Parallel Conservative Scheme for Solving the Shallow-Water Equations

Neta, Beny; Lustman, L.

Parallel Conservative Scheme for Solving the Shallow-Water Equations

B. NETA AND L. LUSTMAN

Department of Mathematics, Naval Postgraduate School, Monterey, California

3 February 1992 and 28 April 1992

ABSTRACT

To improve the simulation of nonlinear aspects of the flow over steep topography, a potential enstrophy– and energy-conserving finite-difference scheme for the shallow-water equations was derived by Arakawa and Lamb.

Here a parallel algorithm is developed for the solution of these equations, which is based on Arakawa and Lamb's scheme. It is shown that the efficiency of the scheme on an eight-node INTEL iPSC/2 hypercube is 81%. Forty mesh points in the x direction and 19 in the y direction were used in each subdomain.

1. Introduction

Arakawa and Lamb (1981) have developed a finite-difference scheme to solve the shallow-water equations with topography. Flow over and near steep mountains is governed during advective processes by the conservation of (absolute) potential vorticity $q = \eta/h$ where η is the (absolute) vorticity and h is the depth of the fluid. They "have found that conventional finite difference schemes for the momentum equation, when applied to the shallow-water equations, correspond to very bad advection schemes for the potential vorticity in the presence of steep mountains." They have developed a scheme to conserve potential enstrophy and total energy. The scheme requires the staggering of the variables u, v, and h as shown in Fig. 1.

In the next section we summarize the scheme as given in Arakawa and Lamb (1981). The domain decomposition and the parallel algorithm will be detailed in section 3. There we also comment on how domain decomposition is performed on the sphere. We conclude with an example showing the efficiency of the parallel algorithm on an INTEL iPSC/2 hypercube.

2. Finite-difference conservative scheme

Here we follow the notation used in Arakawa and Lamb (1981) but generalize the discretization to the case that $\Delta x \neq \Delta y$. The governing equations for quasi-static motion in a homogeneous incompressible fluid with a free surface are

$$\frac{\partial \mathbf{v}}{\partial t} + q\mathbf{k} \times \mathbf{v}^* + \nabla(K + \Phi) = 0 \tag{1}$$

$$\frac{\partial h}{\partial t} + \nabla \cdot \mathbf{v}^* = 0, \tag{2}$$

where

q	(absolute) potential enstrophy
η	(absolute) vorticity
q	$= \eta/h = (f + \zeta)/h$
f	Coriolis parameter
ζ	relative vorticity $= \mathbf{k} \cdot \nabla \times \mathbf{v}$
$\mathbf{v}^*$	mass flux $= h\mathbf{v}$
$\mathbf{v}$	horizontal velocity
$\mathbf{k}$	vertical unit vector
h	vertical extent of a fluid column above the bottom surface
K	kinetic energy per unit mass $= \frac{1}{2}\lvert\mathbf{v}\rvert^2$
g	gravitational acceleration
h_s	bottom surface height
Φ	$= g(h + h_s)$

The discretization of the momentum equation (1) is given by

$$\begin{aligned}\frac{\partial}{\partial t} u_{ij+1/2} &- \alpha_{ij+1/2} v^*_{i+1/2j+1} - \beta_{ij+1/2} v^*_{i-1/2j+1} \\ &- \gamma_{ij+1/2} v^*_{i-1/2j} - \delta_{ij+1/2} v^*_{i+1/2j} \\ &+ \epsilon_{i+1/2j+1/2} u^*_{i+1/2j+1/2} - \epsilon_{i-1/2j+1/2} u^*_{i-1/2j+1/2} \\ &+ d^{-1}[(K + \Phi)_{i+1/2j+1/2} \\ &- (K + \Phi)_{i-1/2j+1/2}] = 0 \end{aligned} \tag{3}$$

Corresponding author address: Prof. Beny Neta, Department of Mathematics, Naval Postgraduate School, Code MA/ND, Monterey, CA 93943-5100.

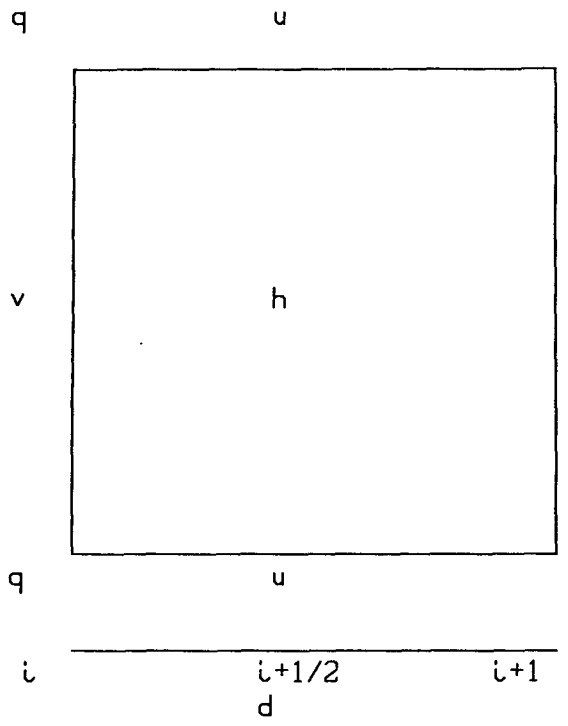

FIG. 1. The C grid.

$$\frac{\partial}{\partial t} v_{i+1/2j} + \gamma_{i+1j+1/2} u^*_{i+1j+1/2} + \delta_{ij+1/2} u^*_{ij+1/2} + \alpha_{ij-1/2} u^*_{ij-1/2} + \beta_{i+1j-1/2} u^*_{i+1j-1/2} + \phi_{i+1/2j+1/2} v^*_{i+1/2j+1} - \phi_{i+1/2j-1/2} v^*_{i+1/2j-1} + d^{-1}[(K+\Phi)_{i+1/2j+1/2} - (K+\Phi)_{i+1/2j-1/2}] = 0, \quad (4)$$

where the parameters α, β, γ, δ, ϵ, and ϕ are given by linear combinations of q

$$\alpha_{ij+1/2} = \frac{1}{24}(2q_{i+1j+1} + q_{ij+1} + 2q_{ij} + q_{i+1j}) \quad (5)$$

$$\beta_{ij+1/2} = \frac{1}{24}(q_{ij+1} + 2q_{i-1j+1} + q_{i-1j} + 2q_{ij}) \quad (6)$$

$$\gamma_{ij+1/2} = \frac{1}{24}(2q_{ij+1} + q_{i-1j+1} + 2q_{i-1j} + q_{ij}) \quad (7)$$

$$\delta_{ij+1/2} = \frac{1}{24}(q_{i+1j+1} + 2q_{ij+1} + q_{ij} + 2q_{i+1j}) \quad (8)$$

$$\epsilon_{i+1/2j+1/2} = \frac{1}{24}(q_{i+1j+1} + q_{ij+1} - q_{ij} - q_{i+1j}) \quad (9)$$

$$\phi_{i+1/2j+1/2} = \frac{1}{24}(-q_{i+1j+1} + q_{ij+1} + q_{ij} - q_{i+1j}) \quad (10)$$

and

$$q_{ij} = \frac{(f+\zeta)_{ij}}{h^q_{ij}} \quad (11)$$

where

$$\zeta_{ij} = \frac{(u_{ij-1/2} - u_{ij+1/2})}{\Delta y} + \frac{(v_{i+1/2j} - v_{i-1/2j})}{\Delta x}, \quad (12)$$

$$h^{(q)}_{ij} = \frac{1}{4}(h_{i+1/2j+1/2} + h_{i-1/2j+1/2} + h_{i-1/2j-1/2} + h_{i+1/2j-1/2}). \quad (13)$$

Also

$$u^*_{i+1j+1/2} \equiv [h^{(u)}u]_{i+1j+1/2}, \quad (14)$$

$$v^*_{i+1/2j} \equiv [h^{(v)}v]_{i+1/2j}, \quad (15)$$

$h^{(u)}$, $h^{(v)}$ are the h values at u and v points defined by the following averaging process (and denoted by overbar x or overbar y for averaging in the x, y direction, respectively):

$$h^{(u)}_{ij+1/2} \equiv \overline{h}^x_{ij+1/2}, \quad (16)$$

$$h^{(v)}_{i+1/2j} \equiv \overline{h}^y_{i+1/2j}, \quad (17)$$

and

$$K_{i+1/2j+1/2} = \left(\overline{\frac{1}{2}u^2}^x + \overline{\frac{1}{2}v^2}^y\right)_{i+1/2j+1/2}. \quad (18)$$

The discretization of (2) is given by

$$\frac{\partial}{\partial t} h_{i+1/2j+1/2} + \frac{u^*_{i+1j+1/2} - u^*_{ij+1/2}}{\Delta x} + \frac{v^*_{i+1/2j+1} - v^*_{i+1/2j}}{\Delta y} = 0. \quad (19)$$

Note that (12) and (19) are the only equations affected by the fact that $\Delta x \neq \Delta y$.

3. Domain decomposition

This section describes how to decompose the domain (a channel) into p overlapping subdomains. This is conducted for the example discussed by Arakawa and Lamb (1981). The domain is a channel of length 6000 km and width 2000 km. Periodic boundary conditions are assumed in the x direction and rigid-wall boundary conditions in the y direction. The mean surface height H_0 is 5 km, the acceleration of gravity g is 9.8 m s^{-2}, and the Coriolis parameter f is 10^{-4} s^{-1}. The bottom topography is a narrow ridge, centered at $x = 3000$ km, that uniformly extends across the channel in y and has a triangular shape in x, with a maximum height of 2 km and a bottom width of 1000 km (see Fig. 2).

The domain is decomposed into p horizontal strips;

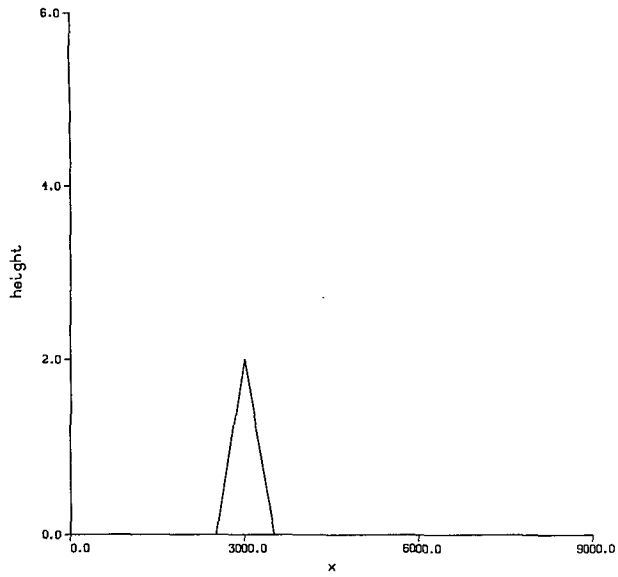

FIG. 2. Topography in the channel.

thus, each processor implements simple periodic boundary conditions in x. Since Arakawa's scheme requires the lines $j-2$ to $j+2$ in order to advance the solution on line j, we require the subdomains to overlap on two grid lines (see Fig. 3). Each processor will receive and send the updated values of u, v, and h on two border lines to its nearest neighbors. By using the Gray code (see, e.g., Brualdi 1992, p. 103), the subdomains are distributed among the processors in such a way that the updating of neighboring strips is always completed by communication along the shortest path, along one hypercube edge.

There will usually be two messages sent and two received per time step: one for $j_{\min}$ and one for $j_{\max}$. By a judicious mapping, the message data may be made contiguous in memory, so the program must only know the starting address and the message length. There is no need of buffering, as these messages never overwrite each other. It is also possible to use asynchronous routines, although the time saved seems small.

The subdomains contain about the same number of horizontal grid lines, to promote load balancing. It is not possible, however, to have exactly the same work completed in each processor, because two of the processors treat actual boundary conditions ($v = 0$).

Since the length of the channel is much larger than the width, we have decided to subdivide the domain into vertical strips instead of horizontal strips. This will allow the use of a larger number of processors each having enough grid points to compute the solution. As it turns out, this is also more efficient to implement.

If one is to solve the shallow-water equations on a hemisphere, a polar stereographic projection can be used. This will lead to a rectangular domain. Distortion becomes severe, however, if the projection is extended into the opposite hemisphere. Also, artificial boundary conditions placed in the equatorial region give rise to errors that eventually propagate into higher latitudes (see Haltiner and Williams 1980). If a global model is to be solved, we suggest that the sphere be decomposed into latitudinal strips: that is, one subdomain containing each pole, and all have boundaries coinciding with latitudes. The width of each strip will depend on experience that will ensure load balancing. The reason for latitudinal strips is so that each pole will be in one subdomain and the communication will be minimized.

4. Numerical experiments

Numerical experiments were performed on an INTEL iPSC/2 hypercube using eight processors. A serial version (running on one of the nodes) and a parallel version (using all eight processors) of the code were run. A grid size $d = 50$ km was used, so that each of the processors will have five horizontal grid lines $[\frac{1}{8}(2000/50)]$. This is a fine mesh (finer than those used by Arakawa and Lamb), and it is required in order to have at least one nonoverlapping line in each subdomain.

The initial conditions are a uniform zonal current $u = 20$ m s^{-1} and a horizontal free surface. The integration is performed until $t = 1330$ min using $\Delta t = 1$ min. Figure 4 shows the wind speed and height at $t = 1330$ min using the serial version. Figure 5 shows the result of the parallel version with eight processors. Notice that the results are identical but the run time for the serial is 4140 s and that of the parallel is 809 s.

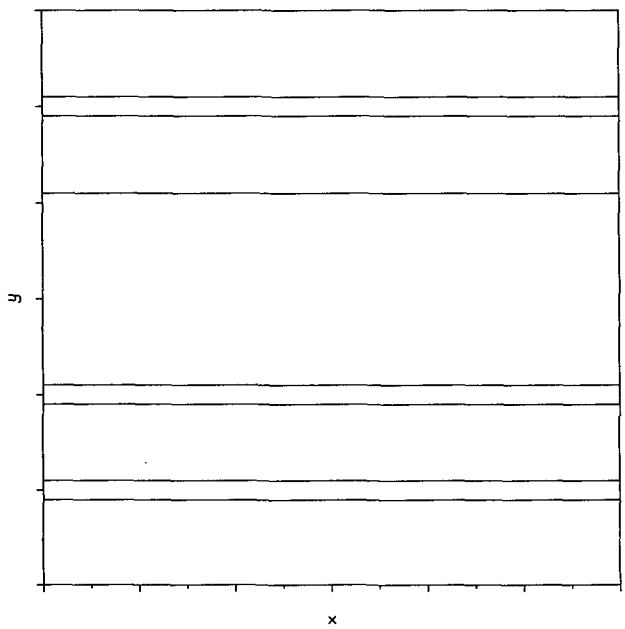

FIG. 3. The domain decomposed into horizontal strips.

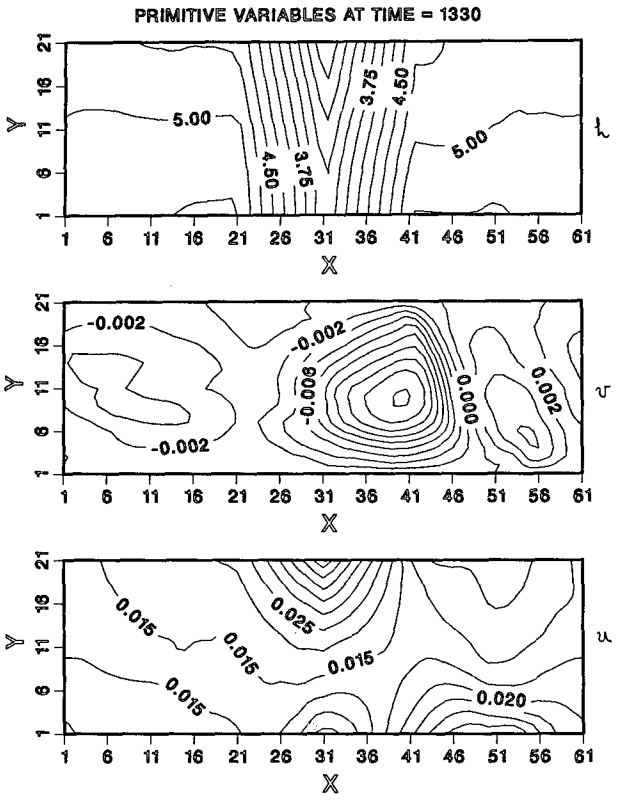

FIG. 4. The solution at t = 1330 min using the serial version.

As mentioned earlier, this mesh size is fine, and since the scheme as given by (3)–(19) requires a uniform mesh ($\Delta x = \Delta y = d$), we had to use too many points in the x direction. The code was run using Δx = 150 km, Δy = 50 km, and Δt = 1 min, and the results were unchanged, as shown in Fig. 6. Equations (12) and (19) were modified in a way to accommodate $\Delta x \neq \Delta y$.

To show the benefit of this parallel algorithm, we have measured the speedup defined by

$$S = \frac{T_p(1)}{T_p(p)} \tag{20}$$

where $T_p(i)$ is the execution time required when using i processors. This is the most common formulation of speedup.

As defined, the speedup should ideally be directly proportional to p, the number of processors used. The efficiency, defined as

$$E = \frac{S}{p}, \tag{21}$$

provides a quantitative measure of how closely the observed speedup approaches the ideal result. For the example shown in Figs. 4 and 5, the speedup is

$$S = \frac{4140}{809} = 5.12 \tag{22}$$

and the efficiency is

$$E = \frac{5.12}{8} = 0.64 = 64\%. \tag{23}$$

This is not efficient enough, and thus we have decided to divide the channel by vertical lines. This turns out to be more efficient and yields the identical solution. The run time for the same example is 635 s. Thus,

$$S = \frac{4140}{635} = 6.52, \tag{24}$$

$$E = \frac{6.52}{8} = 81\%. \tag{25}$$

When using four processors, the run time is 1149 s, and thus S = 3.6 and the efficiency increases to

$$E = \frac{3.6}{4} = 90\%. \tag{26}$$

Clearly, the amount of communication is smaller (relative to time spent on computation). If one takes

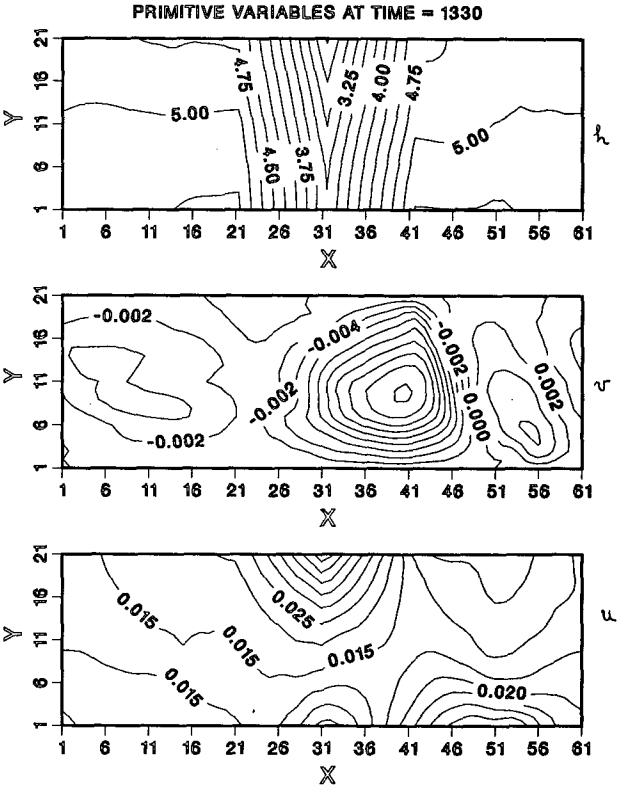

FIG. 5. The solution at t = 1330 min using eight processors.

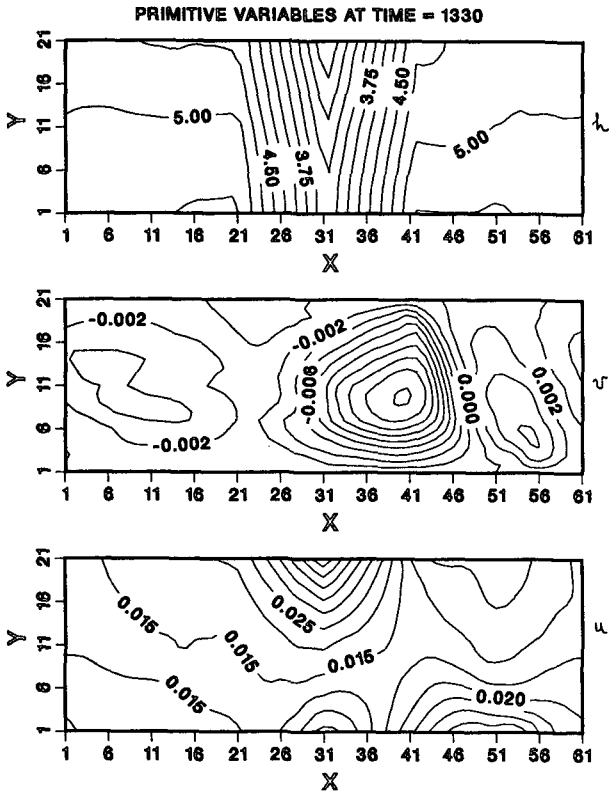

FIG. 6. The solution at t = 1330 min using eight processors and a nonuniform grid.

a simpler scheme (such as the unstaggered A grid), the efficiency will be lower because the time spent on computation will be closer to the communication time.

The Fortran software developed during this project is available on request.

5. Conclusions

An efficient parallel algorithm for the solution of the shallow-water equations was developed and tested. It is based on the potential enstrophy– and energy-conserving scheme developed by Arakawa and Lamb. The efficiency of the scheme when using eight processors is 81% on an iPSC/2 INTEL hypercube computer.

Acknowledgments. We would like to thank the referees for their helpful comments. This paper was prepared in part for the Office of Naval Research. Funding was provided by the Naval Postgraduate School.

REFERENCES

Arakawa, A., and V. R. Lamb, 1981: A potential enstrophy and energy conserving scheme for the shallow water equations. *Mon. Wea. Rev.*, **109**, 18–36.

Brualdi, R. A., 1992: *Introductory Combinatorics,* North Holland, 618 pp.

Haltiner, G. J., and R. T. Williams, 1980: *Numerical Prediction and Dynamic Meteorology,* J. Wiley and Sons, 477 pp.

Lustman, L., and B. Neta, 1992: Software for the parallel conservative scheme for the shallow water equations, Naval Postgraduate School Tech. Rep., NPS-MA-92-004, Monterey, California, 28 pp.