A Class of Single-Cell High-Order Semi-Lagrangian Advection Schemes

FENG XIAO

Computational Science Laboratory, Institute of Physical and Chemical Research, Saitama, Japan

(Manuscript received 13 November 1998, in final form 20 May 1999)

ABSTRACT

A class of semi-Lagrangian schemes has been derived for solving the advection equation. Compared with other semi-Lagrangian-type schemes, the presented schemes require less computational stencils for interpolation construction. Besides the dependent variable itself, its spatial derivatives are also evaluated based on a Lagrangian invariant solution. This makes estimating the first-order derivatives from the values of the dependent variable at neighboring grid points unnecessary. The resulting numerical formula appears spatially compact and only one mesh cell is needed for constructing the interpolation profile.

The 2D basic formulation is based on a rational interpolation function. It shows an ability to prevent numerical oscillation. Some variants of the scheme can be easily obtained by minor modifications. It is easy to get the desired numerical properties such as diffusion reduction, oscillation suppression, or (more strongly) monotonicity with the presented schemes.

Grid refinement analysis shows that all the schemes presented in this paper have convergence factors larger than 2 based on an l_2 norm.

The presented schemes need some extra memory space to store the derivatives of the interpolation function, but do appear competitive with other conventional semi-Lagrangian methods based on Hermite interpolants, in terms of arithmetic operation counts.

Parallel implementation shows that the presented schemes are easily portable to a parallel environment with distributed memory architecture and data communications take place only on the cells on the boundaries of the parallel partition.

1. Introduction

Advection is generally known as an important process in atmospheric dynamics. Solving the advection equation numerically in an atmospheric model has been extensively studied during the past decades and various schemes have been proposed. Semi-Lagrangian advection schemes have been widely incorporated into numerical models for atmospheric flows. Surveys of such a topic were given by Staniforth and Côté (1991) and Smolarkiewicz and Pudykiewicz (1992).

For an advection equation, semi-Lagrangian methods that trace back along the characteristics in time depend on an interpolation of the initial profile to determine the value at the upstream departure point (which may not coincide with computational grid point). Among the most well-known interpolators are various polynomial functions such as linear, quadratic Lagrange, cubic Lagrange, cubic spline, and quintic Lagrange (Staniforth and Côté 1991). However, all of these schemes (except those using the linear interpolation function) need at least three points in constructing interpolation approximations in one dimension and more than four (nine) mesh cells in two (three) dimensions. A more compact scheme by which one can construct interpolation function of high accuracy with less computational stencils is desirable in many situations, such as the calculations of discontinuities or of large gradients. As mentioned by Ritchie (1985), semi-Lagrangian schemes based on a compact support appear to be able to localize dispersion errors to the regions where large local gradients occur, for example, the moisture fields around fronts. In a model with a limited computational domain, different approximations for the derivatives have to be used at points close to boundaries; these approximations are usually of lower order than the approximations used deeper in the interior. Thus, a scheme that uses less stencils may be advantageous in treating computational boundaries since less boundary points need to be handled. Another attractive feature of reducing stencils may be the reduction in data transfer in parallel implementations on distributed memory architectures.

The cubic interpolated pseudoparticle (CIP) method, developed by Yabe and Aoki (1991) and Yabe et al. (1991), is constructed by a cubic polynomial function. The first-order spatial derivatives of the interpolation function are treated as dependent variables in the CIP method. The governing equations for those derivatives

Corresponding author address: Dr. Feng Xiao, Computational Science Laboratory, Institute of Physical and Chemical Research, Hirosawa 2-1, Wako, Saitama, 351-0198, Japan.
E-mail: xiao@atlas.riken.go.jp

are derived by differentiating the advection equation with respect to the spatial coordinates. This scheme is completely different from conventional semi-Lagrangian methods regarding the computation of derivatives. In the latter methods, as mentioned above, the gradient is calculated based on the function values at neighboring grid points by either assuming the continuity of the quantity, the first-and sometimes the second-order derivatives of the quantity at the mesh boundaries (Purnell 1976), or using approximations based on local grid points (Williamson and Rasch 1989). By special treatment of the first derivatives of the interpolation function, the CIP method achieves a compact form that uses only one mesh cell to construct the interpolation profile.

In practical implementation, on the other hand, an attractive advection scheme should be both less diffusive and free of oscillation. In Eulerian modern high-resolution schemes, manipulations, such as numerical viscosity, are usually taken to degrade the scheme to lower order in the presence of discontinuities in order to eliminate spurious oscillation. Some of these schemes are reviewed by Godlewski and Raviart (1996). Semi-Lagrangian schemes also have a problem in preserving the shape of the advected field. A scheme with an interpolator higher than second order will produce spurious oscillations near large gradients or discontinuities. Williamson and Rasch (1989) discussed several shape-preserving interpolation schemes. The monotonicity of a scheme is enhanced by imposing derivative constraints on a Hermite cubic or a rational cubic interpolation function. Bermejo and Staniforth (1992) also reported their work on overcoming the numerical oscillation for semi-Lagrangian schemes, where a minimum/maximum limiter is imposed on the calculated results from any conventional semi-Lagrangian scheme, based on the argument that no new extremum should be created by an advection scheme.

The original CIP method, which uses a cubic polynomial interpolant, produces numerical oscillations near the area where the dependent variable has a degree of smoothness no more than 1, for example, a step or a triangle distribution that is only C_0. In order to get a shape-preserving scheme for a CIP-type method, a numerical scheme for solving the advection equation was developed from a rational interpolation function over one mesh cell (Xiao et al. 1996a,b). This scheme shows good properties in keeping the topological nature of data, such as convexity or concavity preserving. Thus, it is attractive in suppressing overshoots and undershoots even in the vicinity of a discontinuity. It forms a basic formulation, which includes switching parameters to switch the interpolant between a cubic polynomial and a rational function and is designed so that several variants can be easily obtained with only minor modifications.

In this paper, I present a class of advection schemes that can be easily obtained from the basic formulation. The original CIP method is obtained by putting the switching parameters zero in the basic form. In such a case, a cubic polynomial interpolant is retrieved. When the switching parameters are valued 1, the interpolation function becomes a rational form. The resulting scheme at this time is called the rational function interpolation propagation (RIP) method. By devising automatic switching based on a measure of the field quantity smoothness, we get the rational-cubic interpolation propagation (RCIP) method that chooses the interpolant between a rational function and a cubic polynomial. A rigorous monotone scheme, the maximum and minimum bounded CIP (MmBCIP) scheme, is constructed by imposing upper and lower bounds to the computed solution from the CIP.

In section 2, the basic formulation of the advection schemes is first given. Variants of it are then proposed. Numerical tests that show the properties of the schemes are discussed in section 3. The computational effort required by a CIP-type scheme is compared with that of other conventional semi-Lagrangian methods by examining the arithmetic operation counts of both schemes, in section 4. A parallel implementation of the RCIP scheme is discussed in section 5 and section 6 gives a short summary.

2. The advection schemes

a. The basic formulation

The model equation to be solved is a 2D advection equation:

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} + \mathbf{u} \cdot \nabla \phi(\mathbf{x}, t) = 0, \tag{2.1}$$

where t refers to the time, $\mathbf{x}$ the spatial coordinate, $\mathbf{u} = [u(x, y, t), v(x, y, t)]$ the characteristic speed, and $\phi(x, y, t)$ the dependent variable. The gradient operator is defined as $\nabla = (\partial_x, \partial_y)$.

A semi-Lagrangian method usually makes use of the solution as a Lagrangian invariant,

$$\phi(\mathbf{x}, t) = \phi(\mathbf{x} - \hat{\mathbf{x}}, t - \Delta t), \tag{2.2}$$

where $\hat{\mathbf{x}}$ is the distance a particle travels in time increment Δt and is the solution of the following initial value problem:

$$\frac{d\mathbf{x}_1}{dt} = \mathbf{u}(\mathbf{x}_1, t), \tag{2.3}$$

$$\mathbf{x}_{1|t=t-\Delta t} = \mathbf{x}. \tag{2.4}$$

Generally, the velocity field $[u(x, y, t), v(x, y, t)]$ varies in time. The approximation of the trajectory equation (2.3), as concluded by Staniforth and Côté (1991), should have an accuracy of $O(\Delta t^2)$ and is usually achieved by an iterative procedure. In the current study, we focus on the spatial approximation and assume a time-independent velocity field. Thus, a simple estimate of the displacement $\hat{\mathbf{x}}$ is used as follows:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{u}(\mathbf{x})\Delta t. \tag{2.5}$$

The main effort in semi-Lagrangian methods is constructing the interpolation function based on grid values to determine the field value at a departure point not coinciding with a grid point. In order to get a highly accurate interpolant it is necessary to use not only the dependent variable itself but also its spatial derivatives, which in turn need to remain known at each step. In a conventional semi-Lagrangian scheme, the derivatives are generally estimated with the function values at grid points. This kind of treatment requires knowledge of more surrounding cells and it is not easy to make the resultant scheme more local.

A CIP-type method evaluates the first-order derivatives of the profile by treating the derivatives as additional dependent variables and adding the following equations, derived from Eq. (2.1) as

$$\frac{\partial(\partial_\chi \phi)}{\partial t} + \mathbf{u}\cdot\nabla(\partial_\chi\phi) = -\frac{\partial \mathbf{u}}{\partial \chi}\cdot\nabla\phi, \tag{2.6}$$

where χ denotes the space coordinate x or y and $\partial_\chi\phi$ represents $\partial\phi/\partial\chi$.

One then gets an additional advection equation in each dimension with a forcing term for the first-order derivative of that respective dimension. The set of governing equations for the two-dimensional advective problem is consequently written as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{U} = \mathbf{G}, \tag{2.7}$$

where

$$\mathbf{U} = \begin{pmatrix} \phi \\ \partial_x\phi \\ \partial_y\phi \end{pmatrix} \quad \text{and}$$

$$\mathbf{G} = \begin{pmatrix} 0 \\ -\partial_x u\partial_x\phi - \partial_x v\partial_y\phi \\ -\partial_y u\partial_x\phi - \partial_y v\partial_y\phi \end{pmatrix}.$$

A CIP-type scheme is equivalent to the following two-step semi-Lagrangian procedure:

$$\tilde{\mathbf{U}}(\mathbf{x}, t) = \mathbf{U}(\mathbf{x} - \mathbf{u}\Delta t, t - \Delta t) \quad \text{and} \tag{2.8}$$

$$\mathbf{U}(\mathbf{x}, t) = \tilde{\mathbf{U}}(\mathbf{x}, t) + \Delta t\mathbf{G}(\tilde{\mathbf{U}}). \tag{2.9}$$

Instead of a cubic polynomial as used in the original CIP method, a new scheme can be constructed by making use of a rational function.

As with any other semi-Lagrangian schemes, the scheme we will discuss in this paper can bear a large time step only if one tracks back, according to the foot trajectory, to the corresponding departure point and constructs the interpolation profile over the mesh cell covering the departure point. For the sake of simplicity and without losing generality, the following manipulations will be based on the neighboring cell of the grid point of interest.

We consider a two-dimensional rational interpolation function

$$R^{2D}_{i,j}(X, Y) = \left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}X^pY^q\right)^{-1}\left(\sum_{0\le l_x+l_y\le 3} C_{l_x,l_y}X^{l_x}Y^{l_y}\right) \tag{2.10}$$

over one mesh cell $[x_i, x_{i-\mathrm{isign}}] \times [y_j, y_{j-\mathrm{jsign}}]$ with isign and jsign being either 1 or -1 according to the orientation of the characteristics of the advection equation. Here, (X, Y) indicates the position of any point within the cell with $X = x - x_i$ and $Y = y - y_j$.

The indices, p, q and l_x, l_y are nonnegative integer pairs as shown in Table 1. The numerator of (2.10) is a cubic polynomial while the denominator is linear.

By differentiating with respect to spatial coordinates, we get

$$\partial_x R^{2D}_{i,j}(X, Y) \equiv \frac{\partial R^{2D}_{i,j}(X, Y)}{\partial x} = \left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}X^pY^q\right)^{-1} \times \left[\sum_{0\le l_x+l_y\le 2} (l_x+1)C_{l_x+1,l_y}X^{l_x}Y^{l_y} - \alpha_{1,0}\beta_{1,0}\left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}X^pY^q\right)^{-1}\left(\sum_{0\le l_x+l_y\le 3} C_{l_x,l_y}X^{l_x}Y^{l_y}\right)\right] \quad \text{and} \tag{2.11}$$

$$\partial_y R^{2D}_{i,j}(X, Y) \equiv \frac{\partial R^{2D}_{i,j}(X, Y)}{\partial y} = \left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}X^pY^q\right)^{-1} \times \left[\sum_{0\le l_x+l_y\le 2} (l_y+1)C_{l_x,l_y+1}X^{l_x}Y^{l_y} - \alpha_{0,1}\beta_{0,1}\left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}X^pY^q\right)^{-1}\left(\sum_{0\le l_x+l_y\le 3} C_{l_x,l_y}X^{l_x}Y^{l_y}\right)\right]. \tag{2.12}$$

TABLE 1. The indices involved in (2.10).

(p,q)	(0,0), (1,0), (0,1)
(l_x, l_y)	(0,0), (1,0), (0,1), (1,1), (2,0) (0,2), (2,1), (1,2), (3,0), (0,3)

Here $\alpha_{0,0} = 1$, and $\alpha_{1,0}$, $\alpha_{0,1} \in [0, 1]$ are the switching parameters that alternate the interpolant between a cubic polynomial and a rational function.

Supposing the original dependent variable $\phi_{i,j}$ and its derivatives $d_{xi,j}$ and $d_{yi,j}$ are given for all points of the grid, we can determine the unknown coefficients from the assumed continuity conditions at surrounding grid points. The coefficients $\{\beta_{p,q}\}$ are predetermined (Xiao et al. 1996b) as

$$\beta_{0,0} = 1.0, \tag{2.13}$$

$$\beta_{1,0} = [|(S_x - d_{xi,j})/(d_{xi-\text{isign},j} - S_x)| - 1]/\Delta_x, \tag{2.14}$$

$$\beta_{0,1} = [|(S_y - d_{yi,j})/(d_{yi,j-\text{jsign}} - S_y)| - 1]/\Delta_y, \tag{2.15}$$

where

$$\begin{cases} S_x = (\phi_{i-\text{isign},j} - \phi_{i,j})/\Delta_x, \\ S_y = (\phi_{i,j-\text{jsign}} - \phi_{i,j})/\Delta_y, \\ d_{xi,j} = \partial_x \phi_{i,j}, \\ d_{yi,j} = \partial_y \phi_{i,j}, \\ \Delta_x = x_{i-\text{isign}} - x_i, \\ \Delta_y = y_{j-\text{jsign}} - y_j. \end{cases}$$

The $\{C_{l_x,l_y}\}$ are calculated from the continuity conditions imposed on the interpolation function. From Table 1, 10 unknowns of $\{C_{l_x,l_y}\}$ need to be determined.

Considering an interpolation function over only one cell as indicated by $[x_i, x_{i-\text{isign}}] \times [y_j, y_{j-\text{jsign}}]$, we impose the continuity conditions at the vertices of the neighboring cell of grid (i, j) as follows:

$$\begin{cases} R^{2D}_{i,j}(x_i, y_j) = \phi_{i,j}, \\ \partial_x R^{2D}_{i,j}(x_i, y_j) = d_{xi,j}, \\ \partial_y R^{2D}_{i,j}(x_i, y_j) = d_{yi,j}, \\ R^{2D}_{i,j}(x_{i-\text{isign}}, y_j) = \phi_{i-\text{isign},j}, \\ \partial_x R^{2D}_{i,j}(x_{i-\text{isign}}, y_j) = d_{xi-\text{isign},j}, \\ \partial_y R^{2D}_{i,j}(x_{i-\text{isign}}, y_j) = d_{yi-\text{isign},j}, \\ R^{2D}_{i,j}(x_i, y_{j-\text{jsign}}) = \phi_{i,j-\text{jsign}}, \\ \partial_x R^{2D}_{i,j}(x_i, y_{j-\text{jsign}}) = d_{xi,j-\text{jsign}}, \\ \partial_y R^{2D}_{i,j}(x_i, y_{j-\text{jsign}}) = d_{yi,j-\text{jsign}}, \\ R^{2D}_{i,j}(x_{i-\text{isign}}, y_{j-\text{jsign}}) = \phi_{i-\text{isign},j-\text{jsign}}. \end{cases}$$

From Eqs. (2.10), (2.11), and (212) and the above continuity conditions, we have

$$C_{2,1} = [\alpha_{0,1}\beta_{0,1}\phi_{i-\text{isign},j} + (1 + \alpha_{1,0}\beta_{1,0}\Delta_x)d_{yi-\text{isign},j} - C_{0,1}] \div \Delta_x^2 - C_{1,1}/\Delta_x, \tag{2.16}$$

$$C_{1,2} = [\alpha_{1,0}\beta_{1,0}\phi_{i,j-\text{jsign}} + (1 + \alpha_{0,1}\beta_{0,1}\Delta_y)d_{xi,j-\text{jsign}} - C_{1,0}] \div \Delta_y^2 - C_{1,1}/\Delta_y, \tag{2.17}$$

$$\begin{aligned} C_{1,1} = {} & [\alpha_{0,1}\beta_{0,1}\phi_{i-\text{isign},j} + (1 + \alpha_{1,0}\beta_{1,0}\Delta_x)d_{yi-\text{isign},j}]/\Delta_x \\ & + [\alpha_{1,0}\beta_{1,0}\phi_{i,j-\text{jsign}} + (1 + \alpha_{0,1}\beta_{0,1}\Delta_y)d_{xi,j-\text{jsign}}]/\Delta_y \\ & + \left[C_{0,0} - \left(\sum_{0\le p+q\le 1} \alpha_{p,q}\beta_{p,q}\Delta_x^p\Delta_y^q\right)\phi_{i-\text{isign},j-\text{jsign}}\right] \\ & \div \Delta_x/\Delta_y + C_{3,0}\Delta_x^2/\Delta_y + C_{0,3}\Delta_y^2/\Delta_x + C_{2,0}\Delta_x/\Delta_y \\ & + C_{0,2}\Delta_y/\Delta_x, \end{aligned} \tag{2.18}$$

$$C_{3,0} = [(1 + \alpha_{1,0}\beta_{1,0}\Delta_x)(d_{xi-\text{isign},j} - S_x) + d_{xi,j} - S_x]/\Delta_x^2, \tag{2.19}$$

$$C_{0,3} = [(1 + \alpha_{0,1}\beta_{0,1}\Delta_y)(d_{yi,j-\text{jsign}} - S_y) + d_{yi,j} - S_y]/\Delta_y^2, \tag{2.20}$$

$$C_{2,0} = [(1 + \alpha_{1,0}\beta_{1,0}\Delta_x)\phi_{i-\text{isign},j} - C_{0,0} - C_{1,0}\Delta_x]/\Delta_x^2 - C_{3,0}\Delta_x, \tag{2.21}$$

$$C_{0,2} = [(1 + \alpha_{0,1}\beta_{0,1}\Delta_y)\phi_{i,j-\text{jsign}} - C_{0,0} - C_{0,1}\Delta_y]/\Delta_y^2 - C_{0,3}\Delta_y, \tag{2.22}$$

$$C_{1,0} = d_{xi,j} + \alpha_{1,0}\beta_{1,0}C_{0,0}, \tag{2.23}$$

$$C_{0,1} = d_{yi,j} + \alpha_{0,1}\beta_{0,1}C_{0,0}, \tag{2.24}$$

$$C_{0,0} = \phi_{i,j}. \tag{2.25}$$

An upwind orientation is defined by

$$\text{isign} = \begin{cases} -1, & u_{i,j} \le 0; \\ 1, & u_{i,j} > 0 \end{cases} \quad \text{and}$$

$$\text{jsign} = \begin{cases} -1, & v_{i,j} \le 0; \\ 1, & v_{i,j} > 0. \end{cases}$$

With the interpolation function and its derivatives being completely determined, the semi-Lagrangian solutions (2.8) and (2.9) can be easily computed, and we end up with the algorithm in 2D as

$$\phi^{n+1}_{i,j} = R^{2D}_{i,j}(\xi, \eta), \tag{2.26}$$

$$\partial_x\tilde{\phi}_{i,j} = \partial_x R^{2D}_{i,j}(\xi, \eta); \tag{2.27}$$

$$\partial_x\phi^{n+1}_{i,j} = \partial_x\tilde{\phi}_{i,j} - \Delta t\left(\frac{\partial \mathbf{u}}{\partial x}\cdot\nabla\tilde{\phi}\right)_{i,j}, \tag{2.28}$$

$$\partial_y\tilde{\phi}_{i,j} = \partial_y R^{2D}_{i,j}(\xi, \eta); \tag{2.29}$$

$$\partial_y\phi^{n+1}_{i,j} = \partial_y\tilde{\phi}_{i,j} - \Delta t\left(\frac{\partial \mathbf{u}}{\partial y}\cdot\nabla\tilde{\phi}\right)_{i,j}, \tag{2.30}$$

where $\xi = -u\Delta t$ and $\eta = -v\Delta t$.

This is the basic formulation from which a variety of individual schemes can be derived. We will define some subcases of (2.26)–(2.30) in the remaining parts of this section.

b. The CIP scheme

It is obvious that in the case of $\alpha_{1,0} = \alpha_{0,1} = 0$ the interpolation function (2.10) becomes the cubic polynomial used in the CIP scheme, and consequently the scheme retrieves the CIP method.

c. The RIP scheme

The scheme is called RIP when $\alpha_{1,0} = \alpha_{0,1} = 1$ are set in (2.10)–(2.12). In this case a full rational function is used as the interpolant.

d. The RCIP scheme

Rather than fixing the switching parameters $\alpha_{1,0}$ and $\alpha_{0,1}$ in advance, we devise an automatic switching based on detecting the smoothness of the profile (Xiao et al. 1996b).

The switching parameters in 2D are controlled by the following smoothness monitoring expressions:

$$\alpha_{1,0} = \begin{cases} 1, & d_{xi,j} d_{xi-\text{isign},j} < 0; \\ 0, & d_{xi,j} d_{xi-\text{isign},j} \geq 0 \quad \text{and} \end{cases} \tag{2.31}$$

$$\alpha_{0,1} = \begin{cases} 1, & d_{yi,j} d_{yi,j-\text{jsign}} < 0; \\ 0, & d_{yi,j} d_{yi,j-\text{jsign}} \geq 0, \end{cases} \tag{2.32}$$

where d_x and d_y represent the derivatives of ϕ in the x and y directions, respectively. It means that we use the cubic polynomial in a smooth region and switch to a rational function when a local extremum is detected. We call a scheme with these switchings [(2.31)–(2.32)] the RCIP method.

e. The MmBCIP scheme

As discussed by Bermejo and Staniforth (1992), a monotone solution can easily be obtained by bounding the numerical solution with the upper and lower limits in the related domain from the previous step. It stems from the common understanding that an advection process should not create any new local extremum. Suppose $\phi_{i,j}^{n+1}$ is the newly computed value from a semi-Lagrangian method with a high-order interpolation function, the monotonization procedure modifies $\phi_{i,j}^{n+1}$ as

$$\begin{aligned} \phi_{i,j}^{n+1} &= \phi_{\max}^{n} \quad \text{if } \phi_{i,j}^{n+1} \geq \phi_{\max}^{n}, \\ \phi_{i,j}^{n+1} &= \phi_{\min}^{n} \quad \text{if } \phi_{i,j}^{n+1} \leq \phi_{\min}^{n}, \\ \phi_{i,j}^{n+1} &= \phi_{i,j}^{n+1}, \quad \text{otherwise,} \end{aligned} \tag{2.33}$$

where

$$\phi_{\max}^{n} = \max(\phi_{i,j}^{n}, \phi_{i-\text{isign},j}^{n}, \phi_{i,j-\text{jsign}}^{n}, \phi_{i-\text{isign},j-\text{jsign}}^{n}) \quad \text{and}$$

$$\phi_{\min}^{n} = \min(\phi_{i,j}^{n}, \phi_{i-\text{isign},j}^{n}, \phi_{i,j-\text{jsign}}^{n}, \phi_{i-\text{isign},j-\text{jsign}}^{n}).$$

The scheme is called the MmBCIP method if the CIP method is incorporated with the monotonization procedure (2.33).

TABLE 2. Grid refinement analysis of the schemes. The convergence ratio is defined by $\text{CVG}_{\text{ratio}} = e_{1/40}/e_{1/80}$.

Scheme	$e_{1/40}$	$e_{1/80}$	$\text{CVG}_{\text{ratio}}$
First order	0.0393	0.0302	1.3013
Lax–Wendroff	0.0324	0.0176	1.8409
CIP	0.0124	0.0057	2.1754
RIP	0.0184	0.0087	2.1149
RCIP	0.0139	0.0060	2.3137
MmBCIP	0.0169	0.0075	2.2533

3. Numerical tests

The convergence behavior of the schemes was investigated with a solid rotating problem on a unit square centered at the origin of coordinates. The initial profile is defined by the following continuous function:

$$\phi(x, y, 0) = e^{-25|\mathbf{r}-\mathbf{r}_0|}, \tag{3.1}$$

where $|\mathbf{r} - \mathbf{r}_0| = [(x - x_0)^2 + (y - y_0)^2]^{1/2}$ and $(x_0, y_0) = (-0.3, 0.0)$. A time-independent velocity field is given by

$$(u, v) = (\omega y, -\omega x).$$

The angular velocity $\omega = 1$. As mentioned in section 2, the departure point is assumed to be in a cell next to the grid point (i, j) in the current study. Thus the time increment is restricted for computational stability. However, as with any other semi-Lagrangian scheme, this restriction can be eliminated by just considering a departure point far away from grid point (i, j) and constructing the interpolation over the cell where the departure point falls. The time increment used in the following calculations is $\Delta t = 2\pi\omega/480$, and 480 time steps are required for one revolution.

An l_2 norm error of the computed result at any instant t_n is defined as

$$e_h = \left[\sum_{i=1}^{I} \sum_{j=1}^{J} (\phi_{i,j}^{n} - \Phi_{i,j}^{n})^2 \Delta_x \Delta_y \right]^{1/2}, \tag{3.2}$$

where ϕ is the numerical solution and Φ the exact solution. The I and J denote the total numbers of grid points in the x and y directions, respectively. In addition, Δ_x and Δ_y denote the mesh spacings. A uniform mesh $\Delta_x = \Delta_y = h$ is used in the calculations.

Table 2 shows the errors measured by e_h for the solutions after one revolution with different grid resolutions ($h = 1/40$, $h = 1/80$). For the purpose of comparison, we also included the first-order upstream method and the Lax–Wendroff method.

The first-order upstream method, which heavily diffuses the solution, shows the lowest convergence ratio as the grid is doubly refined. The Lax–Wendroff scheme on the other hand has second-order accuracy in space and shows a convergence ratio of 1.8. The error from the Lax–Wendroff scheme is mainly caused by the remarkable dispersion incorrectness as discussed by Takacs (1985). All the schemes introduced in the present

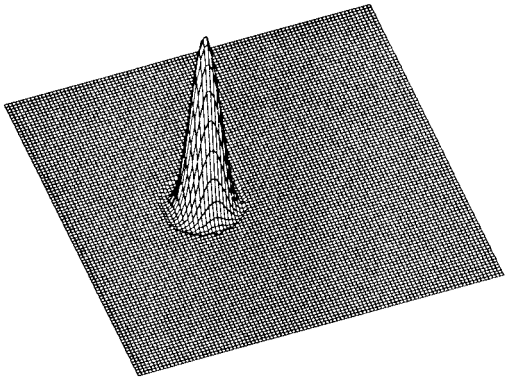

FIG. 1. The rotating cone test problem computed by the CIP method.

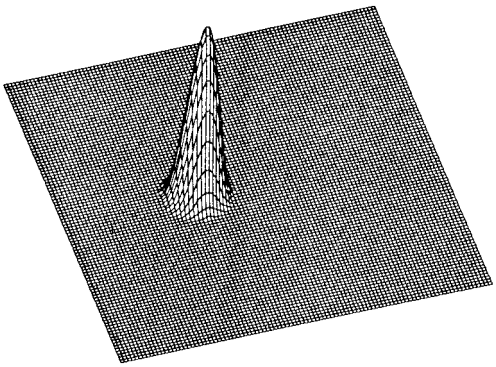

FIG. 3. Same as Fig. 1 but by the RCIP method.

paper have convergence ratios higher than 2. The RCIP method gives the best value of 2.3. Not only superior in global accuracy and convergence speed, the present schemes, as can be seen from the following numerical tests, also appear attractive in conserving the shape of the advected quantity and being accurate with phase velocity.

Following Takacs (1985) and Bermejo and Staniforth (1992), a measure of error was defined as

$$E_{\mathrm{TOT}} = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J}(\phi_{i,j}^{n} - \Phi_{i,j}^{n})^2, \tag{3.3}$$

where IJ is the total number of grid points.

It is easy to know

$$E_{\mathrm{TOT}} = E_{\mathrm{DISS}} + E_{\mathrm{DISP}} \tag{3.4}$$

with

$$E_{\mathrm{DISS}} = [\sigma(\phi) - \sigma(\Phi)]^2 + (\overline{\phi} - \overline{\Phi})^2 \quad \text{and}$$

$$E_{\mathrm{DISP}} = 2(1 - \rho)\sigma(\phi)\sigma(\Phi).$$

Here, $\overline{\phi}$ and $\sigma(\phi)$ represent the mean and the standard deviation of ϕ; ρ is the correlation coefficient between the values of ϕ and Φ on the grid points. In order to compare the performance of the schemes in mass conservation and the numerical oscillation, the rate of the total mass at instant t against its initial value RFM $= \int \phi(t)/\int \phi(0)$, the maximum and the minimum of the computed values are inspected as well.

The schemes are tested with a rotating cone problem (Bermejo and Staniforth 1992). The grid spacing is $h = 1/100$, a cone is initially centered at $(-0.14, 0)$. The height of the cone is 1 and the radius of its bottom circle is $8h$. The results after 480 steps are plotted in Fig. 1 for CIP, Fig. 2 for RIP, Fig. 3 for RCIP, and Fig. 4 for MmBCIP. For all schemes the sharp peak is satisfactorily preserved. This is simply because the schemes presented here are highly compact in space discretization. The continuity of the piece-wise interpolation function is assumed for only one cell, which makes it quite easy for the resultant schemes to fit a sharp corner without clipping it. A more careful inspection can be made with Table 3. As can be expected, the CIP method causes the largest undershoot, but has the smallest numerical diffusion. Being not exactly monotone, the RIP and RCIP do not get rid of spurious oscillation completely, but the rational function tends to produce an oscillation-free profile due to its convexity-preserving nature (Xiao et al. 1996a). The undershooting is reduced by one order for the RCIP and more than two orders for the RIP. The latter appears more diffusive. Bounding the upper and lower values of the solution, the MmBCIP does not give any undershoot, but results in the most diffused cone

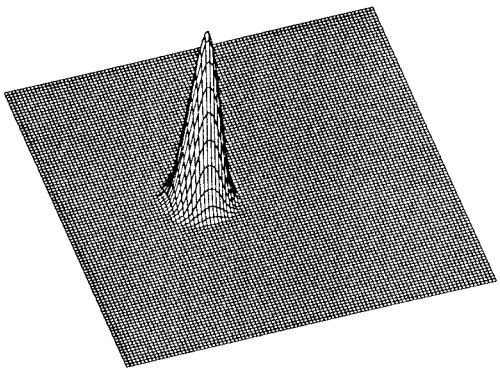

FIG. 2. Same as Fig. 1 but by the RIP method.

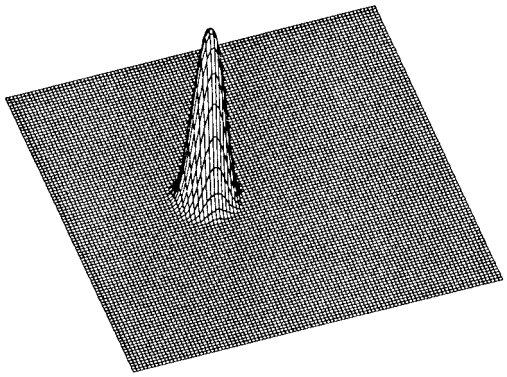

FIG. 4. Same as Fig. 1 but by the MmBCIP method.

TABLE 3. Numerical properties of different schemes for the rotating cone problem.

Scheme	RFM	Max	Min	E_{DISS}	E_{DISP}	E_{TOT}
CIP	0.9223	0.8496	-1.0207×10^{-2}	9.1675×10^{-6}	5.7041×10^{-5}	6.6209×10^{-5}
RIP	0.9781	0.7995	-7.5285×10^{-5}	2.4976×10^{-5}	6.1116×10^{-5}	8.6092×10^{-5}
RCIP	0.9469	0.8438	-1.9308×10^{-3}	9.6917×10^{-6}	5.6118×10^{-5}	6.5810×10^{-5}
MmBCIP	0.9369	0.7805	0.0000	1.5686×10^{-5}	5.7061×10^{-5}	7.2747×10^{-5}

peak. Similar to other semi-Lagrangian methods, none of these schemes conserve the total mass exactly. All of the schemes show a loss of advected mass. The CIP and RCIP result in dissipation errors one or two factors smaller than the other two, but do not show remarkable superiority regarding the dispersion errors. All the schemes appear competitive with conventional semi-Lagrangian methods in terms of computational accuracy.

Another test problem is a rotating slotted cylinder (Bermejo and Staniforth 1992), which presents a more discontinuous profile. As with the above example, the grid spacing is $h = 1/100$. A slotted cylinder with a radius of $15h$ is initially centered at (0.23, 0). The slot is $6h$ in width and $22h$ in depth. The numerical results are given in Figs. 5 through 8 and Table 4. Each scheme shows some similar behavior to the previous example. Without any restriction on monotonicity, the CIP scheme produces the most oscillatory profile. The RIP scheme, which is based on a full rational function, gives a more diffused result, with the overshoot reduced by an order of one and the undershoot by orders of more than three. The RCIP, taking a compromise between CIP and RIP, gives a solution that is less oscillatory than CIP and less diffusive than RIP. The MmBCIP, as ex-

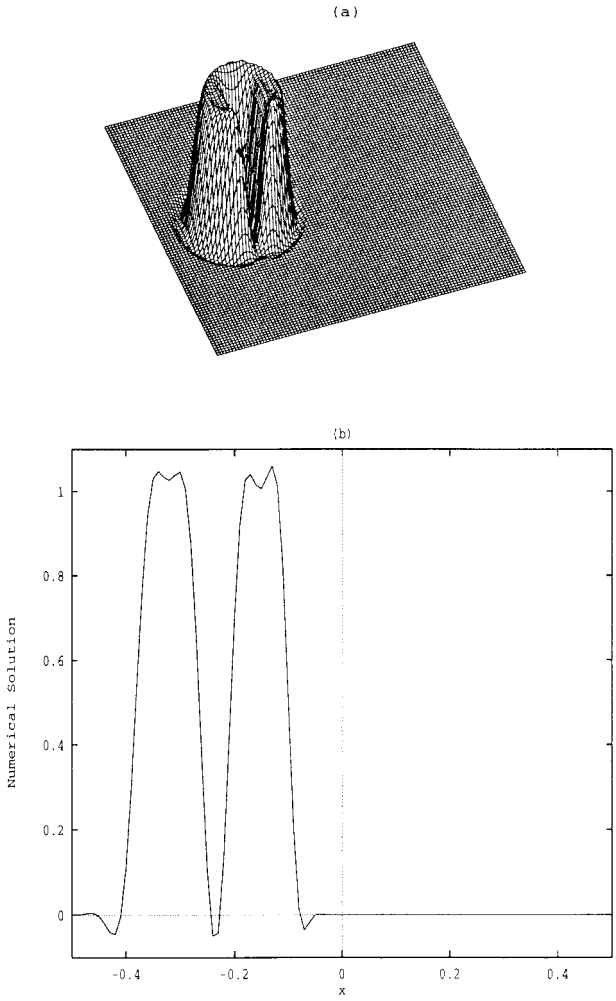

FIG. 5. The slotted cylinder problem computed by the CIP method. (a) A bird's-eye view. (b) A cross section at $y = 0$.

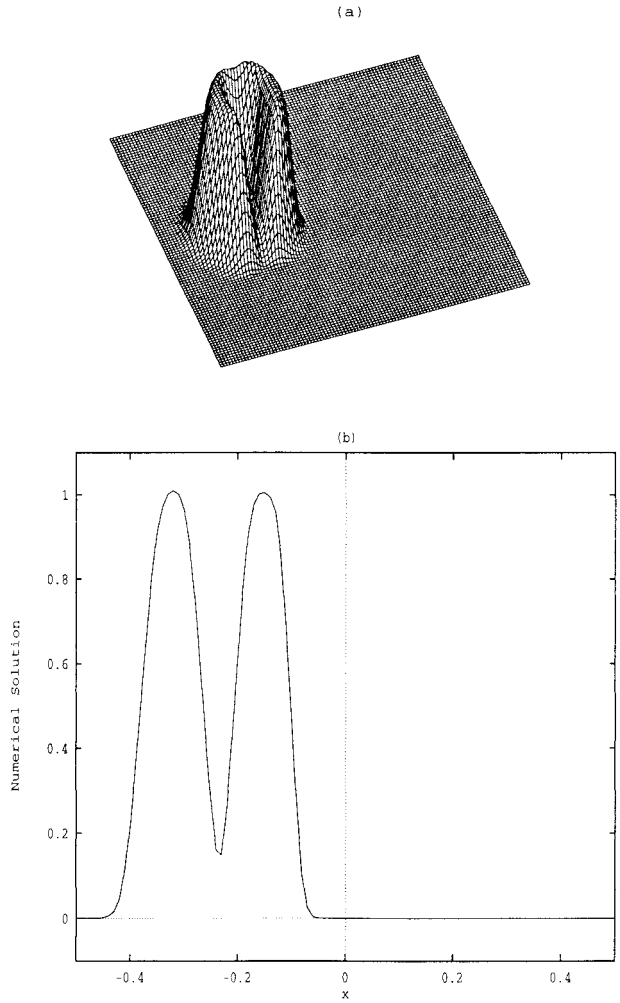

FIG. 6. Same as Fig. 5 but by the RIP method. (a) A bird's-eye view. (b) A cross section at $y = 0$.

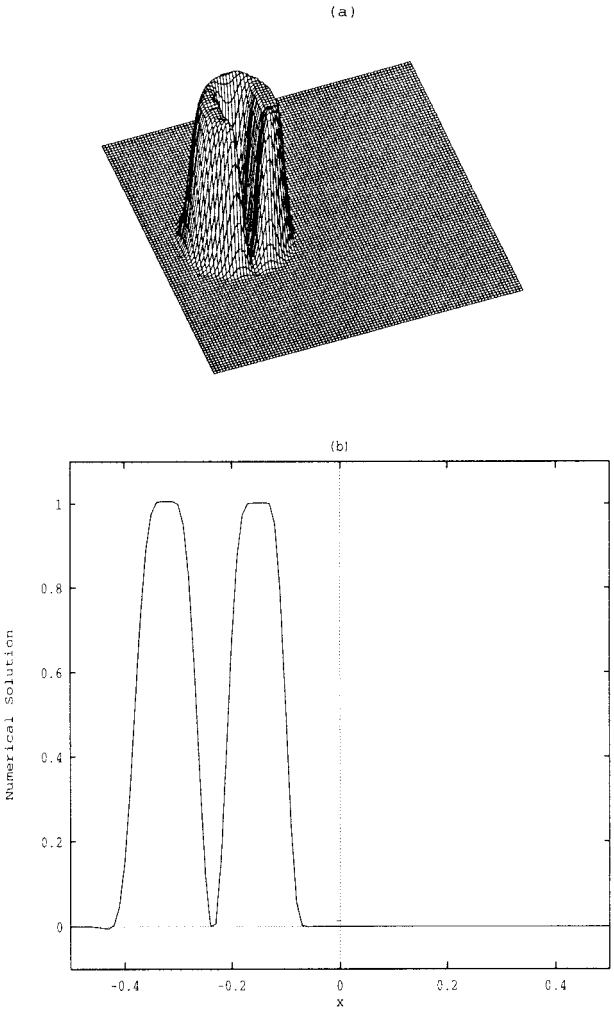

FIG. 7. Same as Fig. 5 but by the RCIP method. (a) A bird's-eye view. (b) A cross section at $y = 0$.

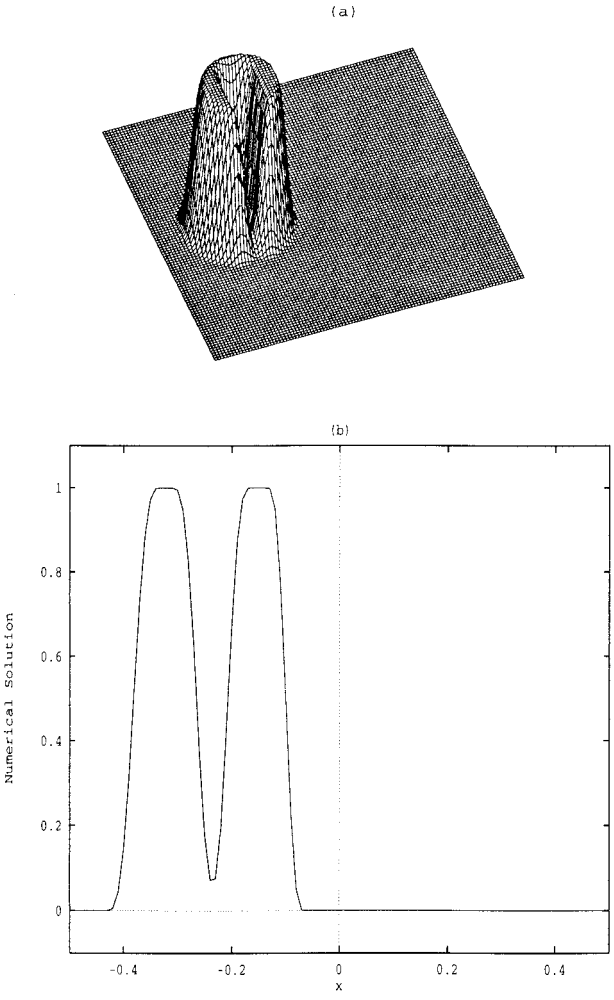

FIG. 8. Same as Fig. 5 but by the MmBCIP method. (a) A bird's-eye view. (b) A cross section at $y = 0$.

pected, limits the solution exactly between the initial maximum and minimum. None of the schemes show much difference in the l_2 norm error. RIP appears more erroneous in dissipation.

From the above discussions, we see that the semi-Lagrangian schemes presented in this work appear accurate for at least the numerical problems tested and should potentially be applicable to atmospheric dynamics. In case the numerical oscillation becomes a problem for the simulation, the RCIP or the MmBCIP may be considered as the favorable candidates.

4. Remarks on the computational effort of CIP-type schemes

A CIP-type scheme can be classified as one based on a Hermite interpolation. The key point that makes a CIP-type method different from the conventional Hermite interpolation based method is the way in which the derivatives are estimated. A conventional semi-Lagrangian scheme in a Hermite form, as extensively discussed by Williamson and Rasch (1989), calculates the derivatives of the interpolation function from the values at the surrounding grid points. A CIP-type method, however,

TABLE 4. Numerical properties of different schemes for the rotating slotted cylinder problem.

Scheme	RFM	Max	Min	E_{DISS}	E_{DISP}	E_{TOT}
CIP	0.9205	1.1120	-5.112×10^{-2}	6.0101×10^{-4}	6.6181×10^{-3}	7.2191×10^{-3}
RIP	0.9287	1.0172	-4.0103×10^{-5}	1.7000×10^{-3}	7.2017×10^{-3}	8.9017×10^{-3}
RCIP	0.9195	1.0426	-7.9230×10^{-4}	1.0728×10^{-3}	6.2945×10^{-3}	7.3673×10^{-3}
MmBCIP	0.9261	1.0000	0.0000	1.1058×10^{-3}	6.4201×10^{-3}	7.5259×10^{-3}

makes use of the Lagrangian solution of the advection as well to shift the derivatives. Thus, the difference of computational efforts between a CIP scheme and a conventional semi-Lagrangian scheme comes only from the derivative estimation.

For the sake of simplicity but without losing generality, we consider a one-dimensional semi-Lagrangian scheme using the following cubic polynomial interpolant,

$$F_i^{1D}(x) = \sum_{0 \le l_x \le 3} C_{l_x} X^{l_x}, \tag{4.1}$$

where $X = x - x_i$ and $X \in [x_i, x_{i+1}]$.

The resultant scheme from the basic formulation discussed in section 2 is then the CIP method.

We list below outlines of the computing operation sequences for a conventional semi-Lagrangian scheme and for the CIP scheme.

The conventional scheme

With $\{f_i^n\}$ known,

- * estimate the derivative d_i using the grid values of $\{f_i^n\}$ [among the most popular formulas for such a computation are the cubic spline (Purnell 1976), cubic Lagrange (McDonald 1984), Hyman derivative estimate (Hyman 1983), and Akima derivative estimate (Akima 1970)],
- compute $C_0 = f_i$,
- compute $C_1 = d_i$,
- compute $C_2 = 3s_i/\Delta_i - (2d_i + d_{i+1})/\Delta_i$,
- compute $C_3 = (d_i + d_{i+1})/\Delta_i^2 - 2s_i/\Delta_i^2$,
- advance the dependent variable by

$$f_i^{n+1} = f(x_i, t_{n+1}) = F_i^{1D}(x_i - u\Delta t)$$
$$= C_0 + C_1\xi + C_2\xi^2 + C_3\xi^3, \quad \text{then}$$

- go to next loop.

The CIP scheme

With both $\{f_i^n\}$ and $\{d_i^n\}$ known,

- compute $C_0 = f_i$,
- compute $C_1 = d_i$,
- compute $C_2 = 3s_i/\Delta_i - (2d_i + d_{i+1})/\Delta_i$,
- compute $C_3 = (d_i + d_{i+1})/\Delta_i^2 - 2s_i/\Delta_i^2$,
- advance the dependent variable by

$$f_i^{n+1} = f(x_i, t_{n+1}) = F_i^{1D}(x_i - u\Delta t)$$
$$= C_0 + C_1\xi + C_2\xi^2 + C_3\xi^3, \quad \text{then}$$

- * predict the derivative d_i using a Lagrangian solution and correct the derivative according to the velocity field by

$$d_i^* = \partial_x f_i^{n+1} = \partial_x F_i^{1D}(x_i - u\Delta t)$$
$$= (C_1 + 2C_2\xi + 3C_3\xi^2) \quad \text{and}$$
$$d_i^{n+1} = d_i^*\left[1 - \Delta t\left(\frac{\partial u}{\partial x}\right)_i\right], \quad \text{and}$$

- go to next loop.

Here, $d_i = [dF_i^{1D}(x_i)]/dx$, $s_i = (f_{i+1} - f_i)/\Delta_i$, and $\xi = -u\Delta t$.

We observe from the above lists that the difference in computational expenses stems from the first step of a conventional Hermite interpolation-based method and the last step of a CIP procedure, as indicated by the asterisks. For this simple case, it is possible to estimate the computational consumption for the two types of schemes by counting the arithmetic operations. Taking a computer architecture based on the reduced instruction set computer (RISC) methodology into account, we distinguish the arithmetic instructions of addition, multiplication, and subtraction from those of division.

Provided that we use central differencing to calculate the velocity gradient, the derivative is computed in the CIP method by

$$d_i^* = (C_1 + 2C_2\xi + 3C_3\xi^2) \quad \text{and} \tag{4.2}$$
$$d_i^{n+1} = d_i^*\left[1 - \Delta t\left(\frac{u_{i+1} - u_{i-1}}{\Delta_i}\right)\right]. \tag{4.3}$$

From Eqs. (4.2) and (4.3), we know that 11 operations of addition, multiplication, and subtraction, and one count of division, are needed for calculating the derivative at one grid point in the CIP method.

In a semi-Lagrangian method based on a conventional Hermite interpolation, the derivative d_i is calculated via a formula using grid values of f. Assuming a uniform grid spacing ($\Delta_i = x_{i+1} - x_i = \Delta$ = constant), we have the cubic derivative estimate, for example, as

$$d_i = \frac{-2f_{i-1} - 3f_i + 6f_{i+1} - f_{i+2}}{6\Delta}. \tag{4.4}$$

The Hyman derivative estimate, on the other hand, reads

$$d_i = \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta}. \tag{4.5}$$

It is obvious that estimating d_i requires seven operations of addition, multiplication, and subtraction, and one operation of division, using the cubic derivative estimate, while the Hyman formula requires six operations of addition, multiplication, and subtraction, and one count of division. Therefore, we know that a CIP-type scheme does not create a significant rise in arithmetic instructions compared with conventional semi-Lagrangian schemes based on a Hermite interpolation procedure. For a varying grid spacing, the above cubic derivative estimate and Hyman estimate need more op-

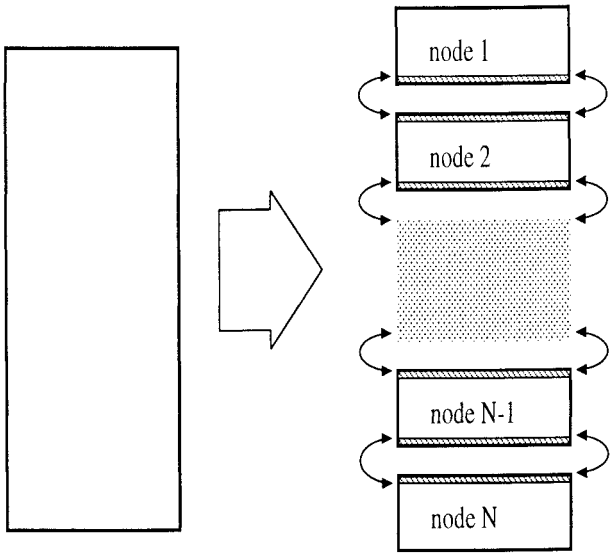

FIG. 9. Domain decomposition based on a one-dimensional partition. The shaded areas indicate the boundary cells where interprocessor message passing takes place.

eration counts, while no extra increase in computational cost is needed for the CIP method since the CIP scheme requires just one cell to construct the interpolation. When a rational function is used for the interpolation, as with the basic formulation discussed in the current paper, a modest rise in operation count is required, but it still appears comparable to those monotonicity-enhanced schemes discussed by Williamson and Rasch (1989) in terms of computational efficiency.

We should note, however, that a CIP-type scheme needs to save all of the derivatives as dependent variables and therefore requires extra memory space. This might become a problem for hardware that has limited memory and necessitates further investigation toward saving memory.

5. Parallel implementation

We have implemented a parallelization of the RCIP method on an IBM RS/6000 SP at the Institute of Physical and Chemical Research, Japan (RIKEN). The RS/6000 SP at RIKEN has 11 RISC processor nodes. Each node consists of four-way 332-MHz SMP of PowerPC 604e. All the nodes are connected by a fast network, SP switch, which provides a bidirectional data-transfer rate of up to 160 MB s^{-1}. Each 332-MHz SMP node has 256 Mb of memory.

The code was parallelized in a single program multiple data fashion with message passing. In order to achieve load balance, the 2D computational domain was decomposed equally into several subdomains, as shown in Fig. 9. The data in each subdomain were ported to one individual node. Communications are required when a subdomain on each processor node needs its neighbors' boundary information. Since one cell is needed to construct the interpolation for the schemes discussed in this paper, only one line of data along each boundary needs to be exchanged. An MPI library was used for intertask communication. The nonblocking message passing routines, MPI_ isend and MPI_irecv, were used to exchange the boundary data.

Provided that computation is conducted with an $L \times L$ grid on N nodes, we observe that the amount of data to be communicated is in proportion to $W_T = 3L \times 2(N - 1)$ and the data to be computed are in proportion to $W_C = L \times L$. Thus, the execution time of a parallel run can be estimated by

$$t_e(L, N) = a\frac{W_C}{N} + b\frac{W_T}{N} = a\frac{L^2}{N} + 6bL\left(1 - \frac{1}{N}\right). \quad (5.1)$$

The constants a and b depend on the hardware and can be evaluated from two actual runs. We used the elapsed time from the runs on 2 and 10 nodes with a fixed size of 600×600 and solved the constants from (5.1) as

$$a = 3.842 \times 10^{-6}, \quad \text{and} \quad b = 3.610 \times 10^{-6}.$$

The elapsed time estimated from (5.1) is plotted in Fig. 10 against those from actual runs. We find that (5.1) gives a good estimate for the execution time of a parallel implementation.

Figure 11 displays the actual parallel speedup for problems of different sizes. The computation becomes linearly scalable as the problem size increases.

From (5.1), the parallel efficiency can be evaluated by

$$\frac{t_e(L, 1)}{t_e(L, N) \times N} = \frac{1}{1 + \dfrac{6b}{aL}(N - 1)}. \quad (5.2)$$

Equation (5.2) enables us to predict the parallel efficiency for various problem sizes and different numbers of processor nodes. Figure 12 shows the efficiency of parallel implementation as estimated by (5.2). We observe that for a cluster with fixed nodes, an increase in problem size rapidly improves the parallel efficiency. For a problem of large size, there is no noticeable difference in parallel efficiency between a 2-node cluster and a 10-node cluster. Equation (5.1) also demonstrates that the CPU time increases at an order of $O(L^2)$, while the communication overhead increases at an order of $O(L)$. Thus, a better parallel performance can be obtained for a problem of a larger size with fixed processor nodes.

A rough estimation of scalability for the parallel implementation in a massively parallel processing (MPP) environment can also be obtained from Eq. (5.1). Supposing there are two problems of different sizes L_1^2 and L_2^2 to be solved on clusters of N_1 and N_2 nodes, respectively, and $1 \ll N_1 < N_2$, we consider the cases where $aL_1^2/N_1 = aL_2^2/N_2 = T$. The increase in the elapsed

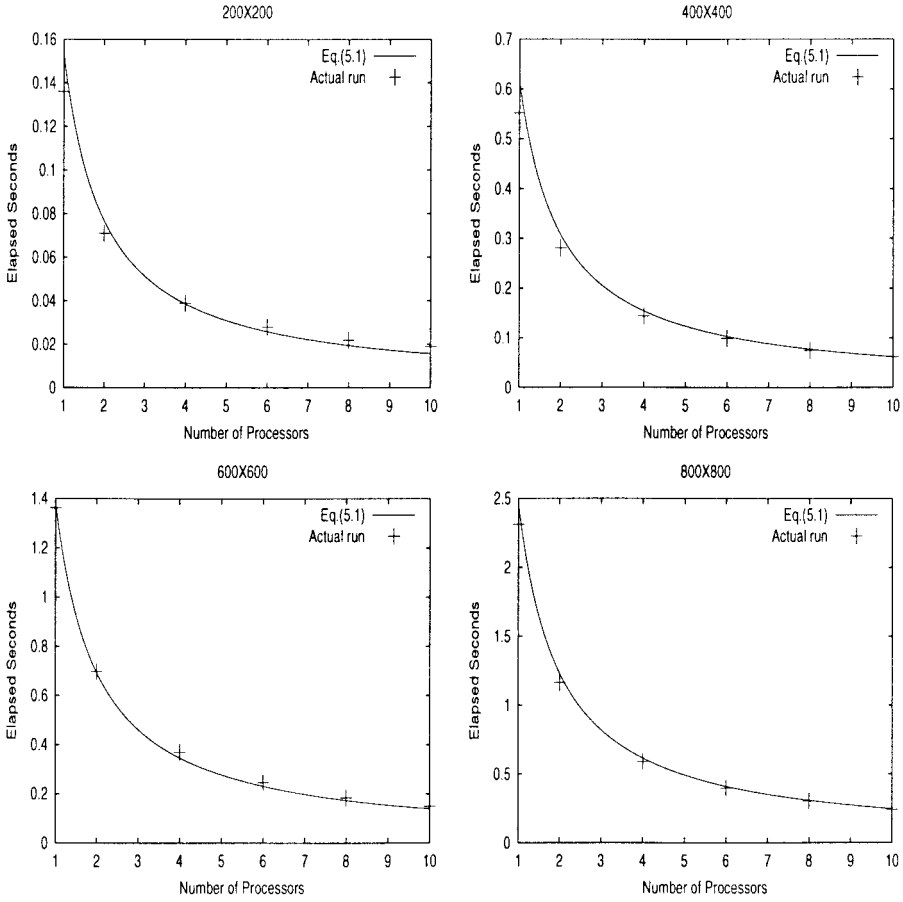

FIG. 10. Comparisons between the actual elapsed time and that estimated by (5.1).

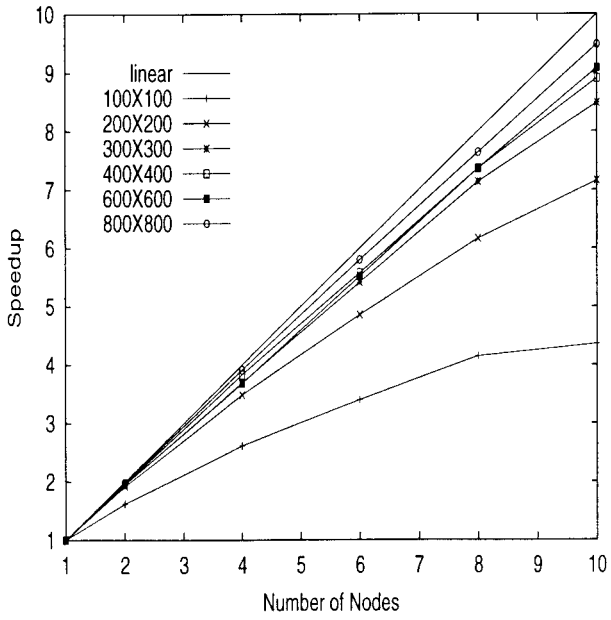

FIG. 11. Variation of speedup with number of nodes.

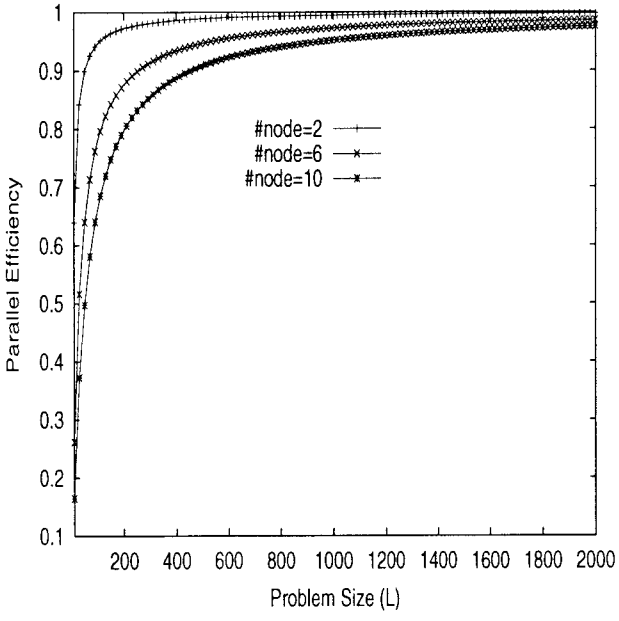

FIG. 12. Efficiency of parallel implementation predicted from (5.2).

time (second) of the two cases is then calculated by (5.1) as

$$\Delta T = t_e(L_2, N_2) - t_e(L_1, N_1) \approx 2.166 \times 10^{-5} (L_2 - L_1).$$

Thus, when scaling the problem size with the number of processor nodes, the relative rise in elapsed time caused by communication overhead, $\Delta T/T$, is not significant if the computational time T is at an order larger than a millisecond, and the scheme can be expected to perform efficiently in an MPP computation.

For a CIP-type scheme, high locality is achieved at the expense of a rise in memory used for storing the derivatives as dependent variables. Thus, in 2D cases, the data to be transfered across a boundary are $3L$, where L represents the number of the grid points along the boundary. Estimating a derivative in a conventional Hermite interpolation-based scheme, on the other hand, requires at least four stencils of grid for a formula having an accuracy higher than third order, as can be seen from Eqs. (4.4) and (4.5). In such a case, no less than three inner lines next to the boundary are involved in data communications. Therefore, for at least a 2D computation, we can expect that a CIP-type scheme is competitive with a conventional semi-Lagrangian scheme using Hermite interpolation of the same order of accuracy, in terms of communication efficiency in parallel processing. We should mention that a CIP-type scheme in 3D needs to store the derivatives in three directions and $4L$ data needs to be transfered, and further comparisons with other Hermite interpolation schemes are required to evaluate the efficiency of the data transfer of the method.

6. Summary

A class of semi-Lagrangian schemes is presented. Making use of the Lagrangian invariant solution to not only the dependent variable but also its first-order spatial derivatives results in a highly compact numerical formulation. The interpolation procedure can be done over only one mesh cell, and minor modifications are easily implemented to make the scheme less diffusive or monotone. The present studies show that the schemes discussed in this paper appear competitive with other semi-Lagrangian methods with respect to both accuracy and computational efficiency. We believe that these types of schemes can be utilized in atmospheric simulations as practical numerical solvers.

Acknowledgments. The author thanks the anonymous reviewers for their constructive suggestions, which substantially improved the appearance of this paper. The author gratefully acknowledge Prof. T. Yabe for helpful discussions and C. W. Stern for a linguistic check of the manuscript.

REFERENCES

Akima, H., 1970: A new method of interpolation and smooth curve fitting based on local procedures. *J. Assoc. Comput. Mach.,* **17,** 589–602.

Bermejo, R., and A. Staniforth, 1992: The conversion of semi-Lagrangian advection schemes to quasi-monotone schemes. *Mon. Wea. Rev.,* **120,** 2622–2632.

Godlewski, E., and P. Raviart, 1996: *Numerical Approximation of Hyperbolic System of Conservation Laws.* Springer, 509 pp.

Hyman, J. M., 1983: Accurate monotonicity preserving cubic interpolations. *SIAM J. Sci. Stat. Comput.,* **4,** 645–654.

McDonald, A., 1984: Accuracy of multiply-upstream, semi-Lagrangian advection schemes. *Mon. Wea. Rev.,* **112,** 1267–1275.

Purnell, D. K., 1976: Solution of the advective equation by upstream interpolation with a cubic spline. *Mon. Wea. Rev.,* **104,** 42–48.

Ritchie, H., 1985: Application of a semi-Lagrangian integration scheme to the moisture equation in a regional forecast model. *Mon. Wea. Rev.,* **113,** 424–435.

Smolarkiewicz, P. K., and J. A. Pudykiewicz, 1992: A class of semi-Lagrangian approximations for fluids. *J. Atmos. Sci.,* **49,** 2082–2096.

Staniforth, A., and J. Côté, 1991: Semi-Lagrangian integration scheme for atmospheric models—A review. *Mon. Wea. Rev.,* **119,** 2206–2223.

Takacs, L. L., 1985: A two-step scheme for advection equation with minimized dissipation and dispersion errors. *Mon. Wea. Rev.,* **113,** 1050–1065.

Williamson, D. L., and P. J. Rasch, 1989: Two-dimensional semi-Lagrangian transport with shape-preserving interpolation. *Mon. Wea. Rev.,* **117,** 102–129.

Xiao, F., T. Yabe, and T. Ito, 1996a: Constructing oscillation preventing scheme for advection equation by rational function. *Comput. Phys. Commun.,* **93,** 1–12.

——, ——, G. Nizam, and T. Ito, 1996b: Constructing a multi-dimensional oscillation preventing scheme for advection equation by rational function. *Comput. Phys. Commun.,* **94,** 103–118.

Yabe, T., and T. Aoki, 1991: A universal solver for hyperbolic-equations by cubic-polynomial interpolation. 1. One-dimensional solver. *Comput. Phys. Commun.,* **66,** 219–232.

——, T. Ishikawa, P. Y. Wang, T. Aoki, Y. Kadota, and F. Ikeda, 1991: A universal solver for hyperbolic-equations by cubic-polynomial interpolation. 2. 2-dimensional and 3-dimensional solvers. *Comput. Phys. Commun.,* **66,** 233–242.