JOURNAL OF COMPUTATIONAL PHYSICS **100**, 143–162 (1992)

Momentum Advection on a Staggered Mesh*

DAVID J. BENSON

Department of AMES R-011, University of California, San Diego, La Jolla, California 92093

Received August 11, 1988; revised March 23, 1991

Eulerian and ALE (arbitrary Lagrangian–Eulerian) hydrodynamics programs usually split a timestep into two parts. The first part is a Lagrangian step, which calculates the incremental motion of the material. The second part is referred to as the Eulerian step, the advection step, or the remap step, and it accounts for the transport of material between cells. In most finite difference and finite element formulations, all the solution variables except the velocities are cell-centered while the velocities are edge- or vertex-centered. As a result, the advection algorithm for the momentum is, by necessity, different than the algorithm used for the other variables. This paper reviews three momentum advection methods and proposes a new one. One method, pioneered in YAQUI, creates a new staggered mesh, while the other two, used in SALE and SHALE, are cell-centered. The new method is cell-centered and its relationship to the other methods is discussed. Both pure advection and strong shock calculations are presented to substantiate the mathematical analysis. From the standpoint of numerical accuracy, both the staggered mesh and the cell-centered algorithms can give good results, while the computational costs are highly dependent on the overall architecture of a code.

INTRODUCTION

During the addition of a simplified ALE [1] (arbitrary Eulerian–Lagrangian) capability to DYNA2D [2], the author discovered that there was no direct comparison of the different algorithms for advecting momentum in the literature. It is the goal of this paper to fill that void.

In the analysis presented in this paper, the hydrodynamic calculation is assumed to be split into separate Lagrangian and Eulerian steps. The Lagrangian step may be performed by any of a number of well-known finite difference stencils (e.g., [3–5]) or finite element formulations (e.g., [6–8]). The only assumption made here about the Lagrangian step is that the data are staggered: the velocities are centered at the nodes while the stress, density, internal energy, and all the history variables are centered in the cells.

The nomenclature of the finite difference community is adopted in this paper although the methods discussed are equally applicable to finite element formulations. A logically regular mesh is assumed. In one dimension, nodes are assigned integer numbers from left to right. The cell defined by nodes j and $j+1$ is labelled $j+\frac{1}{2}$. In two dimensions, the mesh defined by intersecting k and l lines; see Fig. 1. Nodes are referred to by the numbers of their intersecting k and l lines, e.g., (k, l), while cells are offset by $\pm\frac{1}{2}$, e.g., $(k+\frac{1}{2}, l+\frac{1}{2})$.

The advection algorithms for the momentum are constructed from the cell-centered algorithms used for the other variables. Two different advection algorithms are used in this paper, but the conclusions presented here are valid in general. The first one is the popular van Leer MUSCL [9] algorithm, which is used in a number of Eulerian and ALE codes such as CAVEAT [10], CSQ [11], DYNA2D [12], and PISCES [13]. It is monotonic, an important attribute for the advection of variables that have a limited range, and it is nonlinear, making a dispersion analysis impossible. The implicit SUPG (streamline, upwind Petrov–Galerkin) method [15], developed by Hughes and others, is a linear method for the pure advection problem and it can, therefore, be characterized by a classical dispersion analysis. Aside from its lack of monotonicity, it works as well as

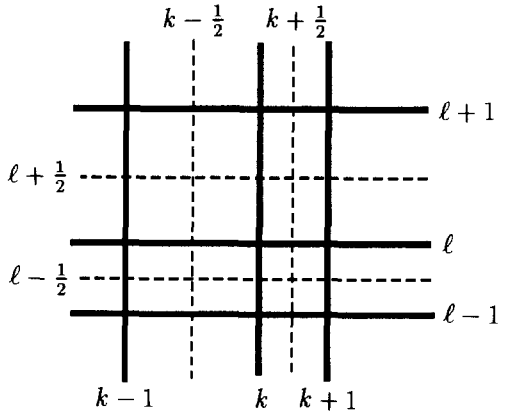


FIG. 1. The two-dimensional mesh is defined by the intersections of the k and l lines. Nodes are numbered by the integer pairs of their intersecting mesh lines, and the cell numbers are staggered with respect to the nodes by $\pm\frac{1}{2}$.

* This work was supported by Nippon Oil and Fats Co., Ltd.

0021-9991/92 $5.00

the van Leer algorithm. Both of these algorithms are summarized in the Appendix.

1. A SIMPLIFIED STAGGERED MESH ALGORITHM

Advection in One Dimension

All the shortcomings of the algorithms discussed in this paper can be demonstrated in one dimension. Clearly an algorithm that fails in one dimension will not work in two or three dimensions. The momentum advection algorithms will be presented in one and two dimensions, but their mathematical analysis will be largely restricted to one dimension. This section establishes the notation used in the one-dimensional analyses.

The one-dimensional scalar advection equation is a natural, nontrivial test case for an advection algorithm. The solution variable, $\phi(x, t)$, is transported with a constant velocity c relative to the spatial coordinate x. The exact solution for $\phi(x, t)$ is $\phi_0(x - ct)$.

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = 0, \qquad \phi(x, 0) = \phi_0(x). \tag{1.1}$$

At each node j, the volume transported at node j, f_j, is $c\,\Delta t$, where Δt is the timestep size. The volume of cell $j+\frac{1}{2}$ is denoted $V_{j+1/2}$, and the variables before advection are superscripted with a negative sign, after the advection, with a plus sign. Using these definitions, the advection of $\phi_{j+1/2}$ is described by Eq. (1.2),

$$\begin{aligned} \phi^+_{j+1/2} &= \frac{\phi^-_{j+1/2} V^-_{j+1/2} + \phi_j f_j - \phi_{j+1} f_{j+1}}{V^+_{j+1/2}} \\ V^+_{j+1/2} &= V^-_{j+1/2} + f_j - f_{j+1}. \end{aligned} \tag{1.2}$$

The advection algorithm described by Eq. (1.2) is conservative in ϕ; i.e., the integral of ϕ over the mesh is not changed by the advection.

The Staggered Mesh Algorithm in One Dimension

YAQUI [16] was the first code to construct a staggered mesh for the momentum advection, and the basic idea is still in common use. A mesh is constructed staggered with respect to the original mesh so that the original cell centroids become the new nodes; see Fig. 1. A cell-centered advection algorithm is then applied to new staggered mesh. The data necessary for the advection algorithm are the cell volumes, the values of ϕ in the cells, and the fluxes between the cells. All the data are readily defined except the fluxes. DeBar [18] invoked the following consistency condition to derive a set of staggered mass fluxes: If a body has a uniform velocity and a variable density before advection, then the body must have the same uniform velocity after advection. The equation corresponding to Eq. (1.2) on the staggered mesh is Eq. (1.3), where the transport mass, $\rho_{j+1/2} f_{j+1/2}$, is denoted $\tilde{f}_{j+1/2}$ to distinguish it from the transport volume,

$$v^+_j = \frac{M^-_j v^-_j + v_{j-1/2}\tilde{f}_{j-1/2} - v_{j+1/2}\tilde{f}_{j+1/2}}{M^+_j} \tag{1.3}$$

$$\begin{aligned} M^+_j &= M^-_j + \tilde{f}_{j-1/2} - \tilde{f}_{j+1/2} \\ \tilde{f}_{j+1/2} &= \tfrac{1}{2}(\rho_j f_j + \rho_{j+1} f_{j+1}) \\ &= \tfrac{1}{2}(\tilde{f}_j + \tilde{f}_{j+1}). \end{aligned} \tag{1.4}$$

Calculations performed by the author involving strong shocks (Section 6) have shown that while advecting momentum with an average of the transport volumes leads to extreme shock overheating, advecting the velocity with the average of the transport masses, Eq. (1.4), gives good results.

A dispersion analysis starts with Eq. (1.1) discretized on a uniform mesh and a constant value of c. From the construction of the staggered mesh, it is clear that this momentum advection algorithm inherits the dispersion properties of the underlying cell-centered advection algorithm.

Advection in Two Dimensions

A quadrilateral cell is surrounded by eight other cells, and it can exchange material with each of them. The transport volumes are labelled by the logical location of their centroid, e.g., the volume flux on the right edge of cell $(k+\frac{1}{2}, l+\frac{1}{2})$ is $f_{(k+1,l+1/2)}$; see Fig. 2. By convention, the transport volumes are considered to be positive if they transport material in either of the two positive logical directions.

There is a degree of latitude in how the staggered mesh is defined in two dimensions that is not present in one dimension. Only the simplest extension of the one-dimensional

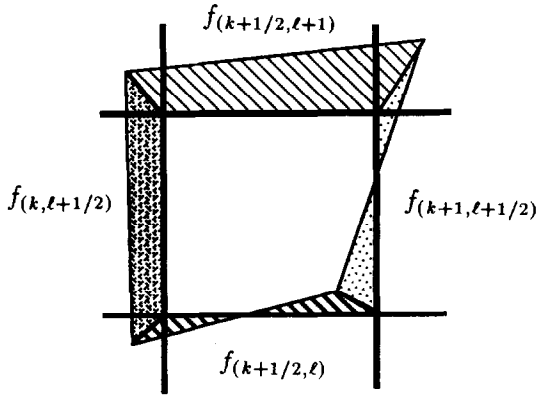


FIG. 2. Mass transport is assumed to occur only through the cell edges. The coupling between diagonal cells is ignored.

idea is presented. This extension differs from most staggered mesh implementations (e.g., Ref. [20]), including the original YAQUI algorithm, in that it does not explicitly define control volumes. A simple lumped mass matrix formulation is assumed: the mass of a cell is divided evenly among its nodes,

$$M_{(k,l)} = \tfrac{1}{4}(M_{(k-1/2,l-1/2)} + M_{(k+1/2,l-1/2)} + M_{(k+1/2,l+1/2)} + M_{(k-1/2,l+1/2)}). \quad (1.5)$$

The consistency condition is again used to define the staggered transport masses for the momentum advection. By assumption, the mass transport along the k lines is independent of the mass transport along the l lines and vice versa:

$$\begin{aligned}
\bar{f}_{(k-1/2,l)} &= \tfrac{1}{4}(\bar{f}_{(k-1,l-1/2)} + \bar{f}_{(k-1,l+1/2)} + \bar{f}_{(k,l-1/2)} + \bar{f}_{(k,l+1/2)}) \\
\bar{f}_{(k+1/2,l)} &= \tfrac{1}{4}(\bar{f}_{(k+1,l-1/2)} + \bar{f}_{(k+1,l+1/2)} + \bar{f}_{(k,l-1/2)} + \bar{f}_{(k,l+1/2)}) \\
\bar{f}_{(k,l-1/2)} &= \tfrac{1}{4}(\bar{f}_{(k-1/2,l-1)} + \bar{f}_{(k+1/2,l-1)} + \bar{f}_{(k-1/2,l)} + \bar{f}_{(k+1/2,l)}) \\
\bar{f}_{(k,l+1/2)} &= \tfrac{1}{4}(\bar{f}_{(k-1/2,l+1)} + \bar{f}_{(k-1/2,l+1)} + \bar{f}_{(k-1/2,l)} + \bar{f}_{(k+1/2,l)}).
\end{aligned} \quad (1.9)$$

Advantages and Disadvantages

The two primary advantages of the staggered mesh algorithm are: (1) it has the same dispersion and monotonicity characteristic as the cell-centered algorithm, and (2) each velocity component is advected as a single variable. As will be shown later, the cell-centered advection algorithms require several variables (two for one dimension, four for two dimensions, and eight for three dimensions) to be advected for each velocity component in order to avoid adding dispersion errors to the solution.

One criticism of the staggered mesh algorithm is that it smears strong shocks [17], but the author has not experienced this problem in his low Courant number ALE calculations or in the one-dimensional Eulerian test calculations in this paper, nor has Christensen [21] experienced it with his two-dimensional Eulerian calculations. The cell-centered algorithm in SALE was compared to different staggered mesh algorithms and it was found to be superior by its authors [1]. The programs mentioned here differ significantly, making it difficult to determine why some researchers are critical of the staggered mesh algorithm while others are not.

The argument that motivated the author to consider the cell-centered momentum advection algorithms, which are discussed in the following sections, is the complexity of implementing the staggered mesh algorithm on unstructured meshes. The number of edges for a staggered cell is determined by the number of nodes surrounding a particular node. Staggered cells with three or five edges are not uncommon in finite element calculations and vectorizing their advection adds complexity. An additional difficulty is many of the higher order accurate monotonic advection algorithms require quadrilateral cells in two dimensions (see, however, Zalesak [28]).

2. THE SALE MOMENTUM ADVECTION ALGORITHM

One-Dimensional Advection

The SALE algorithm was originally developed as a two-dimensional algorithm, but the one-dimensional simplification is described here because it gives a great deal of insight into the behavior of the two-dimensional algorithm. It is the only cell-centered momentum advection algorithm considered here that requires the advection of only one variable for each velocity component. While its efficiency is excellent, the use of a single variable introduces significant dispersion errors that can be corrected only by advecting additional variables.

The basic idea behind the SALE algorithm is that the change in the momentum of a node is one half of the sum of the changes in the momentum of its two adjacent cells. The specific cell momentum, $p_{j+1/2}$, the total cell momentum, $P_{j+1/2}$, and the nodal momentum, P_j, are defined by Eq. (2.1):

$$\begin{aligned}
p_{j+1/2} &= \tfrac{1}{2}\rho_{j+1/2}(v_j + v_{j+1}) \\
P_{j+1/2} &= \tfrac{1}{2}M_{j+1/2}(v_j + v_{j+1}) \\
P_j &= M_j v_j.
\end{aligned} \quad (2.1)$$

The new velocity is calculated by using the momentum fluxes. The change in the cell momentum from the advection is denoted $\Delta P_{j-1/2}$.

$$\begin{aligned}
\Delta P_{j-1/2} &= p_{j-1} f_{j-1} - p_j f_j \\
P_j^+ &= P_j^- + \tfrac{1}{2}(\Delta P_{j-1/2} + \Delta P_{j+1/2}) \\
v_j^+ &= P_j^+ / M_j^+.
\end{aligned} \quad (2.2)$$

This algorithm can be implemented using mass coordinates and mass fluxes by advecting the average cell velocity, $\bar{v}_{j+1/2}$, instead of the specific momentum. The only change in Eq. (22) is $p_j f_j$ is replaced by $\bar{v}_j \bar{f}_j$.

The consistency condition is then trivially satisfied because the change in the mass of a node is equal to one half of the sum of the changes in the mass of the adjacent cells,

$$v_j^+ = \frac{[M_j^- + (1/2)(\rho_{j-1} f_{j-1} - \rho_{j+1} f_{j+1})]\, v_j^-}{M_j^+} = v_j^-. \quad (2.3)$$

It is important to realize that this algorithm must be implemented in a form equivalent to Eq. (2.2) and that the final velocity should not simply be derived from the final momenta of the two cells. The diffusivity associated with using the latter approach can be appreciated by considering the result when the volume fluxes are zero and M^- equals M^+,

$$\begin{aligned} v_j^+ &= \frac{\frac{1}{2}\left[\begin{array}{l}(1/2)\, M^-_{j-1/2}(v^-_{j-1}+v^-_j) \\ \quad +(1/2)\, M^-_{j+1/2}(v^-_j+v^-_{j+1})\end{array}\right]}{M^+_j} \\ &= \frac{1}{2}v^-_j + \frac{M^+_{j-1/2}v^-_{j-1}+M^+_{j+1/2}v^-_{j+1}}{4M^+_j}. \end{aligned} \tag{2.4}$$

Equation (2.4) conserves the momentum, but it smears out any velocity discontinuity in only a few timesteps. This error can be regarded as an inversion error because the cell momentum, a function of the nodal velocities, is not inverted exactly to recover the nodal velocities. An exact inversion, which is performed later, requires the solution of a set of simultaneous linear equations and the specification of extra boundary conditions.

A von Neumann Analysis

A dispersion analysis demonstrates that the SALE algorithm in Eq. (2.2) has dispersion errors that are independent of the underlying cell-centered advection method. For an excellent review of the von Neumann analysis method and its application to dispersion problems, refer to Trefethen [22].

An explicit advection method that can be written in the form of Eq. (2.5) will give a complex dispersion equation, where $\mathscr{P}$ is a complex polynomial, the value of ϕ at node j for timestep n is ϕ_j^n, the constant mesh spacing is J, and the timestep is h:

$$\phi_j^{n+1} = \phi_j^n + \mathscr{F}(c, h, J, ..., \phi^n_{j-1}, \phi^n_j, \phi^n_{j+1}, ...) \tag{2.5}$$

$$e^{i\omega h} = 1 + \mathscr{P}(e^{i\xi J})$$

$$\mathscr{P}(e^{i\xi J}) = \sum_j \beta_j e^{i\xi jJ} \qquad \text{for the appropriate range of } j. \tag{2.6}$$

The dispersion equation has the general form given in Eq. (2.7), where $\mathscr{P}_\mathrm{r}$ and $\mathscr{P}_\mathrm{i}$ denote the real and imaginary parts of $\mathscr{P}$, respectively:

$$\omega h = \tan^{-1}\left(\frac{\mathscr{P}_\mathrm{i}}{1+\mathscr{P}_\mathrm{r}}\right). \tag{2.7}$$

Recognizing that the relations in the above equations are periodic in ωh and ξJ, the normalized frequency and wave number are defined to simplify the notation.

$$\bar{\omega} = \omega h, \qquad \bar{\xi} = \xi J. \tag{2.8}$$

The normalized phase velocity, the normalized group velocity, and the amplification factor, G, are calculated:

$$\bar{c}_p = \bar{\omega}/\bar{\xi} \tag{2.9}$$

$$\bar{c}_g = \frac{\cos^2(\bar{\omega})}{(1+\mathscr{P}_\mathrm{r})^2}\left[(1+\mathscr{P}_\mathrm{r})\frac{d\mathscr{P}_\mathrm{i}}{d\bar{\xi}} - \mathscr{P}_\mathrm{i}\frac{d\mathscr{P}_\mathrm{r}}{d\bar{\xi}}\right]. \tag{2.10}$$

$$G = \sqrt{\mathscr{P}_\mathrm{i}^2 + (1+\mathscr{P}_\mathrm{r})^2} \tag{2.11}$$

The von Neumann analysis of the SALE algorithm proceeds by first calculating the increment in the cell momentum:

$$\begin{aligned} P^n_{j+1/2} &= \tfrac{1}{2}(v^n_j + v^n_{j+1}) = \tfrac{1}{2}(1+e^{-i\xi})\, v^n_j \\ \Delta P^{n+1}_{j+1/2} &= P^{n+1}_{j+1/2} - P^n_{j+1/2} = \tfrac{1}{2}(1+e^{-i\xi})\, \mathscr{P} v^n_j. \end{aligned} \tag{2.12}$$

The velocity is updated from the changes in the cell momentum:

$$\begin{aligned} v^{n+1}_j &= v^n_j + \tfrac{1}{2}(\Delta P_{j+1/2} + \Delta P_{j-1/2}) \\ e^{i\bar{\omega}} v^n_j &= \tfrac{1}{4}(1+e^{i\xi})(1+e^{-i\xi})\, \mathscr{P} v^n_j \\ &= \tfrac{1}{2}(1+\cos(\bar{\xi}))\, \mathscr{P} v^n_j. \end{aligned} \tag{2.13}$$

The dispersion relation for the SALE advection algorithm is given by Eq. (2.14),

$$\bar{\omega} = \tan^{-1}\left(\frac{(1/2)(1+\cos(\bar{\xi}))\, \mathscr{P}_\mathrm{i}}{1+(1/2)(1+\cos(\bar{\xi}))\, \mathscr{P}_\mathrm{r}}\right). \tag{2.14}$$

By comparing Eq. (2.14) to Eq. (2.7), we see that the cell-centered advection algorithm has the effect of introducing an extra dispersion factor, $\mathscr{C}$, of $\frac{1}{2}(1+\cos(\bar{\xi}))$ in front of the spatial part of the difference stencil, $\mathscr{P}$. For small values of $\bar{\xi}$, $\frac{1}{2}(1+\cos(\bar{\xi}))$ is approximately one, and the behavior of the node-centered and cell-centered algorithms is almost identical.

When $\bar{\xi}$ is π, the dispersion problems with the SALE algorithm appear. The coefficient $\frac{1}{2}(1+\cos(\bar{\xi}))$ is zero, making $\bar{\omega}$ zero, and therefore, by Eq. (2.9), the phase velocity must be zero independent of the choice of $\mathscr{P}$. By defining $\bar{\mathscr{P}}$ as $\frac{1}{2}(1+\cos(\bar{\xi}))\mathscr{P}$, and using Eq. (2.7), the group velocity must also be zero independent of the choice of the underlying advection algorithm:

$$\begin{aligned} \frac{d\bar{\mathscr{P}}_\mathrm{i}}{d\bar{\xi}} &= \frac{1}{2}(1+\cos(\bar{\xi}))\frac{d\mathscr{P}_\mathrm{i}}{d\bar{\xi}} - \frac{1}{2}\sin(\bar{\xi})\, \mathscr{P}_\mathrm{i} = 0 \\ \frac{d\bar{\mathscr{P}}_\mathrm{r}}{d\bar{\xi}} &= \frac{1}{2}(1+\cos(\bar{\xi}))\frac{d\mathscr{P}_\mathrm{r}}{d\bar{\xi}} - \frac{1}{2}\sin(\bar{\xi})\, \mathscr{P}_\mathrm{r} = 0 \\ \bar{c}_g &= \frac{\cos^2(\bar{\omega})}{(1+\bar{\mathscr{P}}_\mathrm{r})^2}\left[(1+\bar{\mathscr{P}}_\mathrm{r})\frac{d\bar{\mathscr{P}}_\mathrm{i}}{d\bar{\xi}} - \bar{\mathscr{P}}_\mathrm{i}\frac{d\bar{\mathscr{P}}_\mathrm{r}}{d\bar{\xi}}\right] = 0. \end{aligned} \tag{2.14}$$

Not only is the wave associated with $\bar{\xi}$ equal to π not advected, it is also not damped out,

$$G=\sqrt{(1+(1/2)(1+\cos(\bar{\xi}))\,\mathscr{P}_r)^2+(1/4)(1+\cos(\bar{\xi}))^2\,\mathscr{P}_i^2}=1. \tag{2.15}$$

The magnitude of these dispersion errors is demonstrated later with example calculations. As a consequence of the dispersion errors, the algorithm does not inherit the monotonicity of the underlying cell-centered advection algorithm.

Two-Dimensional Advection

The two-dimensional algorithm is a straightforward extension of the one-dimensional algorithm; the change in the nodal momentum is the average of the changes in the momentum of the cells surrounding it. The x and y components of the velocity are advected in an identical manner and the symbol "v" is used to represent both components without distinction:

$$\begin{aligned}p_{(k+1/2,l+1/2)}&=\tfrac{1}{4}\rho_{(k+1/2,l+1/2)}(v_{(k,l)}+v_{(k+1,l)}\\&\quad+v_{(k+1,l+1)}+v_{(k,l+1)})\\P_{(k+1/2,l+1/2)}&=\tfrac{1}{4}M_{(k+1/2,l+1/2)}(v_{(k,l)}+v_{(k+1,l)}\\&\quad+v_{(k+1,l+1)}+v_{(k,l+1)}).\end{aligned} \tag{2.16}$$

The change in the cell momentum due to the advection is denoted $\Delta P_{(k+1/2,l+1/2)}$,

$$\begin{aligned}v^+_{(k,l)}&=\frac{M^-_{(k,l)}}{M^+_{(k,l)}}v^-_{(k,l)}+\frac{1}{4M^+_{(k,l)}}\\&\quad\times[\Delta P_{(k-1/2,l-1/2)}+\Delta P_{(k+1/2,l-1/2)}\\&\quad+\Delta P_{(k+1/2,l+1/2)}+\Delta P_{(k-1/2,l+1/2)}].\end{aligned} \tag{2.17}$$

A von Neumann analysis of Eq. (2.17) gives a result that is similar in form to the one-dimensional results presented above. The mesh spacings in the k and l directions are K and L, $\bar{\xi}$ is ξK, and $\bar{\eta}$ is ηL:

$$\begin{aligned}\bar{\omega}&=\tan^{-1}\left(\frac{\mathscr{C}\mathscr{P}_i}{1+\mathscr{C}\mathscr{P}_r}\right)\\\mathscr{C}(\bar{\xi},\bar{\eta})&=\tfrac{1}{4}[1+\cos(\bar{\xi})+\cos(\bar{\eta})+\cos(\bar{\xi})\cos(\bar{\eta})].\end{aligned} \tag{2.19}$$

The dispersion characteristics of the cell-centered momentum advection algorithm differ from the underlying cell-centered advection algorithm because of $\mathscr{C}$. The one-dimensional results of the previous section are recovered in the k or l directions by setting $\bar{\eta}$ or $\bar{\xi}$ to zero, and the group and phase velocities can be calculated in other directions if so desired [22].

3. THE SHALE MOMENTUM ADVECTION ALGORITHM

One-Dimensional Advection

Margolin and Beason [17] found that the SALE algorithm created spurious oscillations in SHALE [14] that dominated the solution near regions of steep gradients. They indentified the weak point of the SALE algorithm as being the redistribution of the cell-centered momentum to the nodes and sought to improve it. The method they created and analyzed advects two variables in one dimension and three variables in two dimensions.

In the development of their method, Margolin and Beason use a translation operator and they compare the difference between the Taylor expansions of the continuum translation operator and the discrete representation of the translation operator generated by the advection stencil. Only their algorithm is presented; the interested reader is referred to their paper [17] for its derivation.

Instead of considering the momentum distribution to be a piecewise constant distribution on a staggered mesh, it is approximated as a linear function on each cell. In addition to the cell momentum, $P_{j+1/2}$, the momentum gradient, $S_{j+1/2}$, appears in their Taylor expansion. Note that the cell-centered definition of the momentum here differs from the definition used in the previous section,

$$\begin{aligned}P_j&=P_{j+1/2}+S_{j+1/2}(x_j-x_{j+1/2})=M_jv_j\\P_{j+1}&=P_{j+1/2}+S_{j+1/2}(x_{j+1}-x_{j+1/2})=M_{j+1}v_{j+1}\\P_{j+1/2}&=\tfrac{1}{2}(P_{j+1}+P_j)=\tfrac{1}{2}(M_jv_j+M_{j+1}v_{j+1})\\S_{j+1/2}&=\frac{P_{j+1}-P_j}{x_{j+1}-x_j}.\end{aligned} \tag{3.1}$$

The momentum at node j could be updated by advecting P and S and then using Eq. (3.1), but the result would not be symmetric. A symmetric form for updating P_j is obtained by using the Taylor expansions in cells $j+\frac{1}{2}$ and $j-\frac{1}{2}$ and averaging the result. Instead of updating the cell momentum and the momentum gradient, the momentum is updated directly from the fluxes. The coordinates used in Eq. (3.2) are the rezoned spatial coordinates, not the original Lagrangian coordinates:

$$\begin{aligned}P^+_j&=P^-_j+\tfrac{1}{2}[\Delta P_{j-1/2}+\Delta S_{j-1/2}(x_j-x_{j-1/2})\\&\quad+\Delta P_{j+1/2}+\Delta S_{j+1/2}(x_j-x_{j+1/2})]\\v^+_j&=P^+_j/M^+_j.\end{aligned} \tag{3.2}$$

Proceeding as before, P and S are expressed in terms of v:

$$\begin{aligned}v^+_j&=v^-_j+\tfrac{1}{2}[\Delta P_{j-1/2}+\Delta P_{j+1/2}\\&\quad+\tfrac{1}{2}\Delta S_{j-1/2}-\tfrac{1}{2}\Delta S_{j+1/2}]\\S_{j+1/2}&=v^-_{j+1}-v^-_j\\P_{j+1/2}&=\tfrac{1}{2}(v^-_{j+1}+v^-_j).\end{aligned} \tag{3.3}$$

The von Neumann analysis shows that the new algorithm, Eq. (3.3), has the same dispersion properties as the underlying advection algorithm, Eq. (2.12):

$$\begin{aligned} S_{j+1/2} &= (e^{i\xi} - 1)\, v_j \\ P_{j+1/2} &= \tfrac{1}{2}(e^{i\xi} + 1)\, v_j \end{aligned} \tag{3.4}$$

$$\begin{aligned} e^{i\omega} v_j &= [1 + \tfrac{1}{4}((1 + e^{i\xi})(1 + e^{-i\xi}) \\ &\quad + (e^{i\xi} + 1)(e^{i\xi} - 1))]\, \mathscr{P} v_j \\ &= [1 + \mathscr{P}]\, v_j. \end{aligned} \tag{3.5}$$

The von Neumann method is only applicable to linear advection methods and most of the popular methods are nonlinear and monotonic. Unfortunately, the SHALE method does not preserve the monotonicity characteristics of the underlying advection algorithm. Margolin and Beason [17] advect S with the donor cell algorithm to reduce the spurious oscillations of their method, but this does not completely solve the problem. Recent work on this algorithm by Margolin shows that special flux limiters must be constructed using the nodal velocities in order to guarantee monotonicity [23]. It is not clear how these limiters could be translated into the slope-limiting format developed by van Leer.

Advection in Two Dimensions

The extension of the method into two dimensions is accomplished by keeping the constant and linear terms in a Taylor series expansion in two dimensions. The x and y superscripts denote the direction of the momentum gradient, S:

$$\begin{aligned} P(x, y) &= P_{(k+1/2,\, l+1/2)} \\ &\quad + S^x_{(k+1/2,\, l+1/2)}(x - x_{(k+1/2,\, l+1/2)}) \\ &\quad + S^y_{(k+1/2,\, l+1/2)}(y - y_{(k+1/2,\, l+1/2)}) \\ P_{(k+1/2,\, l+1/2)} &= \tfrac{1}{4}(P_{(k,l)} + P_{(k+1,l)} \\ &\quad + P_{(k+1,l+1)} + P_{(k,l+1)}) \\ S^x_{(k+1/2,\, l+1/2)} &= \left.\frac{\partial P}{\partial x}\right|_{(k+1/2,\, l+1/2)} \\ S^y_{(k+1/2,\, l+1/2)} &= \left.\frac{\partial P}{\partial y}\right|_{(k+1/2,\, l+1/2)}. \end{aligned} \tag{3.6}$$

The increment in the momentum at node (k, l) is the average of the momentum increments for node (k, l) calculated from its four surrounding cells. As in the one-dimensional case, the momentum is updated by using the fluxes directly, and ΔP and ΔS are notational conveniences. For compactness, the rezoned displacement vectors $x_{(k,l)} - x_{(k\pm1/2,\, l\pm1/2)}$ and $y_{(k,l)} - y_{(k\pm1/2,\, l\pm1/2)}$ are denoted $\Delta x_{(\pm1/2,\, \pm1/2)}$ and $\Delta y_{(\pm1/2,\, \pm1/2)}$,

$$\begin{aligned} P^+_{(k,l)} &= P^-_{(k,l)} + \frac{1}{4}\sum_{K=-1/2}^{1/2}\sum_{L=-1/2}^{1/2} [\Delta P_{(k+K,\, l+L)} \\ &\quad + \Delta S^x_{(k+K,\, l+L)}\, \Delta x_{(K,L)} \\ &\quad + \Delta S^y_{(k+K,\, l+L)}\, \Delta y_{(K,L)}]. \end{aligned} \tag{3.7}$$

The reason the momentum update must be performed with the fluxes instead of the updated values of P and S is that the velocity field is bilinear (i.e., it contains a term involving xy) and the momentum calculated at the nodes by using Eq. (3.6) does not agree exactly with the nodal momentum. This situation is similar to the one encountered with the SALE algorithm. The diffusivity that is introduced by using the updated values can be assessed by assuming that the fluxes are zero, the mesh is uniform, and the density is constant. The final velocity can be written entirely in terms of the initial velocities just as in Eq. (2.4):

$$\begin{aligned} v^+_{(k,l)} &= \tfrac{1}{16}[12 v^-_{(k,l)} + 2(v^-_{(k,l-1)} + v^-_{(k+1,l)} \\ &\quad + v^-_{(k,l+1)} + v^-_{(k-1,l)}) - (v^-_{(k-1,l-1)} + v^-_{(k+1,l-1)} \\ &\quad + v^-_{(k+1,l+1)} + v^-_{(k-1,l+1)})]. \end{aligned} \tag{3.8}$$

If the problem is one-dimensional and it is aligned with the mesh direction, such that either $v_{(k\pm n,\, l)}$ or $v_{(k,\, l\pm n)}$ equals $v_{(k,l)}$ for all values of n, then this error is eliminated.

The two-dimensional algorithm does not introduce any extra dispersion errors in the mesh direction, but it does affect the underlying advection algorithm along the mesh diagonals. The extra dispersion coefficient, $\mathscr{C}$, for this algorithm is given by Eq. (3.9). When $\bar{\xi}$ or $\bar{\eta}$ is zero, $\mathscr{C}$ equals 1.0 and no extra dispersion is introduced into the problem:

$$\mathscr{C}(\bar{\xi}, \bar{\eta}) = \tfrac{1}{4}[3 + \cos(\bar{\xi}) + \cos(\bar{\eta}) - \cos(\bar{\xi})\cos(\bar{\eta})]. \tag{3.9}$$

Eliminating the Dispersion Error

The author and Margolin [23] independently discovered that the dispersion error defined by Eq. (3.9) can be eliminated by including the bilinear term in the Taylor expansion:

$$\begin{aligned} P(x, y) &= P_{(k+1/2,\, l+1/2)} \\ &\quad + S^x_{(k+1/2,\, l+1/2)}(x - x_{(k+1/2,\, l+1/2)}) \\ &\quad + S^y_{(k+1/2,\, l+1/2)}(y - y_{(k+1/2,\, l+1/2)}) \\ &\quad + S^{xy}_{(k+1/2,\, l+1/2)}(x - x_{(k+1/2,\, l+1/2)}) \\ &\quad \times (y - y_{(k+1/2,\, l+1/2)}) \\ S^{xy}_{(k+1/2,\, l+1/2)} &= \left.\frac{\partial^2 P}{\partial x\, \partial y}\right|_{(k+1/2,\, l+1/2)}. \end{aligned} \tag{3.10}$$

The bilinear term is just the hourglass mode momentum. In most hydrodynamics codes, the hourglass velocity, s^{xy}, is approximated by Eq. (3.11), but it is orthogonal to the straining and rigid body modes only for rectangular cells. The cell area is A:

$$s^{xy}_{(k+1/2,\,l+1/2)} = \frac{1}{A}\left(-v_{(k,l)} + v_{(k+1,l)} - v_{(k+1,l+1)} + v_{(k,l+1)}\right). \tag{3.11}$$

Flanagan and Belytschko [24] defined the hourglass velocity precisely as the bilinear part of the velocity field and derived an explicit expression for $s^{xy}_{(k+1/2,\,l+1/2)}$ based on that definition. The same mode shape is derived by Margolin and Pyun [25], but they use it to filter the velocities. Kosloff and Frazier [26] also used the same definition in an earlier paper, but their approach required the solution of a system of linear equations. Note that all the definitions agree when the mesh is rectangular.

The bilinear term must be advected, so that four variables are advected for each velocity component instead of original three proposed by Margolin and Beason, and Eq. (3.7) is modified to include the bilinear contributions:

$$\begin{aligned} P^{+}_{(k,l)} = P^{-}_{(k,l)} + \frac{1}{4}\sum_{K=-1/2}^{1/2}\sum_{L=-1/2}^{1/2}\big[&\Delta P_{(k+K,\,l+L)} \\ &+ \Delta S^{x}_{(k+K,\,l+L)}\,\Delta x_{(K,L)} + \Delta S^{y}_{(k+K,\,l+L)}\,\Delta y_{(K,L)} \\ &+ \Delta S^{xy}_{(k+K,\,l+L)}\,\Delta x_{(K,L)}\,\Delta y_{(K,L)}\big]. \end{aligned} \tag{3.12}$$

Applying the analysis methods used on Eq. (3.7), it can be shown with some algebra that the inclusion of the bilinear term eliminates the zero flux error in Eq. (3.8) and the dispersion error in Eq. (3.9).

4. A GENERAL CLASS OF CELL-CENTERED ALGORITHMS

The Inversion Error and Dispersion

The general scheme of the previous two algorithms is to define a set of cell-centered variables in terms of the nodal velocities and masses, advect them, and then update the nodal velocities from the increments in the cell-centered variables. A class of cell-centered algorithms can be written in the form of Eq. (4.1), where $\Psi_{(\alpha,\,j+1/2)}$ are the new cell-centered variables and α ranges over the number of variables. A and B are linear transformations that can be functions of any of the cell or nodal variables. Two indices, J and N, take on both positive and negative values and their range is dependent on the advection stencil. All the summations are indicated explicitly in this section:

$$\begin{aligned} \Psi^{-}_{(\alpha,\,j+J/2)} &= \sum_{N} A_{(\alpha,\,j+J/2+N/2)}\, v^{-}_{j+J/2+N/2} \\ \Delta\Psi_{(\alpha,\,j+J/2)} &= \Psi^{+}_{(\alpha,\,j+J/2)} - \Psi^{-}_{(\alpha,\,j+J/2)} \\ &= \mathscr{F}\left(\Psi^{-}_{(\alpha,\,j+J/2)}\right) \\ \Delta v_j &= v^{+}_j - v^{-}_j = \sum_{\alpha,J} B_{(\alpha,\,j+J/2)}\,\Delta\Psi_{(\alpha,\,j+J/2)}. \end{aligned} \tag{4.1}$$

The single argument in the advection stencil, $\mathscr{F}$, denotes the variable that is being advected and which cell is being advected. All the other arguments are suppressed for notational simplicity. For a dispersion analysis, the advection algorithm is assumed to be linear:

$$\begin{aligned} v^{+}_j &= v^{-}_j + \sum_{\alpha,J} B_{(\alpha,\,j+J/2)}\,\mathscr{F}\left(\Psi_{(\alpha,\,j+J/2)}\right) \\ &= v^{-}_j + \sum_{\alpha,J,N} B_{(\alpha,\,j+J/2)} A_{(\alpha,\,j+J/2+N/2)}\,\mathscr{F}\left(v^{-}_{j+J/2+N/2}\right) \\ \mathscr{C} &= \sum_{\alpha,J,N} B_{\alpha,\,j+J/2} A_{(\alpha,\,j+J/2+N/2)} e^{-i\xi(J/2+N/2)} \\ e^{i\bar{\omega}} &= 1 + \mathscr{C}\mathscr{P}. \end{aligned} \tag{4.2}$$

The underlying cell-centered advection algorithm's dispersion characteristics are altered unless $\mathscr{C}$ is 1.0.

The inversion error is described by the relationship between the final nodal velocity and the initial velocities when the fluxes are zero and the nodal velocity is updated from the final values of Ψ (which remain unchanged in this case):

$$v^{+}_j = \sum_{\alpha,J,N} B_{(\alpha,\,j+J/2)} A_{(\alpha,\,j+J/2+N/2)}\, v^{-}_{j+J/2+N/2}. \tag{4.3}$$

By comparing Eq. (4.2) and Eq. (4.3), and method that has an inversion error must also modify the dispersion characteristics of the underlying cell-centered algorithm. To avoid dispersion problems, B must be an inverse of A. The same arguments can be made in two or three dimensions by replacing each scalar index j, by a vector index, $\mathbf{j}$, having two (j_1, j_2) or three (j_1, j_2, j_3) components.

The New Algorithm

The simplest form for A that results in a symmetric algorithm is the identity:

$$\begin{Bmatrix} \Psi_{(1,\,j+1/2)} \\ \Psi_{(2,\,j+1/2)} \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} v_j \\ v_{j+1} \end{Bmatrix}. \tag{4.4}$$

To conserve momentum, Ψ is advected with the mass fluxes:

$$\Psi^{+}_{(m,\,j+1/2)} = \frac{M^{-}_{j+1/2}\,\Psi^{-}_{(m,\,j+1/2)} + \Psi_{(m,\,j-1)} f_{j-1} - \Psi_{(m,\,j+1)} f_{j+1}}{M^{+}_{j+1/2}}. \tag{4.5}$$

The inverse of Eq. (4.4) can be parameterized by θ:

$$v_j = [\theta \;\; 1-\theta] \begin{Bmatrix} \Psi_{(1,j+1/2)} \\ \Psi_{(2,j-1/2)} \end{Bmatrix}. \tag{4.6}$$

Mass weighting is chosen for the inverse here:

$$v_j = \frac{1}{2M_j} [M_{j+1/2} \;\; M_{j-1/2}] \begin{Bmatrix} \Psi_{(1,j+1/2)} \\ \Psi_{(2,j-1/2)} \end{Bmatrix}. \tag{4.7}$$

The two-dimensional extension follows where $[\,]'$ denotes the transpose:

$$\begin{Bmatrix} \Psi_{(1,k+1/2,l+1/2)} \\ \Psi_{(2,k+1/2,l+1/2)} \\ \Psi_{(3,k+1/2,l+1/2)} \\ \Psi_{(4,k+1/2,l+1/2)} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} v_{(k,l)} \\ v_{(k+1,l)} \\ v_{(k+1,l+1)} \\ v_{(k,l+1)} \end{Bmatrix} \tag{4.8}$$

$$v_{(kl)} = \frac{1}{4M_{(k,l)}} \begin{bmatrix} M_{(k+1/2,l+1/2)} \\ M_{(k+1/2,l-1/2)} \\ M_{(k-1/2,l-1/2)} \\ M_{(k+1/2,l-1/2)} \end{bmatrix}' \begin{Bmatrix} \Psi_{(1,k+1/2,l+1/2)} \\ \Psi_{(2,k+1/2,l-1/2)} \\ \Psi_{(3,k-1/2,l-1/2)} \\ \Psi_{(4,k+1/2,l-1/2)} \end{Bmatrix}. \tag{4.9}$$

This algorithm is called the HIS (half index shift) algorithm in the remainder of this paper.

Monotonicity

A function is monotonic over an interval if its derivative does not change sign. Monotonic advection algorithms guarantee that when a monotonic distribution is numerically advected the resulting distribution is also monotonic. This means that no new maximums or minimums are created. Additionally, monotonic algorithms are also TVNI (total variation not increasing) [27].

The HIS algorithm unconditionally inherits the monotonicity properties of the underlying algorithm. The velocity of node j is shifted left to cell $j-\frac{1}{2}$, advected, then shifted right to cell $j-\frac{1}{2}$ and advected. The mass-weighted average of the two shifted velocities defines the new velocity at node j. The sign of the derivative of v is determined by the differences between the values of v at succeeding nodes and it is independent of the mesh spacing. In particular, on a general mesh with arbitrary zoning, shifting the velocity profile to the left or right by a constant logical value does not change the monotonicity of the velocity profile.

The issues associated with monotonicity in two or three dimensions are much more complex than they are in one dimension. A great deal of research has focused on multidimensional advection algorithms in recent years [28–31]. Momentum advection algorithms that are not monotonic in one dimension, however, will not suddenly become monotonic in two dimensions. The HIS algorithm is monotonic in multidimensions for the same reason that it is monotonic in one dimension: a logical shift of the velocity field does not alter its monotonicity and the final velocity i the mass-weighted average of the shifted velocities.

For the scalar advection equation with uniform zoning the values of v from the HIS algorithm will agree to within the roundoff error with the values calculated using the staggered mesh algorithm regardless of the underlying advection algorithm. The reason for this result is that on a uniform mesh both the original mesh and the staggered mesh have the same spacing.

5. TAYLOR EXPANSIONS OF THE ADVECTION ALGORITHMS

The purpose of this section is to demonstrate that the HIS algorithm, a modification of the SHALE algorithm, and the staggered mesh algorithm are first-order approximations of each other. When the underlying cell-centered advection algorithm is linear, a modified version of the SHALE algorithm, the HIS, and the staggered mesh algorithms give identical answers. The SALE algorithm also approximates these algorithms, but it is shown to have a first-order truncation error.

A one-dimensional analysis is presented first. The flux of Ψ through node j is denoted $F(\boldsymbol{\Psi}_j, \mathbf{x}_j, f_j)$, where the vector arguments denote arrays of values centered about node j. The length of the arrays is dependent on the choice of the advection algorithm. Although x is used as a coordinate here, the analysis is equally valid for volume and mass coordinates,

$$\begin{aligned} \boldsymbol{\Psi}_j &= \{\Psi_{j-m+1/2}, \ldots, \Psi_{j-1/2}, \Psi_{j+1/2}, \ldots, \Psi_{j+m-1/2}\} \\ \mathbf{x}_j &= \{x_{j-m}, \ldots, x_j, \ldots, x_{j+m}\}. \end{aligned} \tag{5.1}$$

Using the flux notation, the staggered mesh momentum advection algorithm is described by Eq. (5.2),

$$\begin{aligned} v_j^+ = \frac{M_j^-}{M_j^+} v_j^- + \frac{1}{M_j^+} [&F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1/2}, \bar{f}_{j-1/2}) \\ &- F(\mathbf{v}_{j+1/2}, \mathbf{x}_{j+1/2}, \bar{f}_{j+1/2})], \end{aligned} \tag{5.2}$$

where

$$\begin{aligned} \mathbf{x}_{j-1/2} &= \tfrac{1}{2}(\mathbf{x}_{j-1} + \mathbf{x}_j), \qquad \mathbf{x}_{j+1/2} = \tfrac{1}{2}(\mathbf{x}_j + \mathbf{x}_{j+1}) \\ \bar{f}_{j-1/2} &= \tfrac{1}{2}(\bar{f}_{j-1} + \bar{f}_j), \qquad \bar{\mathbf{f}}_{j+1/2} = \tfrac{1}{2}(\bar{f}_j + \bar{f}_{j+1}). \end{aligned} \tag{5.3}$$

The HIS algorithm, expressed in the same notation, is given by Eq. (5.4):

$$\begin{aligned} v_j^+ = \frac{1}{2M_j^+} [&M_{j-1/2}^- v_j^- + F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1}, \bar{f}_{j-1}) \\ &- F(\mathbf{v}_{j+1/2}, \mathbf{x}_j, \bar{f}_j) + M_{j+1/2}^- v_j^- \\ &+ F(\mathbf{v}_{j-1/2}, \mathbf{x}_j, \bar{f}_j) - F(\mathbf{v}_{j+1/2}, \mathbf{x}_{j+1}, \bar{f}_{j+1})]. \end{aligned} \tag{5.4}$$

The terms in Eq. (5.4) can be regrouped and written in terms of Taylor expansions about the fluxes centered at $j-\frac{1}{2}$ and $j+\frac{1}{2}$,

$$v_j^+ = \frac{M_j^-}{M_j^+} v_j^- + \frac{1}{M_j^+}\left[\tfrac{1}{2}(F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) + F(\mathbf{v}_{j-1/2}, \mathbf{x}_j, \tilde{f}_j)) - \tfrac{1}{2}(F(\mathbf{v}_{j+1/2}, \mathbf{x}_j, \tilde{f}_j) + F(\mathbf{v}_{j+1/2}, \mathbf{x}_{j+1}, \tilde{f}_{j+1}))\right]$$

$$= \frac{M_j^-}{M_j^+} v_j^- + \frac{1}{M_j^+}\left[F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1/2}, \tilde{f}_{j-1/2}) - F(\mathbf{v}_{j+1/2}, \mathbf{x}_{j+1/2}, \tilde{f}_{j+1/2}) + \mathcal{O}(\Delta\tilde{f}^2) + \mathcal{O}(|\Delta\mathbf{x}|^2)\right]. \tag{5.5}$$

Extending this analysis to two dimensions is straightforward although the expressions become quite lengthy. The staggered mass fluxes are defined in terms of the cell edge fluxes, Eq. (1.9), and the staggered geometry is also a linear combination of the cell geometry. The Taylor expansions for the fluxes will have the same basic relationships as those in Eq. (1.9) but with the appropriate truncation terms added to the right-hand side. For example, the Taylor expansion for the flux of v through the staggered edge $(k-\frac{1}{2}, l)$ is the average of the fluxes of v through cell edges $(k-1, l-\frac{1}{2})$, $(k-1, l+\frac{1}{2})$, $(k, l-\frac{1}{2})$, and $(k, l+\frac{1}{2})$ with second-order truncation errors in $\Delta\mathbf{x}$ and $\tilde{f}$.

It does not appear possible to generate a higher order approximation of the staggered mesh algorithm and to retain monotonicity by extending the HIS algorithm. To demonstrate the difficulty, it is sufficient to consider only the argument x in the fluxes. All the other arguments in the flux function F are suppressed. The flux at $x_{j+1/2}$ is to be approximated as a linear combination of fluxes evaluated at the nodes:

$$F(x_{j+1/2}) = \sum_k \alpha_k F(x_k). \tag{5.6}$$

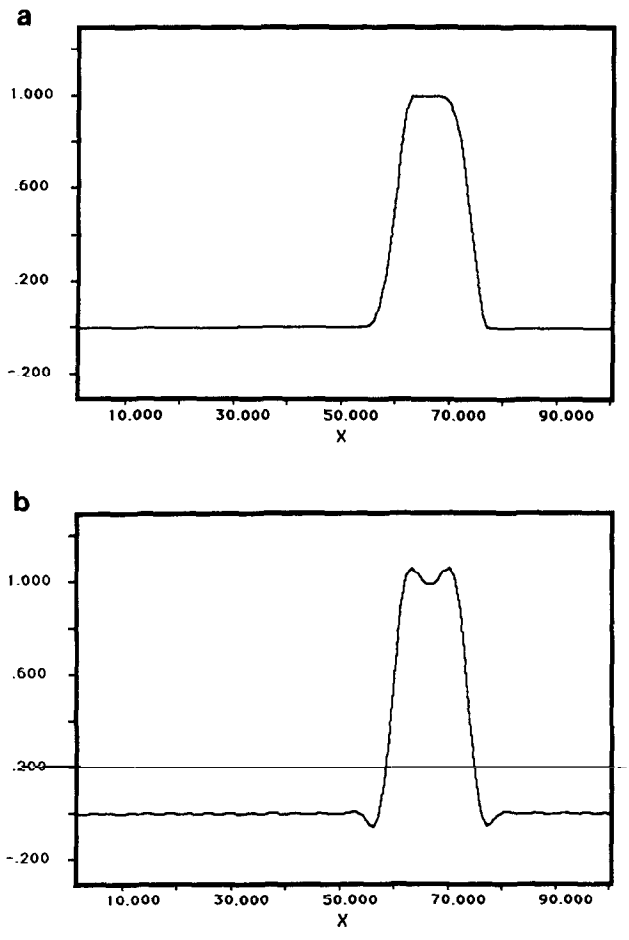


FIG. 3. The linear, scalar advection equation is solved with (a) the van Leer algorithm and (b) the SUPG algorithm for c equal to 0.1.

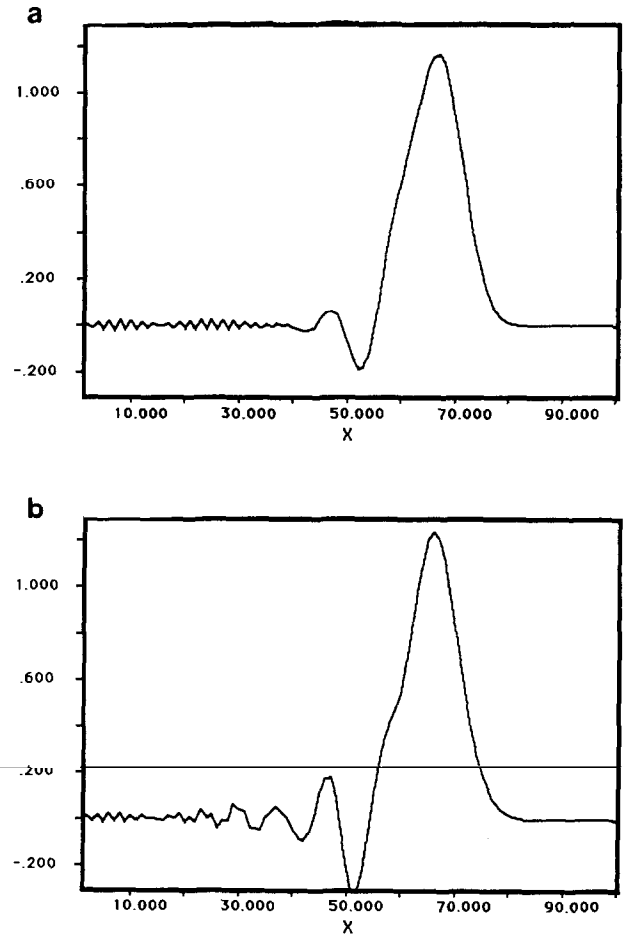


FIG. 4. Both (a) the van Leer and (b) the SUPG algorithms exhibit severe dispersion errors with the SALE algorithm.

A Taylor series expansion generates three equations that the coefficients α_k must satisfy in order to have a truncation error of $\mathcal{O}(|x_k - x_{j+1/2}|^3)$:

$$\begin{aligned} \sum_k \alpha_k &= 1 \\ \sum_k \alpha_k (x_k - x_{j+1/2}) &= 0 \\ \sum_k \alpha_k (x_k - x_{j+1/2})^2 &= 0. \end{aligned} \tag{5.7}$$

Choosing a set of coefficients that satisfies the three equations is not difficult, but to retain monotonicity, all the α_k must be positive by the arguments in the previous section. The last of the three equations has the strictly positive coefficients $(x_k - x_{j+1/2})^2$ for each α_k, which imposes the requirement that some of the α_k must be negative in order for the right-hand side to equal zero. Hence, it is not possible to construct a higher order approximation to the staggered mesh algorithm that is also monotonic.

A modified version of the SHALE algorithm can also be shown to approximate the staggered mesh algorithm. The velocity average and difference across the cell are calculated and advected using mass fluxes,

$$\begin{aligned} p_{j+1/2} &= \tfrac{1}{2}(v_j + v_{j+1}) \\ s_{j+1/2} &= (v_{j+1} - v_j). \end{aligned} \tag{5.8}$$

The velocity of node j is calculated from Eq. (5.9):

$$\begin{aligned} v_j^+ = \frac{1}{2M_j^+} [&M^+_{j-1/2}(p^+_{j-1/2} + \tfrac{1}{2}s^+_{j-1/2}) \\ &+ M^+_{j+1/2}(p^+_{j+1/2} - \tfrac{1}{2}s^+_{j+1/2})]. \end{aligned} \tag{5.9}$$

The algorithm is consistent (s is zero for a uniform velocity) and for the scalar advection problem on a uniform mesh, this algorithm reduces to the SHALE algorithm. Like the SHALE algorithm, it is not monotonic.

The analysis proceeds in the same manner as for the HIS

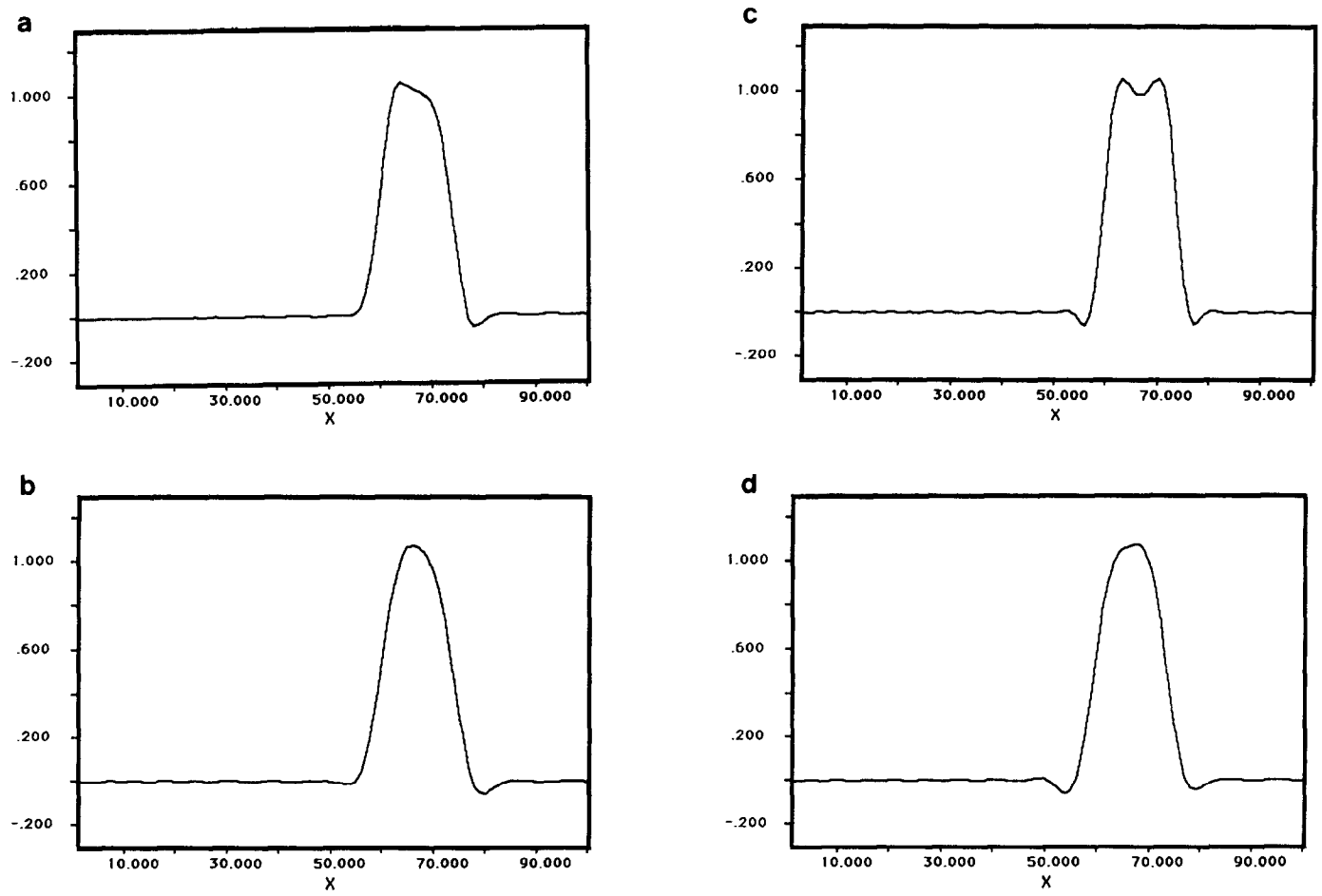


FIG. 5. The SHALE algorithm does not have dispersion problems, but it does not inherit the monotonicity of the underlying cell-centered advection algorithm: (a) p and s with van Leer; (b) p—van Leer, s—donor; (c) p and s with SUPG; (d) p—SUPG, s—donor.

algorithm, but with a few small differences. First, the s and p terms are expressed in terms of their fluxes:

$$M^+_{j-1/2}\,p^+_{j-1/2} = M^-_{j-1/2}\,p^-_{j-1/2} + F(\mathbf{p}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) - F(\mathbf{p}_j, \mathbf{x}_j, \tilde{f}_j)$$
$$M^+_{j-1/2}\,s^+_{j-1/2} = M^-_{j-1/2}\,s^-_{j-1/2} + F(\mathbf{s}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) - F(\mathbf{s}_j, \mathbf{x}_j, \tilde{f}_j). \tag{5.10}$$

Pairs of s and p fluxes, with the appropriate coefficients, are expressed in terms of the nodal velocities through a Taylor series:

$$F(\mathbf{p}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) + \tfrac{1}{2}F(\mathbf{s}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) = F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) + \mathcal{O}(|\Delta\mathbf{v}|^2). \tag{5.11}$$

Results similar to those in Eq. (5.5) immediately follow with the addition of the truncation terms associated with the velocities from Eq. (5.11).

Analyzing the original SHALE algorithm in this manner is complicated by its use of volume fluxes instead of mass fluxes and because the updated cell coordinates appear in the momentum update.

If the average cell velocity is advected and the mass fluxes are used, the SALE algorithm can be viewed as an approximation to the staggered mesh algorithm, but unlike the other algorithms, it has a first-order truncation error. The analysis assumes that the nodal momentum is updated from the increments in the cell momenta. Two terms that cancel are added in the next equation to clarify the steps,

$$v_j^+ = \frac{M_j^-}{M_j^+}\,v_j^- + \frac{1}{2M_j^+}\,[F(\mathbf{p}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) + F(\mathbf{p}_j, \mathbf{x}_j, \tilde{f}_j)] - \frac{1}{2M_j^+}\,[F(\mathbf{p}_j, \mathbf{x}_j, \tilde{f}_j) + F(\mathbf{p}_{j+1}, \mathbf{x}_{j+1}, \tilde{f}_{j+1})]. \tag{5.12}$$

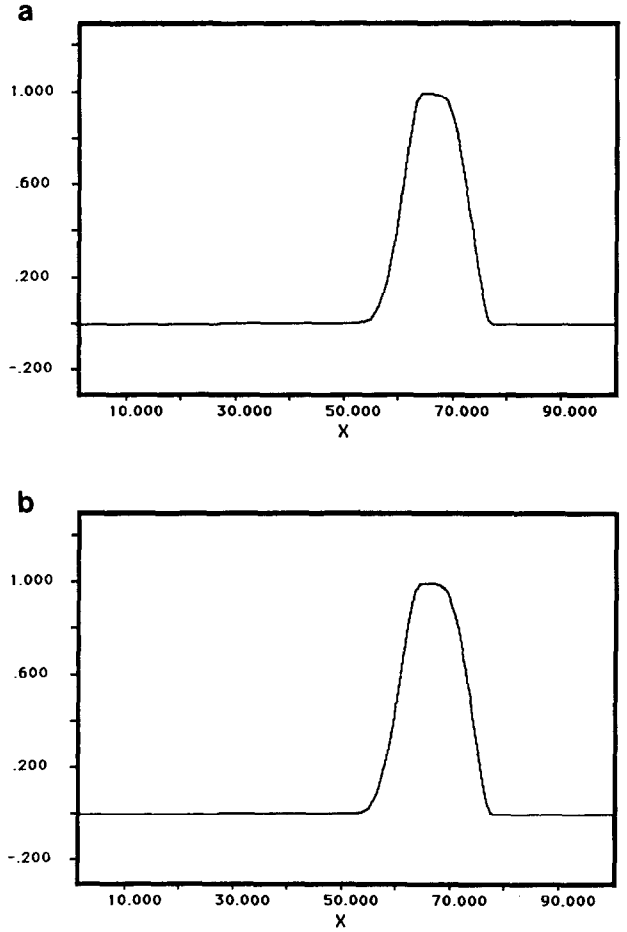


FIG. 6. The HIS algorithm is not very sensitive to mesh distortion. None of the momentum algorithms exhibit a large sensitivity to mesh distortion: (a) van Leer solution on a distorted mesh; (b) HIS van Leer solution on a distorted mesh.

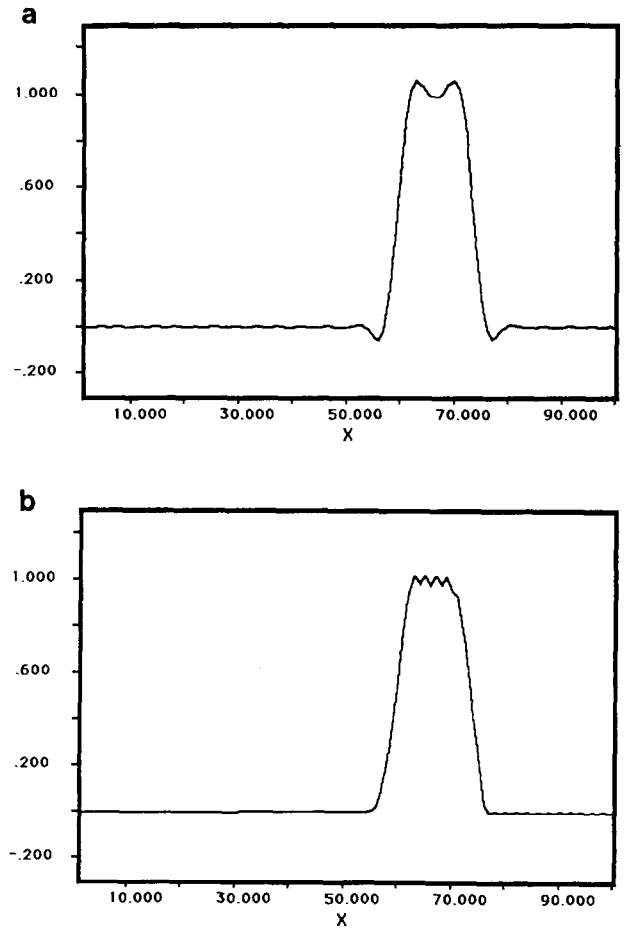


FIG. 7. An exact inverse is used in the SALE algorithm to demonstrate that the approximate inverse is the source of the dispersion errors: (a) SALE SUPG with an exact inverse; (b) SALE van Leer with an exact inverse.

A first-order truncation error is generated by the approximation of $\mathbf{v}_{j+1/2}$:

$$\begin{aligned}
&\tfrac{1}{2}F(\mathbf{p}_{j-1}, \mathbf{x}_{j-1}, \tilde{f}_{j-1}) + \tfrac{1}{2}F(\mathbf{p}_j, \mathbf{x}_j, \tilde{f}_j) \\
&\quad = F(\tfrac{1}{2}(\mathbf{p}_{j-1} + \mathbf{p}_j), \mathbf{x}_{j-1/2}, \tilde{f}_{j-1/2}) \\
&\qquad + \mathcal{O}(|\Delta\mathbf{x}|^2, |\Delta\tilde{f}|^2, |\Delta\mathbf{p}|^2) \\
&\quad = F(\tfrac{1}{4}(\mathbf{v}_{j-3/2} + 2\mathbf{v}_{j-1/2} + \mathbf{v}_{j+1/2}), \mathbf{x}_{j-1/2}, \tilde{f}_{j-1/2}) \\
&\qquad + \mathcal{O}(|\Delta\mathbf{x}|^2, |\Delta\tilde{f}|^2, |\Delta\mathbf{v}|^2) \\
&\quad = F(\mathbf{v}_{j-1/2}, \mathbf{x}_{j-1/2}, \tilde{f}_{j-1/2}) \\
&\qquad + \mathcal{O}(|\Delta\mathbf{v}|) + \mathcal{O}(|\Delta\mathbf{x}|^2, |\Delta\tilde{f}|^2, |\Delta\mathbf{v}|^2). \qquad (5.13)
\end{aligned}$$

All the other errors are second order because the appropriate variables are centered, but the velocity term $\frac{1}{4}(\mathbf{v}_{j-3/2} + 2\mathbf{v}_{j-1/2} + \mathbf{v}_{j+1/2})$ contributes a first-order truncation error because $\mathbf{v}_{j-1/2}$ is not centered between the adjacent velocity vectors.

6. EXAMPLE CALCULATIONS

The Linear Scalar Advection Equation

The dispersion errors discussed in the previous sections are demonstrated by solving the linear scalar advection equation. A square pulse 13 cells wide (cells 10 through 22)

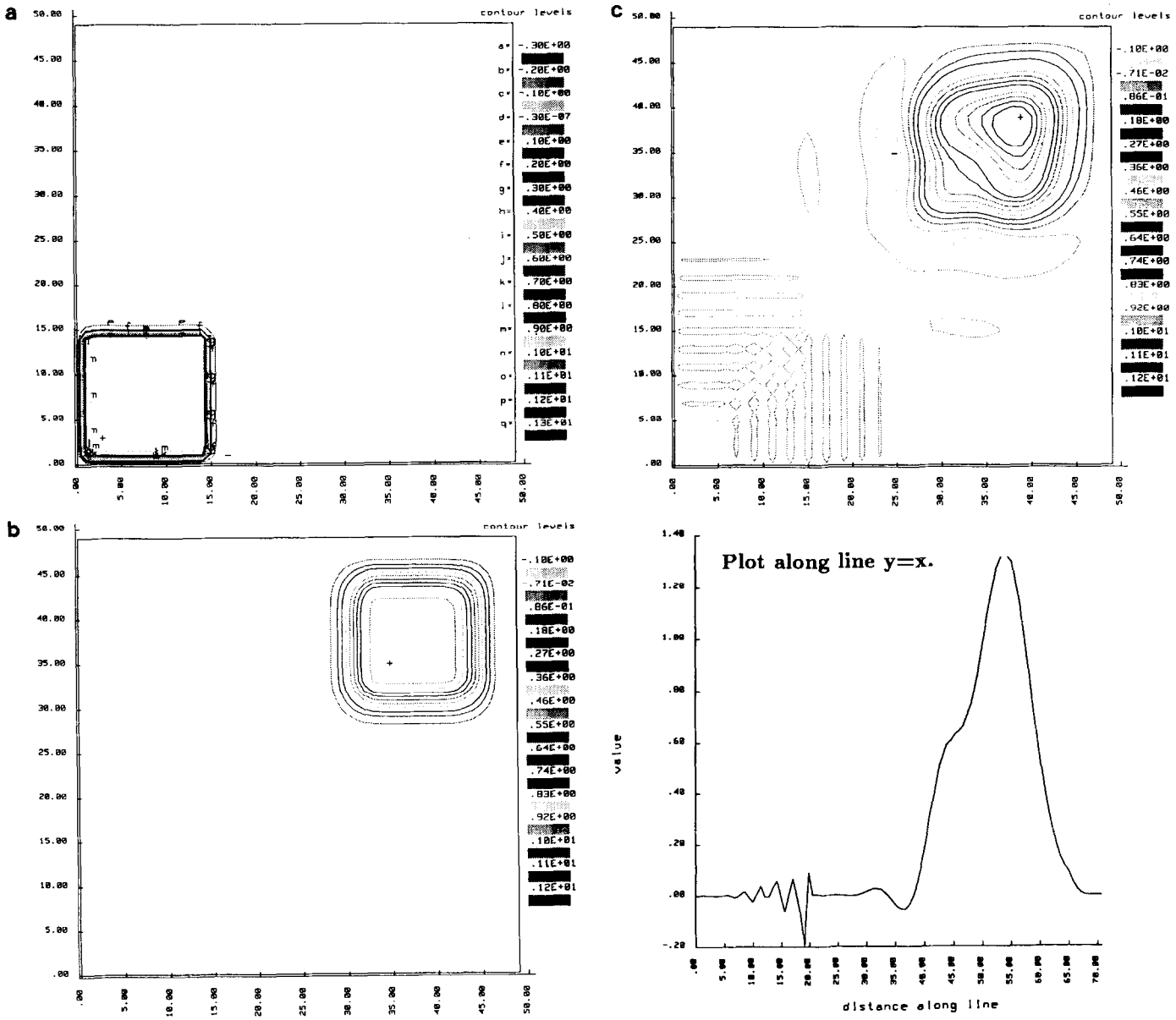


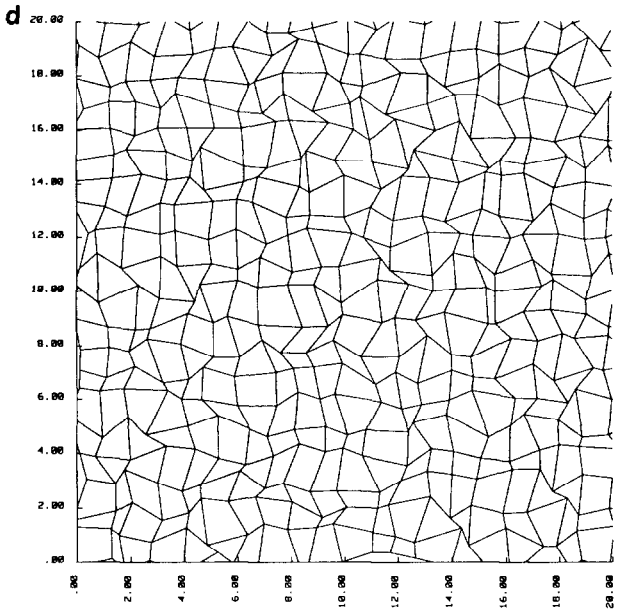
FIG. 8. (a) A square pulse is advected along the mesh diagonal using a spatial operator split. (b) The staggered mesh algorithm solution. (c) The SALE solution exhibits dispersion errors in both directions. (d) The randomly distorted mesh used in the HIS solution. (e) The HIS solution on a distorted mesh.

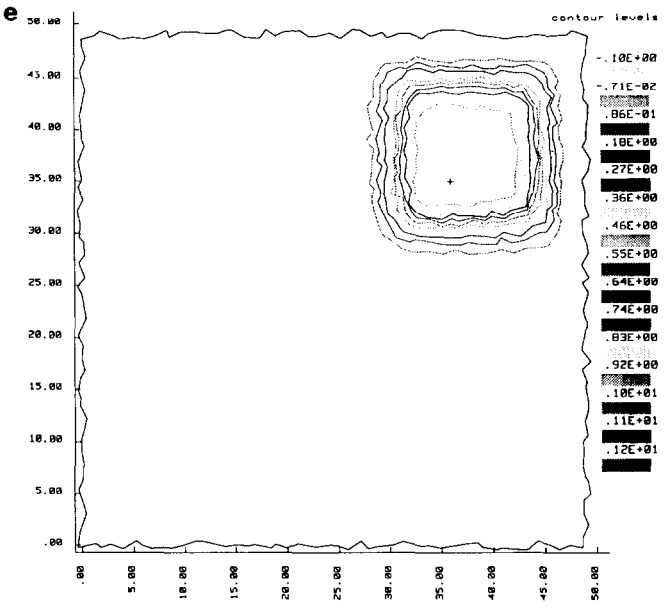
with an amplitude of 1.0 is transported with a velocity of 0.1 for 500 timesteps. There are 100 cells with a uniform width of 1.0.

The solutions obtained by using the van Leer [9] and the SUPG [15] advection algorithms are shown in Fig. 3. The SALE solutions are shown in Fig. 4. Note that the high frequency noise is located at the original position of the pulse and that it appears in both the nonlinear van Leer and the linear SUPG solutions. The van Leer SHALE solutions are shown in Figs. 5a and b, where the van Leer algorithm is used to advect both P and S in Fig. 5a, while the donor cell algorithm is used for S in Fig. 5b. In both cases monotonicity is lost. The SUPG SHALE solutions are shown in Figs. 5c and d, and they demonstrate that the SHALE algorithm does not change the dispersion characteristics of the underlying advection algorithm. Note that the solution in Fig. 5c is identical to the solution in Fig. 3b.

For a uniform mesh, the staggered mesh and HIS algorithms give solutions that are identical to those shown in Fig. 3. To demonstrate the effect of a nonuniform mesh, the square pulse was advected with the cell width defined by Eq. (6.1) for cells 20 through 80. The remaining cells have a width of 1.0:

$$\Delta x_i = 1.0 + 0.8 \sin\left(\frac{2\pi}{60}(i-20)\right). \tag{6.1}$$





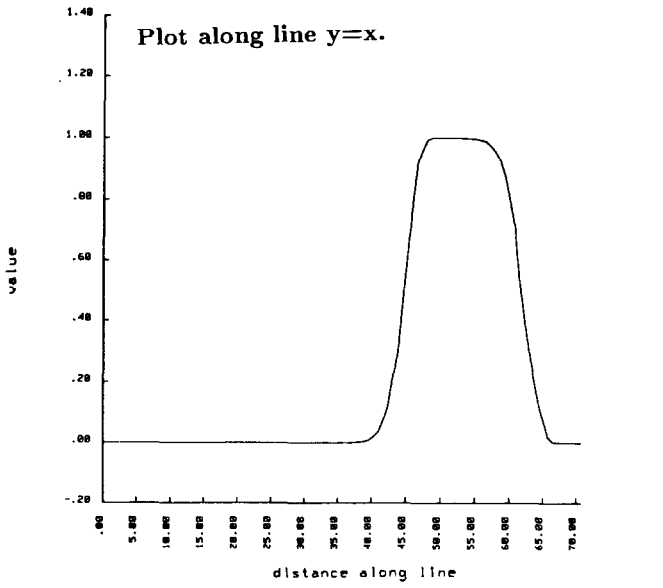


FIG. 8—Continued

The unmodified van Leer solution is shown in Fig. 6a, and the HIS solution, which is almost identical, is shown in Fig. 6b. The staggered mesh solution is virtually identical to Fig. 6b and is not shown. The other two momentum advection algorithms give solutions that are essentially identical to the ones shown in Figs. 4 and 5.

A relationship was demonstrated between the inversion error and the dispersion error in Section 4. The SALE algorithm was modified by exactly inverting the velocity-momentum equation, Eq. (2.18), to verify the relationship. There is one more node than there are cells in one dimension, requiring that the value of v must be assigned at one node. For this demonstration, v_1 is assigned the value of 0.0, and a recursive relation, Eq. (6.2), gives the remaining values,

$$v_{j+1} = 2P_{j+1/2} - v_j. \tag{6.2}$$

As shown in Fig. 7a, using the exact inverse with the linear SUPG algorithm gives results that are identical to using the SUPG algorithm alone, Fig. 3b. Monotonicity is lost when the van Leer algorithm is used, Fig. 7b, but the high frequency oscillations travel with the square pulse and they can not be attributed to dispersion errors. The high frequency oscillations are a result of the nonlinearity of the van Leer algorithm: the sum of two functions advected independently, $\Psi_1^+ + \Psi_2^+$, is not equal to the value of the two functions advected together, $(\Psi_1 + \Psi_2)^+$.

The same errors that appear in one dimension also appear in two dimensions. A square pulse, shown in Fig. 8a, is advected along the mesh diagonal with c equal to 0.25, using the van Leer algorithm and alternating directional sweeps. The staggered mesh solution is shown in Fig. 8b. As shown in Fig. 8c, the SALE algorithm exhibits stationary dispersion errors, just as it did in one dimension. The results for the HIS algorithm on a distorted mesh, a section of which is shown in Fig. 8d, is shown in Fig. 8e.

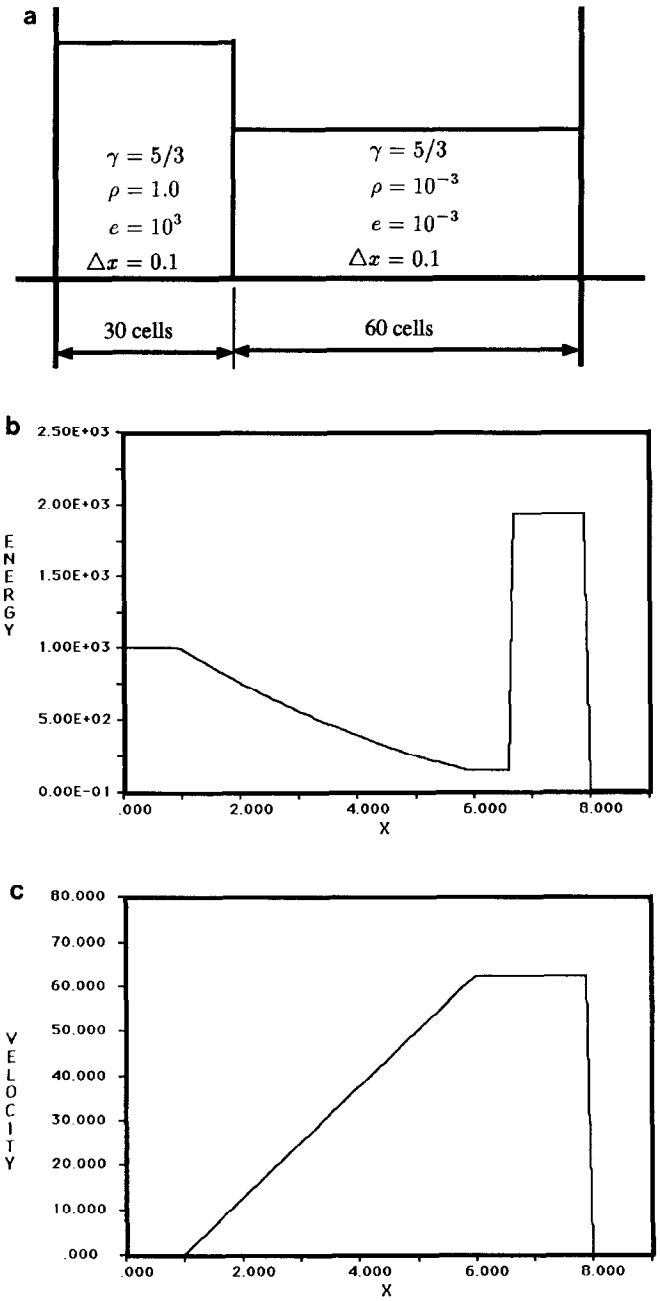


FIG. 9. A strong shock problem on a coarse mesh is solved to demonstrate the differences between the different momentum advection algorithms: At $t = 0.06$, $v = 62.5$, and $e = 1950$ after the shock.

The Shocktube Problem

Although the dispersion and monotonicity problems associated with cell-centered momentum advection algorithms have been demonstrated both analytically and numerically, the real value of the algorithms is dependent on how well they work for physical problems. To magnify the differences between the different algorithms, a strong shock problem is solved on a coarse mesh. The data for this problem, see Fig. 9a, were suggested by Christensen [21], and the analytical solutions are shown in Figs. 9b and c. A single material is used in the calculations, with the initial properties of the gas being spatially dependent. For problems less challenging than this one, all the algorithms give essentially identical solutions.

The Lagrange step used here is fairly standard. The only departure from standard practice is the flux-limited shock viscosity which was developed by Christensen [21, 32]. The constants C_1 and C_2 are set to 1.33 and 1.00, a is the soundspeed, and $\nabla_j^{\mathrm{VL}} u$ is the van Leer-limited velocity gradient at node j. This viscosity gives a sharper shock front than the standard formulations:

$$q = \begin{cases} \rho(C_1[u]^2 - C_2\, a[u]) & \text{if } [u] \leqslant 0 \\ 0 & \text{if } [u] > 0 \end{cases} \tag{6.3}$$
$$[u] = u_{j+1} - u_j + \tfrac{1}{2}\Delta x(\nabla_{j+1}^{\mathrm{VL}} u + \nabla_j^{\mathrm{VL}} u).$$

The Eulerian step uses the van Leer advection algorithm. The density and internal energy are advected with transport volumes. Energy conservation is enforced by approximating the kinetic energy lost by the momentum advection and adding it to the internal energy. This update is performed only on the cells that have a nonzero shock viscosity. The cell kinetic energy per unit mass, t, is defined as $\frac{1}{4}(v_j^2 + v_{j+1}^2)$ and it is advected using mass coordinates and transport masses. The average kinetic energy per unit mass transported through node j by the momentum advection, t', is defined as $\frac{1}{4}((v_j^+)^2 + (v_j^-)^2)$. The increment in the total internal energy of cell $j + \frac{1}{2}$, $\Delta E_{j+1/2}$, is calculated according to Eq. (6.4),

$$\Delta E_{j+1/2} = \max[0, \tilde{f}_j(t_j - t'_j) - \tilde{f}_{j+1}(t_{j+1} - t'_{j+1})]. \tag{6.4}$$

The two similar solutions obtained by using the staggered mesh algorithm with mass and volume coordinates are shown in Figs. 10a and b, respectively. In both cases consistent transport masses are used for the momentum advection. The only difference between the mass and volume coordinate solutions is which coordinates are used in the van Leer algorithm to calculate the velocities. The solution obtained with mass coordinates is slightly closer to the theoretical value of 1950 for the internal energy at the shock.

The SALE solutions, using transport masses and volumes, are shown in Figs. 11a and b, respectively. The internal energy is higher in the transport volume solution than in the transport mass solution, a feature that is found to be common to all algorithms. A sharp velocity peak occurs in both solutions at the front of the shock.

Unfortunately, the SHALE algorithm, as the author understands it, did not run the problem to completion. The modified version that was presented in Section 5 worked quite well. The solution obtained with the modified algorithm is shown in Fig. 12a, and it compares favorably to the staggered mesh solution. Both s and p were advected with the van Leer algorithm; using the donor cell algorithm for s made little difference in the solution. A second solution was obtained by scaling s and p by the cell density and then advecting them with the transport volumes instead of the transport masses; see Fig. 12b. The volume scheme also

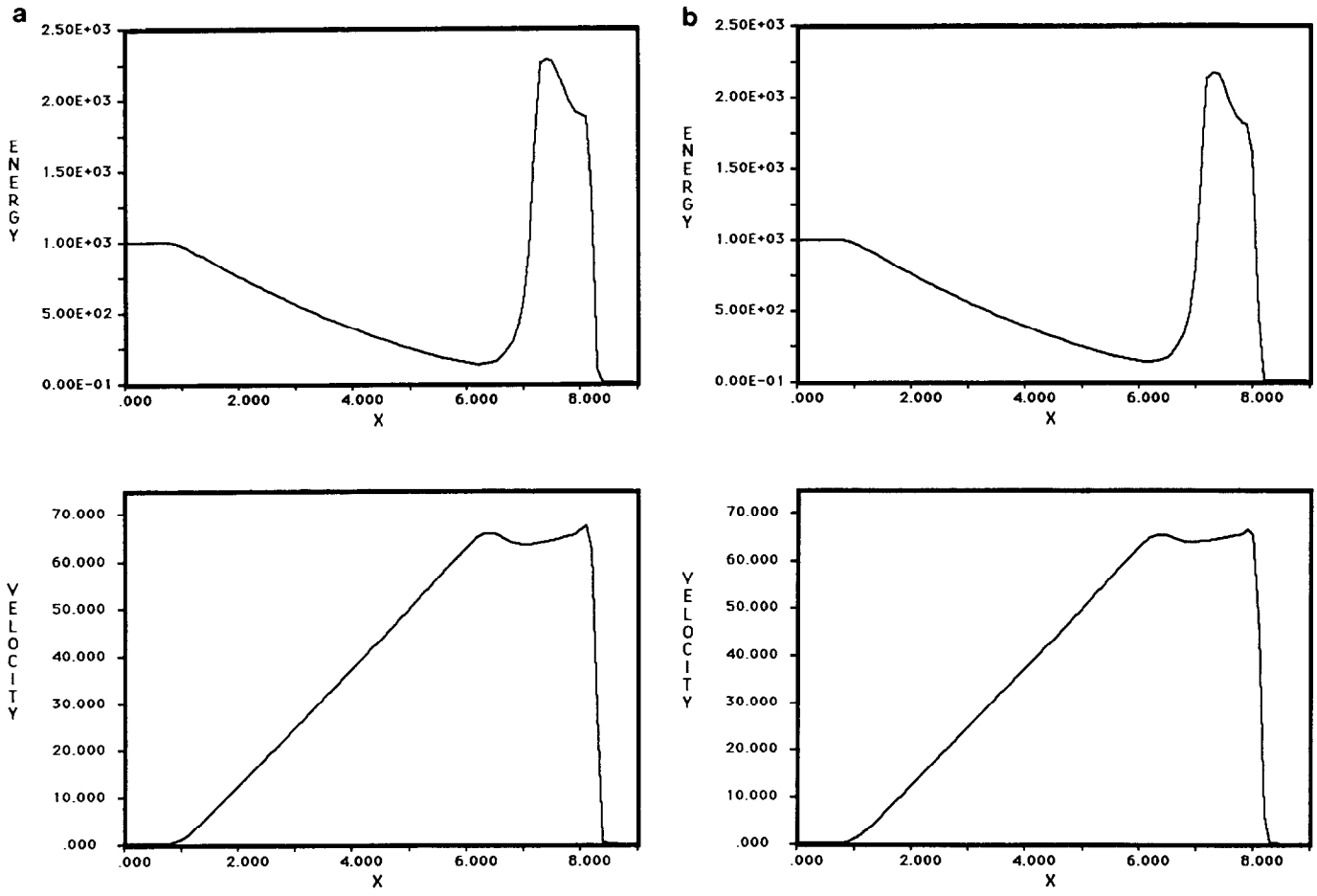


FIG. 10. The YAQUI staggered mesh algorithm gives the best solution; (a) with mass coordinates; (b) with volume coordinates.

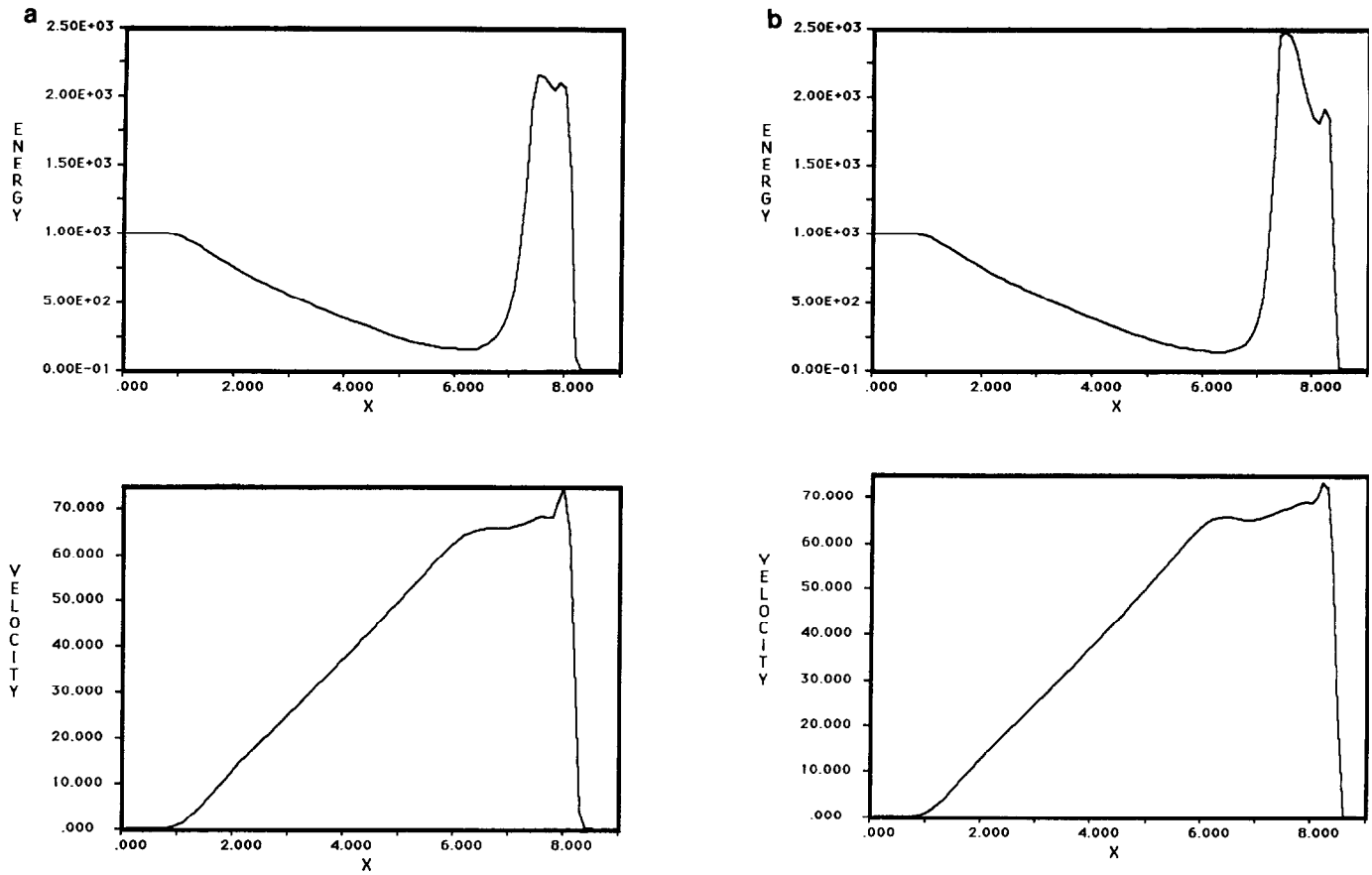


FIG. 11. The SALE algorithm is the most efficient, but the solution is not as good as the other cell-centered solutions obtained with transport masses: (a) with mass coordinates and fluxes; (b) with volume coordinates and fluxes.

satisfies DeBar's consistency condition: a uniform velocity will remain uniform. Both the internal energy and the velocity are too high by at least 25%.

A similar result is obtained for the HIS algorithm; see Fig. 13. The transport mass algorithm works, but the transport volume scheme (the right-hand side of Eq. (4.4) is scaled by $\rho_{j+1/2}$) gives a very poor solution. In fact, the modified SHALE and HIS algorithms give nearly identical errors when transport volumes are used because they approximate each other to within second-order truncation errors. An attempt was made to run the staggered mesh algorithm with transport volumes, but the problem could not be run to completion because the shock overheating was so extreme that the time step became almost zero.

7. CONCLUSIONS AND RECOMMENDATIONS

The staggered mesh algorithm with consistent transport masses is attractive for a logically regular mesh based on both the analysis and the numerical experiments. It is more difficult to implement on an arbitrary mesh than a cell-centered algorithm because a staggered cell may have an arbitrary number of edges.

The SALE algorithm is the most economical cell-centered algorithm, but it adds extra dispersion errors and it does not inherit the monotonicity properties of the underlying cell-centered advection algorithm. It does, however, work surprisingly well in strong shock calculations. When transport volumes are used, it gives the least objectionable answers of any of the cell-centered algorithms. For most simplified ALE calculations, where the Courant number is relatively low, this algorithm is more than adequate.

The modified SHALE, HIS, and staggered mesh solutions are almost identical when transport masses are used. They give much larger errors than the SALE algorithm when the transport volumes are used. Unlike the SALE and staggered mesh algorithms, the HIS and SHALE algorithms require the advection of more than one variable for each velocity component. Of the three cell-centered momentum advection algorithms, only the HIS algorithm inherits the dispersion and monotonicity characteristics of the underlying advection algorithm, a definite advantage over the other two algorithms. The relative computational cost of

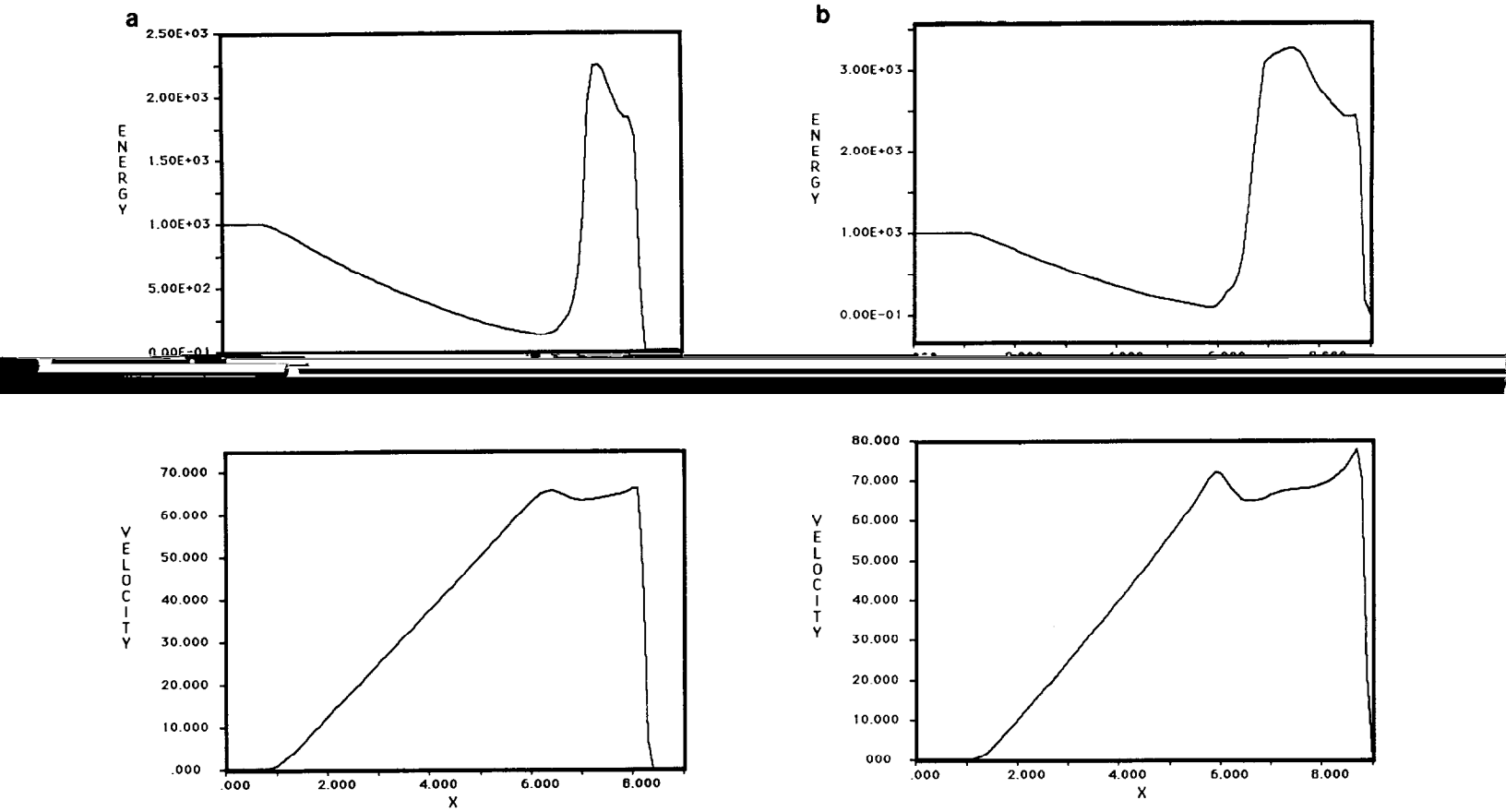


FIG. 12. The similarity between the solution obtained by using a modified version of the SHALE algorithm with transport masses and the YAQUI solution verifies that the SHALE algorithm is an approximation of the YAQUI algorithm. When the transport volumes are used to advect the momentum, there is a large amount of shock overheating. (a) with mass coordinates and fluxes; (b) with volume coordinates and fluxes.

the algorithms is heavily dependent on the overall architecture of the programs. For example, the architecture of DYNA2D seems to favor the staggered mesh algorithm, while Margolin reports that cell-centered algorithms are faster in SHALE [23].

An interesting aspect of the SHALE and HIS algorithms is that they both satisfy the consistency condition with transport volumes, and they do not add dispersion errors, but they still give poor answers with transport volumes and good answers with transport masses. Based on the strong shock calculations shown here, it is clear that transport masses should be used with the staggered mesh, SHALE, and HIS algorithms.

8. SUMMARY

Three momentum advection algorithms were reviewed, and the HIS algorithm was introduced. The dispersion and monotonicity characteristics of the four algorithms were analyzed. All the momentum advection algorithms reviewed here are adequate for strong shock calculations provided that transport masses are used. The staggered mesh and HIS algorithms are the only two that inherit both the monotonicity and the dispersion characteristics of the underlying cell-centered advection algorithm. The SHALE, HIS, and staggered mesh advection algorithms closely approximate each other, based on their Taylor expansions. Cell-centered algorithms that use transport volumes and which also satisfy the consistency condition were shown to produce large errors in strong shock calculations.

APPENDIX

The van Leer Advection Algorithm

Van Leer has contributed quite heavily to the literature on advection, and the purpose of this section is to define the specific algorithm that is labelled the "van Leer algorithm" within this paper. The algorithm used here is often referred to as the van Leer MUSCL algorithm [9].

The essential idea is to replace the piecewise constant distribution of ϕ within a cell with a piecewise linear distribution. The coordinate x can be a volume or mass coordinate.

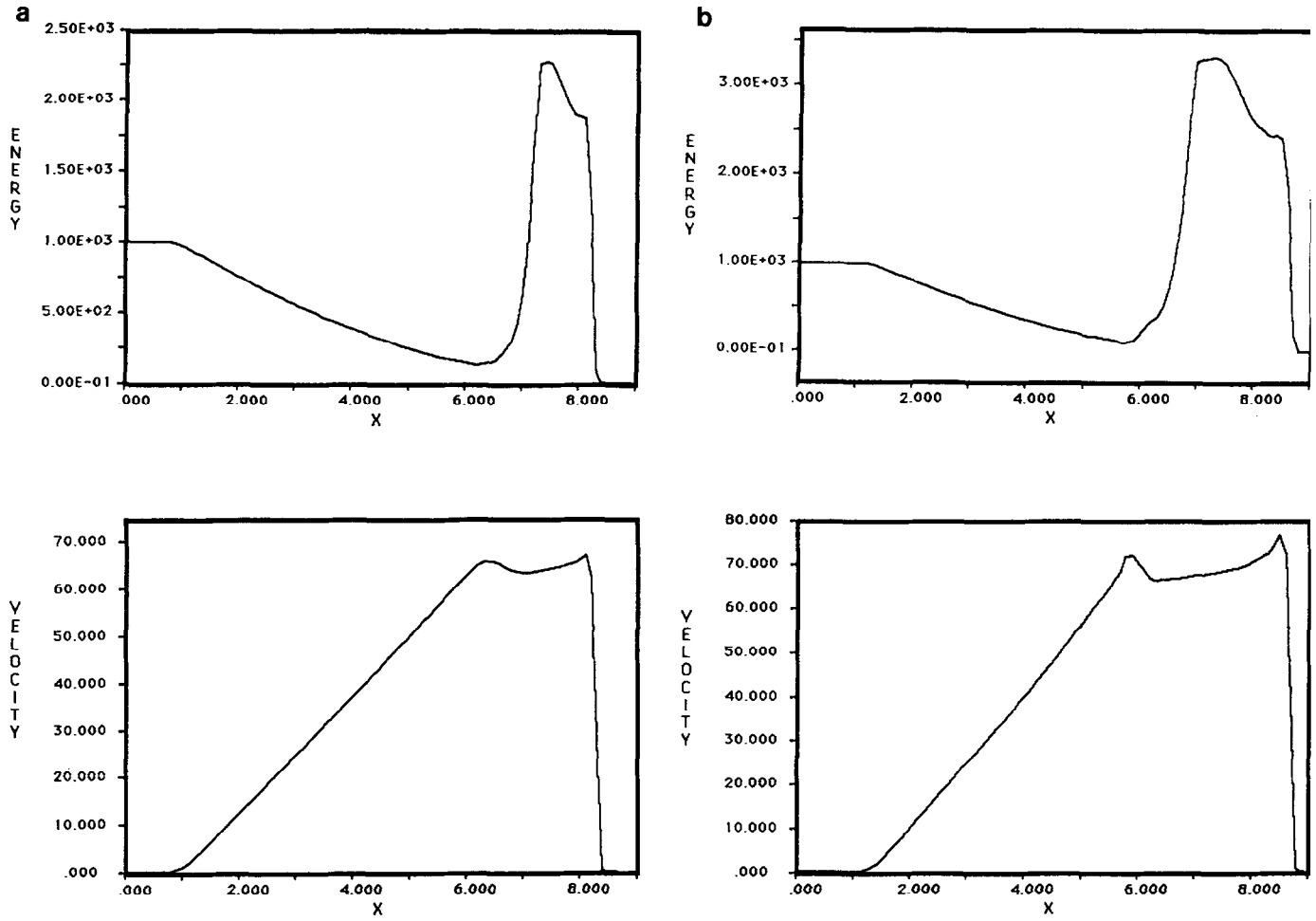


FIG. 13. Like the SHALE solution, the HIS solution using transport masses is almost identical to the YAQUI solution. The errors in the SHALE and HIS solutions are almost identical when the transport volumes are used to advect momentum. (a) with mass coordinates and fluxes; (b) with volume coordinates and fluxes.

The value of ϕ at $x_{j+1/2}$ is $\bar{\phi}_{j+1/2}$, and the gradient of ϕ is $s^{\phi}_{j+1/2}$,

$$\phi(x) = \bar{\phi}_{j+1/2} + s^{\phi}_{j+1/2}(x - x_{j+1/2}). \tag{A.1}$$

The van Leer algorithm is an upwind algorithm. When material is transported from cell $j-\frac{1}{2}$ to $j+\frac{1}{2}$, the value of ϕ_j, associated with f_j, is evaluated using the linear distribution in cell $j-\frac{1}{2}$. If the direction of the flow is reversed, then the distribution in cell $j+\frac{1}{2}$ is used.

The values of ϕ_{j+1} and ϕ_j, which are evaluated using Eq. (A.1) at the centroids f_{j+1} and f_j, are restricted to lie within the maximum and minimum values of $\phi^{-}_{j-1/2}$, $\phi^{-}_{j+1/2}$, and $\phi^{-}_{j+3/2}$, in order to maintain monotonicity. Letting the initial width of the cell be $\Delta x_{j+1/2}$ and the width of f_{j+1} and f_j be Δx_j and Δx_{j+1}, the magnitude of the slope must satisfy the inequality given by Eq. (A.2),

$$\operatorname{abs}(s^{\phi}_{j+1/2}) \leqslant 2 \min\left[\operatorname{abs}\left(\frac{\phi^{-}_{j+3/2} - \phi^{-}_{j+1/2}}{\Delta x_{j+1/2} - \max(0, \Delta x_{j+1})} \right), \times \operatorname{abs}\left(\frac{\phi^{-}_{j+1/2} - \phi^{-}_{j-1/2}}{\Delta x_{j+1/2} + \min(0, \Delta x_j)} \right) \right]. \tag{A.2}$$

The inclusion of the terms involving the max and min functions allows steeper slopes than would otherwise be obtained while still satisfying the monotonicity condition.

The sign of the slope must be the same as the sign of the slope of the monotonic function, and it is set to zero if it is a local minimum of maximum:

$$\operatorname{sgn}(s^{\phi}_{j+1/2}) = \tfrac{1}{2}[\operatorname{sgn}(\phi_{j+3/2} - \phi_{j+1/2}) + \operatorname{sgn}(\phi_{j+1/2} - \phi_{j-1/2})]. \tag{A.3}$$

To achieve second-order accuracy in regions that have

smooth solutions, a second-order approximation of the slope is calculated by fitting a parabola through the values of ϕ in the cells. The secon-order approximation is $s^{C}_{j+1/2}$, while the two first-order slopes appearing in Eq. (A.4) are $s^{L}_{j+1/2}$ and $s^{R}_{j+1/2}$:

$$
\begin{aligned}
s^{C}_{j+1/2} &= \frac{(\Delta x^{R}/\Delta x^{L})\,\phi^{L} + (\Delta x^{L}/\Delta x^{R})\,\phi^{R}}{\Delta x^{L} + \Delta x^{R}} \\
s^{L}_{j+1/2} &= \frac{2\phi^{L}}{\Delta x_{j+1/2} + \min(0, \Delta x_{j})} \\
s^{R}_{j+1/2} &= \frac{2\phi^{R}}{\Delta x_{j+1/2} - \max(0, \Delta x_{j+1})} \\
\phi^{R} &= \bar{\phi}_{j+3/2} - \bar{\phi}_{j+1/2} \\
\phi^{L} &= \bar{\phi}_{j+1/2} - \bar{\phi}_{j-1/2} \\
x^{R} &= x_{j+3/2} - x_{j+1/2} \\
x^{L} &= x_{j+1/2} - x_{j-1/2}.
\end{aligned}
\tag{A.4}
$$

The SUPG Advection Algorithm

The streamline upwind Petrov–Galerkin (SUPG) method [15] is an implicit algorithm for the Navier–Stokes equations. It was developed by Hughes and his colleagues using finite element techniques. When it is reduced to an advection algorithm, it is linear and not monotonic. Strictly speaking, it is only applicable to nodal variables, but because the mesh and the velocity are uniform for the dispersion analysis, it can be applied in this case. The general form of the algorithm in one dimension is given by Eq. (A.5),

$$
\int W_{k}\left(\dot{\phi} + c\frac{\partial \phi}{\partial x}\right) dx = 0. \tag{A.5}
$$

The weighting function, W_k, and the interpolation of ϕ are given by Eq. (A.6). The optimal value of α is $1/\sqrt{15}$ for the pure advection problem:

$$
\begin{aligned}
W_{k} &= N_{k} + c\alpha\frac{\partial N_{k}}{\partial x} \\
\phi &= \phi_{j} N_{j}(x) \\
N_{j}(x) &= \begin{cases} \dfrac{x_{j+1} - x}{x_{j+1} - x_{j}} & \text{if} \quad x_{j+1} \geqslant x \geqslant x_{j} \\ \dfrac{x - x_{j-1}}{x_{j} - x_{j-1}} & \text{if} \quad x_{j-1} \leqslant x \leqslant x_{j} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{A.6}
$$

SUPG is a semidiscrete algorithm, requiring that a choice be made for integrating in time a set of ordinary differential equations. A second-order accurate midpoint scheme is used here. The superscript refers to the timestep number,

$$
\dot{\phi}^{n+1/2}_{j} = \frac{(\phi^{n+1}_{j} - \phi^{n}_{j})}{\Delta t}, \tag{A.7}
$$

$$
\phi^{n+1/2}_{j} = \frac{(\phi^{n+1}_{j} + \phi^{n}_{j})}{2}. \tag{A.8}
$$

The difference stencil in Eq. (A.9) is obtained by substituting Eq. (A.7) and Eq. (A.8) into Eq. (A.6):

$$
K_{1}\phi^{n+1}_{j-1} + K_{2}\phi^{n+1}_{j} + K_{3}\phi^{n+1}_{j+1} = R_{1}\phi^{n}_{j-1} + R_{2}\phi^{n}_{j} + R_{3}\phi^{n}_{j+1} \tag{A.9}
$$

$$
\begin{aligned}
K_{1} &= \frac{1}{\Delta t}\left(\frac{1}{6} + \frac{1}{2}\alpha\right) - \frac{1}{2}c\left(\frac{1}{2} + \alpha\right) \\
K_{2} &= \frac{2}{3\,\Delta t} + c\alpha \\
K_{3} &= \frac{1}{\Delta t}\left(\frac{1}{6} - \frac{1}{2}\alpha\right) + \frac{1}{2}c\left(\frac{1}{2} - \alpha\right) \\
R_{1} &= \frac{1}{\Delta t}\left(\frac{1}{6} + \frac{1}{2}\alpha\right) + \frac{1}{2}c\left(\frac{1}{2} + \alpha\right) \\
R_{2} &= \frac{2}{3\,\Delta t} - c\alpha \\
R_{3} &= \frac{1}{\Delta t}\left(\frac{1}{6} - \frac{1}{2}\alpha\right) - \frac{1}{2}c\left(\frac{1}{2} - \alpha\right).
\end{aligned}
\tag{A.10}
$$

The von Neumann analysis presented in the paper was performed on the assumption that the advection algorithm is explicit, but it is readily extended to implicit advection algorithms. Letting $\mathscr{P}^{R}$ and $\mathscr{P}^{L}$ denote the complex polynomials associated with stencils on the left- and right-hand sides of the equation, the equivalent form for $\mathscr{P}$ is derived in Eq. (A.11),

$$
\begin{aligned}
\mathscr{P}^{R} v^{j} &= \mathscr{P}^{L} v_{j} \\
v^{j} &= \frac{\mathscr{P}^{L}}{\mathscr{P}^{R}} v_{j} = (1 + \mathscr{P})\, v_{j} \\
\mathscr{P} &= \frac{\mathscr{P}^{L}}{\mathscr{P}^{R}} - 1.
\end{aligned}
\tag{A.11}
$$

ACKNOWLEDGMENTS

I thank Nippon Oil and Fats, Ltd. for funding this work. Both Dr. R. Christensen and Dr. Margolin of Lawrence Livermore National Laboratory gave invaluable advice during many long discussions.

REFERENCES

1. A. A. Amsden, H. M. Ruppel, and C. W. Hirt, Los Almos Scientific Laboratory Report No. LA-8095, 1980 (unpublished).
2. D. Benson, *Comput. Methods. Appl. Mech. Eng.* **72**, 305 (1989).
3. M. L. Wilkins, in *Methods in Computational Physics, Vol. 3, Fundamental Methods in Hydrodynamics* (Academic Press, New York, 1964), p. 211.
4. G. Maenchen and S. Sack, in *Methods in Computational Physics, Vol. 3, Fundamental Methods in Hydrodynamics* (Academic Press, New York, 1964), p. 181.
5. W. F. Noh, in *Methods in Computational Physics, Vol. 3, Fundamental Methods in Hydrodynamics* (Academic Press, New York, 1964), p. 117.
6. T. J. R. Hughes, *The Finite Element Method, Linear, Static and Dynamic Finite Element Analysis* (Prentice–Hall, Englewood Cliffs, NJ, 1987).
7. O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, 3rd ed. (McGraw–Hill, London, 1989).
8. J. O. Hallquist, Lawrence Livermore National Laboratory Report UCID-19401, 1982 (unpublished).
9. B. Van Leer, *J. Comput. Phys.* **23**, 276 (1977).
10. F. L. Addessio, D. E. Carroll, J. K. Dukowicz, F. H. Harlow, J. N. Johnson, B. A. Kaskiwa, M. E. Maltrud, and H. M. Ruppel, Los Almos National Laboratory Report UC-32, 1988 (unpublished).
11. S. L. Thompson, Sandia National Laboratories Report SAND74-0122, 1975 (unpublished).
12. J. O. Hallquist, Lawrence Livermore National Laboratory Report UCID-18756 Rev. 2, 1984 (unpublished).
13. S. Hancock, *PISCES 2DELK Theoretical Manual*, Physics International, 1985 (unpublished).
14. R. B. Demuth, L. G. Margolin, B. D. Nichols, T. F. Adams, and B. W. Smith, Los Almos National Laboratory Report LA-10236, 1985 (unpublished).
15. A. N. Brooks and T. J. R. Hughes, *Comput. Meths. Appl. Mech. Eng.* **32**, 199 (1982).
16. A. A. Amsden and C. W. Hirt, Los Almos Scientific Laboratory Report LA-5100, 1973 (unpublished).
17. L. G. Margolin and C. W. Beason, "Remapping on the Staggered Mesh," Fifth Nuclear Code Developers' Conference, October 11–14, 1988, Boulder, CO.
18. R. B. DeBar, Lawrence Livermore Laboratory Report UCIR-760, 1974 (unpublished).
19. P. J. Roache, *Computational Fluid Dynamics* (Hermosa, Albuguergue, New Mexico, 1976).
20. R. W. Sharp and R. T. Barton, Lawrence Livermore National Laboratory Report UCID-17809 Rev. 1, 1981 (unpublished).
21. R. Christensen, Lawrence Livermore National Laboratory, Livermore, CA, private communication (1989).
22. L. N. Trefethen, *SIAM Rev.* **24**, No. 2 (1982).
23. L. G. Margolin, Lawrence Livermore National Laboratory, Livermore, CA, private communication (1989).
24. D. P. Flanagan and T. Belytschko, *Int. J. Num. Methods Eng.* **17**, 679 (1981).
25. L. G. Margolin and J. J. Pyun, in *Proceedings, Fourth International Conference on Numerical Methods in Laminar and Turbulent Flow, Montreal, Quebec, Canada, July 6–10, 1987.*
26. D. Kosloff and G. Frazier, *Num. Anal. Methods Geomech.* **2**, 57 (1978).
27. A. Harten, *J. Comput. Phys.* **49**, 357 (1983).
28. S. T. Zalesak, *J. Comput. Phys.* **31**, 335 (1979).
29. P. K. Smolarkiewicz, *J. Comput. Phys.* **54**, 325 (1984).
30. J. B. Bell, C. N. Dawson, and G. R. Shubin, *J. Comput. Phys.* **74**, 1 (1988).
31. B. Van Leer, *Computing Methods in Applied Sciences and Engineering, VI* (Elsevier Science, New York, 1984), p. 493.
32. D. J. Benson, A new two-dimensional flux-limited shock viscosity, *Comput. Methods Appl. Mech. Eng.*, submitted.