Gridspec: A standard for the description of grids used in Earth System models

V. Balaji[*] Princeton University

Alistair Adcroft Princeton University

Zhi Liang RSIS Inc.

7 June 2007

Abstract

The comparative analysis of output from multiple models, and against observational data analysis archives, has become a key methodology in reducing uncertainty in climate projections, and in improving forecast skill of medium- and long-term forecasts. There is considerable momentum toward simplifying such analyses by applying comprehensive community-standard metadata to observational and model output data archives.

The representation of gridded data is a critical element in describing the contents of model output. We seek here to propose a standard for describing the grids on which such data are discretized. The standard is drafted specifically for inclusion within the Climate and Forecasting (CF) metadata conventions.

Contents

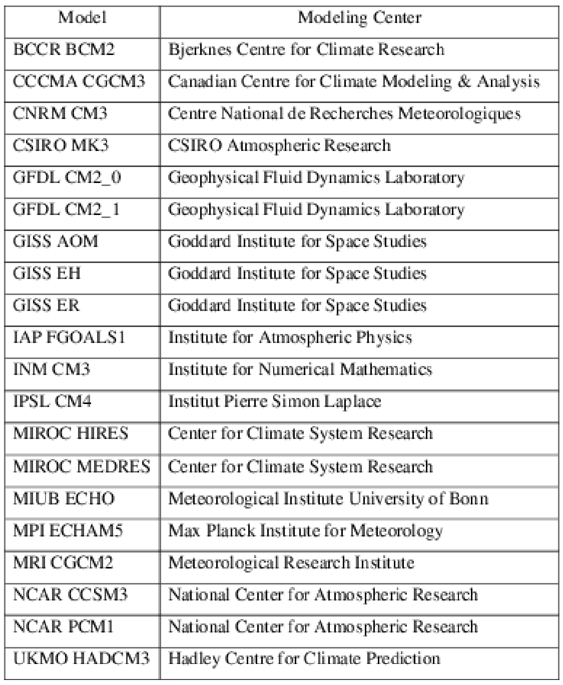
[*]Corresponding author: V. Balaji, Princeton University and NOAA/GFDL, 201 Forrestal Road, Princeton NJ 08540-6646. Email: `balaji@princeton.edu`

1. Introduction

1.1. *Methodology of international modeling campaigns*

The current decade (2000-2010) may be regarded as the decade of the coming-of-age of Earth System models. Such models are coming into routine use in both research and operational settings: for understanding the planetary climate in terms of feedbacks and balances between its many components; for translating such understanding into projections that inform policy to address anthropogenic climate change; and increasingly for medium- and long-term forecasts that require coupled models as well.

These activities manifest themselves in aspects of current scientific methodology. Earth System science is becoming “big science” where experiments systematically involve large international modeling campaigns, matching in scale the observational campaigns that are responsible for producing the climate record. A key example of such a modeling campaign is the activity surrounding the Inter-Governmental Panel on Climate Change (IPCC) Assessment Reports. These reports, issued every 6 years, are a culmination of systematic and coordinated modeling experiments run at multiple institutions around the world. Figure 1 shows a list of participating IPCC institutions from the recently concluded Fourth Assessment Report (IPCC AR4) **(missing ref:)**. A comparative study of results from multiple models run under the same external forcings remains our best tool for understanding the climate system, and for generating consensus and uncertainty estimates of climate change. Several key papers based on the IPCC AR4 data archive at PCMDI document recent leaps in understanding of aspects of the climate system in stable and warming climates, such as ENSO (Guilyardi 2006; van Oldenborgh et al. 2001), the tropical circulation (e.g Vecchi et al. 2006), Southern ocean circulation (Russell et al. 2006), and others **(missing ref:)**. Other similar campaigns underway include the Aqua-Planet Experiment (APE) **(missing ref:)**, the ENSEMBLES project (Hewitt and Griggs 2004) as well as several older ones.



Model	Modeling Center
BCCR BCM2	Bjerknes Centre for Climate Research
CCCMA CGCM3	Canadian Centre for Climate Modeling & Analysis
CNRM CM3	Centre National de Recherches Meteorologiques
CSIRO MK3	CSIRO Atmospheric Research
GFDL CM2_0	Geophysical Fluid Dynamics Laboratory
GFDL CM2_1	Geophysical Fluid Dynamics Laboratory
GISS AOM	Goddard Institute for Space Studies
GISS EH	Goddard Institute for Space Studies
GISS ER	Goddard Institute for Space Studies
IAP FGOALS1	Institute for Atmospheric Physics
INM CM3	Institute for Numerical Mathematics
IPSL CM4	Institut Pierre Simon Laplace
MIROC HIRES	Center for Climate System Research
MIROC MEDRES	Center for Climate System Research
MIUB ECHO	Meteorological Institute University of Bonn
MPI ECHAM5	Max Planck Institute for Meteorology
MRI CGCM2	Meteorological Research Institute
NCAR CCSM3	National Center for Atmospheric Research
NCAR PCM1	National Center for Atmospheric Research
UKMO HADCM3	Hadley Centre for Climate Prediction

Figure 1: Participating institutions in the IPCC AR4 series of experiments.

It has also become apparent that a similar molt-model ensemble approach is of utility in seasonal and interannual forecasting as well. An example of such a modeling campaign is the DEMETER project (Palmer et al. 2004). Studies (e.g Hagedorn et al. 2005) show that such operational ensemble forecasts have demonstrably better forecast skill than any individual ensemble member.

A third trend in current modeling studies is the increased use of *downscaling*, reviewed in Wilby and Wigley (1997). Where fine-scale simulation over some domain is sought, and it is either useless (because there is limited impact of fine-scale structure on larger scales) or impractical (for computational reasons) to extend the high resolution over the entire domain, one often creates model chains, where models over larger domains at coarser resolution are used to force finer-scale models nested within. The use of model chains is also a sort of multi-model study, where output data from one model serves as input to another. In all the approaches above, the need for data standards to enable ready access to data from diverse models is apparent.

1.2. *Community approaches to models and data*

As Earth System science increasingly comes to depend on models created from multiple components, and on comparative studies of output from such models, standardization has become a serious issue as we grapple with the practicalities of carrying out such studies. Emerging efforts at standardization of model component interfaces include the Earth System Modeling Framework (ESMF) (ESMF: Hill et al. 2004; Collins et al. 2005) and the PRISM project **`(missing ref: eric,sophie)`**.

Model output data in the Earth System Science community increasingly converges on the *netCDF format*[1], and, to a lesser degree, the *HDF5 format*[2]. In the weather forecasting domain, the WMO-mandated GRIB and BUFR formats **`(missing ref: )`** continue to be used. While the data formats themselves are relatively mature, recent efforts in this domain focus on developing consistent and comprehensive *metadata*, data descriptors that provide human- and machine-readable information about the data necessary in interpreting its contents. Metadata vocabularies are intended eventually to enable the inclusion of data into a *semantic web* (Berners-Lee 1999; Berners-Lee and Hendler 2001) which human and other reasoning agents will be able to use to make useful inferences about found entities. In the climate and weather modeling domain, efforts at developing a common vocabulary for metadata have converged on the Climate and Forecasting (CF) conventions. Similar initiatives for observational data (e.g the *Marine Metadata Initiative (MMI)*[3]) abound, and there are attempts underway to align the CF vocabularies with the observational ones. The *Open Geospatial Consortium (OGC)*[4] is a possible mechanism to shepherd the CF conventions toward a formal standard.

[1]`http://www.unidata.ucar.edu/software/netcdf`
[2]`http://hdf.ncsa.uiuc.edu/HDF5`
[3]`http://marinemetadata.org`
[4]`http://www.opengeospatial.org`

1.3. *Rationale for a grid standard*

This paper focuses on a key element of the metadata under development: the *grids* on which model data is discretized. Experience from the international modeling campaigns cited above in Section 1.1 indicates that there is a wide diversity in the model grids used; and further, it appears that this diversity is only increasing. However, in the absence of a standard representation of grids, it has been rather difficult to perform comparative analyses of data from disparate model grids. Rather, the lead institutions in these campaigns insist upon having data delivered on very simple grids, on the credible argument that the sites running the models are best placed to perform regridding operations of appropriate quality, meeting the relevant scientific criteria of conservation, and so on.

This approach was followed in the IPCC AR4 campaign, and while the resulting data archive was an extraordinary boon to data *consumers* (analysts of model output), the burden it placed on data *producers* (modeling centres) was considerable. Further, the issues surrounding regridding are common to most modeling centres, capable of being abstracted to common software. We believe a suite of common regridding methods and tools is now possible, given a grid standard.

The grid standard becomes even more necessary in considering the other sorts of uses outlined in Section 1.1, such as in model chains where gridded data from one model becomes input to another. And last but not least, multiple model grids and data transformations between them are intrinsic to modern Earth System models themselves, and are the basis for coupled model development from components developed across the entire community.

This paper proposes a *grid standard*: a convention for describing model grids. We have described so far its general features and purposes:

- the standard will describe the grids commonly used in Earth system models from global scale to fine scale, and also with an eye looking forward (toward emerging discrete representations) and sideways (to allied research domains: space weather, geosciences);
- the standard will contain all the information required to enable commonly performed scientific analysis and visualization of data;
- the standard will contain all the information required to perform transformations from one model grid to another, satisfying constraints of conservation and preservation of essential features, as science demands;
- the standard will make possible the development of shared regridding software, varying from tools deployable as web services to perform on-the-fly regridding from data archives, to routines to be used within coupled models. It will enable, but not mandate, the use of these standard techniques.

An outline of such a grid standard is the topic of this paper.

1.4. *Overview of paper*

The paper is structured as follows. In Section 2 we survey the types of grids currently in use, and potentially to be used in emerging models, that the standard must cover. This includes the issue of vector fields and staggered grids. In Section 2.7 we develop the key abstractions of mosaics, required for handling nested grids and other "non-standard" tilings of the sphere. In Section 2.9 we cover the issue of masks and exchange grids, required for transformations of data between grids. In Section 3 we develop a vocabulary for describing grids in the context of the CF conventions.

2. Grid terminology for Earth System science

We begin by developing a terminology for describing the types of grids used in Earth System science models and datasets. Grids for Earth System science can be considerably specialized with respect to the more general grids used in computational fluid dynamics. Specifically, the vertical extent is considerably smaller ($\sim$10 km) than the horizontal ($\sim$1000 km), and the fluid in general strongly stratified in the vertical. The treatment of the vertical is thus generally separable; and model grids can generally be described separately in terms of a horizontal 2D grid with coordinates X and Y, and a vertical coordinate Z.

2.1. *Geometry*

The underlying *geometry* being modeled is most often a thin spherical shell[5], especially when it is the actual planetary dynamics that is being modeled. However, more idealized studies may use geometries that simplify the rotational properties of the fluid, such as an f-plane or β-plane, or even simply a cartesian geometry.

Where the actual Earth or planetary system is being modeled, *geospatial mapping* or *geo-referencing* is used to map model coordinates to standard spatial coordinates, usually *geographic longitude* and *latitude*. Vertical mapping to pre-defined levels (e.g height, depth or pressure) is also often employed as a standardization technique when comparing model outputs to each other, or to observations.

2.2. *Vertical coordinate*

The vertical coordinate can be *space-based* (height or depth with respect to a reference surface) or *mass-based* (pressure, density, potential temperature). *Hybrid* coordinates with a mass-based element are considered to be mass-based.

The *reference surface* is a digital elevation map of the planetary surface. This can be a detailed topography or bathymetry digital elevation dataset, or a more idealized

[5]Except at very fine scales, the geometry is treated as a sphere, not a geoid. This may be a problem when geo-referencing to very precise datasets that consider the surface as a geoid.

one such as the representation of a single simplified mountain or ridge, or none at all. Vertical coordinates requiring a reference surface are referred to as *terrain-following*. Both space-based (e.g Gal-Chen **(missing ref:)**, ζ **(missing ref:)**) and mass-based (e.g σ) terrain-following coordinates are commonly used.

The rationale for developing this minimal taxonomy to classify vertical coordinates is that translating one class of vertical coordinate into another is generally model- and problem-specific, and should *not* be attempted by standard regridding software.

2.3. *Horizontal coordinates*

Horizontal spatial coordinates may be *polar* (θ,ϕ) coordinates on the sphere, or *planar* (x,y), where the underlying geometry is cartesian, or based on one of several *projections* of a sphere onto a plane. Planar coordinates based on a spherical projection define a *map factor* allowing a translation of (x,y) to (θ,ϕ).

Curvilinear coordinates may be used in both the polar and planar instances, where the model refers to a pseudo-longitude and latitude, that is then mapped to geographic longitude and latitude by geo-referencing. Examples include the displaced-pole grid (Jones et al. 2005) and the tripolar grid (Murray 1996).

Horizontal coordinates may have the important properties of *orthogonality* (when the Y coordinate is normal to the X) and *uniformity* (when grid lines in either direction are uniformly spaced). Numerically generated grids may not be able to satisfy both constraints simultaneously.

A third type of horizontal coordinate often used in this domain is not spatial, but spectral. *Spectral coordinates* on the sphere represent the horizontal distribution of a variable in terms of its spherical harmonic coefficients. These coefficients can be uniquely mapped back and forth to polar coordinates based on Fourier and Legendre transforms, yielding uniformly spaced longitudes, and latitudes defined by a Gaussian quadrature. This grid specification will not consider spectral representations directly; rather, it assumes that the data have been transformed to polar coordinates, and only seeks to encode the *truncation* used to restrict the representation to a finite set of values.

Spectral coordinates on the plane have also recently been used in this domain. These methods generally employ *spectral elements* (Thomas and Loft 2002; Iskandarani et al. 2002) projecting the sphere onto a series of planes of finite spatial extent, within each of which the representation is spectral. Spectral elements are also uniquely bound to geospatial coordinates by a series of transforms, and it is in these coordinates that the data are assumed to have been written.

2.4. *Time coordinate*

As for the fourth coordinate, time, it is already reasonably well-covered in the CF conventions. Both instantaneous and time-averaged values are represented. Key issues that still remain include the definition and treatment of non-standard *calendars*, and for simulation data, a standard vocabulary to define aspects of a running experiment, such as the absolute start time of the simulation.

2.5. *Discretization*

In translating a data variable to a discrete representation, we must decide what aspects are necessary for inclusion in a standard grid specification. We have chosen two classes of operations that the grid standard must enable: *vector calculus*, differential and integral operations on scalar and vector fields; and *conservative regridding*, the transformation of a variable from one grid to another in a manner that preserves chosen moments of its distribution, such as area and volume integrals of 2D and 3D scalar fields. We recognize that higher-order methods that preserve variances or gradients may entail some loss of accuracy. In the case of vector fields, grid transformations that preserve streamlines are required.

To enable vector calculus and conservative regridding, the following aspects of a grid must be included in the specification:

- *distances* between gridpoints, to allow differential operations;
- *angles* of grid lines with respect to a reference, usually geographic East and North, to enable vector operations. One may also choose to include an *arc type* (e.g "great circle"), which specifies families of curves to follow while integrating a grid line along a surface.
- *areas* and *volumes* for integral operations. This is generally done by defining the boundaries of a grid cell represented by a point value. In Section 2.9 below we will also consider fractional areas and volumes in the presence of a *mask*, which defines the sharing of cell between two or more components.

A taxonomy of grids may now be defined. A discretization is *logically rectangular* if the coordinate space (x, y, z) is translated one-to-one to index space **`(i,j,k)`**. Note that the coordinate space may continue to be physically curvilinear; yet, in index space, grid cells will be rectilinear boxes.

The most commonly used discretization in Earth system science is logically rectangular, and that will remain the principal object of study here. Beyond the simplest logically rectangular grids may include more specialized grids such as the tripolar grid of Murray (1996) shown in Figure 2 and the cubed-sphere grid of Rancic et al. (1996), shown in Figure 3.

Triangular discretizations are increasingly voguish in the field. A *structured triangular* discretization of an icosahedral projection is a popular new approach resulting in a geodesic grid (Majewski et al. 2002; Randall et al. 2002). An example of a structured triangular grid is shown in Figure 4 from Majewski et al. (2002). The grid is generated by recursive division of the 20 triangular faces of an icosahedron.

Numerically generated *unstructured triangular* discretization, such as shown in Figure 5 are often used, especially over complex terrain. High resolution models in-

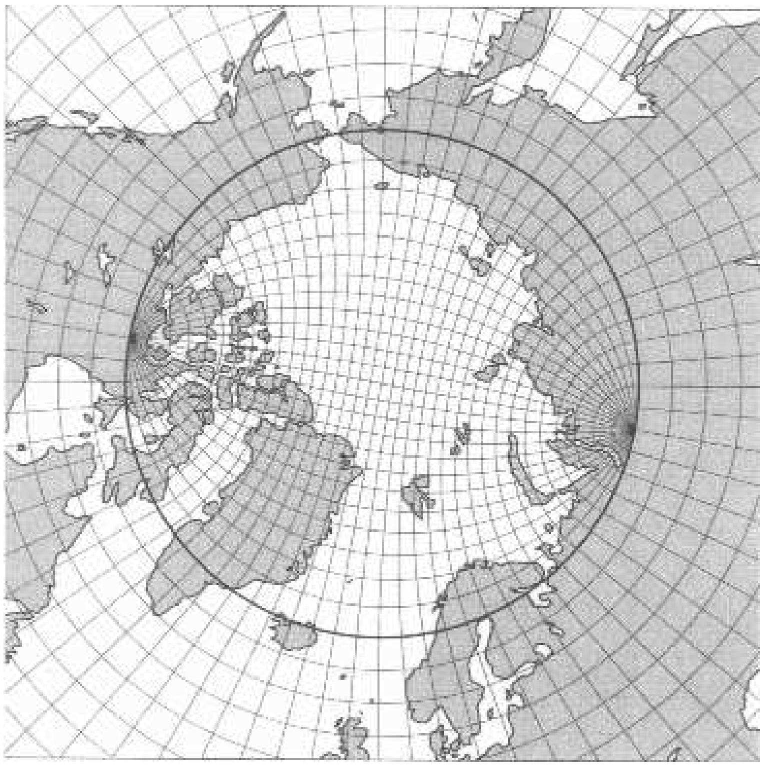


Figure 2: The tripolar grid, often used in ocean modeling. Polar singularities are placed over land and excluded from the simulation.

teracting with real topography increasingly use such unstructured grids. Section 3.4 visits the issue of the specification of such grids.

There is no need for unstructured grids to have only triangular elements (although we shall see in Section 2.6 that the *supergrid* abstraction allows us to build all such grids out of UTGs). *Unstructured polygonal grids* of arbitrary polygonal elements are a completely general abstraction, where each cell might have any number of vertices. In practice, we usually find somewhat more restrictive formulations such as in *Spectral Element Ocean Model (SEOM)*[6] of Iskandarani et al. (2002) cited earlier: an example SEOM grid for the ocean is shown in Figure 6.

A reasonably complete taxonomy of grid discretizations for the near- to mid-future in Earth System science would include:

LRG logically rectangular grid.

STG structured triangular grid.

UTG unstructured triangular grid.

UPG unstructured polygonal grid.

[6]`http://oceanmodeling.rsmas.miami.edu/seom/seom_intro.html`

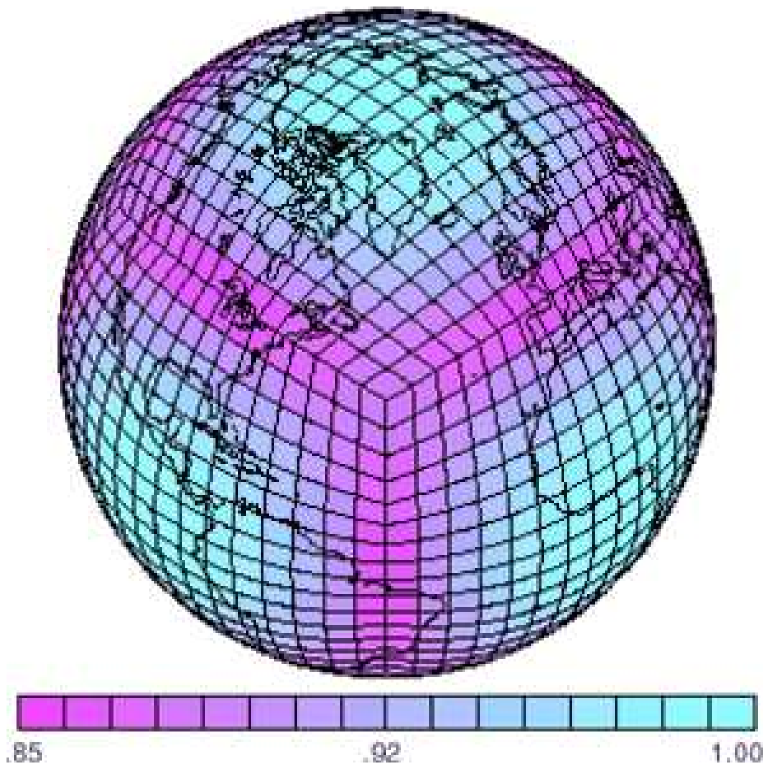


Figure 3: The cube-sphere grid, projecting the sphere onto the six faces of a cube. Polar singularities are avoided, at the expense of some grid distortion near the cube's vertices.

PCG pixel-based catchment grids: gridboxes made up of arbitrary collections of contiguous fine-grained pixels, usually used to demarcate *catchments* defined by surface elevation isolines (Koster et al. 2000).

EGG Escher gecko grid.

While developing a vocabulary and placeholders for all of the above, we shall focus here principally on logically rectangular discretizations. We shall expose the key concepts of *supergrids* (Section 2.6) and *mosaics* (Section 2.7) based on LRGs, and aim to show their relevance for other discretization types as well. We expect the specification to be extended to these other discretization types by the relevant domain experts, as in Section 3.4.

2.6. *Staggering, refinement, and the supergrid*

Algorithms place quantities at different locations within a grid cell ("staggering"). In particular, the Arakawa grids, covered in standard texts such as Haltiner and Williams (1980) show different ways to represent velocities and masses on grids, as shown in Figure 8.

This has led to considerable confusion in terminology and design: are the velocity

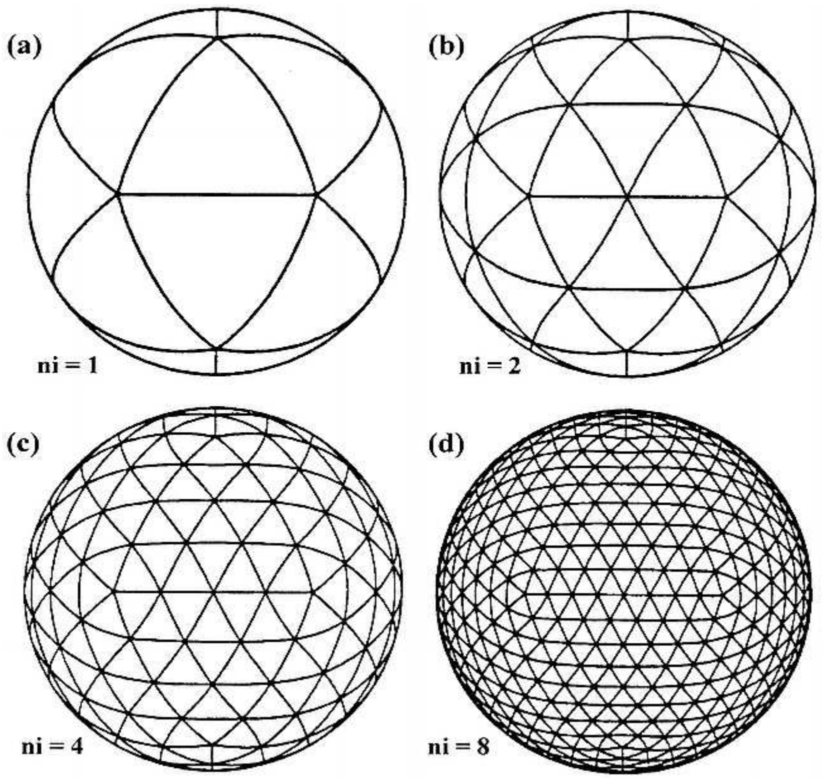


Figure 4: A structured triangular discretization of the sphere. Note that all vertices at any truncation level `ni` are also vertices at any higher level of truncation.

and mass grids to be constructed independently, or as aspects ("subgrids") of a single grid? How do we encode the relationships between the subgrids, which are necessarily fixed and algorithmically essential?

In this approach, we dispense with subgrids, and instead invert the specification: we define a *supergrid*. The supergrid is an object potentially of higher refinement than the grid that an algorithm will use; but every such grid needed by an application is a subset of the supergrid.

Given a complete specification of distances, angles, areas and volumes on a supergrid, any operation on any Arakawa grid is completely defined.

The refinement of an Arakawa grid is always 2: here we generalize the refinement factor to an arbitrary integer, so that a single high-resolution grid specification may be used to run simulations at different resolutions.

We can now define a *cell* without ambiguity: it is an element of a supergrid. A *cell* on the grid itself may be overspecified, but this guarantees that any set of staggered grids will have consistent coordinate distances and areas.

The supergrid cell itself does not have a "center": in constructing a grid from a supergrid, the grid center is indeed a vertex on the supergrid. However, certain applications of supergrids require the specification of a *centroid* (e.g Jones 1999), a

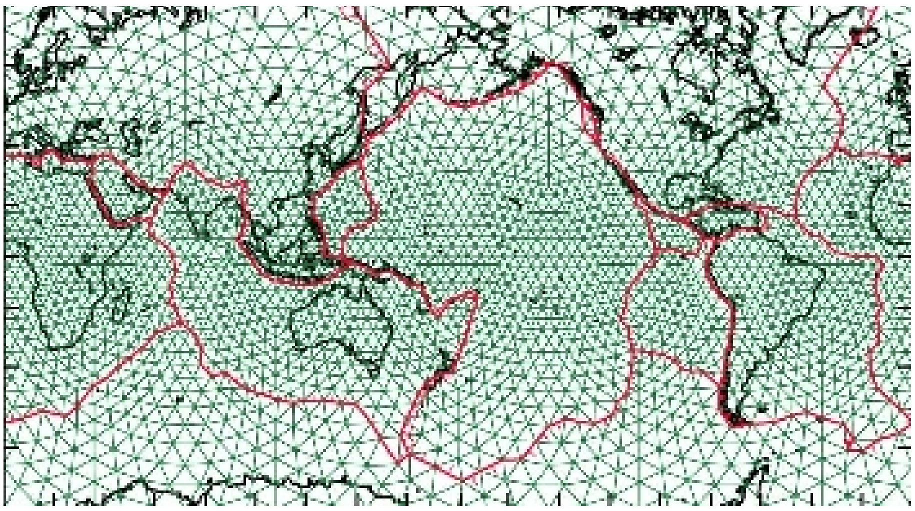


Figure 5: An unstructured triangular discretization of the sphere.

representative cell location. This is nominally some the center of some weighting field distributed about its area; but it is incorrect to try and compute a distance from centroid to a vertex.

Staggered arrays may be defined as *symmetric* or *asymmetric arrays*. Taking the Arakawa C-grid (Figure 9) as an example, we have a 8×8 supergrid. Scalars, at cell centres, will form a 4×4 array. A *symmetric array* representing the velocity component U will be of size 5×4. Quite often, though, all arrays may be defined to be 4×4, in which case, one must also specify if the U points are biased to the "east" or "west", i.e if the array value **`u(i,j)`** refers to the point $U(i+\frac{1}{2},j)$ or $U(i-\frac{1}{2},j)$. While this can be inferred from the array size, it is probably wise to include this information in the specification for readability.

Grid *refinement* is another application of supergrids. A refined grid is usually a fine grid overlying a coarse grid, with some integer factor of resolution in index space. The vertices on the coarse grid are also vertices on the fine grid, as shown in the example of Figure 10.

The coincidence of certain vertices of refined grids in contact permit certain operations more specialized than the completely generalized overlap contact region specified in Section 2.9. The supergrid plays a role here, as the vertices of a single logically rectangular supergrid can capture all of the grid information for a refined grid. Of course, adaptive refinement techniques where grids may be indefinitely refined may not allow for the prior definition of that supergrid.

2.6.1. Triangular supergrids Can the supergrid idea be extended to non-rectangular grids? It is somewhat less intuitive in this case, but it is argued in this article that the supergrid idea is equally applicable to grids that are not logically rectangular. There are several reasons to attempt to encode unstructured grids in this fashion. First, we

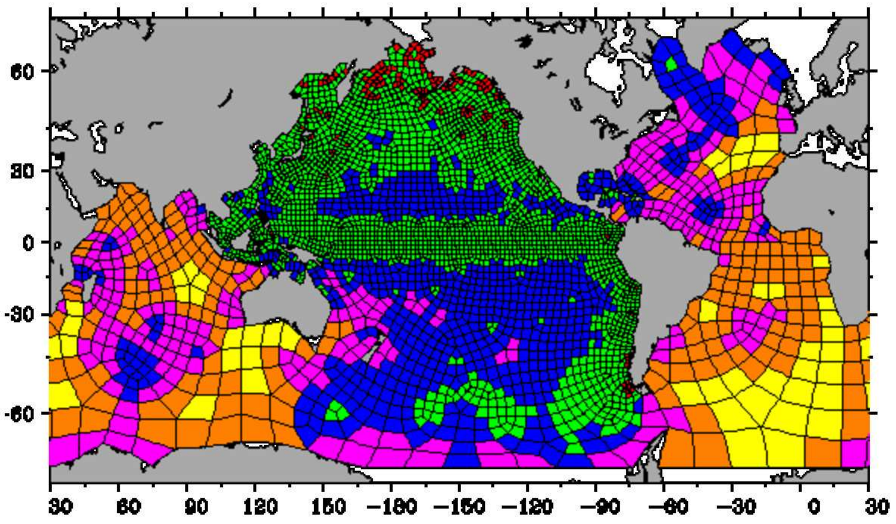


Figure 6: The SEOM unstructured grid.

see in the STG of Figure 4 that coarse resolution grids, say at **`ni`** = 1, 2 or 4, can be constructed by subsampling a supergrid defined at **`ni`** = 8. Second, staggering is a concept equally at home on triangular grids. It is common practice on STGs and UTGs to define vertex-, cell-, and face-centered quantities. Furthermore, several key interpolative algorithms on UTGs depend on these quantities, as shown in Figure 11 from Majewski et al. (2002).

The proposed treatment of unstructured grids, detailed below in Section 3.4, is to define a specification of UTGs that represent a supergrid, i.e including all vertex-, cell-, and face-centered locations. *Only UTGs need to be considered in defining a supergrid, as a triangular supergrid underlies any unstructured grid, including those containing polygons with arbitrary vertex counts.*

2.6.2. Raster grids Raster grids are a discretization of a surface into high-resolution pixels of an atomic nature: a “point” is the location of its containing raster, and any “line” is made up of discrete segments that follow raster edges but which cannot intersect them. The “area” of any grid cell on a raster is defined merely by counting the pixels within its bounding curve.

An application of raster grids is the use of catchment grids or PCGs (Koster et al. 2000). Catchment grids follow digital elevation isolines to form bounding boxes following topography to facilitate modeling land surface processes. PCGs are defined entirely in terms of an underlying raster grid.

A raster grid can also be defined on the basis of a high-resolution supergrid. Typically, these are created on the basis of high-resolution digital elevation datasets defined on a sphere. Thus raster grids are defined here as LRG supergrids. The centroid defines the raster location.

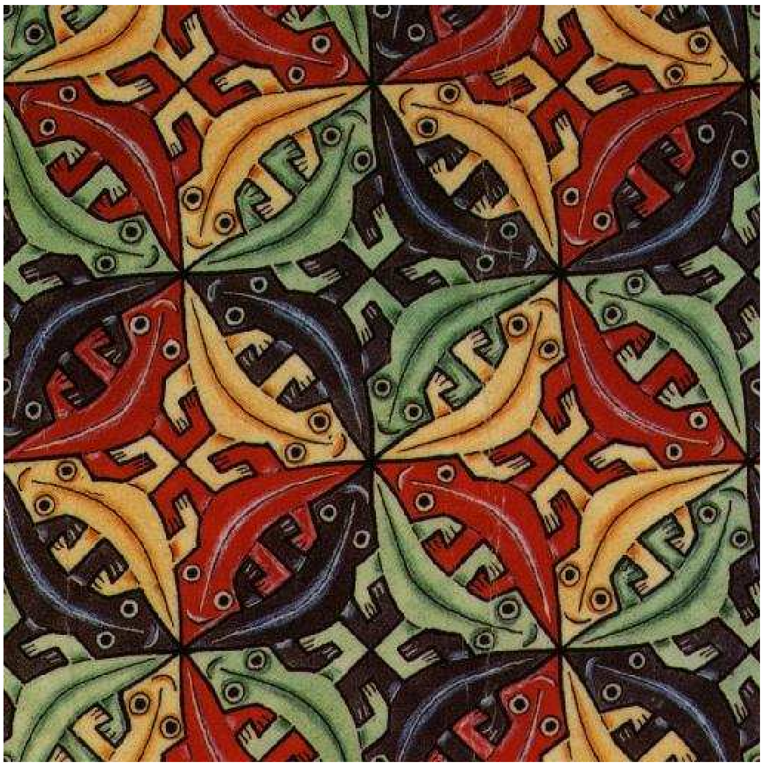


Figure 7: Another possible discretization of the plane.

2.7. *Mosaics*

In many applications, it makes sense to divide up the model into a set of *grid tiles*[7], each of which is independently discretized. An example above is the cubed-sphere of Figure 3, which is defined by six grid tiles, on which a data field may be represented by several arrays, one per tile. We call such a collection of grid tiles a grid mosaic, as shown in Figure 12.

A *grid mosaic* is constructed recursively by referring to child mosaics, with the tree terminating in leaves defined by *grid tiles* (Figure 13).

Aside from the grid information in the grid tiles, the grid mosaic additionally specifies connections between pairs of tiles in the form of *contact regions* between *pairs* of grid tiles.[8]

Contact regions can be *boundaries*, topologically of one dimension less than the grid tiles (i.e, planes between volumes, or lines between planes), or *overlaps*, topologically equal in dimension to the grid tile. In the cubed-sphere example the contact

[7]The words *grid* and *tile* separately are overused, and can mean many things depending on context. We will somewhat verbosely try always to use the term *grid tile* to avoid ambiguity.

[8]It is not necessarily possible to deduce contact regions by geospatial mapping: there can be applications where geographically collocated regions do *not* exchange data, and also where there is implicit contact between non-collocated regions.

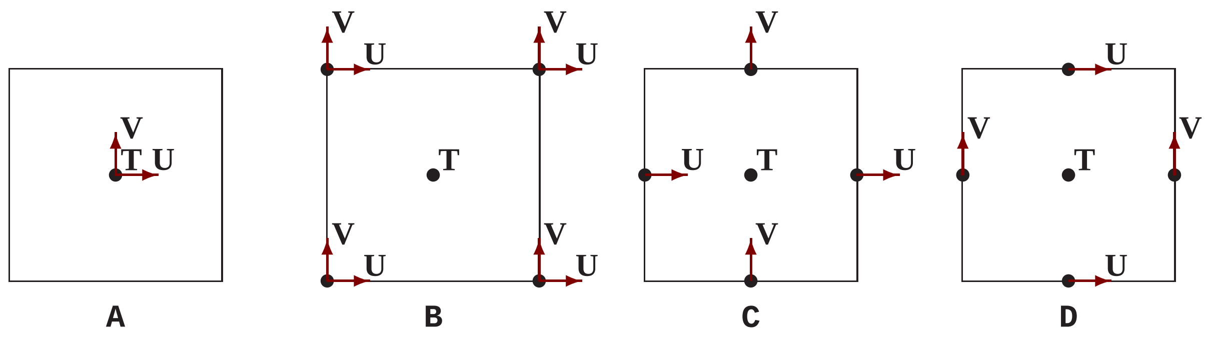


Figure 8: The Arakawa staggered grids.

regions between grid tiles are 1D boundaries: other grids may contain tiles that overlap. In the example of the *yin-yang* grid (Kageyama et al. 2004) of Figure 14 the grid mosaic contains two grid tiles that are each lon-lat grids, with an overlap. The overlap is also specified in terms of a *contact region* between pairs of grid tiles. Issues relating to boundaries are described in Section 2.8. Overlaps are described in terms of an exchange grid (e.g Balaji et al. 2006a), outlined in Section 2.9.

The grid mosaic is a powerful abstraction making possible an entire panoply of applications. These include:

- the use of overset grids such as the yin-yang grid of Figure 14;
- the representation of nested grids (e.g Kurihara et al. 1990, see Figure 15);
- the representation of reduced grids (e.g Rasch 1994). Currently these typically use full arrays and a specification of the “ragged edge”. A reduced grid can instead be written as a grid mosaic where each reduction appears as a separate grid tile.
- An entire coupled model application or dataset can be constructed as a hierarchical mosaic. Grid mosaics representing atmosphere, land, ocean components and so on, as well as contact regions between them, all can be represented using this abstraction. This approach is already in use at many modeling centres including GFDL, though not formalized.
- Finally, grid mosaics can be used to overcome performance bottlenecks associated with parallel I/O and very large files. Representing the model grid by a mosaic permits one to save data to multiple files, and the step of *aggregation* is deferred. This approach is already used at GFDL to perform distributed I/O from a parallel application, where I/O aggregation is deferred and performed on a separate I/O server sharing a filesystem with the compute server.

All of these applications make the grid mosaic abstraction central to this specification.

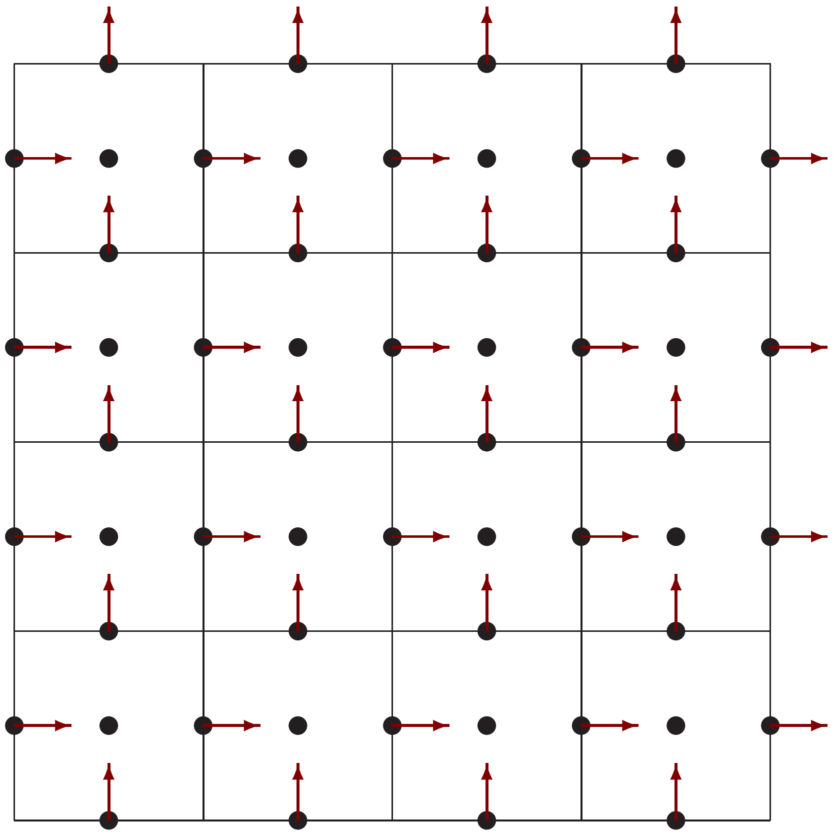


Figure 9: A 4×4 (not 8×8!) Arakawa C-grid.

2.8. *Boundary contact regions*

Boundaries for LRG tiles are specified in terms of an *anchor point* and an *orientation*. An anchor point is a boundary point that is common to the two grid tiles in contact. When possible, it is specified as integers giving index space locations of the anchor point on the two grid tiles. When there is no common grid point, the anchor point is specified in terms of floating point numbers giving a geographic location. The *orientation* of the boundary specifies the index space direction of the running boundary on each grid tile.

Figure 16 shows an example of boundaries for the cubed-sphere grid mosaic. Colored lines show shared boundaries between pairs of grid tiles: note how orientation may change so that a “north” edge on one grid tile may be in contact with a “west” edge of another. Orientation changes indicate how vector quantities are transformed when transiting a grid tile boundary.

Note that cyclic boundary conditions can be expressed as a contact region of a grid tile with itself, on opposite edges, and the polar fold in Figure 2 likewise.

Boundary conditions are considerably simplified when certain assumptions about

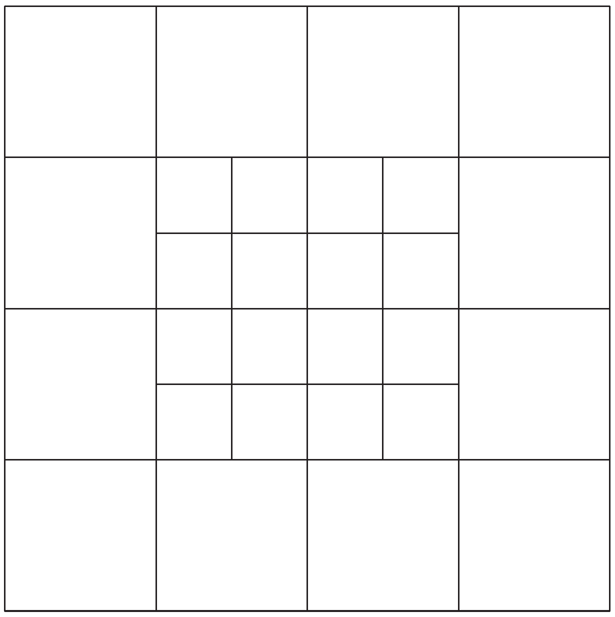


Figure 10: Nested grids with integer refinement: an inner 4×4 grid at twice the resolution is nested within the coarse 4×4 grid.

grid lines can be made. These are illustrated in Figure 17 for various types of boundaries.

A boundary has the property of *alignment* when there is an anchor point in *index space* shared by the two grid tiles, i.e it is possible to state that some point **`(i1,j1)`** on grid tile 1 is the same physical point as **`(i2,j2)`** on grid tile 2. An aligned boundary has *no refinement* when the grid lines crossing the boundary are *continuous*, as in grid tiles 1 and 2 in Figure 17. The refinement is *integer* when grid lines from the coarse grid are continuous on the fine grid, but not vice versa, see grid tiles 5 and 6. The refinement is *rational* in the example of tile 3, when the contact grid tiles have grid line counts that are co-prime.

These properties, if present, will aid in the creation of simple and fast methods for transforming data between grid tiles. If none of the conditions above are met, there is no alignment. Anchor points are then represented by geo-referenced coordinates, and remapping is mediated by an exchange, as described below in Section 2.9.

2.9. *Overlap contact regions: Exchange grids and masks*

When there are overlapping grid tiles, the *exchange grid* construct of Balaji et al. (2006a) is a useful encapsulation of all the information for conservative interpolation of scalar quantities.[9] The exchange grid, defined here, does not imply or force any

[9]Streamline-preserving interpolation of vector quantities between grids is still under study, and may result in extensions to this proposed grid standard.

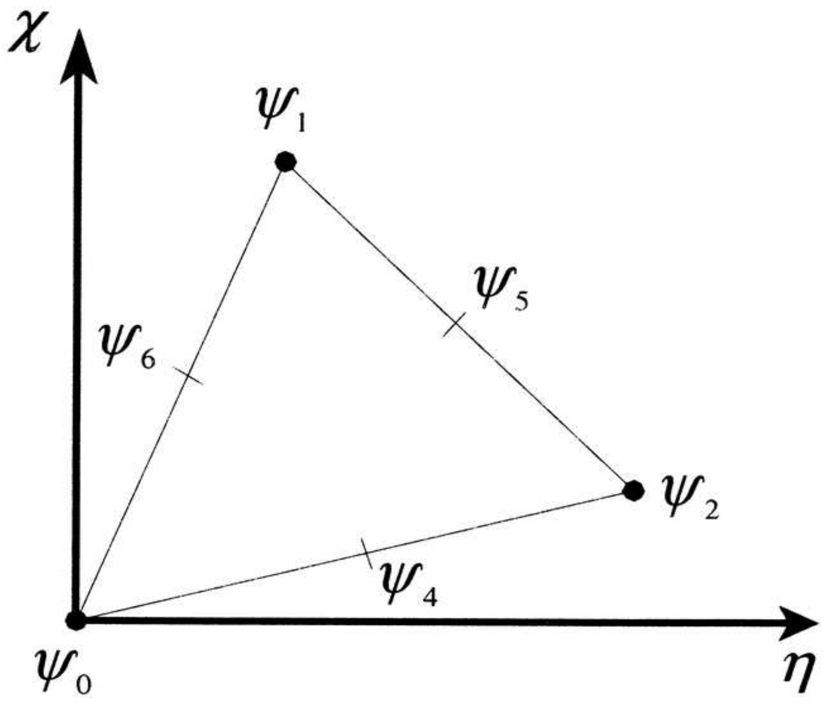


Figure 11: Vertex- and face-centered locations on a triangular grid. All of these quantites are needed for certain accurate interpolation algorithms on these grids. Further, different quantities may be placed on a different subset of points associated with this cell.

particular algorithm or conservation requirement; rather it enables conservative regridding of any order. Methods for creation of exchange grids are briefly discussed, but the standard is of course divorced from any implementation.

Given two grid tiles, an *exchange grid* is the set of cells defined by the union of all the vertices of the two parent grid tiles. This is illustrated in Figure 18 in 1D, with two parent grid tiles ("atmosphere" and "land"). (Figure 19 shows an example of a 2D exchange grid, most often used in practice). As seen here. each exchange grid cell can be uniquely associated with exactly one cell on each parent grid tile, and *fractional areas* with respect to the parent grid cells. Quantities being transferred from one parent grid tile to the other are first interpolated onto the exchange grid using one set of fractional areas; and then averaged onto the receiving grid using the other set of fractional areas. If a particular moment of the exchanged quantity is required to be conserved, consistent moment-conserving interpolation and averaging functions of the fractional area may be employed. This may require not only the cell-average of the quantity (zeroth-order moment) but also higher-order moments to be transferred across the exchange grid.

Given N cells of one parent grid tile, and M cells of the other, the exchange grid is, in the limiting case in which every cell on one grid overlaps with every cell on the other, a matrix of size $N \times M$. In practice, however, very few cells overlap, and the

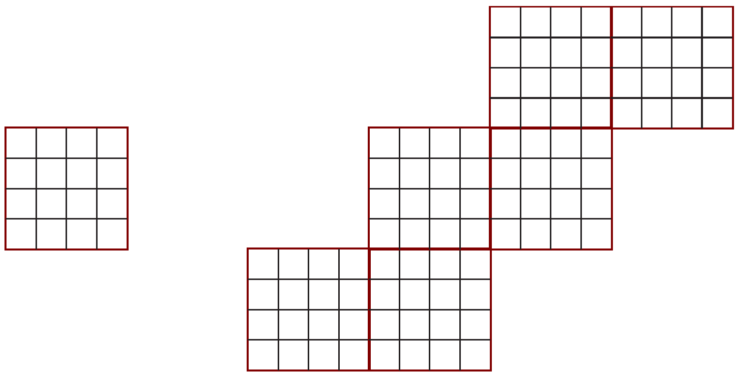


Figure 12: A grid tile: a quadrilateral grid shown in index space. A grid mosaic: a number of tiles sharing boundaries or contact regions.

exchange grid matrix is extremely sparse. In code, we typically treat the exchange grid cell array as a compact 1D array (thus shown in Figure 18 as E_l rather than E_{nm}) with indices pointing back to the parent grid tile cells. Table 1 shows the characteristics of exchange grids at typical climate model resolutions. The first is the current GFDL model CM2 (Delworth et al. 2006), and the second for a projected next-generation model still under development. As seen here, the exchange grids are extremely sparse.

Atmosphere	Ocean	Xgrid	Density	Scalability
144×90	360×200	79644	8.5×10^{-5}	0.29
288×180	1080×840	895390	1.9×10^{-5}	0.56

Table 1: Exchange grid sizes for typical climate model grids. The first column shows the horizontal discretization of an atmospheric model at “typical" climate resolutions of 2°and 1°respectively. The “ocean" column shows the same for an ocean model, at 1°and $\frac{1}{3}^{\circ}$. The “Xgrid" column shows the number of points in the computed exchange grid, and the density relates that to the theoretical maximum number of exchange grid cells. The “scalability" column shows the load imbalance of the exchange grid relative to the overall model when it inherits its parallel decomposition from one of the parent grid tiles.

The computation of the exchange grid itself could be time consuming, for parent grid tiles on completely non-conformant curvilinear coordinates. In practice, this issue is often sidestepped by precomputing and storing the exchange grid. The issue must be revisited if either of the parent grid tiles is adaptive. Methods for exchange grid computation include the *SCRIP*[10] package (Jones 1999) and others based on discretizing the underlying continuous geometry as a raster of high-resolution pixels (Koster et al. 2000).

[10]`http://climate.lanl.gov/Software/SCRIP`

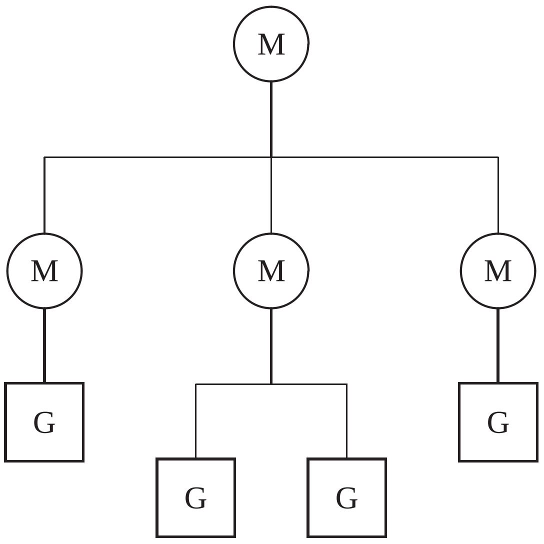


Figure 13: A grid mosaic M is constructed hierarchically; each branch of the tree terminates in a grid tile G.

This illustration of exchange grids restricts itself to 2-dimensional LRGs on the planetary surface. However, there is nothing in the exchange grid concept that prevents its use in any of the discretizations of Section 2.5, or in exchanges between grids varying in 3, or even 4 (including time) dimensions.

2.9.1. Masks A complication arises when one of the surfaces is partitioned into *complementary components*: in Earth system models, a typical example is that of an ocean and land surface that together tile the area under the atmosphere. Conservative exchange between *three* components may then be required: quantities like CO_2 have reservoirs in all three media, with the *total* carbon inventory being conserved.

Figure 19 shows such an instance, with an atmosphere-land grid and an ocean grid of different resolution. The green line in the first two frames shows the *land-sea mask* as discretized on the two grids, with the cells marked **L** belonging to the land. Due to the differing resolution, certain exchange grid cells have ambiguous status: the two blue cells are claimed by both land and ocean, while the orphan red cell is claimed by neither.

This implies that the mask defining the boundary between complementary grids can only be accurately defined on the exchange grid: only there can it be guaranteed that the cell areas exactly tile the global domain. Cells of ambiguous status are resolved here, by adopting some ownership convention. For example, in the FMS exchange grid, we generally modify the land model as needed: the land grid cells are quite independent of each other and amenable to such transformations. We add cells

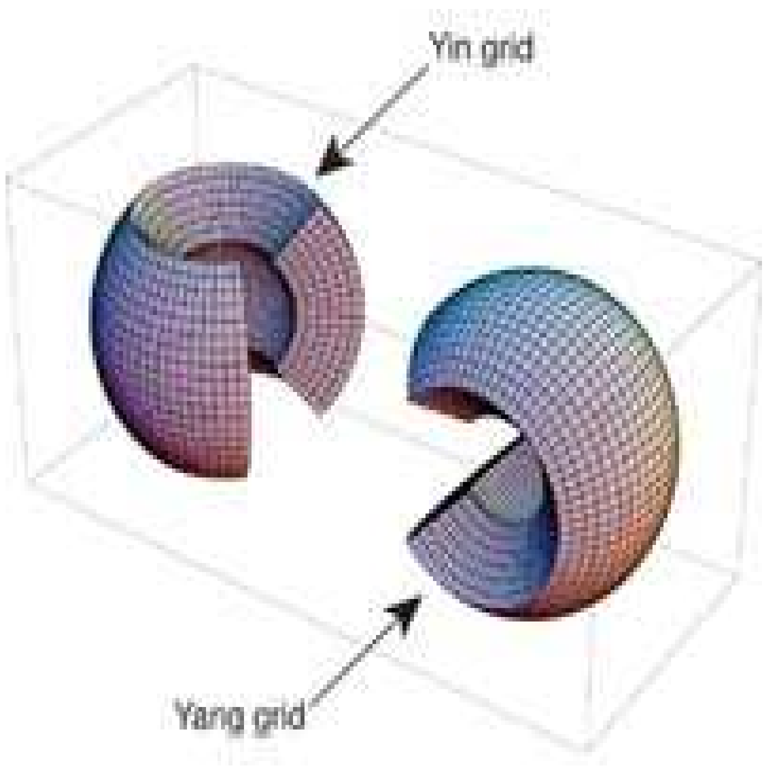


Figure 14: The yin-yang grid consists of two longitude-latitude bands with mutually orthogonal axes, and an overlap.

to the land grid until there are no orphan “red” cells left on the exchange grid, then get rid of the “blue” cells by *clipping* the fractional areas on the land side.

3. Representing the grid vocabulary in the CF conventions

The CF conventions have been developed in the context of the netCDF data format. The current momentum is toward using technologies such as OpenDAP to achieve format neutrality for data; and to develop the conventions themselves toward a standard through a mechanism such as OGC. As the standardization process continues, it is likely that much of CF metadata will be stored in databases in a readily-harvested form such as XML. For the purposes of this paper, however, we will continue to represent the contents of the grid standard using netCDF terminology, as now.

The current CF standard covers data fields for single grid tiles very well. As there are considerable data archives already storing data in this form, we have tried to do the least violence to existing data representations of variables on single grid tiles. The proposed extensions serve as enhancements to CF that will allow a full expression for

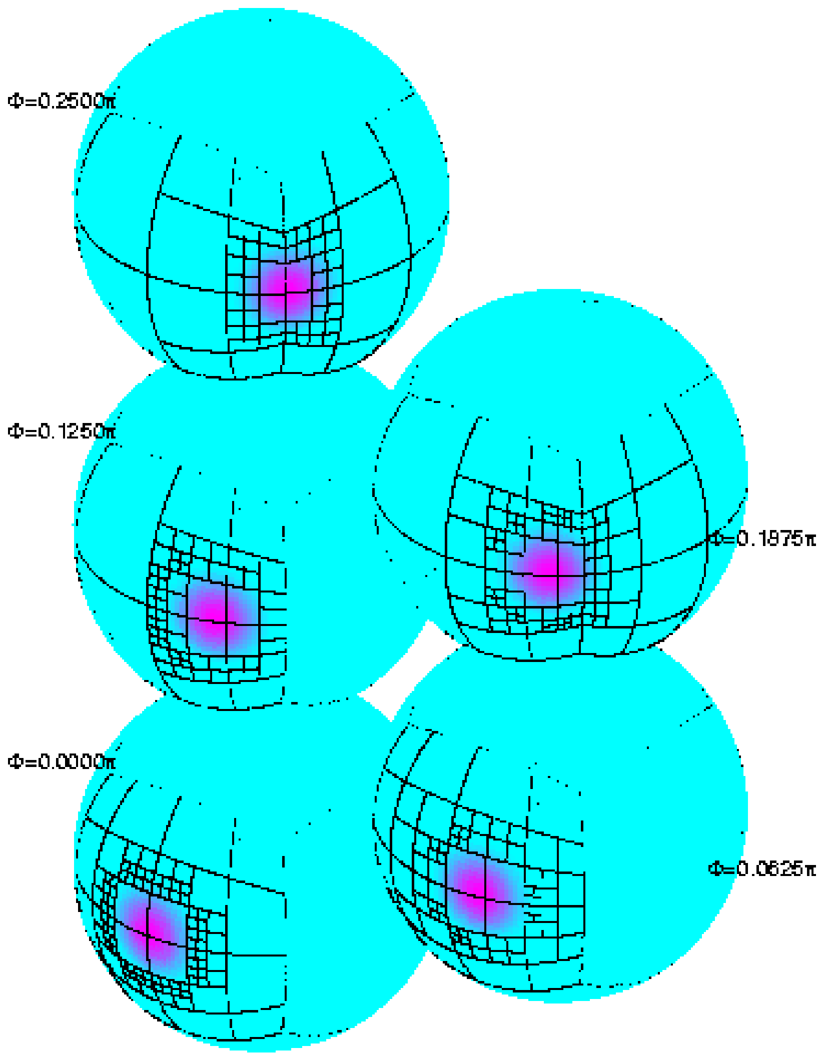


Figure 15: A cubed-sphere with embedded nests.

data discretized on grid mosaics. Features to highlight include:

- a standard grid specification dataset (or *gridspec*) for grid mosaics. The grid specification is comprehensive and is potentially a very large file. Various CF attributes will be used to indicate properties of the grid that permit a succinct description from which the complete gridspec is readily reconstructed.
- an extended family of CF standard names for grid specification;
- netCDF and CF currently assume that all information is present in a single *file*. This assumption is already currently broken in many ways: for instance it is customary to store a long time series of a variable in multiple files. The assumption is also often flawed for vector fields: vector components may be stored as mul-

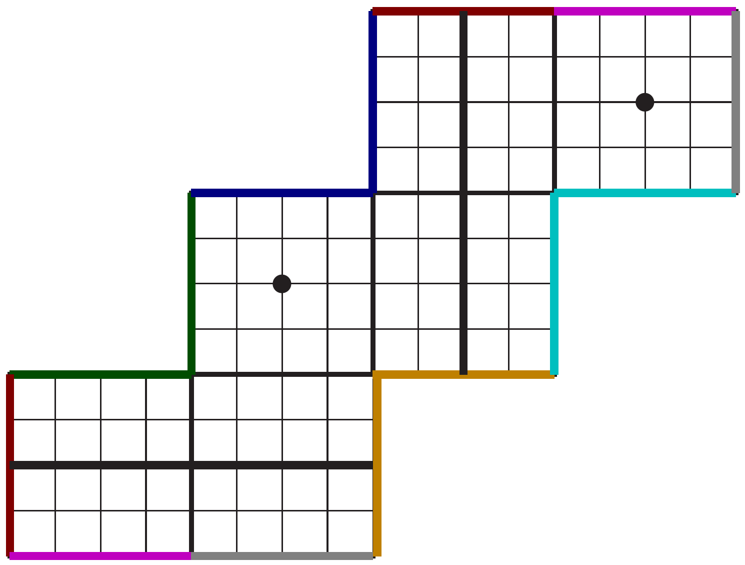


Figure 16: The cubed-sphere grid mosaic.

tiple files. We propose here a mechanism for storing a CF-compliant *dataset* in multiple *files*[11], and for preserving (or at least verifying) integrity of a multi-file dataset.

- The gridspec is a work in progress, and is designed for extensibility. We expect to see considerable evolution in the near term. It is therefore liberally sprinkled with *version* metadata.

The general approach is as follows. Datasets are generally archived in a way whereby one approaches the dataset following metadata that describes the experiment to which it belongs. The gridspec forms part of the experiment metadata. For Earth System models, comprehensive model metadata is under development. A gridspec describing the complete grid mosaic of an entire coupled model (shown schematically in Figure 13) will be stored under the experiment, and we expect software processing any dataset associated with the experiment to have access to the gridspec.[12]

Datasets holding physical variables will not themselves refer to the gridspec; the connection is made at the metadata level above.

Physical variables discretized on a mosaic of more than one grid tile may be stored in multiple files, where each file contains one or more grid tiles.

[11]The HDF5 specification, with which netCDF will merge, takes a filesystem-within-a-file approach to this problem, which by all accounts is not very efficient **(missing ref:)**. The proposed approach will allow very efficient approaches to dataset aggregation.

[12]As the gridspec is also intended for use as model input, said software might indeed be an Earth system model.

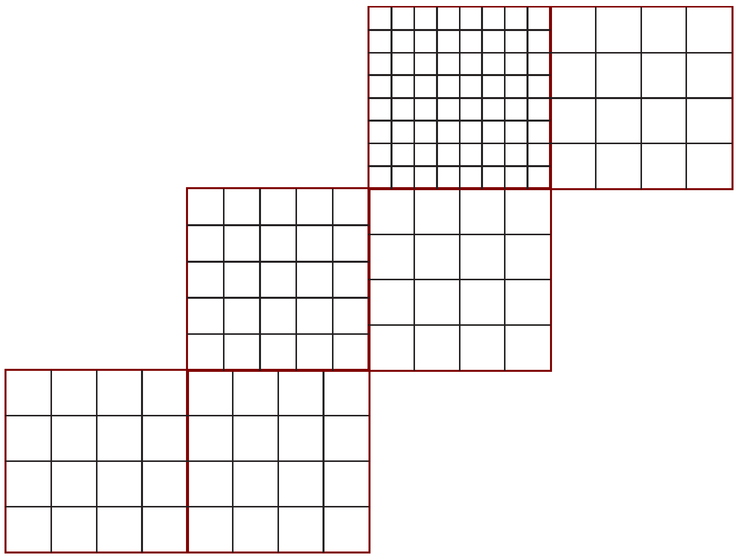


Figure 17: Grid refinement on a cubed-sphere grid mosaic.

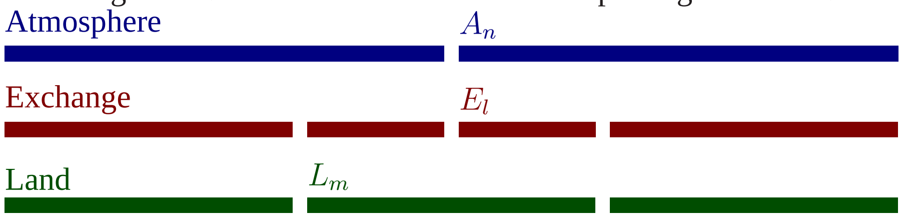


Figure 18: One-dimensional exchange grid.

3.1. *Linkages between files*

We propose that links be directed and acyclic: e.g grid mosaic files point to constituent grid tile files, but the "leaf" files do not point back.

Files may be described using local pathnames or remote URIs (URLs, OpenDAP IDs). File descriptors may be absolute or relative to a base address, as in HTML.

When pointing to an external file, attributes holding the timestamp and MD5 checksum[13] may optionally be specified. If the checksum of an external file does not match, it is an error. The timestamp is not definitive, but may be used to decide whether or not to trigger a checksum.

[13]MD5 checksums are standard practice. One can intentionally generate, by bit exchanges, erroneous files that give the same MD5 checksum, but the probability of this occurring by coincidence is vanishingly small. MD5 checksums have been measured to take about a minute for a 10Gb dataset.

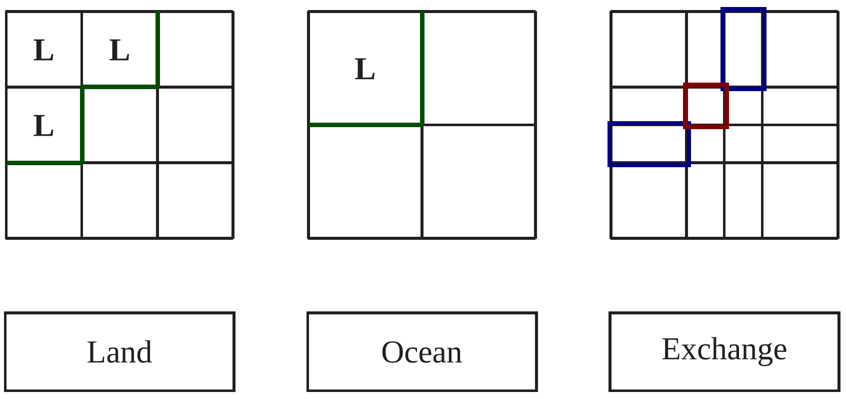


Figure 19: The mask problem. The land and atmosphere share the grid on the left, and their discretization of the land-sea mask is different from the ocean model, in the middle. The exchange grid, right, is where these may be reconciled: the red “orphan" cell is assigned (arbitrarily) to the land, and the land cell areas “clipped" to remove the doubly-owned blue cells.

```
dimensions:
   string = 255;
variables:
   char base(string);
   char external(string);
   char local(string);
base = "http://www.gfdl.noaa.gov/CM2.1/";
   base:standard_name = "link_base_path";
external = "foo.nc";
   external:standard_name = "link_path";
   external:md5_checksum = "g0bbl3dyg00k";
   external:timestamp = "20060509T012800.33Z";
local = "/home/foo/bar.nc";
   local:standard_name = "link_path";
   local:link_spec_version = "0.2";
```
(1)

Encoding pathnames, checksums and timestamps carries a penalty: the system is brittle to any changes. The use of relative pathnames is recommended: this at least permits whole directory trees to be moved with little pain.
Summary: two new standard names `link_base_path` and `link_path`. Optional attributes: `link_spec_version`, `md5_checksum` and `timestamp`.

3.2. *Grid mosaic*

The grid mosaic specification is identified by a unique string name which qualifies its interior namespace. As shown schematically in Figure 13, its children can be mosaics or grid tiles. Contact regions are specified between pairs of grid tiles, using the fully qualified grid tile specification *mosaic:mosaic:...:tile*.

```
dimensions:
   nfaces = 6;
   ncontact = 12;
   string = 255;
variables:
   char mosaic(string);
   char gridfaces(nfaces,string);
   char contacts(ncontact,string);
mosaic = "AM2C45L24";
   mosaic:standard_name = "grid_mosaic_spec";
   mosaic:mosaic_spec_version = "0.2";
   mosaic:children = "gridfaces";
   mosaic:contact_regions = "contacts";
   mosaic:grid_descriptor = "C45L24 cubed_sphere";
gridfaces =
   "Face1",
   "Face2",
   "Face3",
   "Face4",
   "Face5",
   "Face6";
contacts =
   "AM2C45L24:Face1::AM2C45L24:Face2",
   "AM2C45L24:Face1::AM2C45L24:Face3",
   "AM2C45L24:Face1::AM2C45L24:Face5",
   "AM2C45L24:Face1::AM2C45L24:Face6",
   "AM2C45L24:Face2::AM2C45L24:Face3",
   "AM2C45L24:Face2::AM2C45L24:Face4",
   "AM2C45L24:Face2::AM2C45L24:Face6",
   "AM2C45L24:Face3::AM2C45L24:Face4",
   "AM2C45L24:Face3::AM2C45L24:Face5",
   "AM2C45L24:Face4::AM2C45L24:Face5",
   "AM2C45L24:Face4::AM2C45L24:Face6",
   "AM2C45L24:Face5::AM2C45L24:Face6";
```

(2)

Summary: a new standard names `grid_mosaic_spec`. Grid mosaic specs have attributes `mosaic_spec_version`, `children` and `contact_regions`. Optional attributes `children_links` and `contact_region_links` may point to external files containing the specifications for the children and their contacts.

The *grid_descriptor* is an optional text description of the grid that uses commonly used terminology, but may not in general be a sufficient description of the field (many grids are numerically generated, and do not admit of a succinct description). Examples of grid descriptors include:

- **`spectral_gaussian_grid`**
- **`regular_lon_lat_grid`**
- **`reduced_gaussian_grid`**
- **`displaced_pole_grid`** (different from a *rotated pole grid*: any grid could have a rotated north pole);
- **`tripolar_grid`**
- **`cubed_sphere_grid`**
- **`icosahedral_geodesic_grid`**
- **`yin_yang_grid`**

The grid descriptor could additionally contain common shorthand descriptions such as **`t42`**, or perhaps could go further toward machine processing using terms like **`triangular_truncation`**.

3.3. *Grid tile*

```
dimensions:
   string = 255;
   nx = 90;
   ny = 90;
   nxv = 91;
   nyv = 91;
   nz = 24;
variables:
   char tile(string);
      tile:standard_name = "grid_tile_spec";
      tile:tile_spec_version = "0.2";
      tile:geometry = "spherical";
      tile:north_pole = "0.0 90.0";
      tile:projection = "cube_gnomonic";
      tile:discretization = "logically_rectangular";
      tile:conformal = "true";
   double area(ny,nx);
      area:standard_name = "grid_cell_area";
      area:units = "m2̂";
   double dx(ny+1,nx);
      dx:standard_name = "grid_edge_x_distance";
      dx:units = "metres";
   double dy(ny,nx+1);
      dy:standard_name = "grid_edge_y_distance";
      dy:units = "metres";
   double angle_dx(ny+1,nx);
      angle_dx:standard_name =
         "grid_edge_x_angle_WRT_geographic_east";
      angle_dx:units = "radians";
   char arcx(string);
      arcx:standard_name = "grid_edge_x_arc_type";
   double zeta(nz);
arcx = "great_circle";
tile = "Face1";
```

(3)

Horizontal vertex location specifications may be of different rank depending on their regularity or uniformity. (Note that the geo-referencing information may still be 2D even for regular coordinates).

An irregular horizontal grid requires a 2D specification of vertex locations:

```
variables:
   float geolon(ny+1,nx+1);
      geolon:standard_name = "geographic_longitude";
   float geolat(ny+1,nx+1);
      geolat:standard_name = "geographic_latitude";
   float xvert(ny+1,nx+1);
      xvert:standard_name = "grid_longitude";
      xvert:geospatial_coordinates = "geolon geolat";
   float yvert(ny+1,nx+1);
      yvert:standard_name = "grid_latitude";
      yvert:geospatial_coordinates = "geolon geolat";
```

(4)

The vertical geo-mapping is expressed by reference to “standard levels”.

Summary: several new standard names to describe properties of a grid: distances, angles, areas and volumes. The *arc type* is a new variable with no equivalent in CF. Currently, we are considering values of `great_circle` and `small_circle`, but others may be imagined. The `small_circle` arc type requires the specification of a pole.

The grid tile spec has attributes geometry (Section 2.1), projection (Section 2.3: a value of `none` indicates no projection) and discretization (Section 2.5). The optional attributes `regular`, `conformal` and `uniform` may be used to shrink the grid tile spec.

3.4. *Unstructured grid tile*

The unstructured grid tile is an UTG. The current specification follows an actual example used by the FVCOM model **(missing ref: Gross; Signell)**. While in the LRG example above, the number of vertices can be deduced from the number of cells, it cannot in the unstructured case.

Each cell is modeled as triangular. Distances, arc types, angles and areas are cell properties. Additional elements of the UTG specification are variables with standard names of `vertex_index` and `neighbor_cell_index` to contain the indices of a cell’s 3 vertices and its 3 neighbours, respectively. The ordering line segments, neighbors, etc., all follow the ordering of vertices.

```
dimensions:
   string = 255;
   node = 871;
   nele = 1620;
variables:
   char tile(string);
      tile:standard_name = "grid_tile_spec";
      tile:tile_spec_version = "0.2";
      tile:geometry = "spherical";
      tile:north_pole = "0.0 90.0";
      tile:discretization =
         "unstructured_triangular";
   double area(nele);
      area:standard_name = "grid_cell_area";
      area:units = "m2̂";
   double ds(3,nele);
      ds:standard_name = "grid_edge_distance";
      ds:units = "metres";
   double angle_ds(3,nele);
      angle_ds:standard_name =
         "grid_edge_angle_WRT_geographic_east";
      angle_ds:units = "radians";
   char arcx(string);
      arcx:standard_name = "grid_edge_arc_type";
   int nv(3,nele);
      nv:standard_name = "neighbor_cell_index";
   int node_index(3,nele);
      node_index:standard_name = "vertex_index";
arcx = "great_circle";
tile = "fvcom_grid";
```

(5)

```
variables:
   float geolon(node);
      geolon:standard_name = "geographic_longitude";
   float geolat(node);
      geolat:standard_name = "geographic_latitude";
   float xvert(node);
      xvert:standard_name = "grid_x_coordinate";
      xvert:units = "metres";
      xvert:geospatial_coordinates = "geolon geolat";
   float yvert(node);
      yvert:standard_name = "grid_y_coordinate";
      yvert:units = "metres";
      yvert:geospatial_coordinates = "geolon geolat";
```
(6)

3.5. *Contact regions*

```
dimensions:
   string = 255;
variables:
   int anchor(2,2);
      anchor:standard_name =
         "anchor_point_shared_between_tiles";
   char orient(string);
      orient:standard_name =
         "orientation_of_shared_boundary";
   char contact(string);
      contact:standard_name = "grid_contact_spec";
      contact:contact_spec_version = "0.2";
      contact:contact_type = "boundary";
      contact:alignment = "true";
      contact:refinement = "none";
      contact:anchor_point = "anchor";
      contact:orientation = "orient";
contact = "AM2C45L24:Face1::AM2C45L24:Face2";
orient = "Y:Y";
anchor = "90 1 1 1";
```
(7)

```
dimensions:
   string = 255;
   ncells = 1476;
variables:
   double frac_area(2,ncells);
      frac_area:standard_name =
         "fractional_area_of_exchange_grid_cell";
   int tile1_cell(2,ncells);
      tile1_cell:standard_name="parent_cell_indices";
   int tile2_cell(2,ncells);
      tile2_cell:standard_name="parent_cell_indices";
   char contact(string);
      contact:standard_name = "grid_contact_spec";
      contact:contact_spec_version = "0.2";
      contact:contact_type = "exchange";
      contact:fractional_area_field = "frac_area";
      contact:parent1_cell = "tile1_cell";
      contact:parent2_cell = "tile2_cell";
contact = "CM2:LM2::AM2C45L24:Face2";
```

(8)

3.6. *Variables*

Variables are held in CF-compliant files that are separate from the gridspec but can link to it following the link spec in Section 3.1. Variables on a single grid tile can follow CF-1.0, with no changes. The additional information provided by the gridspec can be linked in, as shown in this example of a U velocity component on a C grid (Figure 9).

```
dimensions:
   nx = 46;
   ny = 45;
variables:
   int nx_u(nx);
   int ny_u(ny);
   float u(ny,nx);
      u:standard_name = "grid_eastward_velocity";
      u:staggering = "c_grid_symmetric";
GLOBAL ATTRIBUTES:
   gridspec = "foo.nc";
nx_u = 1,3,5,...
ny_u = 2,4,6,...
```

(9)

The *staggering* field expresses what is implicit in the values of **`nx_u`** and **`ny_u`**, but is useful nonetheless[14]. Possible values of **`staggering`** include:

- **`c_grid_symmetric`**
- **`c_grid_ne`**
- **`b_grid_sw`**
- ... and so on.

Using this information, it is possible to perform correct transformations, such as combining this field with a V velocity from another file, transforming to an A-grid, and then rotating to geographic coordinates.

4. Examples

4.1. *Cartesian geometry*

```
dimensions:
   string = 255;
   nx = 8;
   ny = 8;
variables:
   char tile(string);
      tile:standard_name = "grid_tile_spec";
      tile:tile_spec_version = "0.2";
      tile:geometry = "planar";
      tile:projection = "cartesian";
      tile:discretization = "logically_rectangular";
      tile:conformal = "true";
      tile:uniform = "true";
   double area;
      area:standard_name = "grid_cell_area";
      area:units = "m2̂";
   double dx;
      dx:standard_name = "grid_edge_x_distance";
      dx:units = "metres";
   double dy;
      dy:standard_name = "grid_edge_y_distance";
      dy:units = "metres";
tile = "Descartes";
```

(10)

The Cartesian grid spec of CodeBlock 10 illustrates several simplifications with respect to CodeBlock 3.

[14]In general, there may be a lot of redundancy in the gridspec, which poses a *consistency* problem. In general, consistency checking and validation are relatively simple, as in the instance here.

- The **`geometry:planar`** attribute (Section 2.1) indicates that geo-referencing is not possible.
- Since the **`uniform`** attribute (Section 2.3) is set, the **`area`**, **`dx`** and **`dy`** fields reduce to simple scalars.
- The combination of a conformal attribute and the planar geometry means that it is not required to store angles: grid lines are orthogonal, and that's that.
- The tile name is of course arbitrary: we have chosen to type the tile as a string to avoid using the derived or complex types of *netCDF-4*[15]. Mosaic processing tools will enforce the absence of two tiles bearing the same name.

Note that this gridspec might actually represent a supergrid of a 4×4 grid: we cannot tell from the gridspec alone. We would need to examine a field containing a physical variable (Section 3.6).

[15]**`http://www.unidata.ucar.edu/software/netcdf/netcdf-4`**

4.2. *Gaussian grid*

```
dimensions:
   string = 255;
   nx = 320;
   ny = 160;
variables:
   char tile(string);
      tile:standard_name = "grid_tile_spec";
      tile:tile_spec_version = "0.2";
      tile:geometry = "spherical";
      tile:north_pole = "0.0 90.0";
      tile:discretization = "logically_rectangular";
      tile:horizontal_grid_descriptor = "gaussian_grid";
      tile:conformal = "true";
      tile:regular = "true";
   double area(ny,nx);
      area:standard_name = "grid_cell_area";
      area:units = "m2̂";
   double dx(nx);
      dx:standard_name = "grid_edge_x_distance";
      dx:units = "metres";
   double dy(ny);
      dy:standard_name = "grid_edge_y_distance";
      dy:units = "metres";
   double angle_dx(,nx);
      angle_dx:standard_name =
         "grid_edge_x_angle_WRT_geographic_east";
      angle_dx:units = "radians";
   char arcx(string);
      arcx:standard_name = "grid_edge_x_arc_type";
   double zeta(nz);
arcx = "small_circle";
tile = "T106";
```

(11)

```
dimensions:
   string = 255;
variables:
   int anchor(2,2);
      anchor:standard_name =
         "anchor_point_shared_between_tiles";
   char orient(string);
      orient:standard_name =
         "orientation_of_shared_boundary";
   char contact(string);
      contact:standard_name = "grid_contact_spec";
      contact:contact_spec_version = "0.2";
      contact:contact_type = "boundary";
      contact:alignment = "true";
      contact:refinement = "none";
      contact:anchor_point = "anchor";
      contact:orientation = "orient";
contact = "Gaussian::Gaussian";
orient = "Y:Y";
anchor = "320 1 1 1";
```

(12)

A Gaussian grid is a spatial grid where locations on a sphere are generated by "Gaussian quadrature" from a given truncation of spherical harmonics in spectral space.

- There is no projection onto a plane.
- Since this is a **`regular`** grid (Section 2.3), **`dx`** and **`dy`** are 1D rather than 2D arrays. The specification of angles is similarly reduced by the **`conformal`** attribute.
- The contact spec in CodeBlock 12 specifies periodicity in X.
- The associated mosaic specification is not shown here, as a simple Gaussian grid is a mosaic of a single tile. The **`horizontal_grid_descriptor`** (Section 3.2) is given a value of **`spectral_gaussian_grid`**: this value belongs to a **`controlled vocabulary`** of grid descriptors. The combination of this descriptor with the truncation level is enough to completely specify the gaussian grid.

4.3. *Reduced gaussian grid*

A Gaussian grid is of course a kind of **`regular_lat_lon_grid`**, and can suffer from various numerical problems owing to the convergence of longitudes near the poles. The *reduced* Gaussian grid of Hortal and Simmons (1991) overcomes this problem by reducing the number of longitudes within latitute bands approaching the pole, as shown in Figure 20.

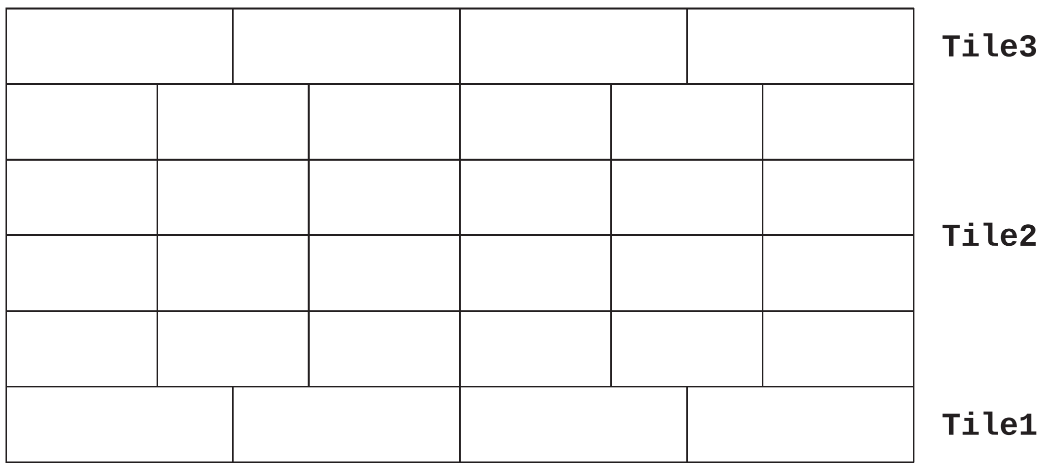


Figure 20: Reduced Gaussian grid.

```
dimensions:
   ntiles = 6;
   ncontact = 5;
   string = 255;
variables:
   char mosaic(string);
   char gridtiles(nfaces,string);
   char contacts(ncontact,string);
mosaic = "Hortal";
   mosaic:standard_name = "grid_mosaic_spec";
   mosaic:mosaic_spec_version = "0.2";
   mosaic:children = "gridfaces";
   mosaic:contact_regions = "contacts";
   mosaic:grid_descriptor = "reduced_gaussian_grid";
gridtiles =
   "Tile1",
   "Tile2",
   "Tile3";
contacts =
   "Hortal:Tile1::Hortal:Tile1",
   "Hortal:Tile2::Hortal:Tile2",
   "Hortal:Tile3::Hortal:Tile3",
   "Hortal:Tile1::Hortal:Tile2",
   "Hortal:Tile2::Hortal:Tile3";
...
contact = "Hortal:Tile1::Hortal:Tile1";
orient = "Y:Y";
anchor = "1 1 5 1";
contact = "Hortal:Tile2::Hortal:Tile2";
orient = "Y:Y";
anchor = "1 1 7 1";
contact = "Hortal:Tile3::Hortal:Tile3";
orient = "Y:Y";
anchor = "1 1 5 1";
contact = "Hortal:Tile1::Hortal:Tile2";
orient = "X:X";
anchor = "1 2 1 1";
contact = "Hortal:Tile2::Hortal:Tile3";
orient = "X:X";
anchor = "1 5 1 1";
```

(13)

The reduced Gaussian grid of Figure 20 is represented as a mosaic of multiple grid tiles, each of which is restricted to a latitude band, and has different longitudinal resolution.

- The mosaic as a whole has the **`reduced_gaussian_grid`** descriptor.

- It consists of 3 tiles, as shown in Figure 20, and 5 contact regions. The first 3 contacts express periodicity in X within a tile; the last two express contacts between tiles at the latitude where the zonal resolution changes.

4.4. *Tripolar grid*

The tripolar grid of Figure 2 is a LRG mosaic consisting of a single tile. The tile is in contact with itself in the manner of a sheet of paper folded in half. In the X direction, we have simple periodicity. Along the north edge, there is a fold, which is best conceived of a boundary in contact with itself with reversed orientation. Thus, given a tripolar grid called **`murray`** of $M \times N$ points, we would have:

```
contact = "murray::murray X";
orient = "Y:Y";
anchor = "1 M 1 1";
contact = "murray::murray Y";
orient = "X:-X";
anchor = "1 N M N;
```
(14)

4.5. *Unstructured triangular grid*

We show here an example of fields on a UTG following the FVCOM example of Section 3.4. The example shows vertex-centred scalars and cell-centered velocities:

```
variables:
   float u(nele);
      u:standard_name = "eastward_velocity";
      u:staggering = "cell_centred";
   float v(nele);
      v:standard_name = "northward_velocity";
      v:staggering = "cell_centred";
   float t(node);
      t:standard_name = "temperature";
      t:staggering = "vertex_centred";
GLOBAL ATTRIBUTES:
   gridspec = "foo.nc";
```
(15)

5. Gridspec implementations

There are two pioneering implementations of the Mosaic Gridspec. One is a complete XML schema developed on the basis of the Gridspec; the other is a complete netCDF-3 implementation.

5.1. *The GENIE Gridspec*

The *GENIE project*[16] has the objective of building a Grid-based Earth system model that will built out of component models drawn from various sites across the Grid. Component models will be on their own grid mosaics; the Gridspec will be used to generate custom coupler and regridding code on the basis of the PRISM/OASIS coupler using the BFG (Dahl 1982).

The implementation was done completely in XML. To quote the *GENIE Gridspec*[17],

> The gridspec has been implemented as an XML schema in preference to NetCDF to fit in with the XML metadata implementation used by BFG; eventually the gridspec should be available in both NetCDF/CF and XML formats, making it accessible to a wide range of Earth system modelling tools and programs.

Indeed, the second implementation cited here uses the netCDF-3 specification of the Gridspec.

5.2. *The GFDL implementation*

The GFDL Earth system models have long used the *exchange grid* (Balaji et al. 2006b) as a means of flexible transfer between model components on independent grids. The exchange grid can be expensive to compute, and so has always been pre-computed and stored as a netCDF file within GFDL. As we expand the scope of our models to include mosaics (for instance, a cubed-sphere atmospheric model), it has become necessary to revise the grid specification. It was in the process of this revision that the Mosaic Gridspec was devised.

The Mosaic Gridspec 0.2 specification is currently being deployed in GFDL production codes that couple an atmosphere on a **`cube_sphere_grid`**, an ocean and a sea-ice model on a **`tripolar_grid`**, and a land model on a **`lat_lon_grid`**. The same Gridspec is also used for transformations of saved data between the various grids.

A complete suite of netCDF files expressing this gridspec, and a set of C programs for generating these, are being made available through Balaji's grid page.

6. Summary

The grid specification proposed serves two purposes: in various contexts, these purposes have been described as *descriptive* and *prescriptive*; *semantic* and *syntactic*; or

[16] **`http://www.genie.ac.uk`**
[17] **`http://source.ggy.bris.ac.uk/wiki/GENIE_Gridspec`**

discovery and *use* metadata. The first purpose serves up information for human consumption: attaching this metadata to model output will enable a user to ask several key questions to understand its content, find out whether indeed the dataset meets a given scientific need. The second purpose is to delve further and perform operations upon datasets: as stated in Section 1.3, these include commonly performed scientific analysis and visualization: differential and integral calculus on vector and scalar fields; and transformations from one grid to another. The intent is that given the existence of a standard representation of grids, many of these operations will be abstracted into commonly available tools and analysis packages, and in fact may be available as web services.

An abstract representation (UML diagram) of the first class of metadata is shown in Figure 21. It is expected that this content will eventually appear as part of a standard XML schema to be applied to data discovery. The content of this schema will be part of extensible controlled vocabularies to be defined by the appropriate domain specialists.

The second class of metadata is far more detailed (Figure 22). This UML diagram shows schematically how locations, distances, areas and volumes of grid cells are conceptually linked into a structure culminating in a grid mosaic. While also in principle represented by a schema, these metadata are likely to be large in size and stored in datasets in some standard data format, netCDF being the canonical example shown here.

It is possible to deduce the semantic content from the syntactic: for instance, one could work out whether a model used a C-grid by comparing vector and scalar field locations. Nonetheless, it would be recommended and probably mandatory to include the very useful semantic descriptors. Validators could be used to address the consistency problem and ensure that the redundant information was indeed correct.

The draft specification, accompanied by prototype tools for producing and using some example gridspec files, will be released to the CF community in early 2007.

Acknowledgments

References

Balaji, V., J. Anderson, I. Held, M. Winton, J. Durachta, S. Malyshev, and R. J. Stouffer, 2006a: The Exchange Grid: a mechanism for data exchange between Earth System components on independent grids. *Parallel Computational Fluid Dynamics: Theory and Applications, Proceedings of the 2005 International Conference on Parallel Computational Fluid Dynamics, May 24-27, College Park, MD, USA*, A. Deane, G. Brenner, A. Ecer, D. Emerson, J. McDonough, J. Periaux, N. Satofuka, and D. Tromeur-Dervout, eds., Elsevier.

— 2006b: The Exchange Grid: a mechanism for data exchange between Earth System

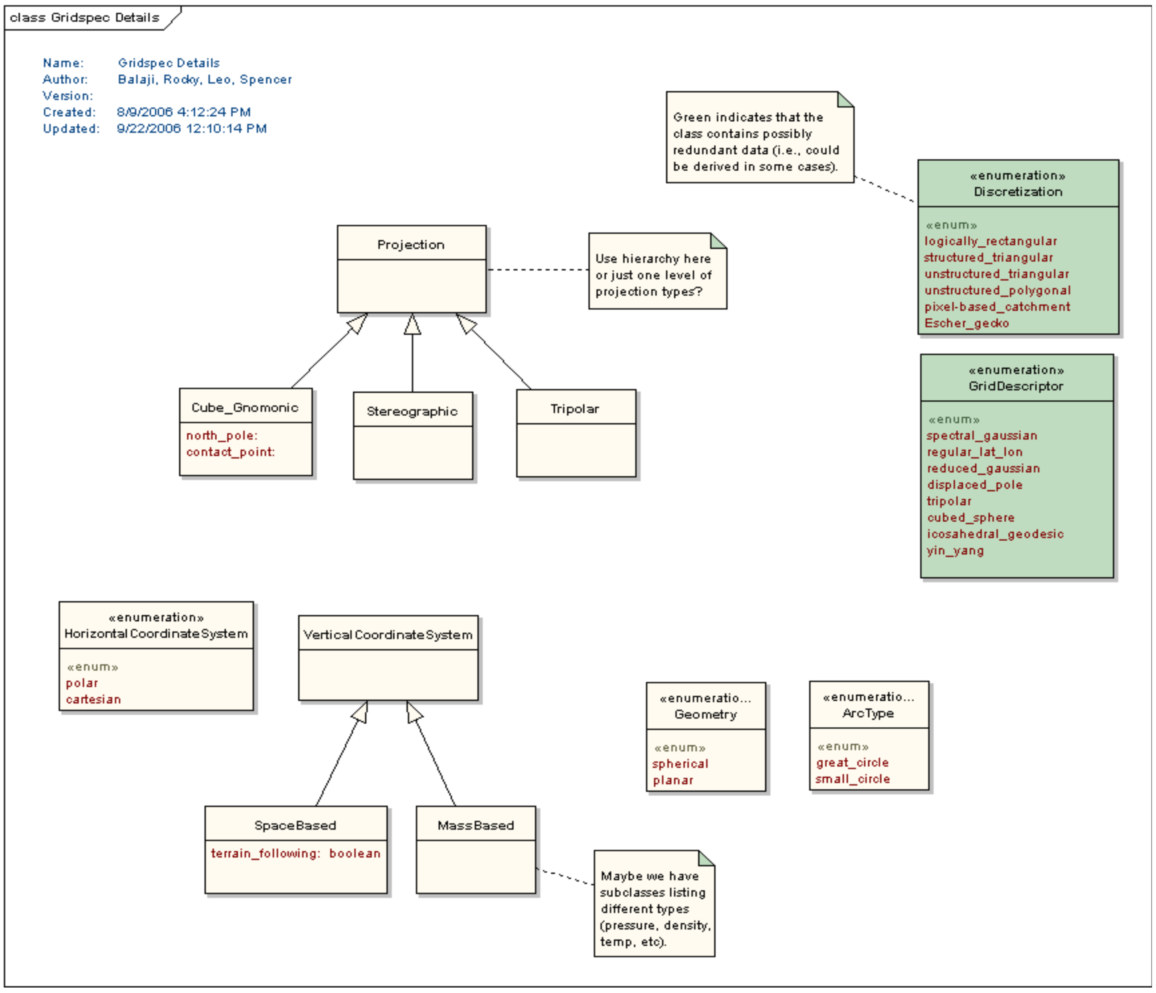

class Gridspec Details
Name: Gridspec Details
Author: Balaji, Rocky, Leo, Spencer
Version:
Created: 8/9/2006 4:12:24 PM
Updated: 9/22/2006 12:10:14 PM
Green indicates that the class contains possibly redundant data (i.e., could be derived in some cases).
«enumeration» Discretization
«enum» logically_rectangular structured_triangular unstructured_triangular unstructured_polygonal pixel-based_catchment Escher_gecko
Projection
Use hierarchy here or just one level of projection types?
Cube_Gnomonic
north_pole: contact_point:
Stereographic
Tripolar
«enumeration» GridDescriptor
«enum» spectral_gaussian regular_lat_lon reduced_gaussian displaced_pole tripolar cubed_sphere icosahedral_geodesic yin_yang
«enumeration» HorizontalCoordinateSystem
«enum» polar cartesian
VerticalCoordinateSystem
«enumeratio... Geometry
«enum» spherical planar
«enumeratio... ArcType
«enum» great_circle small_circle
SpaceBased
terrain_following: boolean
MassBased
Maybe we have subclasses listing different types (pressure, density, temp, etc).

Figure 21: Semantics of grid specification. This UML diagram shows a vocabulary for describing grid coordinate systems, projections and discretizations. The proposal is that this semantic content uses a controlled, yet extensible, vocabulary maintained by the CF convention committees.

components on independent grids. *Parallel Computational Fluid Dynamics: Theory and Applications, Proceedings of the 2005 International Conference on Parallel Computational Fluid Dynamics, May 24-27, College Park, MD, USA*, A. Deane, G. Brenner, A. Ecer, D. Emerson, J. McDonough, J. Periaux, N. Satofuka, , and D. Tromeur-Dervout, eds., Elsevier.

Berners-Lee, T., 1999: *Weaving the web*. Orion Business Books.

Berners-Lee, T. and J. Hendler, 2001: Publishing on the semantic web. *Nature*, **410**, 1023–1024.

Collins, N., G. Theurich, C. DeLuca, M. Suarez, A. Trayanov, V. Balaji, P. Li,

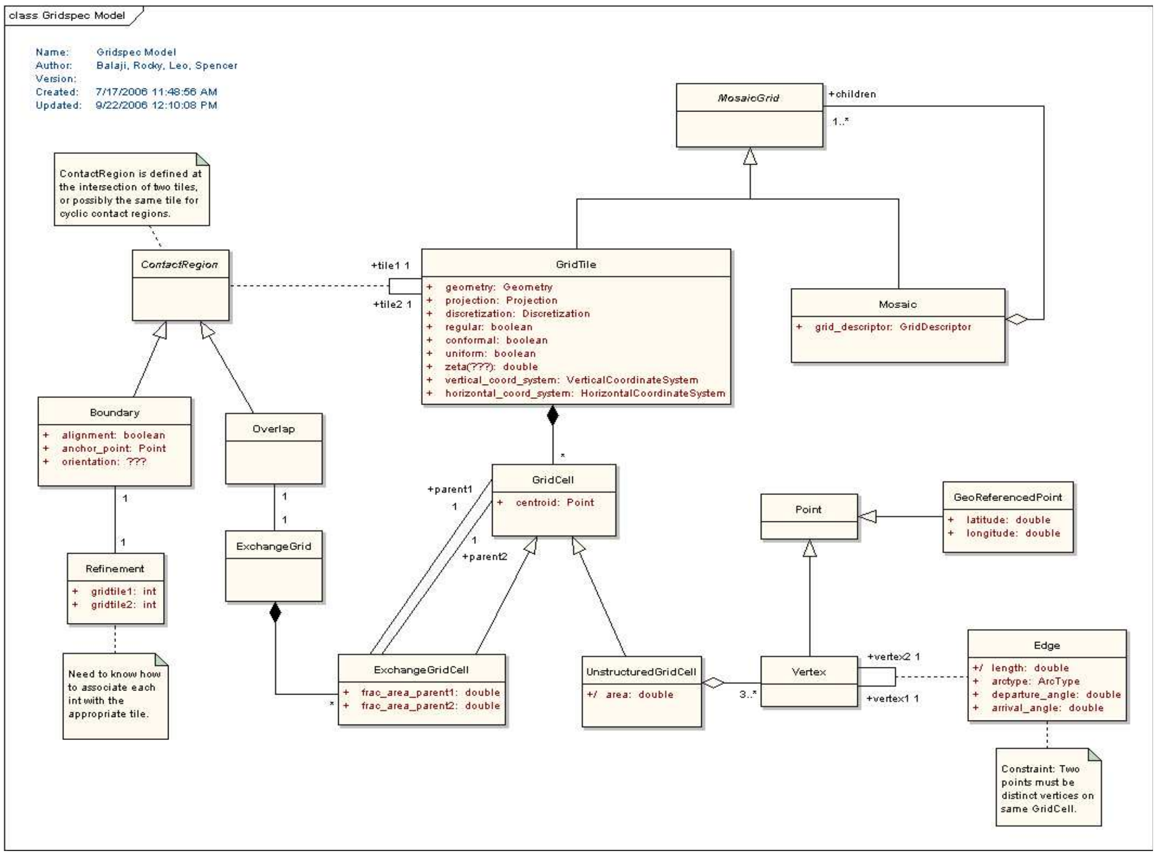


Figure 22: Abstraction of the grid specification. This UML diagram shows schematically how locations, distances, areas and volumes of grid cells are conceptually linked into a structure culminating in a grid mosaic.

W. Yang, C. Hill, and A. da Silva, 2005: Design and Implementation of Components in the Earth System Modeling Framework. *International Journal of HPC Applications*, **19**, 341–350.

Dahl, R., 1982: *The BFG*. Farrar, Strauss and Giroux.

Delworth, T., A. Broccoli, A. Rosati, R. Stouffer, V. Balaji, J. Beesley, W. Cooke, K. Dixon, J. Dunne, K. Dunne, J. Durachta, K. Findell, P. Ginoux, A. Gnanadesikan, C. Gordon, S. Griffies, R. Gudgel, M. Harrison, I. Held, R. Hemler, L. Horowitz, S. Klein, T. Knutson, P. Kushner, A. Langenhorst, H.-C. Lee, S.-J. Lin, J. Lu, S. Malyshev, P. D. Milly, V. Ramaswamy, J. Russell, M. Schwarzkopf, E. Shevliakova, J. Sirutis, M. Spelman, W. Stern, M. Winton, A. Wittenberg, B. Wyman, F. Zeng, and R. Zhang, 2006: GFDL's CM2 Global Coupled Climate Models. Part 1: Formulation and Simulation Characteristics. *Journal of Climate*, **19**, 643–674.

Guilyardi, E., 2006: El Niño: mean state-seasonal cycle interactions in a multi-model ensemble. *Climate Dynamics*, **26**, 329–348.

Hagedorn, R., F. J. Doblas-Reyes, and T. N. Palmer, 2005: The rationale behind the success of multi-model ensembles in seasonal forecasting: I. basic concept. *Tellus A*, **57**, 219–233.

Haltiner, G. J. and R. T. Williams, 1980: *Numerical Prediction and Dynamic Meteorology*. John Wiley and Sons, New York.

Hewitt, C. D. and D. J. Griggs, 2004: Ensembles-based prediction of climate changes and their impacts (ENSEMBLES). *Eos*, **85**, 566–567.

Hill, C., C. DeLuca, V. Balaji, M. Suarez, A. da Silva, and the ESMF Joint Specification Team, 2004: The Architecture of the Earth System Modeling Framework. *Computing in Sci. and Engg.*, **6**, 1–6.

Hortal, M. and A. Simmons, 1991: Use of reduced gaussian grids in spectral models. *Mon. Wea. Rev.*, **119**, 1057–1074.

Iskandarani, M., D. B. Haidvogel, J. C. Levin, E. Curchitser, and C. A. Edwards, 2002: Multi-scale geophysical modeling using the spectral element method. *Computing in Science & Engineering*, **4**, 42–48, doi:10.1109/MCISE.2002.1032428.

Jones, P. W., 1999: First- and second-order conservative remapping schemes for grids in spherical coordinates. *Mon. Wea. Rev.*, **127**, 2204–2210.

Jones, P. W., P. H. Worley, Y. Y. 3, J. B. White, and J. Levesque, 2005: Practical performance portability in the Parallel Ocean Program (POP). *Concurrency and Computation*, **17**, 1317–1327.

Kageyama, A., M. Kameyama, S. Fujihara, M. Yoshida, M. Hyodo, and Y. Tsuda, 2004: A 15.2 TFlops Simulation of Geodynamo on the Earth Simulator. *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*, 35.

Koster, R. D., M. J. Suarez, A. Ducharne, M. Stieglitz, and P. Kumar, 2000: A catchment-based approach to modeling land surface processes in a GCM, part I. Model Structure. *J. Geophys. Res.*, **105**, 24809–24822.

Kurihara, Y., M. A. Bender, R. E. Tuleya, , and R. J. Ross, 1990: Prediction experiments of hurricane gloria (1985) using a multiply nested movable mesh model. *Mon. Wea. Rev.*, **118**, 2185–2198.

Majewski, D., D. Liermann, P. Prohl, B. Ritter, M. Buchhold, T. Hanisch, G. Paul, W. Wergen, and J. Baumgardner, 2002: The operational global icosahedral-hexagonal gridpoint model gme: Description and high-resolution tests. *Mon. Wea. Rev.*, **130**, 319–338.

Murray, R. J., 1996: Explicit generation of orthogonal grids for ocean models. *J. Comp. Phys.*, **126**, 251–273.

Palmer, T. N., A. Alessandri, U. Andersen, P. Cantelaube, M. Davey, P. Délécluse, M. Dequé, E. Diez, F. J. Doblas-Reyes, H. Feddersen, R. Graham, S. Gualdi, J.-F. Gueremy, R. Hagedorn, M. Hoshen, N. Keenlyside, M. Latif, A. Lazar, E. Maisonnave, V. Marletto, A. P. Morse, B. Orfila, P. Rogel, J. M. Terres, and M. C. Thomson, 2004: Development of a European multimodel ensemble system for seasonal-to-interannual prediction (DEMETER). *Bull. Amer. Met. Soc.*, **85**, 853–.

Rancic, M., R. Purser, and F. Mesinger, 1996: A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates. *Quart. J. Roy. Meteor. Soc.*, **122**, 959 – 982.

Randall, D. A., T. D. Ringler, R. P. Heikes, P. W. Jones, and J. Baumgardner, 2002: Climate modeling with spherical geodesic grids. *Computing in Sci. and Engg.*, **4**, 32–41.

Rasch, P. J., 1994: Conservative shape-preserving two-dimensional transport on a spherical reduced grid. *Mon. Wea. Rev.*, **122**, 1337–1350.

Russell, J.-E., R. Stouffer, and K. Dixon, 2006: Intercomparison of the Southern Ocean Circulations in the IPCC Coupled Model Control Simulations. *J. Climate*, in press.

Thomas, S. R. and R. D. Loft, 2002: Semi-implicit spectral element atmospheric model. *J. Sci. Comp.*, **17**, 339–350.

van Oldenborgh, G. J., S. Y. Philip, and M. Collins, 2001: El Niño in a changing climate: a multi-model study. *Ocean Science*, **1**, 81–95.

Vecchi, G. A., B. J. Soden, A. T. Wittenberg, I. M. Held, A. Leetmaa, and M. J. Harrison, 2006: Weakening of tropical pacific atmospheric circulation due to anthropogenic forcing. *Nature*, **441**, 73–76.

Wilby, R. L. and T. M. L. Wigley, 1997: Downscaling general circulation model output: a review of methods and limitations. *Progress in physical geography*, **21**, 530–548.