A Barotropic Global Ocean Model and Its Parallel Implementation on Unstructured Grids

H. Öksüzoğlu and A.G.M. van Hees
Faculty of Physics and Astronomy
Computational Physics Group
Utrecht University, the Netherlands

Abstract

Unstructured grids can represent the complex geometry of the ocean basin with high fidelity. The lack of development tools supporting irregular grid problems discourages the use of such grids on parallel architectures. The state of the art ocean models are based on logically rectangular grids which makes it difficult to fit the complex ocean boundaries. In this paper,we demonstrate the use of unstructured triangular grids for solving a barotropic ocean model in spherical geometry with realistic continental boundaries. The model is based on the shallow water equations with a Coriolis force. A realistic wind forcing and a simple bottom friction term are also included. The numerical method is a cell based upwind finite volume scheme with explicit time stepping. From a parallelization point of view, that means there is only a nearest neighbour communication. A heuristic domain partitioning method was employed to distribute the load among processors. The resulting decomposition resembles 2D grid topology with some long distance communication paths. The model was implemented using the PVM message passing library and tested on a cluster of workstations and on an IBM SP2.

1 Introduction

Oceans play an essential role in shaping long term weather patterns. That is because the heat capacity of the ocean is much larger than the atmosphere's heat capacity. The pole-ward heat transport by the ocean currents, for example, makes the climate of the Northern Europe milder than it would be without this extra source of heat. The understanding of ocean physics is an important factor in predicting climate changes. The range of scales that need to be modelled for a realistic simulation has been beyond the capacity of the fastest supercomputers. The recent developments in computer technology made it possible to develop numerical ocean models that give a qualitative description of ocean currents [1] [8] [11]. There is a need for faster computers and more sophisticated numerical models for quantitative predictions. For a review of the recent developments see [10] [7].

In this study, a numerical ocean circulation model has been developed to run on parallel computers. This is a two dimensional wind driven ocean model using unstructured triangulated grids. The use of this type of grid makes it possible to describe the complicated geometry of the ocean boundaries efficiently, and opens the possibility of local refinement in a very flexible way. The model includes a free moving surface, the Coriolis force and a bottom friction term. For a realistic simulation also salt and heat transport need to be added, which requires a full three dimensional model. As far as parallelization is concerned, the depth averaged two dimensional wind driven circulation is the most challenging part of a numerical ocean model. After getting good performance on this two dimensional version it is expected to be easier to have a three dimensional model on a parallel computer.

2 Barotropic Ocean Model

The geometry of the ocean basin, which is thousands of kilometers wide and a few kilometers deep, calls for a simplification of the equations of the fluid dynamics. In the vertical direction gravity and in the horizontal direction the Coriolis force dictates the flow characteristics. With these observations in mind, it is possible to obtain a simplified set of equations that can model the wind driven barotropic component of the ocean circulation [2].

The shallow water equations in vector notation are

$$\begin{bmatrix} h \\ h\mathbf{U} \end{bmatrix}_t + \begin{bmatrix} \nabla \cdot (h\mathbf{U}) \\ \nabla \cdot (h\mathbf{U} \otimes \mathbf{U} + \frac{h^2}{2}\mathbf{I}) \end{bmatrix} = \begin{bmatrix} 0 \\ -\mathbf{f} \wedge h\mathbf{U} - \tau_{\mathbf{b}} + \tau_{\mathbf{w}} \end{bmatrix}$$

where $\mathbf{U}$ is the velocity vector, $h = gH$, H is the height, g is the gravitational acceleration), $\mathbf{I}$ is the identity matrix, $\mathbf{f}$ is the Coriolis vector (normal to surface and with a magnitude changing with latitude) , $\tau_{\mathbf{b}}$ is the bottom friction and $\tau_{\mathbf{w}}$ represents the wind forcing. These equations express the conservation of mass and momentum in the horizontal direction and can be derived from the Navier-Stokes equations. They can be a building block for a more complete three dimensional ocean model such as MICOM [1].

Since the molecular diffusion is far too small to be resolved on any reasonable grid, its effects need to be modelled. The simplest approach is to add surface friction terms as in the momentum equation above. A more realistic model would also include a turbulent diffusion term.

This model has two slightly different interpretations. First, it can represent a flat bottomed ocean with a free moving upper surface. Second, it can represent a single layer on top of a motionless abyss with a slightly different density. The interface between the layer and the abyss is free to move. The gravitational acceleration is scaled with a dimensionless density difference, $h = \Delta\rho g H/\rho$. This is called a reduced gravity $1\frac{1}{2}$ layer model.

Since there is no horizontal diffusion in the equations, the lateral boundary condition is that the normal velocity should be zero. But, the tangential velocity can have any value. These equations form a hyperbolic system very similar to the equations of isentropic compressible flow.

3 Numerical Method

The current implementation is a cell based upwind finite volume scheme for the shallow water equations on a sphere and can be first or second order accurate in space and time. Because of upwinding, the method is stable without any artificial diffusion terms. For each triangle, mass and momentum conservation is enforced. So, the flow variables are associated with the triangles and the fluxes are associated with the edges. The time stepping scheme is explicit and there is only a nearest neighbour dependence. This makes it easy to parallelize but limits the time step size due to fast moving surface gravity waves.

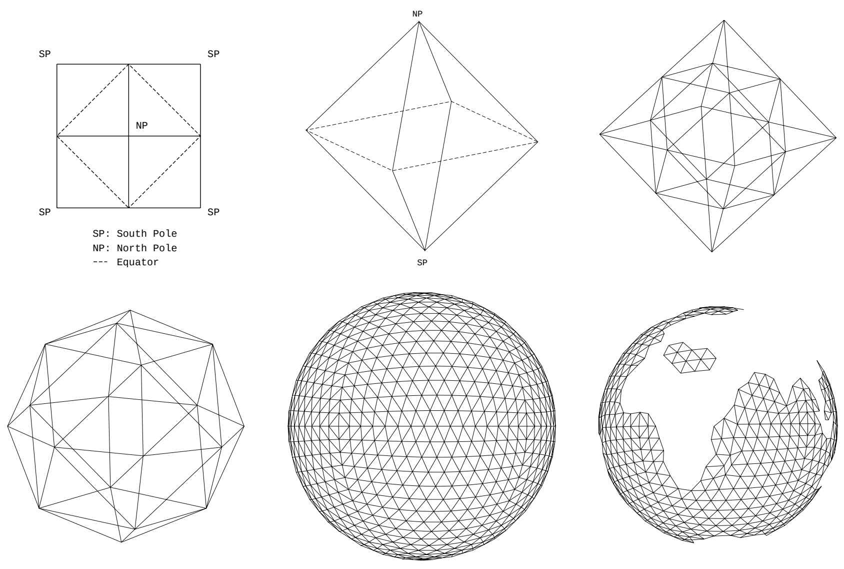

Figure 1: Triangulation of the sphere

The most obvious way to generate a grid on the sphere is to use spherical coordinates. The resulting mesh has a grid concentration near poles which is undesirable from a numerical point of view. Smaller cells near the poles are a waste of computational resources where high resolution is not needed. They also severely limit the time step size for explicit time stepping. This type of grid has been widely used despite its disadvantages. Instead of this conventional approach, we make use of a nearly uniform triangulation of the sphere. We start with a square (Figure 1). The center of the square corresponds to the north pole, and the four corners correspond to the south pole. The dashed line is along the equator. Then, we fold it along the dashed lines and stretch it to obtain an octahedron. Next, we recursively subdivide the triangles until the desired level of resolution is obtained. By mapping this onto a sphere we get a triangulation of the sphere. The reason we picked the octahedron instead of the icosahedron, for example, is its simplicity and symmetry properties. After including topography information, we can get rid of land points. The resulting grid is unstructured and is composed of triangles covering the sea points. The use of triangular grids for barotropic models in spherical geometry was suggested before [12] and there is a renewed interest in the use of irregular grids [5]. Another approach for generating nearly uniform grids is to map a cube onto a sphere as presented in [9] which is more suitable for atmospheric applications.

Here, we give an example solution with the second order accurate version. For this example, we have used a realistic wind forcing dataset, which is an average of the observations made between the years 1870-1976 [4]. The main features of the wind driven surface currents have been captured (Figure 2).

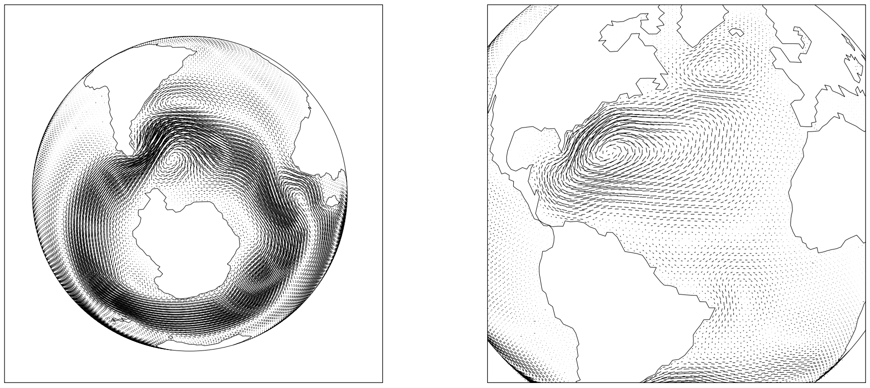

Figure 2: The Antarctic Circumpolar Current and the Gulf-stream, Velocity Vectors, 12280 vertices, 23397 triangles

4 Parallel Implementation

We have used a message passing paradigm for parallelization. The PVM message passing library [3] was found suitable for a portable implementation. For the partitioning of the domain, we start with the square mentioned in the grid generation stage (Figure 1). We simply subdivide this square into smaller rectangles to obtain the desired number of sub-domains. Sub-domains need to communicate with the nearest neighbours except at the edges of the square, where there is a long distance communication[1]. In Figure 3, a 2×2 partitioning

[1]Similar to periodic boundary conditions

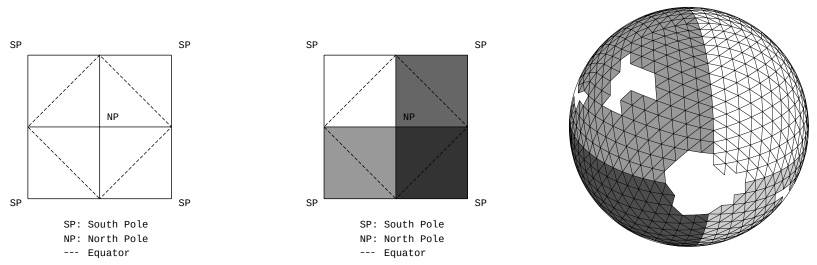

Figure 3: Partitioning of the domain into four sub-domains

of the sphere is shown. In order to get 6 subdomains, for example, we could divide the square into 3×2 or 6×1 rectangles.

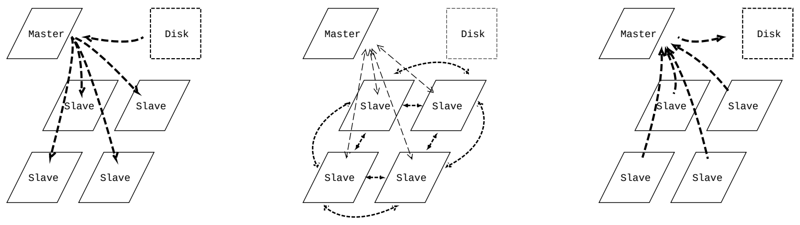

Figure 4: Input, time stepping and output with four sub-domains

A master process reads the grid information and the initial data and distributes the relevant information to the slave processes for each sub-domain (Figure 4). Then slaves start time stepping and exchange flux information at the end of each time step. They also talk to the master to determine the time step size to be taken. At the end of computation they all send the results to the master and the master writes the output. For the first order accurate version, only the fluxes that are associated with the edges need to be communicated. The second order accurate version will need additional information to be exchanged in order to construct piecewise linear functions for each cell. However, the parallel performance is expected to be similar to the first order case , because there is more computation to be done per cell. The parallelization of the second order version is under development.

5 Results and Conclusion

To test the parallel efficiency, we have run the PVM implementation of the first order model on an IBM SP2 and a cluster of Hewlett-Packard 700 series of workstations connected by an Ethernet network. We have used four problems with different size to see the effect of the problem size on efficiency. The problems have 1520, 5910, 23397 and 92713 triangles, respectively. Timings are done

Speedup	1	2	4	8	16	32	64
1520 T	1	1.51	2.27	1.70	1.57		
5910 T	1	1.52	2.78	4.29	3.97		
23397 T	1	1.20	2.53	4.72	8.85		
92713 T	1	0.62	1.62	3.66	8.17	15.84	15.74

Table 1: Speedup on IBM SP2 using PVM3

Speedup	1	2	4	8
1520 T	1	1.66	2.38	3.10
5910 T	1	1.57	3.03	4.52
23397 T	1	1.55	2.94	5.20
92713 T	1	1.11	2.57	3.64

Table 2: Speedup on a cluster of HP workstations using PVM3

for 500 time steps to reduce the effect of initialization cost and input-output operations have been excluded. The observation from that experiment is that the best performance is near the diagonal (Table 1 and 2). This behavior can be explained as follows. The cost of computation is proportional to the number of triangles and the cost of communication is proportional to the square root of the number of triangles. Computation scales with the domain area, communication scales with the domain edge length. That means the communication cost will eventually dominate with increasing number of processors when the problem size is fixed. That explains the diminishing returns for the smaller problems. The performance degradation for a fixed number of processors with increasing problem size is due to the inefficient use of memory by the master process. When there are n slaves running on n processors, the master is sharing the resources of one processor. This can be fixed in the future implementations. For the largest problem, the partitioning method which ignores the existence of continents shows its weakness for large number of domains. For the 64 node case (Table 1), some of the nodes have little or no work to do, because the corresponding domain is on continents. A more balanced mesh partitioning may improve the performance. Optimum partitioning of unstructured meshes for parallel computing is an active area of research and several software packages are under development. This simple domain partitioning strategy will be replaced by a general purpose mesh partitioner like Metis [6]. In order to explain the difference in speedup figures for the HP cluster and SP2, more detailed timings are needed.

Finally, we give the actual timings for the sequential code (Table 3). The timings for the parallel case can be computed by dividing with the corresponding speedup.

Timings	IBM SP2	HP 712
1520 T	41	143
5910 T	159	543
23397 T	629	2118
92713 T	2519	8425

Table 3: Actual timings for 500 time steps for the sequential code (in Seconds)

6 Acknowledgements

This work was supported by FOM under MPR project and the computer resources were provided by SARA. The authors would like to thank Cees Vreugdenhil and Aad van der Steen for their support in various stages of this work.

References

[1] R. Bleck, S. Dean, M. O'Keefe, and A. Sawdey. A comparison of data-parallel and message-passing versions of the miami isopycnic coordinate ocean model. *Parallel Computing*, 21:1695–1720, 1995.

[2] B. Cushman-Roisin. *Introduction to Geophysical Fluid Dynamics*. Prentice Hall, 1994.

[3] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, W. Jaing, and V. Sunderam. *PVM: A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.

[4] S. Hellerman and M. Rosenstein. Normal monthly wind stress over the world ocean with error estimates. *Journal of Physical Oceanography*, 13:1093–1104, 1983.

[5] M. Iskandarani, D. B. Haidvogel, and J. P. Boyd. A staggered spectral element model with application to the oceanic shallow water equations. *International Journal for Numerical Methods in Fluids*, 20:393–414, 1995.

[6] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, to appear.

[7] J. C. McWilliams. Modeling the ocean general circulation. *Annual Review of Fluid Mechanics*, 28:215–248, 1996.

[8] R. J. Procassini, S. R. Whitman, and W. P. Dannevik. Porting a global ocean model onto a shared-memory multiprocessor: Observations and guidelines. *The Journal of Supercomputing*, 7:287–321, 1993.

[9] C. Ronchi, R. Iacono, and P.R. Paolucci. Finite difference approximation to the shallow water equations on a quasi-uniform spherical grid. In B. Hertzberger and G. Serazzi, editors, *Lecture Notes in Computer Science 919*. Springer-Verlag, May 1995. Proceedings of HPCN Europe 1995.

[10] A. J. Semtner. Modeling ocean circulation. *Science*, 269:1379–1385, 1995.

[11] R. D. Smith, J. K. Dukowicz, and R. C. Malone. Parallel ocean general circulation modeling. *Physica D*, 60:38–61, 1992.

[12] D. L. Williamson. Integration of the primitive barotropic model over a spherical geodesic grid. *Monthly Weather Review*, 98:512–520, 1970.