JOURNAL OF COMPUTATIONAL PHYSICS 111, 24–32 (1994)

Flux-Corrected Transport in a Moving Grid*

KAMBIZ SALARI

Ecodynamics Research Associates, Inc., P.O. Box 8172, Albuquerque, New Mexico 87198

AND

STANLY STEINBERG

Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131

Received August 10, 1991; revised February 25, 1993

A flux-corrected transport algorithm in moving grids is developed and applied to transport problems involving solution adapted grids. The success of the algorithm is demonstrated numerically, as are certain limitations. © 1994 Academic Press, Inc.

1. INTRODUCTION

In this paper, a numerical algorithm for tracking one-dimensional waves is developed. The algorithm is a generalization of the flux-corrected transport (FCT) algorithms developed by Boris and Book [4], Zalesak [26], and Kunhardt and Wu [10]. The FCT algorithm is generalized to moving grids, where the grid motion is determined by a solution-adapted grid algorithm (for information on adaptive grids, see Anderson [1], Anderson and Rai [2], Dwyer, Kee, and Sanders [6], and Dwyer, Smooke, and Kee [7]).

Many of the cited authors have shown that the FCT algorithm is capable of tracking a wave whose motion is governed by the one-way wave equation (see Eq. (2.1) below). This algorithm is particularly good at tracking wave fronts without introducing spurious oscillations. However, various versions of the FCT algorithm have some problems with general waves; the algorithm tends to clip extrema and introduce staircases on the sloped portions of the wave (see Leonard [11]). Nevertheless, the usual FCT algorithms are very useful. Note that the techniques developed in this paper will apply to a wide range of algorithms that are suitable for tracking wave fronts.

Previously, Oran and Boris [17] developed a moving-grid FCT algorithm called LCPFCT. Although the main ideas used to develop LCPFCT and our algorithms are similar, many of the details are different. In addition, LCPFCT was applied to problems quite different than the applications described in this paper.

Adaptive grid-generation algorithms can be used to provide higher resolution of near wave fronts, thus reducing the number of points needed to track the wave and, hopefully, moderating the problems that the FCT algorithm has with general waves. Wave fronts are given by discontinuities in the solution or the derivative of the solution of the one-way wave equation, so the weights in the adaptive FCT algorithm are determined by using the first and second derivatives of the function giving the wave displacement, then the grid is compressed at jumps or steep transitions in the wave, where the first derivative is large or infinite, and at sharp corners of the wave, where the second derivative is large or infinite.

The new algorithm is called the FCT in moving grids algorithm (FCTMG). It is very successful at tracking wave fronts, as demonstrated by the data in Fig. 1. The calculations displayed in this figure were made with 101 points (including boundary points) or 100 cells, in the interval $0 \leqslant x \leqslant 100$. The wave speed is $u = 0.2$, the time at which the plots are made is $t = 800$, while the time step is chosen adaptively as 0.7 times the minimum time step which maintains the local CFL number less than one. Figure 1a shows a calculation that was done with a uniform grid. The spatial resolution that was chosen is coarse enough to show the FCT algorithm having difficulties maintaining the shape of the wave. Figure 1b shows a calculation with the adapted grid. The FCTMG algorithm has no difficulty maintaining the wave form with the same number of points that were used in the previous calculation. Figure 1c shows the ratio of the adapted grid length to the uniform grid length. In this case, the adaptivity is very precise. For more information on the choice of adaptive weights, see Section 5.

Leonard [11] notes that "semi-ellipse is particularly

* This work was partially supported by DoE through grants from Sandia National Laboratory.

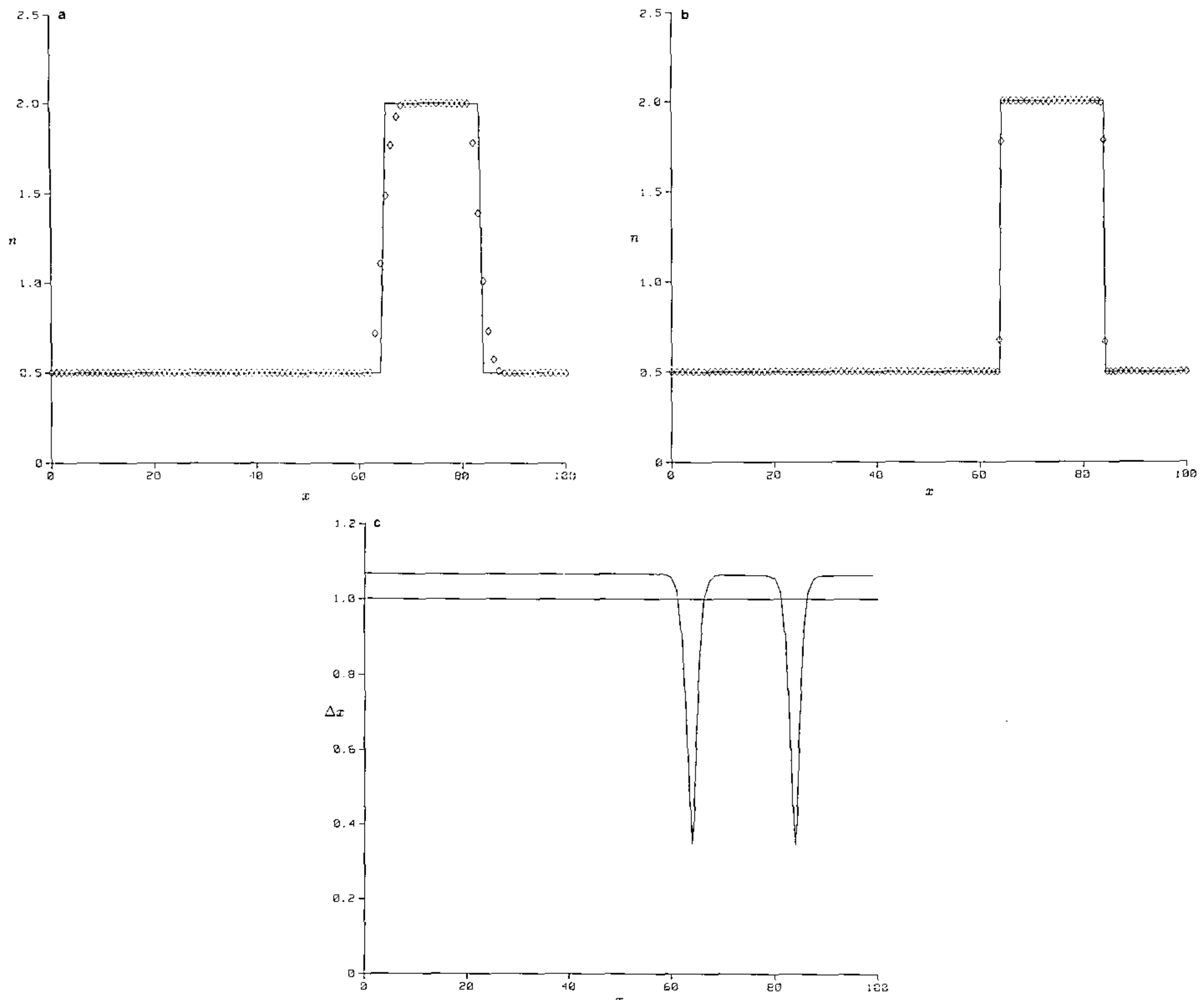

FIG. 1. (a) FCT on a square wave. (b) FCTMG on a square wave. (c) Adapted grid.

challenging because of its combination of sudden and gradual changes in gradient." Figure 2 shows that the solution adaptivity exacerbates the problems the FCT has with this wave. The data for these calculations is the same as the data used in Fig. 1. Figure 2a shows a semi-ellipse or parabolic profile being tracked in a uniform grid; note the flattening of the wave just forward of the maximum and the staircase further down on the wave front. Figure 2b shows an adapted-grid result which is clearly inferior to the uniform-grid result. We attribute the problems to the nonlinear interactions of the higher-order method in the FCT algorithm with the adaptivity functionals that is apparent from the grid spacing shown Fig. 2c. Thus, the adaptive algorithm must be used with some care.

Many numerical tests were run; Fig. 1 and 2 summarize the important aspects of these tests. Even with its problems, the FCTMG algorithm is superior to many standard algorithms. In fact, the FCTMG algorithms was applied to a gas-discharge problem [19], producing better results than those given in Morrow [13], Steinle and Morrow [22], and Steinle, Morrow, and Roberts [23].

The FCT algorithm has three main steps: (1) calculate a diffusive solution; (2) calculate a higher-order solution; and (3) make a flux correction using a nonlinear filter. The new algorithm starts with the generation of a solution-adaptive grid. If one travels with a wave front, then the solution adaptive grid appears to flow into the front from the down-wind side and away from the front on the up-wind side of

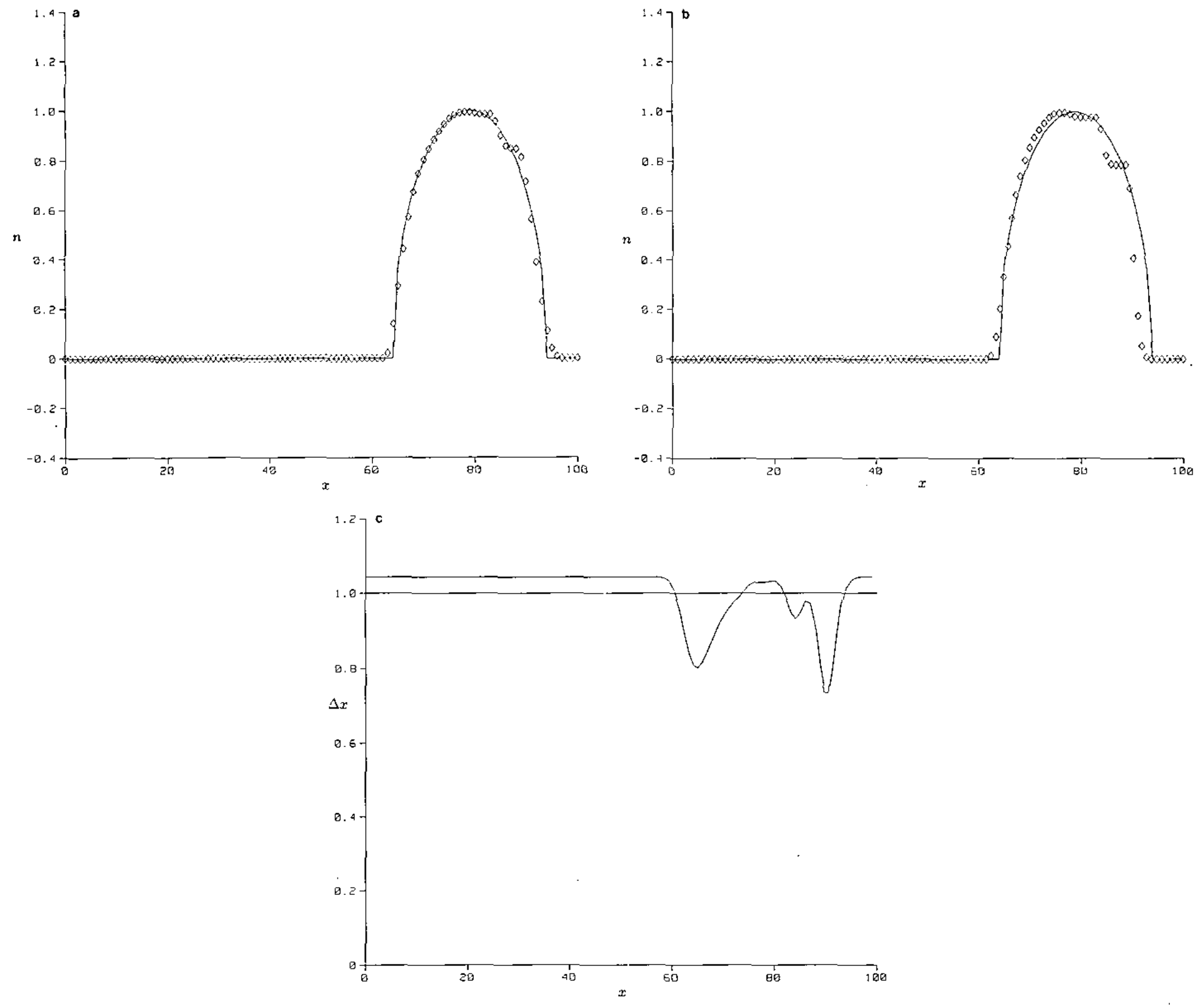

FIG. 2. (a) FCT on a parabolic wave. (b) FCTMG on a parabolic wave. (c) Adapted grid.

the front. This typically strong relative motion of the grid with respect to the front requires that some care be used when developing a FCTMG algorithm. To solve a wave equation on a nonuniform moving grid, the wave equation is transformed to general coordinates, called logical coordinates, and then the transformed equation is discretized. The transformed equation has the same form as the original equation; therefore, in principle, the FCT algorithm can be applied to the logical-space equation. In fact, there are a number of subtle points. Keep in mind that it is the properties of the wave in physical space that have direct physical and geometric meaning. Consequently, if some important property is not preserved under the transformation and discretization, then problems should be expected.

The first difficulty occurs in step (1). Consider the situation where the velocity of a wave is constant. Then, the wave just translates. If the wave height is constant, then this is a solution to the one-way wave equation. However, such a wave is not a solution of the discretized equations that are used in step (1), unless the differences of the coordinates of the grid satisfy an identity called the metric identity. Moreover, if this identity is not satisfied, then the diffusive approximation that is used in logical space is not always smooth in physical space. In such circumstances, the FCT algorithm does not work. On the other hand, the new algorithm is not sensitive to the details of step (2), the computation of the higher-order solution. In step (3), the filter involves geometric properties of the wave, while the algorithm involves logical space quantities. As a consequence, the new algorithm uses a combination of physical-space and logical-space quantities. The FCTMG algorithm is sensitive to the choices made in step (3) of the algorithm.

In fact, implementing finite-difference algorithms in non-uniform or time-dependent grids has a number of subtleties (see Thomas and Lombard [24], Vinokur [25], or Obayashi [16]). Thus, some difficulties should be expected when implementing an FCT algorithm in such grids. Note that Morrow and Cram [14] previously implemented the FCT algorithm in a non-uniform mesh. Rather than using a moving mesh, a local grid-refinement approach could be used. However, such algorithms are far more complicated than the algorithm studied here (see Berger and Colella [5]). Also, the FCTMG algorithm compares well to the moving finite-element method (see Gelinas, Doss, and Miller [8] and Miller [12]). In particular, [8, p. 218] compares the Lax–Wendroff, leapfrog, donor cell, and reversible FCT algorithms to the moving finite-element method. Kansa [9] implemented a shock-capturing scheme in a moving grid, where the grid motion is determined by having the solution variables stationary in the least squares sense.

Because of the clipping and staircase problems, it makes sense to consider other algorithms. In fact, some significant improvements for the FCT algorithm were made in the paper [15] by McDonald and Ambrosiono. Moreover, there are other competing algorithms such as TVD, Godunov, MUSCL, PPM, and ENO (see Przekwas and Yang [18], Leonard [11], and Shu and Osher [20] for more information). Przekwas and Yang [18] performed extensive numerical experiments to compare many variants of such one-dimensional, advanced computational fluid dynamics algorithms. We are currently in the process of implementing and testing a number of these schemes.

The implementation strategy used here should work for all such advanced computational fluid dynamics algorithms; it has worked well for the ones that we have implemented.

In Section 2, the one-way wave equation is written in general coordinates, while in Section 3, the grid layout is given. The FCTMG algorithm is constructed in Section 4, and the grid adaptivity algorithm is described in Section 5.

2. THE MODEL PROBLEM IN GENERAL COORDINATES

The initial value problem for the one-way wave equation in one space variable x and time variable t is given by

$$\frac{\partial n}{\partial t}+\frac{\partial (un)}{\partial x}=0, \qquad n(x,0)=g(x), \qquad 0\leqslant x\leqslant A, \quad t\geqslant 0, \tag{2.1}$$

where A is a given constant. The initial value problem (2.1) is solved for the density $n=n(x,t)$ when the velocity $u=u(x,t)$ and the initial density $g(x)$ are given. To have a well-posed problem, a boundary condition must be given, for example, $n(0,t)=0$ if $u(x,t)>0$. Boundary conditions do not play an important role in this paper.

To apply solution-adapted grid-generation techniques, the initial-value problem (2.1) is transformed to a general moving coordinate system. The transformation has the form

$$\tau=t, \qquad \xi=\xi(x,t). \tag{2.2}$$

The x, t-space is called *physical* space, while the ξ, τ-space is called *logical* space. The inverse of the transformation is given by

$$t=\tau, \qquad x=x(\xi,\tau). \tag{2.3}$$

Note that the adaptive grid generation algorithm produces the inverse transformation. Therefore, quantities involving the transformation must, eventually, be eliminated in favor of quantities determined from the inverse transformation.

The chain rule is used to represent the physical derivatives $\partial_t=\partial/\partial t$ and $\partial_x=\partial/\partial x$ in terms of logical derivatives $\partial_\tau=\partial/\partial\tau$ and $\partial_\xi=\partial/\partial\xi$:

$$\begin{bmatrix}\partial_t\\ \partial_x\end{bmatrix}=\begin{bmatrix}1 & \xi_t\\ 0 & \xi_x\end{bmatrix}\begin{bmatrix}\partial_\tau\\ \partial_\xi\end{bmatrix}. \tag{2.4}$$

The Jacobian, that is, the determinant of the Jacobian matrix of the transformation is simply given by

$$J=\xi_x. \tag{2.5}$$

The chain rule can be used to write the derivatives of the transformation in terms of the inverse transformation:

$$\xi_t=-\frac{x_\tau}{x_\xi}, \qquad \xi_x=\frac{1}{x_\xi}. \tag{2.6}$$

The grid *velocity* is given by x_τ.

Applying the chain rule (2.4) to the one-way wave equation (2.1) gives

$$\partial_\tau n+\xi_t\partial_\xi n+\xi_x\partial_\xi(nu)=0, \qquad n(\xi,0)=g(\xi), \tag{2.7}$$

where $n=n(\xi,\tau)=n(x(\xi,\tau),\tau)$, $u=u(\xi,\tau)=u(x(\xi,\tau),\tau)$, and $g(\xi)=g(x(\xi,0))$.

To retain the discontinuity capturing properties of the usual FCT algorithm, Eq. (2.7) should be put into conservation form (see Anderson, Tannehil, and Pletcher [3] and Steinberg and Roache [21] for more details). First, both sides of the chain-rule form of the one-way wave equation (2.7) are multiplied by J^{-1}. Then, the chain-rule (2.4) is applied to all terms, producing

$$\partial_\tau\frac{n}{J}+\partial_\xi\frac{nU}{J}-n\left[\partial_\tau\frac{1}{J}+\partial_\xi\frac{\xi_t}{J}\right]+nu\partial_\xi\frac{\xi_x}{J}=0, \tag{2.8}$$

where

$$U = \xi_t + u\xi_x; \qquad (2.9)$$

U is called the contravariant velocity. Equation (2.6) can be used to eliminate the derivatives of the transformation in favor of derivatives of the inverse transformation in Eq. (2.8). Then, the equality of mixed-partial derivatives of the transformation implies that all but the first two terms in (2.8) cancel giving the *conservation* form of the one-way wave equation:

$$\partial_\tau\left(\frac{n}{J}\right) + \partial_\xi\left(\frac{nU}{J}\right) = 0. \qquad (2.10)$$

This argument does not involve the derivatives of the solution which may be discontinuous. If the new dependent variables

$$N = n/J, \qquad F = NU, \qquad (2.11)$$

are introduced, then the conservation form of the one-way wave equation can be put into the compact form

$$\partial_\tau N + \partial_\xi F = 0. \qquad (2.12)$$

3. BASIC GRID LAYOUT

The computational region in logical space is given by $0 \leqslant \xi \leqslant N$, $\tau \geqslant 0$, where N is a positive integer. The grid that is listed in Table I and shown in Fig. 3 is placed on logical space. Because finite-volume arguments are used, logical space is divided into cells (see Fig. 4). Also, the Euler or leapfrog scheme is used in time, so nodes are needed at the time-like cell-face centers, again, see Fig. 4.

It is assumed that the transformation $\xi(x, t)$ maps the interval $0 \leqslant x \leqslant A$ to the interval $0 \leqslant \xi \leqslant N$ for each value of $t \geqslant 0$. Recall that the grid generator produces the inverse transformation. It is assumed that the generator gives the coordinates of the nodes: x_i^j, $0 \leqslant i \leqslant N$, $j \geqslant 0$. The points in physical space are given by

$$x = x_i^j, \qquad t = j\,\Delta t, \qquad 0 \leqslant i \leqslant N, \quad j \geqslant 0. \qquad (3.13)$$

TABLE I

Space-Time Grid

	ξ	τ		$j \geqslant$
Nodes	i	j	$0 \leqslant i \leqslant N$	0
Cell corners	$i+1/2$	j	$0 \leqslant i \leqslant N-1$	0
Time-like face centers	$i+1/2$	$j+1/2$	$0 \leqslant i \leqslant N-1$	0
Space-like face centers	i	j	$0 \leqslant i \leqslant N$	0
Cell centers	i	$j+1/2$	$0 \leqslant i \leqslant N-1$	0

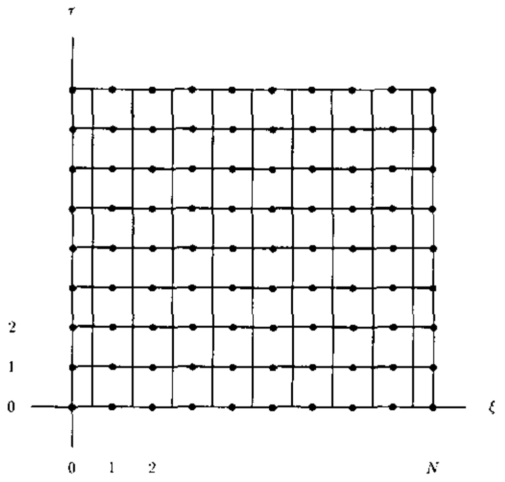

FIG. 3. Logical space grid.

Transformed points are computed at a half-index point by averaging:

$$\begin{aligned} x_{i+1/2}^j &= \frac{x_{i+1}^j + x_i^j}{2}, \\ x_i^{j+1/2} &= \frac{x_i^{j+1} + x_i^j}{2}, \\ x_{i+1/2}^{j+1/2} &= \frac{x_{i+1}^{j+1} + x_{i+1}^j + x_i^{j+1} + x_i^j}{4}. \end{aligned} \qquad (3.14)$$

In one dimension, it is convenient to use unit steps ($\Delta\xi = 1$) in logical space. In other circumstances, it is convenient to have a general $\Delta\xi$. In the remainder of this paper, it is not assumed that $\Delta\xi = 1$. If I and J are either integer or half-integer indices, then the derivatives of the inverse of the transformation are computed using

$$(x_\xi)_I^J = \frac{x_{I+1/2}^J - x_{I-1/2}^J}{\Delta\xi}, \qquad (x_\tau)_I^J = \frac{x_I^{J+1/2} - x_I^{J-1/2}}{\Delta\tau}. \qquad (3.15)$$

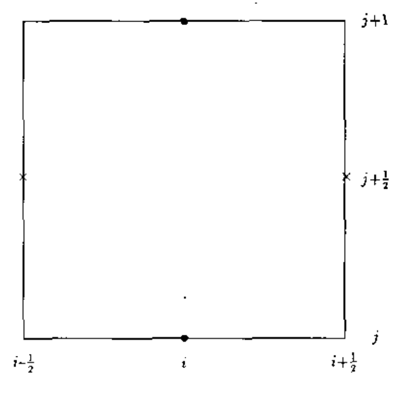

FIG. 4. Generic grid cell.

The derivatives of the transformation are computed using

$$(\xi_x)_i^j = \frac{1}{(x_\xi)_i^j}, \qquad (\xi_t)_i^j = \frac{(x_\tau)_i^j}{(x_\xi)_i^j}, \qquad J_i^j = \frac{1}{(x_\xi)_i^j}. \tag{3.16}$$

4. THE FCT ALGORITHM ON A MOVING GRID

The FCTMG algorithm is similar to the FCT algorithm described in Zalesak [26], except that the computations are done in logical space. The Zalesak algorithm, with certain modifications, is applied to Eq. (2.12), giving the formulas described below. In the model problem, it is assumed that u is given and n needs to be computed. (In more realistic problems, u is also computed.) Thus, u_i^j are the values of u at the node points, while $u_{i+1/2}^{j+1/2}$ are the values of u at the time-like face centers. The FCTMG algorithm uses three time steps. Thus, assume that x_i^j, x_i^{j-1}, n_i^j, and n_i^{j-1}, have been computed. The task is to compute x_i^{j+1} and n_i^{j+1}. The first step is to use a grid generator to compute x_i^{j+1}. The computation of n_i^{j+1} requires three steps: (1) compute a low-order diffusive approximation; (2) compute a high-order approximation; and (3) filter the approximations. The computation of the low-order approximation is particularly sensitive to geometric errors.

Since computations are all done using logical variables, the logical-space quantities need to be computed in terms of the physical-space quantities:

$$N_i^j = \frac{n_i^j}{J_i^j}; \qquad U_i^j = (\xi_t)_i^j + (\xi_x)_i^j u_i^j; \qquad F_i^j = N_i^j U_i^j. \tag{4.17}$$

In fact, the N_i^j, not the n_i^j, are stored by the code. Thus, the algorithm updates the logical space quantities and then transforms them to physical space using the inverse of Eq. (4.17):

$$n_i^j = J_i^j N_i^j; \qquad u_i^j = (x_\tau)_i^j + (x_\xi)_i^j U_i^j; \qquad f_i^j = n_i^j u_i^j. \tag{4.18}$$

Now the problem is to compute N_i^{j+1} given N_i^j and N_i^{j-1}.

4.1. *The Diffusive Difference Approximation*

First, a diffusive density $(N^D)_i^{j+1}$ is computed. The differential equation (2.10) is differenced as a flux-balance equation in a cell,

$$\frac{(N^D)_i^{j+1} - N_i^j}{\Delta\tau} + \frac{F_{i+1/2}^{j+1/2} - F_{i-1/2}^{j+1/2}}{\Delta\xi} = 0; \tag{4.19}$$

that is,

$$(N^D)_i^{j+1} = N_i^j - \frac{\Delta\tau}{\Delta\xi}\left(F_{i+1/2}^{j+1/2} - F_{i-1/2}^{j+1/2}\right). \tag{4.20}$$

Here N_i^j has already been computed and the flux F is computed using

$$F_{i+1/2}^{j+1/2} = N_{i+1/2}^{j+1/2} U_{i+1/2}^{j+1/2}, \tag{4.21}$$

where

$$U_{i+1/2}^{j+1/2} = (\xi_t)_{i+1/2}^{j+1/2} + (\xi_x)_{i+1/2}^{j+1/2} u_{i+1/2}^{j+1/2}, \qquad N_{i+1/2}^{j+1/2} = \frac{n_{i+1/2}^{j+1/2}}{J_{i+1/2}^{j+1/2}}. \tag{4.22}$$

If u is tabulated at the nodes, then the time-like face-center values of the velocity $u_{i+1/2}^{j+1/2}$ are computed using linear interpolation, while the time-like face-center densities $n_{i+1/2}^{j+1/2}$ are computed by using the upwind density:

$$\begin{aligned} n_{i+1/2}^{j+1/2} &= n_i^j \qquad \text{if} \quad U_{i+1/2}^{j+1/2} \geqslant 0, \\ n_{i+1/2}^{j+1/2} &= n_{i+1}^j \qquad \text{if} \quad U_{i+1/2}^{j+1/2} < 0. \end{aligned} \tag{4.23}$$

One of the goals is to have a constant n generate a solution to Eq. (4.20), (4.21), and (4.22) when u is constant. This is achieved by the careful differencing of the logical and physical quantities that are used in these equations.

4.2. *The Metric Identity*

It is critical that if u_i^j is a constant, then n_i^j constant is a solution to the difference equation. Let $u_i^j = u$. Then $u_{i+1/2}^{j+1/2} = u$ and $n_{i+1/2}^{j+1/2} = n$. The identities

$$\begin{aligned} \frac{1}{J_i^j} &= (x_\xi)_i^j, \\ \frac{(\xi_t)_{i+1/2}^{j+1/2}}{J_{i+1/2}^{j+1/2}} &= -(x_\tau)_{i+1/2}^{j+1/2}, \\ \frac{(\xi_x)_{i+1/2}^{j+1/2}}{J_{i+1/2}^{j+1/2}} &= 1 \end{aligned} \tag{4.24}$$

reduce the difference equation to $M_i^{j+1/2} = 0$, where

$$M_i^{j+1/2} = \frac{(x_\xi)_i^{j+1} - (x_\xi)_i^j}{\Delta\tau} - \frac{(x_\tau)_{i+1/2}^{j+1/2} - (x_\tau)_{i+1/2}^{j+1/2}}{\Delta\xi}, \tag{4.25}$$

when n_i^j and u_i^j are non-zero constants. Note that

$$M_i^{j+1/2} \approx x_{\xi\tau} - x_{\tau\xi} = 0, \tag{4.26}$$

and the equality of the mixed partial was used to put the differential equations in conservation form. Also, note that Eq. (4.25) depends only on Eqs. (4.24) and not on the details of the definitions of the various metric terms.

Now,

$$\begin{aligned}\Delta\tau\,\Delta\xi\,M_i^{j+1/2} = {} & +x_{i+1/2}^{j+1} - x_{i-1/2}^{j+1} \\ & - x_{i+1/2}^{j} + x_{i-1/2}^{j} \\ & - x_{i+1/2}^{j+1} + x_{i+1/2}^{j} \\ & + x_{i-1/2}^{j+1} - x_{i-1/2}^{j} \\ \equiv {} & 0. \end{aligned} \tag{4.27}$$

Thus, when the velocity is constant, constant density functions are solutions to the difference scheme. Again, note that this conclusion is independent of how the corner values $x_{i+1/2}^j$ of the transformation are defined as long as all other metric quantities are computed in terms of these corner quantities.

4.3. *The High-Order Approximation*

An intermediate higher-order approximation N^I of the density is computed with leapfrog differencing, that is, a flux-balance equation for a cell that is composed of two cells in the time direction and an intermediate higher-order approximation of the flux F^I. The flux-balance equation is

$$\frac{(N^I)_i^{j+1} - N_i^{j-1}}{2\Delta\tau} + \frac{(F^I)_{i+1/2}^j - (F^I)_{i-1/2}^j}{\Delta\xi} = 0, \tag{4.28}$$

which implies that

$$(N^I)_i^{j+1} = N_i^{j-1} - \frac{\Delta\xi}{2\Delta\tau}\left((F^I)_{i+1/2}^j - (F^I)_{i-1/2}^j\right). \tag{4.29}$$

Recall that the densities N_i^j and N_i^{j-1} and the fluxes $F_i^j = N_i^j U_i^j$ have already been computed. The higher-order flux is given by an interpolation

$$(F^I)_{i+1/2}^j = \tfrac{7}{12}\left((F^I)_{i+1}^j + (F^I)_i^j\right) - \tfrac{1}{12}\left(F_{i+2}^j + F_{i-1}^j\right). \tag{4.30}$$

This flux is a second-order approximation at the cell corners provided that the fluxes at the space-like face centers are exact. Moreover, $(F_{i+1/2}^j - F_{i-1/2}^j)/\Delta\xi$ is a fourth-order approximation of $\partial F/\partial\xi$ at a space-like cell-face center.

4.4. *The FCT Algorithm*

The next step in the FCT algorithm is to compute a "corrected" F^C that can be used in a one-cell flux-balance equation to calculate the final density N_{i+1}^{j+1} (see Fig. 4). The flux-balance equation is

$$\frac{N_i^{j+1} - N_i^j}{\Delta\tau} + \frac{(F^C)_{i+1/2}^{j+1/2} - (F^C)_{i-1/2}^{j+1/2}}{\Delta\xi} = 0. \tag{4.31}$$

If the corrected flux is written as a correction A to the diffusive flux

$$F^C = F^D + A^C, \tag{4.32}$$

then N_i^{j+1} can be computed by using

$$N_i^{j+1} = (N^D)_i^{j+1} - \frac{\Delta\tau}{\Delta\xi}\left((A^C)_{i+1/2}^{j+1/2} - (A^C)_{i-1/2}^{j+1/2}\right). \tag{4.33}$$

The correction A^C is calculated by first calculating a correction to the diffusive flux and then limiting that correction. A higher-order flux is computed by interpolating the existing fluxes to the time-like cell-face centers. First average the fluxes to the cell centers by using

$$(F^*)_i^{j+1/2} = \frac{F_i^j + F_i^{j+1}}{2} = \frac{N_i^j U_i^j + (N^I)_i^{j+1} U_i^{j+1}}{2}. \tag{4.34}$$

This is a natural generalization of the formula (A3) used in Zalesak [26, p. 361], but different from the third and fourth formulas on page 130 of Kunhardt and Wu [10]. The contravariant velocity U_i^{j+1} is computed with a second-order backward-difference approximation to $(x_\tau)_i^{j+1}$. As mentioned above, interpolate these quantities to the time-like cell-face centers,

$$\begin{aligned}(F^H)_{i+1/2}^{j+1/2} = {} & \tfrac{7}{12}\left((F^*)_{i+1}^{j+1/2} + (F^*)_i^{j+1/2}\right) \\ & - \tfrac{1}{12}\left((F^*)_{i+2}^{j+1/2} + (F^*)_{i-1}^{j+1/2}\right),\end{aligned} \tag{4.35}$$

and then define the correction by

$$A_{i+1/2}^{j+1/2} = (F^H)_{i+1/2}^{j+1/2} - (F^D)_{i+1/2}^{j+1/2}. \tag{4.36}$$

The main part of the FCT algorithm is the computation of a flux limiter $C_{i+1/2}^{j+1/2}$; this is described below. Once the flux limiter is computed, then limited flux correction A^C is given by

$$(A^C)_{i+1/2}^{j+1/2} = C_{i+1/2}^{j+1/2} A_{i+1/2}^{j+1/2}. \tag{4.37}$$

The Zalesak method of avoiding clipping by predicting maxima and minima between grid points is used in the flux limiting. Note that the properties of the physical-space density n, and not the logical-space density N need to be controlled. However, it is the contravariant velocities that determine wheter or not material flows into a cell. Consequently, the flux-limiter algorithm uses n and U as its primary variables. Thus, let

$$(n^D)_i^j = J_i^j (N^D)_i^j, \qquad n_i^j = J_i^j N_i^j, \tag{4.38}$$

and then the flux constraints are defined as follows.

If

$$A^{j}_{i+1/2}((n^D)^{j+1}_{i+1/2} - (n^D)^{j+1}_{i}) < 0, \tag{4.39}$$

and either

$$A^{j+1/2}_{i+1/2}((n^D)^{j+1}_{i+2} - (n^D)^{j+1}_{i+1}) < 0$$

or

$$A^{j+1/2}_{i+1/2}((n^D)^{j+1}_{i} - ((n^D)^{j+1}_{i-1}) < 0, \tag{4.40}$$

then set

$$A^{j+1/2}_{i+1/2} = 0. \tag{4.41}$$

Now the limiter coefficients $C^{j}_{i+1/2}$ are computed as follows. Set

$$\begin{aligned}(n^a)^{j+1}_i &= \max(n^j_i, (n^D)^{j+1}_i),\\ (n^{\max})^{j+1}_i &= \max((n^a)^{j+1}_{i-1}, (n^a)^{j+1}_i, (n^a)^{j+1}_{i+1}),\end{aligned} \tag{4.42}$$

$$\begin{aligned}(n^b)^{j+1}_i &= \min(n^j_i, (n^D)^{j+1}_i),\\ (n^{\min})^{j+1}_i &= \min((n^b)^{j+1}_{i-1}, (n^b)^{j+1}_i, (n^b)^{j+1}_{i+1}),\end{aligned} \tag{4.43}$$

and

$$\begin{aligned}P^+_i &= \max(0, A^{j+1/2}_{i-1/2}) - \min(0, A^{j+1/2}_{i+1/2}),\\ Q^+_i &= ((n^{\max})^{j+1}_i - (n^D)^{j+1}_i(x_\xi)^{j+1}_i \frac{\Delta\xi}{\Delta\tau},\end{aligned} \tag{4.44}$$

$$\begin{aligned}P^-_i &= \max(0, A^{j+1/2}_{i+1/2}) - \min(0, A^{j+1/2}_{i-1/2}),\\ Q^-_i &= ((n^D)^{j+1}_i - (n^{\min})^{j+1}_i)(x_\xi)^{j+1}_i \frac{\Delta\xi}{\Delta\tau}\end{aligned} \tag{4.45}$$

(see Kunhardt and Wu [10] for an alternative formulation). Finally,

$$\begin{aligned}R^+_i &= \begin{cases}\min(1, Q^+_i/P^+_i) & \text{if } P^+_i > 0\\ 0 & \text{if } P^+_i = 0,\end{cases}\\ R^-_i &= \begin{cases}\min(1, Q^-_i/P^-_i) & \text{if } P^-_i > 0\\ 0 & \text{if } P^-_i = 0,\end{cases}\end{aligned} \tag{4.46}$$

and then

$$C^{j+1/2}_{i+1/2} = \begin{cases}\min(R^+_{i+1}, R^-_i) & \text{if } A^{j+1/2}_{i+1/2} \geqslant 0\\ \min(R^+_i, R^-_{i+1}) & \text{if } A^{j+1/2}_{i+1/2} < 0.\end{cases} \tag{4.47}$$

In Eq. (4.44) and (4.45), the term $(x_\xi)^{j+1}_i\, \Delta\xi/\Delta\tau$ is used to improve the performance of the algorithm as the time and space steps are changed.

5. THE SOLUTION ADAPTIVE GRID

A time dependent solution adaptive grid is used to help resolve wave fronts (for more information on adaptive grids, see Anderson [1], Anderson and Rai [2], Dwyer, Kee, and Sanders [6], and Dwyer, Smooke, and Kee [7]). The grid is generated by solving the boundary value problem

$$x_{\xi\xi} + \frac{w_\xi}{w} x_\xi = 0, \qquad x(0) = 0, \qquad x(N) = L, \tag{5.48}$$

where L is the length of the physical region. This grid generator is used to make the lengths of the grid segments proportional to the weight function w. If the segments are proportional to the weight function, then in the limit,

$$x_\xi = Cw, \tag{5.49}$$

for some constant C. Dividing by this equation by w and then differentiating gives

$$\left(\frac{x_\xi}{w}\right)_\xi = 0 \tag{5.50}$$

which is equivalent to (5.48).

In this application, the weight function is selected to be

$$w(\xi) = 1 + a\,|n_\xi|^\alpha + b\,|n_{\xi\xi}|^\beta, \tag{5.51}$$

where a, b, α, and β are the constants used to control the amount of grid adaption (these constants are specified for each numerical experiment). The grid is adapted at each time step using the current values of n. For Figs. 1 and 2, $\alpha = \beta = 1$ and $b = 0$. In Fig. 1, $a = 3.0$, while in Fig. 2, $a = 5.0$. Experiments were run with many other values of these parameters; no clear conclusions could be made as to what constitutes the most effective values.

REFERENCES

1. D. A. Anderson, *Appl. Math. Comput.* **35**, 209 (1990).
2. D. A. Anderson and M. M. Rai, "The Use of Solution Adaptive Grids in Solving Partial Differential Equations," in *Numerical Grid Generation*, edited by J. F. Thompson (North-Holland, New York, 1982), p. 317.
3. D. A. Anderson, J. C. Tannehil, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer* (Hemisphere, New York, 1984).
4. J. P. Boris and D. L. Book, *J. Comput. Phys.* **20**, 397 (1976).
5. M. J. Berger and P. Colella, *J. Comput. Phys.* **82**, 64 (1989).
6. H. A. Dwyer, R. Kee, and B. Sanders, *AIAA J.* **18**, 1205 (1980).
7. H. A. Dwyer, M. D. Smooke, and R. Kee, "Adaptive Gridding for Finite Difference Solutions to Heat and Mass Transfer Problems, in *Numerical Grid Generation*, edited by J. F. Thompson (North-Holland, New York, 1982), p. 339.

8. R. J. Gelinas, S. K. Doss, and K. Miller, *J. Comput. Phys.* **40**, 202 (1981).
9. E. J. Kansa, *Comput. Math. Appl.* **15**, 623 (1988).
10. E. E. Kunhardt and C. Wu, *J. Comput. Phys.* **68**, 127 (1987).
11. B. P. Leonard, *Comput. Methods Appl. Mech. Eng.* **88**, 17 (1991).
12. K. Miller, *SIAM J. Numer. Anal.* **18**, 1033 (1981).
13. R. Morrow, *J. Comput. Phys.* **43**, 1 (1981).
14. R. Morrow and L. E. Cram, *J. Comput. Phys.* **57**, 129 (1985).
15. B. E. McDonald and J. Ambrosiano, *J. Comput. Phys.* **56**, 448 (1984).
16. S. Obayashi, NASA Contractor Report 177572, 1991 (unpublished).
17. E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flows* (Elsevier, New York, 1987).
18. A. J. Przekwas and H. Q. Yang, SBIR Phase I Final Report, 1989, CFD Research Corporation, 3313 Bob Wallace Avenue, Suite 205, Huntsville, AL, 35805.
19. A. E. Rodriguez, K. J. Touryan, W. M. Moeny, and W. L. Morgan, Final Contractor Report SAND91-7065, Sandia National Laboratories, Albuquerque, NM, 87185.
20. C. W. Shu and S. Osher, *J. Comput. Phys.* **83**, 32 (1989).
21. S. Steinberg and P. J. Roache, Differencing symmetric operators in general coordinates, in preparation.
22. P. Steinle and R. Morrow, *J. Comput. Phys.* **80**, 61 (1989).
23. P. Steinle, R. Morrow, and A. J. Roberts, *J. Comput. Phys.* **85**, 483 (1989).
24. P. D. Thomas and C. K. Lombard, *AIAA J.* **17**, 1030 (1978).
25. M. Vinokur, *J. Comput. Phys.* **81**, 1 (1989).
26. S. T. Zalesak, *J. Comput. Phys.* **31**, 335 (1979).