regional mask $mskhr_{i,jrow}$is also written to file *tracer_avg.yyyyyy.mm.dd.dta* when the initialization boolean *itavg* is true. It should be set true on the first run but false thereafter.

40.22 tracer_yz

Option *tracer_yz* computes instantaneous values of the zonal integral of the tracer and tracer equation, term by term for each component. Contraction of the tracer equation along longitude yields a two dimensional equation as a function of latitude and depth. Using operators described in Section 22.8.7 yields

$$\begin{aligned}\frac{1}{L^T_{k,j}}\sum_{i=2}^{imt-1}\Delta^T_{i,k,j}\cdot\delta_t(T_{i,k,j,n,\tau}) = & -\frac{1}{L^T_{k,j}}\sum_{i=2}^{imt-1}\Delta^T_{i,k,j}(ADV_Tz_{i,k,j}+ADV_Ty_{i,k,j})\\ & +\frac{1}{L^T_{k,j}}\sum_{i=2}^{imt-1}\Delta^T_{i,k,j}(DIFF_Tz_{i,k,j}+DIFF_Ty_{i,k,j})\\ & +\frac{1}{L^T_{k,j}}\sum_{i=2}^{imt-1}\Delta^T_{i,k,j}\cdot source_{i,k,j}\end{aligned} \tag{40.275}$$

where n is the tracer and the relation between j and *jrow* is as described in Section 14.2. The zonal advection and diffusion are not shown because they are eliminated by the summing operation. The length element and total length are given by

$$\Delta^T_{i,k,j} = tmask_{i,k,j}\cdot dxt_i \tag{40.276}$$

$$L^T_{k,j} = \sum_{i=2}^{imt-1}\Delta^T_{i,k,j} \tag{40.277}$$

Each sum in Equation (40.275) is output as one data field along with the zonally averaged tracer. The output from this diagnostic is only written in NetCDF format (no option for IEEE) to file *tracer_yz.yyyyyy.mm.dd.dta.nc*. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *tyzint*.

40.23 trajectories

Option *trajectories* integrates particles along trajectories[6] using a forward time step and a particle velocity determined by instantaneous linear interpolation every time step.
Let n particles be placed at positions $\mathcal{P}_1(x,y,z), \mathcal{P}_2(x,y,z), \mathcal{P}_3(x,y,z)\cdots\mathcal{P}_n(x,y,z)$ at time $t0$. The position of the *nth* particle at time $t1$ is given as

$$\mathcal{P}_n(x,y,z) = \int_{t0}^{t1} V_n(x,y,z)\cdot dt \tag{40.278}$$

which is discretized as

$$\mathcal{P}_n^{\tau+1}(x,y,z) = \mathcal{P}_n^{\tau}(x,y,z) + \Delta\tau\cdot V_n^{\tau}(x,y,z) \tag{40.279}$$

[6]On the fly while MOM integrates

where V_n^τ is the particle velocity at $\mathcal{P}_n^\tau(x, y, z)$ arrived at by linear interpolation from $u_{i,k,j,1,\tau}$, $u_{i,k,j,2,\tau}$, and $adv_vbt_{i,k,j}$.

Particles are neutrally buoyant and are initially spread uniformly within an arbitrary volume as indicated by the example in *ptraj.F*. If option *lyapunov* is enabled, the deformation rate matrix *em* is also calculated[7] as the particles are integrated (Pierrehumpert, Yang 1993). The Lyapunov exponent λ is useful in quantifying the dispersion of the particle cloud and can be computed from the eigenvalues of this matrix as follows. Let

$$c = (em_{2,2} - em_{1,1})^2 + 4(em_{1,2} \cdot em_{2,1}) \tag{40.280}$$

Then the Lyapunov coefficient is given by

$$\lambda \quad = \quad \log(|\theta|)/T \tag{40.281}$$

If $c >= 0$ then

$$|\theta| = \frac{|(em_{1,1} + em_{2,2})^2 \pm \sqrt{(c)}|}{2} \tag{40.282}$$

Otherwise

$$|\theta| = \frac{\sqrt{(em_{1,1} + em_{2,2})^2} + |c|}{2} \tag{40.283}$$

This diagnostic is useful for investigating the evolution of water masses, three dimensional flow structure, and mixing properties of currents and waves (Stokes drift, chaotic mixing, etc).

The storage requirement is six times[8] the number of particles and the computation is relatively fast. How many particles are reasonable? Start with about 10,000 and depending on how the results look, make adjustments. Particle positions are saved to the restart file to provide the necessary starting point for integrating the particles for arbitrary lengths of time. The output from Equation (40.279) which consists of an (x,y,z) position and a set of 3 integers per particle may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *particles.yyyyyy.mm.dd.dta*. If option *netcdf* or *trajectories_netcdf* is enabled, data is written in NetCDF format to file *particles.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *trajint* and the control is specified by variable *iotraj*.

40.24 save_xbts

Originally, option *save_xbts* sampled temperatures and salinities at various latitude and longitude locations on the model grid down to some prescribed level and produced time averages

[7]Only in two dimensions: longitude and latitude.
[8]Nine times if option *lyapunov* is enabled.

of these quantities. In time, it grew to construct time averages of all terms in the prognostic equations[9] at each model level down to a specified depth for a set of stations. Station locations and depth at each station can be specified by looking at the USER INPUT section of *xbt.F* and following the examples. This diagnostic is useful when trying to understand the time evolution of dominant balances at specific locations in MOM. For instance, deploying a group of XBTs can elucidate how waves propagate or currents meander. It is also useful for planning where to deploy instrumented arrays or moorings in ship based experiments. *save_xbts* is similar to *term_balances* but instead of averaging over space in various regions of the domain, it averages over time at specific stations to prevents aliasing. Unlike *term_balances* this diagnostic is fast although it can get expensive in memory[10]. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *xbt.yyyyyy.mm.dd.dta*. If option *netcdf* or *xbts_netcdf* is enabled, data is written in NetCDF format to file *xbt.yyyyyy.mm.dd.dta.nc* rather than in unformatted IEEE. The "yyyyyy.mm.dd" is a place holder for year, month, and day and this naming convention is explained further in Section 39.2. The interval between output is specified by variable *xbtint* and the control is specified by variable *ioxbt*. The averaging period *xbtper* is typically set equal to the interval but may be specified shorter. How about producing monthly averages when months vary in length? Enabling option *monthly_averages* will over-ride values in *xbtper* and *xbtint* and make it so. However, this option is global and applies to all time averaged diagnostics that are enabled. Refer to Section 40.11 for more details. Variables *xbtper* and *xbtint* are input through namelist. Refer to Section 14.4 for information on namelist variables.
Define m stations with coordinates (λ_m, ϕ_m) which are sampled to a depth of z cm. Discretize these locations to the nearest model grid points ($xbtlon_m$, $xbtlat_m$, $xbtdpt_m$).

40.24.1 Momentum Equations

Using operators and arrays described in Section 22.9.5, all components of the Momentum Equation are averaged in time for each station m and at each level k. The averaging period L corresponds to the number of time steps in the average. Subscript $n = 1$ refers to the zonal velocity component and $n = 2$ refers to the meridional component.

The canonical form of the terms for the *mth* station at level k are given as

$$uxbt_{1,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau} \tag{40.284}$$

$$uxbt_{2,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -grad_p_{i,k,j,n} \tag{40.285}$$

$$uxbt_{3,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Ux_{i,k,j} \tag{40.286}$$

$$uxbt_{4,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Uy_{i,k,j} \tag{40.287}$$

$$uxbt_{5,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Uz_{i,k,j} \tag{40.288}$$

[9] This option has outgrown its name. Think of it as an enhanced XBT.
[10] 67 items are computed for each T and U grid cell combination

$$uxbt_{6,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Ux_{i,k,j} \tag{40.289}$$

$$uxbt_{7,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Uy_{i,k,j} \tag{40.290}$$

$$uxbt_{8,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Uz_{i,k,j} \tag{40.291}$$

$$uxbt_{9,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_metric_{i,k,j,n} \tag{40.292}$$

$$uxbt_{10,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} CORIOLIS_{i,k,j,n} \tag{40.293}$$

$$uxbt_{11,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} source_{i,k,j} \tag{40.294}$$

$$uxbt_{12,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -grad_p^s_{i,k,j,n} \tag{40.295}$$

$$uxbt_{13,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} ADV_metric_{i,k,j,n} \tag{40.296}$$

$$\tag{40.297}$$

where the surface pressure gradient terms $grad_p^s_{i,k,j,1}$ and $grad_p^s_{i,k,j,2}$ are reconstructed from Equations (29.3) and (29.4) by replacing the vertically integrated velocities by ψ with the aid of Equations (29.6) through (29.10) to yield

$$\begin{aligned} grad_p^s_{i,k,j,1} \equiv \frac{1}{\rho_\circ \cos\phi^U_{jrow}}\delta_\lambda(\overline{p^s_{i,jrow}}^{\phi}) &= \frac{1}{2\Delta\tau}\Big(\frac{1}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^{\lambda})\Big) \\ &+ \frac{\tilde{f}_{jrow}}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^{\phi}) + zu_{i,jrow,1} \end{aligned} \tag{40.298}$$

$$\begin{aligned} grad_p^s_{i,k,j,2} \equiv \frac{1}{\rho_\circ}\delta_\phi(\overline{p^s_{i,jrow}}^{\lambda}) &= \frac{1}{2\Delta\tau}\Big(-\frac{1}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^{\phi})\Big) \\ &+ \frac{\tilde{f}_{jrow}}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^{\lambda}) + zu_{i,jrow,2} \end{aligned} \tag{40.299}$$

where $\tilde{f}_{jrow}$ is given by Equation (29.5). The equation for the *kth* grid cell at the *mth* station is given by

$$uxbt_{1,k,n,mth} = \sum_{\ell=2}^{13} uxbt_{\ell,k,n,mth} \tag{40.300}$$

Note that terms 3,4, and 5 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the physical advection in λ, ϕ, z because they contain divergent components. The canonical form of the physical advection in λ, ϕ, z is given below by terms 14,15, and 16.

$$uxbt_{14,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv_veu_{i,k,j} - adv_veu_{i-1,k,j}}{dxu_i \cdot \cos\phi^U_{jrow}} - ADV_Ux_{i,k,j} \quad (40.301)$$

$$uxbt_{15,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv_vnu_{i,k,j} - adv_vnu_{i,k,j-1}}{dyu_{jrow} \cdot \cos\phi^U_{jrow}} - ADV_Uy_{i,k,j} \quad (40.302)$$

$$uxbt_{16,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv_vbu_{i,k-1,j} - adv_vnu_{i,k,j}}{dzt_k} - ADV_Uz_{i,k,j} \quad (40.303)$$

$$uxbt_{17,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \quad (40.304)$$

$$xbtw_{mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{adv_vbu_{i,k-1,j} + adv_vbu_{i,k,j}}{2} \quad (40.305)$$

$$uxbtsf_{n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} smf_{i,j,n} \quad (40.306)$$

Term 17 represents the average horizontal velocity components within the cell and *xbtw* is the average vertical component. *uxbtsf* represents the windstress acting on the top of the near surface cell.

40.24.2 Tracer Equations

Using operators and arrays described in Section 22.8.7, all components of the Tracer Equation are averaged in time for each station m and at each level k. The averaging period L corresponds to the number of time steps in the average. Subscript $n = 1$ refers to the temperature component and $n = 2$ refers to the salinity.

$$txbt_{1,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}}{2\Delta\tau} \quad (40.307)$$

$$txbt_{2,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Tx_{i,k,j} \quad (40.308)$$

$$txbt_{3,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Ty_{i,k,j} \quad (40.309)$$

$$txbt_{4,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV_Tz_{i,k,j} \quad (40.310)$$

$$txbt_{5,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Tx_{i,k,j} \quad (40.311)$$

$$txbt_{6,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Ty_{i,k,j} \quad (40.312)$$

$$txbt_{7,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF_Tz_{i,k,j} \quad (40.313)$$

$$txbt_{8,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} source_{i,k,j} \quad (40.314)$$

$$txbt_{9,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} explicit\ \ convection \tag{40.315}$$

$$\tag{40.316}$$

When options *gent_mcwilliams* and *gm_advect* are enabled, then Equations (40.308), (40.309), and (40.310) also include the flux form of the advection terms from option *gm_advect* given by $-ADV_Txiso_{i,k,j}$, $-ADV_Tyiso_{i,k,j}$, and $-ADV_Tziso_{i,k,j}$ respectively. Alternatively, when options and *gent_mcwilliams* and the default *gm_skew* are enabled, of if option *biharmonic_rm* is enabled, then the operators $DIFF_Tx_{i,k,j}$, $DIFF_Ty_{i,k,j}$, and $DIFF_Tz_{i,k,j}$ include the flux form of the skew-diffusive terms from option *gm_skew*. In Equation (40.315), the quantity *explicit convection* comes from solving Equation (40.232) before and after explicit convection. The equation for the *kth* grid cell in station *m* is given by

$$txbt_{1,k,n,mth} = \sum_{\ell=2}^{9} txbt_{\ell,k,n,mth} \tag{40.317}$$

Note that terms 2,3, and 4 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the physical advection in λ, ϕ, z because they contain divergent components. The physical advection in λ, ϕ, z is given below by terms 11,12, and 13.

$$txbt_{11,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv_vet_{i,k,j} - adv_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} - ADV_Tx_{i,k,j} \tag{40.318}$$

$$txbt_{12,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv_vnt_{i,k,j} - adv_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} - ADV_Ty_{i,k,j} \tag{40.319}$$

$$txbt_{13,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv_vbt_{i,k-1,j} - adv_vnt_{i,k,j}}{dzt_k} - ADV_Tz_{i,k,j} \tag{40.320}$$

$$txbt_{15,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \tag{40.321}$$

$$txbtsf_{n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} stf_{i,j,n} \tag{40.322}$$

When options *gent_mcwilliams* and *gm_advect* are enabled, Equations (40.318), (40.319), and (40.320) also include the transport velocities from option *gm_advect* given by $adv_vetiso_{i,k,j}$, $adv_vntiso_{i,k,j}$, and $adv_vbtiso_{i,k,j}$ respectively. Term 15 represents the average tracer within the cell and term *txbtsf* represents the surface tracer flux acting on the top of the surface cells.

Chapter 41

Diagnostics for numerical analysis

Inevitably, problems appear with the model code or configuration. The diagnostic options discussed in Chapter 40 will likely be the first place one detects problems, since these options are designed to show the physical properties of the solution. After problems are seen with those diagnostics, it might be useful to turn on some of the options discussed in this chapter, which are mostly designed for numerical purposes.

41.1 General debug options

In addition to the diagnostics listed in the following sections, there are numerous "debug" options in critical areas of the source code. These options can be enabled to give more information for debugging purposes. An example of one of these options is be *debug_adv_vel* near the bottom of file *adv_vel.F* which computes advective velocities. When enabled, *debug_adv_vel* gives the components of the divergence of advective velocities for all T-cells and U-cells in the vertical at any location identified by indices "i, *jrow*" . These "debug" options can also be found using UNIX *grep* as described in Section 3.1.

41.2 stability_tests

Option *stability_tests* computes various stability criteria and related items within a portion or all of the model domain. The limits of volume of domain to be considered when testing are set through namelist. Refer to Section 14.4 for information on namelist variables. If MOM blows up, this diagnostic is useful in finding where it went unstable. The following are computed:

1. Based on local velocities within each cell, a maximum local time step is computed for the three principal directions as

$$\Delta\tau^{x}_{i,k,j} = \frac{\cos\phi^{U}_{jrow} \cdot dxu_i}{2 \cdot u_{i,k,j,1,\tau}} \tag{41.1}$$

$$\Delta\tau^{y}_{i,k,j} = \frac{dyu_{jrow}}{2 \cdot u_{i,k,j,2,\tau}} \tag{41.2}$$

$$\Delta\tau^{z}_{i,k,j} = \frac{dzw_k}{2 \cdot adv_vbt_{i,k,j}} \tag{41.3}$$

These local time steps are compared with the model specified time step and the location of the largest is chosen as the position of the most unstable cell. If the local time step exceeds the model time step by an amount which can be set through namelist, then a CFL violation is detected. Refer to Section 14.4.4 for a discussion of the CFL condition and choosing time step lengths. The number of times a CFL violation is allowed may also be set through namelist. Refer to section 14.4.7 for setting a region over which stability calculations are performed. The default region is the entire domain. Variables within the local neighborhood of the offending cells are shown and when the number of offenses exceeds the allowable number, the model is brought down.

2. The local Reynolds number is estimated along each of the principle directions as

$$rey^x_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot \cos\phi^U_{jrow} dxu_i}{visc_ceu_{i,k,j}} \tag{41.4}$$

$$rey^y_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dyu_{jrow}}{visc_cnu_{i,k,j}} \tag{41.5}$$

$$rey^z_{i,k,j} = \frac{adv_vbu_{i,k,j} \cdot dzw_k}{visc_cbu_{i,k,j}} \tag{41.6}$$

and the location of the maximum is found and printed.

3. The local Peclet number is estimated along each of the principle directions as

$$pecl^x_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot \cos\phi^U_{jrow} dxu_i}{diff_cet_{i,k,j}} \tag{41.7}$$

$$pecl^y_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dyu_{jrow}}{diff_cnt_{i,k,j}} \tag{41.8}$$

$$pecl^z_{i,k,j} = \frac{adv_vbt_{i,k,j} \cdot dzw_k}{diff_cnt_{i,k,j}} \tag{41.9}$$

and the location of the maximum is found and printed.

4. The locations where numerics are breaking down and producing spurious tracer extrema are determined. This is done by searching the immediate neighborhood of cell (i, k, j) for extrema in temperature at τ and $\tau - 1$. If $t_{i,k,j,1,\tau+1}$ exceeds this extrema by an amount which may be specified through namelist, then there is numerical truncation at (i, k, j). Note that this statement can only be made because of the incompressibility condition. If there are more than 100 locations exhibiting numerical truncation, only the first 100 locations are shown.

5. The locations are shown where predicted temperatures or salinities are outside the bounds of temperatures and salinities which were used for the construction of density coefficients. If there are more than 100 locations where this occurs, only the first 100 locations are shown.

6. The location of maximum error in continuity is calculated considering all U cells and T cells separately.

7. The maximum error in vertical velocity at the ocean bottom on T cells is computed. This is the residual error from integrating Equation (22.17) vertically from the surface to the ocean bottom.

8. The maximum vertical velocity at the ocean bottom on U cells from Equation (22.24) is computed. This is non-zero if the bottom has a slope since the bottom flow is required to parallel the bottom slope.

As described above, if more than a specified number of CFL violations are found when this diagnostic is active (only at times specified by the interval), the integration will stop indicating where the most unstable locations are with matrix printouts of variables in the neighborhoods. If a violation is not found, statistics will be printed indicating how close the integration is to violating the CFL condition along with other information described above. The output from this diagnostic may only be written as ascii to the model *printout*. The interval between output is specified by variable *stabint*.

41.3 trace_coupled_fluxes

This diagnostic is useful as an aid to diagnosing problems with coupling to atmosphere models using option *coupled*. It gives some gross statistics for the surface boundary conditions at various key places as they are being constructed for both atmosphere and ocean models. Output from this diagnostic is only written to file printout and is not intended for post processing.

41.4 trace_indices

This diagnostic is useful when trying to understand how the memory window operates for various settings of *jmw*. Refer to Section 11.3.2 for a description of how the memory window works. It gives a trace of the latitude loop indices indicating the dataflow from disk to memory window as MOM executes. Also indicated are the latitude rows being worked on by various subroutines in preparation for solving the equations on rows in the memory window. As the memory window is moved northward, a listing of "which rows are copied where" is given. As new code is added to MOM, it is strongly recommended that this option be included as an aid in verifying that the code is correct. When using this option, disable all diagnostics and execute the model for only a limited number of time steps since the output can get voluminous.

Part IX

Appendices and references

Appendix A

Kinetic energy budget

The budget of kinetic energy is of fundamental interest in ocean modeling. The manners in which various processes contribute to this budget are considered in this chapter. In particular, a breakdown of the budget for the total kinetic energy, the external mode kinetic energy, and the internal mode kinetic energy are derived for the continuum equations in the case of a free surface. The rigid lid results are also indicated. After doing so, two aspects of the discrete model energetics are considered. First, a proof that the work done by the discrete pressure terms is equal to the work done by buoyancy is given. In addition, the arguments from Bryan (1969) and Semtner(1974) are summarized concerning the conservation of first and second moments for velocity. This result holds for the case of zero forcing, zero dissipation, and when employing centered-differenced advection of momentum. The conservation of these two moments for momentum prevent systematic errors that accumulate in time (i.e., spurious growth or decay of kinetic energy). Additionally, the preservation of the second moment eliminates the problems with Phillip's (1959) non-linear instability (Arakawa 1966, Bryan 1969). These two points have provided the strong motivation for employing centered advection of momentum in the GFDL ocean model. Note that there are alternative advection schemes for tracers, as discussed in Chapter 32. The use of these schemes for tracers does not compromise the numerical stability maintained by centered differenced momentum advection.

In general, it is important for an ocean model to provide a diagnostic of the kinetic energy density. Pragmatically, the implementation of numerous algorithms in MOM have been debugged through an analysis of the kinetic energy budget. The diagnostic option *energy_analysis* (Section 40.4) provides the domain averaged budget for the kinetic energy density for MOM.

A.1 Continuum version of the kinetic energy budget

This section discusses a kinetic energy budget for the ocean primitive equations in their continuous form. For the rigid lid approximation, the work of Holland (1975) is classic. Bryan and Lewis (1979), F. Bryan (1986), Treguier (1992), and Goddard (1995) provide further discussions and examples. The present derivations are given for the free surface, with the rigid lid results obtained in the limit of a fixed surface height.

A.1.1 The kinetic energy density

For the scalings relevant for a hydrostatic and Boussinesq fluid (i.e., the ocean primitive equations using the "Traditional Approximation"), the kinetic energy per unit volume (kinetic en-

ergy density) is determined just by the energy in the horizontal currents

$$e \equiv \frac{\rho_o}{2}(u^2 + v^2) = \frac{\rho_o}{2}\,\vec{u}_h \cdot \vec{u}_h. \tag{A.1}$$

Hence, to develop an equation for the kinetic energy, it is necessary to consider the horizontal momentum equations

$$u_t = -\nabla \cdot (\vec{u}\,u) + v\left(f + \frac{u \tan\phi}{a}\right) - \frac{p_\lambda}{a\rho_o \cos\phi} + (\kappa_m u_z)_z + F^u \tag{A.2}$$

$$v_t = -\nabla \cdot (\vec{u}\,v) - u\left(f + \frac{u \tan\phi}{a}\right) - \frac{p_\phi}{a\rho_o} + (\kappa_m v_z)_z + F^v, \tag{A.3}$$

where the horizontal frictional terms

$$F^{\vec{u}} = (F^u, F^v, 0) \tag{A.4}$$

were defined in Equations (9.187) and (9.193), and

$$\vec{u} = (u, v, w) = (\vec{u}_h, w) \tag{A.5}$$

is the velocity field.

A.1.2 External and internal mode kinetic energies

In MOM, the splitting of the flow into a vertically averaged velocity and a deviation from that average prompts an analysis of the kinetic energy which takes such a split into account. For this purpose, it is useful to introduce a depth averaging operator

$$\overline{\alpha} \equiv \frac{1}{H+\eta}\int_{-H}^{\eta} dz\,\alpha. \tag{A.6}$$

The symbol for deviations from the depth average is given by a hat

$$\widehat{\alpha} = \alpha - \overline{\alpha}. \tag{A.7}$$

Using this notation, the horizontal velocity components can be split into the external (depth averaged) and internal modes

$$(u, v) = (\overline{u}, \overline{v}) + (\widehat{u}, \widehat{v}). \tag{A.8}$$

Substituting these velocities into the kinetic energy density yields

$$e = \frac{\rho_o}{2}(\overline{u}\,\overline{u} + \overline{v}\,\overline{v}) + \frac{\rho_o}{2}(\widehat{u}\,\widehat{u} + \widehat{v}\,\widehat{v}) + \rho_o\,(\overline{u}\,\widehat{u} + \overline{v}\,\widehat{v}). \tag{A.9}$$

The depth averaged kinetic energy density is given by

$$\overline{e} = \frac{\rho_o}{2}(\overline{u}\,\overline{u} + \overline{v}\,\overline{v}) + \frac{\rho_o}{2}(\overline{\widehat{u}\,\widehat{u}} + \overline{\widehat{v}\,\widehat{v}}). \tag{A.10}$$

Note the uncoupling of the external and internal modes in the depth averaged kinetic energy density. Hence, the depth averaged kinetic energy density can be thought of as a contribution from the external mode kinetic energy density

$$e_{ext} = \frac{\rho_o}{2}(\overline{u}\,\overline{u} + \overline{v}\,\overline{v}), \tag{A.11}$$

and the depth averaged internal mode kinetic energy density

$$\overline{e_{int}} = \frac{\rho_o}{2}(\overline{\widehat{u}\,\widehat{u}} + \overline{\widehat{v}\,\widehat{v}}). \tag{A.12}$$

Of central interest is how the budget for the volume averaged kinetic energy density, derived in equation (A.37), breaks up into external and internal mode components. Namely, with

$$\begin{aligned}
\langle e \rangle &= V^{-1} \int d\Omega \int_{-H}^{\eta} dz\, e \\
&= V^{-1} \int d\Omega\, (H+\eta)\, e_{ext} + V^{-1} \int d\Omega \int_{-H}^{\eta} dz\, e_{int} \\
&= \langle e_{ext} \rangle + \langle e_{int} \rangle,
\end{aligned} \tag{A.13}$$

what are the terms determining the individual time evolution of $\langle e_{ext} \rangle$ and $\langle e_{int} \rangle$? That question is answered for the external mode in Section A.1.5, and the internal mode in the Section A.1.6.

A.1.3 Budget for the local kinetic energy

Taking the scalar product of $\rho_o(u,v) = \rho_o\, \vec{u}_h$ with the horizontal momentum equations (A.2) and (A.3) yields for the time tendency of the kinetic energy density

$$e_t = -\nabla \cdot (\vec{u}\, e) - \vec{u}_h \cdot \nabla_h p + \rho_o\, \vec{u}_h \cdot (F^{\vec{u}} + [\kappa_m (\vec{u}_h)_z]_z). \tag{A.14}$$

Therefore, the time tendency is determined by the three-dimensional convergence of the advective flux of kinetic energy density

$$-\nabla \cdot (\vec{u}\, e), \tag{A.15}$$

the advection of horizontal pressure gradients (i.e., work against the horizontal pressure gradients)

$$-\vec{u}_h \cdot \nabla_h p, \tag{A.16}$$

and the effects of frictional dissipation

$$\mathcal{F} \equiv \rho_o\, \vec{u}_h \cdot (F^{\vec{u}} + [\kappa_m (\vec{u}_h)_z]_z). \tag{A.17}$$

Using the continuity equation (4.61) and hydrostatic relation $p_z = -\rho\, g$, the kinetic energy density budget becomes

$$\begin{aligned}
e_t &= -\nabla \cdot (e\, \vec{u}) - \vec{u}_h \cdot \nabla_h p + \mathcal{F} \\
&= -\nabla \cdot (e\, \vec{u}) - \nabla_h \cdot (p\, \vec{u}_h) + p\, \nabla_h \cdot \vec{u}_h + \mathcal{F} \\
&= -\nabla \cdot (e\, \vec{u}) - \nabla_h \cdot (p\, \vec{u}_h) - p\, w_z + \mathcal{F} \\
&= -\nabla \cdot (e\, \vec{u}) - \nabla \cdot (p\, \vec{u}) + w\, p_z + \mathcal{F} \\
&= -\nabla \cdot [(e+p)\, \vec{u}] - w\, \rho\, g + \mathcal{F}.
\end{aligned} \tag{A.18}$$

In this form, the terms contributing to the time tendency of kinetic energy density are given by the three-dimensional convergence of an advective flux of kinetic energy density plus pressure

$$-\nabla \cdot [(e+p)\, \vec{u}], \tag{A.19}$$

the effects from buoyancy

$$-w\,\rho\,g, \tag{A.20}$$

and the effects from friction $\mathcal{F}$. Importantly, notice how for a closed domain, the contribution to the time tendency from buoyancy is equivalent to that arising from the horizontal pressure gradients. This equivalence is important to maintain in a numerical model, and is discussed more fully in Section A.2.4.

A.1.4 Budget for the volume averaged kinetic energy and kinetic energy density

The kinetic energy budget over a finite volume of ocean is derived in this section. The particular case of a vertical column extending from the ocean surface to the bottom is considered. The column is assumed to have fixed side and bottom boundaries, yet to have a generally non-constant free surface top. The derivation proceeds with the free surface, and specializes in relevant places to the rigid lid.

A.1.4.1 Budget for the kinetic energy within a vertical column

Start with the time tendency for the kinetic energy density given in the form

$$e_t \;=\; -\nabla\cdot[(e+p)\,\vec{u}] - w\,\rho\,g + \mathcal{F}. \tag{A.21}$$

Integrating the left hand side over a vertical column extending from the ocean bottom to the free surface yields

$$\int_{-H}^{\eta} dz\, e_t = -e(\eta)\,\eta_t + \partial_t \int_{-H}^{\eta} dz\, e. \tag{A.22}$$

The term $e(\eta)\,\eta_t$ vanishes in the rigid lid case. Rearrangement, integration, use of the kinematic boundary conditions (4.29) and (4.24), and use of the free surface equation (7.18), yields

$$\begin{aligned}
\partial_t \int_{-H}^{\eta} dz\, e \;&=\; e(\eta)\,\eta_t - [w(p+e)]_{z=\eta} + [w(p+e)]_{z=-H} \\
&+\; \int_{-H}^{\eta} dz\,(-\nabla_h\cdot[\vec{u}_h\,(p+e)] - w\,\rho\,g + \mathcal{F}) \\
&=\; [e(\eta)+p(\eta)]\,[\eta_t - w(\eta)] - p(\eta)\,\eta_t - [\vec{u}_h\cdot\nabla_h H\,(p+e)]_{z=-H} \\
&-\; \int_{-H}^{\eta} dz\,\nabla_h\cdot[\vec{u}_h\,(p+e)] + \int_{-H}^{\eta} dz\,(-w\,\rho\,g + \mathcal{F}) \\
&=\; [e(\eta)+p(\eta)]\,[Q_w - \vec{u}_h\cdot\nabla_h\eta] - p(\eta)\,\eta_t - [\vec{u}_h\cdot\nabla_h H\,(p+e)]_{z=-H} \\
&-\; \int_{-H}^{\eta} dz\,\nabla_h\cdot[\vec{u}_h\,(p+e)] + \int_{-H}^{\eta} dz\,(-w\,\rho\,g + \mathcal{F}) \\
&=\; e(\eta)\,Q_w + p(\eta)\,(Q_w - \eta_t) \\
&-\; \nabla_h\cdot\left(\int_{-H}^{\eta} dz\,\vec{u}_h\,(p+e)\right) + \int_{-H}^{\eta} dz\,(-w\,\rho\,g + \mathcal{F}) \\
&=\; e(\eta)\,Q_w + p(\eta)\,\nabla_h\cdot\vec{U} \\
&-\; \nabla_h\cdot\left(\int_{-H}^{\eta} dz\,\vec{u}_h\,(p+e)\right) + \int_{-H}^{\eta} dz\,(-w\,\rho\,g + \mathcal{F}).
\end{aligned} \tag{A.23}$$

A volume integration over fixed side walls, fixed bottom, and generally non-constant free surface height, leads to the following budget for the total kinetic energy within a vertical column

$$\begin{aligned}\partial_t \left(\int d\Omega \int_{-H}^{\eta} dz\, e \right) &= \int d\Omega \, \left(e\, Q_w + p\, \nabla_h \cdot \vec{U} \right) \Big|_{z=\eta} - \int d\Omega \, \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \vec{u}_h \, (p+e) \right) \\ &+ \int d\Omega \int_{-H}^{\eta} dz \, (-w\, \rho\, g + \mathcal{F}). \end{aligned} \tag{A.24}$$

A.1.4.2 Interpreting the terms in the kinetic energy budget

An interpretation of these terms is the following. The term

$$\int d\Omega \, e(\eta)\, Q_w \tag{A.25}$$

represents the power (energy/time) contributed by the fresh water flux at the free surface. This term is zero for the rigid lid case, since for the rigid lid there is no fresh water flux. Rather, there is a *virtual salt flux*. The term

$$\int d\Omega \, \left(p\, \nabla_h \cdot \vec{U} \right) \Big|_{z=\eta} \tag{A.26}$$

represents the work/time done by the surface pressure as it acts on the vertically integrated flow. This term is zero in the rigid lid since the vertically integrated flow is divergence-free (see Section 6.1 which discusses the barotropic streamfunction). It arguably should be in the rigid lid's budget, however, since the surface pressure in the rigid lid case is not zero. However, as seen in the derivation above, this term does not appear naturally in a rigid lid budget since it is ultimately related to the time tendency of the free surface height. The term

$$\begin{aligned} - \int d\Omega \, \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \vec{u}_h \, (p+e) \right) = &- a \int_{\phi_1}^{\phi_2} d\phi \, \left(\int_{-H}^{\eta} dz\, u(e+p) \right) \Big|_{\lambda_1}^{\lambda_2} \\ &- a \int_{\lambda_1}^{\lambda_2} d\lambda \, \left(\int_{-H}^{\eta} dz\, v \cos\phi \, (e+p) \right) \Big|_{\phi_1}^{\phi_2}, \end{aligned} \tag{A.27}$$

represents the convergence of the advective flux of $p+e$ across the lateral boundaries of the vertical column. If the lateral domain encompasses the whole ocean domain, this term vanishes due to the no-normal flow condition on the velocity field at the solid boundaries. In a periodic channel, this term also vanishes. For the rigid lid, the upper limit on the vertical integral is set to $z=0$ rather than $z=\eta$. The term

$$- \int d\Omega \int_{-H}^{\eta} dz\, w\, \rho\, g \tag{A.28}$$

represents the work/time done by the buoyancy forces. Again, it appears in the rigid lid case with the upper limit set to $z=0$. Finally, the term

$$\begin{aligned} \int d\Omega \int_{-H}^{\eta} dz\, \mathcal{F} &= \rho_o \int d\Omega \int_{-H}^{\eta} dz \;\; \vec{u}_h \cdot (F^{\vec{u}} + [\kappa_m (\vec{u}_h)_z]_z) \\ &= \rho_o \int d\Omega \int_{-H}^{\eta} dz \, [\vec{u}_h \cdot F^{\vec{u}} - \kappa_m \, (\vec{u}_h)_z \cdot (\vec{u}_h)_z] \\ &+ \int d\Omega \, \vec{\tau}_{winds} \cdot (\vec{u}_h)_{z=\eta} - \int d\Omega \, \vec{\tau}_{bottom} \cdot (\vec{u}_h)_{z=-H} \end{aligned} \tag{A.29}$$

represents the effects from internal dissipation, wind power, and bottom dissipation. This term appears in the rigid lid budget as well, with the upper limit set to $z = 0$. Note that this result employed the following upper and lower boundary conditions on the horizontal currents

$$\rho_o \kappa_m (u_z, v_z)_{z=z_{up}} \equiv (\tau^\lambda, \tau^\phi)_{wind} \tag{A.30}$$

$$\rho_o \kappa_m (u_z, v_z)_{z=-H} \equiv (\tau^\lambda, \tau^\phi)_{bottom}, \tag{A.31}$$

with $\vec{\tau}_{wind}$ the wind stress ($dyne/cm^2$), $\vec{\tau}_{bottom}$ the bottom stress ($dyne/cm^2$), and the horizontal currents dotted into each of these stresses taken as the surface and bottom currents, respectively. It is interesting to note that the bottom topography explicitly appears in the kinetic energy budget only through the bottom stress term.

A.1.4.3 Budget for the averaged kinetic energy density within a column

The previous discussion considered the budget for the total kinetic energy within a vertical column of the ocean. It is useful to also consider the budget for the averaged kinetic energy density over this column. For this purpose, define the average over the vertical column of any quantity

$$\langle \alpha \rangle \equiv V^{-1} \int d\Omega \int_{-H}^{\eta} dz\, \alpha, \tag{A.32}$$

where the volume of the vertical column is given by

$$V = \int d\Omega \int_{-H}^{\eta} dz. \tag{A.33}$$

For the free surface, this volume is time dependent

$$V_t = \int d\Omega\, \eta_t, \tag{A.34}$$

unless the volume is for the global domain, in which case it is constant. For the rigid lid, all cell volumes are constant.

The time tendency of the averaged kinetic energy density is given by

$$\partial_t \langle e \rangle = -\langle e \rangle\, \partial_t \ln V + V^{-1}\, \partial_t \left(\int d\Omega \int_{-H}^{\eta} dz\, e \right). \tag{A.35}$$

Substituting the result for the total kinetic energy budget from equation (A.24), yields the following budget for the averaged kinetic energy density over a full vertical column

$$\begin{aligned} \partial_t \langle e \rangle &= -(\langle e \rangle / V) \int d\Omega\, \eta_t + V^{-1} \int d\Omega\, \left(e\, Q_w + p\, \nabla_h \cdot \vec{U} \right) \Big|_{z=\eta} \\ &- V^{-1} \int d\Omega\, \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \vec{u}_h\, (p + e) \right) \\ &- \langle w\, \rho\, g \rangle + \rho_o \langle \vec{u}_h \cdot F^{\vec{u}} - \kappa_m\, (\vec{u}_h)_z \cdot (\vec{u}_h)_z \rangle \\ &+ V^{-1} \int d\Omega\, [\vec{\tau}_{winds} \cdot (\vec{u}_h)_{z=\eta} - \vec{\tau}_{bottom} \cdot (\vec{u}_h)_{z=-H}]. \end{aligned} \tag{A.36}$$

For columns which have a positive area integrated free surface height tendency, the first term acts to reduce the kinetic energy density in the volume due to the increasing volume. The converse holds for negative area integrated free surface height tendencies. The remaining terms have the same interpretation as in the budget for the total kinetic energy. Again, all the terms in the first line vanish for the rigid lid case regardless of the budget's domain.

A.1.4.4 Budget for the globally averaged kinetic energy density

If the budget is taken over the full model domain, then $\int d\Omega\ \eta_t$ vanishes whenever the fresh water budget is closed. The reason is that the total volume of ocean water is constant when assuming an incompressible fluid (see Section 4.6). Additionally, the horizontal convergence term in the second line vanishes for a budget over the full model domain. In summary, the global budget for the kinetic energy density takes the form

$$\begin{aligned}
\partial_t \langle e \rangle_{gbl} &= V^{-1} \int d\Omega \left(e\, Q_w + p\, \nabla_h \cdot \vec{U} \right) \Big|_{z=\eta} \\
&+ V^{-1} \int d\Omega\, [\vec{\tau}_{winds} \cdot (\vec{u}_h)_{z=\eta} - \vec{\tau}_{bottom} \cdot (\vec{u}_h)_{z=-H}]. \\
&- \langle w\, \rho\, g \rangle + \rho_o \langle \vec{u}_h \cdot F^{\vec{u}} - \kappa_m\, (\vec{u}_h)_z \cdot (\vec{u}_h)_z \rangle
\end{aligned} \tag{A.37}$$

A.1.5 External mode kinetic energy budget

The time tendency for the external mode's kinetic energy density is given by

$$\partial_t\, e_{ext} = \rho_o\, (\overline{u}\, \partial_t\, \overline{u} + \overline{v}\, \partial_t\, \overline{v}). \tag{A.38}$$

The time derivative of the vertically integrated zonal velocity is given by

$$\begin{aligned}
\partial_t\, \overline{u} &= \partial_t \left(\frac{1}{H+\eta} \int_{-H}^{\eta} dz\, u \right) \\
&= \frac{\eta_t\, [u(\eta) - \overline{u}]}{H+\eta} + \frac{1}{H+\eta} \int_{-H}^{\eta} dz\, u_t \\
&= \frac{\eta_t\, [u(\eta) - \overline{u}]}{H+\eta} + \overline{u_t}.
\end{aligned} \tag{A.39}$$

A similar relation holds for the meridional velocity. Therefore, the time tendency for the external mode's kinetic energy density takes the form

$$\partial_t\, e_{ext} = \left(\overline{\vec{u}} \cdot \vec{u}_h(\eta) - \overline{\vec{u}} \cdot \overline{\vec{u}} \right) \frac{\rho_o\, \eta_t}{H+\eta} + \rho_o \left(\overline{\overline{u}\, u_t} + \overline{\overline{v}\, v_t} \right). \tag{A.40}$$

The first terms vanish for the rigid lid approximation.

A.1.5.1 Partitioning the budget into physical processes

It is useful to decompose the terms $\overline{\overline{u}\, u_t} + \overline{\overline{v}\, v_t}$ in order to identify various physical processes contributing to the time tendency. For this purpose, vertically integrate the zonal momentum equation

$$\begin{aligned}
\int_{-H}^{\eta} dz\, u_t &= (-w\, u + \kappa_m\, u_z)|_{-H}^{\eta} - \int_{-H}^{\eta} dz\, \nabla_h \cdot (u\, \vec{u}_h) \\
&+ \int_{-H}^{\eta} dz \left[v \left(f + \frac{u \tan\phi}{a} \right) - \frac{p_\lambda}{a\, \rho_o \cos\phi} + F^u \right].
\end{aligned} \tag{A.41}$$

The surface and bottom kinematic boundary conditions (4.29) and (4.24), along with the free surface height equation (7.18), bring the first part of the surface term to the form

$$\begin{aligned}
-w(\eta)\, u(\eta) + w(-H)\, u(-H) &= -u(\eta)\, (\eta_t + \vec{u}_h(\eta) \cdot \nabla_h \eta - Q_w) - u(-H)\, \vec{u}_h(-H) \cdot \nabla_h H \\
&= u(\eta)\, \nabla_h \cdot \vec{U} - u(\eta)\, \vec{u}_h(\eta) \cdot \nabla_h \eta - u(-H)\, \vec{u}_h(-H) \cdot \nabla_h H
\end{aligned} \tag{A.42}$$

These results bring the vertically integrated zonal velocity equation to the form

$$\begin{aligned}
\int_{-H}^{\eta} dz\, u_t &= (H+\eta)\,\overline{u_t} \\
&= u(\eta)\,\nabla_h \cdot \vec{U} + (\kappa_m\, u_z)|_{-H}^{\eta} - \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, u\, \vec{u}_h \right) \\
&+ \int_{-H}^{\eta} dz\, \left[v \left(f + \frac{u \tan\phi}{a} \right) - \frac{p_\lambda}{a\, \rho_o \cos\phi} + F^u \right] . \qquad \text{(A.43)}
\end{aligned}$$

The term $u(\eta)\,\nabla_h \cdot \vec{U}$ is not present in the rigid lid approximation. Otherwise, this result is valid for the rigid lid, with the upper boundary at $z = 0$ rather than $z = \eta$. Similar manipulations yields the vertically integrated meridional velocity equation

$$\begin{aligned}
\int_{-H}^{\eta} dz\, v_t &= (H+\eta)\,\overline{v_t} \\
&= v(\eta)\,\nabla_h \cdot \vec{U} + (\kappa_m\, v_z)|_{-H}^{\eta} - \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, v\, \vec{u}_h \right) \\
&+ \int_{-H}^{\eta} dz\, \left[-u \left(f + \frac{u \tan\phi}{a} \right) - \frac{p_\phi}{a\, \rho_o} + F^v \right] . \qquad \text{(A.44)}
\end{aligned}$$

Adding these results, multiplied by the external mode velocity and ρ_o, yields

$$\begin{aligned}
(H+\eta)\,\partial_t\, e_{ext} &= \rho_o\, \eta_t \left(\overline{\vec{u}} \cdot \vec{u}_h(\eta) - \overline{\vec{u}} \cdot \overline{\vec{u}} \right) + \rho_o\, (H+\eta)\, [\overline{u\, u_t} + \overline{v\, v_t}] \\
&= -2\, \eta_t\, e_{ext} + \overline{\vec{u}} \cdot (\rho_o\, \vec{u}_h(\eta)\, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom}) \\
&- \rho_o \left(\overline{u}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, u\, \vec{u}_h + \overline{v}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, v\, \vec{u}_h \right) \\
&+ \overline{\vec{u}} \cdot \left(\int_{-H}^{\eta} dz\, \left(u\, \vec{u}_h \wedge \hat{z} \right) (\rho_o/a) \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right) . \qquad \text{(A.45)}
\end{aligned}$$

Notice how the Coriolis term dropped out. Also, equation (7.18) for the free surface height was used to introduce the fresh water flux term. For subsequent development, it is useful to massage this result in order to extract a convergence from the middle term

$$\begin{aligned}
-\left(\overline{u}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, u\, \vec{u}_h + \overline{v}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, v\, \vec{u}_h \right) &= \int_{-H}^{\eta} dz\, \vec{u}_h \cdot (u\, \nabla_h \overline{u} + v\, \nabla_h \overline{v}) \\
&- \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \vec{u}_h\, (\overline{\vec{u}} \cdot \vec{u}_h) \right) . \qquad \text{(A.46)}
\end{aligned}$$

These results lead to the following expression for the time tendency of the external mode's kinetic energy density, weighted by the total depth,

$$\begin{aligned}
(H+\eta)\,\partial_t\, e_{ext} &= -2\, e_{ext}\, \eta_t + \overline{\vec{u}} \cdot (\rho_o\, \vec{u}_h(\eta)\, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom}) \\
&+ \rho_o \int_{-H}^{\eta} dz\, \vec{u}_h \cdot (u\, \nabla_h \overline{u} + v\, \nabla_h \overline{v}) - \rho_o \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \vec{u}_h\, (\overline{\vec{u}} \cdot \vec{u}_h) \right) \\
&+ \overline{\vec{u}} \cdot \left(\int_{-H}^{\eta} dz\, \left(u\, \vec{u}_h \wedge \hat{z} \right) (\rho_o/a) \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right) . \qquad \text{(A.47)}
\end{aligned}$$

A.1.5.2 Basic interpretation of the terms in the budget

The first term

$$-2\, e_{ext}\, \eta_t \tag{A.48}$$

accounts for a reduction in energy density when the free surface height increases, which corresponds to an increase in volume of the vertical column. The second term

$$\rho_o\, \overline{\vec{u}} \cdot \vec{u}_h(\eta)\, Q_w + \overline{\vec{u}} \cdot (\vec{\tau}_{winds} - \vec{\tau}_{bottom}) \tag{A.49}$$

represents the combined effects from the fresh water flux, wind stress, and bottom stress. Note that the fresh water flux term is absent in the rigid lid approximation. The third term, written in the original form

$$-\rho_o \left(\overline{u}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, u\, \vec{u}_h + \overline{v}\, \nabla_h \cdot \int_{-H}^{\eta} dz\, v\, \vec{u}_h \right), \tag{A.50}$$

represents a nonlinear coupling between the external mode and the vertically integrated convergence of the horizontal momentum advective flux. The last term

$$\overline{\vec{u}} \cdot \left(\int_{-H}^{\eta} dz\ (\rho_o/a)\, (u\, \vec{u}_h \wedge \hat{z})\, \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right) \tag{A.51}$$

represents the combined effects from the curvature of the sphere, the work done by the vertically integrated horizontal pressure gradients, and the effects of vertically integrated horizontal friction.

A.1.5.3 Budget for the global volume averaged external mode energy density

Now that the time tendency for the external mode's kinetic energy density has been established, it is necessary to consider the tendency for the external mode's energy density averaged over some finite volume extending from the ocean surface to the bottom

$$\begin{aligned}
\partial_t \langle e_{ext} \rangle &= \partial_t \left(V^{-1} \int d\Omega \int_{-H}^{\eta} dz\, e_{ext} \right) \\
&= -\langle e_{ext} \rangle\, \partial_t \ln V + V^{-1} \int d\Omega\, e_{ext}\, \eta_t + V^{-1} \int d\Omega\, \partial_t\, e_{ext}\, (H + \eta) \\
&= V^{-1} \int d\Omega\, \eta_t\, (e_{ext} - \langle e_{ext} \rangle) + V^{-1} \int d\Omega\, (H + \eta)\, \partial_t\, e_{ext}.
\end{aligned} \tag{A.52}$$

The results from equation (A.47) yields the budget for the globally averaged external mode kinetic energy density

$$\begin{aligned}
\partial_t \langle e_{ext} \rangle_{gbl} &= V^{-1} \int d\Omega\, \left[-\eta_t\, e_{ext} + \overline{\vec{u}} \cdot (\rho_o\, \vec{u}_h(\eta)\, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom}) \right] \\
&+ \rho_o\, V^{-1} \int d\Omega\, \int_{-H}^{\eta} dz\, \vec{u}_h \cdot (u\, \nabla_h \overline{u} + v\, \nabla_h \overline{v}) \\
&+ V^{-1} \int d\Omega\, \overline{\vec{u}} \cdot \left(\int_{-H}^{\eta} dz\ (u\, \vec{u}_h \wedge \hat{z})\, (\rho_o/a)\, \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right) \\
&= V^{-1} \int d\Omega\, \left[-\eta_t\, e_{ext} + \overline{\vec{u}} \cdot (\rho_o\, \vec{u}_h(\eta)\, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom}) \right] \\
&+ \rho_o\, \langle \vec{u}_h \cdot (u\, \nabla_h \overline{u} + v\, \nabla_h \overline{v}) \rangle + \left\langle \overline{\vec{u}} \cdot \left[(u\, \vec{u}_h \wedge \hat{z})\, (\rho_o/a)\, \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right] \right\rangle
\end{aligned} \tag{A.53}$$

Note that $\int d\Omega\, \eta_t = 0$ was used to reach this result, and the total convergence terms were dropped due to the no-normal flow boundary condition. The basic interpretation of these terms has been discussed already. Again, the only surface terms which remain for the rigid lid are the wind and bottom stresses.

A.1.6 Internal mode global kinetic energy density budget

In this section, only the globally averaged budget of the internal mode kinetic energy is presented. For this purpose, the previous two budgets can be combined to form the domain averaged internal mode kinetic energy density tendency

$$\partial_t \langle e_{int} \rangle = \partial_t \langle e \rangle - \partial_t \langle e_{ext} \rangle. \tag{A.54}$$

A.1.6.1 Comparing the external mode and full energy density budgets

For ease in discussing the manipulations, recall the results for the global averaged kinetic energy density

$$\begin{aligned} \partial_t \langle e \rangle_{gbl} &= V^{-1} \int d\Omega \left(e\, Q_w + p\, \nabla_h \cdot \vec{U} \right) \Big|_{z=\eta} \\ &+ V^{-1} \int d\Omega \left[\vec{\tau}_{winds} \cdot (\vec{u}_h)_{z=\eta} - \vec{\tau}_{bottom} \cdot (\vec{u}_h)_{z=-H} \right] \\ &- \langle w\, \rho\, g \rangle + \rho_o \langle \vec{u}_h \cdot F^{\vec{u}} - \kappa_m (\vec{u}_h)_z \cdot (\vec{u}_h)_z \rangle. \end{aligned} \tag{A.55}$$

and the global averaged external mode kinetic energy density

$$\begin{aligned} \partial_t \langle e_{ext} \rangle_{gbl} &= V^{-1} \int d\Omega \left[-\eta_t\, e_{ext} + \overline{\vec{u}} \cdot (\rho_o\, \vec{u}_h(\eta)\, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom}) \right] \\ &+ \rho_o \langle \vec{u}_h \cdot (u\, \nabla_h \overline{u} + v\, \nabla_h \overline{v}) \rangle + \left\langle \overline{\vec{u}} \cdot \left[(u\, \vec{u}_h \wedge \hat{z})\, (\rho_o / a)\, \tan\phi - \nabla_h p + \rho_o\, F^{\vec{u}} \right] \right\rangle \end{aligned} \tag{A.56}$$

At this point, it is interesting to directly compare these budgets before looking at the internal mode's budget. First, notice how all the nonlinear advection terms and spherical terms drop out from the budget for the domain averaged kinetic energy density, whereas such terms are present in the external mode's budget. Next, for the domain averaged kinetic energy, notice how the effects from pressure reduce to just the surface pressure acting on the divergence of the vertically integrated flow. For the external mode, pressure appears in a domain integrated sense. A slightly more direct comparison of the pressure effects can be rendered through massaging the external mode's pressure contribution

$$\begin{aligned} -\overline{\vec{u}} \cdot \int_{-H}^{\eta} dz\, \nabla_h p &= -\int_{-H}^{\eta} dz\, \overline{\vec{u}} \cdot \nabla_h p \\ &= -\int_{-H}^{\eta} dz\, [\nabla_h \cdot (\overline{\vec{u}}\, p) - p\, \nabla_h \cdot \overline{\vec{u}}] \\ &= \overline{\vec{u}} \cdot [p(\eta)\, \nabla_h \eta + p(-H)\, \nabla_h H] + \nabla_h \cdot \overline{\vec{u}} \int_{-H}^{\eta} dz\, p \\ &- \nabla_h \cdot \left(\int_{-H}^{\eta} dz\, \overline{\vec{u}}\, p \right). \end{aligned} \tag{A.57}$$

The convergence term drops out for the domain averaged budget through either the no-normal flow boundary condition for closed domains, or periodicity for channels. Hence, the external

mode's global kinetic energy density budget can be written in the form

$$\begin{aligned}
\partial_t \langle e_{ext} \rangle_{gbl} &= -V^{-1} \int d\Omega \, \eta_t \, e_{ext} \\
&+ V^{-1} \int d\Omega \, \overline{\vec{u}} \cdot \left(\rho_o \, \vec{u}_h(\eta) \, Q_w + \vec{\tau}_{winds} - \vec{\tau}_{bottom} + p(\eta) \, \nabla_h \eta + p(-H) \, \nabla_h H \right) \\
&+ \left\langle \rho_o \, \vec{u}_h \cdot (u \, \nabla_h \overline{u} + v \, \nabla_h \overline{v}) + \overline{\vec{u}} \cdot \left[(u \, \vec{u}_h \wedge \hat{z}) \, (\rho_o / a) \, \tan \phi + \rho_o \, F^{\vec{u}} \right] \right\rangle \\
&+ \langle p \, \nabla_h \cdot \overline{\vec{u}} \rangle . \qquad \text{(A.58)}
\end{aligned}$$

A.1.6.2 Budget for the internal mode's global averaged kinetic energy density

The previous results combine to yield the budget for the internal mode's global averaged kinetic energy density

$$\begin{aligned}
\partial_t \langle e_{int} \rangle_{gbl} &= V^{-1} \int d\Omega \, \left[\eta_t \, e_{ext} + \rho_o \, \vec{u}_h(\eta) \cdot \left(\vec{u}_h(\eta)/2 - \overline{\vec{u}} \right) \, Q_w \right] \\
&+ V^{-1} \int d\Omega \, \left(\vec{\tau}_{winds} \cdot (\vec{u}_h(\eta) - \overline{\vec{u}}) - \vec{\tau}_{bottom} \cdot (\vec{u}_h(-H) - \overline{\vec{u}}) \right) \\
&+ V^{-1} \int d\Omega \, \left(p(\eta) \, \nabla \cdot \vec{U} - \overline{\vec{u}} \cdot [p(\eta) \, \nabla_h \eta + p(-H) \, \nabla_h H] \right) \\
&- \left\langle \rho_o \, \vec{u}_h \cdot (u \, \nabla_h \overline{u} + v \, \nabla_h \overline{v}) + \overline{\vec{u}} \cdot \left[(u \, \vec{u}_h \wedge \hat{z}) \, (\rho_o / a) \, \tan \phi + \rho_o \, F^{\vec{u}} \right] \right\rangle \\
&+ \langle p \, \nabla_h \cdot \overline{\vec{u}} \rangle - \langle w \, \rho \, g \rangle + \rho_o \langle \vec{u}_h \cdot F^{\vec{u}} - \kappa_m \, (\vec{u}_h)_z \cdot (\vec{u}_h)_z \rangle . \qquad \text{(A.59)}
\end{aligned}$$

Each of the surface terms in the first line vanishes for the rigid lid. Only the work done by the bottom pressure acting on sloped topography survives from the third line with a rigid lid. The remaining terms are the same for the free surface and the rigid lid.

A.1.7 Concerning the diagnostic option *energy_analysis*

The discussion of the kinetic energy balance given in the previous sections allows for a partitioning of the terms determining the kinetic energy evolution into various physical processes. In MOM, it is also useful to determine such budgets for certain physical and numerical reasons. For example, by computing separately both sides to the various kinetic energy budgets, it is possible to determine the consistency of the various discretizations of physical processes. This calculation is provided by the diagnostic option *energy_analysis* (Section 40.4). This diagnostic has proven an indispensible tool in formulating various new algorithms in MOM. The following discussion provides the basic approach used to determine the internal and external mode energy budgets as implemented in option *energy_analysis*. Note that some of the manipulations presented in earlier sections can be generalized quite simply to those required for this section. However, for the sake of completeness, the manipulations are presented here in their fundamental form.

A.1.7.1 Splitting of the energy density

In the following, brackets will always denote a global volume average

$$\langle \alpha \rangle = V_{globe}^{-1} \int^{globe} d\Omega \int_{-H}^{\eta} dz \, \alpha , \qquad \text{(A.60)}$$

where V_{globe} is the constant global volume of the ocean water. The "globe" subscript is dropped in further equations in this section. Recall that V_{globe} is constant, even with the free surface, since the ocean is incompressible and it is assumed that the budget for the surface fresh water forcing is closed.

Because

$$\langle \widehat{u}\,\overline{u}\rangle = \langle \widehat{v}\,\overline{v}\rangle = 0, \tag{A.61}$$

the volume averaged kinetic energy density splits into two terms

$$\langle e\rangle = \langle e_{ext}\rangle + \langle e_{int}\rangle, \tag{A.62}$$

where

$$\langle e_{ext}\rangle = (\rho_o/2)\langle \overline{u}\,\overline{u} + \overline{v}\,\overline{v}\rangle \tag{A.63}$$

and

$$\langle e_{int}\rangle = (\rho_o/2)\langle \widehat{u}\,\widehat{u} + \widehat{v}\,\widehat{v}\rangle. \tag{A.64}$$

Now the time tendency of the global averaged external mode kinetic energy density is given by

$$\begin{aligned} \partial_t \langle e_{ext}\rangle &= \rho_o \langle \overline{u}\,\partial_t\,\overline{u} + \overline{v}\,\partial_t\,\overline{v}\rangle \\ &= \rho_o \langle \overline{u}\,\partial_t\,(u-\widehat{u}) + \overline{v}\,\partial_t\,(v-\widehat{v})\rangle. \end{aligned} \tag{A.65}$$

Similarily, the time tendency of the global averaged internal mode kinetic energy density is given by

$$\begin{aligned} \partial_t \langle e_{int}\rangle &= \rho_o \langle \widehat{u}\,\partial_t\,\widehat{u} + \widehat{v}\,\partial_t\,\widehat{v}\rangle \\ &= \rho_o \langle \widehat{u}\,\partial_t\,(u-\overline{u}) + \widehat{v}\,\partial_t\,(v-\overline{v})\rangle. \end{aligned} \tag{A.66}$$

A.1.7.2 A useful result

These expressions can be reduced by using the following result

$$\begin{aligned} \int_{-H}^{\eta} dz\,\widehat{u}\,\partial_t\,\overline{u} &= \int_{-H}^{\eta} dz\,[\partial_t\,(\widehat{u}\,\overline{u}) - \overline{u}\,\partial_t\,\widehat{u}] \\ &= \partial_t\left(\int_{-H}^{\eta} dz\,\widehat{u}\,\overline{u}\right) - \widehat{u}(\eta)\,\overline{u}\,\eta_t - \overline{u}\,\partial_t\left(\int_{-H}^{\eta}\widehat{u}\right) + \overline{u}\,\widehat{u}(\eta)\,\eta_t \\ &= \partial_t\left(\overline{u}\int_{-H}^{\eta} dz\,\widehat{u}\right) - \overline{u}\,\partial_t\left(\int_{-H}^{\eta}\widehat{u}\right) \\ &= 0, \end{aligned} \tag{A.67}$$

where

$$\int_{-H}^{\eta} dz\,\widehat{u} = 0 \tag{A.68}$$

has been used. This result says that there is zero depth integrated correlation between a component of the internal mode velocity, and time tendency for the corresponding component of the external mode velocity tendency. This result of course also holds for the rigid lid. It follows that

$$\langle \widehat{u}\,\partial_t\,\overline{u}\rangle = \langle \widehat{v}\,\partial_t\,\overline{v}\rangle = 0. \tag{A.69}$$

A.1.7.3 Algorithm for the internal mode

The result (A.69) indicates that the internal mode's globally averaged kinetic energy density has a time tendency given by

$$\partial_t \langle e_{int} \rangle = \rho_o \langle \widehat{u}\, \partial_t u + \widehat{v}\, \partial_t v \rangle. \tag{A.70}$$

The terms contributing to this tendency are given in the budget equation (A.59). In the model, $\partial_t \langle e_{int} \rangle$ is directly computed by taking the time tendency for the global averaged internal mode kinetic energy density. The right hand side is then computed through taking the scalar product of the internal mode velocity and the various terms in the horizontal velocity equations (A.2) and (A.3), and then averaging this scalar product over the ocean domain (the result is the right hand side of equation (A.59)). The calculation of $\partial_t \langle e_{int} \rangle$ should agree with the scalar product calculation, to within computer roundoff. Differences indicate a problem with some portion of the numerical discretization in the model. This diagnostic has been quite useful in tracking down problems with discretizations, especially with those concerning the discretization of pressure gradients.

A.1.7.4 Algorithm for the external mode

The result (A.69) allows for the following reduction in the external mode's budget

$$\begin{aligned} \langle \overline{u}\, \partial_t \widehat{u} \rangle &= \langle \partial_t (\overline{u}\, \widehat{u}) \rangle \\ &= \partial_t \langle \overline{u}\, \widehat{u} \rangle - V^{-1} \int d\Omega\, \eta_t\, \overline{u}\, \widehat{u}(\eta) \\ &= -V^{-1} \int d\Omega\, \eta_t\, \overline{u}\, \widehat{u}(\eta), \end{aligned} \tag{A.71}$$

where $\langle \overline{u}\, \widehat{u} \rangle = 0$ was used. Hence, the time tendency for the globally averaged external mode kinetic energy density can be written in the form

$$\partial_t \langle e_{ext} \rangle = \rho_o \langle \overline{u}\, \partial_t u + \overline{v}\, \partial_t v \rangle + V^{-1} \int d\Omega\, \eta_t\, (\widehat{u}(\eta)\overline{u} + \widehat{v}(\eta)\overline{v}). \tag{A.72}$$

This result is basically the same as that derived in Section A.1.2. It says that the time tendency for the domain averaged external mode kinetic energy density is given by two general terms. The first term

$$\rho_o \langle \overline{u}\, \partial_t u + \overline{v}\, \partial_t v \rangle \tag{A.73}$$

arises from correlations between the external mode velocity and the time tendency for the full velocity field. This term is present for the rigid lid and free surface. As for the internal mode, for purposes of budget analysis, this term is computed in the diagnostic option *energy_analysis* by taking the scalar product of the external mode's velocity and the right hand side of the horizontal velocity equations (A.2) and (A.3). This scalar product is then averaged over the full model domain. The second term

$$V^{-1} \int d\Omega\, \eta_t\, [\widehat{u}(\eta)\overline{u} + \widehat{v}(\eta)\overline{v}] \tag{A.74}$$

arises from the time tendency of the surface height and the projection of the internal mode's surface velocity onto the external mode's velocity. This term is absent in the rigid lid. It is computed directly in the option *energy_analysis* when the free surface option is enabled.

A.1.7.5 Special case of a flat bottom and rigid lid

For a flat bottom, rigid lid ocean, which is a common idealized model configuration, the domain averaged pressure forces can do no work on the external mode. To prove this property, it is useful to start with the identity

$$\begin{aligned} \nabla_h \overline{p} &= \nabla_h \left(H^{-1} \int_{-H}^{0} dz\, p \right) \\ &= \overline{\nabla p} + (p_{bottom} - \overline{p})\, \nabla \ln H, \end{aligned} \tag{A.75}$$

where p_{bottom} is the pressure at $z = -H$. The second term vanishes for a flat bottom domain ($\nabla H = 0$) and so the domain averaged pressure work on the external mode becomes

$$- < \overline{\vec{u}} \cdot \overline{\nabla_h p} > = - < \overline{\vec{u}} \cdot \nabla_h \overline{p} > . \tag{A.76}$$

The continuity equation implies $\nabla_h \cdot \vec{u} + w_z = 0$. Hence, $\nabla_h \cdot \overline{\vec{u}} = w(z = -H) - w(z = z_{up})$, where $w(z = z_{up}) = 0$ for a rigid lid and $w(z = -H) = 0$ for a flat bottom. Therefore, $\overline{\vec{u}} \cdot \nabla_h \overline{p} = \nabla_h \cdot (\overline{\vec{u}}\, \overline{p})$, which vanishes when averaged over a closed domain due to the no-normal flow boundary condition place on velocity.

A.2 Energetics on the discrete grid

The purpose of this section is to present a discussion of the energetics for the discrete equations.

A.2.1 Conservative advection: part I

MOM always implements the advection of momentum using centered differences. Tracers can likewise be advected, but there are alternatives as discussed in Chapter 32. To illustrate the generic conservative properties of centered differences, let q be any three dimensional prognostic quantity at the centers of grid cells $q_1, q_2, q_3, \cdots q_N$ with volumes $\alpha_1, \alpha_2, \alpha_3, \cdots \alpha_N$ respectively. What will be shown is that the finite difference analogue of the advection equation

$$\partial_t(q) + \nabla \cdot (\vec{v} q) = 0 \tag{A.77}$$

keeps the two quantities

$$\sum_{n=1}^{N} \alpha_n q_n \qquad \sum_{n=1}^{N} \alpha_n q_n^2 \tag{A.78}$$

constant in time. Using Stokes' theorem to change volume integrals into surface integrals, the finite difference form of the volume integral of Equation (A.77) is

$$\sum_{n=1}^{N} (\alpha_n \frac{dq_n}{dt} + \sum_{i=1}^{6} A_n^i V_n^i \overline{q}_n^i) = 0 \tag{A.79}$$

with

$$\overline{q}_n^i = \frac{(q_n + q_n^i)}{2} \tag{A.80}$$

where q_n is the value in the nth cell, A_n^i is the area of the ith face of that cell, V_n^i is the velocity normal to the ith face, and $\overline{q}_n^i$ is the average value of q on the ith face with q_n^i being the value in the other cell that shares that face. Using Equations (A.79) and (A.80), the time change of the volume integral of q_n when integrated over all cells can be written as

$$\frac{d}{dt}\sum_{n=1}^{N}\alpha_n q_n = \sum_{n=1}^{N}\alpha_n\frac{dq_n}{dt} = -\sum_{n=1}^{N}\sum_{i=1}^{6}A_n^i V_n^i\frac{(q_n+q_n^i)}{2} = 0. \tag{A.81}$$

To obtain this result, the normal velocity V_n^i must be zero at land boundaries. Note that V_n^i is anti-symmetric on the common face between adjacent cells due to the change in sign of the normal vector, whereas the average $\overline{q}_n^i$ is symmetric with respect to adjacent cells. This result indicates that integrals of first moments are conserved by Equation (A.80). It should be further noted that constructing $\overline{q}_n^i$ by any linear interpolation of q_n and q_n^i will also conserve first moments.

Using similar techniques, the time change of the volume integral of q^2 is given by:

$$\frac{d}{dt}\sum_{n=1}^{N}\alpha_n q_n^2 = 2\sum_{n=1}^{N}\alpha_n q_n\frac{dq_n}{dt} = -\sum_{n=1}^{N}q_n^2(\sum_{i=1}^{6}A_n^i V_n^i) - \sum_{n=1}^{N}\sum_{i=1}^{6}A_n^i V_n^i q_n q_n^i = 0. \tag{A.82}$$

The third term in Equation (A.82) vanishes because the fluid is incompressible and volume conservation applies over each cell. The fourth term vanishes because $q_n q_n^i$ is symmetric and V_n^i is anti-symmetric for pairs of adjacent cells (again due to the change in sign of the normal vector) and $V_n^i = 0$ on all land boundaries. It should be noted here that, in particular, constructing $\overline{q}_n^i$ as $\beta \cdot q_n + \gamma \cdot q_n^i$ where $\beta \neq \gamma \neq 1/2$ will <u>not</u> conserve second moments because neither the third nor fourth term will vanish. MOM uses Equation (A.80) to construct averaged quantities for the second order centered advective scheme. For this case, it conserves first and second moments. In general, advective schemes other than the second order centered scheme do not conserve second moments.

A.2.2 Conservative advection: part II

It is useful to provide a more explicit derivation using spherical coordinates and non-uniform grids as employed by MOM. Similar steps are required to show that both the kinetic energy and tracer variance is conserved using centered advection. For the kinetic energy derivation, let $q_n = u_{i,k,j,n,\tau}$ and construct the volume integral of the work done by the advection terms using operators defined in Section 22.9.5

$$\begin{aligned} A \;=\; & \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2} u_{i,k,j,n,\tau}\cos\phi^U_{jrow} dxu_i\, dyu_{jrow}\, dzt_k \times \\ & \left[ADV_Ux_{i,k,j} + ADV_Uy_{i,k,j} + ADV_Uz_{i,k,j} + ADV_metric_{i,k,j,n}\right]. \end{aligned} \tag{A.83}$$

Expansion of the advection operators yields the work done in terms of advective fluxes on all faces of the U cells

$$\begin{aligned} A \;=\; & \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2} u_{i,k,j,n,\tau}\cos\phi^U_{jrow} dxu_i\, dyu_{jrow}\, dzt_k \times \\ & \left[\frac{adv_fe_{i,k,j} - adv_fe_{i-1,k,j}}{2\cos\phi^U_{jrow} dxu_i} + \frac{adv_fn_{i,k,j} - adv_fn_{i,k,j-1}}{2\cos\phi^U_{jrow} dyu_{jrow}}\right. \end{aligned}$$

$$+ \frac{adv_fb_{i,k-1,j} - adv_fb_{i,k,j}}{2dzt_k} \mp \frac{\tan\phi^U_{jrow}}{radius} u_{i,k,j,1,\tau} \cdot u_{i,k,j,3-n,\tau} \Bigg], \tag{A.84}$$

where the minus sign on the metric term is used when $n = 1$ and the plus sign is used when $n = 2$. The metric term trivially vanishes since

$$\begin{aligned} \mp & \sum_{n=1}^{2} u_{i,k,j,n,\tau}\, u_{i,k,j,1,\tau}\, u_{i,k,j,3-n,\tau} \\ = & -u^2_{i,k,j,1,\tau}\, u_{i,k,j,2,\tau} + u_{i,k,j,2,\tau}\, u_{i,k,j,1,\tau}\, u_{i,k,j,1,\tau} \\ = & \; 0. \end{aligned} \tag{A.85}$$

For the remaining terms, expand the advective fluxes further to yield

$$\begin{aligned} A \;=\; & \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{n=1}^{2} u_{i,k,j,n,\tau} \Bigg[\\ & \left(adv_veu_{i,k,j} \frac{u_{i,k,j,n,\tau} + u_{i+1,k,j,n,\tau}}{2} - adv_veu_{i-1,k,j} \frac{u_{i,k,j,n,\tau} + u_{i-1,k,j,n,\tau}}{2} \right) dyu_{jrow}\, dzt_k \\ + & \left(adv_vnu_{i,k,j} \frac{u_{i,k,j,n,\tau} + u_{i,k,j+1,n,\tau}}{2} - adv_vnu_{i,k,j-1} \frac{u_{i,k,j,n,\tau} + u_{i,k,j-1,n,\tau}}{2} \right) dxu_i\, dzt_k \\ + & \left(adv_vbu_{i,k-1,j} \frac{u_{i,k-1,j,n,\tau} + u_{i,k,j,n,\tau}}{2} - adv_vbu_{i,k,j} \frac{u_{i,k,j,n,\tau} + u_{i,k+1,j,n,\tau}}{2} \right) \cos\phi^U_{jrow} dxu_i dyu_{jrow} \\ & \Bigg] \\ = & \frac{1}{2} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{n=1}^{2} u^2_{i,k,j,n,\tau} \\ & \Bigg[(adv_veu_{i,k,j} - adv_veu_{i-1,k,j}) dyu_{jrow}\, dzt_k \\ + & (adv_vnu_{i,k,j} - adv_vnu_{i,k,j-1}) dxu_i\, dzt_k \\ + & (adv_vbu_{i,k-1,j} - adv_vbu_{i,k,j}) \cos\phi^U_{jrow} dxu_i dyu_{jrow} \Bigg] \\ + & \frac{1}{2} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{n=1}^{2} u_{i,k,j,n,\tau} \\ & \Bigg[(adv_veu_{i,k,j}\, u_{i+1,k,j,n,\tau} - adv_veu_{i-1,k,j}\, u_{i-1,k,j,n,\tau}) dyu_{jrow}\, dzt_k \\ + & (adv_vnu_{i,k,j}\, u_{i,k,j+1,n,\tau} - adv_vnu_{i,k,j-1}\, u_{i,k,j-1,n,\tau}) dxu_i\, dzt_k \\ + & (adv_vbu_{i,k-1,j}\, u_{i,k-1,j,n,\tau} - adv_vbu_{i,k,j}\, u_{i,k+1,j,n,\tau}) \cos\phi^U_{jrow} dxu_i dyu_{jrow} \Bigg]. \end{aligned} \tag{A.86}$$

The first group of terms vanishes identically due to the conservation of volume over each grid cell. The second group also vanishes, so long as there is a zero normal flux at the side boundaries or periodicity. For example, consider the zonal term $n = 1$ and keep only the zonal label

$$\sum_{i=2}^{imt-1} u_i (adv_veu_i\, u_{i+1} - adv_veu_{i-1}\, u_{i-1}) \;=\; \sum_{i=3}^{imt} u_{i-1}\, u_i\, adv_veu_{i-1} - \sum_{i=2}^{imt-1} u_i\, u_{i-1}\, adv_veu_{i-1}$$

$$= \quad -u_1\, u_2\, adv_veu_1 + u_{imt-1}\, u_{imt}\, adv_veu_{imt-1}. \tag{A.87}$$

For solid walls, $u_1 = u_{imt} = 0$. For zonally periodic conditions, $adv_veu_1\, u_1 = adv_veu_{imt-1}\, u_{imt-1}$ and $u_2 = u_{imt}$. The other flux components cancel likewise. Hence, in both the case of solid walls and zonal periodicity, $A = 0$, and so the advection terms do not contribute to the kinetic energy. It should be noted that if $u_{i,k,j,n,\tau}$ in equation (A.83) is replaced by the internal or external mode velocity, $\hat{u}_{i,k,j,n,\tau}$ or $\overline{u}_{i,j,n,\tau}$, then A will not vanish; the full velocity must be used to get the cancelation.

A.2.3 Zero work by the Coriolis force

There is no contribution from integrating the work done by the Coriolis force. The summation is

$$C = \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{n=1}^{2} u_{i,k,j,n,\tau} \cdot 2\Omega \sin \phi^U_{jrow} \cdot u_{i,k,j,3-n,\tau} = 0 \tag{A.88}$$

Again, only use of the full velocity guarantees that the sum will vanish.

A.2.4 Work done by pressure terms

In Section A.1.3, it was shown in the continuous equations that the net change in kinetic energy due to pressure forces equals the net change in energy due to buoyancy. The discrete counterpart of this result is given below using the definition of variables, indices and the relation between *jrow* and *j* as described in Section 14.2.1. In this terminology of MOM, the change in kinetic energy due to pressure forces summed over all ocean U cells is given by

$$-\frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \left(u_{i,k,j,1,\tau} \cdot \frac{1}{\cos \phi^U_{jrow}} \delta_\lambda(\overline{p_{i,k,j}}^{\phi}) + u_{i,k,j,2,\tau} \cdot \delta_\phi(\overline{p_{i,k,j}}^{\lambda}) \right) dvol_{i,k,j} \tag{A.89}$$

where the U-cell volume element $dvol_{i,k,j}$ is

$$dvol_{i,k,j} = dxu_i \cos \phi^U_{jrow}\, dyu_{jrow}\, dzt_k \tag{A.90}$$

and pressure p is defined on T cells. Applying Equation (21.15) to the "i" summation for the first term in Equation (A.89) and similarly to the "jrow" summation for the second term yields

$$\begin{aligned} & \frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \delta_\lambda(u_{i-1,k,j,1,\tau})\, \overline{p_{i,k,j}}^{\phi}\, dxt_i\, dyu_{jrow}\, dzt_k \\ + \quad & \delta_\phi(u_{i,k,j-1,2,\tau} \cos \phi^U_{jrow-1})\, \overline{p_{i,k,j}}^{\lambda}\, dxu_i\, dyt_{jrow}\, dzt_k \end{aligned} \tag{A.91}$$

Applying Equation (21.14) to the "jrow" summation for the first term in Equation (A.91) and to the "i" summation for the second term yields

$$\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\Big(\overline{\delta_\lambda(u_{i-1,k,j-1,1,\tau})\,dyu_{jrow-1}}^{\phi}\,dxt_i\,dzt_k$$
$$+\quad\overline{\delta_\phi(u_{i-1,k,j-1,2,\tau}\cos\phi^U_{jrow-1})\,dxu_{i-1}}^{\lambda}\,dyt_{jrow}\,dzt_k\Big)\;p_{i,k,j} \tag{A.92}$$

Defining the advective velocities on the eastern and northern face of a T-cell as

$$adv_vet_{i,k,j} = \frac{\overline{u_{i,k,j-1,1,\tau}\cdot dyu_{jrow-1}}^{\phi}}{dyt_{jrow}} \tag{A.93}$$

$$adv_vnt_{i,k,j} = \frac{\overline{u_{i-1,k,j,2,\tau}\cdot dxu_{i-1}}^{\lambda}}{dxt_i}\cos\phi^U_{jrow} \tag{A.94}$$

and substituting into Equation (A.92) yields

$$\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\big(\delta_\lambda(adv_vet_{i-1,k,j,1,\tau})\,dyt_{jrow}\,dxt_i\,dzt_k$$
$$+\quad\delta_\phi(adv_vnt_{i,k,j-1,2,\tau})\,dxt_i\,dyt_{jrow}\,dzt_k\,)\;p_{i,k,j} \tag{A.95}$$

Note that the finite difference counterpart of incompressibility, Equation (4.3), for T-cells uses advective velocities defined on the faces of T cells

$$\frac{1}{\cos\phi^T_{jrow}}(\delta_\lambda(adv_vet_{i-1,k,j})+\delta_\phi(adv_vnt_{i,k,j-1}))+\delta_z(adv_vbt_{i,k-1,j})=0 \tag{A.96}$$

Substituting Equation (A.96) into Equation (A.95) yields

$$-\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}p_{i,k,j}\cdot\delta_z(adv_vbt_{i,k-1,j})\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow}dzt_k \tag{A.97}$$

Once again, using Equation (21.15) to re-arrnage the summation on "k" in Equation (A.97) yields

$$\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}adv_vbt_{i,k-1,j}\cdot\delta_z(p_{i,k-1,j})\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow}\,dzw_{k-1} \tag{A.98}$$

Substituting the discrete hydrostatic equation given by Equation (21.42) reduces Equation (A.98) to

$$-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z}\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow}\,dzw_{k-1} \tag{A.99}$$

Equation (A.99) is the discrete counterpart of a result derived for the continuum in equation (A.18). It represents the change in kinetic energy due to horizontal pressure terms. Comparing with Equation (A.102) indicates that the change in kinetic energy due to horizontal pressure forces is compensated by an equal change in energy due to buoyancy effects.

A.2.5 Work done by Buoyancy

In Section A.1.3, it was shown in the continuous equations that the net change in kinetic energy due to pressure forces equals buoyancy effects in the absence of sources and sinks. In what follows, the discrete form of the net change in potential energy is arrived at by summing the horizontal and vertical advection of tracers multiplied by $grav \cdot zt_k/\rho_\circ$ over the entire ocean volume. If it is assumed that density ρ is a linear function of tracers (temperature and salinity) then the net change in potential energy is given by

$$-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} zt_k\left(\frac{1}{\cos\phi^T_{jrow}}\left[\delta_\lambda(adv_vet_{i-1,k,j}\cdot\overline{\rho_{i-1,k,j}}^{\lambda})+\delta_\phi(adv_vnt_{i,k,j-1}\cdot\overline{\rho_{i,k,j-1}}^{\phi})\right]\right.$$
$$\left.+\quad\delta_z(adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z})\right)\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow}\,dzt_k \tag{A.100}$$

Noting that $adv_vet_{i,k,j}$ and $adv_vnt_{i,k,j}$ are zero on land boundaries and summing in the horizontal over all cells for any level k eliminates the horizontal advection terms reducing the kernal to

$$-\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} zt_k\cdot\delta_z(adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z})\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow}\,dzt_k \tag{A.101}$$

Using the vertical equivalent of Equation (21.15) to re-arrange the summation on "k" in the vertical (for a rigid lid only) yields

$$\frac{grav}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} adv_vbt_{i,k-1,j}\cdot\overline{\rho_{i,k-1,j}}^{z}\cdot dzw_{k-1}\,dxt_i\cos\phi^T_{jrow}\,dyt_{jrow} \tag{A.102}$$

which should be compared to Equation (A.99) to show that the change in potential energy is exactly compensated for by the change in kinetic energy from horizontal pressure terms.

Appendix B

Tracer mixing kinematics

In the process of arriving at a new formulation for the isopycnal diffusion scheme in MOM, it was useful to understand the basic kinematical properties of tracer mixing when parameterized by a second order mixing tensor. This appendix serves to document certain of these properties which, although perhaps well known by some, were not found to be readily accessible in the literature. In this appendix, only continuum equations will be discussed, and coordinates are referenced to a frame tangent to the geopotential [i.e., (x, y, z)]. The MOM code uses spherical coordinates and the coordinate transformations are straightforward (e.g., Haltiner and Williams, Chapter 1).

The basic equation to be considered here is the tracer mixing equation

$$(\partial_t + \vec{u} \cdot \nabla)\, T = R(T), \tag{B.1}$$

where the mixing operator $R(T)$ is given in an orthogonal coordinate frame by

$$R(T) = \partial_m (J^{mn} \partial_n T). \tag{B.2}$$

In these expressions, the Einstein summation convention is assumed in which repeated indices (m, n) are summed over the three spatial directions. $\mathbf{J}$ is a second order tensor whose contravariant components are written as J^{mn}. The three-dimensional velocity field $\vec{u}$ is assumed to be divergence-free ($\nabla \cdot \vec{u} = 0$). T represents any tracer such as potential temperature, salinity, or a passive tracer.

In general, this equation represents a parameterization of mixing due to many different processes occurring over a wide range of scales. Therefore, the explicit form for the tensor $\mathbf{J}$ depends on the particular phenomenon being parametrized. Two basic forms of mixing are considered here as distinguished by the symmetry of the tensor. Either the process is purely dissipative, as represented by a symmetric positive semi-definite diffusion tensor, or purely advective, as represented by an anti-symmetric tensor.

B.1 Basic properties

Consider a mixing process governed by a tensor $\mathbf{J}$. This tensor in general contains both symmetric and anti-symmetric components[1] given by

$$J^{mn} = \frac{1}{2}(J^{mn} + J^{nm}) + \frac{1}{2}(J^{mn} - J^{nm}) \equiv K^{mn} + A^{mn}, \tag{B.3}$$

[1]The anti-symmetric component is sometimes called the *skew-symmetric* component in the literature.

where the symmetric part of the tensor is written K^{mn} and the anti-symmetric part as $A^{mn} = -A^{nm}$. For diffusive or dissipative mixing, K^{mn} is symmetric and positive semi-definite. Note that anti-symmetry implies the diagonal terms in A^{mn} vanish. Additionally, anti-symmetry is a frame invariant property of a tensor.

B.1.1 Kinematics of an anti-symmetric tensor

Anti-symmetry introduces constraints on the form of mixing described by anti-symmetric tensors. All these constraints originate from the property that the tensor contraction of a symmetric tensor with an anti-symmetric tensor vanishes. Explicitly, consider a symmetric tensor ($S^{mn} = S^{nm}$) and its contraction (under a flat Euclidean metric) with an anti-symmetric tensor ($A^{mn} = -A^{nm}$).

$$\begin{aligned} S_{mn}A^{mn} &= S^{12}A^{12} + S^{21}A^{21} + S^{13}A^{13} + S^{31}A^{31} + S^{23}A^{23} + S^{32}A^{32} && \text{(B.4)} \\ &= S^{12}A^{12} - S^{12}A^{12} + S^{13}A^{13} - S^{13}A^{13} + S^{23}A^{23} - S^{23}A^{23} && \text{(B.5)} \\ &= 0, && \text{(B.6)} \end{aligned}$$

where the symmetry of S^{mn} and the anti-symmetry of A^{mn} were used. This property implies, for example, that

$$\begin{aligned} A^{mn}\partial_m T \partial_n T &= 0 && \text{(B.7)} \\ A^{mn}\partial_m \partial_n T &= 0 && \text{(B.8)} \\ \partial_m \partial_n A^{mn} &= 0. && \text{(B.9)} \end{aligned}$$

B.1.1.1 Effective advection velocity

Consider the tracer mixing operator constructed from an anti-symmetric tensor

$$R(T)_A = \partial_m (A^{mn}\partial_n T). \tag{B.10}$$

Property (B.8) implies

$$R(T)_A = (\partial_m A^{mn})\partial_n T, \tag{B.11}$$

which allows for the identification of a velocity

$$u^n_A \equiv -\partial_m A^{mn}, \tag{B.12}$$

and which brings the anti-symmetric mixing process into the form of an advection

$$R(T)_A = -\vec{u}_A \cdot \nabla T. \tag{B.13}$$

It is important to note that the advection velocity $\vec{u}_A$ is divergence-free

$$\begin{aligned} \nabla \cdot \vec{u}_A &= \partial_n u^n_A && \text{(B.14)} \\ &= -\partial_n \partial_m A^{mn} && \text{(B.15)} \\ &= 0, && \text{(B.16)} \end{aligned}$$

where the last identity used relation (B.9) above. Therefore, tracer mixing as parameterized with an anti-symmetric tensor

$$\frac{DT}{Dt} \equiv (\partial_t + \vec{u} \cdot \nabla)\, T = R(T)_A \tag{B.17}$$

can be written

$$[\partial_t + (\vec{u} + \vec{u}_A) \cdot \nabla]\, T = 0, \tag{B.18}$$

which allows for the identification of an *effective advective transport velocity* (Plumb and Mahlman 1987)

$$\vec{U} \equiv \vec{u} + \vec{u}_A. \tag{B.19}$$

B.1.1.2 Skew or anti-symmetric flux

Rhines (1986) and Middleton and Loder (1989) discuss the *skew flux* instead of the effective advection velocity. This flux is defined as

$$F_A^m \equiv A^{mn} \partial_n T, \tag{B.20}$$

which allows the skew-symmetric mixing process to be written as

$$(\partial_t + \vec{u} \cdot \nabla) T = \nabla \cdot \vec{F_A}. \tag{B.21}$$

The skew flux is directed orthogonal to the local tracer gradient since

$$\vec{F_A} \cdot \nabla T = F_A^m \partial_m T = A^{mn} \partial_n T \partial_m T = 0. \tag{B.22}$$

B.1.2 Tracer moments

Multiplying the tracer equation

$$(\partial_t + \vec{u} \cdot \nabla) T = \partial_m (J^{mn} \partial_n T) \tag{B.23}$$

by T^{N-1}, where N is some positive integer, yields

$$\begin{aligned}
\frac{1}{N} \partial_t T^N + \frac{1}{N} \nabla \cdot (\vec{u} T^N) &= T^{N-1} \partial_m (J^{mn} \partial_n T) && \text{(B.24)} \\
&= \partial_m (T^{N-1} J^{mn} \partial_n T) - \partial_m T^{N-1} \partial_n T J^{mn} && \text{(B.25)} \\
&= \frac{1}{N} \partial_m (J^{mn} \partial_n T^N) - (N-1) T^{N-2} \partial_m \partial_n J^{mn}. && \text{(B.26)}
\end{aligned}$$

The time tendency is therefore given by

$$\begin{aligned}
\partial_t T^N &= -\nabla \cdot (\vec{u} T^N) + \partial_m (J^{mn} \partial_n T^N) - N(N-1) T^{N-2} \partial_m T \partial_n T J^{mn} && \text{(B.27)} \\
&= \nabla \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) T^{N-2} \partial_m T \partial_n T K^{mn}, && \text{(B.28)}
\end{aligned}$$

where the flux

$$\begin{aligned}
F^m &= J^{mn} \partial_n T && \text{(B.29)} \\
&= K^{mn} \partial_n T + A^{mn} \partial_n T && \text{(B.30)} \\
&\equiv F_K^m + F_A^m && \text{(B.31)}
\end{aligned}$$

was introduced. Note that the above steps assumed a divergence-free velocity field ($\nabla \cdot \vec{u} = 0$) and the identity (B.7) ($A^{mn} \partial_m T \partial_n T = 0$) was employed.

Integrating the previous equation over some domain yields

$$\partial_t \int d\vec{x}\, T^N = \int d\vec{x} \left[\nabla \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) T^{N-2} K^{mn} \partial_m T \partial_n T \right]. \tag{B.32}$$

Stokes' Theorem brings this expression to

$$\partial_t \int d\vec{x}\, T^N = \int dA\, \hat{n} \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) \int d\vec{x}\, T^{N-2} K^{mn} \partial_m T \partial_n T, \tag{B.33}$$

where $\hat{n}$ is the outward normal to the boundary. Assuming the normal velocity vanishes at the domain boundary ($\vec{u} \cdot \hat{n} = 0$) yields for the time tendency of the globally integrated $N'th$ tracer moment

$$\partial_t \int d\vec{x}\, T^N = \int dA\, \hat{n} \cdot (\vec{F}_K + \vec{F}_A)\, T^{N-1} - N(N-1) \int d\vec{x}\, T^{N-2} K^{mn} \partial_m T \partial_n T. \tag{B.34}$$

It is of interest to consider some special cases. First, consider the case with no diffusive mixing, which means there is a zero symmetric component to the tracer mixing tensor ($K^{mn} = F_K^m = 0$). Therefore, all moments of the tracer are conserved when there is no normal skew flux at the boundaries ($\vec{F}_A \cdot \hat{n} = 0$), or equivalently there is zero normal induced velocity ($\vec{u}_A \cdot \hat{n} = 0$) at the boundaries. For example, the skew flux associated with the Gent-McWilliams effective transport velocity satisfies these conditions and so preserves all tracer moments (Gent and McWilliams 1990).

For the case with zero anti-symmetric but nonzero symmetric tracer mixing tensor, the first moment, or the total tracer, is conserved in source free regions where there is no tracer flux normal to the boundary; i.e., if $\hat{n} \cdot \vec{F}_K \equiv \hat{n}_m K^{mn} \partial_n T = 0$ at the domain boundaries. In the absence of sources, the tracer's second moment satisfies the evolution equation

$$\partial_t \int d\vec{x}\, T^2 = -2 \int d\vec{x}\, \partial_m T K^{mn} \partial_n T. \tag{B.35}$$

Therefore, this mixing tensor dissipates the tracer variance [2] over the source-free domain if the tensor K^{mn} is a positive semi-definite tensor (i.e., the right hand side is negative semi-definite if K^{mn} is positive semi-definite). This property of diffusion is fundamental, and is taken as the foundation for the numerical discretization of isopycnal diffusion in MOM. All higher moments of the tracer are also dissipated by diffusion assuming the usual case of a non-negative tracer concentration.

B.2 Horizontal-vertical diffusion

The previous discussion is now specialized to particular forms of mixing used in MOM. The first form is Cartesian or *horizontal-vertical diffusion* in which the principle axes are assumed to be defined by the local tangent to the geopotential surface; i.e., the constant *Eulerian* (x, y, z) frame (the term *z-level* is used in the subsequent). The tracer mixing tensor is diagonal in the (x, y, z) coordinate system and there is no anti-symmetric component. Additionally, the components of the diffusion tensor in the horizontal directions are generally taken to be roughly 10^7 times larger than the vertical components. This large anisotropy arises from the strong vertical stratification in most regions of the ocean which suppresses vertical mixing.

B.3 Isopycnal diffusion

From the standpoint of tracer mixing, the natural set of coordinates are those defined with respect to the isopycnal or neutral directions. These coordinates define what is termed here

[2] The variance of the tracer is given by $var(T) = V^{-1}[\int d\vec{x}\, T^2 - V^{-1}(\int d\vec{x}\, T)^2] \geq 0$, with $V = \int d\vec{x}$ the domain volume. Reducing $\int d\vec{x}\, T^2$ is therefore equivalent to reducing $var(T)$.

the *orthonormal isopycnal frame*. Diffusion in this frame is assumed to be diagonal in the formulation of Redi (1982). The along isopycnal diffusion is on the order of 10^7 times greater than the *diapycnal* diffusion. Refer to Redi (1982), McDougall (1987), Gent and McWilliams (1990) for discussions motivating such diffusive mixing.

This section presents some technical details related to representing the isopycnal diffusion tensor, which is diagonal and thus simple in the *orthogonal isopycnal frame*, in the *Eulerian* (x, y, z) frame which is relevant for MOM. Some of these results are derived by Redi (1982) but using a different formalism.

B.3.0.1 Basis vectors

Consider a first order tensor, or a vector $\vec{V}$. This object has any number of representations determined by the particular frame of reference. For example, a basis for two frames of interest yield the representations

$$\vec{V} = V^m \vec{e}_m, \tag{B.36}$$
$$= V^{\overline{m}} \vec{e}_{\overline{m}}, \tag{B.37}$$

where the space-time dependent functions V^m and $V^{\overline{m}}$ are the *coordinates* for the vector as represented in the respective frame. There are two sets of basis vectors which define the frames considered here:

$$\vec{e}_1 = \hat{x} \tag{B.38}$$
$$\vec{e}_2 = \hat{y} \tag{B.39}$$
$$\vec{e}_3 = \hat{z}, \tag{B.40}$$

which is the familiar Cartesian unit basis for the z-level frame, and

$$\vec{e}_{\overline{1}} = \frac{\hat{z} \times \nabla\rho}{|\hat{z} \times \nabla\rho|} \tag{B.41}$$
$$\vec{e}_{\overline{2}} = \vec{e}_{\overline{3}} \times \vec{e}_{\overline{1}}, \tag{B.42}$$
$$\vec{e}_{\overline{3}} = \frac{\nabla\rho}{|\nabla\rho|}, \tag{B.43}$$

which defines the orthonormal isopycnal frame determined by the fluctuating geometry of a locally referenced isopycnal surface.

Transforming from the z-level frame to the orthonormal isopycnal frame requires a linear transformation. As a tensor equation, the transformation is written $\vec{e}_{\overline{m}} = \Lambda^m_{\overline{m}} \vec{e}_m$. For the purposes of organizing the components of the transformation, this equation can be written as

$$(\vec{e}_{\overline{1}}\ \vec{e}_{\overline{2}}\ \vec{e}_{\overline{3}}) = (\vec{e}_1\ \vec{e}_2\ \vec{e}_3) \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \tag{B.44}$$

where $\vec{S} = \nabla_\rho z = -z_\rho \nabla_z \rho = (S_x, S_y, 0) = (-\rho_x/\rho_z, -\rho_y/\rho_z, 0)$ is the isopycnal slope vector with magnitude S. Since the transformation is between two orthonormal frames, this transformation matrix is a rotation (unit determinant and inverse given by the transpose) and so can be interpreted in terms of Euler angles (e.g., Redi 1982).

B.3.0.2 Orthonormal isopycnal frame

The relevance of the orthonormal isopycnal frame arises from the diagonal nature of diffusion within this frame, as assumed by Redi (1982). In general, diffusion is thought to occur predominantly along the two orthogonal directions $\vec{e}_{\overline{1}}$ and $\vec{e}_{\overline{2}}$, which define the neutral directions that are tangent to the locally referenced isopycnal surface. Diffusion in the diapycnal direction $\vec{e}_{\overline{3}}$ typically occurs with a diffusion coefficient which is on the order of 10^{-7} times smaller. Therefore, in this frame the symmetric diffusion tensor takes on the diagonal form

$$K^{\overline{mn}} = \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_D \end{pmatrix}, \tag{B.45}$$

where A_I are the along isopycnal diffusion coefficients and $A_D \approx 10^{-7} A_I$ is the diapycnal diffusion coefficient [3]. The diffusion tensor written in terms of projection operators takes the form

$$K^{\overline{mn}} = A_I(\delta^{\overline{mn}} - e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}) + A_D e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}} \tag{B.46}$$

with $\vec{e}_{\overline{3}}$ having the components $(0,0,1)^T$ in the orthonormal isopycnal frame. Explicitly, the diffusion operator in these coordinates is

$$R(T) = \partial_{\vec{e}_{\overline{1}}}(A_I \partial_{\vec{e}_{\overline{1}}} T) + \partial_{\vec{e}_{\overline{2}}}(A_I \partial_{\vec{e}_{\overline{2}}} T) + \partial_{\vec{e}_{\overline{3}}}(A_D \partial_{\vec{e}_{\overline{3}}} T) \tag{B.47}$$

where the diffusion coefficients are generally nonconstant.

B.3.0.3 z-level frame

The orthogonal isopycnal frame allowed for a simple prescription of the isopycnal diffusion process. In order to describe such a diffusion process in other frames of reference, the components of K must be transformed. Such rules of transformation are standard (e.g., Aris 1962). The diagonal components $K^{\overline{mn}}$ from the orthogonal isopycnal frame are transformed to the z-level frame through

$$K^{mn} = \Lambda^m_{\overline{m}} K^{\overline{mn}} \Lambda^n_{\overline{n}}, \tag{B.48}$$

where the transformation matrix is given by equation (B.44). Written as matrices with Λ having components $\Lambda^n_{\overline{n}}$, this transformation takes the form $K = \Lambda \overline{K} \Lambda^T$, where $\overline{K}$ has the diagonal form given in equation (B.45). Performing the matrix multiplication

$$K^{mn} = \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_D \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix}, \tag{B.49}$$

yields the components of the diffusion tensor in the z-level system given by Redi (1982)

$$K^{mn} = \frac{A_I}{(1+S^2)} \begin{pmatrix} 1 + \frac{\rho_y^2 + \epsilon \rho_x^2}{\rho_z^2} & (\epsilon - 1)\frac{\rho_x \rho_y}{\rho_z^2} & (\epsilon - 1)\frac{\rho_x}{\rho_z} \\ (\epsilon - 1)\frac{\rho_x \rho_y}{\rho_z^2} & 1 + \frac{\rho_x^2 + \epsilon \rho_y^2}{\rho_z^2} & (\epsilon - 1)\frac{\rho_y}{\rho_z} \\ (\epsilon - 1)\frac{\rho_x}{\rho_z} & (\epsilon - 1)\frac{\rho_y}{\rho_z} & \epsilon + S^2 \end{pmatrix}, \tag{B.50}$$

[3] Since this frame is flat (metric is the unit tensor), there is no distinction between lower and upper labels on the components of a tensor. In general, the convention for tensors such as $K^{\overline{mn}}$, when written as a matrix, is that the first index indicates the row and the second index is for the column.

which can also be written

$$K^{mn} = \frac{A_I}{(1+S^2)} \begin{pmatrix} 1+S_y^2+\epsilon S_x^2 & (\epsilon-1)S_xS_y & (1-\epsilon)S_x \\ (\epsilon-1)S_xS_y & 1+S_x^2+\epsilon S_y^2 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon+S^2 \end{pmatrix}. \tag{B.51}$$

Note that Redi uses the symbol δ instead of S in her expression. The ratio of the diapycnal to isopycnal diffusivities defines the typically small number $\epsilon = A_D/A_I \approx 10^{-7}$. An equivalent form is suggested by writing the tensor in the orthonormal isopycnal frame as

$$K^{\overline{mn}} = A_I \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + A_I(\epsilon-1) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{B.52}$$

which separates the effects of the anisotropy between the along and across isopycnal directions. This expression can be written in the compact form

$$K^{\overline{mn}} = K_{\overline{mn}} = A_I[\delta_{\overline{mn}} - (\vec{e}_{\overline{3}})_{\overline{m}}(\vec{e}_{\overline{3}})_{\overline{n}}] + A_D(\vec{e}_{\overline{3}})_{\overline{m}}(\vec{e}_{\overline{3}})_{\overline{n}}\,, \tag{B.53}$$

as noted in equation (B.46). Transforming to the z-level frame yields

$$K^{mn} = A_I \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{A_I(\epsilon-1)}{\rho_x^2+\rho_y^2+\rho_z^2} \begin{pmatrix} \rho_x^2 & \rho_x\rho_y & \rho_x\rho_z \\ \rho_x\rho_y & \rho_y^2 & \rho_y\rho_z \\ \rho_x\rho_z & \rho_y\rho_z & \rho_z^2 \end{pmatrix}, \tag{B.54}$$

which can also be written

$$K^{mn} = K_{mn} = A_I\delta_{mn} + A_I(\epsilon-1)\frac{\partial_m\rho\,\partial_n\rho}{|\nabla\rho|^2} = A_I(\delta_{mn} - \hat{\rho}_m\hat{\rho}_n) + A_D\hat{\rho}_m\hat{\rho}_n, \tag{B.55}$$

where $\hat{\rho}_m = \partial_m\rho/|\nabla\rho|$ are the components of the diapycnal unit vector $\vec{e}_{\overline{3}}$ written in the z-level frame [see equation (B.43)]. Note that equation (B.55) could have been written down immediately once equation (B.53) was hypothesized, and the unit vectors (B.41), (B.42), and (B.43) were determined.

B.3.0.4 Small angle approximation

There have been no small angle approximations (i.e., $S << 1$) made in computing the representation K^{mn}. Therefore, for modeling the physical process of diffusion employing the isopycnal diffusive mixing hypothesis in the z-level frame, K^{mn} given here is *the* form of the diffusion tensor to be used. In so doing, the physics of diapycnal diffusion is isolated in the parameter ϵ. The small angle approximation of Cox (1987) is an additional statement regarding the behavior of the bulk of the ocean (that isopycnals have only a rather small slope except in convection regions). This approximation recovers the form quoted in Gent and McWilliams (1990)

$$K^{mn}_{\text{small}} = A_I \begin{pmatrix} 1 & 0 & (\epsilon-1)\frac{\rho_x}{\rho_z} \\ 0 & 1 & (\epsilon-1)\frac{\rho_y}{\rho_z} \\ (\epsilon-1)\frac{\rho_x}{\rho_z} & (\epsilon-1)\frac{\rho_y}{\rho_z} & \epsilon+S^2 \end{pmatrix} = A_I \begin{pmatrix} 1 & 0 & (1-\epsilon)S_x \\ 0 & 1 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon+S^2 \end{pmatrix} \tag{B.56}$$

Note that Cox (1987) originally retained the $(1,2) = (2,1)$ elements equal to $-S_xS_y$. In the small angle approximation, however, this term is negligible. Additionally, and most crucially,

if this term is retained, the small angle tensor will diffuse locally referenced potential density whereas the full tensor will not. Hence, it is important to use the physically consistent form of the small angle approximation which drops the $(1,2) and (2,1)$ elements.

As the small angle approximation is commonly used, it is perhaps interesting to ask the following question: Given the small angle approximation representation of the diffusion tensor in the z-level frame, what is the representation of this tensor in the orthonormal isopycnal frame? This tensor cannot be the diagonal form given in equation (B.45) since that form transformed into the full non-small angle representation of equation (B.50). Using the full transformation back to the orthonormal isopycnal frame, $K^{\overline{mn}}_{\text{small}} = \Lambda^{\overline{m}}{}_{m} K^{mn}_{\text{small}} \Lambda^{\overline{n}}{}_{n}$, or in matrix form

$$K^{\overline{mn}}_{\text{small}} = A_I \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \times \tag{B.57}$$

$$\begin{pmatrix} 1 & 0 & (1-\epsilon)S_x \\ 0 & 1 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon + S^2 \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix}, \tag{B.58}$$

yields the orthonormal isopycnal frame representation of the small angle approximated tensor[4]

$$K^{\overline{mn}}_{\text{small}} = \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I(1+S^2) & 0 \\ 0 & 0 & A_D \end{pmatrix} + \frac{A_D S^2}{1+S^2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & S \\ 0 & S & 1 \end{pmatrix}. \tag{B.59}$$

The small angle approximation is seen to add a small amount of along isopycnal mixing (the $(2,2)$ term $A_I(1+S^2)$) as well as a term proportional to the generally small number $A_D S^2$. Dropping these terms is consistent with the small angle approximation, which then recovers the purely diagonal mixing tensor $K^{\overline{mn}} = A_I(\delta^{\overline{mn}} - e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}) + A_D e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}$.

B.3.0.5 Errors with z-level mixing

Consider the traditional diffusive mixing (Section B.2) with some tensor **I** that is diagonal in the z-level frame. Diffusive mixing with **I** is quite different than diffusive mixing with **K**, as can be seen clearly by transforming **I** to the orthonormal isopycnal frame

$$I^{\overline{mn}} = \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \begin{pmatrix} A_H & 0 & 0 \\ 0 & A_H & 0 \\ 0 & 0 & A_V \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix}, \tag{B.60}$$

which yields

$$I^{\overline{mn}} = \frac{A_H}{1+S^2} \begin{pmatrix} 1+S^2 & 0 & 0 \\ 0 & 1+\tilde{\epsilon}S^2 & -S(1-\tilde{\epsilon}) \\ 0 & -S(1-\tilde{\epsilon}) & \tilde{\epsilon}+S^2 \end{pmatrix}, \tag{B.61}$$

where $\tilde{\epsilon} = A_V/A_H$ is the ratio of the vertical to horizontal diffusion coefficient.

[4]Note the rotation need not transform the $\hat{x} \leftrightarrow \hat{y}$ symmetry present in the (x,y,z) form of the small angle mixing tensor into a $\vec{e}_{\overline{1}} \leftrightarrow \vec{e}_{\overline{2}}$ symmetry in the $(\vec{e}_{\overline{1}}, \vec{e}_{\overline{2}}, \vec{e}_{\overline{3}})$ form. The (x,y) coordinate symmetry, however, is preserved.

Therefore, diffusive mixing with **I** in the z-level frame introduces first order in slope errors in the off diagonal terms, whereas the diagonal terms contain second order in slope errors. The error in the $(3,3)$ component, however, is the most relevant as it represents an added source of diapycnal mixing (i.e., a *false diapycnal mixing*). For example, with the usual diffusivity ratio $\tilde{\epsilon} \approx 10^{-7}$, modest slopes $S \approx 3 \times 10^{-4}$ are sufficient to add diapycnal mixing through the S^2 term which is on the same order as $\tilde{\epsilon}$. It is for this reason that horizontal mixing, especially in regions of the ocean with larger than modest slopes but still within the small slope approximation, is incompatible with the hypothesis that mixing is predominantly along the isopycnal directions.

As seen above, the distinction between horizontal and along isopycnal mixing is quite important. However, the distinction between vertical mixing and diapycnal mixing is not generally important except for extremely large slopes. The reason for this ambiguity can be easily understood by looking at the small angle approximation to the isopycnal tensor [equation (B.56], and setting A_I to zero in order to focus on the diapycnal piece

$$K^{mn}(A_I = 0)_{\text{small}} = A_D \begin{pmatrix} 0 & 0 & -S_x \\ 0 & 0 & -S_y \\ -S_x & -S_y & 1 \end{pmatrix}. \tag{B.62}$$

The off diagonal terms represent the difference between vertical mixing and diapycnal mixing. These terms are down by a single factor of the slope. To completely determine the error introduced by neglecting these terms, it is useful to write the diffusion operator $R(T)$ arising from this tensor and performing a scaling within the small slope approximation. For this purpose, consider the horizontal direction of steepest slope to be the x-direction. The y-direction will therefore be ignored. Also assume a constant diapycnal mixing coefficient A_D. With these approximations, the diffusion operator takes the form

$$\begin{aligned} R(T) &= A_D[\partial_z(S_x \partial_x T) + \partial_z \partial_z T] \\ &\approx A_D \left(\frac{\Delta T}{(\Delta x)^2} + \frac{\Delta T}{(\Delta z)^2} \right) \\ &\approx A_D \frac{\Delta T}{(\Delta z)^2}, \end{aligned} \tag{B.63}$$

where the last approximation assumed $\Delta z << \Delta x$. Hence, the off-diagonal pieces in the diffusion tensor are quite negligible in the small slope approximation. Therefore, with this scaling, the distinction between vertical and diapycnal mixing is negligible as well.

B.4 Symmetric and anti-symmetric tensors

The horizontal-vertical and isopycnal mixing of the previous two sections employed symmetric positive semi-definite tensors representing dissipative mixing processes. The presence of an anti-symmetric component to the mixing tensor, as seen in Section B.1, can be thought of as an added advective transport of tracers [see in particular the equations leading up to (B.19)]. The transport velocity defined by such a tensor is divergence-free [equation (B.16)] and preserves all tracer moments if it has zero normal value on the boundaries [discussion following equation (B.34)]. The parameterization of Gent and McWilliams (1990) has been reformulated in terms of such an effective transport velocity by Gent et. al., (1995). Other types of mixing can be parameterized by anti-symmetric mixing tensors. For example, Middleton and Loder (1989) summarize the effects of mixing by non-zero angular momentum waves which gives rise to skew-fluxes.

B.5 Summary

The traditional manner of parameterizing tracer mixing in ocean models is through the use of a second order mixing tensor. The basic mathematical properties of such mixing have been reviewed in this appendix. In general, the mixing tensor contains both a symmetric positive-semi-definite component and an anti-symmetric component. The symmetric component represents dissipative processes associated with diffusive mixing. The numerical discretization of this process is discussed further in Appendix C. The anti-symmetric component can be formulated in terms of an advective velocity which, in addition to the usual Eulerian velocity, contributes to conservative tracer mixing.

Because there will always be mixing processes too small to resolve in ocean models, there will always be a need to rationally parameterize these processes. Determining the nature of mixing tensors relevant for these parameterizations, if such objects exist, is proving to be one of the most challenging problems in physical oceanography today.

Appendix C

Isoneutral diffusion discretization

The purpose of this appendix is to present a full derivation of MOM's isoneutral diffusion discretization. Most readers will find this appendix quite uninteresting. It is presented as part of the MOM manual, nevertheless, since it illustrates in the utility of functional approaches to deriving discretizations. A more streamlined, and hopefully interesting, discussion can be found in Griffies, Gnanadesikan, Pacanowski, Larichev, Dukowicz, and Smith (hereafter, Griffies *et al*. 1998).

Brief history of this appendix:

- Early 1996: Derivation presented for full bottom cells.
- June 1997: Generalization to partial bottom cells.
- March 1998: Typos corrected in the discussion of the meridional flux. These mistakes were *not* made in the original MOM 2 code.
- May 1999: Bug found in computation of vertical flux component for partial cells. Division by *dht* should have been division by *dhwt*. Code and manual updated.

Note that in this Appendix, diagonal diffusive fluxes are considered positive if they are down the tracer gradient. The code in MOM, however, employs the opposite convention. The diffusion operator in MOM picks up a minus sign which renders it the same as discussed in this Appendix. Clarification of this comment is provided in Section C.4.1.

C.0.1 Summary and Caveats

The new scheme does the following:

1. It produces a zero flux of locally referenced potential density. The Cox (1987) scheme did not respect this property, and this led to the instability of that scheme.
2. It reduces tracer variance, and produces downgradient oriented tracer fluxes when considering a particular finite volume.
3. It computes the best approximation to the neutral directions within the limitations of the discrete lattice.
4. It requires zero background diffusion to remain stable; the Cox (1987) scheme blew-up without this diffusion.

Some limitations of the new scheme:

1. Because it only guarantees downgradient oriented tracer fluxes over finite volumes, rather than individual tracer cells, tracers can move outside their physically constrained range. As such, the new scheme does not guarantee positive-definiteness of tracers. This problem appears most pernicious for passive tracers in steep sloped regions. The temperature and salinity, being constrained by the need to produce a zero flux of locally referenced potential density, appear less problematic, though they too can move outside physical bounds. In order to do the pure isoneutral diffusion problem for all slopes, it appears necessary to employ a positive-definite isoneutral diffusion scheme. Speculations about the requirements of such a scheme were given by Beckers *et al.* (1998). Currently, no scheme exists. Note that since the problems appear mostly in steep sloped regions, it might be physically justifiable to apply horizontal fluxes in these regions (e.g., see Treguier, Held, and Larichev 1997). If this is the case, then the issue becomes much less problematic, and one perhaps is not constrained by this technical limitation of the current diffusion scheme.

2. Upon stabilizing the diffusion scheme, it is possible to remove the horizontal background diffusion previously required with the Cox (1987) scheme. In so doing, one is now exposed to problems with dispersive advection schemes. These problems are most apparent next to topography in regions where the tracers are aligned with isoneutrals, and so diffusive fluxes are weak. This "Peclet grid noise" problem is fundamental to dispersive advection schemes. It is *not* a problem with the rotated diffusion scheme.

C.0.2 Functional formalism

A fundamental property of diffusion is that it dissipates tracer variance. Unfortunately, it is not guaranteed that every discretization of a diffusion operator will satisfy this desired dissipative property. Fortunatly, there is a means to ensure variance reduction by employing a straightforward formalism. The mathematical property that is exploited with this formalism is to note that the diffusion operator, when acting on a passive tracer, is a linear self-adjoint operator. As such, it has an associated negative semi-definite functional (e.g., Courant and Hilbert, 1953). For example, Laplacian diffusion in an isotropic media, $R(T) = \nabla^2 T$, is identified with the functional derivative $R(T) = \delta\mathcal{F}/\delta T$, where $\mathcal{F} \equiv -(1/2)\int |\nabla T|^2 d\vec{x} \leq 0$ is the associated functional. For a general diffusion tensor, the functional is given by $\mathcal{F} \equiv (1/2)\int d\vec{x}\, \vec{F}\cdot\nabla T = (1/2)\int d\vec{x}\, T_m\, K^{mn}\, T_n$, where $T_m = \partial_m T$ and repeated indices are summed. The negative semi-definiteness of the functional $\mathcal{F}$ is related to the dissipative property of the diffusion operator; i.e., one implies the other. It is also directly related to the symmetric positive semi-definiteness of the diffusion tensor $K^{mn} = K^{nm}$ and the associated downgradient orientation of the diffusive flux $\vec{F}\cdot\nabla T \leq 0$.

On the lattice, not every consistent[1] discrete diffusion operator corresponds to a negative semi-definite discrete functional. Therefore, a consistent numerical diffusion operator does not necessarily possess the dissipative properties of the continuum operator. For the Laplacian in an isotropic media, it is trivial to produce a dissipative numerical operator. In the anisotropic case, such as isoneutral diffusion, it is nontrivial. Indeed, the original discretization of the isoneutral diffusion operator in the GFDL model (Cox, 1987) is numerically consistent but not dissipative. The approach taken in the derivation of the new discretization of this operator is to

[1]Consistent in that the discretization reduces to the correct continuum operator as the grid size goes to zero.

focus on discretizing the functional first and then to take the discrete version of the functional derivative in order to derive the discrete diffusion operator. This approach ensures that the discretized operator is dissipative, no matter how the functional is discretized.

C.0.3 Neutral directions

Within the functional framework, provision is made for a discretization of the diffusion fluxes which are aligned according to a *self-consistent* approximation to the neutral directions. Self-consistency means that there is a zero along isoneutral flux of locally referenced potential density. A zero isoneutral flux of locally referenced potential density implies a balance between the neutral direction flux of the two active tracers: $\alpha\vec{F}(\theta) = \beta\vec{F}(s)$, where $\alpha = -\rho_\theta/\rho$ and $\beta = \rho_s/\rho$ are the thermal expansion and saline contracion coefficients. In order to ensure this balance in a z-coordinate ocean model, it is sufficient to compute the density gradients in terms of the active tracer gradients and the thermal and saline coefficients (see Griffies *et al.* 1998 for more details). Special care must be taken when choosing the reference points for evaluating these gradients, and the details are given in Section C.2.7. Without a self-consistent discretization which guarantees a zero flux of locally referenced potential density, the isoneutral diffusion operator will produce grid noise, even if it ensures variance does not increase.

C.0.4 Full isoneutral diffusion tensor

Traditionally, the implementation of isoneutral diffusion has been with the small slope approximation made to the full tensor (Cox 1987, Gent and McWilliams 1990). In MOM, both the small and full Redi tensors are discretized in this appendix. At this time (June, 1997) there is no compelling reason to employ the full tensor. As a result, little effort has been made to optimize the computational efficiency of the full tensor diffusion operator. Currently, it runs roughly six times slower than the small angle tensor. It remains as a frozen part of the code. It remains in this appendix since it turned out that the derivation was a easier to check when using the full tensor due to the greater amount of three-dimensional symmetry than present with the small tensor.

C.0.5 Active tracers versus passive tracers

When acting on the active tracers, the isoneutral diffusion operator is nonlinear since the diffusion tensor is a function of the active tracers. This point is important since the functional formalism assumes the diffusion operator to be a linear self-adjoint operator. However, since it is assumed that the diffusion operator is the same for both the active and passive tracers, the restriction of the functional formalism to the passive tracers is no problem. What is done is to use the functional machinery assuming the tracer is passive. When it comes time to discretize the neutral directions, the understanding of how to properly align the slopes so that the diffusion operator is self-consistent (i.e., so that it will not flux locally referenced potential density along the neutral directions) will be incorporated. Without ensuring self-consistency, the variance reducing diffusion operator can produce an accumulation of active tracer variance at the grid scale, which can create a useless numerical solution full of grid noise.

C.1 Functional for isoneutral diffusion

The diffusion tensor representing along and across isoneutral mixing is given in the projection operator form as

$$\begin{aligned} K^{mn} &= A_I(\delta^{mn} - \hat{\gamma}^m\hat{\gamma}^n) + A_D\hat{\gamma}^m\hat{\gamma}^n \\ &= (A_I - A_D)(\delta^{mn} - \hat{\gamma}^m\hat{\gamma}^n) + A_D\delta^{mn} \end{aligned} \tag{C.1}$$

with δ^{mn} the Kronecker delta, $\hat{\gamma} = \nabla\rho/|\nabla\rho|$ the unit vector normal to the neutral direction (i.e., the dianeutral unit vector), $\nabla\rho = \rho_\theta\nabla\theta + \rho_s\nabla s$ the gradient of locally referenced potential density, $\rho_\theta = -\rho\,\alpha$, $\rho_s = \rho\,\beta$, where α and β are the usual thermal and saline expansion coefficients (Gill, 1982). Note the absence of pressure gradients in this expression for the gradient of locally referenced potential density (see Griffies *et al.* 1998 for more discussion of neutral directions within the present context). The isoneutral and dianeutral diffusion coefficients A_I and A_D are non-negative and can in general be functions of space-time. When acting on a vector, the isoneutral piece of the diffusion tensor projects the vector onto the local neutral direction, and the dianeutral piece projects the vector into the dianeutral direction.

Given this diffusion tensor, the functional for isoneutral diffusion is written

$$\mathcal{F} = -\frac{1}{2}\int d\vec{x}\,(A_I - A_D)\frac{|\nabla\rho\wedge\nabla T|^2}{|\nabla\rho|^2} - \frac{1}{2}\int d\vec{x}\,A_D|\nabla T|^2, \tag{C.2}$$

or more explicitly,

$$\begin{aligned} \mathcal{F} = &- \frac{1}{2}\int d\vec{x}\,(A_I - A_D)\frac{(T_x\rho_y - T_y\rho_x)^2 + (T_y\rho_z - T_z\rho_y)^2 + (T_z\rho_x - T_x\rho_z)^2}{\rho_x^2 + \rho_y^2 + \rho_z^2} \\ &- \frac{1}{2}\int d\vec{x}\,A_D(T_x^2 + T_y^2 + T_z^2). \end{aligned} \tag{C.3}$$

The small slope approximation amounts to taking the limit of $|\rho_x|, |\rho_y| << |\rho_z|$, and dropping terms of order $slope * (A_D/A_I)$, with *slope* the small isoneutral slope. The resulting functional is

$$\mathcal{F}^{small} = -\frac{1}{2}\int d\vec{x}\,A_I\frac{(T_y\rho_z - T_z\rho_y)^2 + (T_z\rho_x - T_x\rho_z)^2}{\rho_z^2} - \frac{1}{2}\int d\vec{x}\,A_D(T_z)^2. \tag{C.4}$$

For comparison with the results in the discrete case, it is useful to obtain the continuum diffusion operator using the relation

$$R(T) = -\nabla\cdot\vec{F} = \frac{\delta\mathbf{L}}{\delta T} - \partial_m(\frac{\delta\mathbf{L}}{\delta T_m}), \tag{C.5}$$

where $\mathcal{F} = \int d\vec{x}\,\mathbf{L}$ defines the integrand $\mathbf{L}$ to the functional. With the quadratic form defined in Equation (C.3), and $\delta\mathbf{L}/\delta T = 0$, the diffusive fluxes are given by

$$\vec{F} = \frac{\delta\mathbf{L}}{\nabla T}. \tag{C.6}$$

For the full tensor, this leads to the isoneutral flux

$$\vec{F}(T) = -(A_I - A_D)(\hat{\gamma}\wedge\nabla T)\wedge\hat{\gamma} - A_D\nabla T, \tag{C.7}$$

and the small angle result is given by

$$\vec{F} = -\left(\frac{A_I}{\rho_z}\right)(\nabla\rho \wedge \nabla T) \wedge \hat{z} - \hat{z}\, A_D\, T_z \tag{C.8}$$

In order to directly compare with the discrete form of the flux, it is useful to expand these vector expressions into components:

$$-F^x = -\frac{\delta \mathbf{L}}{\delta T_x} = (A_I - A_D)\frac{\rho_y(T_x\rho_y - T_y\rho_x) + \rho_z(T_x\rho_z - T_z\rho_x)}{|\nabla\rho|^2} + A_D T_x, \tag{C.9}$$

$$-F^y = -\frac{\delta \mathbf{L}}{\delta T_y} = (A_I - A_D)\frac{\rho_z(T_y\rho_z - T_z\rho_y) + \rho_x(T_y\rho_x - T_x\rho_y)}{|\nabla\rho|^2} + A_D T_y, \tag{C.10}$$

$$-F^z = -\frac{\delta \mathbf{L}}{\delta T_z} = (A_I - A_D)\frac{\rho_x(T_z\rho_x - T_x\rho_z) + \rho_y(T_z\rho_y - T_y\rho_z)}{|\nabla\rho|^2} + A_D T_z. \tag{C.11}$$

$$-F^x = A_I(T_x - T_z\frac{\rho_x}{\rho_z}) \tag{C.12}$$

$$-F^y = A_I(T_y - T_z\frac{\rho_y}{\rho_z}) \tag{C.13}$$

$$-F^z = A_I\left(-T_x\,\frac{\rho_x}{\rho_z} - T_y\,\frac{\rho_y}{\rho_z} + T_z\frac{\rho_x^2 + \rho_y^2}{\rho_z^2}\right) + A_D T_z, \tag{C.14}$$

The first term in these expressions vanishes when the tracer is parallel to the neutral directions. For most oceanographic cases, the difference $A_I - A_D \approx A_I$ to many orders of accuracy. This assumption is made in the model implementation of the full isoneutral diffusion tensor. Note the presence of the dianeutral pieces $A_D T_x$ and $A_D T_y$ in the full tensor. In the case of very steep neutral directions, these terms become relevant.

C.2 Discretization of the diffusion operator

In various iterations with this derivation, it was found that the form of the functional given by Equation (C.3) is most convenient to discretize as it allows for projection onto two-dimensional planar regions rather than having to consider a full three-dimensional discretization as might be necessary for discretizing the full tensor. The derivation of the full tensor is presented in these notes with the small slope results obtained from the small slope limit of the full tensor. An independent derivation starting from a discretization of the small slope version of the functional (equation (C.4)) verifies the validity of the small slope limit from the full tensor. Additionally, the derivation is presented for the MOM default grid in which for nonuniform grids, the T-point is not in the center of the T-cell. The form of the discretized operator is dependent on this choice of T-cell placement. As of this writing, only the MOM default grid discretization of the diffusion operator has been implemented. Therefore, it is recommended that one not use the option *t_center*. Idealized tests (flat bottom model) of the new diffusion scheme with option *t_center*, however, show no problems.

As seen in the discussion from Section C.1, the discretization of the diffusion operator at a particular grid point is derived from the functional derivative of the discretized functional. On a discrete lattice, this derivative is simply the partial derivative

$$R(T)_{i,k,j} = \frac{1}{V_{T_{i,k,j}}}\frac{\partial \mathcal{F}[T_{i,k,j}]}{\partial T_{i,k,j}}. \tag{C.15}$$

It is necessary to divide out the volume of the T-cell in the discrete case since with finite cell volumes, the appropriate delta function which occurs in the derivative is the dimensionless Kronecker delta rather than the dimensionful (dimensions 1/volume) Dirac delta which appears in the continuum case. The procedure, therefore, is to identify those pieces of the discretized functional which contain contributions from the discretized tracer value $T_{i,k,j}$, as these are the pieces which contribute to the diffusion operator for this T-cell. An enumeration of these pieces depends on the particular discretization of the functional. One overriding principle used to guide this discretization is to recover the familiar discretization of the Laplacian (5-point in the two-dimensional case) in the case of flat neutral directions. Furthermore, all details about the discretization will be made at the level of the functional. These details include the particular grid choice and the choice for reference points to be used in approximating the neutral directions.

In the following, no explicit reference will be made to the time step. As it is necessary to lag the time step by one step for numerical stability of the diffusion equation, a time step of $\tau - 1$ will be implicit throughout.

C.2.1 A one-dimensional warm-up

In order to illustrate the general framework provided by the functional approach, it is useful to consider the trivial case of one-dimensional diffusion. It should be noted that in one-dimension, there is no issue of isoneutral diffusion and so this example cannot illustrate any of the subtle issues related to the discretization of the neutral directions. The issues of partial vertical cells are avoided in this one-dimensional case as well. Both of these issues will be discussed at the appropriate point in the derivation of the three-dimensional case.

Figure C.1 shows the grid, which corresponds to the x-axis in Figure C.2. Let $V(n)$ and $A(n)$ be the volume and diffusion coefficient corresponding to the subcell n. Consider the following discretization of the functional

$$\mathcal{F} = -\frac{1}{2}\sum_i \sum_n A(n)V(n)(\delta_x T(n))^2, \tag{C.16}$$

where the n sum is over the subcells relevant for each T-cell. In particular, the four terms containing a contribution from T_i are given by

$$\begin{aligned} 2\mathcal{F}[T_i] = \; & - \; A(7)V(7)(\delta_x T_{i-1})^2 - A(1)V(1)(\delta_x T_{i-1})^2 \\ & - \; A(2)V(2)(\delta_x T_i)^2 - A(5)V(5)(\delta_x T_i)^2, \end{aligned} \tag{C.17}$$

where the tracer derivative is assumed the same for the two subcells cells 7 and 1 and the two subcells 2 and 5. The volumes of the subcells are

$$V(1) = V(7) = \frac{dxu_{i-1}}{2} \tag{C.18}$$

$$V(2) = V(5) = \frac{dxu_i}{2}, \tag{C.19}$$

and the derivatives are

$$\delta_x T_{i-1} = \frac{T_i - T_{i-1}}{dxu_{i-1}} \tag{C.20}$$

$$\delta_x T_i = \frac{T_{i+1} - T_i}{dxu_i}. \tag{C.21}$$

The derivative of the functional with respect to T_i is

$$\frac{\partial \mathcal{F}}{\partial T_i} = -\frac{A(7)V(7)}{dxu_{i-1}}\,\delta_x T_{i-1} - \frac{A(1)V(1)}{dxu_{i-1}}\,\delta_x T_{i-1} + \frac{A(2)V(2)}{dxu_i}\,\delta_x T_i + \frac{A(5)V(5)}{dxu_i}\,\delta_x T_i, \tag{C.22}$$

which can be rearranged to

$$\frac{\partial \mathcal{F}}{\partial T_i} = \overline{A(2)}^x \delta_x T_i - \overline{A(7)}^x \delta_x T_{i-1} \tag{C.23}$$

$$= dxt_i\,\delta_x\left(\overline{A(7)}^x \delta_x T_{i-1}\right), \tag{C.24}$$

yielding the discretized diffusion operator acting on T_i

$$\frac{1}{dxt_i}\frac{\partial \mathcal{F}}{\partial T_i} = R(T)_i = \delta_x\left(\overline{A(7)}^x \delta_x T_{i-1}\right). \tag{C.25}$$

The averaging operation for the diffusion coefficient is given by

$$\overline{A(2)}^x = \frac{A(2)+A(5)}{2} \tag{C.26}$$

$$\overline{A(7)}^x = \frac{A(7)+A(1)}{2}. \tag{C.27}$$

It is this averaging which is the only difference from the discretization derived using more traditional approaches. In the isoneutral case, the freedom to define the diffusion coefficient differently for each of the subcells will be an important degree of freedom exploited for ensuring stability of the diffusion scheme in the case of large neutral direction slopes.

The main points to be taken from this example are (A) The assumption that the tracer gradient is the same across the two adjacent subcells 1 & 7 and 2 & 5, respectively, (B) The presence of an average operator which arises from the recombination of terms, and (C) The freedom to prescribe a different diffusion coefficient to each of the sub-cells. The assumption (A) was necessary in order to derive the traditional 3-point Laplacian starting from the functional.

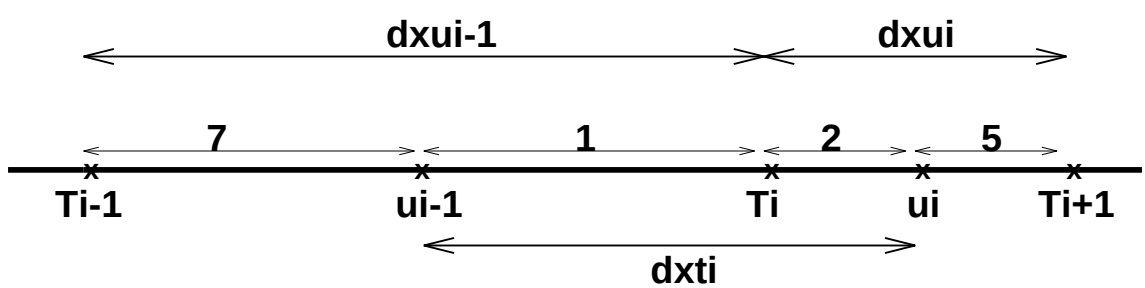


Figure C.1: One dimensional grid with subcells 7,1,2,5 corresponding to the x-axis cells in the x-y plane shown in Figure C.2.

C.2.2 Grid partitioning

Using the functional in the form given in Equation (C.3) for the full tensor, or Equation (C.4) for the small tensor, motivates a discretization which projects separately onto the three planar

slices x-y, y-z, and x-z. The only derivative in the orthogonal direction is within the $|\nabla\rho|^{-2}$ piece in the full tensor. The discretization of the $A_D|\nabla T|^2$ term can be done using the traditional 5-point Laplacian and will not be discussed further.

Figures C.2 and C.3 show the projections for the x-y and x-z planes as defined on the MOM default grid. Partial vertical cells are discussed in the Section C.2.3. The y-z plane is similar. Within a plane, the central T-cell is partitioned into 4 generally non-equal squares, with one corner being the T-point and the other corners being the corners of the T-cell. The sides of these squares have dimensions 1/2 the relevant cell distance. Note that the T-cell in the MOM default can generally be off centered; hence, the different sizes for the four quarter cells. There are an additional 8 quarter cells which surround the 4 central cells, each of size determined by the 1/2 cell distances and each of which has one corner lying at a T-point. In the discretization considered here, these are the 12 cells for a particular x-y or x-z projection which contain contributions from the central T-point. They correspond to the 4 subcells discussed in the previous one-dimensional example. For the three planes, there are a total of 36 quarter cells to which $T_{i,k,j}$ contributes. The direction perpendicular to the plane has distance given by the T-distance, either $dht_{i,k,j}$ for the x-y plane, dxt_i for the y-z plane, or dyt_j for the x-z plane. Inside of these quarter cells, it is possible to prescribe a different diffusion coefficient. This added freedom in choosing the diffusion coefficients is essential to the method employed for maintaining the numerical stability when the slopes get large. These details for slope checking will be given in Section C.2.9

C.2.3 Partial cells

The use of partial vertical cells (Chapter 26) introduces some added complication to the discretization of the isoneutral diffusion operator. Figures C.4 and C.5 illustrate the basic issues. When describing partial cells, it is necessary to introduce the following grid fields

$$dht_{i,k,j} = zw_{i,k,j} - zw_{i,k-1,j} \tag{C.28}$$

$$dhwt_{i,k-1,j} = zt_{i,k,j} - zt_{i,k-1,j}. \tag{C.29}$$

The field $zw_{i,k,j}$ defines the vertical distance from the ocean surface to the bottom of T-cell $T_{i,k,j}$. The field $zt_{i,k,j}$ defines the vertical distance from the ocean surface to the center of the T-cell $T_{i,k,j}$. The grid dimension $dht_{i,k,j}$ measures the vertical dimension of the T-cell $T_{i,k,j}$. When the partial cells are not used, $dht_{i,k,j}$ is known as dzt_k. The grid dimension $dhwt_{i,k-1,j}$ measures the vertical distance between the grid points within the T-cells $T_{i,k,j}$ and $T_{i,k-1,j}$. When the partial cells are not used, $dhwt_{i,k-1,j}$ is known as dzw_{k-1}.

Inspection of Figures C.4 and C.5 indicates that in order to construct the volume of the quarter cells, it is necessary to take differences such as $zw_{i,k,j} - zt_{i,k,j}$ (vertical dimension of quarter-cells 15 & 16 and 27 & 28), and $zt_{i,k,j} - zw_{i,k-1,j}$ (vertical dimension of quarter-cells 13 & 14 and 25 & 26). It will therefore prove useful to define the following field

$$\delta^{(i,k1,j)}_{(i,k2,j)} = 2(-1)^{k1-k2}(zw_{i,k1,j} - zt_{i,k2,j}). \tag{C.30}$$

Explicitly, for example, one has

$$\delta^{i,k-1+kr,j}_{i,k,j} = \begin{cases} 2(zt_{i,k,j} - zw_{i,k-1,j}) & \text{if } kr = 0 \\ 2(zw_{i,k,j} - zt_{i,k,j}) & \text{if } kr = 1 \end{cases} \tag{C.31}$$

and

$$\delta^{i,k,j}_{i,k+kr,j} = \begin{cases} 2(zw_{i,k,j} - zt_{i,k,j}) & \text{if } kr = 0 \\ 2(zt_{i,k+1,j} - zw_{i,k,j}) & \text{if } kr = 1. \end{cases} \tag{C.32}$$

Note that the added factor of 2 is used in order to allow for the equations with the partial vertical cells to reduce more easily to those of the full vertical cells.

Although δ as defined above represents the vertical distance of a quarter cell, this distance is not always that which is relevant for constructing the volume of the quarter cell for the purposes of the discretizing the functional. The reason is that when taking horizontal tracer derivatives across partial cells, a linear interpolation is performed to approximate the tracer which lives at the deeper point to the depth of the shallower point. For example, when taking a horizontal derivative between points $T_{i,k,j}$ and $T_{i+1,k,j}$ shown in Figure C.4, the value of $T_{i+1,k,j}$ is approximated by a linear interpolation to the same depth level as $T_{i,k,j}$. The method used for performing this interpolation is described in Section 26.2.3.

This interpolation, which in general is an interpolation to the *minimum* depth between two laterally adjacent tracer cells, effectively reduces the vertical spacing of quarter cell 17 to that of quarter cell 14. A similar consideration applies between quarter cells 19 and 13, for which the vertical spacing for quarter cell 19 is effectively the same as the smaller vertical spacing in quarter cell 13. Note that with this prescription for determining the horizontal gradients through interpolation, the isoneutral diffusion scheme that is derived in this appendix will correctly reduce to the horizontal diffusion scheme described in Section 26.2.3 for the case of zero neutral slopes.

In general, the above *minimum vertical spacing rule* implies that it is sufficient to only explicitly denote the vertical height of the quarter cells living in the eastern half, say, of a particular T-cell. Such is the case for full vertical cells as well, in which the vertical height is a function only of the vertical grid position (see Figures C.2 and C.3). Therefore, the only difference between the volume specification for the partial cells and the full cells is that for the partial cells, it is necessary to introduce the minimum operation. With this *minimum vertical spacing rule*, the vertical spacing relevant for defining the x-z plane quarter cell volumes will be written

$$\Delta^{(i,k1,j)}_{(i,k2,j)} = \min\left(\delta^{(i,k1,j)}_{(i,k2,j)}, \delta^{(i+1,k1,j)}_{(i+1,k2,j)}\right). \tag{C.33}$$

Likewise, the vertical spacing relevant for defining the y-z plane quarter cell volumes will be written

$$\Delta^{(i,k1,j)}_{(i,k2,j)} = \min\left(\delta^{(i,k1,j)}_{(i,k2,j)}, \delta^{(i,k1,j+1)}_{(i,k2,j+1)}\right). \tag{C.34}$$

Explicitly, the vertical spacings for the 12 x-z plane quarter cells in Figure C.4 are given by

$$\delta(13) = 2(zt_{i,k} - zw_{i,k-1}) = \delta^{i,k-1}_{i,k} \tag{C.35}$$

$$\delta(14) = 2(zt_{i,k} - zw_{i,k-1}) = \delta^{i,k-1}_{i,k} \tag{C.36}$$

$$\delta(15) = 2(zw_{i,k} - zt_{i,k}) = \delta^{i,k}_{i,k} \tag{C.37}$$

$$\delta(16) = 2(zw_{i,k} - zt_{i,k}) = \delta^{i,k}_{i,k} \tag{C.38}$$

$$\delta(17) = 2(zt_{i+1,k} - zw_{i+1,k-1}) = \delta^{i+1,k-1}_{i+1,k} \tag{C.39}$$

$$\delta(18) = 2(zw_{i+1,k} - zt_{i+1,k}) = \delta^{i+1,k}_{i+1,k} \tag{C.40}$$

$$\delta(19) = 2(zt_{i-1,k} - zw_{i-1,k-1}) = \delta^{i-1,k-1}_{i-1,k} \tag{C.41}$$

$$\delta(20) = 2(zw_{i-1,k} - zt_{i-1,k}) = \delta^{i-1,k}_{i-1,k} \tag{C.42}$$

$$\delta(21) = 2(zw_{i,k-1} - zt_{i,k-1}) = \delta^{i,k-1}_{i,k-1} \tag{C.43}$$

$$\delta(22) = 2(zw_{i,k-1} - zt_{i,k-1}) = \delta_{i,k-1}^{i,k-1} \tag{C.44}$$

$$\delta(23) = 2(zt_{i,k+1} - zw_{i,k}) = \delta_{i,k+1}^{i,k} \tag{C.45}$$

$$\delta(24) = 2(zt_{i,k+1} - zw_{i,k}) = \delta_{i,k+1}^{i,k}, \tag{C.46}$$

whereas the vertical spacing used to compute the volumes of these quarter cells is given by

$$\Delta(13) = \min\left(\delta_{i-1,k}^{i-1,k-1}, \delta_{i,k}^{i,k-1}\right) = \Delta_{(i-1,k)}^{(i-1,k-1)} \tag{C.47}$$

$$\Delta(14) = \min\left(\delta_{i,k}^{i,k-1}, \delta_{i+1,k}^{i+1,k-1}\right) = \Delta_{(i,k)}^{(i,k-1)} \tag{C.48}$$

$$\Delta(15) = \min\left(\delta_{i-1,k}^{i-1,k}, \delta_{i,k}^{i,k}\right) = \Delta_{(i-1,k)}^{(i-1,k)} \tag{C.49}$$

$$\Delta(16) = \min\left(\delta_{i,k}^{i,k}, \delta_{i+1,k}^{i+1,k}\right) = \Delta_{(i,k)}^{(i,k)} \tag{C.50}$$

$$\Delta(17) = \min\left(\delta_{i,k}^{i,k-1}, \delta_{i+1,k}^{i+1,k-1}\right) = \Delta_{(i,k)}^{(i,k-1)} \tag{C.51}$$

$$\Delta(18) = \min\left(\delta_{i,k}^{i,k}, \delta_{i+1,k}^{i+1,k}\right) = \Delta_{(i,k)}^{(i,k)} \tag{C.52}$$

$$\Delta(19) = \min\left(\delta_{i-1,k}^{i-1,k-1}, \delta_{i,k}^{i,k-1}\right) = \Delta_{(i-1,k)}^{(i-1,k-1)} \tag{C.53}$$

$$\Delta(20) = \min\left(\delta_{i-1,k}^{i-1,k}, \delta_{i,k}^{i,k}\right) = \Delta_{(i-1,k)}^{(i-1,k)} \tag{C.54}$$

$$\Delta(21) = \min\left(\delta_{i-1,k-1}^{i-1,k-1}, \delta_{i,k-1}^{i,k-1}\right) = \Delta_{(i-1,k-1)}^{(i-1,k-1)} \tag{C.55}$$

$$\Delta(22) = \min\left(\delta_{i,k-1}^{i,k-1}, \delta_{i+1,k-1}^{i+1,k-1}\right) = \Delta_{(i,k-1)}^{(i,k-1)} \tag{C.56}$$

$$\Delta(23) = \min\left(\delta_{i-1,k+1}^{i-1,k}, \delta_{i,k+1}^{i,k}\right) = \Delta_{(i-1,k+1)}^{(i-1,k)} \tag{C.57}$$

$$\Delta(24) = \min\left(\delta_{i,k+1}^{i,k}, \delta_{i+1,k+1}^{i+1,k}\right) = \Delta_{(i,k+1)}^{(i,k)}. \tag{C.58}$$

The j label has been omitted in these x-z expressions for purposes of brevity. Likewise, the vertical spacing for the 12 y-z plane quarter cells in Figure C.5 is given by

$$\delta(25) = 2(zt_{k,j} - zw_{k-1,j}) = \delta_{k,j}^{k-1,j} \tag{C.59}$$

$$\delta(26) = 2(zt_{k,j} - zw_{k-1,j}) = \delta_{k,j}^{k-1,j} \tag{C.60}$$

$$\delta(27) = 2(zw_{k,j} - zt_{k,j}) = \delta_{k,j}^{k,j} \tag{C.61}$$

$$\delta(28) = 2(zw_{k,j} - zt_{k,j}) = \delta_{k,j}^{k,j} \tag{C.62}$$

$$\delta(29) = 2(zt_{k,j+1} - zw_{k-1,j+1}) = \delta_{k,j+1}^{k-1,j+1} \tag{C.63}$$

$$\delta(30) = 2(zw_{k,j+1} - zt_{k,j+1}) = \delta_{k,j+1}^{k,j+1} \tag{C.64}$$

$$\delta(31) = 2(zt_{k,j-1} - zw_{k-1,j-1}) = \delta_{k,j-1}^{k-1,j-1} \tag{C.65}$$

$$\delta(32) = 2(zw_{k,j-1} - zt_{k,j-1}) = \delta_{k,j-1}^{k,j-1} \tag{C.66}$$

$$\delta(33) = 2(zw_{k-1,j} - zt_{k-1,j}) = \delta_{k-1,j}^{k-1,j} \tag{C.67}$$

$$\delta(34) = 2(zw_{k-1,j} - zt_{k-1,j}) = \delta_{k-1,j}^{k-1,j} \tag{C.68}$$

$$\delta(35) = 2(zt_{k+1,j} - zw_{k,j}) = \delta_{k+1,j}^{k,j} \tag{C.69}$$

$$\delta(36) = 2(zt_{k+1,j} - zw_{k,j}) = \delta^{k,j}_{k+1,j}, \tag{C.70}$$

$$\tag{C.71}$$

whereas the vertical spacing used to compute the volumes of these quarter cells is given by

$$\Delta(25) = \min\left(\delta^{k-1,j-1}_{k,j-1}, \delta^{k-1,j}_{k,j}\right) = \Delta^{(k-1,j-1)}_{(k,j-1)} \tag{C.72}$$

$$\Delta(26) = \min\left(\delta^{k-1,j}_{k,j}, \delta^{k-1,j+1}_{k,j+1}\right) = \Delta^{(k-1,j)}_{(k,j)} \tag{C.73}$$

$$\Delta(27) = \min\left(\delta^{k,j-1}_{k,j-1}, \delta^{k,j}_{k,j}\right) = \Delta^{(k,j-1)}_{(k,j-1)} \tag{C.74}$$

$$\Delta(28) = \min\left(\delta^{k,j}_{k,j}, \delta^{k,j+1}_{k,j+1}\right) = \Delta^{(k,j)}_{(k,j)} \tag{C.75}$$

$$\Delta(29) = \min\left(\delta^{k-1,j}_{k,j}, \delta^{k-1,j+1}_{k,j+1}\right) = \Delta^{(k-1,j)}_{(k,j)} \tag{C.76}$$

$$\Delta(30) = \min\left(\delta^{k,j}_{k,j}, \delta^{k,j+1}_{k,j+1}\right) = \Delta^{(k,j)}_{(k,j)} \tag{C.77}$$

$$\Delta(31) = \min\left(\delta^{k-1,j-1}_{k,j-1}, \delta^{k-1,j}_{k,j}\right) = \Delta^{(k-1,j-1)}_{(k,j-1)} \tag{C.78}$$

$$\Delta(32) = \min\left(\delta^{k,j-1}_{k,j-1}, \delta^{k,j}_{k,j}\right) = \Delta^{(k,j-1)}_{(k,j-1)} \tag{C.79}$$

$$\Delta(33) = \min\left(\delta^{k-1,j-1}_{k-1,j-1}, \delta^{k-1,j}_{k-1,j}\right) = \Delta^{(k-1,j-1)}_{(k-1,j-1)} \tag{C.80}$$

$$\Delta(34) = \min\left(\delta^{k-1,j}_{k-1,j}, \delta^{k-1,j+1}_{k-1,j+1}\right) = \Delta^{(k-1,j)}_{(k-1,j)} \tag{C.81}$$

$$\Delta(35) = \min\left(\delta^{k,j-1}_{k+1,j-1}, \delta^{k,j}_{k+1,j}\right) = \Delta^{(k,j-1)}_{(k+1,j-1)} \tag{C.82}$$

$$\Delta(36) = \min\left(\delta^{k,j}_{k+1,j}, \delta^{k,j+1}_{k+1,j+1}\right) = \Delta^{(k,j)}_{(k+1,j)}. \tag{C.83}$$

The i label has been omitted in these y-z expressions for purposes of brevity.

C.2.4 Quarter cell volumes

In order to discretize the functional, it is necessary to construct the volumes of the various subcells which partition the grid into the quarter-cells shown in Figures C.2, C.3, C.4 and C.5. In order to include the possibility of partial bottom cells, it is necessary to expose the three grid labels on the vertical grid factors. The considerations of the previous section are relevant for constructing the volumes of the quarter cells defined in the x-z and y-z planes.

C.2.4.1 x-y plane

On a spherical earth, the areas in the x-y plane for a domain of longitudinal width $\Delta\lambda$ and latitudinal width $\phi_2 - \phi_1$ is given by $(a\Delta\lambda)(a\sin\phi_2 - a\sin\phi_1)$, where a is the radius of the earth. For box 1 in Figure C.2, for example, $a\Delta\lambda = (1/2)dxu_{i-1}$ and so the volume of box 1 is

$$V(1) = (1/2)dxu_{i-1}(a\sin\phi^U_j - a\sin\phi^T_j)dht_{i,k,j}. \tag{C.84}$$

Taking $\phi^T_j = \phi^U_j - dyu_j/(2a)$ gives $\sin\phi^T_j = \sin\phi^U_j - \cos\phi^U_j dyu_j/(2a) + O(dyu_j/(2a))^2$. Hence, within this local β-plane approximation, the volume of box 1 is given by

$$V(1) = (1/4)dxu_{i-1}\, dyu_j\, \cos\phi^U_j\, dht_{i,k,j}. \tag{C.85}$$

The variance reduction property of the discretized diffusion operator is not sacrificed by making this approximation, and so the approximation will be taken in the following. Note that $dht_{i,k,j}$ was used for the vertical spacing of this quarter cell, consistent with quarter cell 1 being part of the T-cell $T_{i,k,j}$.

The above leads to the volumes for the 12 quarter cells in the x-y plane

$$V(1) = \frac{1}{4} dxu_{i-1} \cos\phi_j^U \; dyu_j \; dht_{i,k,j} \tag{C.86}$$

$$V(2) = \frac{1}{4} dxu_i \cos\phi_j^U \; dyu_j \; dht_{i,k,j} \tag{C.87}$$

$$V(3) = \frac{1}{4} dxu_{i-1} \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i,k,j} \tag{C.88}$$

$$V(4) = \frac{1}{4} dxu_i \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i,k,j} \tag{C.89}$$

$$V(5) = \frac{1}{4} dxu_i \cos\phi_j^U \; dyu_j \; dht_{i+1,k,j} \tag{C.90}$$

$$V(6) = \frac{1}{4} dxu_i \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i+1,k,j} \tag{C.91}$$

$$V(7) = \frac{1}{4} dxu_{i-1} \cos\phi_j^U \; dyu_j \; dht_{i-1,k,j} \tag{C.92}$$

$$V(8) = \frac{1}{4} dxu_{i-1} \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i-1,k,j} \tag{C.93}$$

$$V(9) = \frac{1}{4} dxu_{i-1} \cos\phi_j^U \; dyu_j \; dht_{i,k,j+1} \tag{C.94}$$

$$V(10) = \frac{1}{4} dxu_i \cos\phi_j^U \; dyu_j \; dht_{i,k,j+1} \tag{C.95}$$

$$V(11) = \frac{1}{4} dxu_{i-1} \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i,k,j-1} \tag{C.96}$$

$$V(12) = \frac{1}{4} dxu_i \cos\phi_{j-1}^U \; dyu_{j-1} \; dht_{i,k,j-1}. \tag{C.97}$$

Notice that when full vertical cells are used, the vertical spacing for each of these 12 quarter cells is the same: $dht_{i,k,j} \rightarrow dzt_k$. In this special case, the volumes for the 12 cells split into four groups of three: $V(1) = V(7) = V(9)$, $V(2) = V(5) = V(10)$, $V(3) = V(8) = V(11)$, and $V(4) = V(6) = V(12)$. Otherwise, the quarter cells all have different volumes.

C.2.4.2 x-z plane

For the x-z plane, there are no subtleties related to the spherical earth. Referring to Figure C.4 yields, without approximation, the volumes for the 12 quarter cells

$$V(13) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k)}^{(i-1,k-1)} \tag{C.98}$$

$$V(14) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k)}^{(i,k-1)} \tag{C.99}$$

$$V(15) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k)}^{(i-1,k)} \tag{C.100}$$

$$V(16) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k)}^{(i,k)} \tag{C.101}$$

$$V(17) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k)}^{(i,k-1)} \tag{C.102}$$

$$V(18) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k)}^{(i,k)} \tag{C.103}$$

$$V(19) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k)}^{(i-1,k-1)} \tag{C.104}$$

$$V(20) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k)}^{(i-1,k)} \tag{C.105}$$

$$V(21) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k-1)}^{(i-1,k-1)} \tag{C.106}$$

$$V(22) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k-1)}^{(i,k-1)} \tag{C.107}$$

$$V(23) = \frac{1}{4} dxu_{i-1} \cos\phi_j^T \; dyt_j \; \Delta_{(i-1,k+1)}^{(i-1,k)} \tag{C.108}$$

$$V(24) = \frac{1}{4} dxu_i \cos\phi_j^T \; dyt_j \; \Delta_{(i,k+1)}^{(i,k)}. \tag{C.109}$$

The j label has been omitted in the Δ expressions for purposes of brevity.

C.2.4.3 y-z plane

For the y-z plane, there are no subtleties related to the spherical earth. Referring to Figure C.5 yields, without approximation, the volumes for the 12 quarter cells

$$V(25) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \; \Delta_{(k,j-1)}^{(k-1,j-1)} \tag{C.110}$$

$$V(26) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k,j)}^{(k-1,j)} \tag{C.111}$$

$$V(27) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \; \Delta_{(k,j-1)}^{(k,j-1)} \tag{C.112}$$

$$V(28) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k,j)}^{(k,j)} \tag{C.113}$$

$$V(29) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k,j)}^{(k-1,j)} \tag{C.114}$$

$$V(30) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k,j)}^{(k,j)} \tag{C.115}$$

$$V(31) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \; \Delta_{(k,j-1)}^{(k-1,j-1)} \tag{C.116}$$

$$V(32) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \; \Delta_{(k,j-1)}^{(k,j-1)} \tag{C.117}$$

$$V(33) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \; \Delta_{(k-1,j-1)}^{(k-1,j-1)} \tag{C.118}$$

$$V(34) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k-1,j)}^{(k-1,j)} \tag{C.119}$$

$$V(35) = \frac{1}{4} dxt_i \; \cos\phi_{j-1}^U dyu_{j-1} \Delta_{(k+1,j-1)}^{(k,j-1)} \tag{C.120}$$

$$V(36) = \frac{1}{4} dxt_i \; \cos\phi_j^U dyu_j \; \Delta_{(k+1,j)}^{(k,j)}. \tag{C.121}$$

The i label has been omitted in the Δ expressions for purposes of brevity.

C.2.5 Tracer gradients within the 36 quarter cells

Within the 36 quarter cells, the gradient of the tracer and density are required. The form for the grid difference operators are given here. There are further details regarding the reference points to be used in determining the density gradient, and these reference point issues are discussed in Section C.2.7. The difference operator approximating a derivative will be written

$$\delta_x T_{i,k,j} = \frac{T_{i+1,k,j} - T_{i,k,j}}{dxu_i \cos\phi_j^T} \tag{C.122}$$

$$\delta_y T_{i,k,j} = \frac{T_{i,k,j+1} - T_{i,k,j}}{dyu_j} \tag{C.123}$$

$$\delta_z T_{i,k,j} = \frac{T_{i,k,j} - T_{i,k+1,j}}{dhwt_{i,k,j}}. \tag{C.124}$$

On the staggered B-grid used in the model, these difference operators are located at the T-cell faces and provide second order accurate approximations to the continuous derivative operators. Notice how the $\cos\phi_j^T$ is absorbed into the δ_x operator. Also notice that for the partial cells, the horizontal gradients are computed according to the interpolation scheme described in Section C.2.3 and detailed in Section 26.2.3. For the sake of brevity, such interpolation will be implicit in the notation; i.e., each horizontal derivative next to the bottom will be assumed to be computed using this interpolation scheme.

C.2.5.1 x-y plane

For the x-y plane, suppressing the k-index, the tracer gradient in the 12 boxes is given by

$$\nabla T(1) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j} \tag{C.125}$$

$$\nabla T(2) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j} \tag{C.126}$$

$$\nabla T(3) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j} \tag{C.127}$$

$$\nabla T(4) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j} \tag{C.128}$$

$$\nabla T(5) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i+1,j} + \hat{k}\delta_z T_{i+1,j} \tag{C.129}$$

$$\nabla T(6) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i+1,j-1} + \hat{k}\delta_z T_{i+1,j} \tag{C.130}$$

$$\nabla T(7) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i-1,j} + \hat{k}\delta_z T_{i-1,j} \tag{C.131}$$

$$\nabla T(8) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i-1,j-1} + \hat{k}\delta_z T_{i-1,j} \tag{C.132}$$

$$\nabla T(9) = \hat{\imath}\delta_x T_{i-1,j+1} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j+1} \tag{C.133}$$

$$\nabla T(10) = \hat{\imath}\delta_x T_{i,j+1} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j+1} \tag{C.134}$$

$$\nabla T(11) = \hat{\imath}\delta_x T_{i-1,j-1} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j-1} \tag{C.135}$$

$$\nabla T(12) = \hat{\imath}\delta_x T_{i,j-1} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j-1}. \tag{C.136}$$

There is one thing to note regarding the z-derivative in these expressions. It is no longer centered on the i,j plane and so presents a problem. The only place this issue raises its head is in the $|\nabla\rho|^{-2}$ term for the full tensor, in which all three derivatives are required. A centering of the squared z-derivative will be prescribed, in which no computational modes are introduced and proper placement is given. This point will be addressed in the discussion of Equation (C.286); for now, it will be ignored.

The functional derivative of these x-y plane gradients is given by

$$\frac{\partial \nabla T(1)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.137}$$

$$\frac{\partial \nabla T(2)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.138}$$

$$\frac{\partial \nabla T(3)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.139}$$

$$\frac{\partial \nabla T(4)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.140}$$

$$\frac{\partial \nabla T(5)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} \tag{C.141}$$

$$\frac{\partial \nabla T(6)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} \tag{C.142}$$

$$\frac{\partial \nabla T(7)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} \tag{C.143}$$

$$\frac{\partial \nabla T(8)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} \tag{C.144}$$

$$\frac{\partial \nabla T(9)}{\partial T_{i,j}} = -\frac{\hat{\jmath}}{dyu_j} \tag{C.145}$$

$$\frac{\partial \nabla T(10)}{\partial T_{i,j}} = -\frac{\hat{\jmath}}{dyu_j} \tag{C.146}$$

$$\frac{\partial \nabla T(11)}{\partial T_{i,j}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{C.147}$$

$$\frac{\partial \nabla T(12)}{\partial T_{i,j}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{C.148}$$

C.2.5.2 x-z plane

For the x-z plane, suppressing the j-index, the tracer gradient in the 12 boxes is given by

$$\nabla T(13) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k-1} \tag{C.149}$$

$$\nabla T(14) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k-1} \tag{C.150}$$

$$\nabla T(15) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k} \tag{C.151}$$

$$\nabla T(16) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k} \tag{C.152}$$

$$\nabla T(17) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i+1,k} + \hat{k}\delta_z T_{i+1,k-1} \tag{C.153}$$

$$\nabla T(18) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i+1,k} + \hat{k}\delta_z T_{i+1,k} \tag{C.154}$$

$$\nabla T(19) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i-1,k} + \hat{k}\delta_z T_{i-1,k-1} \tag{C.155}$$

$$\nabla T(20) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i-1,k} + \hat{k}\delta_z T_{i-1,k} \tag{C.156}$$

$$\nabla T(21) = \hat{\imath}\delta_x T_{i-1,k-1} + \hat{\jmath}\delta_y T_{i,k-1} + \hat{k}\delta_z T_{i,k-1} \tag{C.157}$$

$$\nabla T(22) = \hat{\imath}\delta_x T_{i,k-1} + \hat{\jmath}\delta_y T_{i,k-1} + \hat{k}\delta_z T_{i,k-1} \tag{C.158}$$

$$\nabla T(23) = \hat{\imath}\delta_x T_{i-1,k+1} + \hat{\jmath}\delta_y T_{i,k+1} + \hat{k}\delta_z T_{i,k} \tag{C.159}$$

$$\nabla T(24) = \hat{\imath}\delta_x T_{i,k+1} + \hat{\jmath}\delta_y T_{i,k+1} + \hat{k}\delta_z T_{i,k}. \tag{C.160}$$

The functional derivative of these x-z plane gradients is given by

$$\frac{\partial \nabla T(13)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.161}$$

$$\frac{\partial \nabla T(14)}{\partial T_{i,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.162}$$

$$\frac{\partial \nabla T(15)}{\partial T_{i,k}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.163}$$

$$\frac{\partial \nabla T(16)}{\partial T_{i,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.164}$$

$$\frac{\partial \nabla T(17)}{\partial T_{i,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} \tag{C.165}$$

$$\frac{\partial \nabla T(18)}{\partial T_{i,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} \tag{C.166}$$

$$\frac{\partial \nabla T(19)}{\partial T_{i,k}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} \tag{C.167}$$

$$\frac{\partial \nabla T(20)}{\partial T_{i,k}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} \tag{C.168}$$

$$\frac{\partial \nabla T(21)}{\partial T_{i,k}} = -\frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.169}$$

$$\frac{\partial \nabla T(22)}{\partial T_{i,k}} = -\frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.170}$$

$$\frac{\partial \nabla T(23)}{\partial T_{i,k}} = \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.171}$$

$$\frac{\partial \nabla T(24)}{\partial T_{i,k}} = \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.172}$$

C.2.5.3 y-z plane

For the y-z plane, suppressing the i-index, the tracer gradient in the 12 boxes is given by

$$\nabla T(25) = \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j,k-1} \tag{C.173}$$

$$\nabla T(26) = \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j,k-1} \tag{C.174}$$

$$\nabla T(27) = \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j,k} \tag{C.175}$$

$$\nabla T(28) = \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j,k} \tag{C.176}$$

$$\nabla T(29) = \hat{\imath}\delta_x T_{j+1,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j+1,k-1} \tag{C.177}$$

$$\nabla T(30) = \hat{\imath}\delta_x T_{j+1,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j+1,k} \tag{C.178}$$

$$\nabla T(31) = \hat{\imath}\delta_x T_{j-1,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j-1,k-1} \tag{C.179}$$

$$\nabla T(32) = \hat{\imath}\delta_x T_{j-1,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j-1,k} \tag{C.180}$$
$$\nabla T(33) = \hat{\imath}\delta_x T_{j,k-1} + \hat{\jmath}\delta_y T_{j-1,k-1} + \hat{k}\delta_z T_{j,k-1} \tag{C.181}$$
$$\nabla T(34) = \hat{\imath}\delta_x T_{j,k-1} + \hat{\jmath}\delta_y T_{j,k-1} + \hat{k}\delta_z T_{j,k-1} \tag{C.182}$$
$$\nabla T(35) = \hat{\imath}\delta_x T_{j,k+1} + \hat{\jmath}\delta_y T_{j-1,k+1} + \hat{k}\delta_z T_{j,k} \tag{C.183}$$
$$\nabla T(36) = \hat{\imath}\delta_x T_{j,k+1} + \hat{\jmath}\delta_y T_{j,k+1} + \hat{k}\delta_z T_{j,k}. \tag{C.184}$$

The functional derivative of these y-z plane gradients is given by

$$\frac{\partial \nabla T(25)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} - \frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.185}$$
$$\frac{\partial \nabla T(26)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.186}$$
$$\frac{\partial \nabla T(27)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.187}$$
$$\frac{\partial \nabla T(28)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.188}$$
$$\frac{\partial \nabla T(29)}{\partial T_{j,k}} = -\frac{\hat{\jmath}}{dyu_j} \tag{C.189}$$
$$\frac{\partial \nabla T(30)}{\partial T_{j,k}} = -\frac{\hat{\jmath}}{dyu_j} \tag{C.190}$$
$$\frac{\partial \nabla T(31)}{\partial T_{j,k}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{C.191}$$
$$\frac{\partial \nabla T(32)}{\partial T_{j,k}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{C.192}$$
$$\frac{\partial \nabla T(33)}{\partial T_{j,k}} = -\frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.193}$$
$$\frac{\partial \nabla T(34)}{\partial T_{j,k}} = -\frac{\hat{k}}{dhwt_{i,k-1,j}} \tag{C.194}$$
$$\frac{\partial \nabla T(35)}{\partial T_{j,k}} = \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.195}$$
$$\frac{\partial \nabla T(36)}{\partial T_{j,k}} = \frac{\hat{k}}{dhwt_{i,k,j}} \tag{C.196}$$

C.2.6 Schematic form of the discretized functional

For each of the three planes, there are 12 components to the functional which contain contributions from the grid tracer value $T_{i,k,j}$. These 12 components correspond to the 12 quarter cells shown in Figures Figures C.2, C.3, C.4 and C.5. For example, the x-y plane functional can be written

$$\mathcal{F}^{(x-y)} = -\frac{1}{2}\sum_{i,k}\sum_{n=1}^{12} A(n)V(n)\frac{|\nabla T(n) \wedge \nabla\rho(n)|^2}{|\nabla\rho(n)|^2}$$

$$\equiv \sum_{i,k} L_{i,k}, \tag{C.197}$$

where the discretized quadratic form

$$\begin{aligned} L_{i,k} &= -\frac{1}{2}\sum_{n=1}^{12} A(n)V(n)\frac{|\nabla T(n)\wedge\nabla\rho(n)|^2}{|\nabla\rho(n)|^2} \\ &\equiv \sum_{n=1}^{12} L_{i,k}^{(n)} \end{aligned} \tag{C.198}$$

consists of 12 non-negative contributions, and $A(n)$ is the non-negative diffusion coefficient for each of the different quarter cells.

C.2.7 Reference points for computing the density gradients

Consider the component of the x-y plane functional arising from the first subcell (see Figure C.2), which has the form

$$L_{i,k,j}^{(1)} = -\frac{A(1)V(1)}{2|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,k,j}\delta_y\rho_{i,k,j} - \delta_y T_{i,k,j}\delta_x\rho_{i-1,k,j}\right)^2. \tag{C.199}$$

The numerator has contributions from density at the three grid points: $\rho_{i,k,j+1}, \rho_{i,k,j}$, and $\rho_{i-1,k,j}$. These three points form a triangle, or triad, in the x-y plane (see Figure C.2). A self-consistent and simple choice of reference point is the corner of this triad, which for this case is the T-cell point i, k, j. Such a choice is valid whether the triad is wholly in a full vertical cell region, or has a part in a partial cell. With this choice, each of the three densities appearing in a triad will employ the *same* thermal and saline partial derivatives which are referenced to this single point. Therefore, the density gradients for this particular contribution to the functional will be discretized as

$$\delta_x\rho_{i-1,k,j} = \alpha_{i,k,j}\,\delta_x\theta_{i-1,k,j} + \beta_{i,k,j}\,\delta_x S_{i-1,k,j} \equiv \delta_x\rho_{i-1,k,j}^{(i,k,j)} \tag{C.200}$$

$$\delta_y\rho_{i,k,j} = \alpha_{i,k,j}\,\delta_y\theta_{i,k,j} + \beta_{i,k,j}\,\delta_y S_{i,k,j} \equiv \delta_y\rho_{i,k,j}^{(i,k,j)}. \tag{C.201}$$

In this expression, the thermal and saline coefficients are written

$$\alpha_{i,k,j} = \rho_\theta\left(\theta_{i,k,j}, S_{i,k,j}, k\right) \tag{C.202}$$

$$\beta_{i,k,j} = \rho_S\left(\theta_{i,k,j}, S_{i,k,j}, k\right), \tag{C.203}$$

where the arguments of the density partial derivatives indicate the value of the potential temperature, salinity, and pressure reference level k for use in their computation. MOM generally employs a cubic approximation to the UNESCO equation of state. Once this approximation is determined, the quadratic expressions for the partial derivatives ρ_θ and ρ_S are found and tabulated along with the cubic equation of state. The superscripts introduced on the density gradients allow for a compact notation which exposes the information about the referece point. Parentheses are included in this notation to help distinguish it from the subscripts referring to explicit grid points. Finally, the discretization of the denominator is prescribed to be consistent with that determined by the numerator. Note that only the x and y gradients are explicitly prescribed since the z gradient does not appear in the numerator. This ambiguity will be settled at a later stage of the derivation. For the present purposes, let

$$(\nabla\rho(1))^2 = (\delta_x\rho_{i-1,k,j}^{(i,k,j)})^2 + (\delta_y\rho_{i,k,j}^{(i,k,j)})^2 + (\delta_z\rho_{i,k,j}^{(i,k,j)})^2. \tag{C.204}$$

Note that the new implementation of the isoneutral diffusion operator employs the same number of reference levels as there are depth levels. This allows for the most accurate approximation to the neutral directions as possible with the particular vertical resolution. Such a procedure was also employed in the original Cox (1987) implementation of isoneutral diffusion.

In an early derivation of the new discretized diffusion operator, the issues of reference points were ignored until the very end of the derivation. At that point, reasonable choices were made for choosing the reference points, which consisted of referencing on the various sides of the T-cells consistent with the location of the diffusive fluxes. However, it was soon realized that the numerical constraint of defining a dissipative diffusion operator was not satisfied by this choice. Rather, in order to ensure a dissipative operator, *all* choices of discretization should be made in the framework of the discretized functional. As noted earlier, the power of this perspective is that whatever the discretization used for the functional, the resulting diffusion operator derived from taking the functional derivative will be dissipative.

C.2.8 Piecing together the discretized functional

The different components of the functional have now been considered, and so it is time to piece things together.

C.2.8.1 Functional in the x-y plane

The x-y plane functional is relevant only for the full tensor as it vanishes in the small angle limit. The quadratic form which contains contributions from the T-point $T_{i,k,j}$ in the x-y plane is given by the sum of the following 12 terms (k index suppressed)

$$L^{(1)}_{i,j} = -\frac{A(1)V(1)}{2|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i-1,j}\right)^2 \tag{C.205}$$

$$L^{(2)}_{i,j} = -\frac{A(2)V(2)}{2|\nabla\rho(2)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i,j}\right)^2 \tag{C.206}$$

$$L^{(3)}_{i,j} = -\frac{A(3)V(3)}{2|\nabla\rho(3)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i-1,j}\right)^2 \tag{C.207}$$

$$L^{(4)}_{i,j} = -\frac{A(4)V(4)}{2|\nabla\rho(4)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j}\right)^2 \tag{C.208}$$

$$L^{(5)}_{i,j} = -\frac{A(5)V(5)}{2|\nabla\rho(5)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i+1,j)}_{i+1,j} - \delta_y T_{i+1,j}\delta_x\rho^{(i+1,j)}_{i,j}\right)^2 \tag{C.209}$$

$$L^{(6)}_{i,j} = -\frac{A(6)V(6)}{2|\nabla\rho(6)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i+1,j)}_{i+1,j-1} - \delta_y T_{i+1,j-1}\delta_x\rho^{(i+1,j)}_{i,j}\right)^2 \tag{C.210}$$

$$L^{(7)}_{i,j} = -\frac{A(7)V(7)}{2|\nabla\rho(7)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i-1,j)}_{i-1,j} - \delta_y T_{i-1,j}\delta_x\rho^{(i-1,j)}_{i-1,j}\right)^2 \tag{C.211}$$

$$L^{(8)}_{i,j} = -\frac{A(8)V(8)}{2|\nabla\rho(8)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i-1,j)}_{i-1,j-1} - \delta_y T_{i-1,j-1}\delta_x\rho^{(i-1,j)}_{i-1,j}\right)^2 \tag{C.212}$$

$$L^{(9)}_{i,j} = -\frac{A(9)V(9)}{2|\nabla\rho(9)|^2}\left(\delta_x T_{i-1,j+1}\delta_y\rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j+1)}_{i-1,j+1}\right)^2 \tag{C.213}$$

$$L^{(10)}_{i,j} = -\frac{A(10)V(10)}{2|\nabla\rho(10)|^2}\left(\delta_x T_{i,j+1}\delta_y\rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j+1)}_{i,j+1}\right)^2 \tag{C.214}$$

$$L_{i,j}^{(11)} = -\frac{A(11)V(11)}{2|\nabla\rho(11)|^2}\left(\delta_x T_{i-1,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j-1}^{(i,j-1)}\right)^2 \tag{C.215}$$

$$L_{i,j}^{(12)} = -\frac{A(12)V(12)}{2|\nabla\rho(12)|^2}\left(\delta_x T_{i,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i,j-1}^{(i,j-1)}\right)^2 \tag{C.216}$$

C.2.8.2 Functional in the x-z plane

The x-z plane quadratic form contains contributions from the T-point $T_{i,k,j}$ in the following 12 terms (j index suppressed)

$$L_{i,k}^{(13)} = -\frac{A(13)V(13)}{2|\nabla\rho(13)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}\right)^2 \tag{C.217}$$

$$L_{i,k}^{(14)} = -\frac{A(14)V(14)}{2|\nabla\rho(14)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k}^{(i,k)} - \delta_x T_{i,k}\delta_z\rho_{i,k-1}^{(i,k)}\right)^2 \tag{C.218}$$

$$L_{i,k}^{(15)} = -\frac{A(15)V(15)}{2|\nabla\rho(15)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k}^{(i,k)}\right)^2 \tag{C.219}$$

$$L_{i,k}^{(16)} = -\frac{A(16)V(16)}{2|\nabla\rho(16)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k}^{(i,k)} - \delta_x T_{i,k}\delta_z\rho_{i,k}^{(i,k)}\right)^2 \tag{C.220}$$

$$L_{i,k}^{(17)} = -\frac{A(17)V(17)}{2|\nabla\rho(17)|^2}\left(\delta_z T_{i+1,k-1}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k-1}^{(i+1,k)}\right)^2 \tag{C.221}$$

$$L_{i,k}^{(18)} = -\frac{A(18)V(18)}{2|\nabla\rho(18)|^2}\left(\delta_z T_{i+1,k}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k}^{(i+1,k)}\right)^2 \tag{C.222}$$

$$L_{i,k}^{(19)} = -\frac{A(19)V(19)}{2|\nabla\rho(19)|^2}\left(\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}\right)^2 \tag{C.223}$$

$$L_{i,k}^{(20)} = -\frac{A(20)V(20)}{2|\nabla\rho(20)|^2}\left(\delta_z T_{i-1,k}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k}^{(i-1,k)}\right)^2 \tag{C.224}$$

$$L_{i,k}^{(21)} = -\frac{A(21)V(21)}{2|\nabla\rho(21)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right)^2 \tag{C.225}$$

$$L_{i,k}^{(22)} = -\frac{A(22)V(22)}{2|\nabla\rho(22)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k-1}^{(i,k-1)} - \delta_x T_{i,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right)^2 \tag{C.226}$$

$$L_{i,k}^{(23)} = -\frac{A(23)V(23)}{2|\nabla\rho(23)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k+1}^{(i,k+1)} - \delta_x T_{i-1,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right)^2 \tag{C.227}$$

$$L_{i,k}^{(24)} = -\frac{A(24)V(24)}{2|\nabla\rho(24)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k+1}^{(i,k+1)} - \delta_x T_{i,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right)^2 \tag{C.228}$$

C.2.8.3 Functional in the y-z plane

The y-z plane quadratic form contains contributions from the T-point $T_{i,k,j}$ in the following 12 terms (i index suppressed)

$$L_{j,k}^{(25)} = -\frac{A(25)V(25)}{2|\nabla\rho(25)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k-1}^{(j,k)}\right)^2 \tag{C.229}$$

$$L_{j,k}^{(26)} = -\frac{A(26)V(26)}{2|\nabla\rho(26)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k-1}^{(j,k)}\right)^2 \tag{C.230}$$

$$L_{j,k}^{(27)} = -\frac{A(27)V(27)}{2|\nabla\rho(27)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k}^{(j,k)}\right)^2 \tag{C.231}$$

$$L_{j,k}^{(28)} = -\frac{A(28)V(28)}{2|\nabla\rho(28)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k}^{(j,k)}\right)^2 \tag{C.232}$$

$$L_{j,k}^{(29)} = -\frac{A(29)V(29)}{2|\nabla\rho(29)|^2}\left(\delta_z T_{j+1,k-1}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k-1}^{(j+1,k)}\right)^2 \tag{C.233}$$

$$L_{j,k}^{(30)} = -\frac{A(30)V(30)}{2|\nabla\rho(30)|^2}\left(\delta_z T_{j+1,k}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k}^{(j+1,k)}\right)^2 \tag{C.234}$$

$$L_{j,k}^{(31)} = -\frac{A(31)V(31)}{2|\nabla\rho(31)|^2}\left(\delta_z T_{j-1,k-1}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k-1}^{(j-1,k)}\right)^2 \tag{C.235}$$

$$L_{j,k}^{(32)} = -\frac{A(32)V(32)}{2|\nabla\rho(32)|^2}\left(\delta_z T_{j-1,k}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k}^{(j-1,k)}\right)^2 \tag{C.236}$$

$$L_{j,k}^{(33)} = -\frac{A(33)V(33)}{2|\nabla\rho(33)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k-1}^{(j,k-1)} - \delta_y T_{j-1,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right)^2 \tag{C.237}$$

$$L_{j,k}^{(34)} = -\frac{A(34)V(34)}{2|\nabla\rho(34)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k-1}^{(j,k-1)} - \delta_y T_{j,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right)^2 \tag{C.238}$$

$$L_{j,k}^{(35)} = -\frac{A(35)V(35)}{2|\nabla\rho(35)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k+1}^{(j,k+1)} - \delta_y T_{j-1,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right)^2 \tag{C.239}$$

$$L_{j,k}^{(36)} = -\frac{A(36)V(36)}{2|\nabla\rho(36)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k+1}^{(j,k+1)} - \delta_y T_{j,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right)^2 \tag{C.240}$$

C.2.9 Slope constraint

The diffusion coefficient $A(n)$ for a particular quarter cell will be chosen to satisfy the numerical stability constraint on the linear diffusion equation (Cox, 1987, Griffies *et al.* 1998). For the small angle tensor, the prescription of Gerdes et al (1991) is the MOM default, in which for large slopes, the diffusion coefficient is quadratically rescaled to a smaller value. Their prescription applied to the present formulation says that for each quarter cell, if the slope $|S(n)| = |\delta_{x_i}\rho/\delta_z\rho| > \delta$, where δ is the maximum slope prescribed by the researcher through namelist, then the corresponding diffusion coefficient $A(n)$ is rescaled as $A(n) \rightarrow A(n)(\delta/S(n))^2$. This prescription provides a unique definition of the diffusion coefficient and thus provides for a unique definition of the diffusive fluxes. An alternative used by Danabasoglu and McWilliams (1995) suggests a hyperobolic tangent rescaling. This second rescaling is implemented in MOM as the option *dm_taper* (Section 35.1). For the full tensor, the rescaling is $A(n) \rightarrow \delta\ A(n)(|S(n)| + |S(n)|^{-1})$ for $S_{(-1)} < |S(n)| < S_{(-1)}^{-1}$ (Griffies *et al.* 1998). For the special case of a grid in which $\delta > 1/2$, the full tensor requires no rescaling of the diffusion coefficients in order to maintain numerical stability. Slope constraints are discussed more completely in Griffies *et al.* (1998). Note that the original slope clipping scheme of Cox (1987) is not supported anymore due to its potential to introduce unphysically large diapycnal fluxes and to interact with convection (Griffies *et al.* 1998).

C.2.10 Derivative of the functional

In order to construct the diffusion operator, it is necessary to take the functional derivative of the functional $\mathcal{F}$ (Griffies *et al.* 1998). On the lattice, the functional derivative is simply a partial derivative of $\mathcal{F}$ with respect to the tracer $T_{i,k,j}$. When taking this derivative, all terms in the functional which are independent of $T_{i,k,j}$ drop out. Hence, it is only necessary to consider that part of $\mathcal{F}$ which includes some contribution from $T_{i,k,j}$. It is this fact which motivated the

focus on just the 36 quarter cells shown in Figures C.2, C.3, C.4 and C.5. The calculation in this section is straightforward, but somewhat tedious.

C.2.10.1 x-y plane

Taking the derivative of the 12 contributions to the functional in the x-y plane yields

$$\begin{aligned}
\frac{\partial L^{(1)}_{i,j}}{\partial T_{i,k,j}} &= -\frac{A(1)V(1)}{|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i-1,j}\right) \\
&\times \left(\frac{\delta_y\rho^{(i,j)}_{i,j}}{\cos\phi^T_j dxu_{i-1}} + \frac{\delta_x\rho^{(i,j)}_{i-1,j}}{dyu_j}\right) \\
&= -\frac{A(1)\ \cos\phi^U_j\ dht_{i,k,j}}{4|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i-1,j}\right) \\
&\times \left(\frac{dyu_j}{\cos\phi^T_j}\ \delta_y\rho^{(i,j)}_{i,j} + dxu_{i-1}\ \delta_x\rho^{(i,j)}_{i-1,j}\right) \qquad \text{(C.241)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L^{(2)}_{i,j}}{\partial T_{i,k,j}} &= -\frac{A(2)V(2)}{|\nabla\rho(2)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i,j}\right) \\
&\times \left(-\frac{\delta_y\rho^{(i,j)}_{i,j}}{\cos\phi^T_j dxu_i} + \frac{\delta_x\rho^{(i,j)}_{i,j}}{dyu_j}\right) \\
&= -\frac{A(2)\ \cos\phi^U_j\ dht_{i,k,j}}{4|\nabla\rho(2)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i,j}\right) \\
&\times \left(-\frac{dyu_j}{\cos\phi^T_j}\ \delta_y\rho^{(i,j)}_{i,j} + dxu_i\ \delta_x\rho^{(i,j)}_{i,j}\right) \qquad \text{(C.242)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L^{(3)}_{i,j}}{\partial T_{i,k,j}} &= -\frac{A(3)V(3)}{|\nabla\rho(3)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i-1,j}\right) \\
&\times \left(\frac{\delta_y\rho^{(i,j)}_{i,j-1}}{\cos\phi^T_j dxu_{i-1}} - \frac{\delta_x\rho^{(i,j)}_{i-1,j}}{dyu_{j-1}}\right) \\
&= -\frac{A(3)\ \cos\phi^U_{j-1}\ dht_{i,k,j}}{4|\nabla\rho(3)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i-1,j}\right) \\
&\times \left(\frac{dyu_{j-1}}{\cos\phi^T_j}\ \delta_y\rho^{(i,j)}_{i,j-1} - dxu_{i-1}\ \delta_x\rho^{(i,j)}_{i-1,j}\right) \qquad \text{(C.243)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L^{(4)}_{i,j}}{\partial T_{i,k,j}} &= -\frac{A(4)V(4)}{|\nabla\rho(4)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j}\right) \\
&\times \left(-\frac{\delta_y\rho^{(i,j)}_{i,j-1}}{\cos\phi^T_j dxu_i} - \frac{\delta_x\rho^{(i,j)}_{i,j}}{dyu_{j-1}}\right) \\
&= \frac{A(4)\ \cos\phi^U_{j-1}\ dht_{i,k,j}}{4|\nabla\rho(4)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j}\right)
\end{aligned}$$

$$\times \quad \left(\frac{dyu_{j-1}}{\cos\phi_j^T}\ \delta_y \rho_{i,j-1}^{(i,j)} + dxu_i\ \delta_x \rho_{i,j}^{(i,j)} \right) \tag{C.244}$$

$$\begin{aligned}
\frac{\partial L_{i,j}^{(5)}}{\partial T_{i,k,j}} &= -\frac{A(5)V(5)}{|\nabla\rho(5)|^2} \left(\delta_x T_{i,j} \delta_y \rho_{i+1,j}^{(i+1,j)} - \delta_y T_{i+1,j} \delta_x \rho_{i,j}^{(i+1,j)} \right) \\
&\times \left(-\frac{\delta_y \rho_{i+1,j}^{(i+1,j)}}{\cos\phi_j^T dxu_i} \right) \\
&= \frac{A(5)\ \cos\phi_j^U\ dht_{i+1,k,j}}{4\ |\nabla\rho(5)|^2} \left(\delta_x T_{i,j} \delta_y \rho_{i+1,j}^{(i+1,j)} - \delta_y T_{i+1,j} \delta_x \rho_{i,j}^{(i+1,j)} \right) \\
&\times \left(\frac{dyu_j}{\cos\phi_j^T}\ \delta_y \rho_{i+1,j}^{(i+1,j)} \right)
\end{aligned} \tag{C.245}$$

$$\begin{aligned}
\frac{\partial L_{i,j}^{(6)}}{\partial T_{i,k,j}} &= -\frac{A(6)V(6)}{|\nabla\rho(6)|^2} \left(\delta_x T_{i,j} \delta_y \rho_{i+1,j-1}^{(i+1,j)} - \delta_y T_{i+1,j-1} \delta_x \rho_{i,j}^{(i+1,j)} \right) \\
&\times \left(-\frac{\delta_y \rho_{i+1,j-1}^{(i+1,j)}}{\cos\phi_j^T dxu_i} \right) \\
&= \frac{A(6)\ \cos\phi_{j-1}^U\ dht_{i+1,k,j}}{4|\nabla\rho(6)|^2} \left(\delta_x T_{i,j} \delta_y \rho_{i+1,j-1}^{(i+1,j)} - \delta_y T_{i+1,j-1} \delta_x \rho_{i,j}^{(i+1,j)} \right) \\
&\times \left(\frac{dyu_{j-1}}{\cos\phi_j^T}\ \delta_y \rho_{i+1,j-1}^{(i+1,j)} \right)
\end{aligned} \tag{C.246}$$

$$\begin{aligned}
\frac{\partial L_{i,j}^{(7)}}{\partial T_{i,k,j}} &= -\frac{A(7)V(7)}{|\nabla\rho(7)|^2} \left(\delta_x T_{i-1,j} \delta_y \rho_{i-1,j}^{(i-1,j)} - \delta_y T_{i-1,j} \delta_x \rho_{i-1,j}^{(i-1,j)} \right) \\
&\times \left(\frac{\delta_y \rho_{i-1,j}^{(i-1,j)}}{\cos\phi_j^T dxu_{i-1}} \right) \\
&= -\frac{A(7)\ \cos\phi_j^U\ dht_{i-1,k,j}}{4|\nabla\rho(7)|^2} \left(\delta_x T_{i-1,j} \delta_y \rho_{i-1,j}^{(i-1,j)} - \delta_y T_{i-1,j} \delta_x \rho_{i-1,j}^{(i-1,j)} \right) \\
&\times \left(\frac{dyu_j}{\cos\phi_j^T}\ \delta_y \rho_{i-1,j}^{(i-1,j)} \right)
\end{aligned} \tag{C.247}$$

$$\begin{aligned}
\frac{\partial L_{i,j}^{(8)}}{\partial T_{i,k,j}} &= -\frac{A(8)V(8)}{|\nabla\rho(8)|^2} \left(\delta_x T_{i-1,j} \delta_y \rho_{i-1,j-1}^{(i-1,j)} - \delta_y T_{i-1,j-1} \delta_x \rho_{i-1,j}^{(i-1,j)} \right) \\
&\times \left(\frac{\delta_y \rho_{i-1,j-1}^{(i-1,j)}}{\cos\phi_j^T dxu_{i-1}} \right) \\
&= -\frac{A(8)\ \cos\phi_{j-1}^U\ dht_{i-1,k,j}}{4|\nabla\rho(8)|^2} \left(\delta_x T_{i-1,j} \delta_y \rho_{i-1,j-1}^{(i-1,j)} - \delta_y T_{i-1,j-1} \delta_x \rho_{i-1,j}^{(i-1,j)} \right) \\
&\times \left(\frac{dyu_{j-1}}{\cos\phi_j^T}\ \delta_y \rho_{i-1,j-1}^{(i-1,j)} \right)
\end{aligned} \tag{C.248}$$

$$
\begin{aligned}
\frac{\partial L_{i,j}^{(9)}}{\partial T_{i,k,j}} &= -\frac{A(9)V(9)}{|\nabla\rho(9)|^2}\left(\delta_x T_{i-1,j+1}\delta_y\rho_{i,j}^{(i,j+1)} - \delta_y T_{i,j}\delta_x\rho_{i-1,j+1}^{(i,j+1)}\right) \\
&\times \left(\frac{\delta_x\rho_{i-1,j+1}^{(i,j+1)}}{dyu_j}\right) \\
&= -\frac{A(9)\ \cos\phi_j^U\ dht_{i,k,j+1}}{4|\nabla\rho(9)|^2}\left(\delta_x T_{i-1,j+1}\delta_y\rho_{i,j}^{(i,j+1)} - \delta_y T_{i,j}\delta_x\rho_{i-1,j+1}^{(i,j+1)}\right) \\
&\times \left(dxu_{i-1}\ \delta_x\rho_{i-1,j+1}^{(i,j+1)}\right) \qquad \text{(C.249)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,j}^{(10)}}{\partial T_{i,k,j}} &= -\frac{A(10)V(10)}{|\nabla\rho(10)|^2}\left(\delta_x T_{i,j+1}\delta_y\rho_{i,j}^{(i,j+1)} - \delta_y T_{i,j}\delta_x\rho_{i,j+1}^{(i,j+1)}\right) \\
&\times \left(\frac{\delta_x\rho_{i,j+1}^{(i,j+1)}}{dyu_j}\right) \\
&= -\frac{A(10)\ \cos\phi_j^U\ dht_{i,k,j+1}}{4|\nabla\rho(10)|^2}\left(\delta_x T_{i,j+1}\delta_y\rho_{i,j}^{(i,j+1)} - \delta_y T_{i,j}\delta_x\rho_{i,j+1}^{(i,j+1)}\right) \\
&\times \left(dxu_i\ \delta_x\rho_{i,j+1}^{(i,j+1)}\right) \qquad \text{(C.250)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,j}^{(11)}}{\partial T_{i,k,j}} &= -\frac{A(11)V(11)}{|\nabla\rho(11)|^2}\left(\delta_x T_{i-1,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j-1}^{(i,j-1)}\right) \\
&\times \left(-\frac{\delta_x\rho_{i-1,j-1}^{(i,j-1)}}{dyu_{j-1}}\right) \\
&= \frac{A(11)\ \cos\phi_{j-1}^U\ dht_{i,k,j-1}}{4|\nabla\rho(11)|^2}\left(\delta_x T_{i-1,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j-1}^{(i,j-1)}\right) \\
&\times \left(dxu_{i-1}\ \delta_x\rho_{i-1,j-1}^{(i,j-1)}\right) \qquad \text{(C.251)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,j}^{(12)}}{\partial T_{i,k,j}} &= -\frac{A(12)V(12)}{|\nabla\rho(12)|^2}\left(\delta_x T_{i,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i,j-1}^{(i,j-1)}\right) \\
&\times \left(-\frac{\delta_x\rho_{i,j-1}^{(i,j-1)}}{dyu_{j-1}}\right) \\
&= \frac{A(12)\ \cos\phi_{j-1}^U dht_{i,k,j-1}}{4|\nabla\rho(12)|^2}\left(\delta_x T_{i,j-1}\delta_y\rho_{i,j-1}^{(i,j-1)} - \delta_y T_{i,j-1}\delta_x\rho_{i,j-1}^{(i,j-1)}\right) \\
&\times \left(dxu_i\ \delta_x\rho_{i,j-1}^{(i,j-1)}\right). \qquad \text{(C.252)}
\end{aligned}
$$

C.2.10.2 x-z plane

Taking the derivative of the 12 contributions to the functional in the x-z plane yields

$$
\frac{\partial L_{i,k}^{(13)}}{\partial T_{i,k,j}} = -\frac{A(13)V(13)}{|\nabla\rho(13)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}\right)
$$

$$
\begin{aligned}
&\times \left(-\frac{\delta_x \rho_{i-1,k}^{(i,k)}}{dhwt_{i,k-1,j}} - \frac{\delta_z \rho_{i,k-1}^{(i,k)}}{\cos\phi_j^T dxu_{i-1}} \right) \\
&= \frac{A(13)\, dyt_j\, \Delta_{(i-1,k)}^{(i-1,k-1)}}{4|\nabla\rho(13)|^2} \left(\delta_z T_{i,k-1} \delta_x \rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k} \delta_z \rho_{i,k-1}^{(i,k)} \right) \\
&\times \left(\frac{dxu_{i-1}\, \cos\phi_j^T}{dhwt_{i,k-1,j}}\, \delta_x \rho_{i-1,k}^{(i,k)} + \delta_z \rho_{i,k-1}^{(i,k)} \right) \qquad \text{(C.253)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(14)}}{\partial T_{i,k,j}} &= -\frac{A(14)V(14)}{|\nabla\rho(14)|^2} \left(\delta_z T_{i,k-1} \delta_x \rho_{i,k}^{(i,k)} - \delta_x T_{i,k} \delta_z \rho_{i,k-1}^{(i,k)} \right) \\
&\times \left(-\frac{\delta_x \rho_{i,k}^{(i,k)}}{dhwt_{i,k-1,j}} + \frac{\delta_z \rho_{i,k-1}^{(i,k)}}{\cos\phi_j^T dxu_i} \right) \\
&= -\frac{A(14)\, dyt_j\, \Delta_{(i,k)}^{(i,k-1)}}{4|\nabla\rho(14)|^2} \left(\delta_z T_{i,k-1} \delta_x \rho_{i,k}^{(i,k)} - \delta_x T_{i,k} \delta_z \rho_{i,k-1}^{(i,k)} \right) \\
&\times \left(-\frac{dxu_i\, \cos\phi_j^T}{dhwt_{i,k-1,j}}\, \delta_x \rho_{i,k}^{(i,k)} + \delta_z \rho_{i,k-1}^{(i,k)} \right) \qquad \text{(C.254)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(15)}}{\partial T_{i,k,j}} &= -\frac{A(15)V(15)}{|\nabla\rho(15)|^2} \left(\delta_z T_{i,k} \delta_x \rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k} \delta_z \rho_{i,k}^{(i,k)} \right) \\
&\times \left(\frac{\delta_x \rho_{i-1,k}^{(i,k)}}{dhwt_{i,k,j}} - \frac{\delta_z \rho_{i,k}^{(i,k)}}{\cos\phi_j^T dxu_{i-1}} \right) \\
&= -\frac{A(15)\, dyt_j\, \Delta_{(i-1,k)}^{(i-1,k)}}{4|\nabla\rho(15)|^2} \left(\delta_z T_{i,k} \delta_x \rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k} \delta_z \rho_{i,k}^{(i,k)} \right) \\
&\times \left(\frac{dxu_{i-1}\, \cos\phi_j^T}{dhwt_{i,k,j}}\, \delta_x \rho_{i-1,k}^{(i,k)} - \delta_z \rho_{i,k}^{(i,k)} \right) \qquad \text{(C.255)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(16)}}{\partial T_{i,k,j}} &= -\frac{A(16)V(16)}{|\nabla\rho(16)|^2} \left(\delta_z T_{i,k} \delta_x \rho_{i,k}^{(i,k)} - \delta_x T_{i,k} \delta_z \rho_{i,k}^{(i,k)} \right) \\
&\times \left(\frac{\delta_x \rho_{i,k}^{(i,k)}}{dhwt_{i,k,j}} + \frac{\delta_z \rho_{i,k}^{(i,k)}}{\cos\phi_j^T dxu_i} \right) \\
&= -\frac{A(16)\, dyt_j\, \Delta_{(i,k)}^{(i,k)}}{4|\nabla\rho(16)|^2} \left(\delta_z T_{i,k} \delta_x \rho_{i,k}^{(i,k)} - \delta_x T_{i,k} \delta_z \rho_{i,k}^{(i,k)} \right) \\
&\times \left(\frac{dxu_i\, \cos\phi_j^T}{dhwt_{i,k,j}}\, \delta_x \rho_{i,k}^{(i,k)} + \delta_z \rho_{i,k}^{(i,k)} \right) \qquad \text{(C.256)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(17)}}{\partial T_{i,k,j}} &= -\frac{A(17)V(17)}{|\nabla\rho(17)|^2} \left(\delta_z T_{i+1,k-1} \delta_x \rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k} \delta_z \rho_{i+1,k-1}^{(i+1,k)} \right) \\
&\times \left(\frac{\delta_z \rho_{i+1,k-1}^{(i+1,k)}}{\cos\phi_j^T dxu_i} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= -\frac{A(17)\ dyt_j\ \Delta_{(i,k)}^{(i,k-1)}}{4|\nabla\rho(17)|^2}\left(\delta_z T_{i+1,k-1}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k-1}^{(i+1,k)}\right) \\
&\times \left(dyt_j\ \delta_z\rho_{i+1,k-1}^{(i+1,k)}\right) \qquad \text{(C.257)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(18)}}{\partial T_{i,k,j}} &= -\frac{A(18)V(18)}{|\nabla\rho(18)|^2}\left(\delta_z T_{i+1,k}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k}^{(i+1,k)}\right) \\
&\times \left(\frac{\delta_z\rho_{i+1,k}^{(i+1,k)}}{\cos\phi_j^T dxu_i}\right) \\
&= -\frac{A(18)\ \Delta_{(i,k)}^{(i,k)}}{4|\nabla\rho(18)|^2}\left(\delta_z T_{i+1,k}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k}^{(i+1,k)}\right) \\
&\times \left(dyt_j\ \delta_z\rho_{i+1,k}^{(i+1,k)}\right) \qquad \text{(C.258)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(19)}}{\partial T_{i,k,j}} &= -\frac{A(19)V(19)}{|\nabla\rho(19)|^2}\left(\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{i-1,k-1}^{(i-1,k)}}{\cos\phi_j^T dxu_{i-1}}\right) \\
&= \frac{A(19)\ \Delta_{(i-1,k)}^{(i-1,k-1)}}{4|\nabla\rho(19)|^2}\left(\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}\right) \\
&\times \left(dyt_j\ \delta_z\rho_{i-1,k-1}^{(i-1,k)}\right) \qquad \text{(C.259)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(20)}}{\partial T_{i,k,j}} &= -\frac{A(20)V(20)}{|\nabla\rho(20)|^2}\left(\delta_z T_{i-1,k}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k}^{(i-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{i-1,k}^{(i-1,k)}}{\cos\phi_j^T dxu_{i-1}}\right) \\
&= \frac{A(20)\ \Delta_{(i-1,k)}^{(i-1,k)}}{4|\nabla\rho(20)|^2}\left(\delta_z T_{i-1,k}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k}^{(i-1,k)}\right) \\
&\times \left(dyt_j\ \delta_z\rho_{i-1,k}^{(i-1,k)}\right) \qquad \text{(C.260)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(21)}}{\partial T_{i,k,j}} &= -\frac{A(21)V(21)}{|\nabla\rho(21)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \\
&\times \left(-\frac{\delta_x\rho_{i-1,k-1}^{(i,k-1)}}{dhwt_{i,k-1,j}}\right) \\
&= \frac{A(21)\Delta_{(i-1,k-1)}^{(i-1,k-1)}}{4|\nabla\rho(21)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \\
&\times \left(\frac{dxu_{i-1}\ \cos\phi_j^T\ dyt_j}{dhwt_{i,k-1,j}}\ \delta_x\rho_{i-1,k-1}^{(i,k-1)}\right) \qquad \text{(C.261)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(22)}}{\partial T_{i,k,j}} &= -\frac{A(22)V(22)}{|\nabla\rho(22)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k-1}^{(i,k-1)} - \delta_x T_{i,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \\
&\times \left(-\frac{\delta_x\rho_{i,k-1}^{(i,k-1)}}{dhwt_{i,k-1,j}}\right) \\
&= \frac{A(22)\,\Delta_{(i,k-1)}^{(i,k-1)}}{4|\nabla\rho(22)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k-1}^{(i,k-1)} - \delta_x T_{i,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \\
&\times \left(\frac{dxu_i\,\cos\phi_j^T\,dyt_j}{dhwt_{i,k-1,j}}\,\delta_x\rho_{i,k-1}^{(i,k-1)}\right) \qquad \text{(C.262)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(23)}}{\partial T_{i,k,j}} &= -\frac{A(23)V(23)}{|\nabla\rho(23)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k+1}^{(i,k+1)} - \delta_x T_{i-1,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \\
&\times \left(\frac{\delta_x\rho_{i-1,k+1}^{(i,k+1)}}{dhwt_{i,k,j}}\right) \\
&= -\frac{A(23)\,\Delta_{(i-1,k+1)}^{(i-1,k)}}{4|\nabla\rho(23)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k+1}^{(i,k+1)} - \delta_x T_{i-1,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \\
&\times \left(\frac{dxu_{i-1}\,\cos\phi_j^T\,dyt_j}{dhwt_{i,k,j}}\,\delta_x\rho_{i-1,k+1}^{(i,k+1)}\right) \qquad \text{(C.263)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{i,k}^{(24)}}{\partial T_{i,k,j}} &= -\frac{A(24)V(24)}{|\nabla\rho(24)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k+1}^{(i,k+1)} - \delta_x T_{i,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \\
&\times \left(\frac{\delta_x\rho_{i,k+1}^{(i,k+1)}}{dhwt_{i,k,j}}\right) \\
&= -\frac{A(24)\,\Delta_{(i,k+1)}^{(i,k)}}{4|\nabla\rho(24)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k+1}^{(i,k+1)} - \delta_x T_{i,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \\
&\times \left(\frac{dxu_i\,\cos\phi_j^T dyt_j}{dhwt_{i,k,j}}\,\delta_x\rho_{i,k+1}^{(i,k+1)}\right) \qquad \text{(C.264)}
\end{aligned}
$$

C.2.10.3 y-z plane

Taking the derivative of the 12 contributions to the functional in the y-z plane yields

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(25)}}{\partial T_{i,k,j}} &= -\frac{A(25)V(25)}{|\nabla\rho(25)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k-1}^{(j,k)}\right) \\
&\times \left(-\frac{\delta_y\rho_{j-1,k}^{(j,k)}}{dhwt_{i,k-1,j}} - \frac{\delta_z\rho_{j,k-1}^{(j,k)}}{dyu_{j-1}}\right) \\
&= \frac{A(25)\,\Delta_{(k,j-1)}^{(k-1,j-1)}}{4|\nabla\rho(25)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k-1}^{(j,k)}\right)
\end{aligned}
$$

$$\times \quad dxt_i \, \cos\phi^U_{j-1} \left(\frac{dyu_{j-1}}{dhwt_{i,k-1,j}} \, \delta_y \rho^{(j,k)}_{j-1,k} + \delta_z \rho^{(j,k)}_{j,k-1} \right) \tag{C.265}$$

$$\begin{aligned}
\frac{\partial L^{(26)}_{j,k}}{\partial T_{i,k,j}} &= -\frac{A(26)V(26)}{|\nabla\rho(26)|^2} \left(\delta_z T_{j,k-1} \delta_y \rho^{(j,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j,k)}_{j,k-1} \right) \\
&\times \left(-\frac{\delta_y \rho^{(j,k)}_{j,k}}{dhwt_{i,k-1,j}} + \frac{\delta_z \rho^{(j,k)}_{j,k-1}}{dyu_j} \right) \\
&= -\frac{A(26)\, \Delta^{(k-1,j)}_{(k,j)}}{4|\nabla\rho(26)|^2} \left(\delta_z T_{j,k-1} \delta_y \rho^{(j,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j,k)}_{j,k-1} \right) \\
&\times \quad dxt_i \, \cos\phi^U_{j} \left(-\frac{dyu_j}{dhwt_{i,k-1,j}} \, \delta_y \rho^{(j,k)}_{j,k} + \delta_z \rho^{(j,k)}_{j,k-1} \right)
\end{aligned} \tag{C.266}$$

$$\begin{aligned}
\frac{\partial L^{(27)}_{j,k}}{\partial T_{i,k,j}} &= -\frac{A(27)V(27)}{|\nabla\rho(27)|^2} \left(\delta_z T_{j,k} \delta_y \rho^{(j,k)}_{j-1,k} - \delta_y T_{j-1,k} \delta_z \rho^{(j,k)}_{j,k} \right) \\
&\times \left(\frac{\delta_y \rho^{(j,k)}_{j-1,k}}{dhwt_{i,k,j}} - \frac{\delta_z \rho^{(j,k)}_{j,k}}{dyu_{j-1}} \right) \\
&= -\frac{A(27)\, \Delta^{(k,j-1)}_{(k,j-1)}}{4|\nabla\rho(27)|^2} \left(\delta_z T_{j,k} \delta_y \rho^{(j,k)}_{j-1,k} - \delta_y T_{j-1,k} \delta_z \rho^{(j,k)}_{j,k} \right) \\
&\times \quad dxt_i \, \cos\phi^U_{j-1} \left(\frac{dyu_{j-1}}{dhwt_{i,k,j}} \, \delta_y \rho^{(j,k)}_{j-1,k} - \delta_z \rho^{(j,k)}_{j,k} \right)
\end{aligned} \tag{C.267}$$

$$\begin{aligned}
\frac{\partial L^{(28)}_{j,k}}{\partial T_{i,k,j}} &= -\frac{A(28)V(28)}{|\nabla\rho(28)|^2} \left(\delta_z T_{j,k} \delta_y \rho^{(j,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j,k)}_{j,k} \right) \\
&\times \left(\frac{\delta_y \rho^{(j,k)}_{j,k}}{dhwt_{i,k,j}} + \frac{\delta_z \rho^{(j,k)}_{j,k}}{dyu_j} \right) \\
&= -\frac{A(28)\Delta^{(k,j)}_{(k,j)}}{4|\nabla\rho(28)|^2} \left(\delta_z T_{j,k} \delta_y \rho^{(j,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j,k)}_{j,k} \right) \\
&\times \quad dxt_i \, \cos\phi^U_{j} \left(\frac{dyu_j}{dhwt_{i,k,j}} \, \delta_y \rho^{(j,k)}_{j,k} + \delta_z \rho^{(j,k)}_{j,k} \right)
\end{aligned} \tag{C.268}$$

$$\begin{aligned}
\frac{\partial L^{(29)}_{j,k}}{\partial T_{i,k,j}} &= -\frac{A(29)V(29)}{|\nabla\rho(29)|^2} \left(\delta_z T_{j+1,k-1} \delta_y \rho^{(j+1,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j+1,k)}_{j+1,k-1} \right) \\
&\times \left(\frac{\delta_z \rho^{(j+1,k)}_{j+1,k-1}}{dyu_j} \right) \\
&= -\frac{A(29)\, \Delta^{(k-1,j)}_{(k,j)}}{4|\nabla\rho(29)|^2} \left(\delta_z T_{j+1,k-1} \delta_y \rho^{(j+1,k)}_{j,k} - \delta_y T_{j,k} \delta_z \rho^{(j+1,k)}_{j+1,k-1} \right) \\
&\times \left(dxt_i \, \cos\phi^U_{j} \, \delta_z \rho^{(j+1,k)}_{j+1,k-1} \right)
\end{aligned} \tag{C.269}$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(30)}}{\partial T_{i,k,j}} &= -\frac{A(30)V(30)}{|\nabla\rho(30)|^2}\left(\delta_z T_{j+1,k}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k}^{(j+1,k)}\right) \\
&\times \left(\frac{\delta_z\rho_{j+1,k}^{(j+1,k)}}{dyu_j}\right) \\
&= -\frac{A(30)\,\Delta_{(k,j)}^{(k,j)}}{4|\nabla\rho(30)|^2}\left(\delta_z T_{j+1,k}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k}^{(j+1,k)}\right) \\
&\times \left(dxt_i\ \cos\phi_j^U\ \delta_z\rho_{j+1,k}^{(j+1,k)}\right) \qquad \text{(C.270)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(31)}}{\partial T_{i,k,j}} &= -\frac{A(31)V(31)}{|\nabla\rho(31)|^2}\left(\delta_z T_{j-1,k-1}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k-1}^{(j-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{j-1,k-1}^{(j-1,k)}}{dyu_{j-1}}\right) \\
&= \frac{A(31)\,\Delta_{(k,j-1)}^{(k-1,j-1)}}{4|\nabla\rho(31)|^2}\left(\delta_z T_{j-1,k-1}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k-1}^{(j-1,k)}\right) \\
&\times \left(dxt_i\ \cos\phi_{j-1}^U\ \delta_z\rho_{j-1,k-1}^{(j-1,k)}\right) \qquad \text{(C.271)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(32)}}{\partial T_{i,k,j}} &= -\frac{A(32)V(32)}{|\nabla\rho(32)|^2}\left(\delta_z T_{j-1,k}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k}^{(j-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{j-1,k}^{(j-1,k)}}{dyu_{j-1}}\right) \\
&= \frac{A(32)\,\Delta_{(k,j-1)}^{(k,j-1)}}{4|\nabla\rho(32)|^2}\left(\delta_z T_{j-1,k}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k}^{(j-1,k)}\right) \\
&\times \left(dxt_i\ \cos\phi_{j-1}^U\ \delta_z\rho_{j-1,k}^{(j-1,k)}\right) \qquad \text{(C.272)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(33)}}{\partial T_{i,k,j}} &= \frac{A(33)V(33)}{|\nabla\rho(33)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k-1}^{(j,k-1)} - \delta_y T_{j-1,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right) \\
&\times \left(\frac{\delta_y\rho_{j-1,k-1}^{(j,k-1)}}{dhwt_{i,k-1,j}}\right) \\
&= \frac{A(33)\,\Delta_{(k-1,j-1)}^{(k-1,j-1)}}{4|\nabla\rho(33)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k-1}^{(j,k-1)} - \delta_y T_{j-1,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right) \\
&\times \left(\frac{dxt_i\ \cos\phi_{j-1}^U\ dyu_{j-1}}{dhwt_{i,k-1,j}}\ \delta_y\rho_{j-1,k-1}^{(j,k-1)}\right) \qquad \text{(C.273)}
\end{aligned}
$$

$$
\frac{\partial L_{j,k}^{(34)}}{\partial T_{i,k,j}} = -\frac{A(34)V(34)}{|\nabla\rho(34)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k-1}^{(j,k-1)} - \delta_y T_{j,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right)
$$

$$
\begin{aligned}
&\times \left(-\frac{\delta_y \rho_{j,k-1}^{(j,k-1)}}{dhwt_{i,k-1,j}} \right) \\
&= \frac{A(34)\ \Delta_{(k-1,j)}^{(k-1,j)}}{4|\nabla\rho(34)|^2} \left(\delta_z T_{j,k-1} \delta_y \rho_{j,k-1}^{(j,k-1)} - \delta_y T_{j,k-1} \delta_z \rho_{j,k-1}^{(j,k-1)} \right) \\
&\times \left(\frac{dxt_i\ \cos\phi_j^U dyu_j}{dhwt_{i,k-1,j}}\ \delta_y \rho_{j,k-1}^{(j,k-1)} \right) \qquad \text{(C.274)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(35)}}{\partial T_{i,k,j}} &= -\frac{A(35)V(35)}{|\nabla\rho(35)|^2} \left(\delta_z T_{j,k} \delta_y \rho_{j-1,k+1}^{(j,k+1)} - \delta_y T_{j-1,k+1} \delta_z \rho_{j,k}^{(j,k+1)} \right) \\
&\times \left(\frac{\delta_y \rho_{j-1,k+1}^{(j,k+1)}}{dzw_k} \right) \\
&= -\frac{A(35)\ \Delta_{(k+1,j-1)}^{(k,j-1)}}{4|\nabla\rho(35)|^2} \left(\delta_z T_{j,k} \delta_y \rho_{j-1,k+1}^{(j,k+1)} - \delta_y T_{j-1,k+1} \delta_z \rho_{j,k}^{(j,k+1)} \right) \\
&\times \left(\frac{dxt_i\ \cos\phi_{j-1}^U\ dyu_{j-1}}{dhwt_{i,k,j}}\ \delta_y \rho_{j-1,k+1}^{(j,k+1)} \right) \qquad \text{(C.275)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(36)}}{\partial T_{i,k,j}} &= -\frac{A(36)V(36)}{|\nabla\rho(36)|^2} \left(\delta_z T_{j,k} \delta_y \rho_{j,k+1}^{(j,k+1)} - \delta_y T_{j,k+1} \delta_z \rho_{j,k}^{(j,k+1)} \right) \\
&\times \left(\frac{\delta_y \rho_{j,k+1}^{(j,k+1)}}{dhwt_{i,k,j}} \right) \\
&= -\frac{A(36)\ \Delta_{(k+1,j)}^{(k,j)}}{4|\nabla\rho(36)|^2} \left(\delta_z T_{j,k} \delta_y \rho_{j,k+1}^{(j,k+1)} - \delta_y T_{j,k+1} \delta_z \rho_{j,k}^{(j,k+1)} \right) \\
&\times \left(\frac{dxt_i\ \cos\phi_j^U\ dyu_j}{dhwt_{i,k,j}}\ \delta_y \rho_{j,k+1}^{(j,k+1)} \right). \qquad \text{(C.276)}
\end{aligned}
$$

C.2.10.4 Recombination of terms in the x-y plane

At this stage, the functional derivatives have been computed and so the diffusion operator is available. In order to organize the results in a more compact form, it is useful to rearrange terms in order to identify Cartesian components to the isoneutral diffusive flux. The contributions to the diffusion operator from the x-y plane portion of the functional yield the x-y cross terms and one of two pieces each for the x-x and y-y diagonal terms. All these terms are of no concern for the numerical stability issues (see Griffies *et al.* 1998). Therefore, the diffusion coefficients in each sub-cell in the x-y plane will be set to their *a priori* value A_I^o. Finally, in this section all reference to depth levels will be omitted except for the vertical grid spacing.

The terms from quarter cells $1+7+9$ are

$$
\left(\frac{A_I^o}{4}\ dyu_j\ \frac{\cos\phi_j^U}{\cos\phi_j^T} \right) \times
$$

$$
\begin{aligned}
& \left(\frac{(dht_{i,k,j}\ \delta_y \rho_{i,j}^{(i,j)})(\delta_x \rho_{i-1,j}^{(i,j)} \delta_y T_{i,j} - \delta_x T_{i-1,j} \delta_y \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4}\ dyu_j\ \frac{\cos\phi_j^U}{\cos\phi_j^T} \right) \times \\
& \left(\frac{(dyt_{i-1,k,j}\ \delta_y \rho_{i-1,j}^{(i-1,j)})(\delta_x \rho_{i-1,j}^{(i-1,j)} \delta_y T_{i-1,j} - \delta_x T_{i-1,j} \delta_y \rho_{i-1,j}^{(i-1,j)})}{(\delta_x \rho_{i-1,j}^{(i-1,j)})^2 + (\delta_y \rho_{i-1,j}^{(i-1,j)})^2 + (\delta_z \rho_{i-1,j}^{(i-1,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4}\ dxu_{i-1}\ \cos\phi_j^U \right) \times \\
& \left(\frac{(dht_{i,k,j}\ \delta_x \rho_{i-1,j}^{(i,j)})(\delta_x \rho_{i-1,j}^{(i,j)} \delta_y T_{i,j} - \delta_x T_{i-1,j} \delta_y \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4}\ dxu_{i-1}\ \cos\phi_j^U \right) \times \\
& \left(\frac{(dht_{i,k,j+1}\ \delta_x \rho_{i-1,j+1}^{(i,j+1)})(\delta_x \rho_{i-1,j+1}^{(i,j+1)} \delta_y T_{i,j} - \delta_x T_{i-1,j+1} \delta_y \rho_{i,j}^{(i,j+1)})}{(\delta_x \rho_{i-1,j+1}^{(i,j+1)})^2 + (\delta_y \rho_{i,j}^{(i,j+1)})^2 + (\delta_z \rho_{i,j+1}^{(i,j+1)})^2} \right). \qquad \text{(C.277)}
\end{aligned}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
& \left(\frac{A_I^o}{4}\ dyu_j\ \frac{\cos\phi_j^U}{\cos\phi_j^T} \right) \times \\
& \sum_{ip=-1}^{0} \left(dht_{i+ip,k,j}\ \delta_y \rho_{i+ip,j}^{(i+ip,j)} \right) \left(\frac{\delta_x \rho_{i-1,j}^{(i+ip,j)} \delta_y T_{i+ip,j} - \delta_x T_{i-1,j} \delta_y \rho_{i+ip,j}^{(i+ip,j)}}{(\delta_x \rho_{i-1,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4}\ dxu_{i-1}\ \cos\phi_j^U \right) \times \\
& \sum_{jq=0}^{1} \left(dht_{i,k,j+jq}\ \delta_x \rho_{i-1,j+jq}^{(i,j+jq)} \right) \left(\frac{\delta_x \rho_{i-1,j+jq}^{(i,j+jq)} \delta_y T_{i,j} - \delta_x T_{i-1,j+jq} \delta_y \rho_{i,j}^{(i,j+jq)}}{(\delta_x \rho_{i-1,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2} \right). \qquad \text{(C.278)}
\end{aligned}
$$

The terms from quarter cells $2 + 5 + 10$ are

$$
\begin{aligned}
& \left(\frac{A_I^o}{4}\ dyu_j\ \frac{\cos\phi_j^U}{\cos\phi_j^T} \right) \times \\
& \left(\frac{(dht_{i,k,j}\ \delta_y \rho_{i,j}^{(i,j)})(\delta_y \rho_{i,j}^{(i,j)} \delta_x T_{i,j} - \delta_y T_{i,j} \delta_x \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4}\ dyu_j\ \frac{\cos\phi_j^U}{\cos\phi_j^T} \right) \times \\
& \left(\frac{(dht_{i+1,k,j}\ \delta_y \rho_{i+1,j}^{(i+1,j)})(\delta_y \rho_{i+1,j}^{(i+1,j)} \delta_x T_{i,j} - \delta_y T_{i+1,j} \delta_x \rho_{i,j}^{(i+1,j)})}{(\delta_x \rho_{i,j}^{(i+1,j)})^2 + (\delta_y \rho_{i+1,j}^{(i+1,j)})^2 + (\delta_z \rho_{i+1,j}^{(i+1,j)})^2} \right)
\end{aligned}
$$

$$
\begin{aligned}
&+ \left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_j^U\right) \times \\
&\left(\frac{(dht_{i,k,j}\, \delta_x\rho_{i,j}^{(i,j)})(\delta_x\rho_{i,j}^{(i,j)}\delta_y T_{i,j} - \delta_x T_{i,j}\delta_y\rho_{i,j}^{(i,j)})}{(\delta_x\rho_{i,j}^{(i,j)})^2 + (\delta_y\rho_{i,j}^{(i,j)})^2 + (\delta_z\rho_{i,j}^{(i,j)})^2}\right) \\
&+ \left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_j^U\right) \times \\
&\left(\frac{(dht_{i,k,j+1}\, \delta_x\rho_{i,j+1}^{(i,j+1)})(\delta_x\rho_{i,j+1}^{(i,j+1)}\delta_y T_{i,j} - \delta_x T_{i,j+1}\delta_y\rho_{i,j}^{(i,j+1)})}{(\delta_x\rho_{i,j+1}^{(i,j+1)})^2 + (\delta_y\rho_{i,j}^{(i,j+1)})^2 + (\delta_z\rho_{i,j+1}^{(i,j+1)})^2}\right).
\end{aligned}
\tag{C.279}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
&\left(\frac{A_I^o}{4}\, dyu_j\, \frac{\cos\phi_j^U}{\cos\phi_j^T}\right) \times \\
&\sum_{ip=0}^{1}\left(dht_{i+ip,k,j}\, \delta_y\rho_{i+ip,j}^{(i+ip,j)}\right)\left(\frac{\delta_y\rho_{i+ip,j}^{(i+ip,j)}\delta_x T_{i,j} - \delta_y T_{i+ip,j}\delta_x\rho_{i,j}^{(i+ip,j)}}{(\delta_x\rho_{i,j}^{(i+ip,j)})^2 + (\delta_y\rho_{i+ip,j}^{(i+ip,j)})^2 + (\delta_z\rho_{i+ip,j}^{(i+ip,j)})^2}\right) \\
+ &\left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_j^U\right) \times \\
&\sum_{jq=0}^{1}\left(dht_{i,k,j+jq}\, \delta_x\rho_{i,j+jq}^{(i,j+jq)}\right)\left(\frac{\delta_x\rho_{i,j+jq}^{(i,j+jq)}\delta_y T_{i,j} - \delta_x T_{i,j+jq}\delta_y\rho_{i,j}^{(i,j+jq)}}{(\delta_x\rho_{i,j+jq}^{(i,j+jq)})^2 + (\delta_y\rho_{i,j}^{(i,j+jq)})^2 + (\delta_z\rho_{i,j+jq}^{(i,j+jq)})^2}\right).
\end{aligned}
\tag{C.280}
$$

The terms from quarter cells $3 + 8 + 11$ are

$$
\begin{aligned}
&\left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
&\left(\frac{(dht_{i,k,j}\, \delta_y\rho_{i,j-1}^{(i,j)})(\delta_x\rho_{i-1,j}^{(i,j)}\delta_y T_{i,j-1} - \delta_x T_{i-1,j}\delta_y\rho_{i,j-1}^{(i,j)})}{(\delta_x\rho_{i-1,j}^{(i,j)})^2 + (\delta_y\rho_{i,j-1}^{(i,j)})^2 + (\delta_z\rho_{i,j}^{(i,j)})^2}\right) \\
+ &\left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
&\left(\frac{(dht_{i-1,k,j}\, \delta_y\rho_{i-1,j-1}^{(i-1,j)})(\delta_x\rho_{i-1,j}^{(i-1,j)}\delta_y T_{i-1,j-1} - \delta_x T_{i-1,j}\delta_y\rho_{i-1,j-1}^{(i-1,j)})}{(\delta_x\rho_{i-1,j}^{(i-1,j)})^2 + (\delta_y\rho_{i-1,j-1}^{(i-1,j)})^2 + (\delta_z\rho_{i-1,j}^{(i-1,j)})^2}\right) \\
+ &\left(\frac{A_I^o}{4}\, dxu_{i-1}\, \cos\phi_{j-1}^U\right) \times \\
&\left(\frac{(dht_{i,k,j}\, \delta_x\rho_{i-1,j}^{(i,j)})(\delta_y\rho_{i,j-1}^{(i,j)}\delta_x T_{i-1,j} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j}^{(i,j)})}{(\delta_x\rho_{i-1,j}^{(i,j)})^2 + (\delta_y\rho_{i,j-1}^{(i,j)})^2 + (\delta_z\rho_{i,j}^{(i,j)})^2}\right) \\
+ &\left(\frac{A_I^o}{4}\, dxu_{i-1}\, \cos\phi_{j-1}^U\right) \times \\
&\left(\frac{(dht_{i,k,j-1}\, \delta_x\rho_{i-1,j-1}^{(i,j-1)})(\delta_y\rho_{i,j-1}^{(i,j-1)}\delta_x T_{i-1,j-1} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j-1}^{(i,j-1)})}{(\delta_x\rho_{i-1,j-1}^{(i,j-1)})^2 + (\delta_y\rho_{i,j-1}^{(i,j-1)})^2 + (\delta_z\rho_{i,j-1}^{(i,j-1)})^2}\right).
\end{aligned}
\tag{C.281}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
& \left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
& \sum_{ip=-1}^{0} \left(dht_{i+ip,k,j}\ \delta_y \rho_{i+ip,j-1}^{(i+ip,j)}\right) \left(\frac{\delta_x \rho_{i-1,j}^{(i+ip,j)} \delta_y T_{i+ip,j-1} - \delta_x T_{i-1,j} \delta_y \rho_{i+ip,j-1}^{(i+ip,j)}}{(\delta_x \rho_{i-1,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j-1}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2}\right) \\
+ \quad & \left(\frac{A_I^o}{4}\, dxu_{i-1}\, \cos\phi_{j-1}^U\right) \times \\
& \sum_{jq=-1}^{0} \left(dht_{i,k,j+jq}\ \delta_x \rho_{i-1,j+jq}^{(i,j+jq)}\right) \left(\frac{\delta_y \rho_{i,j-1}^{(i,j+jq)} \delta_x T_{i-1,j+jq} - \delta_y T_{i,j-1} \delta_x \rho_{i-1,j+jq}^{(i,j+jq)}}{(\delta_x \rho_{i-1,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2}\right)
\end{aligned}
\tag{C.282}
$$

The terms from quarter cells $4 + 6 + 12$ are

$$
\begin{aligned}
& \left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
& \left(\frac{(dht_{i,k,j}\ \delta_y \rho_{i,j-1}^{(i,j)})(\delta_y \rho_{i,j-1}^{(i,j)} \delta_x T_{i,j} - \delta_y T_{i,j-1} \delta_x \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i,j}^{(i,j)})^2 + (\delta_y \rho_{i,j-1}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2}\right) \\
+ \quad & \left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
& \left(\frac{(dht_{i+1,k,j}\ \delta_y \rho_{i+1,j-1}^{(i+1,j)})(\delta_y \rho_{i+1,j-1}^{(i+1,j)} \delta_x T_{i,j} - \delta_y T_{i+1,j-1} \delta_x \rho_{i,j}^{(i+1,j)})}{(\delta_x \rho_{i,j}^{(i+1,j)})^2 + (\delta_y \rho_{i+1,j-1}^{(i+1,j)})^2 + (\delta_z \rho_{i+1,j}^{(i+1,j)})^2}\right) \\
+ \quad & \left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_{j-1}^U\right) \times \\
& \left(\frac{(dht_{i,k,j}\ \delta_x \rho_{i,j}^{(i,j)})(\delta_y \rho_{i,j-1}^{(i,j)} \delta_x T_{i,j} - \delta_y T_{i,j-1} \delta_x \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2}\right) \\
+ \quad & \left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_{j-1}^U\right) \times \\
& \left(\frac{(dht_{i,k,j-1}\ \delta_x \rho_{i,j-1}^{(i,j-1)})(\delta_y \rho_{i,j-1}^{(i,j-1)} \delta_x T_{i,j-1} - \delta_y T_{i,j-1} \delta_x \rho_{i,j-1}^{(i,j-1)})}{(\delta_x \rho_{i,j-1}^{(i,j-1)})^2 + (\delta_y \rho_{i,j-1}^{(i,j-1)})^2 + (\delta_z \rho_{i,j-1}^{(i,j-1)})^2}\right).
\end{aligned}
\tag{C.283}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
& \left(\frac{A_I^o}{4}\, dyu_{j-1}\, \frac{\cos\phi_{j-1}^U}{\cos\phi_j^T}\right) \times \\
& \sum_{ip=0}^{1} \left(dht_{i+ip,k,j}\ \delta_y \rho_{i+ip,j-1}^{(i+ip,j)}\right) \left(\frac{\delta_y \rho_{i+ip,j-1}^{(i+ip,j)} \delta_x T_{i,j} - \delta_y T_{i+ip,j-1} \delta_x \rho_{i,j}^{(i+ip,j)}}{(\delta_x \rho_{i,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j-1}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2}\right) \\
+ \quad & \left(\frac{A_I^o}{4}\, dxu_i\, \cos\phi_{j-1}^U\right) \times
\end{aligned}
$$

$$\sum_{jq=-1}^{0}\left(dht_{i,k,j+jq}\ \delta_x\rho_{i,j+jq}^{(i,j+jq)}\right)\left(\frac{\delta_y\rho_{i,j-1}^{(i,j+jq)}\delta_x T_{i,j+jq}-\delta_y T_{i,j-1}\delta_x\rho_{i,j+jq}^{(i,j+jq)}}{(\delta_x\rho_{i,j+jq}^{(i,j+jq)})^2+(\delta_y\rho_{i,j-1}^{(i,j+jq)})^2+(\delta_z\rho_{i,j+jq}^{(i,j+jq)})^2}\right). \quad \text{(C.284)}$$

There are now a total of eight terms. It is possible to combine the pairs appearing with the same summation into a total of four terms, each with a double summation. These four terms are

$$\begin{aligned}
&\left(\frac{A_I^o}{4\cos\phi_j^T}\right)\sum_{jq=-1}^{0} dyu_{j+jq}\ \cos\phi_{j+jq}^U \sum_{ip=-1}^{0}\left(dht_{i+ip,k,j}\ \delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}\right)\times\\
&\frac{\delta_x\rho_{i-1,j}^{(i+ip,j)}\delta_y T_{i+ip,j+jq}-\delta_x T_{i-1,j}\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}}{(\delta_x\rho_{i-1,j}^{(i+ip,j)})^2+(\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)})^2+(\delta_z\rho_{i+ip,j}^{(i+ip,j)})^2}\\
+&\left(\frac{A_I^o}{4\cos\phi_j^T}\right)\sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1}\left(dht_{i+ip,k,j}\ \delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}\right)\times\\
&\frac{\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}\delta_x T_{i,j}-\delta_y T_{i+ip,j+jq}\delta_x\rho_{i,j}^{(i+ip,j)}}{(\delta_x\rho_{i,j}^{(i+ip,j)})^2+(\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)})^2+(\delta_z\rho_{i+ip,j}^{(i+ip,j)})^2}\\
+&\left(\frac{A_I^o\cos\phi_{j-1}^U}{4}\right)\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{jq=-1}^{0}\left(dht_{i,k,j+jq}\ \delta_x\rho_{i+ip,j+jq}^{(i,j+jq)}\right)\times\\
&\frac{\delta_y\rho_{i,j-1}^{(i,j+jq)}\delta_x T_{i+ip,j+jq}-\delta_y T_{i,j-1}\delta_x\rho_{i+ip,j+jq}^{(i,j+jq)}}{(\delta_x\rho_{i+ip,j+jq}^{(i,j+jq)})^2+(\delta_y\rho_{i,j-1}^{(i,j+jq)})^2+(\delta_z\rho_{i,j+jq}^{(i,j+jq)})^2}\\
+&\left(\frac{A_I^o\ \cos\phi_j^U}{4}\right)\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j+jq}\ \delta_x\rho_{i+ip,j+jq}^{(i,j+jq)}\right)\times\\
&\frac{\delta_x\rho_{i+ip,j+jq}^{(i,j+jq)}\delta_y T_{i,j}-\delta_x T_{i+ip,j+jq}\delta_y\rho_{i,j}^{(i,j+jq)}}{(\delta_x\rho_{i+ip,j+jq}^{(i,j+jq)})^2+(\delta_y\rho_{i,j}^{(i,j+jq)})^2+(\delta_z\rho_{i,j+jq}^{(i,j+jq)})^2}.
\end{aligned} \quad \text{(C.285)}$$

In order to identify a difference operator, in the first term of equation (C.285), let the label ip run from 0 to 1 and adjust the i label accordingly. With this shift, the first and second terms in equation (C.285) take the form

$$\begin{aligned}
-&\left(\frac{A_I^o}{4\cos\phi_j^T}\right)\sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1}\left(dht_{i-1+ip,k,j}\ \delta_y\rho_{i-1+ip,j+jq}^{(i-1+ip,j)}\right)\times\\
&\frac{\delta_x T_{i-1,j}\delta_y\rho_{i-1+ip,j+jq}^{(i-1+ip,j)}-\delta_x\rho_{i-1,j}^{(i-1+ip,j)}\delta_y T_{i-1+ip,j+jq}}{(\delta_x\rho_{i-1,j}^{(i-1+ip,j)})^2+(\delta_y\rho_{i-1+ip,j+jq}^{(i-1+ip,j)})^2+(\delta_z\rho_{i-1+ip,j}^{(i-1+ip,j)})^2}\\
+&\left(\frac{A_I^o}{4\cos\phi_j^T}\right)\sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1}\left(dht_{i+ip,k,j}\ \delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}\right)\times\\
&\frac{\delta_x T_{i,j}\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)}-\delta_x\rho_{i,j}^{(i+ip,j)}\delta_y T_{i+ip,j+jq}}{(\delta_x\rho_{i,j}^{(i+ip,j)})^2+(\delta_y\rho_{i+ip,j+jq}^{(i+ip,j)})^2+(\delta_z\rho_{i+ip,j}^{(i+ip,j)})^2}.
\end{aligned} \quad \text{(C.286)}$$

In order to center the elements of the first term in equation (C.286) to the west side of the T-cell i, k, j, take an average over z of the z-derivative squared in the denominator. This prescription

will also bring the second term to the east side, which allows for a zonal difference operator across the T-cell to be identified. This averaging in the denominator does not disturb any of the numerical or physical properties of the scheme and it allows for an unambiguous placement of these terms without introducing computational modes. With this prescription, these two terms become

$$\left(\frac{A_I^o\, dxt_i}{4}\right) \quad \delta_x\left(\sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi^U_{j+jq}\sum_{ip=0}^{1}\left(dht_{i-1+ip,k,j}\,\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j+jq}\right)\times \frac{\delta_x T_{i-1,j}\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j+jq}-\delta_x\rho^{(i-1+ip,j)}_{i-1,j}\delta_y T_{i-1+ip,j+jq}}{(\delta_x\rho^{(i-1+ip,j)}_{i-1,j})^2+(\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j+jq})^2+.5\sum_{kr=0,1}(\delta_z\rho^{(i-1+ip,k,j)}_{i-1+ip,k-1+kr,j})^2}\right).$$

It is convenient to shift the jq sum and to introduce the volume of the T-cell $V_{T_{i,k,j}} = dxt_i\,\cos\phi^T_j\,dyt_j\,dht_{i,k,j}$, which yields

$$\left(\frac{A_I^o\, V_{T_{i,k,j}}}{4dht_{i,k.j}\,dyt_j\,\cos\phi^T_j}\right) \quad \delta_x\left(\sum_{jq=0}^{1} dyu_{j-1+jq}\cos\phi^U_{j-1+jq}\sum_{ip=0}^{1}\left(dht_{i-1+ip,k,j}\,\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j-1+jq}\right)\times \frac{\delta_x T_{i-1,j}\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j-1+jq}-\delta_x\rho^{(i-1+ip,j)}_{i-1,j}\delta_y T_{i-1+ip,j-1+jq}}{(\delta_x\rho^{(i-1+ip,j)}_{i-1,j})^2+(\delta_y\rho^{(i-1+ip,j)}_{i-1+ip,j-1+jq})^2+.5\sum_{kr=0,1}(\delta_z\rho^{(i-1+ip,k,j)}_{i-1+ip,k-1+kr,j})^2}\right) \tag{C.287}$$

Now let jq run from 0 to 1 in the third term of Equation (C.285) and shift j accordingly, to bring the third and fourth terms of Equation (C.285) to

$$\begin{aligned} &- \left(\frac{A_I^o\,\cos\phi^U_{j-1}}{4}\right)\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j-1+jq}\,\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq}\right)\times \\ &\quad \frac{\delta_y T_{i,j-1}\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq}-\delta_y\rho^{(i,j-1+jq)}_{i,j-1}\delta_x T_{i+ip,j-1+jq}}{(\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq})^2+(\delta_y\rho^{(i,j-1+jq)}_{i,j-1})^2+(\delta_z\rho^{(i,j-1+jq)}_{i,j-1+jq})^2} \\ &+ \left(\frac{A_I^o\,\cos\phi^U_{j}}{4}\right)\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j+jq}\,\delta_x\rho^{(i,j+jq)}_{i+ip,j+jq}\right)\times \\ &\quad \frac{\delta_x\rho^{(i,j+jq)}_{i+ip,j+jq}\delta_y T_{i,j}-\delta_x T_{i+ip,j+jq}\delta_y\rho^{(i,j+jq)}_{i,j}}{(\delta_x\rho^{(i,j+jq)}_{i+ip,j+jq})^2+(\delta_y\rho^{(i,j+jq)}_{i,j})^2+(\delta_z\rho^{(i,j+jq)}_{i,j+jq})^2}, \end{aligned}$$

which can be combined into the meridional difference

$$\left(\frac{A_I^o\, dyt_j}{4}\right) \quad \delta_y\left(\cos\phi^U_{j-1}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j-1+jq}\,\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq}\right)\times \frac{\delta_y T_{i,j-1}\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq}-\delta_y\rho^{(i,j-1+jq)}_{i,j-1}\delta_x T_{i+ip,j-1+jq}}{(\delta_x\rho^{(i,j-1+jq)}_{i+ip,j-1+jq})^2+(\delta_y\rho^{(i,j-1+jq)}_{i,j-1})^2+.5\sum_{kr=0,1}(\delta_z\rho^{(i,k,j-1+jq)}_{i,k-1+kr,j-1+jq})^2}\right),$$

where the z-derivative in the denominator was averaged in order to bring it to the appropriate meridional face of the T-cell. Shifting the ip sum and introducing the volume factor $V_{T_{i,k,j}}$

yields

$$\left(\frac{A_I^o\ V_{T_{i,k,j}}}{4dxt_i\ \cos\phi_j^T\ dht_{i,k,j}}\right) \quad \delta_y\left(\cos\phi_{j-1}^U \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{jq=0}^{1}\left(dht_{i,k,j-1+jq}\ \delta_x\rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)}\right)\times \right.$$
$$\left. \frac{\delta_y T_{i,j-1}\delta_x\rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)} - \delta_y\rho_{i,j-1}^{(i,j-1+jq)}\delta_x T_{i-1+ip,j-1+jq}}{(\delta_x\rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)})^2 + (\delta_y\rho_{i,j-1}^{(i,j-1+jq)})^2 + .5\sum_{kr=0,1}(\delta_z\rho_{i,k-1+kr,j-1+jq}^{(i,k,j-1+jq)})^2}\right) \tag{C.288}$$

C.2.10.5 Recombination of terms in the x-z plane

For the x-z plane, it is important to be explicit about the particular value of the diffusion coefficient to be used in order to ensure that the numerical stability criteria discussed in Griffies *et al.* (1998) is satisfied. Explicit reference to the latitude will be omitted except for dyt_j. The manipulations follow quite analogously to those just performed for the x-y terms.

The three terms $13 + 19 + 21$ combine to form

$$\begin{aligned}
&\left(\frac{\Delta_{(i-1,k)}^{(i-1,k-1)}}{dhwt_{i,k-1}}\right)\left(\frac{dxu_{i-1}\ \cos\phi_j^T\ dyt_j}{4}\right) A(13)\ \delta_x\rho_{i-1,k}^{(i,k)}\times \\
&\left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}}{(\delta_x\rho_{i-1,k}^{(i,k)})^2 + (\delta_y\rho_{i,k}^{(i,k)})^2 + (\delta_z\rho_{i,k-1}^{(i,k)})^2}\right) \\
+ &\left(\frac{\Delta_{(i-1,k-1)}^{(i-1,k-1)}}{dhwt_{i,k-1}}\right)\left(\frac{dxu_{i-1}\ \cos\phi_j^T\ dyt_j}{4}\right) A(21)\ \delta_x\rho_{i-1,k-1}^{(i,k-1)}\times \\
&\left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}}{(\delta_x\rho_{i-1,k-1}^{(i,k-1)})^2 + (\delta_y\rho_{i,k-1}^{(i,k-1)})^2 + (\delta_z\rho_{i,k-1}^{(i,k-1)})^2}\right) \\
+ &\left(\frac{\Delta_{(i-1,k)}^{(i-1,k-1)}\ dyt_j}{4}\right) A(13)\ \delta_z\rho_{i,k-1}^{(i,k)}\times \\
&\left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}}{(\delta_x\rho_{i-1,k}^{(i,k)})^2 + (\delta_y\rho_{i,k}^{(i,k)})^2 + (\delta_z\rho_{i,k-1}^{(i,k)})^2}\right) \\
+ &\left(\frac{\Delta_{(i-1,k)}^{(i-1,k-1)}\ dyt_j}{4}\right) A(19)\ \delta_z\rho_{i-1,k-1}^{(i-1,k)}\times \\
&\left(\frac{\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}}{(\delta_x\rho_{i-1,k}^{(i-1,k)})^2 + (\delta_y\rho_{i-1,k}^{(i-1,k)})^2 + (\delta_z\rho_{i-1,k-1}^{(i-1,k)})^2}\right).
\end{aligned} \tag{C.289}$$

The diffusion coefficient $A(13)$ is chosen to make the x-projection of the neutral direction slope

$$Sx_{(i-1,k|i,k-1)}^{(i,k)} \equiv -\frac{\delta_x\rho_{i-1,k}^{(i,k)}}{\delta_z\rho_{i,k-1}^{(i,k)}} \tag{C.290}$$

satisfy

$$S_{(-)} \le |Sx_{(i-1,k|i,k-1)}^{(i,k)}| \le S_{(-)}^{-1} \tag{C.291}$$

if the grid parameter

$$\delta = min(\frac{\Delta x_i \Delta z_k}{4 A_I \Delta t}) \tag{C.292}$$

is $< 1/2$, where one and only one of the grid spacing is in the vertical. Otherwise, no rescaling of the diffusion coefficient is necessary to maintain numerical stability. To make explicit this association, introduce the notation

$$A(13) \equiv Ax^{(i,k)}_{(i-1,k|i,k-1)}. \tag{C.293}$$

Likewise, let

$$A(19) \equiv Ax^{(i-1,k)}_{(i-1,k|i-1,k-1)}, \tag{C.294}$$

$$A(21) \equiv Ax^{(i,k-1)}_{(i-1,k-1|i,k-1)} \tag{C.295}$$

denote the other diffusion coefficients whose values are set according the value of their respective slopes. Introducing summation notation, the three terms $13 + 19 + 21$ can now be written

$$\begin{aligned}
&\frac{dxu_{i-1}\ \cos\phi_j^T\ dyt_j}{4\ dhwt_{i,k-1}} \sum_{kr=-1}^{0} \Delta^{(i-1,k-1)}_{(i-1,k+kr)}\ Ax^{(i,k+kr)}_{(i-1,k+kr|i,k-1)}\ \delta_x \rho^{(i,k+kr)}_{i-1,k+kr} \times \\
&\left(\frac{\delta_z T_{i,k-1} \delta_x \rho^{(i,k+kr)}_{i-1,k+kr} - \delta_x T_{i-1,k+kr} \delta_z \rho^{(i,k+kr)}_{i,k-1}}{(\delta_x \rho^{(i,k+kr)}_{i-1,k+kr})^2 + (\delta_y \rho^{(i,k+kr)}_{i,k+kr})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k-1})^2} \right) \\
+ \quad &\frac{dyt_j\ \Delta^{(i-1,k-1)}_{(i-1,k)}}{4} \sum_{ip=-1}^{0} Ax^{(i+ip,k)}_{(i-1,k|i+ip,k-1)}\ \delta_z \rho^{(i+ip,k)}_{i+ip,k-1} \times \\
&\left(\frac{\delta_z T_{i+ip,k-1} \delta_x \rho^{(i+ip,k)}_{i-1,k} - \delta_x T_{i-1,k} \delta_z \rho^{(i+ip,k)}_{i+ip,k-1}}{(\delta_x \rho^{(i+ip,k)}_{i-1,k})^2 + (\delta_y \rho^{(i+ip,k)}_{i+ip,k})^2 + (\delta_z \rho^{(i+ip,k)}_{i+ip,k-1})^2} \right).
\end{aligned} \tag{C.296}$$

Similar considerations lead to the quarter cells $14 + 17 + 22$ becoming

$$\begin{aligned}
&\frac{dxu_i\ \cos\phi_j^T\ dyt_j}{4\ dhwt_{i,k-1}} \sum_{kr=-1}^{0} \Delta^{(i,k-1)}_{(i,k+kr)}\ Ax^{(i,k+kr)}_{(i,k+kr|i,k-1)}\ \delta_x \rho^{(i,k+kr)}_{i,k+kr} \times \\
&\left(\frac{\delta_z T_{i,k-1} \delta_x \rho^{(i,k+kr)}_{i,k+kr} - \delta_x T_{i,k+kr} \delta_z \rho^{(i,k+kr)}_{i,k-1}}{(\delta_x \rho^{(i,k+kr)}_{i,k+kr})^2 + (\delta_y \rho^{(i,k+kr)}_{i,k+kr})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k-1})^2} \right) \\
+ \quad &\frac{dyt_j\ \Delta^{(i,k-1)}_{(i,k)}}{4} \sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1)} \delta_z \rho^{(i+ip,k)}_{i+ip,k-1} \times \\
&\left(\frac{\delta_x T_{i,k} \delta_z \rho^{(i+ip,k)}_{i+ip,k-1} - \delta_z T_{i+ip,k-1} \delta_x \rho^{(i+ip,k)}_{i,k}}{(\delta_x \rho^{(i+ip,k)}_{i,k})^2 + (\delta_y \rho^{(i+ip,k)}_{i+ip,k})^2 + (\delta_z \rho^{(i+ip,k)}_{i+ip,k-1})^2} \right),
\end{aligned} \tag{C.297}$$

the quarter cells $15 + 20 + 23$ becoming

$$\frac{dxu_{i-1}\ \cos\phi_j^T\ dyt_j}{4 dhwt_{i,k}} \sum_{kr=0}^{1} \Delta^{(i-1,k)}_{(i-1,k+kr)}\ Ax^{(i,k+kr)}_{(i-1,k+kr|i,k)} \delta_x \rho^{(i,k+kr)}_{i-1,k+kr} \times$$

$$
\left(\frac{\delta_x T_{i-1,k+kr}\delta_z\rho_{i,k}^{(i,k+kr)} - \delta_z T_{i,k}\delta_x\rho_{i-1,k+kr}^{(i,k+kr)}}{(\delta_x\rho_{i-1,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k}^{(i,k+kr)})^2}\right)
$$

$$
+ \quad \frac{dyt_j\,\Delta_{(i-1,k)}^{(i-1,k)}}{4}\sum_{ip=-1}^{0} Ax_{(i-1,k|i+ip,k)}^{(i+ip,k)}\delta_z\rho_{i+ip,k}^{(i+ip,k)} \times
$$

$$
\left(\frac{\delta_z T_{i+ip,k}\delta_x\rho_{i-1,k}^{(i+ip,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i+ip,k}^{(i+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k}^{(i+ip,k)})^2}\right), \tag{C.298}
$$

and the quarter cells $16 + 18 + 24$ becoming

$$
\frac{dxu_i\,\cos\phi_j^T\,dyt_j}{4\,dhwt_{i,k}}\sum_{kr=0}^{1}\Delta_{(i,k+kr)}^{(i,k)}\,Ax_{(i,k+kr|i,k)}^{(i,k+kr)}\delta_x\rho_{i,k+kr}^{(i,k+kr)} \times
$$

$$
\left(\frac{\delta_x T_{i,k+kr}\delta_z\rho_{i,k}^{(i,k+kr)} - \delta_z T_{i,k}\delta_x\rho_{i,k+kr}^{(i,k+kr)}}{(\delta_x\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k}^{(i,k+kr)})^2}\right)
$$

$$
+ \quad \frac{dyt_j\,\Delta_{(i,k)}^{(i,k)}}{4}\sum_{ip=0}^{1} Ax_{(i,k|i+ip,k)}^{(i+ip,k)}\delta_z\rho_{i+ip,k}^{(i+ip,k)} \times
$$

$$
\left(\frac{\delta_x T_{i,k}\delta_z\rho_{i+ip,k}^{(i+ip,k)} - \delta_z T_{i+ip,k}\delta_x\rho_{i,k}^{(i+ip,k)}}{(\delta_x\rho_{i,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k}^{(i+ip,k)})^2}\right). \tag{C.299}
$$

There are now a total of eight terms, which can be combined into four terms, each with two summations. This combination renders

$$
\frac{\cos\phi_j^T dyt_j}{4\,dhwt_{i,k-1}}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=-1}^{0} \quad \Delta_{(i+ip,k+kr)}^{(i+ip,k-1)}\,Ax_{(i+ip,k+kr|i,k-1)}^{(i,k+kr)}\,\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} \times
$$

$$
\left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} - \delta_x T_{i+ip,k+kr}\delta_z\rho_{i,k-1}^{(i,k+kr)}}{(\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k-1}^{(i,k+kr)})^2}\right)
$$

$$
+\frac{\cos\phi_j^T\,dyt_j}{4\,dhwt_{i,k}}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=0}^{1} \quad \Delta_{(i+ip,k+kr)}^{(i+ip,k)}\,Ax_{(i+ip,k+kr|i,k)}^{(i,k+kr)}\,\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} \times
$$

$$
\left(\frac{\delta_x T_{i+ip,k+kr}\delta_z\rho_{i,k}^{(i,k+kr)} - \delta_z T_{i,k}\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)}}{(\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k}^{(i,k+kr)})^2}\right)
$$

$$
+\frac{dyt_j}{4}\sum_{kr=-1}^{0}\Delta_{(i,k)}^{(i,k+kr)}\sum_{ip=0}^{1} \quad Ax_{(i,k|i+ip,k+kr)}^{(i+ip,k)}\,\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} \times
$$

$$
\left(\frac{\delta_x T_{i,k}\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} - \delta_z T_{i+ip,k+kr}\delta_x\rho_{i,k}^{(i+ip,k)}}{(\delta_x\rho_{i,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)})^2}\right)
$$

$$
+\frac{dyt_j}{4}\sum_{kr=-1}^{0}\Delta_{(i-1,k)}^{(i-1,k+kr)}\sum_{ip=-1}^{0} \quad Ax_{(i-1,k|i+ip,k+kr)}^{(i+ip,k)}\,\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} \times
$$

$$\left(\frac{\delta_z T_{i+ip,k+kr}\delta_x \rho^{(i+ip,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z \rho^{(i+ip,k)}_{i+ip,k+kr}}{(\delta_x \rho^{(i+ip,k)}_{i-1,k})^2 + (\delta_y \rho^{(i+ip,k)}_{i+ip,k})^2 + (\delta_z \rho^{(i+ip,k)}_{i+ip,k+kr})^2}\right) \quad \text{(C.300)}$$

In the first term of Equation (C.300), let the kr sum run from 0 to 1 and adjust the k labels appropriately to get the first and second terms into the form

$$\frac{\cos\phi_j^T\, dyt_j}{4\, dhwt_{i,k-1}} \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{kr=0}^{1} \Delta^{(i+ip,k-1)}_{(i+ip,k-1+kr)}\, Ax^{(i,k-1+kr)}_{(i+ip,k-1+kr|i,k-1)}\, \delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr} \times$$
$$\left(\frac{\delta_z T_{i,k-1}\delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr} - \delta_x T_{i+ip,k-1+kr}\delta_z \rho^{(i,k-1+kr)}_{i,k-1}}{(\delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr})^2 + (\delta_y \rho^{(i,k-1+kr)}_{i,k-1+kr})^2 + (\delta_z \rho^{(i,k-1+kr)}_{i,k-1})^2}\right)$$
$$-\frac{\cos\phi_j^T dyt_j}{4\, dhwt_{i,k}} \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{kr=0}^{1} \Delta^{(i+ip,k)}_{(i+ip,k+kr)}\, Ax^{(i,k+kr)}_{(i+ip,k+kr|i,k)}\, \delta_x \rho^{(i,k+kr)}_{i+ip,k+kr} \times$$
$$\left(\frac{\delta_z T_{i,k}\delta_x \rho^{(i,k+kr)}_{i+ip,k+kr} - \delta_x T_{i+ip,k+kr}\delta_z \rho^{(i,k+kr)}_{i,k}}{(\delta_x \rho^{(i,k+kr)}_{i+ip,k+kr})^2 + (\delta_y \rho^{(i,k+kr)}_{i,k+kr})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k})^2}\right). \quad \text{(C.301)}$$

With an average on the y-derivative in the denominator in order to center all terms onto the bottom face of T-cells, these two terms can be identified as the vertical difference across the faces of T-cell (i, k, j):

$$\frac{\cos\phi_j^T\, dyt_j\, dht_{i,k}}{4}\, \delta_z \left(\sum_{ip=-1}^{0} dxu_{i+ip} \sum_{kr=0}^{1} \frac{\Delta^{(i+ip,k-1)}_{(i+ip,k-1+kr)}}{dhwt_{i,k-1}}\, Ax^{(i,k-1+kr)}_{(i+ip,k-1+kr|i,k-1)}\, \delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr} \right.$$
$$\left. \frac{\delta_z T_{i,k-1}\delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr} - \delta_x T_{i+ip,k-1+kr}\delta_z \rho^{(i,k-1+kr)}_{i,k-1}}{(\delta_x \rho^{(i,k-1+kr)}_{i+ip,k-1+kr})^2 + .5\sum_{jq=0,1}(\delta_y \rho^{(i,k-1+kr,j)}_{i,k-1+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k-1+kr)}_{i,k-1})^2} \right).$$

Introducing the volume factor $V_{T_{i,k,j}} = dxt_i\, \cos\phi_j^T\, dyt_j\, dht_{i,k,j}$ and shifting the ip sum, this difference becomes

$$\frac{V_{T_{i,k,j}}}{4dxt_i}\, \delta_z \left(\sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \frac{\Delta^{(i-1+ip,k-1)}_{(i-1+ip,k-1+kr)}}{dhwt_{i,k-1}}\, Ax^{(i,k-1+kr)}_{(i-1+ip,k-1+kr|i,k-1)}\, \delta_x \rho^{(i,k-1+kr)}_{i-1+ip,k-1+kr} \right.$$
$$\left. \frac{\delta_z T_{i,k-1}\delta_x \rho^{(i,k-1+kr)}_{i-1+ip,k-1+kr} - \delta_x T_{i-1+ip,k-1+kr}\delta_z \rho^{(i,k-1+kr)}_{i,k-1}}{(\delta_x \rho^{(i,k-1+kr)}_{i-1+ip,k-1+kr})^2 + .5\sum_{jq=0,1}(\delta_y \rho^{(i,k-1+kr,j)}_{i,k-1+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k-1+kr)}_{i,k-1})^2} \right). \quad \text{(C.302)}$$

In the fourth term of Equation (C.300), shift the ip sum to 0,1 and adjust the i label accordingly to have the third and fourth terms take the form

$$\frac{dyt_j}{4} \sum_{kr=-1}^{0} \Delta^{(i,k+kr)}_{(i,k)} \sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k+kr)}\, \delta_z \rho^{(i+ip,k)}_{i+ip,k+kr} \times$$
$$\left(\frac{\delta_x T_{i,k}\delta_z \rho^{(i+ip,k)}_{i+ip,k+kr} - \delta_z T_{i+ip,k+kr}\delta_x \rho^{(i+ip,k)}_{i,k}}{(\delta_x \rho^{(i+ip,k)}_{i,k})^2 + (\delta_y \rho^{(i+ip,k)}_{i+ip,k})^2 + (\delta_z \rho^{(i+ip,k)}_{i+ip,k+kr})^2}\right)$$

$$-\frac{dyt_j}{4}\sum_{kr=-1}^{0}\Delta_{(i-1,k)}^{(i-1,k+kr)}\sum_{ip=0}^{1} \quad Ax_{(i-1,k|i-1+ip,k+kr)}^{(i-1+ip,k)}\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}\times$$
$$\left(\frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}-\delta_z T_{i-1+ip,k+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2+(\delta_y\rho_{i-1+ip,k}^{(i-1+ip,k)})^2+(\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)})^2}\right),$$

which becomes the zonal difference across the T-cell $T_{i,k,j}$ upon averaging the y-difference in the denominator

$$\frac{dxt_i\cos\phi_j^T dyt_j}{4}\,\delta_x\left(\sum_{kr=-1}^{0}\Delta_{(i-1,k)}^{(i-1,k+kr)}\sum_{ip=0}^{1}Ax_{(i-1,k|i-1+ip,k+kr)}^{(i-1+ip,k)}\,\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}\right.$$
$$\left.\frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}-\delta_z T_{i-1+ip,k+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2+.5\sum_{jq=0,1}(\delta_y\rho_{i-1+ip,k,j-1+jq}^{(i-1+ip,k,j)})^2+(\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)})^2}\right). \tag{C.303}$$

Introducing the volume factor $V_{T_{i,k,j}} = dxt_i\ \cos\phi_j^T\ dyt_j\ dht_{i,k,j}$ and shifting the kr sum, this difference becomes

$$\frac{V_{T_{i,k,j}}}{4dht_{i,k}}\,\delta_x\left(\sum_{kr=0}^{1}\Delta_{(i-1,k)}^{(i-1,k-1+kr)}\sum_{ip=0}^{1}Ax_{(i-1,k|i-1+ip,k-1+kr)}^{(i-1+ip,k)}\,\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)}\right.$$
$$\left.\frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)}-\delta_z T_{i-1+ip,k-1+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2+.5\sum_{jq=0,1}(\delta_y\rho_{i-1+ip,k,j-1+jq}^{(i-1+ip,k,j)})^2+(\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)})^2}\right). \tag{C.304}$$

C.2.10.6 Recombination of terms in the y-z plane

The y-z plane is done similarly to the x-z plane. Its solution can be read from the x-z solution with the appropriate index and $\cos\phi$ changes

$$\frac{V_{T_{i,k,j}}}{4\cos\phi_j^T dyt_j}\,\delta_z\left(\sum_{jq=0}^{1}\cos\phi_{j-1+jq}^{U}dyu_{j-1+jq}\sum_{kr=0}^{1}\frac{\Delta_{(k-1+kr,j-1+jq)}^{(k-1,j-1+jq)}}{dhwt_{k-1,j}}\,Ay_{(k-1+kr,j-1+jq|k-1,j)}^{(k-1+kr,j)}\right.$$
$$\left.\delta_y\rho_{k-1+kr,j-1+jq}^{(k-1+kr,j)}\,\frac{\delta_z T_{k-1,j}\delta_y\rho_{k-1+kr,j-1+jq}^{(k-1+kr,j)}-\delta_y T_{k-1+kr,j-1+jq}\delta_z\rho_{k-1,j}^{(k-1+kr,j)}}{.5\sum_{ip=0,1}(\delta_x\rho_{i-1+ip,k-1+kr,j}^{(i,k-1+kr,j)})^2+(\delta_y\rho_{k-1+kr,j-1+jq}^{(k-1+kr,j)})^2+(\delta_z\rho_{k-1,j}^{(k-1+kr,j)})^2}\right)$$
$$+\quad\frac{V_{T_{i,k,j}}}{4dht_{k,j}\ \cos\phi_j^T}\,\delta_y\left(\cos\phi_{j-1}^{U}\sum_{kr=0}^{1}\Delta_{(k,j-1)}^{(k-1+kr,j-1)}\sum_{jq=0}^{1}Ay_{(k,j-1|k-1+kr,j-1+jq)}^{(k,j-1+jq)}\right.$$
$$\left.\delta_z\rho_{k-1+kr,j-1+jq}^{(k,j-1+jq)}\frac{\delta_y T_{k,j-1}\delta_z\rho_{k-1+kr,j-1+jq}^{(k,j-1+jq)}-\delta_z T_{k-1+kr,j-1+jq}\delta_y\rho_{k,j-1}^{(k,j-1+jq)}}{.5\sum_{ip=0,1}(\delta_x\rho_{i-1+ip,k,j-1+jq}^{(k,j-1+jq)})^2+(\delta_y\rho_{k,j-1}^{(k,j-1+jq)})^2+(\delta_z\rho_{k-1+kr,j-1+jq}^{(k,j-1+jq)})^2}\right) \tag{C.305}$$

C.3 Isoneutral diffusive flux

The above details provide the explicit form for the discretization of the diffusion operator $R(T) = -\nabla\cdot\vec{F}(T)$. All that is needed is to divide out the volume factor $V_{T_{i,k,j}}$ according to Equation (C.15). As an additional step, it is useful to identify the three components to the diffusive flux since MOM is coded in terms of flux components across cell faces. Identifying the

diffusive flux components also allows for an easier implementation of the no-flux boundary conditions than working directly with the diffusion operator. This section provides the explicit form for the discretized components to the isoneutral diffusive flux.

C.3.1 Zonal component to the isoneutral diffusive flux

The zonal component $F^x_{i,k,j}$, defined at the east face of T-cell (i,k,j), is given by

$$\begin{aligned}
-F^x_{i,k,j} &= \left(\frac{1}{4\,dyt_j\,\cos\phi^T_j}\right)\sum_{jq=0}^{1} dyu_{j-1+jq}\,\cos\phi^U_{j-1+jq}\sum_{ip=0}^{1}\left(dht_{i+ip,k,j}\,A^o_I\,\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq}\right)\times \\
&\quad \frac{\delta_x T_{i,j}\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq} - \delta_x\rho^{(i+ip,j)}_{i,j}\delta_y T_{i+ip,j-1+jq}}{(\delta_x\rho^{(i+ip,j)}_{i,j})^2 + (\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \\
&+ \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1}\left(Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr}\right)\times \\
&\quad \frac{\delta_x T_{i,k}\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr} - \delta_z T_{i+ip,k-1+kr}\delta_x\rho^{(i+ip,k)}_{i,k}}{(\delta_x\rho^{(i+ip,k)}_{i,k})^2 + .5\sum_{jq=0,1}(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + (\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr})^2}.
\end{aligned} \tag{C.306}$$

The isoneutral flux simplifies greatly when taking the small slope approximation. In this case, the first term of the zonal flux component is neglected, and only the vertical density gradient in the denominator of the second term is kept. This approximation leads to the zonal flux component

$$\begin{aligned}
-F^{x\,small}_{i,k,j} &= \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1}Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\times \\
&\quad \left(\delta_x T_{i,k} + Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\delta_z T_{i+ip,k-1+kr}\right).
\end{aligned} \tag{C.307}$$

Note that the diffusion coefficient for the small angle approximation flux is subject to a different slope constraint than the full slope (see Griffies *et al.* 1998). Also note that for full cells, $\Delta^{(i,k-1+kr,j)}_{(i,k,j)} = dzw_{k-1+kr}$, which allows for a recovery of the full cell discretization from the partial cell algorithm.

It is important to note that this zonal diffusive flux component is defined with dimensions (tracer $\times\, L^2/t$), rather than (tracer $\times\, L/t$). The reason is that when taking the divergence of this flux in order to determine the contribution to the diffusion operator, the flux is normalized by the vertical size of the T-cell:

$$R^x(T)_{i,k,j} = -\left(\frac{1}{dht_{i,k,j}}\right)\delta_x F^x_{i-1,k,j}. \tag{C.308}$$

This normalization appeared naturally in the derivation of the diffusion operator. For the case of full vertical cells, this factor can be incorporated into the definition of the zonal flux since the factor will then be a function only of the vertical grid point. In this case, for example, the zonal component to the flux takes the form

$$\begin{aligned}
-F^{x\,small,full\,cell}_{i,k}(T) &= \frac{1}{4dzt_k}\sum_{kr=0}^{1}dzw_{k-1+kr}\sum_{ip=0}^{1}A^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\times \\
&\quad \left(\delta_x T_{i,k} + Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\,\delta_z T_{i+ip,k-1+kr}\right),
\end{aligned} \tag{C.309}$$

and the contribution to the diffusion operator is given by $R^x(T)_{i,k,j} = -\delta_x F^{x\ full\ cell}_{i-1,k,j}$.

The off-diagonal terms for the diffusion tensor are not separable from the tracers. The diagonal terms, however, are separable. For purposes of computational speed, it is useful to explicitly identify the non-negative diagonal elements since they can be used for all tracers and so need be computed only once. For the full Redi diffusion tensor, one has the zonal diagonal component

$$\begin{aligned} K^{11}_{i,k,j} &= \left(\frac{1}{4\,dyt_j\,\cos\phi^T_j}\right)\sum_{jq=0}^{1} dyu_{j-1+jq}\,\cos\phi^U_{j-1+jq}\sum_{ip=0}^{1} dht_{i+ip,k,j}\,A^o_I\times \\ &\quad \frac{(\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq})^2}{(\delta_x\rho^{(i+ip,j)}_{i,j})^2+(\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq})^2+.5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \\ &+ \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\times \\ &\quad \frac{(\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr})^2}{(\delta_x\rho^{(i+ip,k)}_{i,k})^2+.5\sum_{jq=0,1}(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2+(\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr})^2}. \end{aligned} \tag{C.310}$$

Using this definition brings the zonal component to the full tensor flux to the form

$$\begin{aligned} -F^x_{i,k,j} &= K^{11}_{i,k,j}\,\delta_x T_{i,j} \\ &- \left(\frac{1}{4\,dyt_j\,\cos\phi^T_j}\right)\sum_{jq=0}^{1} dyu_{j-1+jq}\,\cos\phi^U_{j-1+jq}\sum_{ip=0}^{1}\left(dht_{i+ip,k,j}\,A^o_I\,\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq}\right)\times \\ &\quad \frac{\delta_x\rho^{(i+ip,j)}_{i,j}\delta_y T_{i+ip,j-1+jq}}{(\delta_x\rho^{(i+ip,j)}_{i,j})^2+(\delta_y\rho^{(i+ip,j)}_{i+ip,j-1+jq})^2+.5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \\ &- \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1}\left(Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr}\right)\times \\ &\quad \frac{\delta_z T_{i+ip,k-1+kr}\delta_x\rho^{(i+ip,k)}_{i,k}}{(\delta_x\rho^{(i+ip,k)}_{i,k})^2+.5\sum_{jq=0,1}(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2+(\delta_z\rho^{(i+ip,k)}_{i+ip,k-1+kr})^2}. \end{aligned} \tag{C.311}$$

For the small angle approximation, one has the diagonal Redi tensor component

$$K^{11\ small}_{i,k,j} = \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}, \tag{C.312}$$

which brings the zonal flux to the form

$$\begin{aligned} -F^{x\ small}_{i,k,j} &= K^{11\ small}_{i,k,j}\,\delta_x T_{i,k} \\ &+ \frac{1}{4}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{ip=0}^{1} Ax^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\,Sx^{(i+ip,k)}_{(i,k|i+ip,k-1+kr)}\delta_z T_{i+ip,k-1+kr}. \end{aligned} \tag{C.313}$$

These formulae can be interpreted as a weighted average of four sub-fluxes, each weighted by certain grid spacing factors. The grid weighting factors take the form

$$\left(\frac{dyu_{j-1+jq}\,\cos\phi^U_{j-1+jq}}{4\,dyt_j\,\cos\phi^T_j}\right)\left(dht_{i+ip,k,j}\right) \tag{C.314}$$

for the x-y term, and

$$\left(\frac{1}{4}\right)\left(\Delta_{(i,k,j)}^{(i,k-1+kr,j)}\right) \tag{C.315}$$

for the x-z term. These weighting factors embody information about the grid size and the *minimum vertical spacing rule* discussed in Section C.2.3. An additional global factor $(dht_{i,k,j})^{-1}$ is applied to the flux diveregence.

C.3.2 Meridional component to the isoneutral diffusive flux

The meridional component $F^y_{i,k,j}$, defined at the north face of T-cell (i,k,j), is given by

$$\begin{aligned}
-F^y_{i,k,j} &= \frac{\cos\phi^U_j}{4dxt_i\ \cos\phi^T_j}\sum_{ip=0}^{1} dxu_{i-1+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j+jq}\ A^o_I\ \delta_x\rho^{(i,j+jq)}_{i-1+ip,j+jq}\right)\times \\
&\quad \frac{\delta_y T_{i,j}\delta_x\rho^{(i,j+jq)}_{i-1+ip,j+jq} - \delta_y\rho^{(i,j+jq)}_{i,j}\delta_x T_{i-1+ip,j+jq}}{(\delta_x\rho^{(i,j+jq)}_{i-1+ip,j+jq})^2 + (\delta_y\rho^{(i,j+jq)}_{i,j})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq})^2} \\
&+ \frac{\cos\phi^U_j}{4\cos\phi^T_j}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{jq=0}^{1}\left(Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\ \delta_z\rho^{(k,j+jq)}_{k-1+kr,j+jq}\right)\times \\
&\quad \frac{\delta_y T_{k,j}\delta_z\rho^{(k,j+jq)}_{k-1+kr,j+jq} - \delta_z T_{k-1+kr,j+jq}\delta_y\rho^{(k,j+jq)}_{k,j}}{.5\sum_{ip=0,1}(\delta_x\rho^{(k,j+jq)}_{i-1+ip,k,j+jq})^2 + (\delta_y\rho^{(k,j+jq)}_{k,j})^2 + (\delta_z\rho^{(k,j+jq)}_{k-1+kr,j+jq})^2}.
\end{aligned} \tag{C.316}$$

The small slope approximation leads to the meridional flux component

$$\begin{aligned}
-F^{y\ small}_{i,k,j} &= \frac{\cos\phi^U_j}{4\cos\phi^T_j}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{jq=0}^{1}Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\times \\
&\quad \left(\delta_y T_{k,j} + Sy^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\ \delta_z T_{k-1+kr,j+jq}\right).
\end{aligned} \tag{C.317}$$

As for the zonal flux, the contribution of this flux component to the diffusion operator is given by the normalized divergence

$$R^y(T)_{i,k,j} = -\left(\frac{1}{dht_{i,k,j}}\right)\delta_y F^y_{i,k,j-1}. \tag{C.318}$$

The diagonal meridional component to the full Redi tensor is given by

$$\begin{aligned}
K^{22}_{i,k,j} &= \frac{1}{4dxt_i\ \cos\phi^T_j}\sum_{ip=0}^{1} dxu_{i-1+ip}\sum_{jq=0}^{1}\left(dht_{i,k,j+jq}\ A^o_I\right)\times \\
&\quad \frac{(\delta_x\rho^{(i,j+jq)}_{i-1+ip,j+jq})^2}{(\delta_x\rho^{(i,j+jq)}_{i-1+ip,j+jq})^2 + (\delta_y\rho^{(i,j+jq)}_{i,j})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq})^2} \\
&+ \frac{1}{4\cos\phi^T_j}\sum_{kr=0}^{1}\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\sum_{jq=0}^{1}\left(Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\right)\times \\
&\quad \frac{(\delta_z\rho^{(k,j+jq)}_{k-1+kr,j+jq})^2}{.5\sum_{ip=0,1}(\delta_x\rho^{(k,j+jq)}_{i-1+ip,k,j+jq})^2 + (\delta_y\rho^{(k,j+jq)}_{k,j})^2 + (\delta_z\rho^{(k,j+jq)}_{k-1+kr,j+jq})^2},
\end{aligned} \tag{C.319}$$

which brings the full slope meridional flux component to the form

$$
\begin{aligned}
-F^{y}_{i,k,j} &= \cos\phi^{U}_{j}\; K^{22}_{i,k,j}\; \delta_y T_{i,j} \\
&- \frac{\cos\phi^{U}_{j}}{4dxt_i\;\cos\phi^{T}_{j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{jq=0}^{1} \left(dht_{i,k,j+jq}\; A^{o}_{I}\; \delta_x \rho^{(i,j+jq)}_{i-1+ip,j+jq} \right) \times \\
&\quad \frac{\delta_y \rho^{(i,j+jq)}_{i,j} \delta_x T_{i-1+ip,j+jq}}{(\delta_x \rho^{(i,j+jq)}_{i-1+ip,j+jq})^2 + (\delta_y \rho^{(i,j+jq)}_{i,j})^2 + .5 \sum_{kr=0,1} (\delta_z \rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq})^2} \\
&- \frac{\cos\phi^{U}_{j}}{4\cos\phi^{T}_{j}} \sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{jq=0}^{1} \left(Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\; \delta_z \rho^{(k,j+jq)}_{k-1+kr,j+jq} \right) \times \\
&\quad \frac{\delta_z T_{k-1+kr,j+jq} \delta_y \rho^{(k,j+jq)}_{k,j}}{.5 \sum_{ip=0,1} (\delta_x \rho^{(k,j+jq)}_{i-1+ip,k,j+jq})^2 + (\delta_y \rho^{(k,j+jq)}_{k,j})^2 + (\delta_z \rho^{(k,j+jq)}_{k-1+kr,j+jq})^2}.
\end{aligned}
\tag{C.320}
$$

The small slope approximation leads to

$$
K^{22\,small}_{i,k,j} = \frac{1}{4\cos\phi^{T}_{j}} \sum_{kr=0}^{1} \sum_{jq=0}^{1} \left(\Delta^{(i,k-1+kr,j)}_{(i,k,j)}\; Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)} \right), \tag{C.321}
$$

and the meridional component to the flux

$$
\begin{aligned}
-F^{y\,small}_{i,k,j} &= \cos\phi^{U}_{j}\; K^{22\,small}_{i,k,j}\; \delta_y T_{k,j} \\
&+ \frac{\cos\phi^{U}_{j}}{4\cos\phi^{T}_{j}} \sum_{kr=0}^{1} \Delta^{(i,k-1+kr,j)}_{(i,k,j)} \sum_{jq=0}^{1} Ay^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)} \times \\
&\quad \left(Sy^{(k,j+jq)}_{(k,j|k-1+kr,j+jq)}\; \delta_z T_{k-1+kr,j+jq} \right).
\end{aligned}
\tag{C.322}
$$

These formulae can be interpreted as a weighted average of four sub-fluxes, each weighted by certain grid spacing factors. The grid weighting factors take the form

$$
\left(\frac{dxu_{i-1+ip}\;\cos\phi^{U}_{j}}{4dxt_i\;\cos\phi^{T}_{j}} \right) \left(dht_{i,k,j+jq} \right) \tag{C.323}
$$

for the y-x term, and

$$
\left(\frac{\cos\phi^{U}_{j}}{4\cos\phi^{T}_{j}} \right) \left(\Delta^{(i,k-1+kr,j)}_{(i,k,j)} \right) \tag{C.324}
$$

for the y-z term. These weighting factors embody information about the grid size and the *minimum vertical spacing rule* discussed in Section C.2.3. An additional global factor $(dht_{i,k,j})^{-1}$ is applied to the flux diveregence.

C.3.3 Vertical component to the isoneutral diffusive flux

The vertical component $F^{z}_{i,k,j}$, defined at the bottom face of T-cell (i, k, j), is given by

$$
-F^{z}_{i,k,j} = \frac{1}{4dxt_i\;dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}\; Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} (\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr})
$$

$$
\begin{aligned}
& \frac{\delta_z T_{i,k} \delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr} - \delta_x T_{i-1+ip,k+kr} \delta_z \rho^{(i,k+kr)}_{i,k}}{(\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr})^2 + .5 \sum_{jq=0,1} (\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k})^2} \\
+ \quad & \frac{1}{4 \cos \phi^T_j dyt_j \, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos \phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)} \, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
& \delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq} \frac{\delta_z T_{k,j} \delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq} - \delta_y T_{k+kr,j-1+jq} \delta_z \rho^{(k+kr,j)}_{k,j}}{.5 \sum_{ip=0,1} (\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + (\delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq})^2 + (\delta_z \rho^{(k+kr,j)}_{k,j})^2} . \qquad \text{(C.325)}
\end{aligned}
$$

The small slope approximation leads to the vertical flux

$$
\begin{aligned}
-F^z_{i,k,j} \quad = \quad & \frac{1}{4 dxt_i \, dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)} \, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \\
& Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \left(\delta_x T_{i-1+ip,k+kr} + Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \, \delta_z T_{i,k} \right) \\
+ \quad & \frac{1}{4 \cos \phi^T_j dyt_j \, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos \phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)} \, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
& Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \left(\delta_y T_{k+kr,j-1+jq} + Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \, \delta_z T_{k,j} \right) . \qquad \text{(C.326)}
\end{aligned}
$$

The diagonal vertical component to the full Redi tensor is given by

$$
\begin{aligned}
K^{33}_{i,k,j} \quad = \quad & \frac{1}{4 dxt_i \, dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)} \, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \\
& \frac{(\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr})^2}{(\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr})^2 + .5 \sum_{jq=0,1} (\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k})^2} \\
+ \quad & \frac{1}{4 \cos \phi^T_j dyt_j \, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos \phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)} \, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
& \frac{(\delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq})^2}{.5 \sum_{ip=0,1} (\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + (\delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq})^2 + (\delta_z \rho^{(k+kr,j)}_{k,j})^2} , \qquad \text{(C.327)}
\end{aligned}
$$

which leads to the full slope vertical flux component

$$
\begin{aligned}
-F^z_{i,k,j} \quad = \quad & K^{33}_{i,k,j} \, \delta_z T_{i,k} \\
- \quad & \frac{1}{4 dxt_i \, dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)} \, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} (\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr}) \\
& \frac{\delta_x T_{i-1+ip,k+kr} \delta_z \rho^{(i,k+kr)}_{i,k}}{(\delta_x \rho^{(i,k+kr)}_{i-1+ip,k+kr})^2 + .5 \sum_{jq=0,1} (\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr)}_{i,k})^2} \\
- \quad & \frac{1}{4 \cos \phi^T_j dyt_j \, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos \phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)} \, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
& \delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq} \frac{\delta_y T_{k+kr,j-1+jq} \delta_z \rho^{(k+kr,j)}_{k,j}}{.5 \sum_{ip=0,1} (\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + (\delta_y \rho^{(k+kr,j)}_{k+kr,j-1+jq})^2 + (\delta_z \rho^{(k+kr,j)}_{k,j})^2} . \qquad \text{(C.328)}
\end{aligned}
$$

The small angle approximation leads to

$$
\begin{aligned}
K^{33\,small}_{i,k,j} &= \frac{1}{4dxt_i\, dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \times \\
&\quad \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}\, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}\, (Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)})^2 \\
&+ \frac{1}{4\cos\phi^T_j dyt_j\, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \times \\
&\quad \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)}\, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}\, (Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)})^2, \qquad \text{(C.329)}
\end{aligned}
$$

and the small angle vertical flux component

$$
\begin{aligned}
-F^{z\,small}_{i,k,j} &= K^{33}_{i,k,j}\, \delta_z T_{i,k} \\
&+ \frac{1}{4dxt_i\, dhwt_{i,k,j}} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} \Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}\, Ax^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)} \\
&\quad \times Sx^{(i,k+kr)}_{(i-1+ip,k+kr|i,k)}\, \delta_x T_{i-1+ip,k+kr} \\
&+ \frac{1}{4\cos\phi^T_j dyt_j\, dhwt_{i,k,j}} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} \Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)}\, Ay^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)} \\
&\quad \times Sy^{(k+kr,j)}_{(k+kr,j-1+jq|k,j)}\, \delta_y T_{k+kr,j-1+jq}. \qquad \text{(C.330)}
\end{aligned}
$$

These formulae can be interpreted as a weighted average of four sub-fluxes, each weighted by certain grid spacing factors. The grid weighting factors take the form

$$
\left(\frac{dxu_{i-1+ip}}{4dxt_i}\right)\left(\frac{\Delta^{(i-1+ip,k,j)}_{(i-1+ip,k+kr,j)}}{dhwt_{i,k,j}}\right) \qquad \text{(C.331)}
$$

for the x-z term, and

$$
\left(\frac{\cos\phi^U_{j-1+jq}\, dyu_{j-1+jq}}{4\cos\phi^T_j dyt_j}\right)\left(\frac{\Delta^{(i,k,j-1+jq)}_{(i,k+kr,j-1+jq)}}{dhwt_{i,k,j}}\right) \qquad \text{(C.332)}
$$

for the y-z term. These weighting factors embody information about the grid size and the *minimum vertical spacing rule* discussed in Section C.2.3. Note the presence of four Δ factors, as opposed to the two factors appearing in the zonal and meridional flux components. The reason there are four here is related to the rotated sense of the triads building up the vertical component to the flux, which implies each of the four triads is connected to an independent pair of quarter cells (see Section C.3.4). For the case of full vertical cells, the Δ factors cancel the *dhwt* factors, leaving only horizontal grid weights.

C.3.4 Stencils for small angle flux components

A figure is useful to garner insight into the stencil prescribed by the discretization. Figure C.7 provides such a stencil for the zonal component to the small angle isoneutral flux $F^{x\,small}_{i,k}$. Each

triad is weighted by the smallest vertical distance consistent with the *minimum vertical spacing rule* discussed in Section C.2.3. A similar stencil holds for the meridional component to the isoneutral diffusive flux.

For the vertical component (Figure C.7) to the diffusive flux, a set of four triads are used, each of which is rotated by 90 degrees relative to the triads shown for the zonal component to the flux. The weighting for each triad is a combination of the zonal grid spacing and a ratio of the *minimum vertical spacing rule* spacing, normalized by the relevant vertical T-cell distance. As a result of the rotation, the Δ weighting for each of the four triads is generally different.

C.4 General comments

This section presents some general comments and details regarding the implementation of the new isoneutral diffusion scheme.

C.4.1 Isoneutral diffusion operator

The isoneutral diffusion operator is given by the divergence of the diffusive fluxes

$$R[T]_{i,k,j} = -\left(\frac{1}{dht_{i,k,j}} \, \delta_x F^x_{i-1,k,j} + \frac{1}{dht_{i,k,j} \, \cos\phi^T_j} \, \delta_y F^y_{i,k,j-1} + \delta_z F^z_{i,k-1,j} \right) \tag{C.333}$$

The fluxes admit no computational modes in the density since only nearest neighbor differences are employed. This operator is defined at the center of the T-grid cell $T_{i,k,j}$. Exposing the spherical coordinates, gives

$$\begin{aligned} R[T]_{i,k,j} = &= DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + DIFF_Tz_{i,k,j} \\ &= \frac{\delta_\lambda(diff_fe_{i-1,k,j})}{dht_{i,k,j} \, \cos\phi^T_{jrow}} + \frac{\delta_\phi(diff_fn_{i,k,j-1})}{dht_{i,k,j} \, \cos\phi^T_{jrow}} + \delta_z(diff_fb_{i,k-1,j}), \end{aligned} \tag{C.334}$$

where the diffusive flux vector in the appendix is related to that defined in the model through

$$\vec{F}_{i,k,j} = -(diff_fe_{i,k,j} \; diff_fn_{i,k,j} \; diff_fb_{i,k,j}). \tag{C.335}$$

The flux vector has components defined at the center of the east, north, and bottom of cell $T_{i,k,j}$respectively. The diffusive flux vector satisfies the appropriate flux conditions at the domain boundaries. At the walls, there is no normal flux; at the bottom, there is the option of specifying a bottom flux (for studying, say, geothermal processes; typically assumed zero), and at the top, surface tracer flux information is fed into the vertical flux component. In general, these flux conditions are enforced in the model using the mask array $tmask_{i,k,j}$.

The shifting of the labels on the respective diffusive flux components is necessary in order to bring the difference of the fluxes onto the center of the cell $T_{i,k,j}$. For example, the difference

$$\delta_\lambda(diff_fe_{i-1,k,j}) = \frac{diff_fe_{i,k,j} - diff_fe_{i-1,k,j}}{dxt_i} \tag{C.336}$$

is defined at the center of $T_{i,k,j}$, whereas $\delta_\lambda(diff_fe_{i,k,j})$ is defined at the center of $T_{i+1,k,jrow}$. The denominator dxt_i represents the grid distance between the east and west faces of $T_{i,k,j}$. The

fluxes in the meridional and vertical directions follow similarly, which yields

$$\begin{aligned} R[T]_{i,k,j} &= \frac{diff_fe_{i,k,j} - diff_fe_{i-1,k,j}}{\cos\phi^T_{jrow} \cdot dxt_i \cdot dht_{i,k,j}} + \frac{diff_fn_{i,k,j} - diff_fn_{i,k,j-1}}{\cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dht_{i,k,j}} \\ &+ \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dhz_k}. \end{aligned} \tag{C.337}$$

C.4.2 Vertical diffusion equation

The ability to identify the 3,3 component of the Redi tensor is very useful since it enables the vertical diffusion equation to be solved implicitly in the same manner as with the old scheme.

C.4.3 Dianeutral piece

C.4.3.1 Full tensor

The diffusion coefficients $A(n)$ in the previous discussion of the full tensor corresponded to isoneutral diffusivities, minus any explicit dianeutral diffusivity (e.g., see Equations (C.9)-(C.11)). For modest slopes, these two coefficients are very different, with the isoneutral diffusivity roughly $10^7 - 10^8$ larger than the dianuetral diffusivity. When the full tensor needs rescaling for the intermediate slopes in which it is not stable, the rescaled isoneutral diffusivity is still roughly $10^3 - 10^4$ larger than the dianeutral diffusivity. Therefore, the diffusion coefficient is essentially the isoneutral diffusivity. The discretization of dianeutral diffusion therefore reduces to the discretization of $\partial_m(A_D\partial_m T)$, with the vertical piece done implicitly along with the $K^{3,3}$ piece of the diffusion tensor.

C.4.3.2 Small tensor

For the small tensor, the dianeutral piece is simply $\partial_z(A_D\partial_z T)$, which should be done implicitly along with the $K^{3,3}$ piece of the isoneutral diffusion tensor.

C.4.4 Highlighting the different average operations

There are numerous differences between the new scheme and that implemented by Cox (1987). One difference is in the form of the averaging operations. In particular, for the small slope fluxes, taking a uniform grid in the respective directions and neglecting the specification of the reference points allows for the double sums in the new scheme to collapse to familiar averaging operators. For example, consider the x-z fluxes in the small angle limit. The old scheme used the discretization

$$-F^x_{i,k} = A_I\left(\delta_x T_{i,k} - \frac{\delta_z \overline{T}^{x,z}_{i,k-1}}{\delta_z \overline{\rho}^{x,z}_{i,k-1}}\,\delta_x \rho_{i,k}\right) \tag{C.338}$$

$$-F^z_{i,k} = -\frac{A_I}{\delta_z \rho_{i,k}}\left(\delta_x \overline{T}^{x,z}_{i-1,k}\,\delta_x \overline{\rho}^{x,z}_{i-1,k}\right) + A_I \frac{\delta_z T_{i,k}}{(\delta_z \rho_{i,k})^2}(\delta_x \overline{\rho}^{x,z}_{i-1,k})^2, \tag{C.339}$$

whereas the new fluxes (with uniform grid, neglecting the reference points, and assuming constant diffusion coefficients) are given by

$$-F^x_{i,k,j} = A_I\left(\delta_x T_{i,k,j} - \overline{\left(\frac{\delta_z T_{i,k-1,j}}{\delta_z \rho_{i,k-1,j}}\right)}^{x,z}\,\delta_x \rho_{i,k,j}\right) \tag{C.340}$$

$$-F^z_{i,k,j} = -\frac{A_I}{\delta_z \rho_{i,k,j}} \left(\overline{\delta_x T_{i-1,k,j}\, \delta_x \rho_{i-1,k,j}}^{x,z} \right) + A_I \frac{\delta_z T_{i,k,j}}{(\delta_z \rho_{i,k,j})^2}\, \overline{(\delta_x \rho_{i-1,k,j})^2}^{x,z}. \quad \text{(C.341)}$$

The difference between the fluxes is related to how the averages are applied to the z-derivative terms for the x-flux, and how the averages are applied to the x-derivative terms in the z-flux. Namely, the new scheme applies averages over a product or ratio of fields rather than individually as done in the original scheme. The new procedure provides for a scheme that has no computational modes in the density. Indeed, this discretization might be an inspired guess by one motivated by the desire to eliminate computational modes. It is unclear how one could be inspired to guess the details of the grid weights and the reference points. It is therefore very satisfying that the functional approach provides an objective means to traverse the sea of details inherent in the discretization.

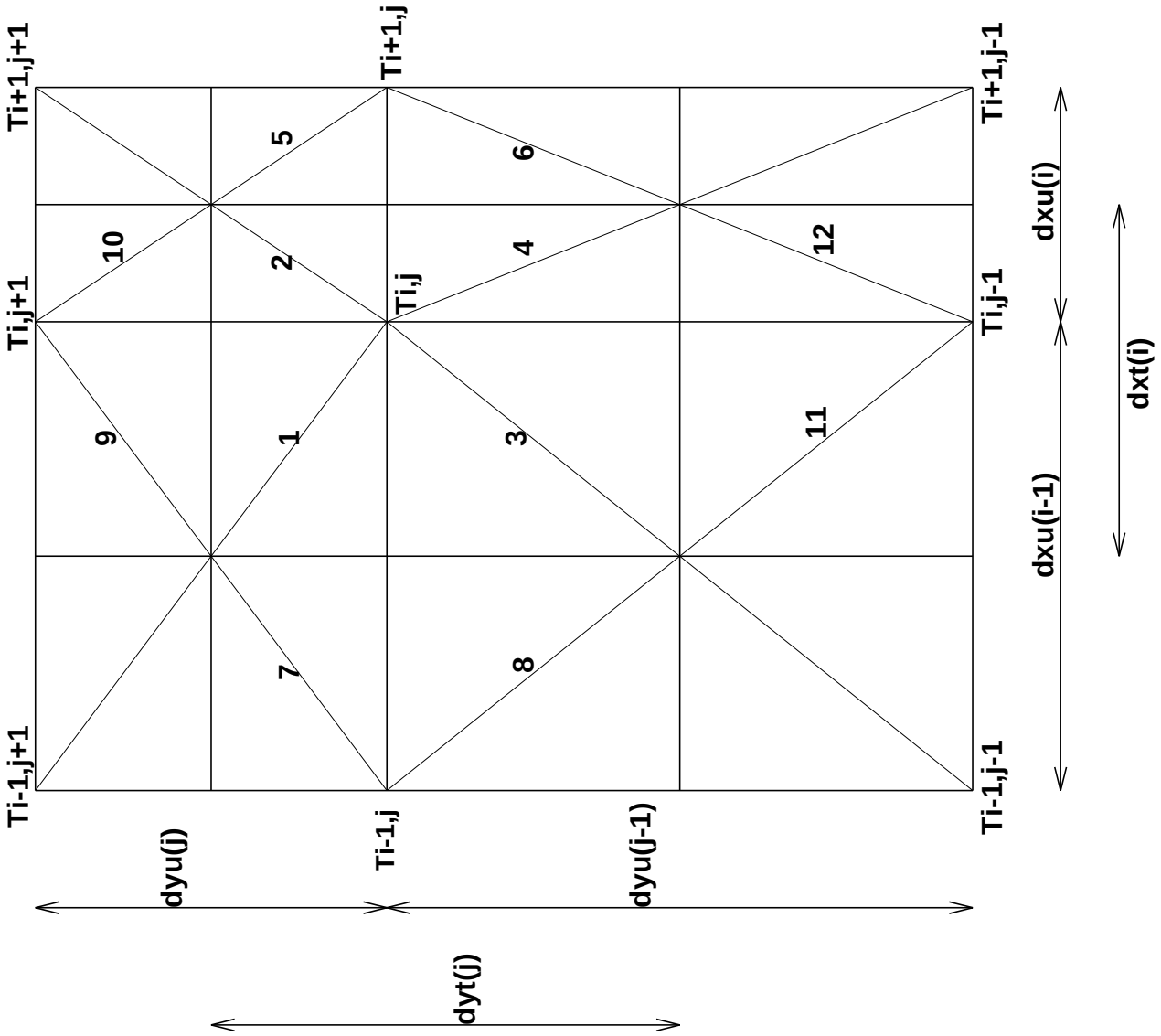


Figure C.2: MOM x-y plane and its partitioning into 12 quarter cells. The generally nonconstant grid spacing is indicated, which implies a non-centered T-point.

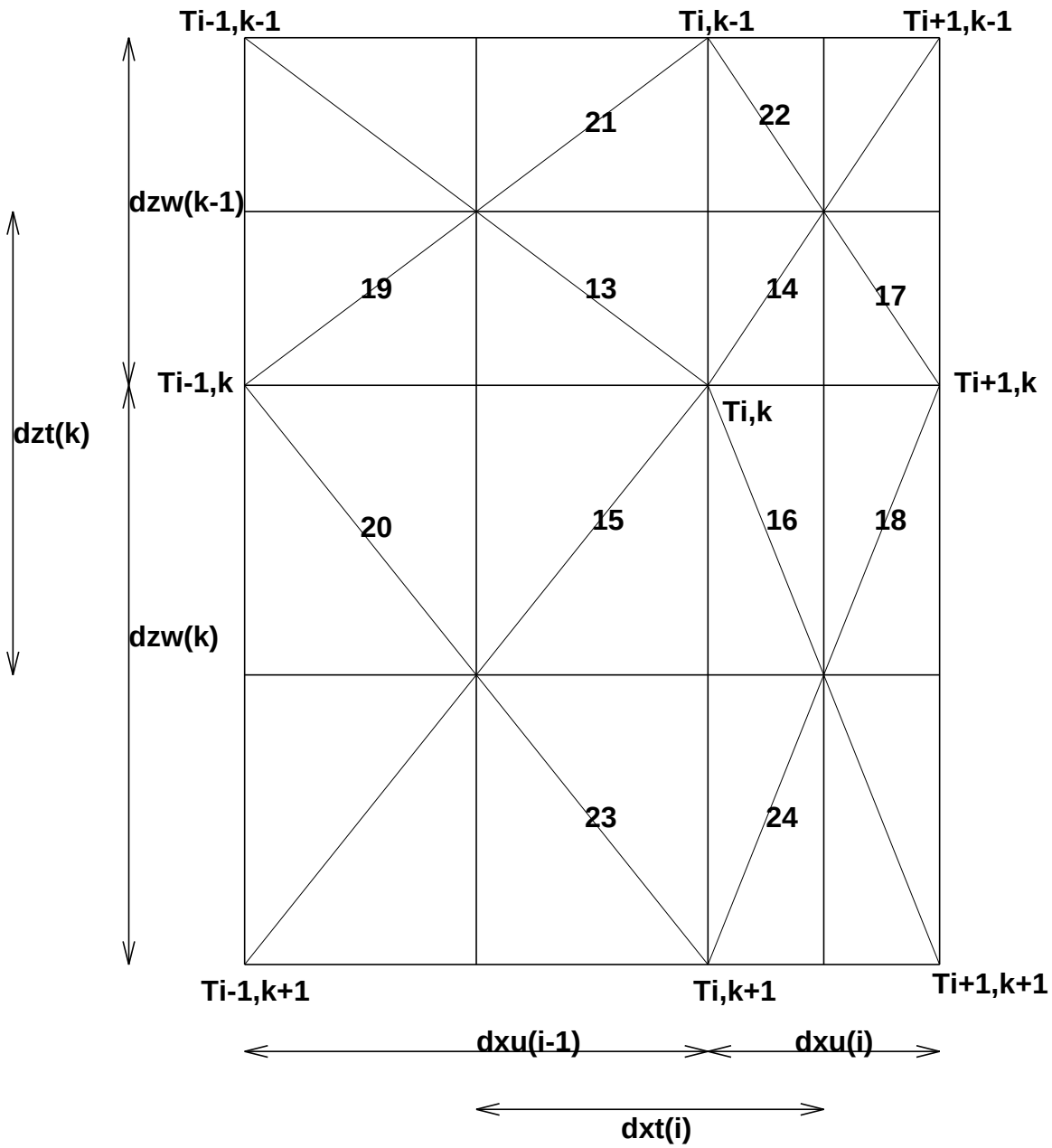


Figure C.3: MOM x-z plane and its partitioning into 12 quarter cells. The generally nonconstant grid spacing is indicated, which implies a non-centered T-point. This figure assumes that the vertical grid spacing is dependent only on the vertical grid position. For the more general case with partial vertical cells, the vertical grid spacing is a function of both the vertical and horizontal grid positions (see Figure C.4).

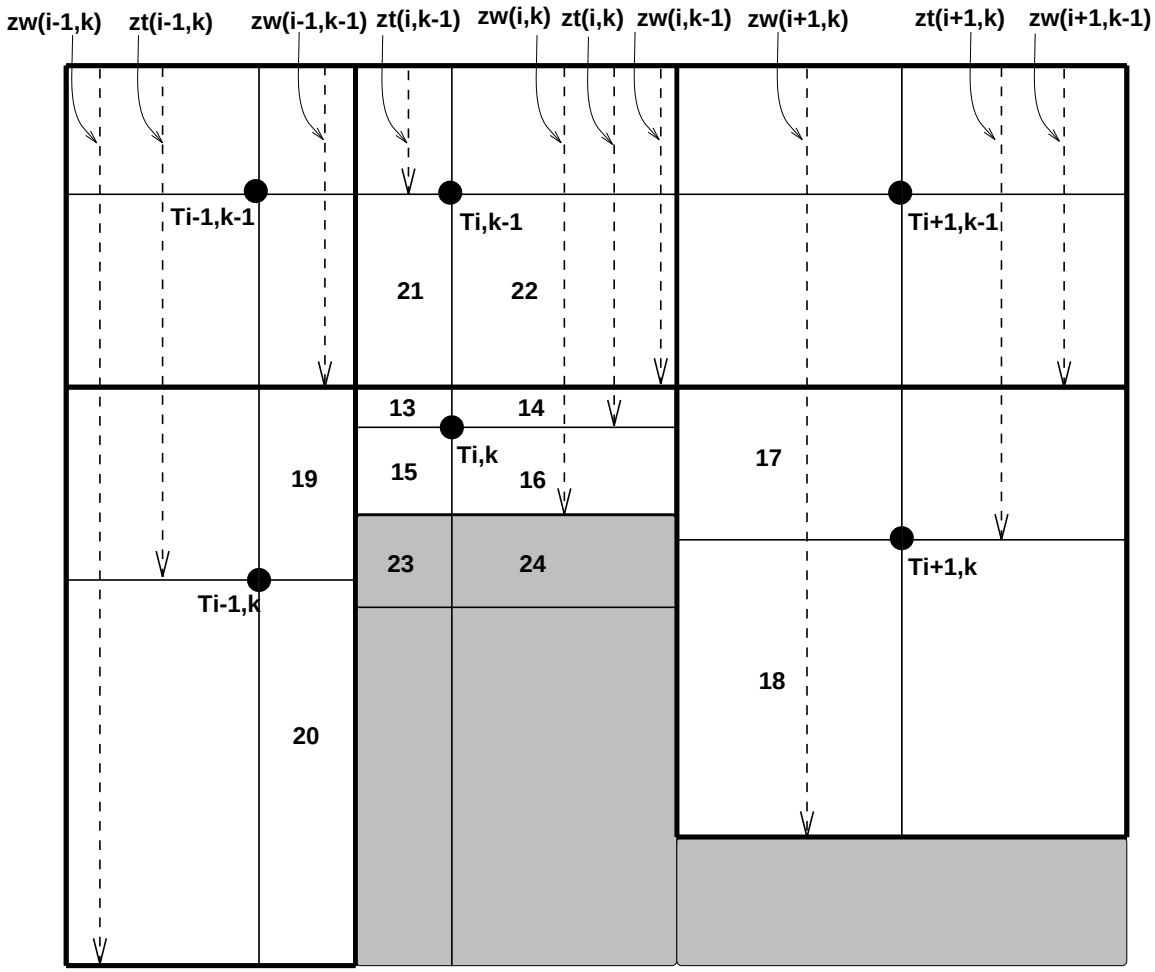


Figure C.4: MOM x-z plane for the case with partial bottom cells. T-cells are surrounded by dark solid lines. Rock is shaded. The thin solid lines define the quarter cells. The vertical position of the T-point in a T-cell maintains the same ratio for all the cells, whether full or partial. The 12 quarter cells corresponding to those in Figure C.3 are shown, where Figure C.3 used full vertical cells. Note that quarter-cells 23 and 24 are in rock for this particular example. Whether they are in rock or are ocean cells, quarter-cells 23 and 24 have the same volume as quarter cells 15 and 16 (as for the full cell case shown in Figure C.3). The vertical distances from the ocean surface to the bottom of the T-cells (zw) and to the T-cell center (zt) are shown by dashed lines.

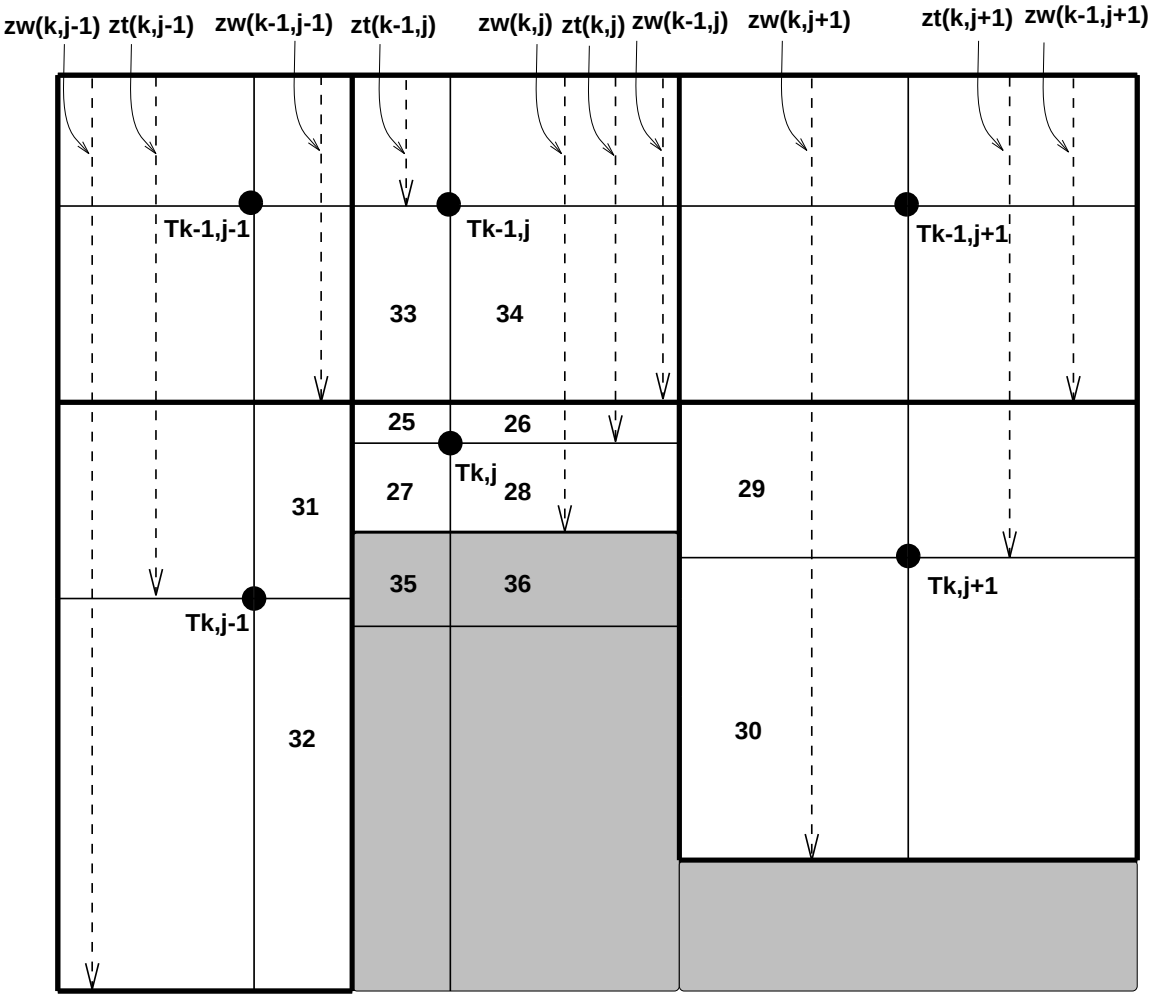


Figure C.5: MOM y-z plane for the case with partial bottom cells. Otherwise, as in Figure C.4.

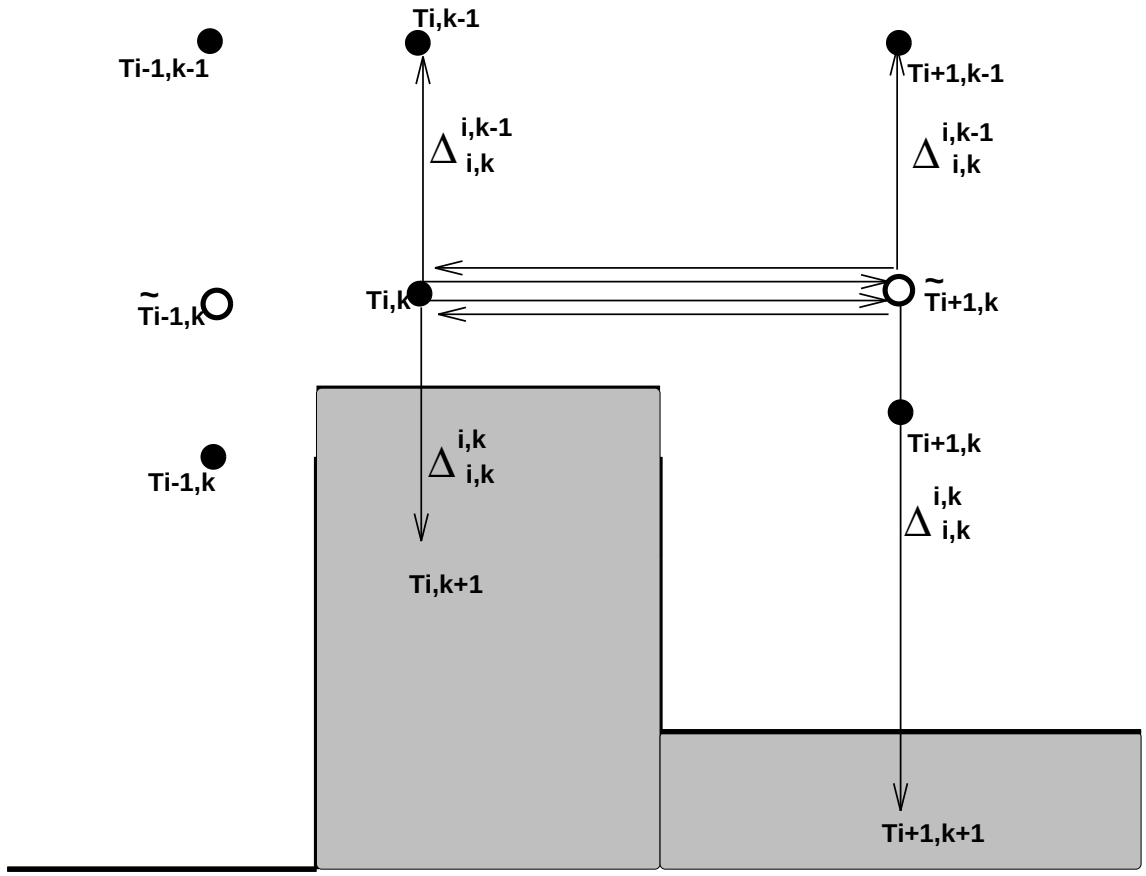


Figure C.6: Stencil for computing the zonal component to the small angle isoneutral flux $F^{x\,small}_{i,k}$, which is located at the east face of T-cell $T_{i,k}$. Shown are four density triads, each weighted by the relevant vertical spacing factor Δ. For the two triads extending upwards, the weighting is $\Delta^{(i,k-1)}_{(i,k)}$, whereas the two triads extending downwards are weighted by $\Delta^{(i,k)}_{(i,k)}$. The tracer points with the open circles denote tracer values which are determined through linear interpolation. For the two triads extending into rock (shaded regions), they are zeroed out in the code since they should not contribute to the diffusion operator.

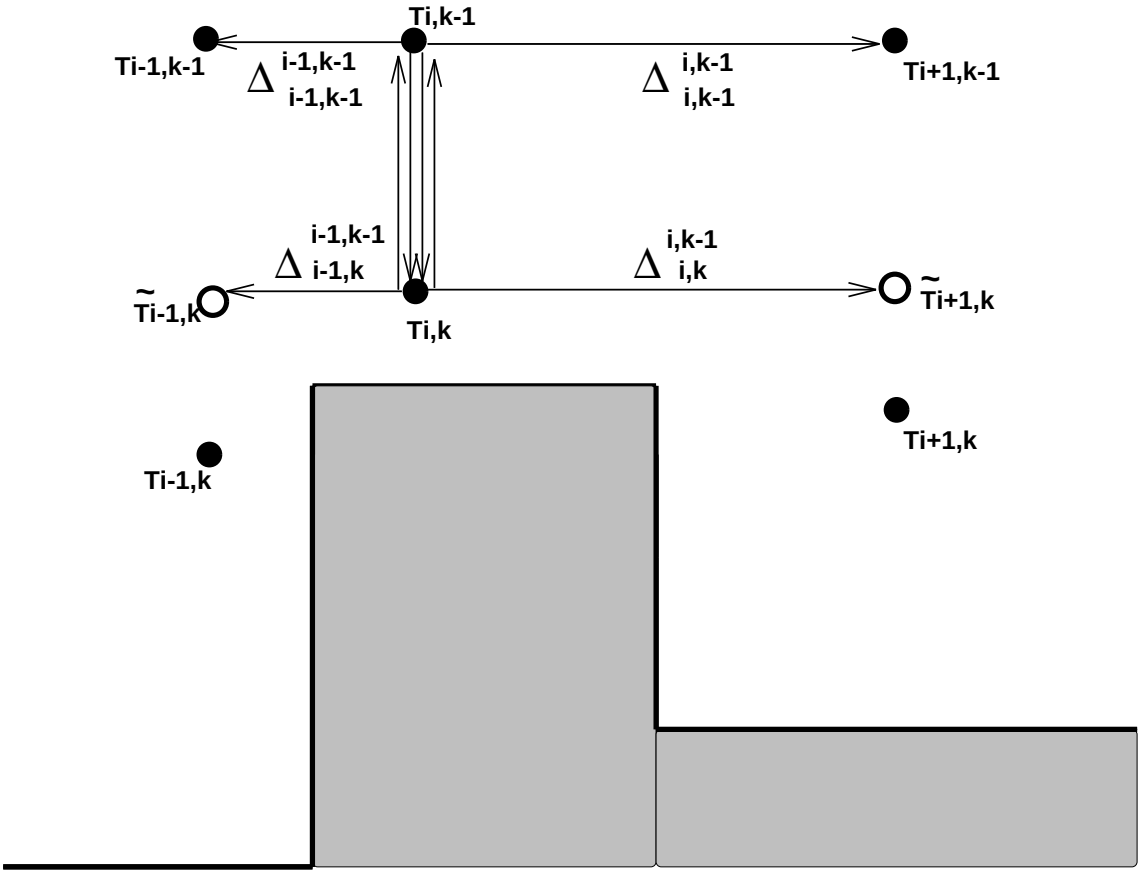


Figure C.7: Stencil for computing the vertical component to the small angle isoneutral flux $F^{z\,small}_{i,k-1}$, which is located at the bottom face of T-cell $T_{i,k-1}$. Shown are four density triads, each weighted by the relevant vertical spacing factor Δ. Note the rotation of the triads relative to those used to construct the zonal flux in Figure C.6. Each of the four triads in this figure employ a generally different vertical weighting factor Δ.

Appendix D

Horizontal friction discretization

The purpose of this appendix is to present a derivation of MOM's discretized horizontal friction operator. Much here follows the functional approach used for the isoneutral diffusion discretization in Appendix C. However, the present discussion is considerably simpler because of the need to only discretize a two dimensional (x-y plane) operator, and because there are no subtleties associated with neutral directions. The approach here is therefore a straightforward application of the functional formalism outlined in Section 9.7 for the continuum friction, and Appendix C for the diffusion operator.

Brief history of this method in MOM:

- Summer 1999: Discretization derived and tested on one-processor and only for fully opened memory window. This scheme will *not* be made available for use in MOM 3 as it is motivated largely by the desire to introduce generalized horizontal coordinates, which is being considered for MOM 4 but not MOM 3.

D.1 Motivation and summary

In a horizontal plane with second order numerics, the discrete Laplacian forms a 5-point stencil. This stencil precludes computational modes, and hence always acts to smooth. It is for this reason that many ocean models employ a Laplacian operator for dissipating momentum in order to satisfy various numerical stability requirements.

On a sphere and in the presence of non-homogeneous grids, boundaries, and various flow regimes, a Laplacian operator which uses a constant viscosity is often insufficient and generally sub-optimal in the sense that it will over-dissipate the simulation. The paper by Griff ies and Hallberg (1999) provide discussion which motivates consideration of the Smagorinsky viscosity in either a second order (Laplacian) or fourth order (biharmonic) friction operator. Additional considerations necessary to preserve angular momentum on a sphere warrant added sophistication to the operator beyond that of the familiar Cartesian form of the Laplacian operator.

Until Summer 1999, MOM has handled the added complications due to sphericity and non-constant viscosities in two basic ways. First, there are the formulae of Bryan (1969) and Wajsowicz (1993), in which friction is written as a "Laplacian plus metric" form. The "Laplacian" is the spherical coordinate form of a Laplacian operator acting on a component of the velocity, as if the component was a scalar field. The "metric term" is an added piece which accounts for the sphericity of the earth and provides for a proper angular momentum budget. Bryan (1969) used metric terms appropriate for constant viscosities, and Wajsowicz (1993)

showed how to add the metric terms appropriate for non-constant viscosities. Standard discretizations of these metric terms on a B-grid lead to computational modes. The presence of the scalar Laplacian, however, can in many situations, especially with constant viscosities, render these modes harmless.

The second manner of discretizing friction is to apply the tensor formalism of Smagorinsky (1963, 1993) in the process of discretizing the Smagorinsky nonlinear viscosity method. This was the approach of Rosati and Miyakoda (1988). In the continuum, this approach is equivalent to Wajsowicz (1993) (see Section 9.8 for details). Yet on the discrete lattice, there are differences. Furthermore, the tensor formalism is considerably more concise, and the ability to generalize to arbitrary orthogonal curvilinear coordinates is cleaner. Chapter 9 provides a full account of these issues.

Experience with the Smagorinsky scheme at GFDL has recently matured to the point of realizing that both of the above approaches to discretization can lead to unsatisfying solutions. Most notably, some model results with the Laplacian Smagorinsky scheme have shown grid noise in the velocity field. The reason for this noise is not well known, but it is likely due to the presence of computational modes which arise in either of the discretizations. Additionally, there is an effort underway to re-write MOM so that it employs general horizontal curvilinear coordinates. Hence, it is desirable to discretize friction in the manner described in Chapter 9, in particular through the tidy expressions (9.126) and (9.127). Otherwise, it will be necessary to introduce cumbersome metric terms which are not simple to derive in general.

In order to derive a discretization for an operator, one can proceed in numerous manners. The first approach might be to take the continuum form and provide reasonable second order discretizations, performing averaging whenever needed to place quantities in their proper position on the grid. This has been the approach taken for the friction in the past with MOM, where the continuum formulas of Bryan (1969) and Wajsowicz (1993), or Smagorinsky (1963, 1993), were the starting point. As discussed in Appendix C, this direct approach led to considerable difficulty with the isoneutral diffusion operator. Although it has not led to such serious problems for friction, it has not been satisfying for the reasons mentioned above.

After having some success with the reformulation of isoneutral diffusion, in which the diffusion operator was shown to be equivalent to the functional derivative of the tracer variance, it was decided to attempt similar approaches for the friction. As shown in Section 9.7, the friction operator is equivalent to the functional derivative of the kinetic energy dissipation. Both of these results follow from the self-adjointness of the diffusion and friction operators. Through this connection, the general approach is to first discretize the functional. Thereafter, the functional derivative of the discrete functional is used to derive the discretized friction.

It is notable that functional discretization of the friction operator leads to the use of velocity triads. These triads result because of the B-grid in which both velocity components are at a single point. Such triads will not emerge, for example, if working on a C-grid. For discretization of the isoneutral diffusion operator, density triads, as seen in Appendix C, are fundamental. Triads emerge for diffusion since all quantities of interest live on the tracer grid. Hence, discrete isoneutral diffusion, derived from a functional, will contain triads on any grid.

D.2 Review of the continuum results

As discussed in Section 9.7, the continuum functional for horizontal friction is given by

$$\mathcal{S} = -\rho_o \int d\xi^1 \, d\xi^2 \, dz \, \sqrt{\mathcal{G}} \, A \, (D_T^2 + D_S^2). \tag{D.1}$$

In this equation, the horizontal tension is

$$D_T = \frac{h_2}{h_1}\left(\frac{h_1}{h_2}\,u^1\right)_{,1} - \frac{h_1}{h_2}\left(\frac{h_2}{h_1}\,u^2\right)_{,2} \tag{D.2}$$

and the horizontal shearing strain is

$$D_S = \frac{h_1}{h_2}\,u^1_{,2} + \frac{h_2}{h_1}\,u^2_{,1}. \tag{D.3}$$

The infinitesimal horizontal distance between two points is written

$$\begin{aligned} ds^2 &= g_{11}\,(d\xi^1)^2 + g_{22}\,(d\xi^2)^2 + dz^2 \\ &= (h_1\,d\xi^1)^2 + (h_2\,d\xi^2)^2 + dz^2 \\ &= dx^2 + dy^2 + dz^2. \end{aligned} \tag{D.4}$$

The non-negative metric components h_1 and h_2 are generally functions of both horizontal directions (ξ^1, ξ^2). In spherical coordinates, with $(\xi^1, \xi^2) = (\lambda, \phi)$ longitude and latitude,

$$ds^2 = (a\,\cos\phi\,d\lambda)^2 + (a\,d\phi)^2 + dz^2 \tag{D.5}$$

The square-root of the metric determinant takes the form

$$\sqrt{\mathcal{G}} = h_1\,h_2\,h_3 \tag{D.6}$$

with $h_3 = 1$. The invariant horizontal volume element is therefore given by $\sqrt{\mathcal{G}}\,d\xi^1\,d\xi^2\,dz$, which in spherical coordinates takes the form $a^2\,\cos^2\phi\,d\lambda\,d\phi\,dz$. The velocity vector components u^a are related to the physical tensor components through

$$(u, v, w) = (h_1\,u^1, h_2\,u^2, w). \tag{D.7}$$

It is u, v which are time stepped in the ocean model. However, for purposes of deriving the discretization, it is important to recognize their distinction from u^a. In terms of the physical velocity and partial derivatives, the tension and strain take the form

$$\begin{aligned} D_T &= h_2\,(u/h_2)_{,x} - h_1\,(v/h_1)_{,y} \\ &= (a\,\cos\phi)^{-1}\,u_{,\lambda} - a^{-1}\,\cos\phi\,(v/\cos\phi)_{,\phi} \end{aligned} \tag{D.8}$$

$$\begin{aligned} D_S &= h_1\,(u/h_1)_{,y} + h_2\,(v/h_2)_{,x} \\ &= \cos\phi\,(u/a\,\cos\phi)_{,\phi} + (a\,\cos\phi)^{-1}\,v_{,\lambda} \end{aligned} \tag{D.9}$$

where the second expression introduced the spherical coordinate form. Given the functional, its functional derivative yields the friction vector through the relation

$$\frac{1}{2\,\rho_o}\,\frac{\delta\,\mathcal{S}}{\delta u^a} = g_{ab}\,F^b, \tag{D.10}$$

where $a, b = 1, 2$. The physical components of the friction are

$$F^x = h_2^{-2}\,(h_2^2\,A\,D_T)_{,x} + h_1^{-2}\,(h_1^2\,A\,D_S)_{,y} \tag{D.11}$$

$$F^y = -h_1^{-2}\,(h_1^2\,A\,D_T)_{,y} + h_2^{-2}\,(h_2^2\,A\,D_S)_{,x}, \tag{D.12}$$

where $F^x = h_1\,F^1$ and $F^y = h_2\,F^2$.

D.3 Discretization of the functional

The goal of this section is to provide a consistent discretization of the functional for horizontal friction. Spherical coordinates will be used, with some sprinkling of ideas relevant for general orthogonal curvilinear coordinates. Since all considerations in this appendix are for the horizontal deformations, the friction arising from vertical deformations is dropped, as is the vertical grid label.

D.3.1 General form of the discrete functional

The discrete friction functional takes the form

$$\begin{aligned} \mathcal{S} &= -\rho_o \sum_{i,j} \sum_{n=1}^{12} V(n)\, A(n)\, (D_T^2(n) + D_S^2(n)) \\ &\equiv \sum_{i,j} \mathcal{S}_{i,j}. \end{aligned} \tag{D.13}$$

Figure D.1 illustrates the nearest neighbor stencil used for discretizing the functional. The summation $n = 1, 12$ arises from the 12 subcells to which the velocity point $U_{i,j}$ contributes when discretizing the functional. $V(n)$ are the volumes of each of the subcells, $A(n)$ are the viscosities, and $D_T^2(n)$ and $D_S^2(n)$ are the corresponding tensions and strains. The stencil is directly analogous to that used for discretizing the isoneutral diffusion functional shown in Figure C.2.

For a finite velocity grid cell $U_{i,j}$, the friction acting on the velocity is given by the discrete functional derivative

$$\frac{1}{2\,\rho_o\, V_{i,j}^U} \frac{\partial \mathcal{S}}{\partial (u^a)_{i,j}} = g_{ab}\, F^b. \tag{D.14}$$

Equivalently, the physical components of the friction vector are

$$F^x = \frac{1}{2\,\rho_o\, (h_1)_{i,j}\, V_{i,j}^U} \frac{\partial \mathcal{S}}{\partial (u^1)_{i,j}} \tag{D.15}$$

$$F^y = \frac{1}{2\,\rho_o\, (h_2)_{i,j}\, V_{i,j}^U} \frac{\partial \mathcal{S}}{\partial (u^2)_{i,j}}. \tag{D.16}$$

In spherical coordinates, $(h_1)_{i,j} = a \cos \phi_j^U$ and $(h_2)_{i,j} = a$. The velocity cell volume $V_{i,j}^U$ accounts for the differing dimensions of the Kronecker and Dirac delta functions. That is,

$$V^{-1}\, \delta^{ab} \rightarrow \delta(x^a - x^b), \tag{D.17}$$

where the zero volume limit is taken.

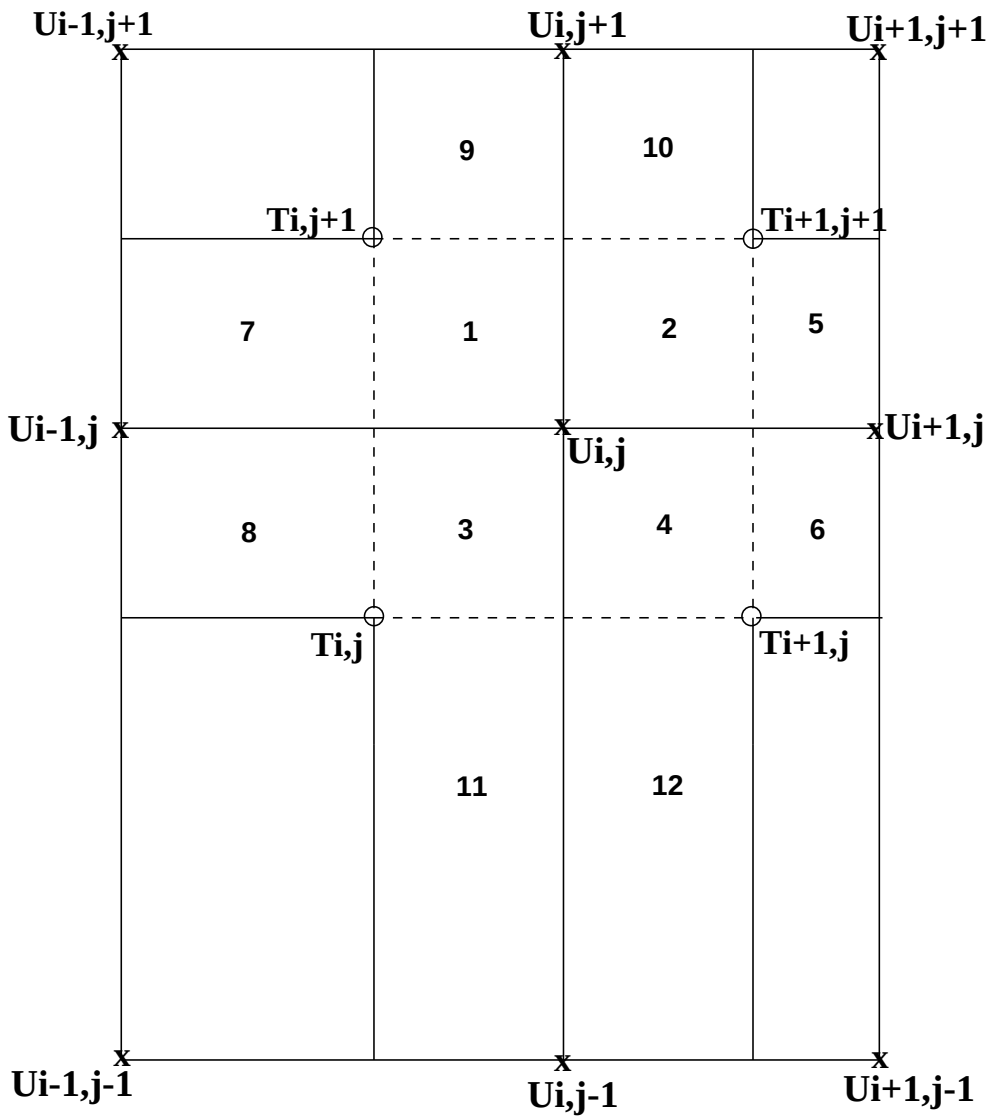


Figure D.1: Stencil for the discrete frictional functional. The 12 quarter cells each contain contributions to the functional from the central velocity point $U_{i,j}$. In general, the grid dimensions are defined as follows: $dxu_{i,j}$ represents the zonal distance between tracer points $T_{i,j}$ and $T_{i+1,j}$, $dyu_{i,j}$ is the meridional distance between tracer points $T_{i,j}$ and $T_{i,j+1}$, $dxt_{i+1,j}$ is the zonal distance between velocity points $U_{i,j}$ and $U_{i+1,j}$, and $dyt_{i,j+1}$ is the meridional distance between velocity points $U_{i,j}$ and $U_{i,j+1}$,. With spherical coordinates, $dxu_{i,j} \rightarrow \cos\phi_j^U \, dxu_i$, which now represents the zonal dimension of the $U_{i,j}$ cell, $dyu_{i,j} \rightarrow dyu_j$ represents the meridional dimension of the $U_{i,j}$ cell, $dxt_{i+1,j} \rightarrow \cos\phi_j^T \, dxt_{i+1}$, represents the zonal dimension of the $T_{i+1,j}$ cell, $dyt_{i,j+1} \rightarrow dyt_{j+1}$ represents the meridional dimension of the $T_{i,j+1}$ cell. For spherical coordinates, the velocity points, denoted by "x", are always defined at the center of the velocity cells, denoted by the dashed lines. The tracer points, denoted by "o", are generally not at the center of the tracer cells.

D.3.2 Subcell volumes

In spherical coordinates, the volumes of the 12 subcells are

$$V(1) = V(2) = V(3) = V(4) = \frac{1}{4}\, dxu_i \cos\phi_j^U\, dyu_j\, dhu_{i,k,j} \tag{D.18}$$

$$V(5) = V(6) = \frac{1}{4}\, dxu_{i+1} \cos\phi_j^U\, dyu_j\, dhu_{i+1,k,j} \tag{D.19}$$

$$V(7) = V(8) = \frac{1}{4}\, dxu_{i-1} \cos\phi_j^U\, dyu_j\, dhu_{i-1,k,j} \tag{D.20}$$

$$V(9) = V(10) = \frac{1}{4}\, dxu_i \cos\phi_{j+1}^U\, dyu_{j+1}\, dhu_{i,k,j+1} \tag{D.21}$$

$$V(11) = V(12) = \frac{1}{4}\, dxu_i \cos\phi_{j-1}^U\, dyu_{j-1}\, dhu_{i,k,j-1} \tag{D.22}$$

D.3.3 Derivative operators

For spherical coordinates, the horizontal finite difference derivative operators are given by

$$\delta_x\, u_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\cos\phi_j^U\, dxt_{i+1}} \tag{D.23}$$

$$\delta_y\, u_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{dyt_{j+1}}. \tag{D.24}$$

D.3.4 Tension for the subcells

In spherical coordinates, the tension in the 12 subcells are

$$D_T(1) = \delta_x\, u_{i-1,j} - \cos\phi_{j+1}^T\, \delta_y(v_{i,j}/\cos\phi_j^U) \tag{D.25}$$

$$D_T(2) = \delta_x\, u_{i,j} - \cos\phi_{j+1}^T\, \delta_y(v_{i,j}/\cos\phi_j^U) \tag{D.26}$$

$$D_T(3) = \delta_x\, u_{i-1,j} - \cos\phi_j^T\, \delta_y(v_{i,j-1}/\cos\phi_{j-1}^U) \tag{D.27}$$

$$D_T(4) = \delta_x\, u_{i,j} - \cos\phi_j^T\, \delta_y(v_{i,j-1}/\cos\phi_{j-1}^U) \tag{D.28}$$

$$D_T(5) = \delta_x\, u_{i,j} - \cos\phi_{j+1}^T\, \delta_y(v_{i+1,j}/\cos\phi_j^U) \tag{D.29}$$

$$D_T(6) = \delta_x\, u_{i,j} - \cos\phi_j^T\, \delta_y(v_{i+1,j-1}/\cos\phi_{j-1}^U) \tag{D.30}$$

$$D_T(7) = \delta_x\, u_{i-1,j} - \cos\phi_{j+1}^T\, \delta_y(v_{i-1,j}/\cos\phi_j^U) \tag{D.31}$$

$$D_T(8) = \delta_x\, u_{i-1,j} - \cos\phi_j^T\, \delta_y(v_{i-1,j-1}/\cos\phi_{j-1}^U) \tag{D.32}$$

$$D_T(9) = \delta_x\, u_{i-1,j+1} - \cos\phi_{j+1}^T\, \delta_y(v_{i,j}/\cos\phi_j^U) \tag{D.33}$$

$$D_T(10) = \delta_x\, u_{i,j+1} - \cos\phi_{j+1}^T\, \delta_y(v_{i,j}/\cos\phi_j^U) \tag{D.34}$$

$$D_T(11) = \delta_x\, u_{i-1,j-1} - \cos\phi_j^T\, \delta_y(v_{i,j-1}/\cos\phi_{j-1}^U) \tag{D.35}$$

$$D_T(12) = \delta_x\, u_{i,j-1} - \cos\phi_j^T\, \delta_y(v_{i,j-1}/\cos\phi_{j-1}^U) \tag{D.36}$$

D.3.5 Strain for the subcells

In spherical coordinates, the strain in the 12 subcells are

$$D_S(1) = \cos\phi_{j+1}^T\, \delta_y\,(u_{i,j}/\cos\phi_j^U) + \delta_x\, v_{i-1,j} \tag{D.37}$$

$$D_S(2) = \cos\phi^T_{j+1}\,\delta_y\,(u_{i,j}/\cos\phi^U_j) + \delta_x\,v_{i,j} \tag{D.38}$$
$$D_S(3) = \cos\phi^T_{j}\,\delta_y\,(u_{i,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i-1,j} \tag{D.39}$$
$$D_S(4) = \cos\phi^T_{j}\,\delta_y\,(u_{i,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i,j} \tag{D.40}$$
$$D_S(5) = \cos\phi^T_{j+1}\,\delta_y\,(u_{i+1,j}/\cos\phi^U_j) + \delta_x\,v_{i,j} \tag{D.41}$$
$$D_S(6) = \cos\phi^T_{j}\,\delta_y\,(u_{i+1,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i,j} \tag{D.42}$$
$$D_S(7) = \cos\phi^T_{j+1}\,\delta_y\,(u_{i-1,j}/\cos\phi^U_j) + \delta_x\,v_{i-1,j} \tag{D.43}$$
$$D_S(8) = \cos\phi^T_{j}\,\delta_y\,(u_{i-1,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i-1,j} \tag{D.44}$$
$$D_S(9) = \cos\phi^T_{j+1}\,\delta_y\,(u_{i,j}/\cos\phi^U_j) + \delta_x\,v_{i-1,j+1} \tag{D.45}$$
$$D_S(10) = \cos\phi^T_{j+1}\,\delta_y\,(u_{i,j}/\cos\phi^U_j) + \delta_x\,v_{i,j+1} \tag{D.46}$$
$$D_S(11) = \cos\phi^T_{j}\,\delta_y\,(u_{i,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i-1,j-1} \tag{D.47}$$
$$D_S(12) = \cos\phi^T_{j}\,\delta_y\,(u_{i,j-1}/\cos\phi^U_{j-1}) + \delta_x\,v_{i,j-1} \tag{D.48}$$

D.4 Discrete friction

Given the form of the functional, it is now a matter of performing the functional derivatives with respect to $(u^1)_{i,j}$ and $(u^2)_{i,j}$, where in spherical coordinates,

$$u_{i,j} = a\,\cos\phi^U_j\,(u^1)_{i,j} \tag{D.49}$$
$$v_{i,j} = a\,(u^2)_{i,j}, \tag{D.50}$$

with

$$u^1 = D\lambda/Dt \tag{D.51}$$
$$u^2 = D\phi/Dt. \tag{D.52}$$

In general, the functional derivative of $\mathcal{S}_{i,j}$ with respect to $(u^b)_{i,j}$ is given by

$$\frac{\partial\mathcal{S}_{i,j}}{\partial(u^b)_{i,j}} = -2\,\rho_o\sum_{n=1}^{12}V(n)\,A(n)\left(D_T(n)\,\frac{\partial D_T(n)}{\partial(u^b)_{i,j}} + D_S(n)\,\frac{\partial D_S(n)}{\partial(u^b)_{i,j}}\right), \tag{D.53}$$

where $b = 1, 2$.

D.4.1 Functional derivative of the physical velocity components

The physical velocity components have the following functional derivatives with respect to the tensorial velocity components

$$\frac{\partial u_{i1,j1}}{\partial(u^1)_{i2,j2}} = a\,\cos\phi^U_{j1}\,\delta^{i2}_{i1}\,\delta^{j2}_{j1} \tag{D.54}$$
$$\frac{\partial v_{i1,j1}}{\partial(u^2)_{i2,j2}} = a\,\delta^{i2}_{i1}\,\delta^{j2}_{j1}. \tag{D.55}$$

All other functional derivatives vanish.

D.4.2 Functional derivative of D_T

The functional derivative of the tension within the 12 subcells is given by

$$\frac{\partial D_T(1)}{\partial (u^b)_{i,j}} = \left(\frac{a}{dxt_i}\right)\delta_b^1 + \left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}\cos\phi_j^U}\right)\delta_b^2 \tag{D.56}$$

$$\frac{\partial D_T(2)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{dxt_{i+1}}\right)\delta_b^1 + \left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}\cos\phi_j^U}\right)\delta_b^2 \tag{D.57}$$

$$\frac{\partial D_T(3)}{\partial (u^b)_{i,j}} = \left(\frac{a}{dxt_i}\right)\delta_b^1 - \left(\frac{a\cos\phi_j^T}{dyt_j\cos\phi_j^U}\right)\delta_b^2 \tag{D.58}$$

$$\frac{\partial D_T(4)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{dxt_{i+1}}\right)\delta_b^1 - \left(\frac{a\cos\phi_j^T}{dyt_j\cos\phi_j^U}\right)\delta_b^2 \tag{D.59}$$

$$\frac{\partial D_T(5)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{dxt_{i+1}}\right)\delta_b^1 \tag{D.60}$$

$$\frac{\partial D_T(6)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{dxt_{i+1}}\right)\delta_b^1 \tag{D.61}$$

$$\frac{\partial D_T(7)}{\partial (u^b)_{i,j}} = \left(\frac{a}{dxt_i}\right)\delta_b^1 \tag{D.62}$$

$$\frac{\partial D_T(8)}{\partial (u^b)_{i,j}} = \left(\frac{a}{dxt_i}\right)\delta_b^1 \tag{D.63}$$

$$\frac{\partial D_T(9)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}\cos\phi_j^U}\right)\delta_b^2 \tag{D.64}$$

$$\frac{\partial D_T(10)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}\cos\phi_j^U}\right)\delta_b^2 \tag{D.65}$$

$$\frac{\partial D_T(11)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_j^T}{dyt_j\cos\phi_j^U}\right)\delta_b^2 \tag{D.66}$$

$$\frac{\partial D_T(12)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_j^T}{dyt_j\cos\phi_j^U}\right)\delta_b^2. \tag{D.67}$$

D.4.3 Functional derivative of D_S

The functional derivative of the strain within the 12 subcells is given by

$$\frac{\partial D_S(1)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}}\right)\delta_b^1 + \left(\frac{a}{\cos\phi_j^U\, dxt_i}\right)\delta_b^2 \tag{D.68}$$

$$\frac{\partial D_S(2)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}}\right)\delta_b^1 - \left(\frac{a}{\cos\phi_j^U\, dxt_{i+1}}\right)\delta_b^2 \tag{D.69}$$

$$\frac{\partial D_S(3)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_j^T}{dyt_j}\right)\delta_b^1 + \left(\frac{a}{\cos\phi_j^U\, dxt_i}\right)\delta_b^2 \tag{D.70}$$

$$\frac{\partial D_S(4)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_j^T}{dyt_j}\right)\delta_b^1 - \left(\frac{a}{\cos\phi_j^U\, dxt_{i+1}}\right)\delta_b^2 \tag{D.71}$$

$$\frac{\partial D_S(5)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{\cos\phi_j^U\, dxt_{i+1}}\right)\delta_b^2 \tag{D.72}$$

$$\frac{\partial D_S(6)}{\partial (u^b)_{i,j}} = -\left(\frac{a}{\cos\phi_j^U\, dxt_{i+1}}\right)\delta_b^2 \tag{D.73}$$

$$\frac{\partial D_S(7)}{\partial (u^b)_{i,j}} = \left(\frac{a}{\cos\phi_j^U\, dxt_i}\right)\delta_b^2 \tag{D.74}$$

$$\frac{\partial D_S(8)}{\partial (u^b)_{i,j}} = \left(\frac{a}{\cos\phi_j^U\, dxt_i}\right)\delta_b^2 \tag{D.75}$$

$$\frac{\partial D_S(9)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}}\right)\delta_b^1 \tag{D.76}$$

$$\frac{\partial D_S(10)}{\partial (u^b)_{i,j}} = -\left(\frac{a\cos\phi_{j+1}^T}{dyt_{j+1}}\right)\delta_b^1 \tag{D.77}$$

$$\frac{\partial D_S(11)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_j^T}{dyt_j}\right)\delta_b^1 \tag{D.78}$$

$$\frac{\partial D_S(12)}{\partial (u^b)_{i,j}} = \left(\frac{a\cos\phi_j^T}{dyt_j}\right)\delta_b^1 \tag{D.79}$$

D.4.4 Rearrangement of terms for the zonal friction

Bringing things together for $b = 1$ leads to the functional derivative of $\mathcal{S}$

$$\begin{aligned}
-\frac{1}{2a\,\rho_o}\frac{\partial \mathcal{S}}{\partial (u^1)_{ij}} &= V(1)\left(\frac{A(1)\,D_T(1) + A(3)\,D_T(3)}{dxt_i} - \frac{A(2)\,D_T(2) + A(4)\,D_T(4)}{dxt_{i+1}}\right) \\
&+ V(7)\left(\frac{A(7)\,D_T(7) + A(8)\,D_T(8)}{dxt_i}\right) - V(5)\left(\frac{A(5)\,D_T(5) + A(6)\,D_T(6)}{dxt_{i+1}}\right) \\
&+ V(1)\left(\frac{\cos\phi_j^T\,[A(3)\,D_S(3) + A(4)\,D_S(4)]}{dty_j} - \frac{\cos\phi_{j+1}^T\,[A(1)\,D_S(1) + A(2)\,D_S(2)]}{dty_{j+1}}\right) \\
&+ V(11)\left(\frac{\cos\phi_j^T\,[A(11)\,D_S(11) + A(12)\,D_S(12)]}{dyt_j}\right) \\
&- V(9)\left(\frac{\cos\phi_{j+1}^T\,[A(9)\,D_S(9) + A(10)\,D_S(10)]}{dyt_{j+1}}\right)
\end{aligned} \tag{D.80}$$

Figure D.1 indicates that a velocity point $U_{i,j}$ is associated with four triads, each of which is used to construct a tension and strain along with a viscosity. Introducing the notation $(1,1)$, $(0,1)$, $(0,0)$, $(1,0)$ to denote the four triads, where $(1,1)$ = northeast triad, $(0,1)$ = northwest triad, $(0,0)$ = southwest triad, and $(1,0)$ = southeast triad (Figure D.2), the functional deriva-

tive can be written

$$\begin{aligned}
\frac{1}{2a\,\rho_o}\frac{\partial \mathcal{S}}{\partial (u^1)_{ij}} &= \frac{V(1)}{dxt_{i+1}}\left(\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i,j}+\frac{V(5)}{V(1)}\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i+1,j}\right)\\
&- \frac{V(1)}{dxt_i}\left(\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i,j}+\frac{V(7)}{V(1)}\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i-1,j}\right)\\
&+ \frac{V(1)\cos\phi^T_{j+1}}{dyt_{j+1}}\left(\left[(A\,D_S)^{(0,1)}+(A\,D_S)^{(1,1)}\right]_{i,j}+\frac{V(9)}{V(1)}\left[(A\,D_S)^{(0,0)}+(A\,D_S)^{(1,0)}\right]_{i,j+1}\right)\\
&- \frac{V(1)\cos\phi^T_{j}}{dyt_{j}}\left(\left[(A\,D_S)^{(0,0)}+(A\,D_S)^{(1,0)}\right]_{i,j}+\frac{V(11)}{V(1)}\left[(A\,D_S)^{(0,1)}+(A\,D_S)^{(1,1)}\right]_{i,j-1}\right).
\end{aligned} \tag{D.81}$$

The subcell volumes from Section D.3.2 brings the two terms proportional to dxt_i^{-1} to the form

$$\begin{aligned}
&\frac{V(1)}{dxt_{i+1}}\left(\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i,j}+\frac{V(5)}{V(1)}\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i+1,j}\right)\\
=\;& \frac{V(1)}{dxu_i\,dhu_{i,k,j}\,dxt_{i+1}}\\
&\left(dxu_i\,dhu_{i,k,j}\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i,j}+dxu_{i+1}\,dhu_{i+1,k,j}\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i+1,j}\right).
\end{aligned} \tag{D.82}$$

Likewise, the other terms can be brought to the form

$$\begin{aligned}
&\frac{V(1)}{dxt_{i}}\left(\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i,j}+\frac{V(7)}{V(1)}\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i-1,j}\right)\\
=\;& \frac{V(1)}{dxu_i\,dhu_{i,k,j}\,dxt_{i}}\\
&\left(dxu_i\,dhu_{i,k,j}\left[(A\,D_T)^{(0,1)}+(A\,D_T)^{(0,0)}\right]_{i,j}+dxu_{i-1}\,dhu_{i-1,k,j}\left[(A\,D_T)^{(1,1)}+(A\,D_T)^{(1,0)}\right]_{i-1,j}\right).
\end{aligned} \tag{D.83}$$

$$\begin{aligned}
&\frac{V(1)\cos\phi^T_{j+1}}{dyt_{j+1}}\left(\left[(A\,D_S)^{(0,1)}+(A\,D_S)^{(1,1)}\right]_{i,j}+\frac{V(9)}{V(1)}\left[(A\,D_S)^{(0,0)}+(A\,D_S)^{(1,0)}\right]_{i,j+1}\right)\\
=\;& \frac{V(1)\cos\phi^T_{j+1}}{\cos\phi^U_j\,dyu_j\,dhu_{i,k,j}\,dyt_{j+1}}\\
&\left(\cos\phi^U_j\,dyu_j\,dhu_{i,k,j}\left[(A\,D_S)^{(0,1)}+(A\,D_S)^{(1,1)}\right]_{i,j}\right.\\
+\;& \left.\cos\phi^U_{j+1}\,dyu_{j+1}\,dhu_{i,k,j+1}\left[(A\,D_S)^{(0,0)}+(A\,D_S)^{(1,0)}\right]_{i,j+1}\right)
\end{aligned} \tag{D.84}$$

$$\frac{V(1)\cos\phi^T_{j}}{dyt_{j}}\left(\left[(A\,D_S)^{(0,0)}+(A\,D_S)^{(1,0)}\right]_{i,j}+\frac{V(11)}{V(1)}\left[(A\,D_S)^{(0,1)}+(A\,D_S)^{(1,1)}\right]_{i,j-1}\right)$$

$$
\begin{aligned}
&= \frac{V(1)\,\cos\phi_j^T}{\cos\phi_j^U\, dyu_j\, dhu_{i,k,j}\, dyt_j} \\
&\quad \Bigg(\cos\phi_j^U\, dyu_j\, dhu_{i,k,j} \left[(A\,D_S)^{(0,0)} + (A\,D_S)^{(1,0)} \right]_{i,j} \\
&+ \cos\phi_{j-1}^U\, dyu_{j-1}\, dhu_{i,k,j-1} \left[(A\,D_S)^{(0,1)} + (A\,D_S)^{(1,1)} \right]_{i,j-1} \Bigg). \qquad \text{(D.85)}
\end{aligned}
$$

The volume of the velocity cell is $V_{i,k,j}^U = 4\,V(1)$. Use of equation (D.15), with $(h_1)_{i,j} = a\,\cos\phi_j^U$ for spherical coordinates, leads to

$$
\begin{aligned}
F_{i,k,j}^x &= \left(\frac{1}{4\,\cos\phi_j^U\, dxu_i\, dhu_{i,k,j}} \right) \\
&\quad \Bigg[\frac{1}{dxt_{i+1}} \Bigg(dxu_i\, dhu_{i,k,j} \left[(A\,D_T)^{(1,1)} + (A\,D_T)^{(1,0)} \right]_{i,j} \\
&+ dxu_{i+1}\, dhu_{i+1,k,j} \left[(A\,D_T)^{(0,1)} + (A\,D_T)^{(0,0)} \right]_{i+1,j} \Bigg) \\
&- \frac{1}{dxt_i} \Bigg(dxu_i\, dhu_{i,k,j} \left[(A\,D_T)^{(0,1)} + (A\,D_T)^{(0,0)} \right]_{i,j} \\
&+ dxu_{i-1}\, dhu_{i-1,k,j} \left[(A\,D_T)^{(1,1)} + (A\,D_T)^{(1,0)} \right]_{i-1,j} \Bigg) \Bigg] \\
&+ \left(\frac{1}{4\,(\cos\phi_j^U)^2\, dyu_j\, dhu_{i,k,j}} \right) \\
&\quad \Bigg[\frac{\cos\phi_{j+1}^T}{dyt_{j+1}} \Bigg(\cos\phi_j^U\, dyu_j\, dhu_{i,k,j} \left[(A\,D_S)^{(0,1)} + (A\,D_S)^{(1,1)} \right]_{i,j} \\
&+ \cos\phi_{j+1}^U\, dyu_{j+1}\, dhu_{i,k,j+1} \left[(A\,D_S)^{(0,0)} + (A\,D_S)^{(1,0)} \right]_{i,j+1} \Bigg) \\
&- \frac{\cos\phi_j^T}{dyt_j} \Bigg(\cos\phi_j^U\, dyu_j\, dhu_{i,k,j} \left[(A\,D_S)^{(0,0)} + (A\,D_S)^{(1,0)} \right]_{i,j} \\
&+ \cos\phi_{j-1}^U\, dyu_{j-1}\, dhu_{i,k,j-1} \left[(A\,D_S)^{(0,1)} + (A\,D_S)^{(1,1)} \right]_{i,j-1} \Bigg) \Bigg] \\
&= \left(\frac{1}{4\,\cos\phi_j^U\, dxu_i\, dhu_{i,k,j}} \right) \\
&\quad \Bigg[\frac{1}{dxt_{i+1}} \sum_{ip=0}^{1} dxu_{i+ip}\, dhu_{i+ip,k,j} \sum_{jq=0}^{1} (A\,D_T)_{i+ip,j}^{(1-ip,jq)} \\
&- \frac{1}{dxt_i} \sum_{ip=0}^{1} dxu_{i-ip}\, dhu_{i-ip,k,j} \sum_{jq=0}^{1} (A\,D_T)_{i-ip,j}^{(ip,jq)} \Bigg] \\
&+ \left(\frac{1}{4\,(\cos\phi_j^U)^2\, dyu_j\, dhu_{i,k,j}} \right) \\
&\quad \Bigg[\frac{\cos\phi_{j+1}^T}{dyt_{j+1}} \sum_{jq=0}^{1} \cos\phi_{j+jq}^U\, dyu_{j+jq}\, dhu_{i,k,j+jq} \sum_{ip=0}^{1} (A\,D_S)_{i,j+jq}^{(ip,1-jq)}
\end{aligned}
$$

$$- \quad \frac{\cos\phi_j^T}{dyt_j} \sum_{jq=0}^{1} \cos\phi_{j-jq}^U \, dyu_{j-jq} \, dhu_{i,k,j-jq} \sum_{ip=0}^{1} (A\,D_S)_{i,j-jq}^{(ip,jq)} \Bigg]. \tag{D.86}$$

Comparison with the continuum zonal friction given by equation (D.11) indicates that the discretization is consistent; i.e., the discrete friction reduces to the continuum as the grid size goes to zero.

D.4.5 Rearrangement of terms for the meridional friction

Similar manipulations for the meridional friction leads to

$$\begin{aligned}
\frac{\cos\phi_j^U}{2\,\rho_o\,a}\frac{\partial\mathcal{S}}{\partial(u^2)_{i,j}} &= -\frac{\cos\phi_{j+1}^T\,V(1)}{dyt_{j+1}}\left[A(1)\,D_T(1) + A(2)\,D_T(2) + \frac{V(9)}{V(1)}\left(A(9)\,D_T(9) + A(10)\,D_T(10)\right)\right] \\
&+ \frac{\cos\phi_j^T\,V(1)}{dyt_j}\left[A(3)\,D_T(3) + A(4)\,D_T(4) + \frac{V(11)}{V(1)}\left(A(11)\,D_T(11) + A(12)\,D_T(12)\right)\right] \\
&+ \frac{V(1)}{dxt_{i+1}}\left[A(2)\,D_S(2) + A(4)\,D_S(4) + \frac{V(5)}{V(1)}\left(A(5)\,D_S(5) + A(6)\,D_S(6)\right)\right] \\
&- \frac{V(1)}{dxt_i}\left[A(1)\,D_S(1) + A(3)\,D_S(3) + \frac{V(7)}{V(1)}\left(A(7)\,D_S(7) + A(8)\,D_S(8)\right)\right],
\end{aligned} \tag{D.87}$$

which leads to the meridional friction

$$\begin{aligned}
F_{i,k,j}^y &= -\left(\frac{1}{4\,(\cos\phi_j^U)^2\,dyu_j\,dhu_{i,k,j}}\right) \\
&\quad \left[\frac{\cos\phi_{j+1}^T}{dyt_{j+1}}\left(\cos\phi_j^U\,dyu_j\,dhu_{i,k,j}\left[(A\,D_T)^{(0,1)} + (A\,D_T)^{(1,1)}\right]_{i,j}\right.\right. \\
&+ \left.\cos\phi_{j+1}^U\,dyu_{j+1}\,dhu_{i,k,j+1}\left[(A\,D_T)^{(0,0)} + (A\,D_T)^{(1,0)}\right]_{i,j+1}\right) \\
&- \frac{\cos\phi_j^T}{dyt_j}\left(\cos\phi_j^U\,dyu_j\,dhu_{i,k,j}\left[(A\,D_T)^{(0,0)} + (A\,D_T)^{(1,0)}\right]_{i,j}\right. \\
&+ \left.\left.\cos\phi_{j-1}^U\,dyu_{j-1}\,dhu_{i,k,j-1}\left[(A\,D_T)^{(0,1)} + (A\,D_T)^{(1,1)}\right]_{i,j-1}\right)\right] \\
&+ \left(\frac{1}{4\,\cos\phi_j^U\,dxu_i\,dhu_{i,k,j}}\right) \\
&\quad \left[\frac{1}{dxt_{i+1}}\left(dxu_i\,dhu_{i,k,j}\left[(A\,D_S)^{(1,0)} + (A\,D_S)^{(1,1)}\right]_{i,j}\right.\right. \\
&+ \left.dxu_{i+1}\,dhu_{i+1,k,j}\left[(A\,D_S)^{(0,0)} + (A\,D_S)^{(0,1)}\right]_{i+1,j}\right) \\
&- \frac{1}{dxt_i}\left(dxu_i\,dhu_{i,k,j}\left[(A\,D_S)^{(0,1)} + (A\,D_S)^{(0,0)}\right]_{i,j}\right. \\
&+ \left.\left.dxu_{i-1}\,dhu_{i-1,k,j}\left[(A\,D_S)^{(1,1)} + (A\,D_S)^{(1,0)}\right]_{i-1,j}\right)\right] \\
&= -\left(\frac{1}{4\,(\cos\phi_j^U)^2\,dyu_j\,dhu_{i,k,j}}\right)
\end{aligned}$$

$$
\begin{aligned}
& \left[\frac{\cos\phi^T_{j+1}}{dyt_{j+1}} \sum_{jq=0}^{1} \cos\phi^U_{j+jq}\, dyu_{j+jq}\, dhu_{i,k,j+jq} \sum_{ip=0}^{1} (A\, D_T)^{(ip,1-jq)}_{i,j+jq}\right. \\
- \quad & \left.\frac{\cos\phi^T_{j}}{dyt_{j}} \sum_{jq=0}^{1} \cos\phi^U_{j-jq}\, dyu_{j-jq}\, dhu_{i,k,j-jq} \sum_{ip=0}^{1} (A\, D_T)^{(ip,jq)}_{i,j-jq}\right] \\
+ \quad & \left(\frac{1}{4\cos\phi^U_j\, dxu_i\, dhu_{i,k,j}}\right) \\
& \left[\frac{1}{dxt_{i+1}} \sum_{ip=0}^{1} dxu_{i+ip}\, dhu_{i+ip,k,j} \sum_{jq=0}^{1} (A\, D_S)^{(1-ip,jq)}_{i+ip,j}\right. \\
- \quad & \left.\frac{1}{dxt_{i}} \sum_{ip=0}^{1} dxu_{i-ip}\, dhu_{i-ip,k,j} \sum_{jq=0}^{1} (A\, D_S)^{(ip,jq)}_{i-ip,j}\right]
\end{aligned}
\tag{D.88}
$$

Comparison with the continuum meridional friction given by equation (D.12) indicates that the discretization is consistent.

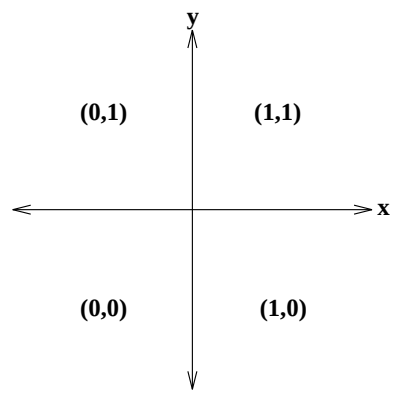


Figure D.2: Notation for the quadrants surrounding a velocity point.

D.5 Discretization of tension and strain for the quadrants

For each velocity point, there are four tensions and strains. Referring to Figure D.2, assuming the central point is $U_{i,j}$, one has for the tensions

$$
\begin{aligned}
(D_T)_{i,j,1,1} &= \delta_x\, u_{i,j} - \cos\phi^T_{j+1}\, \delta_y(v_{i,j}/\cos\phi^U_j) && \text{(D.89)}\\
(D_T)_{i,j,0,1} &= \delta_x\, u_{i-1,j} - \cos\phi^T_{j+1}\, \delta_y(v_{i,j}/\cos\phi^U_j) && \text{(D.90)}\\
(D_T)_{i,j,0,0} &= \delta_x\, u_{i-1,j} - \cos\phi^T_{j}\, \delta_y(v_{i,j-1}/\cos\phi^U_{j-1}) && \text{(D.91)}\\
(D_T)_{i,j,1,0} &= \delta_x\, u_{i,j} - \cos\phi^T_{j}\, \delta_y(v_{i,j-1}/\cos\phi^U_{j-1}), && \text{(D.92)}
\end{aligned}
$$

and for the strains

$$
\begin{aligned}
(D_S)_{i,j,1,1} &= \cos\phi^T_{j+1}\, \delta_y\, (u_{i,j}/\cos\phi^U_j) + \delta_x\, v_{i,j} && \text{(D.93)}\\
(D_S)_{i,j,0,1} &= \cos\phi^T_{j+1}\, \delta_y\, (u_{i,j}/\cos\phi^U_j) + \delta_x\, v_{i-1,j} && \text{(D.94)}
\end{aligned}
$$

$$(D_S)_{i,j,0,0} = \cos\phi_j^T\,\delta_y\,(u_{i,j-1}/\cos\phi_{j-1}^U) + \delta_x\,v_{i-1,j} \tag{D.95}$$

$$(D_S)_{i,j,1,0} = \cos\phi_j^T\,\delta_y\,(u_{i,j-1}/\cos\phi_{j-1}^U) + \delta_x\,v_{i,j}. \tag{D.96}$$

In general, the four tensions can be written

$$(D_T)_{i,j,ip,jq} = \delta_x\,u_{i+ip-1,j} - \cos\phi_{j+jq}^T\,\delta_y(v_{i,j+jq-1}/\cos\phi_{j+jq-1}^U) \tag{D.97}$$

and the four strains can be written

$$(D_S)_{i,j,ip,jq} = \cos\phi_{j+jq}^T\,\delta_y\,(u_{i,j+jq-1}/\cos\phi_{j+jq-1}^U) + \delta_x\,v_{i+ip-1,j}, \tag{D.98}$$

where $ip = 0, 1$ and $jq = 0, 1$. These four tensions and strains are computed in the model. They are then used to compute the friction operator. When the Smagorinsky scheme is enabled (option *velocity_horz_mix_smag*), they are used to compute the Smagorinsky viscosity as well.

D.6 Comments

It is important to note that the tension and strain are generally nonzero for interface points between land and sea, even though the velocity vanishes at such points. The reason is that when computing the tension and strain at such points, one reaches into the interior, which leads to a nonzero shear at the wall. It is therefore important *not* to mask the model's tension and strain fields in order to provide a full accounting of the generally strong shears next to no-slip walls.

In the special case of a Cartesian grid (e.g., model options *f_plane* or *beta_plane*), and with constant viscosity, the triad generated algorithm reduces to the familiar 5-point discrete Laplacian, which is known to have very stable computational properties.

Currently, this algorithm is *not* available for use in MOM 3. It is being considered for use in MOM 4 upon switching to generalized orthogonal coordinates.

Appendix E

A note about computational modes

Fundamental to the discretization of MOM is the discretization of fluxes: advective fluxes, diffusive fluxes, etc. Working with fluxes provides for a useful way to preserve the internal consistency of the transport of momentum and tracers, which means, for example, that there should be no false sources or sinks assuming we have a sound numerical scheme. The convergence of fluxes is what is fundamentally of interest. Therefore, the fluxes must be defined on the boundaries of the relevant grid cell: diffusive fluxes at the boundary of the T-cell and advective fluxes at the boundary of the U-cell. To achieve this placement of fluxes often requires some creative discretization in the form of averaging operations (see Section 21.2 for more details of finite difference operators). When introducing average operators, however, it is important to be aware of the potential to introduce computational modes. A computational mode is basically a configuration of the discretized field which is invisible to the object which is being discretized. With nearest neighbor discretization on the respective T and U grid, which is done in MOM for second order accurate expressions, computational modes take the form of "2-delta X" type waves; i.e., the smallest resolvable lattice wave. Higher order schemes are exposed to computational modes of longer wavelength. The presence of these waves often signal the ability for "grid noise" to manifest in the solution and so should be avoided.

As an example of the what is described above, consider one part of a recent study of isoneutral diffusion in the GFDL model. In Griffies et al., (1998) (see also Appendix C), it was found that one source of grid noise in MOM is the original Cox (1987) discretization of the isoneutral diffusive flux. For the small angle approximated diffusion tensor, the Cox (1987) discretization of the flux in a two-dimensional x-z model is

$$-F^x_{i,k} = A_I \left[\delta_x T_{i,k} - \left(\frac{\delta_x \rho_{i,k}}{\delta_z \overline{\rho}^{x,z}_{i,k-1}} \right) \delta_z \overline{T}^{x,z}_{i,k-1} \right], \tag{E.1}$$

$$-F^z_{i,k} = A_I \left(\frac{\delta_x \overline{\rho}^{x,z}_{i-1,k}}{\delta_z \rho_{i,k}} \right)^2 \left[\delta_z T_{i,k} - \left(\frac{\delta_z \rho_{i,k}}{\delta_x \overline{\rho}^{x,z}_{i-1,k}} \right) \delta_x \overline{T}^{x,z}_{i-1,k} \right]. \tag{E.2}$$

In order to define the diffusive fluxes consistently on the B-grid, the x-flux must be placed at the east face of the T-cell (i,k) and the z-flux at the bottom of this cell. With this placement, the divergence of these fluxes across the T-cell results in a diffusion operator $R(T)_{i,k} = -(\delta_x F^x_{i-1,k} + \delta_z F^z_{i,k-1})$ properly placed at the center of the cell at the location of the tracer $T_{i,k}$. For the off-diagonal terms (the second terms of the fluxes), a spatial averaging in the form of a double average $\overline{(\,)}^{x,z}$ brings the z-derivative terms appearing in the x-flux onto the east face of a T-cell and the x-derivative terms appearing in the z-flux onto the bottom face of the T-cell. This is a

natural choice for averaging when working on the B-grid and provides an example of what is meant in the previous paragraph about "creative discretization".

There is a problem, however, with this discretization due to the presence of both the average and derivative operations acting in the same spatial direction on a single field. The problem is that this combination of operations introduces computational modes. For example, in the x-flux, the z-derivative of the tracer defined on the east face of T-cell (i,k) is

$$\delta_z \overline{T}^{x,z}_{i,k-1} = \frac{T_{i,k-1} - T_{i,k+1} + T_{i+1,k-1} - T_{i+1,k+1}}{4dzt_k}, \tag{E.3}$$

and likewise for the z-derivative of the density. A quick inspection of this forumla indicates that by taking both a z-average and a z-derivative allows for the presence of $2\Delta z$ computational modes $T_{i,k-1} = T_{i,k+1}$ and $\rho_{i,k-1} = \rho_{i,k+1}$. For fields containing this structure, the discretized z-derivative on the east face will vanish. For the z-flux, $2\Delta x$ computational modes exist due to the combination of the x-average and x-derivative. In general, when working on the B-grid and acting on a single field, such combinations of an average in one direction combined with a derivative in the same direction introduces computational modes in this field. The presence of the grid waves, and the ability to increase their amplitude, were two of the fundamental reasons that the Cox (1987) diffusion scheme was unstable and so required background horizontal diffusion. Consult Griffies et al., (1998) for complete details.

Appendix F

References

Adcroft, A. J., 1995: Numerical algorithms for use in a dynamical model of the ocean. Ph.D Thesis, University of London.

Adcroft, A. ,C. Hill, and J. Marshall, 1996: Representation of Topography by Shaved Cells in a Height Coordinate Ocean Model. *Submitted to Monthly Weather Review*

Apel, J. R., 1987: **Principles of ocean physics**, International Geophysics series, No. 38. Academic Press, London. 634 pages.

Arakawa, A., 1966: Computational design for long-term numerical integration of the equations of fluid flow: Two-dimensional incompressible flow. Part I. *Journal of Computational Physics*, **1**, 119–143.

Arakawa, A. and V. R. Lamb, 1977: Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics*, **17**, 174–265.

Aris, R. , 1962: **Vectors, Tensors and the basic equations of Fluid Mechanics**, Dover publishing.

Batchelor, G. K., 1967: **An introduction to fluid dynamics**, Cambridge University Press. 615 pages.

Beckers, J. M. , H. Burchard, J.M. Campin, E. Deleersnijder, and P.P. Mathieu, 1998: Comments on the paper: "Isoneutral diffusion in a z-coordinate ocean model by Griffies *et al.*, *Journal of Physical Oceanography*, accepted.

Bleck, R., C. Rooth, D. Hu, and L. T. Smith, 1992: Salinity-driven thermocline transients in a wind and thermohaline forced isopycnic coordinate model of the North Atlantic. *Journal of Physical Oceanography*, **22**, 1486-1505.

Blumberg, A. F., and G. L. Mellor, 1987: A description of a three-dimensional coastal ocean circulation model. *Three-Dimensional Coastal Ocean Models*. Vol. 4, N. Heaps, Ed., American Geophysical Union. 208 pp.

Böning, C., 1988: Particle disperson and mixing of conservative properties in an eddy-resolving model. *Journal of Physical Oceanography*, **18**, 320–338.

Boussinesq, J., 1903: *Théorie analytique de la chaleur.* Paris: Gathier-Villars, Vol. 2.

Boris, J. P. , D. L. Book, 1973: Flux-corrected transport, I. SHASTA: A fluid transport algorithm that works. *Journal of Computational Physics*, **11**, 38-69.

Brown, J. A. ,K. A. Campana, 1978: An economical time-differencing system for numerical weather prediction. *Monthly Weather Review*, **106**, 1125–1136.

Bryan, F. 1986: Maintenance and variability of the thermohaline circulation. Geophysical Fluid Dynamics Program Thesis, Princeton University.

Bryan, F., 1986: Parameter sensitivity of primitive equation ocean general circulation models. *Journal of Physical Oceanography*, **17**, 970–985.

Bryan, K., 1969: A numerical method for the study of the circulation of the world ocean. *Journal of Computational Physics*, **4**, 347–376.

Bryan, K. and M. D. Cox, 1972: An approximate equation of state for numerical models of the ocean circulation. *Journal of Physical Oceanography*, **2**, 510–514.

Bryan, K., S. Manabe, and R.C. Pacanowski, 1975: A global ocean-atmosphere climate model. Part II. The oceanic circulation. *Journal of Physical Oceanography*, **5**, 30–46.

Bryan, K. and L. J. Lewis, 1979: A water mass model of the world ocean. *Journal of Geophysical Research*, *84*, 2503–2517.

Bryan, K., 1984: Accelerating the convergence to equilibrium of ocean-climate models. *Journal of Physical Oceanography*, **14**, 666–673.

Bryan, K. and J. L. Sarmiento, 1985: Modeling ocean circulation. *Advances in Geophysics*, **28a**, 433–459.

Bryan, K., 1989: The Design of Numerical Models of the Ocean Circulation. *Oceanic Circulation Models: Combining Data and Dynamics, D.L.T. Anderson and J. Willebrand (eds.)*, Kluwer Academic Publishers, 465–500.

Bryan, K., 1991: Michael Cox (1941–1989): His pioneering contributions to ocean circulation modeling. *Journal of Physical Oceanography*, **21**, 1259–1270.

Bryan, K., R. D. Smith, and J. Dukowicz, 1998: A note on the bolus velocity field in an eddy resolving global ocean model. In preparation.

Chandrasekhar, S., 1961: **Hydrodynamic and hydromagnetic stability**, Dover Publications, New York. 652 pages.

Courant and Hilbert, *Methods of Mathematical Physics*, 2 volumes. Wiley-Interscience.

Cox, M. D., 1985: An eddy resolving numerical model of the ventilated thermocline. *Journal of Physical Oceanography*, **15**, 1312–1324.

Cox, M. D. and K. Bryan, 1984: A numerical model of the ventilated thermocline. *Journal of Physical Oceanography*, **14**, 674–687.

Cox, M. D., 1984: A primitive equation, 3-dimensional model of the ocean. GFDL Ocean Group Technical Report No. 1.

Cox, M. D., 1987: Isopycnal diffusion in a z-coordinate ocean model. *Ocean modeling*, **74**, 1–5.

Danabasoglu, G. and J. C. McWilliams, 1995: Sensitivity of the global ocean circulation to parameterizations of mesoscale tracer transports. Journal of Climate, **8**, 2967–2987.

Deardorff, J. W., 1973: The use of subgrid scale transport equations in a three-dimensional model of atmospheric turbulence. *Journal of Fluid Eng.*, **Sep.**, 429-438.

Dewar, W. K., Y. Hsueh, T. J. McDougall, and D. Yuan, 1998: Calculation of Pressure in Ocean Simulations. *Journal of Physical Oceanography*, **28**, 577–588.

Doescher, R. and R. Redler, 1997: The Relative Influence of North Atlantic Overflow and Subpolar Deep Convection on the Thermohaline Circulation in an OGCM, *JPO*, **27**, 1894–1902

DYNAMO groups at IMG Grenoble, JRC Southampton and IfM Kiel: 1994, 1994 scientific report, *EC MAST DYNAMO contract no. MAS2-CT93-0060*, .

Dukowicz, J.K. and R.D. Smith, 1997: Stochastic theory of compressible turbulent fluid transport. *Physics of Fluids*, **9**, 3523-3529.

Dukowicz, J. K. and R. D. Smith, 1994: Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *Journal of Geophysical Research*, **99**, 7991–8014.

Dukowicz, J. K., R. D. Smith, and R. C. Malone, 1993: A reformulation and implementation of the Bryan-Cox-Semtner ocean model on the Connection Machine. *Journal of Atmospheric and Oceanic Technology*, **10**, 195–208.

Farrow, D. E. , D. P. Stevens, 1995: A new tracer advection scheme for Bryan and Cox type ocean general circulation models. *Journal of Physical Oceanography*, **25**, 1731-1741.

Fetter, A. L., and Walecka, J. D., 1980: **Theoretical Mechanics of Particles and Continua**, McGraw-Hill Book Company. New York.

Fiadeiro M. E. , G. Veronis, 1977: On weighted-mean schemes for the finite-difference approximation to the advection-diffusion equation. *Tellus*, **29**, 512-522.

Gent, P. R., and J. C. McWilliam, 1996: Eliassen-Palm fluxes and the momentum equation in non-eddy-resolving ocean circulation models. *Journal of Physical Oceanography*, **26**, 2539–2546.

Gent, P. R., and J. C. McWilliams, 1990: Isopycnal mixing in ocean circulation models. *Journal of Physical Oceanography*, **20**, 150–155.

Gent, P. R., J. Willebrand, T. McDougall, and J. C. McWilliams, 1995: Parameterizing eddy-induced tracer transports in ocean circulation models. *Journal of Physical Oceanography*, **25**, 463–474.

Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dynamics*, **5**, 211–226.

Gerdes, R. 1993: A primitive equation ocean circulation model using a general vertical coordinate transformation 1. Description and testing of the model. *Journal of Geophysical Research*, **98**, 14683–14701.

Gill, A. E., 1982: **Atmosphere-Ocean Dynamics**. *Academic Press Inc*.

Goddard, L. 1995: The energetics of interannual variability in the tropical Pacific Ocean. Program in Atmospheric and Oceanic Sciences Thesis, Princeton University.

Goloviznin, V. M., Samarskii, A. A., and A. P. Favorskii, 1977: A variational approach to constructing finite-difference mathematical models in hydrodynamics. *Sov. Phys. Dokl.*, **22** 432–434.

Greatbatch, R. J., 1994: A note on the representation of steric sea level in models that conserve volume rather than mass. *Journal of Geophysical Research*. **99**, 12767–12771.

Griffies, S. M., 1998: The Gent-McWilliams Skew-Flux. *Journal of Physical Oceanography*, **28**, 831-841.

Griffies, S. M., A. Gnanadesikan, R. C. Pacanowski, V. Larichev, J. K. Dukowicz, and R. D. Smith, 1998: Isoneutral diffusion in a z-coordinate ocean model. *Journal of Physical Oceanography*, **28**, 805–830.

Griffies, S. M., R. C. Pacanowski, and R. W. Hallberg, 1998: Spurious mixing associated with advection in a z-coordinate ocean model. *Monthly Weather Review* in press.

Griffies, S.M. and R.W. Hallberg, 1999: Biharmonic friction with a Smagorinsky viscosity for use in large-scale eddy-permitting ocean models. Accepted by *Monthly Weather Review*.

Griffies, S.M., R. C. Pacanowski, M. Schmidt, and V. Balaji, 2000: The explicit free surface method in the GFDL modular ocean model. Submitted to *Monthly Weather Review*.

Haidvogel, D. B., J. L. Wilkin, and R. E. Young, 1991: A semi-spectral primitive equation ocean circulation model using vertical sigma and orthogonal curvilinear horizontal coordinates. *Journal of Computational Physics*, **94**, 151–185.

Hallberg, R., 1995: Some aspects of the circulation in ocean basins with isopycnals intersecting the sloping boundaries, Ph.D thesis, University of Washington, Seattle, 244 pp.

Haltiner, G. J. and R. T. Williams, 1980: **Numerical Prediction and Dynamic Meteorology**, Wiley.

Hamming, R. W., 1977: **Digital Filters**, Prentice-Hall.

Haney, R. L., 1971: Surface thermal boundary condition for ocean circulation models. *Journal of Physical Oceanography*, **1**, 241–248.

Hankin, S. and M. Denham: 1996, **FERRET: An Analysis Tool for Gridded Data, Users Guide, Version 4.4**, NOAA/PMEL/TMAP, 243 pages.

Hanson, R. J. , and C. L. Lawson, 1969: Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. Comput.*, **23** 787-812.

Held, I. M. and V. D. Larichev, 1996: A scaling theory for horizontally homogeneous baroclinically unstable flow on a beta plane, *Journal of Atmospheric Sciences*, **53**, 946–952.

Hellerman, S. and M. Rosenstein, 1983: Normal Monthly Stress over the World Ocean with Error Estimates. *Journal of Physical Oceanography*, **13**, 1093–1104.

Higdon, R. L. and A. F. Bennett, 1996: Stability analysis of operator splitting for large-scale ocean modeling. *Journal of Computational Physics*, **123**, 311-329.

Hirst, A. C. and T. McDougall, 1996: Deep-water properties and surface buoyancy flux as simulated by a z-coordinate model including eddy-flux advection. *Journal of Physical Oceanography*, **26**, 1320–1343.

Hirst, A. C., D. R. Jackett, and T. J. McDougall, 1996: The meridional overturning cells of a world ocean model in neutral density coordinates. *Journal of Physical Oceanography*, **26**, 775–791.

Holland, W. R., 1975: Energetics of baroclinic oceans. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

Holland, W. R., J. C. Chow, and F. O. Bryan, 1998: Application of a third-order upwind scheme in the NCAR ocean model. *Journal of Climate*, **11**, 1487–1493.

Holloway, Greg, 1992: Representing Topographic Stress for Large-Scale Ocean Models. *Journal of Physical Oceanography*, **22**, 1033–1046.

Holloway, G., 1989: Subgridscale representation. In D. L. T. Anderson and J. Willebrand (eds.), **Oceanic circulation models: combining data and dynamics**. pages 513–593. Kluwer Academic Publishers.

Holton, J. R., 1992: **An introduction to dynamic meteorology**, 3rd Edition. Academic Press. 507 pages.

Huang, 1993: Real freshwater flux as a natural boundary condition for the salinity balance and thermohaline circulation forced by evaporation and precipitation. *Journal of Physical Oceanography*, **23**, 2428–2446.

Jackett, D., R., and T. J. McDougall, 1997: A neutral density variable for the world's oceans. *Journal of Physical Oceanography*, **27**, 237–263.

Jerlov 1968: **Optical oceanography**. *Elsevier*.

Killworth, P. D., 1989: On the parameterisation of deep convection in ocean models. In *Paramterizing small scale processes in the ocean*, Proceedings of the Hawaiian Winter Workshop, 'Aha Huiliko'a.

Killworth, P. D., J. M. Smith, A. E. Gill, 1984: Speeding up ocean ciruclation models. *Ocean Modeling*, **56**, 1–5.

Killworth, P. D., 1987: Topographic instabilities in level model OGCM's. *Ocean Modeling*, **75**, 9–12.

Killworth, P. D., D. Stainforth, D. J. Webb and S. M. Paterson, 1989:
A free surface Bryan-Cox-Semtner model,
Institute of Oceanographic Sciences, Deacon Laboratory,
Report No. 270.

Killworth, P. D., 1997: On the parameterization of eddy transfer. Part I: theory. *Journal of Marine Research*, **55**, 1171–1197.

Killworth, P. D., D. Stainforth, D. J. Webb, and S. M. Paterson, 1991: The development of a free-surface Bryan-Cox-Semtner ocean model. *Journal of Physical Oceanography*, **21**, 1333–1348.

Klinger, B. A., J. Marshall, and U. Send, 1996: Representation of convective plumes by vertical adjustment. *Journal of geophysical research*, **101**, 18175–18182.

Korshiya, T. K., Tishkin, V. F., Favorskii, A. P., and M. Y. Shashkov, 1980: Flow-variational difference schemes for calculating the diffusion of a magnetic field. *Sov. Phys. Dokl.*, **25** 832–834.

Lamport, L., 1986: **LaTeXUser's Guide and Reference Manual**, First edition. Addison-Wesley Publishing Co., 242 pages.

Lamport, L., 1994: **LaTeXUser's Guide and Reference Manual**, Second edition. Addison-Wesley Publishing Co., 272 pages.

Landau, L. D., Lifshitz, E. M., 1986: **Theory of Elasticity**. Course of Theoretical Physics, Volume 7. Pergamon Press, Oxford.

Landau, L. D., Lifshitz, E. M., 1987: **Fluid Mechanics**. Course of Theoretical Physics, Volume 6. Pergamon Press, Oxford.

Large, W. G. , J. C. McWilliams, and S. C. Doney, 1994: Oceanic vertical mixing: A review and a model with a nonlocal boundary layer parameterization. *Rev. of Geophys*, **32**, 363-403.

Larichev, V. D. and I. M. Held, 1995: Eddy amplitudes and fluxes in a homogeneous model of fully developed baroclinic instability. *Journal of Physical Oceanography*, **25**, 2285–2297.

Leith, C. E. 1996: Stochastic models of chaotic systems. *Physica D* **98**, 481–491. In **Nonlinear phenomena in ocean dynamics**, edited by D.D. Holm, R.C. Malone, and Len G. Margolin.

Leonard, B. P., 1979: A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, **19**, 59–98

Levitus, S., 1982: Climatological atlas of the world ocean. *NOAA Prof. Pap. 13, 173 pp.* U. S. Government Printing Office, Washington, D. C.

Love, A. E. H., 1944: **A treatise on the mathematical theory of elasticity**. Dover, New York, 1944; reprinted from the 4th edition published by Macmillian courtesy of Cambridge University Press 1927.

Marotzke, J., 1991: Influence of convective adjustment on the stability of the thermohaline circulation. *Journal of Physical Oceanography*, **21**, 903–907.

Marshall, J., C. Hill, L. Perelman, and A. Adcroft, 1997: Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *Journal of Geophysical Research*, **102**, 5733–5752.

McDougall, T. J., 1987: Neutral surfaces. *Journal of Physical Oceanography*, **17**, 1950–1964.

McDougall, T. J., 1998: Residual-mean theory and its implications for parameterizing the eddy-induced component of the total tracer transport. In **Ocean modeling and parameterization**, edited by Eric P. Chassignet and Jacques Verron. NATO Advanced Study Institute.

McDougall, T. R., and D. R. Jackett, 1998: Potential enthalpy and modified potential temperature: conservative oceanic variables. For submission to *Journal of Physical Oceanography*.

McDougall, T. J., 1991: Parameterizing mixing in inverse models, in *Dynamics of Oceanic Internal Gravity Waves*, edited by P. Müller and D. Henderson. Proceeding of the sixth 'Aha Huliko 'a Hawaiian Winter Workshop, University of Hawaii at Manoa, 355–386.

Mellor, G. L., and T. Ezer, 1995: Sea level variations induced by heating and cooling: an evaluation of the Boussinesq approximation in ocean models. *Journal of Geophysical Research*. **100**, 20565–20577.

Middleton, J.F. and J.W. Loder, 1989: Skew fluxes in polarized wave fields. *Journal of Physical Oceanography*, **19**, 68–76.

Mitchell, A. R., and D. F. Griffiths, 1980: **The finite difference method in partial differential equations**, Wiley-Interscience Publications.

Molenkamp, C. R., 1968: Accuracy of finite-difference methods applied to the advection equation. *Journal of Applied Meteorology*, **7**, 160–167.

Müller, P., 1995: Ertel's potential vorticity theorem in physical oceanography. *Reviews if Geophysics*, **33**, 67–97.

Munk, W. H., 1950: On the wind-driven ocean circulation. *Journal of Meteorology*, **7**, 3–29.

NCAR, 1996: The NCAR CSM Ocean Model. *NCAR Technical Note NCAR/TN 423+STR* Climate and Clobal Dynamics Division, National Center for Atmospheric Research, Boulder, Colorado

Oberhuber, J. M., 1988: An atlas based on the "COADS" data set: The budgets of heat, buoyancy and turbulent kinetic energy at the surface of the global ocean. Technical Report No. 15, Max-Plack-Institut für Meteorologie, Hamburg, Germany.

Oort, A., 1983: Global atmospheric circulation statistics 1958-1973. *NOAA Prof. Pap. 14, 180 pp.* U. S. Government Printing Office, Washington, D. C.

Orlanski, I.: 1976, A simple boundary condition for unbounded hyperbolic flows, *J. Comp. Phys* **21**, 251–269.

Pacanowski, R. C., and A. Gnanadesikan: 1998, Transient response in a z-level ocean model that resolves topography with partial-cells. *Monthly Weather Review* **126**, No. 12, 3248-3270.

Pacanowski, R. C., and G. Philander, 1981: Parametrization of vertical mixing in numerical models of the tropical ocean. *Journal of Physical Oceanography*, **11**, 1442–1451.

Pacanowski, R. C., 1987: Effect of Equatorial Currents on Surface Stress, *Journal of Physical Oceanography*, **17**, No. 6, 833-838.

Pacanowski, R. C., K. Dixon, and A. Rosati, 1991: The GFDL modular ocean model user guide, The GFDL Ocean Group Technical Report No. 2., Geophysical Fluid Dynamics Laboratory, Princeton, USA., 16 pages.

Paulson and Simpson, 1977: Irradiance measurements in the upper ocean. *Journal of Physical Oceanography* **7**, 952-956.

Pierrehumbert, R. T. and H. Yang, 1993: Global chaotic mixing on isentropic surfaces. *Journal of Atmospheric Science*, **50**, No. 15, 2462-2480.

Pinardi, N., A. Rosati, and R. C. Pacanowski, 1995: The sea surface pressure formulation of rigid lid models. Implications for altimetric data assimilation studies. *Journal of Marine Systems*, **6**, 109–119.

Philander, S. G. H. and R. C. Pacanowski, 1986: A model of the Seasonal Cycle in the Tropical Atlantic Ocean. *Journal of Geophysical Research*, **91**, 14192-14206.

Phillips, N. A., 1959: An example of a non-linear computational instability. **The Atmosphere and the Sea in Motion**, Rossby Memorial Volume. Rockefeller Institute Press, pages 501-504.

Plumb, R. A. and J. D. Mahlman, 1987: The zonally averaged transport characteristics of the GFDL general circulation/transport model. *Journal of the Atmospheric Sciences*, **44**, 298–327.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, 1992: Numerical Recipes in Fortran, Second Edition, **Cambridge University Press**

Redi, M. H., 1982: Oceanic isopycnal mixing by coordinate rotation. *Journal of Physical Oceanography*, **12**, 1154–1158.

Redler, R. and C. W. Böning, 1997: Effect of the overflows on the circulation in the subpolar North Atlantic: A regional model study, *JGR*, **102**, 18529–18552

Rhines, P. B., 1986: Lectures on ocean circulation dynamics. In **Large-scale transport processes in oceans and atmospheres**, edited by J. Willebrand and D. L. T. Anderson. D. Reidel Publishing.

Roberts, M., and D. Marshall, 1998: Do we require adiabatic dissipation schemes in eddy-resolving ocean models? Accepted by *Journal of Physical Oceanography*.

Rood, R. B. , 1987: Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Rev. Geophys*, **25**, 71-100.

Rosati, A. and K. Miyakoda, 1988: A general circulation model for upper ocean simulation. *Journal of Physical Oceanography*, **18**, 1601–1626.

Segel, L. A., 1987: **Mathematics applied to continuum mechanics**. Dover Publications, Inc., New York.

Semtner, Jr., A. J., 1974: An oceanic general circulation model with bottom topography. In *Numerical Simulation of Weather and Climate*, Technical Report No. 9, UCLA Department of Meteorology.

Semtner, A. J. and Y. Mintz, 1977: Numerical simulation of the Gulf Stream and mid-ocean eddies. *Journal of Physical Oceanography*. **7**, 208–230.

Semtner, Jr., A. J., 1986a: Finite-difference formulation of a World Ocean model. In *Advanced Physical Oceanographic Numerical Modeling*, J. J. O'Brien, Ed., D. Reidel Publishing Company, 187–202.

Semtner, Jr., A. J., 1986a: History and methodology of modeling the circulation of the World Ocean. In *Advanced Physical Oceanographic Numerical Modeling*, J. J. O'Brien, Ed., D. Reidel Publishing Company, 23–32.

Smagorinsky, J. 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, **91**, 99–164.

Smagorinsky, J. 1993: Some historical remarks on the use of nonlinear viscosities, in **Large eddy simulation of complex engineering and geophysical flows**, edited by B. Galperin and S. A. Orszag. Cambridge University Press.

Smith, R. D., J. K. Dukowicz, and R. C. Malone, 1992: Parallel ocean general circulation modeling. *Physica D*, **60**, 38–61.

Smith, S. D., 1988: Coefficients for sea surface wind stress, heat flux, and wind profiles as a function of wind speed and temperature. *Journal of Geophysical Research*, **93**(C12), 15,467–15,472.

Smith, S. D., and F. W. Dobson, 1984: The heat budget at ocean weather station bravo. *Atmosphere-Ocean*, **22**, 1–22.

Spiegel, E. A., and G. Veronis, 1960: On the Boussinesq approximation for a compressible fluid. *The Astrophysical Journal*, **131**, 442–447.

Stammer, D., 1998: On eddy characteristics, eddy transport, and mean flow properties. *Journal of Physical Oceanography* **28**, 727–739.

Stevens, D. P. 1991: The open boundary condition in the United Kingdom Fine– Resolution Antarctic Model, *Journal of Physical Oceanography* **21**, 1494–1499.

Stevens, D. P. 1990: On open boundary conditions for three dimensional primitive equation ocean circulation models, *Geophysical and Astrophysical Fluid Dynamics* **51**, 103–133.

Stommel, H., 1961: Thermohaline convection with two stable regimes of flow. *Tellus*, **13**, 224–230.

Synge, J. L., and A. Schild, 1949: **Tensor Calculus**. University of Toronto Press, Toronto.

Takacs, L. L., and R. Balgovind, 1983: High latitude filtering in global grid point models. *Monthly Weather Review*, **111**, 2005–2015.

Tishkin, V. F., Favorskii, A. P., and M. Y. Shashkov, 1979: Variational-difference schemes for the heat-conduction equation on nonregular grids. *Sov. Phys. Dokl.*, **24** 446–448.

Treguier, A. M., 1992: Kinetic energy analysis of an eddy resolving, primitive equation model of the North Atlantic. *Journal of Geophysical Research*, **97**, 687–701.

Treguier, A. M., J. K. Dukowicz, and K. Bryan. 1996: Properties of nonuniform grids used in ocean general circulation models. *Journal of Geophysical Research* **101(C9)** 20877-20881.

Treguier, A. M., I. M. Held, and V. D. Larichev, 1997: On the parameterization of quasi-geostrophic eddies in primitive equation ocean models. *Journal of Physical Oceanography*, **27**, 567–580.

Troen, I. B., and L. Mahrt, 1986: A simple model of the atmospheric boundary layer; sensitivity to surface evaporation. *Boundary-Layer Meteor.*, **37**, 129–148.

Turner, J. 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, **91**, 99–164.

Turner, J. S., 1973: **Buoyancy effects in fluids**. *Cambridge University Press*.

Veronis, G., 1973: Large scale ocean circulation. *Advances in Applied Mechanics*, **13**, 2–92.

Veronis, G., 1975: The role of models in tracer studies. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

Visbeck, M., J. Marshall, T. Haine, and M. Spall, 1997: Specification of eddy transfer coefficients in coarse resolution ocean circulation models. *Journal of Physical Oceanography*, **27**, 381–402.

Weaver, A. J. and E. S. Sarachik 1990: On the importance of vertical resolution in certain ocean general circulation models. *Journal of Physical Oceanography*, **20**, 600-609.

Weaver, A. J. and E. S. Sarachik 1991: Evidence for decadal variability in an ocean general circulation model: and advective mechanism. *Atmos. Ocean*, **29**, 197–231.

Wajsowicz, R. C. , 1993: A consistent formulation of the anisotropic stress tensor for use in models of the large-scale ocean circulation. *Journal of Computational Physics*, **105**, 333-338.

Washington, W. M., and C. L. Parkinson, 1986: **An Introduction to Three-dimensional Climate Modeling**, University Science Books, Mill Valley, CA, USA. 422 pages.

Webb, D. J., 1995: The Vertical Advection of Momentum in Bryan-Cox-Semtner OCean General Circulation Models. *Journal of Physical Oceanography*, **25**, 3186–3195.

Weinberg, S., 1972: **Gravitation and Cosmology**, J. Wiley and Sons. 657 pages.

Williams, G. P., 1972: Friction term formulation and convective instability in a shallow atmosphere. *Journal of Atmospheric Sciences*, **29**, 870–876.

Wright, D. K., 1997: A new eddy mixing parametrization and ocean general circulation model. WOCE International Newsletter. Number 26, April, pp. 27-29.

Zalesak S. T. , 1979: Fully multidimensional flux-corrected transport algorithms for fluid. *Journal of Computational Physics*, **31**, 335-362.