CLIMATE MODELING WITH SPHERICAL GEODESIC GRIDS

A new approach to climate simulation uses geodesic grids generated from an icosahedron and could become an attractive alternative to current models.

Most scientists believe that the increasing concentrations of carbon dioxide and greenhouse gases from fossil fuel combustion and other human activities dramatically affect the Earth's climate, but exactly what changes in climate can we expect in the coming years? What will the impact be, for example, on agriculture, unmanaged ecosystems, and sea levels? Which regions of the world will be most affected? We can't learn the answers to these questions in a lab setting, thus computer-based climate simulations are essential to understanding and forecasting potential climate changes.

Climate simulation has come a long way since the 1970s when the US Department of Energy first started sponsoring research in the field. However, one thing hasn't changed: the climate-modeling community still needs the capability to quickly run multiple simulations using ever more detailed and accurate coupled models.

1521-9615/02/$17.00

DAVID A. RANDALL, TODD D. RINGLER, AND ROSS P. HEIKES
Colorado State University
PHIL JONES AND JOHN BAUMGARDNER
Los Alamos National Laboratory

We have implemented an atmospheric general circulation model (AGCM) using a geodesic discretization of the sphere (see Figure 1). Our model uses the message-passing interface and runs efficiently on massively parallel machines. We wanted to design and construct an architecturally unified climate-modeling framework based on spherical geodesic grids. This framework lends itself to the creation of a comprehensive, conservative, accurate, portable, and highly scalable coupled climate model. We believe that this approach is the future of climate modeling.

Models and methods

Over the past several decades, meteorologists and oceanographers have developed various methods for solving the equations that describe fluid flow on a sphere. Such equations are used in both AGCMs and ocean general circulation models (OGCMs).

Some AGCMs use finite-difference methods, in which the ratios of differences approximate the derivatives that appear in the governing differential equations. The so-called pole problem arises when we apply finite-difference methods on latitude–longitude grids, in which the meridians (lines of constant longitude) converge to points at the two poles (see Figure 2). Small

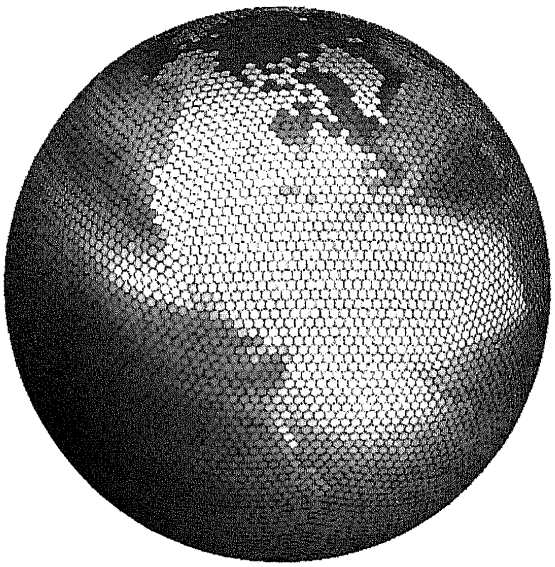

Figure 1. An example of a geodesic grid with a color-coded plot of the observed sea-surface temperature distribution. The continents are depicted in white. This grid has 10,242 cells, each of which is roughly 240 km across. Twelve of the cells are pentagons; the rest are hexagons.

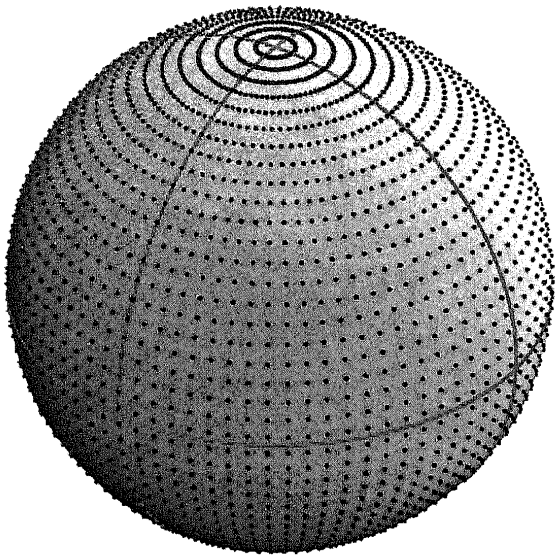

Figure 2. An example of a latitude–longitude grid. A pole is at the top. The black dots represent grid cell centers that are equally spaced in longitude (west to east) and latitude (south to north). The red lines are the Equator and two lines of constant longitude.

time steps could be used, at great expense, to maintain computational stability near the poles, but the high resolution in the east–west direction near them would be wasted because the model uses lower resolution elsewhere. Solutions to the pole problem are sometimes based on longitudinal filtering, but these methods have some unfortunate side effects and scale badly as model resolution increases.

OGCMs also use finite-difference methods.[1] In the Parallel Ocean Program (POP),[2] stretching the spherical coordinates so that the coordinate system's north pole is on a landmass (for example, Greenland) rather than in the Arctic Ocean eliminates the pole problem. Although this ad hoc approach alleviates the problem in the OGCM, the peculiarities of the stretched ocean model grid further complicate the coupling of the atmosphere, ocean, and land surface submodels. Moreover, the stretched grid is tailored for the particular arrangement of the continents and oceans on Earth and thus lacks generality.

Spectral AGCMs are based on spherical harmonic expansions.[3] This approach also avoids the pole problem but has problems with almost-discontinuous functions such as steep terrain and the sharp boundaries of clouds. Moreover, spectral models poorly simulate the transports of highly variable nonnegative scalars, such as the concentration of moisture.[4] This weakness has caused most spectral AGCMs to evolve into hybrid models in which grids represent most or all of the advective processes.

The spectral method (which is based on spherical harmonics) is not applicable to ocean modeling because of the ocean basins' complicated geometry. Researchers are exploring spectral elements based on local expansions, generally of high-order accuracy, as an alternative approach and for possible use in atmosphere modeling.[5]

We found that the best way to avoid the pole problem is to use quasi-uniform spherical geodesic grids—tessellations of the sphere generated from icosahedra or other Platonic solids.[6] Icosahedral grids, first tried in the 1960s, give almost homogeneous and quasi-isotropic coverage of the sphere.[7,8] Modern numerical algorithms including fast elliptic solvers and mimetic discretization schemes have made these grids more attractive, renewing interest in the idea.[9,10]

Grid generation

To construct an icosahedral grid, we begin with an icosahedron (see Figure 3), which has triangular faces and vertices. Euler's famous theorem tells us that the number of edges is

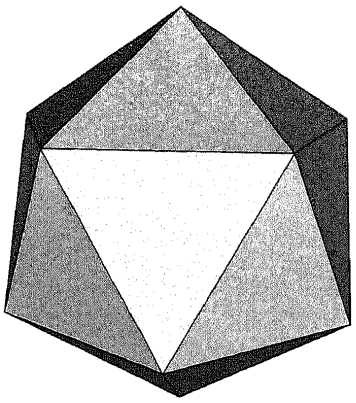

Figure 3. An icosahedron, which has 20 triangular faces, 12 vertices, and 30 edges.

$$E = F + V - 2,$$
so $E = 30$.

Our next step is to construct Voronoi cells centered on the 12 vertices. A Voronoi cell consists of the set of all points that are closer to a given vertex than to any other vertex. The Voronoi cells based on the icosahedron give us a discretization of the sphere that consists of 12 pentagonal faces. However, 12 faces do not give us enough resolution to represent the Earth system's complicated processes; we need finer grids.

As Figure 4 shows, we can halve the linear cell dimensions of each of the icosahedron's triangular faces and produce a finer grid. Bisecting the edges divides a triangular face into four smaller triangles and increases the number of faces by a factor of four. The bisection points become new vertices, so the number of double vertices increases by the number of edges. The new vertices are popped out onto the sphere (see Figure 4), and the bisection process is repeated as needed to generate arbitrarily fine grids. We can generalize the grid-generation algorithm outlined earlier to allow a broad variety of possible horizontal resolutions.

The geodesic grid's most obvious advantage is that all grid cells are nearly the same size. Variational tweaking of the grid can improve the mesh's uniformity (see Table 1; the shaded cells represent different resolutions that we used for climate simulation). The number of cells N_C satisfies $N_C = 5 \times 2^{2n+1} + 2$, where $n \geq 0$ is the "counter" listed in the table's first column. Because of the uniform cell size, computational stability for advection is not an issue, even with conservative finite-volume schemes. For example, with a grid cell 100 km across and a wind speed of 100 ms^{-1}, the allowed time step is 1,000 seconds. Time steps much longer than this would be incompatible with accurate simulations of the day–night cycle and fast adjustment processes (such as thunderstorms), which have short intrinsic timescales.

Even with a quasi-homogeneous grid, rapidly propagating waves can still significantly limit the time step. The fastest waves in our AGCM travel at about 300 ms^{-1}. Semi-implicit time differencing permits a 1,000-second time step without sacrificing the accurate simulation of large-scale atmospheric circulation.

This semi-implicit method creates an elliptic problem that must be solved at each time step. Having an efficient solution to minimize the computational overhead is also important. One version of our geodesic AGCM uses a parallel multigrid method to implement a semi-implicit time-differencing scheme.

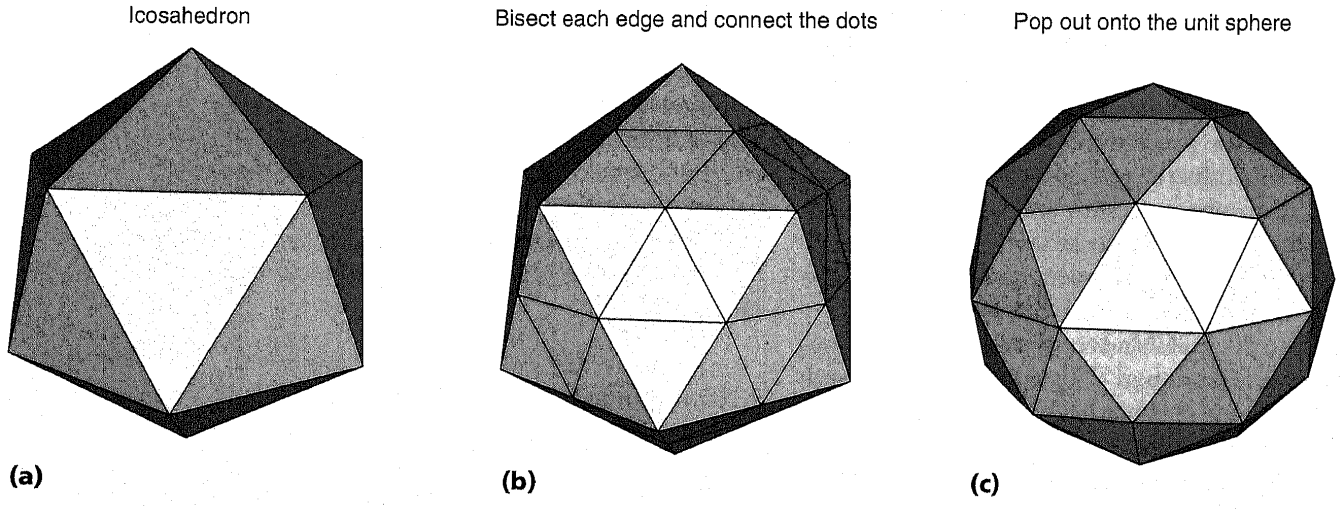

Figure 4. The first bisection of the icosahedron.

Table 1. A summary of the properties of geodesic grids of several different resolutions.

n	Number of cells	Number of cells along Equator	Average cell area in km^2	Area ratio (smallest to largest)	Average distance between cell centers in km	Ratio of smallest to largest distance between cell centers
0	12	0	4.25e7	1	3717.4	1
1	42	10	1.21e7	0.885	3717.4	0.881
2	162	20	3.14e6	0.916	1909.5	0.820
3	642	40	7.94e5	0.942	961.6	0.799
4	2,562	80	1.99e5	0.948	481.6	0.790
5	10,242	160	4.98e4	0.951	240.9	0.789
6	40,962	320	1.24e4	0.952	120.5	0.788

In all ocean models, realizing efficient time integration requires splitting the fast barotropic dynamics from the slower baroclinic dynamics. In all ocean models, realizing efficient time integration requires splitting the rapidly propagating deep waves from the slowly propagating shallower waves. Our ocean model uses a simple subcycling approach to advance the faster barotropic dynamics and a second-order-accurate staggered time-stepping scheme for the slower baroclinic modes. Within this coming year, we plan to explore the reduced gravity approach,[11] which permits long time steps without subcycling, solving an elliptic problem, or creating significant communication overhead. Our goal is to use a time step of 20 to 30 minutes for both the AGCM and OGCM.

Geodesic grids are quasi isotropic. Only three regular polygons tile the plane: equilateral triangles, squares, and hexagons. Figure 5 shows planar grids made up of each of these three possible polygonal elements. On the triangular and square grids, some of a given cell's neighbors lie directly across cell walls whereas others lie across cell vertices. As a result, finite-difference operators constructed on these grids tend to use *wall neighbors* and *vertex neighbors* in different ways. For example, the simplest second-order finite-difference approximation to the gradient, on a square grid, uses only wall neighbors; vertex neighbors are ignored. Although constructing finite-difference operators on square (and triangular) grids using the information from all neighboring cells is possible, these grids' essential asymmetries remain unavoidably manifested in the forms of the finite-difference operators. In contrast, hexagonal grids have the property that all of a given cell's neighbors lie across cell walls; no vertex neighbors exist. Therefore, finite-difference operators constructed on hexagonal grids treat all neighboring cells in the same way, making the operators as symmetrical and isotropic as possible. A geodesic grid on a sphere has 12 pentagonal cells plus many hexagonal cells, yet each geodesic grid cell has only wall neighbors; no vertex neighbors exist anywhere on the sphere.

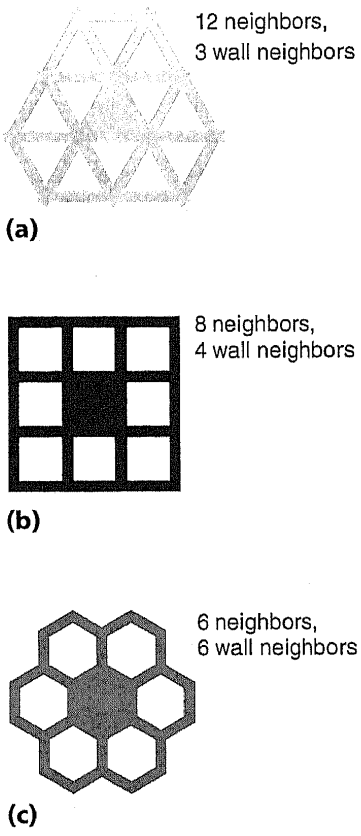

Figure 5. Small grids made up of (a) equilateral triangles, (b) squares, and (c) hexagons. These are the only regular polygons that tile the plane. The hexagonal grid has the highest symmetry. All neighboring cells of a given hexagonal cell are located across cell walls: with either triangles or squares, some neighbors are across walls and other are across corners.

We can cut a geodesic grid into 10 rectangular panels, each corresponding to a pair of triangular faces from the original icosahedron. Figure 6 shows the 10 panels pared to make five logically rectangular panels consisting of rows and columns that form a rectangular array. As the figure shows, two leftover cells correspond to the points where the panels meet at the two poles,

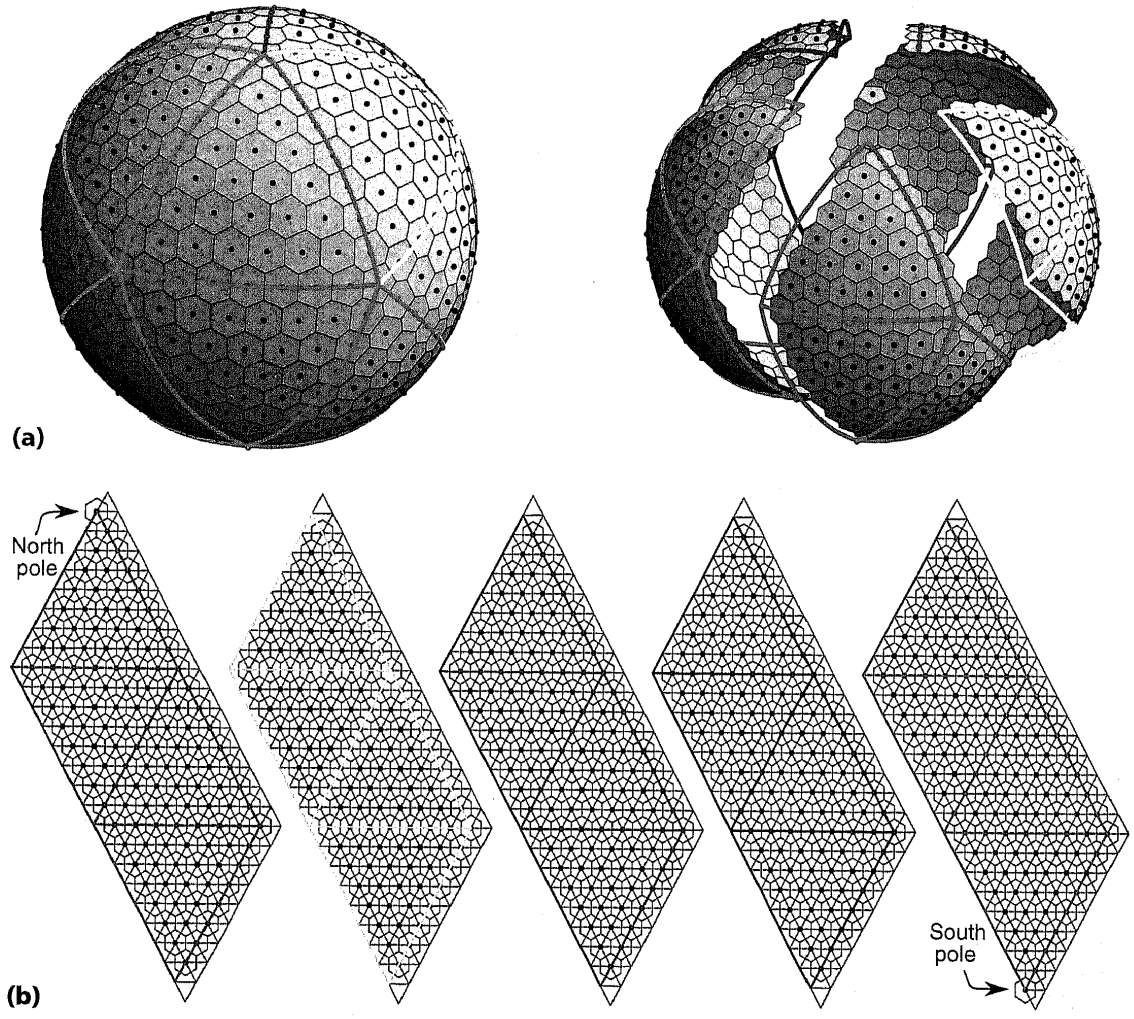

Figure 6. We can cut (a) a spherical geodesic grid into (b) logically rectangular panels, which offers a convenient way to organize the data in a computer's memory.

and both of these points are pentagons. The rectangular panels in the figure correspond to the logically rectangular organization of the gridded data in the computer's memory.

We recognize the need to make the geodesic grid easy and convenient for others to work with. For this purpose, we have created "canned" routines that can

- Generate geodesic grids
- Interpolate data from conventional latitude–longitude grids onto geodesic grids
- Plot data that is defined on geodesic grids

These routines are intended to help newcomers adopt geodesic grids, for both modeling and data analysis (see http://kiwi.atmos.colostate.edu/BUGS/projects/geodesic).

Although we already have an AGCM based on a geodesic grid, we are still developing an OGCM that can use a similar grid of higher resolution. To our knowledge, this will be the first OGCM based on a spherical geodesic grid.

Coding strategy

Currently, the codes we have developed are being used in an atmospheric model. We plan to simulate the fluid motions of both the atmosphere and ocean using the same source-code modules. Some of these modules could also help model sea ice dynamics. An advantage of using similar discretizations across all components of a coupled model is that the various model components can share software components. We plan to implement our geodesic climate model using a layered architecture in which common utilities and data types are pushed to lower layers. Com-

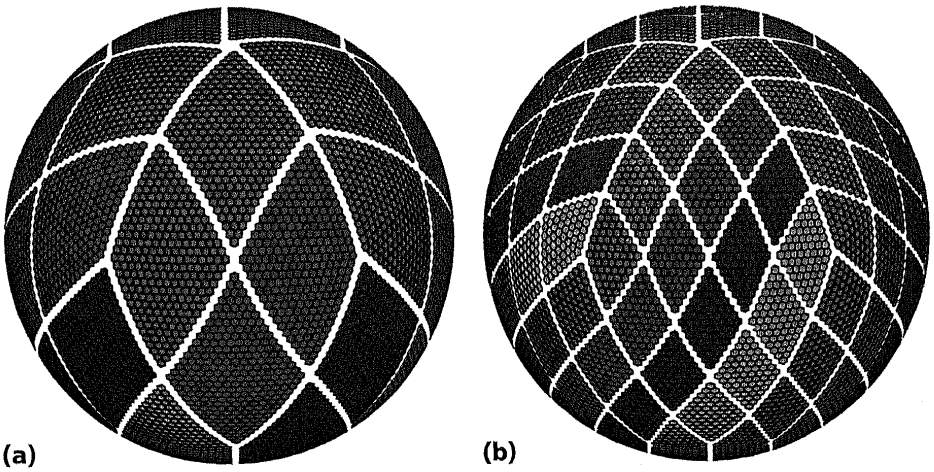

Figure 7. A 2D field decomposed into (a) 40 logically square blocks of data and (b) 160 logically square blocks of data.

ponent models comprise the upper layer and are built using shared code from the lower layers, which consist of low-level utilities and machine-specific code (for example, timers and communications primitives). The next layer contains higher-level shared components including modules to support the common horizontal discretization, common solvers, higher-level data motion abstractions, and model-coupling utilities. Layers above this include the component models and the full coupled climate model itself.

Such a layered approach requires well-defined interfaces to perform stand-alone testing of individual utilities and modules. Moreover, a well-defined interface lets us use standard utilities developed through other efforts. Using shared modules for solvers and horizontal differencing will permit performance optimization of well-defined kernels, thus improving all models and reducing workload.

Current computer architectures are typically networked clusters of nodes consisting of moderate numbers of cache-based microprocessors. Due to hardware constraints, it's often best to use a message-passing paradigm for the cluster level and a shared memory paradigm (for example, OpenMP or other thread-based approaches) among the processors on a node. Typically, these levels of parallelism should occur at a very high level in the code to provide enough work to amortize any overhead. Recent industry developments motivated us to maintain support for vector supercomputers. We believe that the data structure described later gives us the needed flexibility.

We can represent a 2D field on the spherical geodesic grid by using a data decomposition that consists of a collection of logically rectangular 2D arrays. Figure 7 depicts two examples of such data structures. In Figure 7a, we decompose the grid into 40 blocks; Figure 7b uses 160. We can represent 2D fields as arrays of dimension (*ni*, *nj*, *nsdm*), where *ni* and *nj* are the lengths of a block's sides, and *nsdm* is the total number of blocks. At present, we use square blocks of data, for which ($ni = nj$). This limits *nsdm* to the values of 10, 40, 160, 640, and so on. If the need arises, we can generalize the algorithm to accommodate rectangular blocks of data for which $ni \neq nj$.

We can distribute the *nsdm* blocks over *nsdm* processes by assigning one block of data to one process. Alternatively, we can use fewer than *nsdm* processes and assign multiple blocks of data to each process. Each process sees a data structure of size (*ni*, *ni*, *psdm*), where the value of *psdm* generally varies from process to process, and the sum of *psdm* across all processes equals *nsdm*. The generalization from 2D to 3D fields is trivial; each process sees a data decomposition of size (*ni*, *ni*, *nk*, *psdm*), where *nk* is the number of vertical levels.

This domain decomposition strategy has many advantages. First, the algorithm is equally applicable to atmospheric and ocean modeling; in fact, the Ocean Modeling Group at Los Alamos National Laboratory has independently developed and implemented the same domain decomposition strategy.[12]

Second, this approach limits the number of messages and the total size of the messages required to update data at each block's boundaries. This makes the algorithm somewhat insensitive to the particular type of interconnect fabric used.

Third, we built a load-balancing mechanism into the algorithm that lets each process own an arbitrary number of data blocks. Overloaded processes can give one or more blocks of data to underused processes. Currently, we load balance

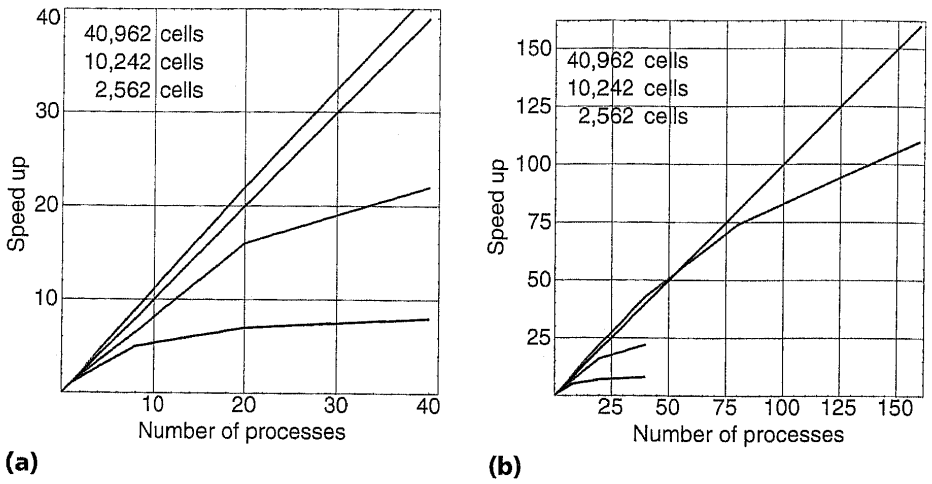

Figure 8. The current atmospheric dynamical core's scaling efficiency. Both panels show the same data: (a) the scaling out to 40 processes; (b) extending the *x*-axis to 160 processes. We performed these tests on an IBM SP-2.

at compile time. In the future, we intend to explore dynamic load balancing. Some blocks of data will not contain any ocean points, so we can cull these blocks to increase the algorithm's efficiency.

Fourth, we can adjust block size to adapt to a given machine's memory hierarchy. For cache-based microprocessors, we can choose the block size to be small enough to remain cache resident for most of the time step; for vector machines, we can choose the block size to be as large as possible for better vector efficiency.

Finally, our approach allows the seamless use of multitasking threads (such as OpenMP) under the message-passing interface library. Because every process owns *psdm* blocks of data, and each of these blocks can be manipulated independently of all other blocks of data, we can multitask over the *psdm* data's index. Threads are spawned at the beginning of each time step and are not closed until near the end of the time step. This implies that each thread is open for a relatively long time, which mitigates the cost of spawning the thread. The multitasking capability lets us best exploit architectures that use both shared and distributed memory paradigms.

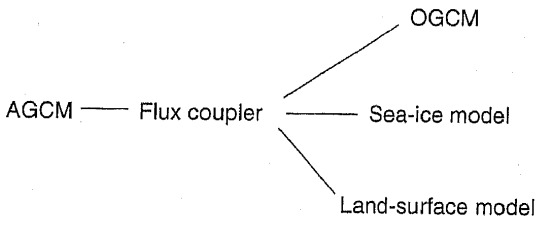

Figure 9. A flux coupler handles communications between the atmosphere and the underlying surface, which includes land, ocean, and sea ice. The flux coupler must take into account differences between the grids used for the atmosphere and the surface.

Figure 8 shows the geophysical fluid dynamics algorithm's scaling efficiency as implemented into our atmospheric dynamical core. When we discretize the sphere by using 2,562 grid cells, the nominal distance between grid cell centers is approximately 450 km; the 10,242 and 40,962 grids have grid spacings of approximately 225 km and 112 km, respectively. As Figure 8 shows, the model using the finest resolution considered here—40,962 cells—scales linearly out to approximately 80 processes and shows a speed up of 110 when we use 160 processes. The superlinear speed up of the model using the 40,962 grid at around 20 processes is due to data caching.

Flux coupling

Climate models must include representations of the atmosphere, ocean, sea ice, and land surface. For reasons having mainly to do with ocean dynamics, the grid used to represent the ocean is usually finer than the one used to represent the atmosphere. Important exchanges of energy, mass, and momentum exist across the Earth's surface—for example, water evaporates from the ocean and appears as water vapor in the atmosphere. In a climate model, an interface routine called a *flux coupler* computes these exchanges (see Figure 9).

In an earlier stage of our work, we used a latitude–longitude finite-difference model coupled with the stretched POP grid mentioned earlier. We then revised our atmospheric model to use the geodesic grid, still with the POP grid for the ocean. We designed a flux coupler that let the curvilinear POP grid communicate with the geodesic AGCM grid accurately and conserva-

tively. Figure 10 summarizes the grid overlap that the flux coupler must handle. The POP curvilinear grid represents the ocean, sea ice, land surface, and land ice submodels. We represent the atmosphere using the spherical geodesic grid. Surface fluxes—for example, evaporation from the ocean surface—are computed on the POP grid. To do this, we must interpolate the atmospheric state variables from the geodesic atmosphere grid to the surface grid and average the computed surface fluxes from the surface grid to the atmosphere grid. We performed these interpolation and averaging steps by using the methods Philip W. Jones developed.[13]

An attraction of using geodesic grids for both the atmosphere and the ocean models is that they greatly simplify the design of an efficient, parallel flux coupler. To ensure satisfactory coupling of the ocean and the atmosphere, we must compute the air–sea fluxes due to turbulence at the ocean model's higher resolution. We must interpolate the various atmospheric variables needed to compute these fluxes from the atmosphere model's coarse grid to the ocean model's fine one. We must then average the fluxes back to the coarser atmosphere grid, in such a way that the globally averaged surface flux is the same on both grids. Moreover, we must interpolate atmospheric fluxes due to precipitation and radiation computed on the atmosphere model's coarse grid to the ocean model's fine grid in such a way that the globally averaged surface flux is the same on both. Similar comments apply to communications between the atmosphere and land surface models. The computational machinery needed to perform these various steps is encapsulated in a flux coupler, which we have designed to facilitate communications and interactions among the various submodels.

In our geodesic climate model, the surface and atmosphere grids are both spherical geodesic grids, albeit with different resolutions. The ocean, sea ice, land surface, and land ice submodels share a single geodesic surface grid.

Each process owns one or more (generally more) atmosphere grid blocks and one or more corresponding surface grid blocks. Atmosphere and surface blocks nearly coincide—that is, one nearly congruent surface block underlies each atmosphere block. The number of grid cells in a surface block is normally much larger than the number of grid cells in the atmosphere block above it because the surface grid is normally finer than the atmosphere grid.

In our model, each process owns N atmosphere points and M surface points, where N and M are the same for all processes. In most if not all existing coupled models, this is virtually impossible to arrange because the atmosphere and ocean grids completely differ from each other. In our geodesic climate model, we can make N and M the same for all processors because the atmosphere and ocean grids have the same shape.

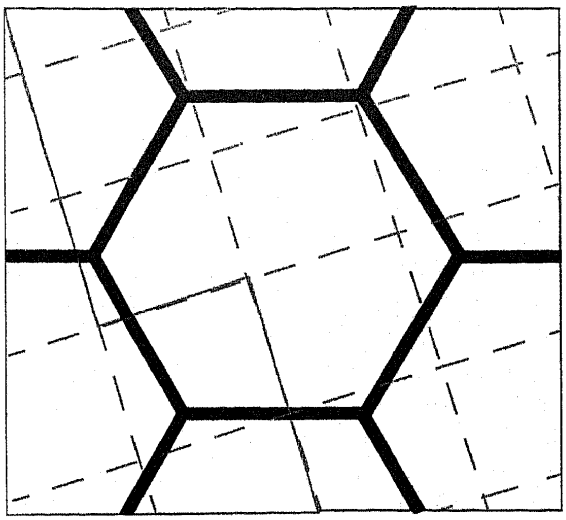

Figure 10. The surface grids used by the Parallel Ocean Program (dashed lines) and the global atmospheric grid (solid lines). The grids communicate in an accurate and conservative manner using a library called SCRIP.

Multiple blocks that belong to a process need not necessarily be geographically contiguous. For example, a particular process might own a block in North America, a block in the south Atlantic, a block in Africa, and a block in the tropical Pacific.

Each process will work on both the atmosphere above and the surface below; here, surface means ocean and land surface. This is a *single executable* paradigm. In contrast, a *multiple executable* paradigm might let the ocean, atmosphere, and land surface models run as distinct executables. Our single executable approach minimizes the message passing needed to compute the fluxes between the atmosphere and the surface.

Because a process can own multiple atmosphere blocks and multiple surface blocks, we can load balance the global model by allocating blocks to processes so that similar mixes of land and ocean points are assigned to all processes. On average, each process will work on about two-thirds of the ocean cells and one-third of the land cells, simply because the continents cover about one-third of the Earth's surface.

Because a flux coupler naturally communicates among a climate model's various components, it is a potential computational bottleneck. In particular, when the atmosphere and ocean models are parallelized using 2D domain decomposition, flux coupler parallelization itself becomes important. A fast flux coupler permits frequent communication among the component models—for example, by permitting resolution of the diurnal cycle, which is crucial over land and is also thought to be important for the atmospheric interactions with the tropical upper ocean. With a serial flux coupler, we can choose less frequent (for example, daily) communication to improve computational speed but at the cost of sacrificing physical realism. We intend to provide frequent flux coupling between our AGCM and OGCM to permit realistic simulations of the diurnal cycle and other high-frequency variability affecting both the atmosphere and ocean. Such frequent flux coupling is practical with the flux coupler design outlined earlier.

In addition to effective load balancing, an efficient flux coupler should require minimal interprocessor communication. Communication is required when a grid cell, generally at the processor's horizontal grid domain boundary, is shared with one or more other processors. The spherical geodesic grids are constructed so that any grid cell on the fine (ocean) grid will overlap with at most three grid cells from the coarse (atmosphere) grid. This property keeps interprocessor communication to a minimum and, along with effective load balancing, promotes the flux coupler scaling to many processors.

Climate models improve with time, but the model development process is never finished. Geodesic grids can solve some of our modeling problems, but others remain. A particularly difficult problem is realistically simulating the distributions of water vapor and clouds and how these distributions change when the climate changes. Our next round of model development will be aimed at improved representations of water vapor and clouds, in part through the use of floating model layers that follow the moisture as it moves vertically through a column of air.

Acknowledgments

This research is supported through a cooperative agreement between the Climate Change Prediction Program of the US Department of Energy and Colorado State University. Our co-investigators on this project are Akio Arakawa (University of California, Los Angeles), Albert J. Semtner, Jr. (US Naval Postgraduate School), Wayne Schubert (Colorado State University), and Scott Fulton (Clarkson University).

References

1. A.J. Semtner, "Modeling Ocean Circulation," *Science*, vol. 269, no. 5229, June 1995, pp. 1379–1385.
2. R.D. Smith, S. Kortas, and B. Meltz, *Curvilinear Coordinates for Global Ocean Models*, tech. report LA-UR-95-1146, Los Alamos Nat'l Lab., Los Alamos, N.M., 1995.
3. M. Jarraud and A.J. Simmons, "The Spectral Technique," *Seminar on Numerical Methods for Weather Prediction*, European Centre for Medium Range Weather Prediction, Reading, UK, 1983, pp. 1–59.
4. D.L. Williamson and P.J. Rasch, "Water Vapor Transport in the NCAR CCM2," *Tellus*, vol. 46A, no. 1, Jan. 1994, pp. 34–51.
5. D.B. Haidvogel et al., "Global Modeling of the Ocean and Atmosphere Using the Spectral Element Method," *Numerical Methods in Atmospheric and Oceanic Modeling*, C.A. Lin, R. Laprise, and H. Ritchie, eds., European Centre for Medium Range Weather Forecasts, Reading, UK, 1999, pp. 505–531.
6. R.P. Heikes and D.A. Randall, "Numerical Integration of the Shallow Water Equations on a Twisted Icosahedral Grid, Part 1: Basic Design and Results of Tests," *Monthly Weather Rev.*, vol. 123, no. 6, June 1995, pp. 1862–1880.
7. D.L. Williamson, "Integration of the Barotropic Vorticity Equation on a Spherical Geodesic Grid" *Tellus*, vol. 20, no. 4, Oct. 1968, pp. 642–653.
8. R. Sadourny, A. Arakawa, and Y. Mintz, "Integration of the Nondivergent Barotropic Equation with an Icosahedral Hexagonal Grid for the Sphere," *Monthly Weather Rev.*, vol. 96, no. 6, June 1968, pp. 351–356.
9. J.R. Baumgardner and P.O. Frederickson, "Icosahedral Discretization of the Two-Sphere," *SIAM J. Numerical Analysis*, vol. 22, no. 6, June 1985, pp. 1107–1115.
10. T.D. Ringler and D.A. Randall, "A Potential Enstrophy and Energy Conserving Numerical Scheme for Solution of the Shallow-Water Equations on a Geodesic Grid," *Monthly Weather Rev.*, vol. 130, no. 5, May 2002, pp. 1397–1410.
11. T.G. Jensen, "Artificial Retardation of Barotropic Waves in Layered Ocean Models," *Monthly Weather Rev.*, vol. 124, no. 6, June 1996, pp. 1272–1283.
12. P.W. Jones, *How to Use the Parallel Ocean Program (POP): A Tutorial for Users and Developers*, tech. report LA-UR-02-502, Los Alamos Nat'l Lab., Los Alamos, N.M., 2002.
13. P.W. Jones, "First- and Second-Order Conservative Remapping Schemes for Grids in Spherical Coordinates," *Monthly Weather Rev.*, vol. 127, no. 9, Sept. 1999, pp. 2204–2210.

David A. Randall is a professor of atmospheric science at Colorado State University. His research interests include general circulation model design and the role of clouds in climate. He received his PhD in atmospheric science from the University of California, Los Angeles. He is a fellow of the American Meterological Society, the American Geophysical Union, and the American Association for the Advancement of Science. Contact him at the Dept. of Atmospheric Science, Colorado State Univ., Fort Collins, CO 80523-1371; randall@atmos.colostate.edu.

Todd D. Ringler is a research scientist in the Department of Atmospheric Science at Colorado State University. His research focus is on modeling the large-scale motions of the atmosphere and ocean by developing numerical models of the climate system. He received a BS in aerospace engineering from West Virginia University and an MS and PhD in aerospace engineering from Cornell University. Contact him at the Dept. of Atmospheric Science, Colorado State Univ., Fort Collins, CO 80523-1371; todd@atmos.colostate.edu.

Ross P. Heikes is a research scientist at Colorado State University. His research interests include numerical methods and atmospheric general circulation. He received a BS in mathematics and atmospheric science and a PhD in atmospheric science from Colorado State University. Contact him at Dept. Atmospheric Science, Colorado State Univ., Fort Collins, CO 80523-1371; ross@cirque.atmos.colostate.edu.

Phil Jones is a technical staff member in the Theoretical Fluid Dynamics Group at Los Alamos National Laboratory. His research interests include coupled climate modeling, ocean modeling, remapping, and interpolation and computational performance of climate models. He holds a PhD in astrophysical, planetary, and atmospheric sciences from the University of Colorado and a BS in physics and math from Iowa State University. He is a member of the American Geophysical Union and the American Meteorological Society. Contact him at T-3 MS B216, Los Alamos Nat'l Lab, Los Alamos, NM 87544; pwjones@lanl.gov; http://home.lanl.gov/pwjones.

John Baumgardner is a technical staff member in the Fluid Dynamics Group of the Theoretical Division at Los Alamos National Laboratory. His research interests include development of a hybrid vertical coordinate global ocean model and mantle dynamics modeling of terrestrial planets. He has a PhD in geophysics and space physics from UCLA and is a member of the American Geophysical Union. Contact him at MS B216, Los Alamos Nat'l Lab., Los Alamos, NM 87545; baumgardner@lanl.gov.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.