Journal of Physics: Conference Series **78** (2007) 012072 doi:10.1088/1742-6596/78/1/012072

Three Dimensional Adaptive Mesh Refinement on a Spherical Shell for Atmospheric Models with Lagrangian Coordinates

Joyce E. Penner[1], Natalia Andronova[1], Robert C. Oehmke[3], Jonathan Brown[1], Quentin F. Stout[1], Christiane Jablonowski[1], Bram van Leer[1], Kenneth G. Powell[1], Michael Herzog[2]

[1]University of Michigan, Ann Arbor, MI; [2]Geophysical Fluid Dynamics Laboratory; [3]National Center for Atmospheric Research

1. Introduction

One of the most important advances needed in global climate models is the development of atmospheric General Circulation Models (GCMs) that can reliably treat convection. Such GCMs require high resolution in local convectively active regions, in both the horizontal and vertical directions. Our research is characterized by an interdisciplinary approach involving atmospheric science, computer science and mathematical/numerical aspects. The work is a close collaboration between the Atmospheric Science, Computer Science and Aerospace Engineering Departments at the University of Michigan. Collaboration with NOAA GFDL and NCAR has also been established.

One of the major building blocks of this project is a parallel adaptive grid library, which is currently under development at the University of Michigan. The library is named the AB library (adaptive block library) (*Oehmke and Stout*, 2001). This MPI-based communication library manages the block-structured data layout, handles ghost cell updates among neighboring blocks and splits a block as refinements occur. The current functionality provides automatic generation of a number of different grid topologies. The one which is used in the current model is a reduced grid. In a reduced grid, the resolution is coarsened in the longitudinal direction as the pole is approached to allow the use of a longer time step. We have developed an Adaptive Mesh Refinement (AMR) dynamical core that uses this library and which can adapt its grid resolution in the horizontal direction (*Jablonowski et al.,* 2006).

The AB library provides support for a particular type of adaptive mesh refinement (AMR) called adaptive blocks. Cells represent grid points, and a block is an array of cells. During adaptive refinement a block breaks into a set of blocks each with the same number of cells as the original, but which represent grid points that have been refined by a factor of two in each dimension. Figure 1 illustrates this process, where the heavily outlined cubes are blocks and the lighter outlined cubes are cells. The multi-celled self-similar nature of adaptive blocks provides a number of efficiencies described in section 3.

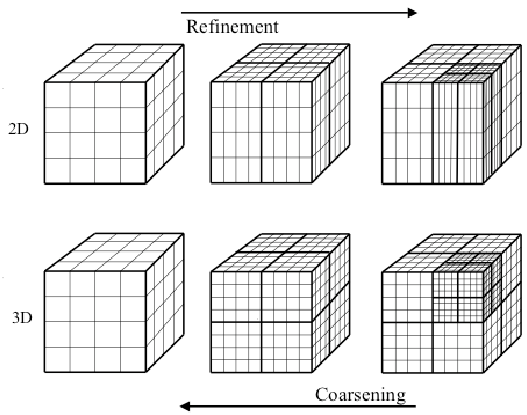

Figure 1 2D and 3D adaptive blocks

The AB library is the only tool that provides a general purpose, multishape implementation of adaptive blocks. Other tools, such as Samrai (*Homung and Kohn*, 1998) and Overture (*Bassetti et al.,* 1998), provide support for other types of adaptive mesh refinement. Perhaps the closest tool to ours is PARAMESH (*MacNeice et al.,* 2000), which also provides adaptive blocks, but not specific support for non-Cartesian topological features such as poles.

The numerical representation of the partial differential equations (PDE) for fine atmospheric dynamical and physical features requires division of the atmospheric volume into

a set of 3D grids, each of which has a not quite rectangular form. Each location on the grid contains data that represents the state of Earth's atmosphere. For successful numerical integration of PDEs the size of each grid box is used to determine the time step needed to satisfy the Courant-Friedrich-Levi (CFL) criterion. 3D adaptive representations of Cartesian space are needed to represent the evolution of clouds and thus require the adaptive block (AB) system with a 3D adaptive representation of a spherical volume.

2. Constructing 3D adaptive spheres

During this project we extended the AB system to enable it to easily generate spherical grids with special functions for manipulating blocks next to a pole and for handling data crossing a pole. Using the AB library the user is able to specify various geometric parameters (e.g., center, radius and the connection properties of the sphere). The sphere module is responsible for creating the sphere-shaped connected group of blocks and for maintaining the geometric information describing the sphere. This module calls for the block module to connect the blocks into a sphere and for the load balancing module to distribute its new blocks. During the project we had a few challenges to resolve, which we list below.

2.1. Complexity

The transition from the 2D adaptive blocks used previously to 3D adaptive blocks increased dramatically the number and types of connections. The three types of connection per element are: the connection between a block and the same level neighbor; the connection between a block and a more refined neighbor; and the connection between a block and a coarser neighbor. These correspond to a two-, one-, or zero-dimensional surface between the block and its neighbor. A 2D block restricted to one level of refinement between neighbors has 12 (8 edge + 4 corners) potential neighbors with 6 (3 edge + 3 corner) types of connection. A 3D block with the same refinement restriction has 56 (24 face + 24 edge + 8 corner) potential neighbors with 9 (3 face + 3 edge + 3 corner) types of connections. These additional connections and types come with an attendant increase in code complexity and size, and an increase in the number of individual cases.

To manage this type of complexity the AB library has been built as a set of modules each managing a particular piece of functionality, reducing the complexity that the developer has to deal with at any one time. The adaptation modules are also constructed so that each type of change to a block is handled separately and transparently to processor location, allowing the developer to concentrate on one connection interaction at a time. The complexity of the 3D blocks points out the necessity of a library such as ours. It serves to isolate the scientific model developer from having to deal with this level of detail and allows them to concentrate on implementation of the numerical aspects and physical processes instead.

2.2. Sphericity

Handling the poles of a sphere is a differentiating feature of a spherical adaptive block structure over a Cartesian one. The pole point presents specific difficulties. The first is that ghost cell data crossing the pole does not behave the same as other ghost cell transfers. This is because connections across the pole reverse the index order in the north-south direction. Thus, these connections need to be marked as special and handled differently. Also, the user will often want to process the data crossing the pole differently than other data.

To handle the differences in the pole connectivity, the library provides a mechanism to signal the user when a particular communication is going across the pole. This allows the user to process their data appropriately. Note that this connection type signaling interface is general

enough to handle other types of connections, and could be useful, for example, when signaling communication across a cubed-sphere corner or edge. The library also provides a way for the user to access just the blocks surrounding the poles should they need to do specific processing of the data there.

3. Implementing 3D adaptive spheres

The library has several modules that provide a layer of abstraction for adaptive refinement: blocks, which contain individual cells of user data; shells – the global geometry for the problem, including a sphere, reduced sphere, and now a 3D sphere; a load balancer for placement of blocks onto processors; and a communication support layer which encapsulates all data movement. Users provide data manipulation functions for performing interpolation of user data when refining or coarsening blocks.

There are a number of serial efficiencies provided by the library. The first of these are just based on the use of adaptive blocks. The fixed size blocks allow for efficient constant-sized loops to be used for calculations, and the block size can be tuned for good cache performance. Another efficiency is that the library allows users to arrange their data for the most efficient access during calculations. There is no additional cost for retrieving data from the library.

There are also a number of efficiencies related to the parallel aspects of the library. The use of block arrays, instead of single cells, reduces the total number of ghost cells that must be communicated. Further, having the same number of cells per block allows for easy load balancing. Some other steps taken for parallel efficiency are: all messages between processors are aggregated into one message to reduce communication startup costs; all intra-processor communication is handled wherever possible by direct memory copies avoiding the extra memory and time taken by buffers; all communication is non-blocking to reduce synchronization costs.

The 3D sphere maintains the iterator interface for individual blocks and data points used throughout the library. This allows the user code to be written with tight loops over the blocks on each processor, irrespective of the refinement or coarsening steps taken, thus allowing for code that can be highly optimized by the compiler. The library maintains the placement of blocks and assignment to processors through refinement or coarsening steps, making the communication transparent to the user.

The 3D sphere differs from the existing sphere model in that it has three types of neighbors - faces, edges, and corners - with each block having neighboring directions with which to communicate ghost cell information; without refinement in the z-direction for altitude, the previous sphere module had only two kinds of neighbors, and at most eight such neighbor directions. The ghost cell information is commensurately greater per block in the 3D sphere. This added complexity is encapsulated in the 3D sphere module, so that a user will be able to use the AB library's 3D sphere module in much the same way as its current sphere module: create a block group as a set of geometry-independent data cells; create the 3D sphere object, passing it the block group as an argument to map data cells to locations in the spherical shell; then, refine or coarsen blocks as needed. This transparency of interface was a design goal: the 3D sphere should be nearly a drop-in replacement for the 2D sphere in the simulation code, and can co-exist with the 2D sphere in the same model.

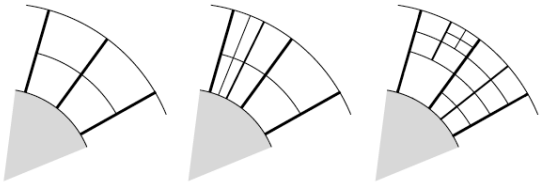

Figure 2 Adaptation in horizontal, and mixed horizontal and vertical

During the past year, we extended the library to treat adaptive mesh refinement in the vertical direction (see Figure 2). For this we have added:

1) support for 3D blocks (new connections; new structures; new methods in the library to support 3D in addition to 2D);

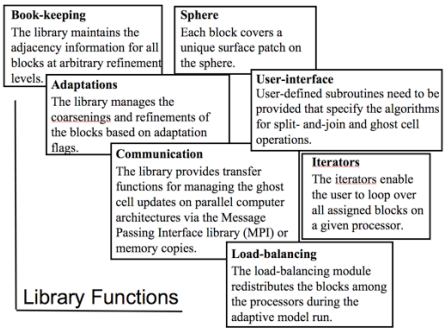

Figure 3 AB library

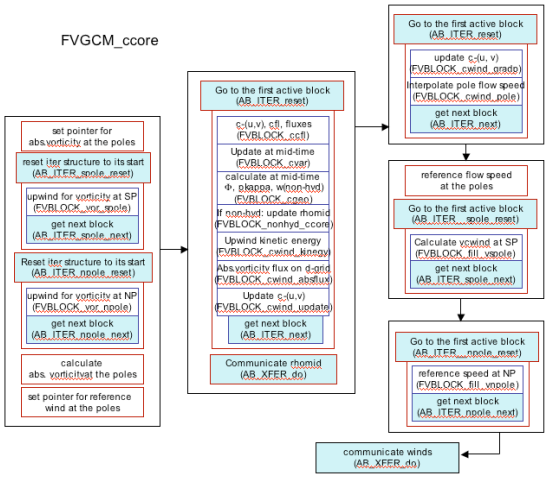

Figure 4 Using the AB Library

2) a new module to generate a 3D sphere using newly added 3D blocks (constructed a spherical shell shaped collection of 3D blocks);
3) functionality to do ghost cell transfers with the new 3D sphere (at this point the library is usable as a 3D spherical grid in models, but without adaptation.)

Figure 3 shows the structure of the modules provided, and Figure 4 illustrates how they are being used in a portion of the code. Blue boxes denote calls to the AB library, including both iterators and communication.

4. Testing the Dynamical Core

Solving differential equations consists of two task levels. The first is tailoring the mathematical description to a particular problem, and the second is its numerical approximation. The product of combining both tasks is a numerical model of a phenomenon. Since our final goal is a rigorous representation of cloud evolution in global atmospheric models, our efforts are directed toward accommodation of small spatial and temporal scales in our 3D Lagrangian Finite-Volume Spherical Global Model. Accommodation of our AB library in the model, as well as utilization of different techniques for defining the model accuracy and stability requires designing tests for different aspects of the model's performance. In previous years we completed testing the model for adaptation in 2D (longitude-latitude). This year we concentrated on finding tests to be implemented with utilization of our 3D AB library. The main difference between the reference cases for the tests and our model implementation is the floating vertical Lagrangian coordinate, which requires re-mapping of the model's variables at each time step.

4.1 The Gravity Wave Test

This test (*Skamaroch and Kemp, 1994*) is designed to test the horizontal dispersion of a small potential temperature disturbance in a stably stratified atmosphere (see Figure 5). It is a 2D test (height vs. one horizontal dimension), however it provides reference solutions for both the hydrostatic and non-hydrostatic versions of the model. This year we implemented the test for the hydrostatic case. The model gives the right direction, speed and expected symmetry for the propagation of gravity waves. However, it also shows the influence of the model's reflective top.

4.2. The Pure Advection Test

This test (*Zubov, et al. 1999*) is designed to test the model's vertical transport of a passive tracer with no atmospheric sources and sinks. It is both a 2 and a 3D test, and bridges the performance of the model's dynamics and physics needed for inclusion of cloud formation. As implemented in the model the tracer moves through the model's domain too fast and we are investigating the reasons for this (see Figure 6).

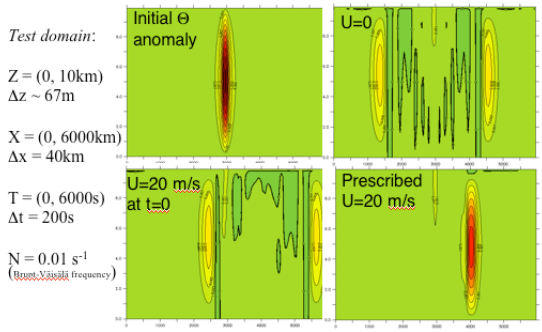

Figure 5 Gravity wave test

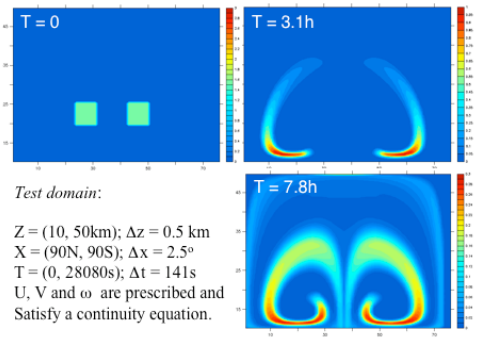

Figure 6 Pure advection of a tracer

5. Concluding Remarks

This work-in-progress is developing an atmospheric model capable of resolving local regions of convective activity within a global GCM. It does this by using adaptive meshes to focus computational resources on the local regions, as opposed to the computationally infeasible approach of using a fine grid globally. This approach requires suitable numerical algorithms to deal with the challenges of uneven gridding, and data structures capable of supporting this on large parallel computers. Both of these aspects have been addressed, and the initial results are encouraging.

References:

Bassetti, P., et al., 1998: Overture: an Object-oriented Framework for High Performance Scientific Computing, *Proceedings of ACM/IEEE SC98*, 9p.

Hornung, R., and Kohn, S., 1998: The Use of Object-Oriented Design Patterns in the SAMRAI Structured AMR Framework, *Proc. SIAM Workshop on Object Oriented Methods for Inter-Operable Scientific and Engineering Computing* Philadelphia, PA.

Jablonowski, C., M. Herzog, J. E. Penner, R. C. Oehmke, Q. F. Stout, B. van Leer and K. G. Powell, 2006: Block-Structured Adaptive Grids on the Sphere: Advection Experiments, *Mon. Wea. Rev.* 134, pp. 3691–3713, DOI: 10.1175/MWR3223.1.

MacNeice, P., et al., 2000: PARAMESH: a Parallel Adaptive Mesh Refinement Community Toolkit, *Computer Physics Communications* 126, pp. 330-354.

Oehmke, R., and Q.F. Stout, 2001, Parallel Adaptive Blocks on a Sphere, *Proc. SIAM Conf. Parallel Proc. for Scientific Computing.*

Skamarock and Klemp, 1994: Efficiency and Accuracy of the Klemp-Wilhelmson Time-Splitting Technique, *Mon. Wea. Rev.,* 122, 2623-2630.

Zubov, V. A., E. V. Rozanov and M. E. Schlesinger, 1999: Hybrid Scheme for Three-Dimensional Advective Transport, *Mon. Wea. Rev,* 127, 1335-1346.