# A parallel spectral atmospheric GCM

**V. Balaji**
**SGI/GFDL**

**CUG T3E Workshop**
**Princeton, September 1999**

# Atmospheric GCMs

Numerical models for the atmospheric general circulation (GCMs) solve discrete forms of the non-linear Navier-Stokes equations for fluid flow in a spherical shell, coupled with equations describing the thermodynamic state of air. The prevalent terminology divides these equations into the *dynamics* and *physics*, where the former refers to the resolved Navier-Stokes dynamics, with the parameterization of thermodynamics and unresolved scales being termed the physics.

# The GFDL climate model interface

The twin needs of scalability and modularity have prompted the design of a new generation of models at GFDL. These models are being designed with a maximum of flexibility in mind. Not only may these codes have to perform over a range of computing architectures, they will also be in constant flux, especially in regard to the physics packages with which they are used. With this in mind, we have isolated the *dynamical core* of these models as far as possible. The dynamical core accepts a single uniform module interface for physics. Thus, the physics packages remain entirely interchangeable. The dynamical cores themselves also retain a modular texture, as a wide variety of algorithms may be used for various aspects of the dynamics depending on the problem being solved and the available computing power on the platform upon which the calculations are being carried out.

The new generation models include a finite-difference gridpoint model and a spectral dynamical model. This paper is concerned with the modular design of a scalable dynamical core for the spectral model.

# The spectral dynamical core

The spectral dynamical core uses the transform method. In the transform method, linear terms (including semi-implicit treatment of gravity waves) are solved in spectral space. Non-linear terms and physics are done in grid space. Thus, all fields are transformed back and forth once per timestep. The efficiency of the transform package is key to the performance of the method.

$$f(\theta,\phi) = \sum_{m=-\infty}^{\infty} \sum_{n=|m|}^{\infty} f_{mn} P_{mn}(\cos\theta)\, e^{im\phi} \tag{1}$$

- $m$: Fourier wavenumber.
- $n$: spherical wavenumber.
- $P_{mn}$: associated Legendre polynomials.

$Y_{mn} \equiv P_{mn} e^{im\phi}$: spherical harmonics:

$$\frac{1}{4\pi} \int_{-\pi}^{\pi} \int_{\mathrm{SP}}^{\mathrm{NP}} Y_{mn} Y^*_{m'n'} d\left(\cos\theta\right) d\phi = \delta_{mm'} \delta_{nn'} \tag{2}$$

The spherical harmonic coefficients $f_{mn}$ of the expansion are given by the inverse transform:

$$f_{mn} = \frac{1}{4\pi} \int_{-\pi}^{\pi} \int_{\mathrm{SP}}^{\mathrm{NP}} f\left(\theta, \phi\right) P_{mn}^{*}\left(\cos\theta\right) e^{im\phi} d\left(\cos\theta\right) d\phi \qquad (3)$$

# Spectral truncation

Truncate the Fourier wavenumber $m$ to $M$.

- *Triangular truncation*: $N = M$.

- *Rhomboidal truncation*: $N = |m| + M$.

Thus T42 refers to a spectral expansion that is triangular-truncated at $M = 42$ and $N = 43$.

# Spatial discretization

- The $I$ longitude points $\phi_i$ are generally chosen to be evenly distributed between 0 and $2\pi$.

- The $J$ latitude points $\theta_j$ are chosen to lie at the *Gaussian quadrature points*. Note that the Gaussian grid is non-uniform in $j$.

The spectral transform requires $I \geq 3M + 1$ and $J = I/2$.

# Spectral transform

$$f_{ij} \equiv f\left(\theta_j, \phi_i\right) = \sum_{m=-M}^{M} \sum_{n} f_{mn} P_{mn}\left(\cos\theta_j\right) e^{-im\phi_i} \tag{4}$$

$$f_{mn} = \frac{1}{4\pi} \sum_{j=1}^{J} \sum_{i=1}^{I} f_{ij} e^{im\phi_i} P_{mn}\left(\cos\theta_j\right) \Delta\left(\cos\theta_j\right) \Delta\phi \tag{5}$$

The number of operations in the Legendre transform can be halved by exploiting the symmetry:

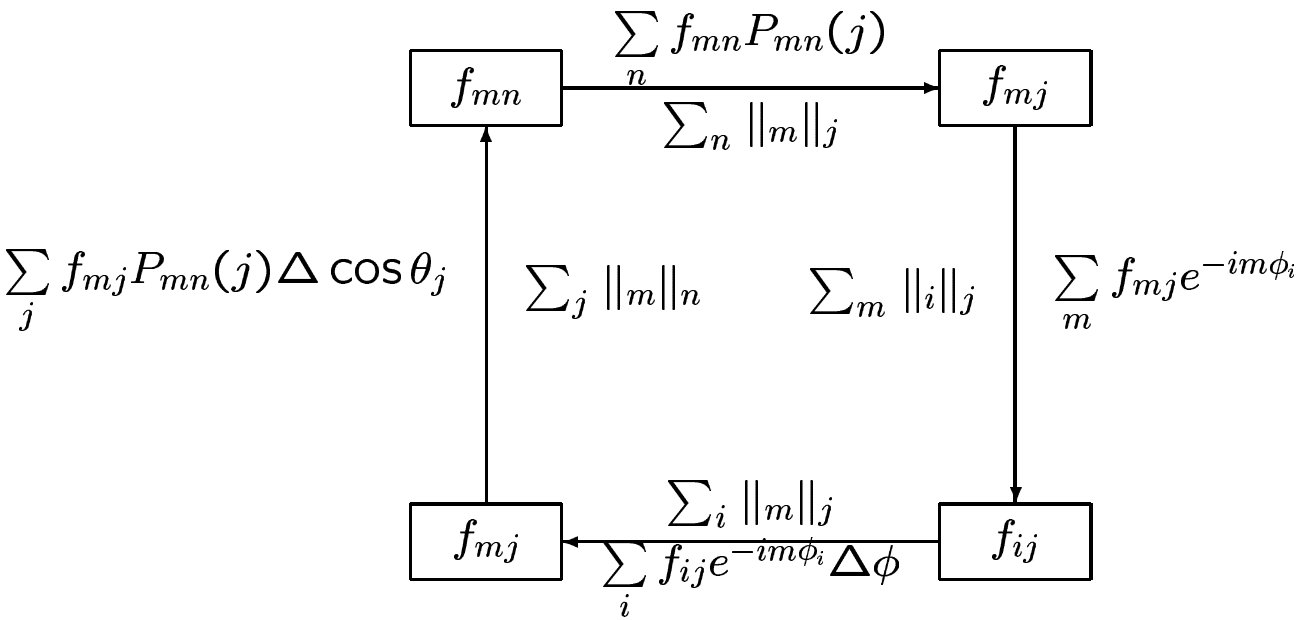
$$P_{mn}(-\lambda) = (-1)^n P_{mn}(\lambda) \tag{6}$$

# Order of summation

The sums are performed in the order shown in Eq. **??** and Eq. **??**. The intermediate step produces the partially transformed quantity $f_{mj}$, which we term the *Fourier representation*:

$$f_{mj} = \sum_n f_{mn} P_{mn}\left(\cos\theta_j\right) = \sum_{i=1}^{I} f_{ij} e^{im\phi_i} \tag{7}$$

# Data dependencies in the spectral transform



# Parallelization strategy

We attempt to keep the Legendre and Fourier transforms on-processor.

Thus spectral space is first parallelized along $m$, and parallelized along $n$ only if NPES is a multiple of $M$.

Grid space is parallelized along $j$, and along $i$ only if NPES is a multiple of $J/2$.

In the discussion below, we assume 1D decomposition, though the generalization to 2D decomposition is straightforward and involves a minimal use of reduction operations.

# Parallel programming interface

GFDL has a home-grown parallelism API written as a set of 3 F90 modules:

- `mpp_mod` is a low-level interface to message-passing APIs (currently SHMEM and MPI; MPI-2 and Co-Array Fortran to come);
- `mpp_domains_mod` is a set of higher-level routines for domain decomposition and domain updates;
- `mpp_io_mod` is a set of routines for parallel I/O.

`http://www.gfdl.gov/~vb`

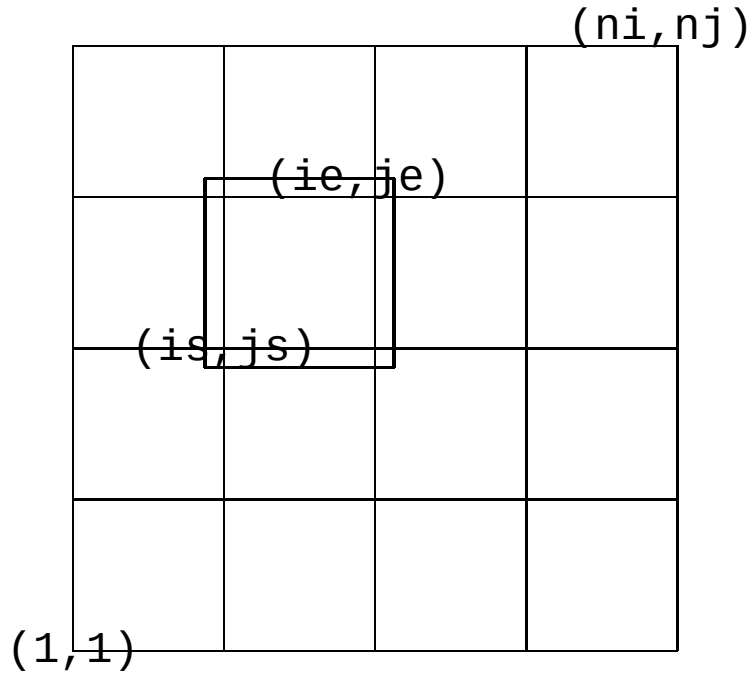
# mpp_domains_mod

Comprehensive domain decomposition information is held in a derived type *domaintype*. We define *domain* as the grid associated with a *task*. We define the *compute domain* as the set of gridpoints that are computed by a task, and the *data domain* as the set of points that are required by the task for the calculation. There can in general be more than 1 task per PE, though often the number of domains is the same as the processor count. We define the *global domain* as the global computational domain of the entire model (i.e, the same as the computational domain if run on a single processor).

2D domains are defined using a derived type `domain2D`, constructed as follows (see comments in code for more details):

```
type, public :: domain_axis_spec
   sequence
   integer :: start_index, end_index, size, max_size
   logical :: is_global
end type domain_axis_spec
type, public :: domain1D
   sequence
   type(domain_axis_spec) :: compute, data, global
   integer :: ndomains
   integer :: pe
   integer, dimension(:), pointer :: pelist
   type(domain1D), pointer :: prev, next
end type domain1D
```

```
!domaintypes of higher rank can be constructed from type domain1D
  type, public :: domain2D
     sequence
     type(domain1D) :: x
     type(domain1D) :: y
     integer :: pe
     type(domain2D), pointer :: west, east, south, north
  end type domain2D
```



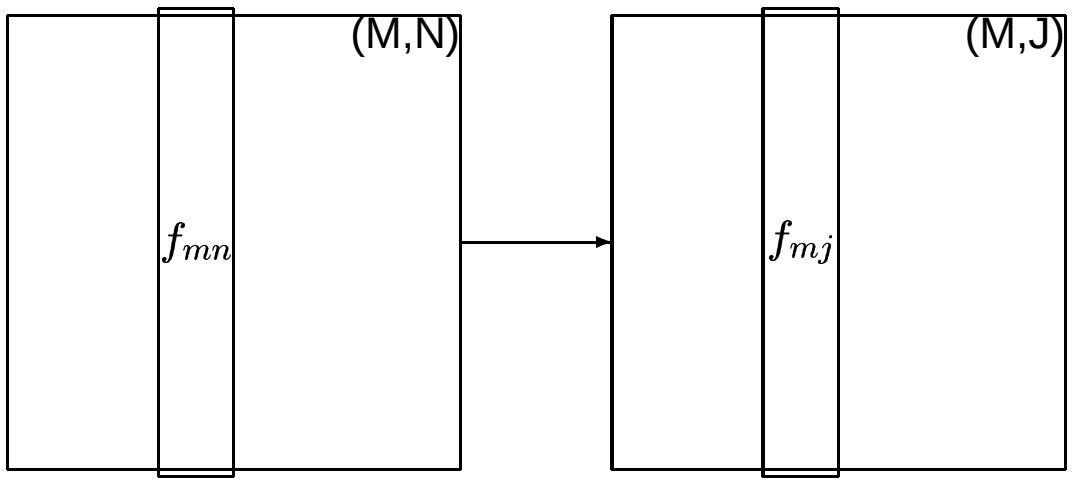
The `domain2D` type contains all the necessary information to define the global, compute and data domains of each task, as well as the PE associated with the task. The PEs from which remote data may be acquired to update the data domain are also contained in a linked list of neighbours.

# mpp_domains_mod calls:

- mpp_define_domains()
- mpp_update_domains()

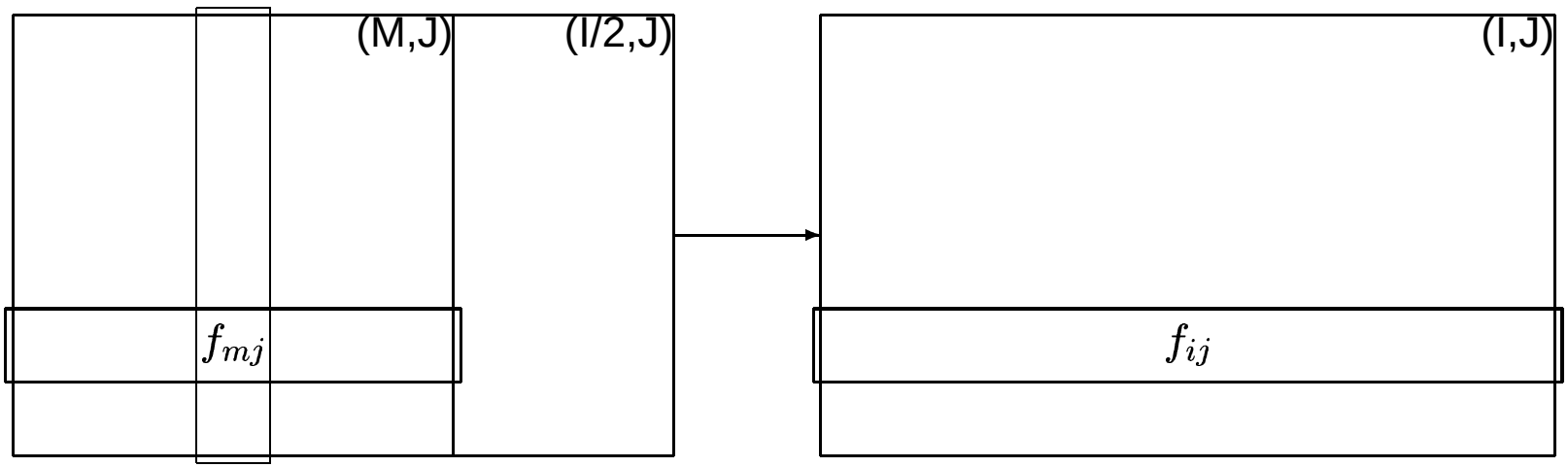
```
type(domain2D) :: domain(0:npes-1)
call mpp_define_domains( (/1,ni,1,nj/), domain, xhalo=2, yhalo=2 )
...
!allocate f(i,j) on data domain
!compute f(i,j) on compute domain
...
call mpp_update_domains( f, domain(pe) )
```

# Forward Legendre transform $f_{mn} \longrightarrow f_{mj}$




```
call trans_spherical_to_fourier( spherical, fourier )
call mpp_sum( fourier, domain%y%pelist )
```

# Forward Fourier transform $f_{mj} \longrightarrow f_{ij}$




```
call mpp_update_domains( fourier, domain, XUPDATE+TRANSPOSE )
call trans_fourier_to_grid( fourier, grid )
```

# Reverse operations

```
call trans_grid_to_fourier( grid, fourier )
call mpp_update_domains( fourier, domain, YUPDATE+TRANSPOSE )
call trans_fourier_to_spherical( fourier, spherical )
call mpp_sum( spherical, domain%y%pelist )
```

# Performance of the parallel spectral transform on the T3E

Coupled climate models used in century-scale climate change simulations are currently run at a resolution of R30. In the next five years, this will increase first to T42 and then T106. At GFDL, we shall be running both over the next 3–5 years.

The parallel spectral transform scales to 29 on 32 PEs at T42, and 75 out of 80 PEs at T106. This is on the Held-Suarez benchmark with simple relaxation physics. This is a lower limit on scalability, since the full physics load is considerably higher, and the physics itself is embarrassingly parallel, barring load balance issues.

# Conclusions

The parallel atmospheric spectral dynamical core is a highly scalable solver of hydrostatic atmospheric dynamics for climate studies.

- Coded in F90, transparent and easy to understand.
- Parallelism uses message-passing, and performs well.
- Could be made available as a pedagogic tool or for actual use.

# Future developments

- Generalization of the domaintype.

- Gridpoint advection (semi-Lagrangian).