THE ASTROPHYSICAL JOURNAL SUPPLEMENT SERIES, 80:753–790, 1992 June

ZEUS-2D: A RADIATION MAGNETOHYDRODYNAMICS CODE FOR ASTROPHYSICAL FLOWS IN TWO SPACE DIMENSIONS. I. THE HYDRODYNAMIC ALGORITHMS AND TESTS

JAMES M. STONE[1] AND MICHAEL L. NORMAN[1]
National Center for Supercomputing Applications, 5600 Beckman Institute, Drawer 25, University of Illinois at Urbana-Champaign,
405 North Mathews Avenue, Urbana, IL 61801
Received 1991 January 9; accepted 1991 October 29

ABSTRACT

In this, the first of a series of three papers, we begin a detailed description of ZEUS-2D, a numerical code for the simulation of fluid dynamical flows in astrophysics including a self-consistent treatment of the effects of magnetic fields and radiation transfer. The algorithms in ZEUS-2D divide naturally into three areas: (1) hydrodynamics (HD), (2) magnetohydrodynamics (MHD), and (3) radiation hydrodynamics (RHD). In this first paper, we give a detailed description of the HD algorithms which form the foundation for the more complex MHD and RHD algorithms.

We use simple, well-developed Eulerian HD algorithms based on the method of finite-differences implemented in a new covariant formalism which allows simulation in any orthogonal coordinate system. The effect of self-gravity on the flow dynamics is accounted for by an iterative solution of the sparse-banded matrix resulting from discretizing the Poisson equation in multidimensions. The results of an extensive series of HD test problems are presented.

Subject headings: hydrodynamics — methods: numerical — MHD — radiative transfer

1. INTRODUCTION

The ubiquity of hydrodynamical (HD) flows in astrophysical systems demonstrates the importance of fluid dynamical modeling to the field. HD processes can be used to describe the astrophysics of systems over a large range in scales, from stars and compact objects to the interstellar or even intergalactic medium. Since the equations of HD generally cannot be solved analytically without restrictive assumptions, much of the theoretical modeling relies on the numerical solution of these equations. Many numerical methods have been developed to generate solutions to the equations of HD, however one of the most well developed and straightforward is the method of finite-differences. Finite-differencing is attractive for many reasons: it it robust, easy to code, efficient, and easily modified and extended with new algorithms.

One of the most important frontiers for theoretical modeling is the inclusion of all of the physical effects which may be dynamically important to the system under study. For instance, the Galactic magnetic field can play a dominant role in the evolution of the interstellar medium via magnetohydrodynamical (MHD) processes, while the physics of accretion flows can be dominated by both the strong magnetic and radiation field from the central object. Realistic theoretical models of such systems can only be produced when the numerical methods adopted to solve the problem treat all of the physical effects in a self-consistent fashion.

With this in mind, we have developed a general purpose fluid dynamics code, called ZEUS-2D, for modeling astrophysical systems in two space dimensions including the effects of magnetic fields and radiation transfer. Generality has been the driving philosophy behind the development of the ZEUS-2D code so that the code may be used to model an extremely wide variety of astrophysical systems. Thus, the HD algorithms in ZEUS-2D have been selected as the most accurate methods to which more complex physical effects can be easily added in a self-consistent fashion. The numerical treatment of the MHD in ZEUS-2D assumes flux-freezing, but it is otherwise general. Great care has been taken to ensure that the algorithm allows for the proper treatment of all wave families in MHD, while simultaneously ensuring that the numerically evolved field satisfies the divergence free constraint at all times. The treatment of the radiation transfer in ZEUS-2D is based on a full transport algorithm. By actually solving the radiation transfer equation to close the equations of radiation hydrodynamics (RHD), we can avoid making the diffusion approximation. Thus, the method is equally applicable to both optically thin and thick media without requiring the use of flux limiters.

In this and two other papers (Stone & Norman 1992, hereafter Paper II; and Stone, Mihalas, & Norman 1992, hereafter Paper III) we provide a comprehensive description of the algorithms, methods, and test problems the authors have used in developing the ZEUS-2D code. At its most fundamental level, ZEUS-2D is a hydrocode, that is, it is an implementation of certain specific numerical algorithms for solving the equations of HD. Thus, in this first paper we focus on these HD algorithms and their implementation in ZEUS-2D. These algorithms form the foundation upon which are built the more complex MHD algorithms (described in Paper II) and RHD algorithms (described in Paper III). The division of the discussion into three papers not only

[1] Also Department of Astronomy, University of Illinois at Urbana-Champaign.

follows the natural organization of the subject matter, but also will allow those only interested in the MHD or RHD algorithms to circumvent the rest of the discussion. The level of the discussion in each paper is intended to be comprehensive enough to allow others to implement the methods, as well as provide a platform for the intercomparison of the results from other methods. Results from applying the code to specific astrophysical problems will not be presented in these papers, but are given elsewhere (Stone 1990).

We begin the description of the HD algorithms in ZEUS-2D in § 2 by writing down the equations of HD which are actually solved in the code. In § 3 we describe the historical roots of the ZEUS-2D code, and the factors which govern the choice of the particular algorithms we have used. Most of this paper is devoted, in § 4, to a detailed description of the ZEUS-2D code. In § 5, we present the results of the tests we have performed, and in § 6 we summarize.

2. THE EQUATIONS OF HYDRODYNAMICS

Astrophysical plasmas are typically composed of dilute mixtures of atoms, molecules, ions and electrons. At any instant of time, the state of the system can be completely described at the microscopic level by specifying the distribution function (phase space density). The time evolution of the system is then given by evolving this distribution function using the Boltzmann equation. However, whenever interparticle collisions are very frequent (at least so frequent that the mean free path of a particle is much smaller than a scale length in the system), then one can use a much simpler, statistical description of the plasma involving only conserved, macroscopic quantities. The time evolution of these quantities is given by velocity moments of the Boltzmann equation. Taking successively higher order moments leads to the familiar equations of HD,

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\boldsymbol{v} = 0\,, \tag{1}$$

$$\rho\frac{D\boldsymbol{v}}{\mathrm{D}t} = -\nabla p - \rho\nabla\Phi\,, \tag{2}$$

$$\rho\frac{D}{Dt}\left(\frac{e}{\rho}\right) = -p\nabla\cdot\boldsymbol{v}\,. \tag{3}$$

Here, the dependent variables are the mass density ρ, the velocity $\boldsymbol{v}$, and the internal energy density e. The D/Dt denotes the Lagrangean or comoving derivative,

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \boldsymbol{v}\cdot\nabla\,. \tag{4}$$

The fluid equations are closed with an equation of state which gives the gas pressure p as a function of the mass and internal energy densities (or equivalently the mass density and temperature), and the Poisson equation which determines the gravitational potential Φ,

$$\nabla^2\Phi = 4\pi G\rho\,. \tag{5}$$

The fluid equations are simply statements of macroscopic conservation laws. For example, the continuity equation (1) expresses conservation of mass, while the equations of motion (2) express conservation of momenta. In this work we use the internal energy equation (3) rather than the conservation law for the total energy to improve the accuracy of the internal energy (and temperature) for highly supersonic flows.

The above discussion is not intended to give a rigorous derivation of the fluid equations, as it glosses over the many issues and subtleties of classical kinetic theory. For our purposes, it is sufficient to note that for many astrophysical gas dynamic problems one can show that interparticle collisions are frequent enough to allow one to use the continuum (macroscopic) description of the system given by equations (1)–(5). Even so, there are many physical processes and effects which are not accounted for by these equations. We will describe the addition of two of these effects, namely magnetic fields and radiation transport, in Papers II and III, respectively. However, there are still many microphysical processes that we will not consider (e.g., heat conduction). Incorporating these processes is beyond the scope of this work; in this paper we will only be concerned with solving the equations of HD as given above.

3. THE ZEUS CODE

The equations of HD are a coupled set of hyperbolic, partial differential equations (PDEs). Without making restrictive assumptions, it is generally very difficult to find analytic solutions to these equations, and we must turn to numerical methods. Numerical algorithms for solving hyperbolic PDEs are well developed and have been implemented in an abundance of computer codes. One such code, designed for solving fluid dynamical problems in astrophysics, is ZEUS. The ZEUS code began as the thesis project of M.

Norman to study the collapse and evolution of rotating interstellar gas clouds during star formation (Norman 1980; Norman, Wilson, and Barton 1980). Over the years, the algorithms were improved and extended (Norman & Winkler 1986), and the code was used extensively for modeling jets in radio galaxies (Norman et al. 1981, 1982). Most recently, David Clarke incorporated a MHD algorithm to model magnetically confined jets in radio galaxies (Clarke 1988; Clarke, Norman, & Burns 1986, 1989).

Several features of the ZEUS code make it particularly well suited for astrophysical fluid dynamics. The code is a time-explicit, two-dimensional Eulerian hydrocode based on the method of finite-differences, characterized by a high degree of simplicity, robustness, and speed. Simple algorithms make for simple coding, thus improvements and changes to the algorithms can be quickly incorporated and tested. With the ability to use different algorithms to solve the same problem, one is led to a deeper understanding of what features of the solution might be strongly affected by the numerics, and what represents physics. In addition, new physics can be incorporated into the code easily. While more complex algorithms can give better results for HD problems (such as a high-order Godunov method as implemented in the PPM code; Colella & Woodward 1984), years of algorithm development are often required to add new physics to them. Finally, the speed of the algorithms in the ZEUS code means that one can afford to solve large problems at high resolution, one of the most important considerations when trying to find the converged numerical solution to a physical problem.

The ZEUS code serves as the foundation of the code described here, but there have been considerable modifications and improvements made during the development of the ZEUS-2D code. For example, we have implemented a more general covariant differencing formalism which allows the use of any orthogonal coordinate system. In addition, we use a more accurate and simpler algorithm for the MHD than was used in the later versions of ZEUS (see Paper II). Code management, control, and portability have also been improved. These additions have led to a completely new code. In this first paper, we describe the implementation and testing of the HD algorithms which serves as the foundation of this new code ZEUS-2D.

4. THE ZEUS-2D CODE

ZEUS-2D solves the fluid equations using the method of finite-differences with a time-explicit, multistep (operator split) solution procedure. An operator split method breaks the solution of the PDEs into parts, with each part representing a single term in the equations. Each part is evaluated successively using the results from the update preceding it. For example, we can write the dynamical equations schematically as

$$\frac{\partial y}{\partial t} = \mathscr{L}(y) . \tag{6}$$

If we assume the operator $\mathscr{L}(y)$ can be split into parts, $\mathscr{L}(y) = \mathscr{L}_1(y) + \mathscr{L}_2(y) + \cdots$, then the operator split solution procedure is

$$(y^1 - y^0)/\Delta t = L_1(y^0) , \tag{7}$$

$$(y^2 - y^1)/\Delta t = L_2(y^1) , \tag{8}$$

$$(y^3 - y^2)/\Delta t = \cdots , \tag{9}$$

where the L_i are finite-difference representations of the corresponding operators $\mathscr{L}_i$. Of course, an operator split method is only a simplified approximation to the correct solution of the full nonlinear, multidimensional operator $\mathscr{L}$. Furthermore, there is considerable freedom in the choice of the order of the individual steps. However, numerical experiments have shown that such a multistep solution procedure is more accurate than a single integration step based on old data, and that there is little difference between solutions generated using different orderings (Hawley, Smarr, & Wilson 1984b; Norman & Winkler 1986).

In ZEUS-2D the individual parts in the solution procedure are grouped into two steps, called the *source* and the *transport* steps. In the source step, we solve finite-difference approximations to

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = -\nabla p - \rho \nabla \Phi - \nabla \cdot \mathbf{Q} , \tag{10}$$

$$\frac{\partial e}{\partial t} = -p \nabla \cdot \boldsymbol{v} - \mathbf{Q} : \nabla \boldsymbol{v} , \tag{11}$$

while in the transport step we account for fluid advection by solving finite-difference approximations to the integral equations

$$\frac{d}{dt} \int_V \rho dV = -\int_{dV} \rho (\boldsymbol{v} - \boldsymbol{v}_g) \cdot d\boldsymbol{S} , \tag{12}$$

$$\frac{d}{dt} \int_V \rho \boldsymbol{v} dV = -\int_{dV} \rho \boldsymbol{v} (\boldsymbol{v} - \boldsymbol{v}_g) \cdot d\boldsymbol{S} , \tag{13}$$

$$\frac{d}{dt}\int_V edV = -\int_{dV} e(\boldsymbol{v}-\boldsymbol{v}_g)\cdot d\boldsymbol{S}\,, \tag{14}$$

where $\boldsymbol{v}_g$ is the grid velocity which allows for a moving grid. The origin of this form of the transport equations is discussed below.

Note that in the source step, we have added terms which do not appear in Euler's equations to account for viscous stresses and dissipation due to an artificial viscosity $\mathbf{Q}$. Much like kinematic viscosity in a real fluid, the artificial viscosity acts to smooth discontinuities which may appear in the flow, where the finite-difference equations break down. Although dissipative processes in real fluids smooth shocks to be only a few particle mean free paths wide, the magnitude of the artificial viscosity is chosen so large that it smears shocks across several zone widths. To minimize the effect in smooth parts of the flow, one normally chooses a nonlinear artificial viscosity that is large in shocks but negligibly small elsewhere. With artificial viscosity, the difference equations will give the correct Rankine-Hugoniot relations across a shock, as well as the correct shock velocity (von Neumann & Richtmyer 1950). However, since the shock will be unphysically (several zones) wide, one cannot study the details of shock structure with this method, unless one uses an adaptive mesh which allows for very small zones.

The integral equations solved in the transport step (eqs. [12]–[14]) can be derived from use of the continuity equation (1) and the divergence theorem. We choose to solve integral forms for the equations so that a conservative differencing scheme may be used, that is, one that preserves the total quantity of the advected variable on the grid to round off error, a very desirable property. Note that we have measured the divergence of each advected variable on a nonstationary grid moving with velocity $\boldsymbol{v}_g$ with respect to the fixed background. By allowing for a moving grid, we can follow global features of the flow (such as collapse or expansion) providing better resolution. We describe methods by which the grid velocity is chosen in more detail in § 4.1.

In curvilinear coordinates, writing equations (12)–(14) using the (scalar) divergence theorem ignores terms generated by the gradient operator acting on the unit vectors in the components of the velocity. These terms, which appear as "inertial forces," must be accounted for elsewhere; in ZEUS-2D we add them in the source step. In the following subsections, we describe in detail how ZEUS-2D solves the operator split fluid equations (10)–(14), including the finite-difference mesh and variable centering, the covariant differencing formulation, the source and transport steps (including the actual finite-difference equations solved by the code), boundary conditions, stability and accuracy, and finally program control and management.

4.1. *The Finite-Difference Mesh*

The finite-difference method of solving hyperbolic PDEs involves discretizing each dependent variable over the spatial computational domain, and solving finite-difference approximations (algebraic equations) to the differential evolution equations on this discrete mesh. In this work, we shall make the restriction that all dependent variables are functions of only two independent spatial coordinates, so that we need only consider two-dimensional meshes. For Cartesian coordinates, this restriction implies slab symmetry; in cylindrical or spherical coordinates it implies axial symmetry. The reason for this restriction is obvious: it greatly reduces the memory and cpu requirements for a numerical simulation. All of the techniques and algorithms we have used and developed in this work, however, are easily extended to three-dimensional calculations, provided sufficient computer resources are available. Indeed, the ZEUS-3D code has already been developed (Clarke, Stone, & Norman 1990).

While we assume only two spatial dimensions, we make no restriction on the components of the vector or tensor variables that are evolved (such as velocity, magnetic field, or radiation pressure). Thus we evolve all components of vector or tensor variables as functions of two spatial coordinates. This formulation has come to be called 2.5-dimensional dynamics, and it is essential for self-consistent simulations of rotating flows, and for the proper treatment of all possible wave modes in MHD (e.g., see Paper II).

With the covariant differencing scheme used in the work (described in the next section), we label the two independent coordinate directions as x_1 and x_2. Figure 1 diagrams how the two-dimensional $x_1 - x_2$ domain is discretized into zones. In the x_1 direction, zones are labeled by the index i, running from ii (for "i-inner") to io (for "i-outer"). In the x_2 direction, zones are labeled by the index j, running from ji to jo. Finite differencing the evolution equations near the boundaries of the grid requires values for the dependent variables to be specified beyond the computational domain. Thus, at each boundary, two rows of "ghost" zones are added. Values for the dependent variables in the ghost zones are specified using boundary conditions appropriate to the geometry and physics of the problem being solved (these boundary conditions are described in § 4.5). Thus, evolution equations are not solved for the ghost zones. Two ghost zones are needed to specify the derivative of the variables, both for inflow boundary conditions and for the higher order interpolation methods used in the transport step.

Spatial positions on the computational grid are defined using coordinate vectors. The positions of the zone boundaries in each direction are specified by the "a-grid," thus $x1a_i$ is the left boundary of the ith zone, while $x2a_j$ is the lower boundary of the jth zone. Zone centers are specified by the "b-grid," thus $(x1b_i, x2b_j)$ is the coordinate position of the center of the (i, j)-zone. Distances between zones can be specified independently, so that $dx1a_i = x1a_{i+1} - x1a_i$ and $dx1b_i = x1b_i - x1b_{i-1}$, and similarly for the $dx2a_j$ and $dx2b_j$ (see Fig. 2). Independent grid spacing allows for nonuniform grids and more efficient zoning of problems. However, since our formulation requires the grid to be orthogonal, a particular grid spacing in one direction (say, in x_1) applies over the entire domain of the direction orthogonal to it (i.e., in x_2).

Discrete values for each dependent variable are stored for each zone. ZEUS-2D uses a staggered mesh so that scalars, and the individual components of vectors and tensors, are centered at different locations on the mesh. In general, scalars and the diagonal components of tensors of even rank are stored at zone centers, while the components of tensors of odd rank are stored at the appropriate zone interfaces (e.g., the 1-component of vectors are stored at zone interfaces in the 1-direction, and the 2-component

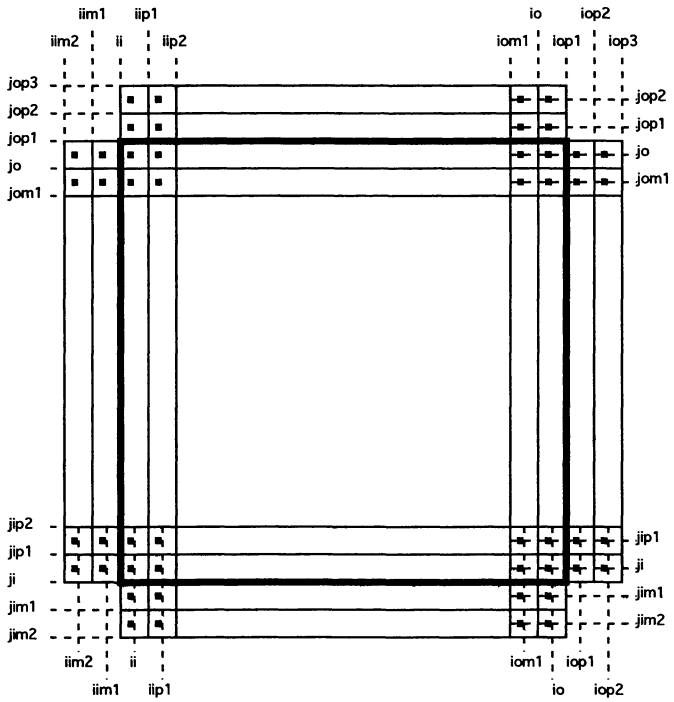

FIG. 1.—Discretization of the two-dimensional computational domain in the ZEUS-2D code. Zone centers, denoted by dots, are located on the "b-mesh," which is labeled by the indices on the bottom and right-hand side. Zone edges define the "a-mesh," which is labeled by the indices on the top and left-hand side. The thick line denotes the boundary of the computational domain, located on the a-mesh at $i = ii$ and $iop1$, and $j = ji$ and $jop1$. Two rows of ghost zones are added beyond each boundary.

of vectors are stored at zone interfaces in the 2-direction). The off-diagonal components of tensors of even rank are stored at the appropriate zone corners. Thus, the transformation from the continuous HD variables to the discrete quantities used in the numerics is (see Fig. 3)

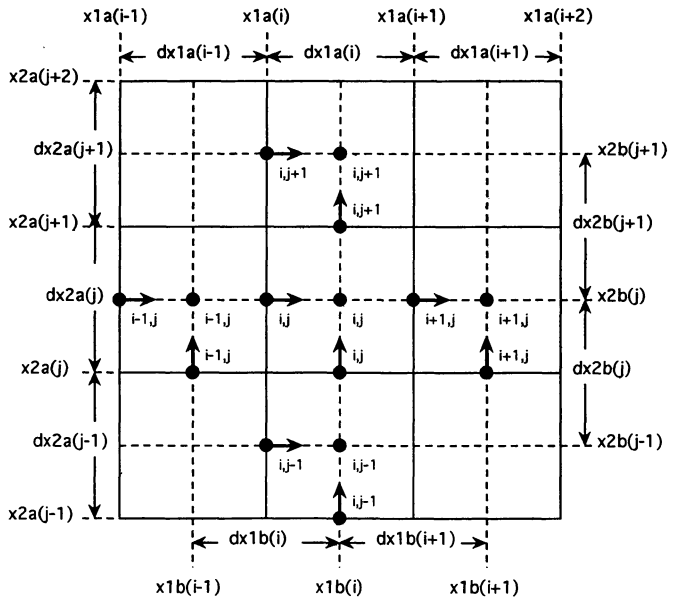

FIG. 2.—Spatial positions of an arbitrary point on the computational domain in the ZEUS-2D code are denoted by the coordinate vectors $x1a(i)$ or $x1b(i)$, and $x2a(j)$ or $x2b(j)$. The a-mesh values are located at zone edges; the b-mesh values are located at zone centers. Grid spacing in each direction is arbitrary and independent. The dots denote the centering and indexing of zone-centered (scalar) and face-centered (vector) variables on the mesh.

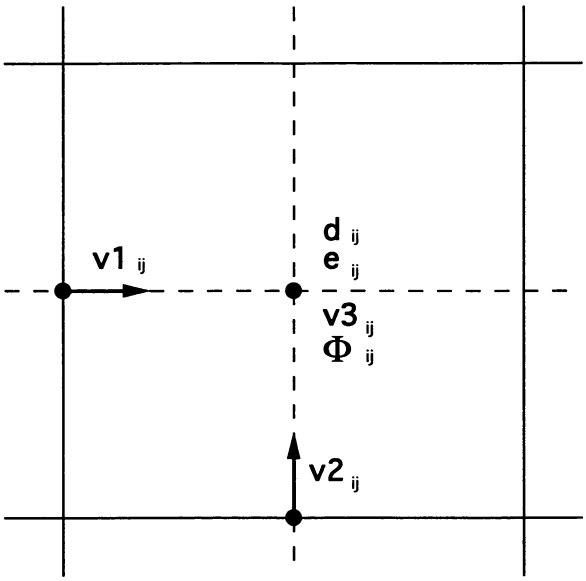

FIG. 3.—Centering of the primary hydrodynamical variables in the ZEUS-2D code. The density, internal energy density, rotational velocity, and gravitational potential are all zone centered, while the 1- and 2-components of the velocity are face centered.

$$\rho(x_1, x_2) \rightarrow d(x1b_i, x2b_j) = d_{i,j} ,$$
$$e(x_1, x_2) \rightarrow e(x1b_i, x2b_j) = e_{i,j} ,$$
$$v_1(x_1, x_2) \rightarrow v1(x1a_i, x2b_j) = v1_{i,j} ,$$
$$v_2(x_1, x_2) \rightarrow v2(x1b_i, x2a_j) = v2_{i,j} ,$$
$$v_3(x_1, x_2) \rightarrow v3(x1b_i, x2b_j) = v3_{i,j} ,$$
$$\Phi(x_1, x_2) \rightarrow \Phi(x1b_i, x2b_j) = \Phi_{i,j} .$$

Note that we have given scalars and vectors the same indices, rather than using the "half-index" notation (e.g., $v1_{i+1/2,j}$), despite the fact that they are centered at different locations on the grid. This should not cause confusion as long as one recalls the centering of the variables, and that the zone interface with the same index i or j as the corresponding zone center is always at the smaller coordinate position (i.e., $x1a_i < x1b_i$ and $x2a_j < x2b_j$; see Fig. 2). Furthermore, using the standard multidimensional implementation of the artificial viscosity (see § 4.3), no rank-two tensor variables are needed for the HD. We will encounter rank-two tensors, however, in the RHD (see Paper III), and for the implementation of a tensor artificial viscosity (Appendix B). Active zones (zones where the evolution equations are used to update the variables) for zone-centered quantities ($d_{i,j}$, $e_{i,j}$, $v3_{i,j}$, and $\Phi_{i,j}$) run from $i = ii$, io and $j = ji, jo$. Active zones for face-centered quantities in the 1-direction ($v1_{i,j}$) run from $i = ii + 1$, io and $j = ji, jo$, while active zones for face-centered quantities in the 2-direction ($v2_{i,j}$) run from $i = ii$, io and $j = ji + 1$, jo.

There are two advantages to using a staggered mesh. The first is that most spatial differences are centered, for example, vectors which are formed from differencing scalars are in a centered location between these scalars. Centered differences are formally second-order accurate, as opposed to the first-order accuracy of forward or backward differences. Second, a staggered mesh reduces the number of interpolations needed for solving the advection equations (12)–(14) in the transport step. Thus, the velocities, when centered on zone interfaces, naturally describe the flux of fluid into or out of a zone. However, staggered meshes often require additional averaging of variables in other parts of the calculation (for instance, forming the momentum from the mass density and velocities), which can ultimately limit the overall accuracy (and convergence rate) of the code (e.g., see Paper II). Here, by convergence rate, we mean the rate at which the error in the numerical solution decreases as the number of grid points is increased. Furthermore, a staggered mesh can increase the difficulty of applying boundary conditions for some variables. In this work, however, we have found the advantages of a staggered mesh outweigh the disadvantages.

For some problems, it is advantageous to allow for globally nonrectangular meshes. For instance, for a simulation of an accretion disk surrounding a central gravitating object in cylindrical coordinates, one would like to be able to exclude the origin $r = z = 0$ from the domain to avoid potential numerical difficulties at the surface layer of the central object. This could be achieved with a series of steps in the boundary of the domain near the origin. This capability is incorporated in ZEUS-2D by allowing the inner and outer index of each coordinate label to be arrays of the other coordinate label. Thus, $ii \rightarrow ii(j)$ and $io \rightarrow io(j)$, where each jth value can be specified independently, and similarly $ji \rightarrow ji(i)$ and $jo \rightarrow jo(i)$. The minimum and maximum indices over the whole grid for each coordinate label are stored in the scalars is, ie and js, je.

Finally, as mentioned in § 4, the coordinate mesh can be moved with respect to a stationary background using grid velocities. Grid velocity vectors are defined at zone interfaces identical to the positions of the fluid velocity vectors. After each time step, the grid velocities are used to recompute the positions of the grid boundaries (a-mesh) and the positions of zone centers (b-mesh). Grid velocities can be prescribed in any arbitrary manner, and grid movement in the two directions is independent. For instance, to make the code semi-Lagrangean, we can equate the grid velocity to the fluid velocity along a particular strip of zones. Or, if we are studying a collapse problem, we might make the grid collapse in some arbitrary manner that approximates the actual dynamical collapse of the flow. These, and several other options, are implemented in ZEUS-2D as choices for the calculation of the grid velocities. Note, however, that we have defined the grid and coordinate system in ZEUS-2D to be orthogonal. Grid motion must preserve that orthogonality, thus along each slice in x_2 at a given x_1, the grid velocity vectors in the 1-direction must be identical, and vice versa for the grid velocities in the 2-direction. This scheme is not as general as a truly two-dimensional adaptive mesh algorithm as developed by R. Fiedler (1990, private communication) or Berger & Oliger (1984), and it limits somewhat our ability to follow accurately two-dimensional flows with the moving mesh. There are many examples of the power of fully adaptive meshes in HD calculations; thus implementing a more sophisticated algorithm in ZEUS-2D would be a very profitable task.

4.2. *The Covariant Formalism*

Having defined the spatial mesh on which the equations of HD will be solved, and discretizing the dependent variables on this mesh, we could now write down the finite-difference approximations to the differential equations that will ultimately be solved by ZEUS-2D in the source and transport steps. Normally, one would proceed by choosing a specific coordinate system in which to write the differential equations, and then finite-differencing these *coordinate-dependent* equations. Of course, the resulting algorithm can then be used to solve problems only in that single coordinate system. Modifying the difference equations (and thus the numerical code) to solve problems in another coordinate system is usually a lengthy and tedious task. In this work, however, we have developed a covariant formalism for differencing the dynamical equations. By using the scale factors of the diagonal metric tensor which describes the particular orthogonal coordinate system in which one would like to solve the equations, we can write all of the vector and tensor operators which appear in the equations in a *coordinate-independent* fashion. Covariant expressions for all vector and tensor operators used in this work are given in Appendix A. We can finite-difference these covariant equations, including the scale factors, in the usual fashion. The resulting difference equations are applicable to any orthogonal coordinate system. The only coordinate-dependent step is the computation of the scale factors, which can be done once and for all at the start of the calculation (provided the mesh is stationary).

In this work, we are primarily interested in three coordinate systems, Cartesian (x, y) coordinates for which

$$(x_1, x_2, x_3) = (x, y, z), \qquad (h_1, h_2, h_3) = (1, 1, 1), \tag{15}$$

cylindrical (RZ) coordinates for which

$$(x_1, x_2, x_3) = (z, r, \phi), \qquad (h_1, h_2, h_3) = (1, 1, r), \tag{16}$$

and spherical polar (RT) coordinates for which

$$(x_1, x_2, x_3) = (r, \theta, \phi), \qquad (h_1, h_2, h_3) = (1, r, r \sin\theta), \tag{17}$$

where the h_i are the metric scale factors. In these cases, x_3 is always the ignorable coordinate for two-dimensional calculations, and the scale factors are simple functions of the coordinates, namely

$$h_1 = 1 \equiv g_1, \tag{18}$$

$$h_2 = f(x_1) \equiv g_2, \tag{19}$$

$$h_3 = f(x_1) f(x_2) \equiv g_{31} g_{32}, \tag{20}$$

where we have used the g_1, g_2, g_{31}, and g_{32} to denote the actual scale factor variables used in the ZEUS-2D code (and which appear in the finite difference equations below). Without loss of generality, we can drop g_1 from the difference equations. Since the remaining scale factor variables g_2, g_{31}, and g_{32} are themselves functions of the coordinates, they must be centered on the mesh appropriately. For each scale factor, we define an a-mesh and a b-mesh value, and center these quantities as shown in Figure 4. For some of the covariant equations, the derivatives of the scale factors are needed. From the scale factors for the three coordinate systems used in this work (eqs. [15]–[17]), the derivatives of the scale factor variables actually used in the code (eqs. [19]–[20]) are, for Cartesian geometry,

$$\frac{\partial g_2}{\partial x_1} = 0, \qquad \frac{\partial g_{31}}{\partial x_1} = 0, \qquad \frac{\partial g_{32}}{\partial x_2} = 0, \tag{21}$$

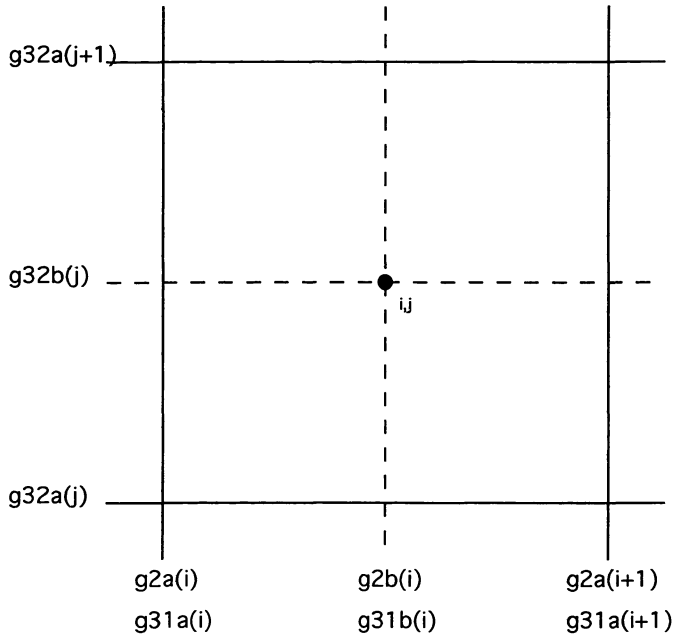

FIG. 4.—Centering of the metric scale factor variables $g2$, $g31$, and $g32$ used in the finite difference equations with the covariant formalism. A zone centered on the arbitrary point $x1b(i)$, $x2b(j)$ is shown. The a-mesh is denoted by solid lines, the b-mesh by dashed lines.

for cylindrical geometry,

$$\frac{\partial g_2}{\partial x_1} = 0, \qquad \frac{\partial g_{31}}{\partial x_1} = 0, \qquad \frac{\partial g_{32}}{\partial x_2} = 1\ , \tag{22}$$

and for spherical polar geometry,

$$\frac{\partial g_2}{\partial x_1} = 1, \qquad \frac{\partial g_{31}}{\partial x_1} = 1, \qquad \frac{\partial g_{32}}{\partial x_2} = \cos\theta\ . \tag{23}$$

Values for these derivatives are also defined on both the a-mesh and the b-mesh and centered appropriately.

By writing the scale factors in the form of equations (18)–(23) we can reduce the memory requirements and floating point operations of the covariant formalism. However, we have also imposed restrictions on the coordinate systems that can be used in ZEUS-2D. For example, the scale factors for an oblate spheroidal coordinate system cannot be written in the form of equations (18)–(20) since $h_1 \neq 1$ in this case, thus we cannot compute using these coordinates (which might be particularly suited to simulations of a rotationally flattened star). However, extending our implementation to increase its generality is straightforward.

By introducing general curvilinear coordinates, we have also introduced the possibility of numerical errors when differencing near coordinate singularities. The occurrence of this error can easily be demonstrated in spherical polar coordinates. For radial derivatives, if we use the simple differencing,

$$\frac{1}{r^2}\frac{\partial}{\partial r}(r^2F) \approx \frac{\Delta(r^2F)}{r^2\Delta r} = \frac{r_i^2F_i - r_{i-1}^2F_{i-1}}{[(r_i + r_{i-1})/2]^2(r_i - r_{i-1})}\ , \tag{24}$$

then near the origin we do not recover the correct difference formula resulting from the volume difference,

$$\frac{1}{r^2}\frac{\partial}{\partial r}(r^2F) \approx \frac{\Delta(r^2F)}{\Delta(r^3/3)} = \frac{r_i^2F_i - r_{i-1}^2F_{i-1}}{r_i^3/3 - r_{i-1}^3/3}\ . \tag{25}$$

Although both these difference approximations are identical at large radii, near the origin ($r_{i-1} \to 0$) there is a serious discrepancy since $[(r_i + r_{i-1})/2]^2[r_i - r_{i-1}] \neq r_i^3/3 - r_{i-1}^3/3$. Similar problems occur from differencing the polar angle derivatives. This is, in fact, an example of a general method of reducing finite difference errors near coordinate singularities that Evans (1986) has termed "coordinate regularization." Mönchmeyer & Müller (1989) have also described a general method of differencing conservation laws on a non-Cartesian mesh. We do not use coordinate regularization in its full generality here, however. In this work, we merely

ensure that we use the volume difference formulae whenever necessary. This is achieved by introducing the volume factor variables, defined for Cartesian geometry as

$$dvl1 = \Delta x, \qquad dvl2 = \Delta y\,, \tag{26}$$

for cylindrical geometry as

$$dvl1 = \Delta z, \qquad dvl2 = \Delta(r^2/2)\,, \tag{27}$$

and for spherical polar geometry as

$$dvl1 = \Delta(r^3/3), \qquad dvl2 = \Delta(-\cos\theta)\,. \tag{28}$$

Like the metric scale factors and their derivatives, the volume factor variables are functions of the coordinates and are therefore defined on both the a- and the b-mesh and centered appropriately. Then, whenever terms like $(h_2h_3)^{-1}(\partial F/\partial x_1)$ or $(h_3)^{-1}(\partial F/\partial x_2)$ appear in the differential equations, they must be differenced using these volume factor variables as

$$\frac{1}{h_2h_3}\frac{\partial F}{\partial x_1} \rightarrow \frac{1}{g_{32}}\frac{\Delta F}{dvl1}, \qquad \frac{1}{h_3}\frac{\partial F}{\partial x_2} \rightarrow \frac{1}{g_{31}}\frac{\Delta F}{dvl2}\,. \tag{29}$$

4.3. *The Source Step*

In the source step, we solve finite-difference approximations to the differential equations (10)–(11). Physically, these equations represent the source and sink terms for each of the dependent variables, which have been operator split from the advection terms added later in the transport step. Following the operator split method, the source step itself is divided into three substeps. In the first, we update the velocities due to pressure gradients, gravitational forces, and coordinate curvature terms (inertial forces); in the second, we use the partially updated velocities to add the artificial viscous stresses and dissipation; and in the third we add the compressional heating term. Let the superscript n denote quantities at the start of the source step (resulting from the last cycle of the calculation), while $n+a$, $n+b$, and $n+c$ represent the partially updated quantities resulting from each of the above three substeps. Then, using the covariant expressions for the spatial operators, the actual difference equations used in ZEUS-2D for each of these substeps are as follows.

4.3.1. *Substep 1*

In this substep we add pressure gradients, gravitational forces, and curvature terms. Thus,

$$\frac{v1^{n+a}_{i,j} - v1^{n}_{i,j}}{\Delta t} = -\frac{p^n_{i,j} - p^n_{i-1,j}}{dx1b_i(d^n_{i,j} + d^n_{i-1,j})/2} - \frac{\Phi^n_{i,j} - \Phi^n_{i-1,j}}{dx1b_i} + \frac{[(v2^n_{i,j} + v2^n_{i,j+1} + v2^n_{i-1,j} + v2^n_{i-1,j+1})/4]^2}{g2a_i}\left(\frac{\partial g2a_i}{\partial x_1}\right) + \frac{[(v3^n_{i,j} + v3^n_{i-1,j})/2]^2}{g31a_i}\left(\frac{\partial g31a_i}{\partial x_1}\right), \tag{30}$$

$$\frac{v2^{n+a}_{i,j} - v2^{n}_{i,j}}{\Delta t} = -\frac{p^n_{i,j} - p^n_{i,j-1}}{g2b_i dx2b_j(d^n_{i,j} + d^n_{i,j-1})/2} - \frac{\Phi^n_{i,j} - \Phi^n_{i,j-1}}{g2b_i dx2b_j} + \frac{[(v3^n_{i,j} + v3^n_{i,j-1})/2]^2}{g2b_i g32a_j}\left(\frac{\partial g32a_j}{\partial x_2}\right). \tag{31}$$

Note that, due to the staggered mesh, averaging of some variables to the appropriate zone interfaces is required in these equations. The pressure $p^n_{i,j}$ is computed at the beginning of the source step using the equation of state. Most commonly, we use an ideal gas law, so that $p^n_{i,j} = (\gamma - 1)e^n_{i,j}$. The gravitational potential $\Phi^n_{i,j}$ is computed at the beginning of the source step from a solution of the Poisson equation (5) using the density $d^n_{i,j}$. In § 4.7 we describe the numerical solution of the Poisson equation as implemented in ZEUS-2D. The curvature terms, which appear only in curvilinear coordinates, are inertial forces due to geometry terms in the divergence operator acting on the momentum flux.

4.3.2. *Substep 2*

In this substep we add the artificial viscous stress and heating terms. The manner in which these terms are differenced depends on the form of the viscosity used. The customary approach is to use the formulation of von Neumann & Richtmyer (1950), which is based on an analysis of planar shocks in one dimension. The method is extended to multidimensions by defining the coefficients of viscosity in each direction independently, which are then used to compute separate scalar artificial pressures q_i for the update in each direction. The success or failure of the von Neumann & Richtmyer approach lies in the proper definition of these scalar viscous pressures. The triumph of the von Neumann & Richtmyer analysis was the realization that a nonlinear viscous pressure, sensitive only to compression, would result in the correct entropy jump across shocks and the correct shock propagation velocity, while

having negligibly small effect away from shocks. Thus, von Neumann & Richtmyer proposed

$$q = \begin{cases} l^2\rho(\partial v/\partial x)^2 & \text{if} \quad (\partial v/\partial x) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{32}$$

where l is a constant with dimensions of length which determines the strength of the artificial viscosity. For a multidimensional implementation, the zone-centered viscous pressure used in the separate updates in each direction is then

$$q1_{i,j} = \begin{cases} C_2 d_{i,j}(v1_{i+1,j} - v1_{i,j})^2 & \text{if} \quad (v1_{i+1,j} - v1_{i,j}) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{33}$$

$$q2_{i,j} = \begin{cases} C_2 d_{i,j}(v2_{i,j+1} - v2_{i,j})^2 & \text{if} \quad (v2_{i,j+1} - v2_{i,j}) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{34}$$

where we have replaced l with the dimensionless constant $C_2 = l/\Delta x$. Physically, C_2 measures the number of zones over which the artificial viscosity will spread a shock and is typically chosen to be $C_2 \approx 3$. Given the scalar pseudoviscous pressures needed for the separate updates in each direction as defined above, the appropriate artificial viscous stress and dissipation terms to be added to the momentum and energy equations are then (see eqs. [10]–[11])

$$\frac{v1_{i,j}^{n+b} - v1_{i,j}^{n+a}}{\Delta t} = -\frac{q1_{i,j} - q1_{i-1,j}}{dx1b_i(d_{i,j}^n + d_{i-1,j}^n)/2}, \tag{35}$$

$$\frac{v2_{i,j}^{n+b} - v2_{i,j}^{n+a}}{\Delta t} = -\frac{q2_{i,j} - q2_{i,j-1}}{g2b_i\, dx2b_j(d_{i,j}^n + d_{i,j-1}^n)/2}, \tag{36}$$

$$\frac{e_{i,j}^{n+b} - e_{i,j}^n}{\Delta t} = -q1_{i,j}\left(\frac{v1_{i+1,j} - v1_{i,j}}{dx1a_i}\right) - q2_{i,j}\left(\frac{v2_{i,j+1} - v2_{i,j}}{g2b_i\, dx2a_j}\right). \tag{37}$$

This extension of the von Neumann and Richtmyer artificial viscosity to multidimensions is ad hoc. A rigorous treatment involves defining an isotropic artificial viscous stress tensor, and then using the components of this tensor in the pseudoviscous terms in the evolution equations. Nonetheless, the formulation described above has proven to be adequate for Cartesian geometry. Its use in curvilinear coordinates, however, can potentially lead to serious numerical difficulties (Tscharnuter & Winkler 1979). For instance, in curvilinear coordinates, radial inflow along converging gridlines represents compression even when $\partial v/\partial r = 0$, and therefore should produce pseudoviscous heating, whereas the von Neumann & Richtmyer approach predicts $q = 0$ in this case. These difficulties can be avoided by using a tensor artificial viscosity (Tscharnuter & Winkler 1979; Winkler & Norman 1986) which implements all of the desirable properties of the von Neumann & Richtmyer method in a covariant formalism. Again, the success of the method lies in the proper definition of the tensor components, the details of which are given in Appendix B. The explicit difference equations for the artificial viscous stress and dissipation terms to be added to the momentum and energy equations are also given in Appendix B.

In practice, we choose to update the velocities and internal energy density using either the von Neumann & Richtmyer form (eqs. [33]–[37]) or the tensor form (Appendix B) of the artificial viscosity. The sole purpose of the nonlinear viscosity is to provide the correct jump conditions and shock velocity. In many problems, numerical experiments have shown that the von Neumann & Richtmyer approach fulfills these goals adequately, even in non-Cartesian geometries. Thus, we have used the tensor formulation only for problems which demand it.

In some test problems, with strong shocks, it is necessary to add linear artificial viscosity to damp oscillations which can occur in stagnant regions of the flow. The scalar linear artificial viscous pressure is defined as (Lapidus 1967; Norman & Winkler 1986)

$$q^{\text{lin}} = C_1\rho C_a \Delta v, \tag{38}$$

where C_1 is a constant of order unity and C_a is the adiabatic speed of sound ($C_a^2 = \gamma p/\rho$). The linear viscosity is sensitive to both compression and expansion, and a different viscous pressure is computed for each direction in multidimensional flows. The explicit difference equations for the linear viscosity update are identical to those for the von Neumann & Richtmyer formulation (eqs. [35]–[37]), except q is replaced by q^{lin} defined above. The update is performed directly after the nonlinear update described above, but before substep 3 below.

4.3.3. *Substep 3*

The source step is completed by adding the compressional heating term. If we are using an ideal gas equation of state, then an

implicit update involving the time-centered pressure $p^{n+1/2}$ can be used to improve energy conservation. Thus,

$$(e^{n+1} - e^n)/(\Delta t) = -p^{n+1/2}\nabla \cdot \boldsymbol{v} , \tag{39}$$

where $p^{n+1/2} = (p^n + p^{n+1})/2$. Using the equation of state, $p = (\gamma - 1)e$, this can be rearranged to give an explicit expression for e^{n+1}, which is then differenced as

$$e_{i,j}^{n+c} = \left[\frac{1 - (\Delta t/2)(\gamma - 1)(\nabla \cdot \boldsymbol{v})_{i,j}}{1 + (\Delta t/2)(\gamma - 1)(\nabla \cdot \boldsymbol{v})_{i,j}}\right] e_{i,j}^{n+b} . \tag{40}$$

Note that $(\nabla \cdot \boldsymbol{v})_{i,j}$ is a zone-centered scalar, an explicit finite-difference expression for which is given in Appendix C. This procedure can be used only for a gamma-law gas. For a more general equation of state, a predictor-corrector method is used,

$$(e_{i,j}^{\text{pred}} - e_{i,j}^{n+b})/(\Delta t) = -p_{i,j}^n(\nabla \cdot \boldsymbol{v})_{i,j} , \tag{41}$$

$$(e_{i,j}^{n+c} - e_{i,j}^{n+b})/(\Delta t) = -\tfrac{1}{2}(p_{i,j}^n + p_{i,j}^{\text{pred}})(\nabla \cdot \boldsymbol{v})_{i,j} , \tag{42}$$

where p^{pred} is computed from the equation of state using e^{pred}. Once the compressional heating term has been added, the source step is completed, and the partially updated variables are ready to be passed to the next step of the calculation, the transport step.

4.4. *The Transport Step*

In the transport step, we solve finite-difference approximations to the integral advection equations (12)–(14). These equations state that the rate of change of the quantity of a variable q within the volume of a zone centered on q (called the control volume) is equal to the divergence of the flux of q through the control volume surfaces. A time finite-difference approximation for these equations which is second-order-accurate in space and time can be written schematically as

$$(q_{i,j}^{n+1}\tau_{i,j}^{n+1} - q_{i,j}^n\tau_{i,j}^n)/(\Delta t) = -(\mathscr{F}_{i+1,j}^{1} - \mathscr{F}_{i,j}^{1} + \mathscr{F}_{i,j+1}^{2} - \mathscr{F}_{i,j}^{2})^{n+1/2} , \tag{43}$$

where τ^n is the control volume at time level n (note $\tau^{n+1} = \tau^n$ if the grid is not moving). The flux in the kth direction is

$$(\mathscr{F}_{i,j}^{k})^{n+1/2} = \langle (vk_{i,j} - vgk_i)q_{k,i,j}^{*}\tilde{A}_{k,i,j}\rangle , \tag{44}$$

where $(vk_{i,j} - vgk_i)$ is the fluid velocity relative to the grid, $q_{k,i,j}^{*}$ is the value of q interpolated to the interface of the control volume, and $\tilde{A}_{k,i,j}$ is the area of the control volume interface, all in the kth direction. The angle brackets denote that all quantities on the right-hand side are time centered. Note if q is zone centered, then due to the staggered mesh, the velocities are naturally centered at the faces of the control volume (a primary reason for introducing a staggered mesh in the first place), otherwise the velocity must be averaged to the faces. Exact conservation of the total quantity of the variable q (at least to machine round off) can be achieved by computing the fluxes for every interface on the grid at once, and then using the same flux to update adjacent zones. Thus, the flux in the 1-direction entering the zone (i, j) is the same as that leaving the zone $(i - 1, j)$. Integrating over the entire computational domain by summing equation (43) over all zones causes the flux to cancel in pairs; thus, except for the flux across the domain boundaries, the total quantity of q will be conserved.

For multidimensional flows, the advection problem is usually simplified by using directional splitting, which involves using a series of one-dimensional advection steps to build up the full solution (Strang 1968). For instance, to solve the two-dimensional problem represented by equation (43), we use two one-dimensional updates, starting with a sweep in the 1-direction,

$$\frac{q_{i,j}^{n+a}\tau_{i,j}^{n+a} - q_{i,j}^n\tau_{i,j}^n}{\Delta t} = -[\langle (v1_{i+1,j} - vg1_{i+1})q_{1,i+1,j}^{*}\tilde{A}_{1,i+1,j}\rangle - \langle (v1_{i,j} - vg1_i)q_{1,i,j}^{*}\tilde{A}_{1,i,j}\rangle] , \tag{45}$$

followed by a sweep in the 2-direction,

$$\frac{q_{i,j}^{n+1}\tau_{i,j}^{n+1} - q_{i,j}^{n+a}\tau_{i,j}^{n+a}}{\Delta t} = -[\langle (v2_{i,j+1} - vg2_{j+1})q_{2,i,j+1}^{*}\tilde{A}_{2,i,j+1}\rangle - \langle (v2_{i,j} - vg2_j)q_{2,i,j}^{*}\tilde{A}_{2,i,j}\rangle] . \tag{46}$$

The fluxes in the second step are computed using the partially updated quantities from the first sweep (e.g., q^{n+a}). In principle, to maintain second-order accuracy for the advection, the directional splitting must be applied in a symmetric fashion. For instance, if X and Y represent the advection operator applied in two orthogonal directions, then the simplest symmetric directionally split update is $(\frac{1}{2}X)(Y)(\frac{1}{2}X)$. Such a scheme increases the computing time by 50% over the simpler, unsymmetrical XY update. However, numerical experiments (Finn & Hawley 1989) have shown that for moderate resolutions ($\leq 10^5$ zones in two dimensions), the small increase in accuracy is not warranted by the large increase in CPU time. For a fixed amount of CPU time, a more

accurate solution is achieved by using greater resolution with an unsymmetrical splitting. In ZEUS-2D we have therefore used an unsymmetrical splitting. We do, however, alternate the order of the update between successive cycles (e.g., we use XY at Δt^n and YX at Δt^{n+1}), which may help to increase the accuracy of the scheme without adding any extra computational overhead.

One of the most challenging aspects of Eulerian HD has been the development of stable and increasingly more accurate algorithms for computing the time-averaged, interpolated value of the variable q at the faces of the control volume, to be used in the computation of the fluxes. By introducing the concept of upwindedness, Godunov (1959) was the first to achieve a stable algorithm for the advection of discontinuities on an Eulerian grid. Upwindedness dictates that the interface value chosen should be that given by the distribution of the variable upstream of the interface. Subsequent work has concentrated on increasing the accuracy of the method by introducing more accurate representations of the distribution of a variable within the upstream zones, using increasingly higher order interpolation polynomials. An essential ingredient of this work has been the realization that a further requirement for stability of a higher order method is that of monotonicity. Physically, this requires that a distribution of $q(x, t)$ which locally increases or decreases monotonically before advection maintains that quality after advection. A monotonic algorithm will not introduce new local extrema into the flow, the essential requirement for stability. Hawley et al. (1984b) and Clarke (1988) have given excellent discussions of the effect of monotonic schemes as implemented in ZEUS-2D in suppressing instabilities.

In ZEUS-2D, we have implemented three schemes for interpolation, the first-order-accurate donor cell method, the second-order-accurate van Leer method (van Leer 1977), and the third-order-accurate piecewise parabolic advection (PPA) method developed by Colella & Woodward (1984). The order of accuracy of each scheme refers to the order of the first term dropped in the Taylor series expansion used to generate the method. Below we describe each of these methods as applied to the calculation of the face-centered upwinded values q_i^* of a zone-centered scalar q_i.

1. *Donor cell (first-order) method.*—The simplest interpolation scheme is to assume that the variable q is constant with a zone, leading to the upwinded values

$$q_i^* = \begin{cases} q_{i-1} & \text{if} \quad v_i - vg_i > 0 \\ q_i & \text{if} \quad v_i - vg_i < 0 \end{cases}. \tag{47}$$

By ignoring any variation of q within a zone, the donor cell method is quite diffusive. Features advected with this method become smeared out quite rapidly; thus, in practice, this method is never used in ZEUS-2D, except for tests.

2. *van Leer (second-order) method.*—We can improve the accuracy of the scheme by improving the order of the interpolation used. The van Leer (1977) scheme uses a piecewise linear function to represent the distribution of q within a zone; thus the upwinded interpolated values are

$$q_i^* = \begin{cases} q_{i-1} + (\Delta x_{i-1} - (v_i - vg_i)\Delta t)(dq_{i-1}/2) & \text{if} \quad v_i - vg_i > 0 \\ q_i - (\Delta x_i + (v_i - vg_i)\Delta t)(dq_i/2) & \text{if} \quad v_i - vg_i < 0 \end{cases}, \tag{48}$$

where the dq_i are the monotonized, van Leer slopes computed from the harmonic average

$$dq_i = \begin{cases} \dfrac{2(\Delta q_{i-1/2}\Delta q_{i+1/2})}{\Delta q_{i-1/2} + \Delta q_{i+1/2}} & \text{if} \quad \Delta q_{i+1/2}\Delta q_{i-1/2} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{49}$$

where $\Delta q_{i+1/2} = (q_{i+1} - q_i)/\Delta x_i$. The definitions of the van Leer slopes in equation (49) takes into account a nonuniform grid spacing. Note that when the van Leer slopes are zero, the method reduces to the donor cell (first-order) method given in equation (47). The van Leer method offers the advantages of improved accuracy (less diffusion) combined with speed. It is therefore the scheme most often used in ZEUS-2D.

3. *PPA (third-order) method.*—The PPA scheme uses parabolic interpolation within a zone to compute upwinded interface values. Schematically, the PPA interface values can be written as

$$q_i^* = \begin{cases} q_{R,i-1} + \xi(q_{i-1} - q_{R,i-1}) + \xi(1 - \xi)(2q_{i-1} - q_{R,i-1} - q_{L,i-1}) & \text{if} \quad v_i - vg_i > 0 \\ q_{L,i} + \xi(q_i - q_{L,i}) + \xi(1 - \xi)(2q_i - q_{R,i} - q_{L,i}) & \text{if} \quad v_i - vg_i < 0 \end{cases}, \tag{50}$$

where $\xi = v_i\Delta t/\Delta x$ = fraction of the width of a zone that the interpolation parabola moves in Δt, and $q_{L,i}$ and $q_{R,i}$ are the monotonized left and right interface values of the ith zone given by the PPA algorithm. The heart of the PPA method lies in the computation of these interface values. In ZEUS-2D we use the method implemented by Clarke (1988) in the ZEUS code, originally described by Colella & Woodward (1984). A detailed description of the method, including the monotonicity constants and contact steepeners, is given in these references and will not be repeated here. However, one key point to be made is that by using parabolae for interpolation, the PPA method requires that two points be specified upwind of the interface. In general, the PPA method uses a five-point molecule centered on the zone being updated. Thus, at boundaries, values must be specified for two ghost zones beyond the computational domain.

For advecting contact discontinuities, PPA is unrivaled in its ability to keep such features sharp, restricting them to a width of only two zones; however, this impressive performance comes at a high cost in CPU time. For many multidimensional *dynamical* problems, it is not clear that this high cost generates a significantly "better" solution than, say, the van Leer scheme run at higher resolution. Furthermore, using a third-order scheme in the advection step does not necessarily mean that the solution generated by the entire code will converge at that rate, since the errors may be dominated by other parts of the solution procedure. For instance, Finn & Hawley (1989) have recently demonstrated that when a spatially varying velocity field is present, the van Leer and PPA algorithms must be modified to account for velocity gradients across a zone in order to maintain their second- and third-order convergence properties. These points will be demonstrated in the results of some of the test problems we have performed. With these points in mind, the interpolation algorithm most often used in ZEUS-2D is the van Leer method, despite the fact that PPA is fully implemented in the code. Note that the piecewise parabolic *advection* (PPA) scheme as implemented in the ZEUS-2D code and to which the above discussion is directed is not equivalent to the piecewise parabolic *method* (PPM), which combines the third-order interpolation described above with a Riemann solver to evolve the HD equations.

Given interpolated and time centered values for a variable using any one of the three schemes described above, we can then proceed to construct time-centered fluxes and perform the advection using finite-difference approximations to the appropriate advection equations. We demonstrate this procedure using the mass density advection equation. With the directionally split algorithm, the update is performed in the 1-direction independently from the update in the 2-direction. The face centered fluxes of mass density in the 1-direction are

$$\mathscr{F}^1_{i,j} = d^*_{1,i,j}(v1_{i,j} - vg1_i)g2a_i^{n+1/2}g31a_i^{n+1/2}dvl2a_j^n \,, \tag{51}$$

where $d^*_{1,i,j}$ is the face-centered, interpolated values for the density, while the $g2a_i^{n+1/2}$ and $g31a_i^{n+1/2}$ are time-centered values for the metric scale factors to account for grid motion. Note that we do not time center grid variables which are functions of the 2-coordinate, since the grid is held fixed in the 2-direction while performing the 1-direction advection. The update for the mass density due to advection in the 1-direction is then

$$(d^{n+1}_{i,j}dvl1a_i^{n+1}dvl2a_j^n - d^n_{i,j}dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{F}^1_{i+1,j} - \mathscr{F}^1_{i,j}) \,, \tag{52}$$

where $dvl1a_i^n$ and $dvl2a_j^n$ are the volume factor variables defined by equations (26)–(28), with $dvl1a_i^{n+1}$ being evaluated at the advanced time to account for grid motion. Once the 1-direction advection is completed for all zones, the update due to advection in the 2-direction is performed using

$$\mathscr{F}^2_{i,j} = d^*_{2,i,j}(v2_{i,j} - vg2_j)g31b_i^n dx1a_i^n g32a_j^{n+1/2} \,, \tag{53}$$

$$(d^{n+1}_{i,j}dvl1a_i^n dvl2a_j^{n+1} - d^n_{i,j}dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{F}^2_{i,j+1} - \mathscr{F}^2_{i,j}) \,, \tag{54}$$

where the $dvl2a_j^{n+1}$ are the volume factor variables evaluated at the advanced time to account for grid motion in the 2-direction. Equations (51)–(54), applied sequentially, represent the two-dimensional advection of mass density on the grid using directional splitting. Note that by completing the update in the 1-direction before starting the update in the 2-direction, the partially updated densities are used in the calculation of the fluxes in the 2-direction (eq. [53]).

Advection of the other HD variables (e, $v1$, $v2$, and $v3$) proceeds in a similar manner. However, there are two important differences in the advection of the other variables as compared to the advection of the mass density. The first is that to improve local conservation for these variables, we use consistent transport, a notion first introduced by Norman et al. (1980). Thus, although the integral equations are globally conservative when differenced using the control volume approach, local conservation of the variables can be seriously affected by the numerical diffusion inherent in the finite-difference equations and interpolation algorithm. Consistent transport attempts to minimize the effect of this diffusion on local conservation by making the fluxes of the fundamental variables consistent with the flux of mass transported through the mesh. As demonstrated by Norman et al. (1980), this can make a dramatic improvement to the local conservation of specific angular momentum in the problem, which can ultimately affect the overall evolution of the system. Consistent transport dictates that the fundamental quantities to be advected are the mass density (ρ), the *specific* internal energy (e/ρ), and the three components of the *specific* momenta defined as $(s_1/\rho) = v_1$, $(s_2/\rho) = h_2 v_2$ and $(s_3/\rho) = h_3 v_3$. Geometric factors are introduced in the definitions of the momenta s_2 and s_3 so that physically these variables represent angular momenta in cylindrical and spherical geometries. By using momenta as the fundamental variable in the transport step, rather that velocity as in the source step, we must transform between these variables at the start and end of the transport step. Since the velocities and density are located at different positions on the staggered mesh (e.g., Fig. 3), arithmetic averaging of the density is required to transform from velocity to momenta, and vice versa. Velocities are held fixed throughout the entire transport step, until they are updated using the newly advected densities and momenta at the conclusion of this step.

The fluxes of these variables are made consistent with the flux of mass density by the use of face-centered mass flux variables defined via

$$\dot{M}^1_{i,j} = d^*_{1,i,j}(v1_{i,j} - vg1_i) \,, \tag{55}$$

$$\dot{M}^2_{i,j} = d^*_{2,i,j}(v2_{i,j} - vg2_j) \,. \tag{56}$$

Apart from the area factors of the zone faces, these formulae are identical to the density fluxes used in the updates of the density (eqs. [51] and [53]). With the mass flux variables defined above, fluxes of the other variables are then constructed, first in the 1-direction using,

$$\mathscr{F}^1_{i,j} = (e/d)^*_{1,i,j}\dot{M}^1_{i,j} g2a_i^{n+1/2} g31a_i^{n+1/2} dvl2a_j^n \,, \tag{57}$$

$$\mathscr{G}^1_{i,j} = v1^*_{1,i,j}\tfrac{1}{2}(\dot{M}^1_{i,j} + \dot{M}^1_{i+1,j}) g2b_i^{n+1/2} g31b_i^{n+1/2} dvl2a_j^n \,, \tag{58}$$

$$\mathscr{H}^1_{i,j} = v2^*_{1,i,j} g2a_i \tfrac{1}{2}(\dot{M}^1_{i,j} + \dot{M}^1_{i,j-1}) g2a_i^{n+1/2} g31a_i^{n+1/2} dvl2b_j^n \,, \tag{59}$$

$$\mathscr{I}^1_{i,j} = v3^*_{1,i,j} g31a_i\, g32b_j \dot{M}^1_{i,j} g2a_i^{n+1/2} g31a_i^{n+1/2} dvl2a_j^n \,, \tag{60}$$

which are used in the update of these variables due to advection in the 1-direction,

$$(e^{n+1}_{i,j} dvl1a_i^{n+1} dvl2a_j^n - e^n_{i,j} dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{F}^1_{i+1,j} - \mathscr{F}^1_{i,j}) \,, \tag{61}$$

$$(s1^{n+1}_{i,j} dvl1b_i^{n+1} dvl2a_j^n - s1^n_{i,j} dvl1b_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{G}^1_{i,j} - \mathscr{G}^1_{i-1,j}) \,, \tag{62}$$

$$(s2^{n+1}_{i,j} dvl1a_i^{n+1} dvl2b_j^n - s2^n_{i,j} dvl1a_i^n dvl2b_j^n)/(\Delta t) = -(\mathscr{H}^1_{i+1,j} - \mathscr{H}^1_{i,j}) \,, \tag{63}$$

$$(s3^{n+1}_{i,j} dvl1a_i^{n+1} dvl2a_j^n - s3^n_{i,j} dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{I}^1_{i+1,j} - \mathscr{I}^1_{i,j}) \,. \tag{64}$$

The area and volume factors which appear in the fluxes (eqs. [57]–[60]) and advection update (eqs. [61]–[64]) demonstrate the second distinction in the advection of these variables as compared to the mass density, namely that the staggered mesh alters the centering of the control volumes for some of the variables. While the control volumes for all zone-centered variables (d, e/d, and $s3$) are the same, they are different for 1-direction vectors ($s1$) and 2-direction vectors ($s2$) (see Fig. 5). Thus, the fluxes of d, e/d, and $s3$ are all face centered in both directions, while the fluxes for $s1$ are zone centered in the 1-direction and face centered in the 2-direction, while the fluxes for $s2$ are face centered in the 1-direction and zone centered in the 2-direction.

Once the update of all variables due to advection in the 1-direction is complete, we then perform the advection in the 2-direction by first computing the fluxes in the 2-direction,

$$\mathscr{F}^2_{i,j} = (e/d)^*_{2,i,j}\dot{M}^2_{i,j} g31b_i^n dx1b_i^n g32a_j^{n+1/2} \,, \tag{65}$$

$$\mathscr{G}^2_{i,j} = s1^*_{2,i,j}\tfrac{1}{2}(\dot{M}^2_{i,j} + \dot{M}^2_{i-1,j}) g31a_i^n dx1b_i^n g32a_j^{n+1/2} \,, \tag{66}$$

$$\mathscr{H}^2_{i,j} = s2^*_{2,i,j} g2b_i^n \tfrac{1}{2}(\dot{M}^2_{i,j} + \dot{M}^2_{i,j+1}) g31b_i^n dx1a_i^n g32b_j^{n+1/2} \,, \tag{67}$$

$$\mathscr{I}^2_{i,j} = s3^*_{2,i,j} g31b_i^n g32a_j^n \dot{M}^2_{i,j} g31b_i^n dx1b_i^n g32a_j^{n+1/2} \,, \tag{68}$$

which are used for the update due to advection in the 2-direction,

$$(e^{n+1}_{i,j} dvl1a_i^n dvl2a_j^{n+1} - e^n_{i,j} dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{F}^2_{i,j+1} - \mathscr{F}^2_{i,j}) \,, \tag{69}$$

$$(s1^{n+1}_{i,j} dvl1b_i^n dvl2a_j^{n+1} - s1^n_{i,j} dvl1b_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{G}^2_{i,j+1} - \mathscr{G}^2_{i,j}) \,, \tag{70}$$

$$(s2^{n+1}_{i,j} dvl1a_i^n dvl2b_j^{n+1} - s2^n_{i,j} dvl1a_i^n dvl2b_j^n)/(\Delta t) = -(\mathscr{H}^2_{i,j} - \mathscr{H}^2_{i,j-1}) \,, \tag{71}$$

$$(s3^{n+1}_{i,j} dvl1a_i^n dvl2a_j^{n+1} - s3^n_{i,j} dvl1a_i^n dvl2a_j^n)/(\Delta t) = -(\mathscr{I}^2_{i,j+1} - \mathscr{I}^2_{i,j}) \,. \tag{72}$$

The transport step is now finished. One cycle, representing the integration of the equations of HD by one time step, is complete, and the current values of the dependent variables are ready to be fed back into the start of the source step to begin the next cycle.

4.5. *Boundary Conditions*

Whenever the dependent variables are updated in the source or transport steps, boundary conditions must be applied to keep the values stored in the ghost zones consistent with the active zones. These conditions are simple, explicit equations which give the values of the dependent variables in the ghost zones from the values in the adjacent active zones, without actually solving the evolution equations. The exact form of the boundary conditions applied depends upon the geometry and physics of the problem being solved. In ZEUS-2D, we have implemented several different conditions which can be applied independently to each variable along each different boundary. These conditions are the following:

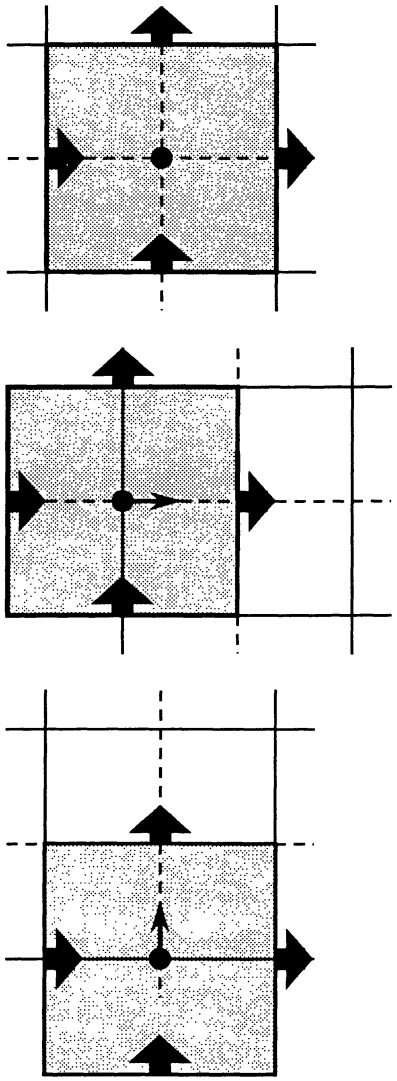

FIG. 5.—The control volumes, denoted by the shaded regions, are different for the zone-centered variables (*top*), face-centered in the 1-direction variables (*middle*), or face-centered in the 2-direction variables (*bottom*). The fluxes are of each type of variable are located on the edges of the appropriate control volumes, as denoted by large arrows. The a- and b-mesh are denoted by solid and dashed lines, respectively.

1. *Reflecting Boundary Condition.*—All zone-centered variables and the tangential components of velocity in the ghost zones are set equal to the corresponding values of their images among the active zones. The normal component of velocity is set to zero on the boundary and reflected for the second ghost zone.

2. *Axis of Symmetry Boundary Condition.*—This condition is identical to the reflecting boundary condition, except that the zone-centered 3-component of velocity is set equal to the negative of the value in the corresponding active zones, forcing it to go to zero on the boundary.

3. *Inflow Boundary Condition.*—Here the values of all the variables in the ghost zones are held equal to a set of predetermined values (which may be allowed to vary in time). Outflow is not permitted. For supersonic inflow, this condition is exact, while for subsonic inflow, some spurious reflection of wave energy may occur (i.e., in principle, the boundary condition should be perfectly transparent to outgoing waves, but in practice, the numerical implementation of the boundary condition always has some nonzero reflection coefficient associated with it).

4. *Outflow Boundary Condition.*—This is the most difficult boundary condition to implement properly. In ZEUS-2D we have used the simplest approach possible and set all values of variables in the ghost zones equal to the values in the corresponding active zones (extrapolated the flow beyond the boundary). For supersonic outflow, this technique is exact. However, for subsonic outflow, spurious reflection of wave energy will occur at the boundary. If the reflected waves affect the dynamics of the simulation, this problem must be addressed, either by using a larger computational domain, or by increasing the accuracy of the boundary condition (e.g., by extrapolating along characteristics rather than gridlines; Thompson 1987).

5. *Periodic Boundary Conditions.*—All zone-centered variables and the tangential component of velocity in the ghost zones are

set equal to the values in the corresponding active zones on the other side of the grid. The normal components of velocity in the second ghost zone is set equal to the value in the appropriate active zone on the opposite side of the grid, while on the boundary (first ghost zone) it is computed from the difference equations.

As implemented in ZEUS-2D, the boundary conditions can be applied independently for every zone on every boundary of the computational domain. This allows for changing the boundary condition along a boundary (e.g., to give a jet nozzle inlet).

In addition, to increase the modularity of the implementation, the boundary conditions for each different variable are implemented in a different subroutine, which can be called from any other part of the code. Thus, modifying or improving the boundary conditions is a straightforward task. Note that the implementation of the boundary conditions described here only applies to the HD variables. The boundary conditions used for the magnetic fields and radiation variables are described in Papers II and III, respectively.

4.6. *Stability and Accuracy*

Upwinding of the advected fluxes as described in the transport step is essential to maintain stability, that is, to prevent the growth of short wavelength (sawtooth) oscillations. In addition, a time-explicit code must also limit the time step used to evolve the dynamical equations to satisfy the Courant-Friedrichs-Lewy (CFL) stability condition. Physically, this condition can be understood as limiting the distance that information can travel in one time step (via waves or fluid motion) to be smaller than one grid zone. We must choose the largest time step possible such that every zone locally fulfills the CFL condition, so that for a one-dimensional calculation,

$$\Delta t \leq \min\,(\Delta x)/(|u| + C_a)\,, \tag{73}$$

where u is the local fluid velocity, C_a is the local adiabatic speed of sound, and the minimum is taken over all grid zones. Although we have stated this condition on physical grounds, it can be derived with mathematical rigor by performing a von Neumann stability analysis of the dynamical equations (Richtmyer & Morton 1957). For multidimensional flows, a suitable time-step limit is the smallest of all one-dimensional CFL conditions in each orthogonal direction. Thus, in ZEUS-2D we choose the explicit time step using

$$\Delta t = C_0/[\max\,(\delta t_1^{-2} + \delta t_2^{-2} + \delta t_3^{-2} + \delta t_4^{-2})]^{1/2}\,. \tag{74}$$

Here, the maximum is taken over all zones, C_0 is a safety factor (called the Courant number, typically $C_0 \approx 0.5$), and the various limiting time steps are defined as

$$\delta t_1 = [\min\,(\Delta x_1, \Delta x_2)]/C_a\,, \tag{75}$$

$$\delta t_2 = \Delta x_1/(v_1 - vg_1), \qquad \delta t_3 = \Delta x_2/(v_2 - vg_2)\,. \tag{76}$$

The δt_4 arises from the inclusion of artificial viscosity in the dynamical equations. This changes the mathematical nature of the momentum equation to a diffusion equation. For stability, explicit diffusion schemes are limited to a time step of

$$\Delta t \leq (\Delta x)^2/4\nu\,, \tag{77}$$

where ν is the coefficient of kinematic viscosity. The equivalent ν for the artificial viscosity can be determined by comparing the artificial viscosity terms to those in the Navier Stokes equations, thus $\nu = l^2\nabla\cdot\boldsymbol{v}$ where l is a constant with dimensions of length which describes the strength of the artificial viscosity. Thus, the fourth time-step limit defined in equation (74) is then

$$\delta t_4 = \min\left(\frac{(\Delta x_1)^2}{4l^2(\Delta v_1/\Delta x_1)}, \frac{(\Delta x_2)^2}{4l^2(\Delta v_2/\Delta x_2)}\right) = \min\left(\frac{\Delta x_1}{4C_2\Delta v_1}, \frac{\Delta x_2}{4C_2\Delta v_2}\right), \tag{78}$$

where we have used the dimensionless coefficient of artificial viscosity C_2 defined through $C_2 = l/\Delta x$ (see § 4.3).

Since the characteristic speeds will in general be changing as the evolution proceeds, the time step must be allowed to vary as well. The time step is limited to an increase of no more than 30% to maintain accuracy, yet may decrease by any amount to maintain stability.

While fully implicit time-differencing schemes are not subject to the CFL stability criterion, they are inefficient for the study of short time-scale dynamical flows; therefore we do not use them here. However, when radiation is included in the dynamical equations, the CFL condition for all possible characteristic speeds can become overly restrictive. We have therefore implemented a fully implicit algorithm for evolving the radiation dynamical variables. In these circumstances, ZEUS-2D operates used a mixed explicit-implicit method. The implementation of the fully implicit RHD algorithm is described in detail in Paper III.

4.7. *The Solution of the Poisson Equation*

In the previous sections, we have described the numerical algorithm implemented in the ZEUS-2D code for solving the HD equations. Since these equations are hyperbolic, we must be concerned with space and time discretization, and such issues as stability and the treatment of discontinuities (shocks). However, since the Poisson equation is elliptic, numerical solutions are much easier to obtain. In this section, we describe our numerical procedure for accomplishing this.

Many numerical methods have been utilized to solve the Poisson equation, as it occurs in many physical situations. For our purposes, we need a solver which will work on a two-dimensional nonuniform mesh with general boundary conditions. Thus, many of the fastest methods developed for specialized problems (e.g., spectral or Fourier transform methods) are ruled out. We will concentrate on solving the sparse matrix which results from finite-differencing the Poisson equation. We begin by discretizing the Poisson equation in two dimensions. The gravitational potential is zone centered; thus using the covariant form of the Laplacian operator given in Appendix A, the finite-difference equation can be written as

$$a_1\Phi_{i+1,j} + a_2\Phi_{i,j+1} + a_3\Phi_{i,j} + a_4\Phi_{i-1,j} + a_5\Phi_{i,j-1} = S_{i,j} , \tag{79}$$

where

$$a_1 = (g2a_{i+1}\,g31a_{i+1}\,dvl2a_j)/dx1b_{i+1}$$

$$a_2 = (g32a_{j+1}\,dvl1a_i)/(g2b_i^2\,dx2b_{j+1}) ,$$

$$a_3 = \frac{dvl1a_i}{g2b_i^2}\left(\frac{g32a_{j+1}}{dx2b_{j+1}} + \frac{g32a_j}{dx2b_j}\right) + dvl2a_j\left(\frac{g2a_{i+1}\,g31a_{i+1}}{dx1b_{i+1}} + \frac{g2a_i\,g31a_i}{dx1b_i}\right) ,$$

$$a_4 = (g2a_i\,g31a_i\,dvl2a_j)/dx1b_i ,$$

$$a_5 = (g32a_j\,dvl1a_i)/(g2b_i^2\,dx2b_j) ,$$

$$S_{i,j} = dvl1a_i\,dvl2a_j 4\pi G d_{i,j} .$$

To avoid discretization errors at the origin in curvilinear coordinates, we have used the volume differencing described in § 4.2. Clearly, the Laplacian results in a five-point molecule on a two-dimensional orthogonal mesh. We cannot obtain a solution for $\Phi_{i,j}$ without simultaneously achieving a solution for all its nearest neighbors. Thus, solving the Poisson equation represents solving N^2 coupled linear equations, where N is the number of zones on a side.

Rewriting equation (79) in matrix form results in a sparse banded matrix whose structure is diagrammed in Figure 6. This sparse banded pattern occurs universally for a two-dimensional operator using a five-point difference molecule; we will encounter matrices similar in structure for the implicit solution of the radiation moment equations in Paper III.

The appropriate boundary conditions for the problem are incorporated directly into the matrix elements themselves or the right-hand side. For each boundary of the domain, we have implemented two possible boundary types, (1) Neumann, in which the slope of the gravitational potential is set to zero, and (2) Dirichlet, in which the value of Φ_B in the ghost zones is specified. Neumann boundary conditions are used at symmetry boundaries (axis or equator), while Dirichlet are applied at outer boundaries, far from most of the mass distribution. Neither the Neumann nor the Dirichlet boundary conditions affect the structure of the matrix shown in Figure 6. Neumann boundary conditions simply modify the diagonal elements of the row in which they are applied, while Dirichlet boundary conditions modify the right-hand side.

For the case of Dirichlet boundaries we compute Φ_B using a multipole expansion formula (Jackson 1975),

$$\Phi_B = G\sum_l P_l(\mu_B) r_B^{-(l+1)} M_l , \tag{80}$$

where P_l are the Legendre polynomials which are functions of the cosine of the angle μ_B between the position vector and axis of symmetry, r is the magnitude of the position vector of the boundary point, and the multipole moments M_l are given by

$$M_l = \int_V \rho(r) r^l P_l(\mu) dV . \tag{81}$$

This method only applies to curvilinear coordinates for which there is an axis of symmetry. The Legendre polynomials can be generated using the recursion relation

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) , \tag{82}$$

where $P_0(x) = 1$, and $P_1(x) = x$.

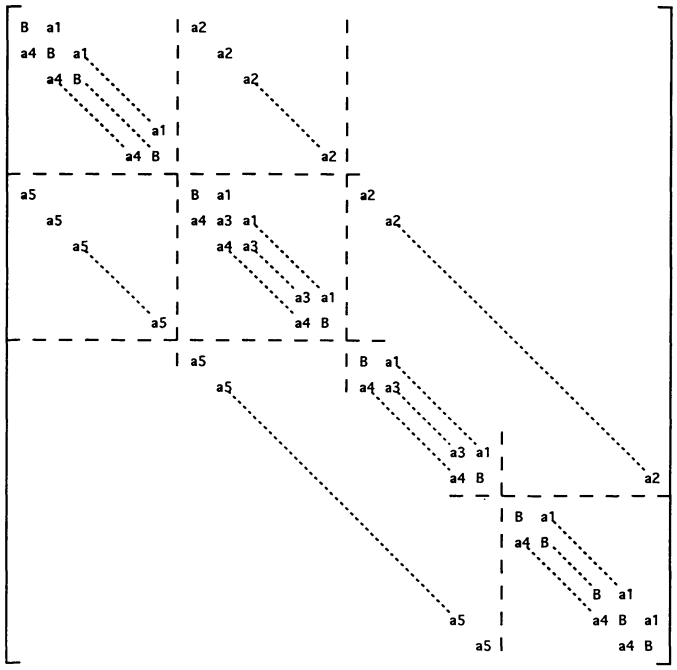

FIG. 6.—Schematic structure of the sparse banded matrix resulting from differencing the Poisson equation in two dimensions. The "a1, . . . , a5" are the coefficients of the dependent variables as defined in eq. (79). A value of B denotes that matrix element can be affected by Neumann boundary conditions.

Equations (80)–(81) are discretized as

$$\Phi_{B,i,j} = G \sum_l P_l(\mu_{B,i,j}) r_{B,i,j}^{-(l+1)} M_l \,, \tag{83}$$

$$\text{where} \qquad M_l = \sum_i \sum_j d_{i,j} r_{i,j}^l P_l(\mu_{i,j}) dvl1a_i \, dvl2a_j \,. \tag{84}$$

In practice, when the problem boundaries are far from most of the mass distribution, then the series in equation (83) converges rapidly, and only a few moments are needed. However, we have found that when the density becomes significant in the zones near the boundary, the series converges slowly, and many moments must be taken to achieve an accurate solution. (In fact, if the mass is not centrally condensed, the series may diverge). As implemented in ZEUS-2D, we continue to add higher moments until Φ_B has converged to one part in 10^3, up to a maximum of 100 terms.

Careful inspection of equation (79) reveals that the sparse banded matrix resulting from discretizing the Poisson equation is positive definite and symmetric. Methods for solving such matrix equations are widely available, and we have used several. First, we have implemented the alternating direction implicit (ADI) method described by Norman & Winkler (1986) and Black & Bodenheimer (1975). With this method, one looks for steady state solutions to the diffusion equation

$$\frac{\partial \Phi}{\partial g} = \nabla^2 \Phi - 4\pi G\rho \,, \tag{85}$$

where g is some iterative time scale which has no relation to the true dynamical time. An iterative method which breaks this multidimensional equation into a series of one-dimensional sweeps is used; these are repeated until convergence is achieved. Thus, the matrix problem is reduced to solving tridiagonal matrices, for which very fast and vectorizable algorithms are well known (Richtmyer & Morton 1957). Recently, adaptive methods (dynamic ADI, or DADI) have been developed (Larson, Hewett, & Anderson 1989) to choose the optimal convergence time-step in the solution of equation (85) which greatly speeds convergence over the simple time-step ordering described in Norman & Winkler (1986) and used here.

We have also used several other methods to solve the matrix, including the Incomplete Cholesky decomposition Conjugate Gradient (ICCG) and General Mean Residual (GMRES) methods (R. Bramley 1989, private communication). We have found that roughly comparable timing is achieved by all of these methods for typical problem sizes of interest (grids of $\sim 10^5$ zones). In practice, however, we nearly always use one of the latter solvers (those other than ADI) for simulations. These methods are also used in the implicit RHD, thus we can share the storage required by the solver between uses. In addition, by basing all of the sparse matrix solvers on library routines, we can easily implement new efficient solvers (such as DADI) as they become available. This can be a

vital point, since when in use, sparse matrix solvers occupy the bulk of the CPU time needed for a simulation. In fact, unless the density distribution is changing rapidly, a solution of the Poisson equation may not be needed every cycle. We keep track of the relative error in the currently stored gravitational potential compared to the current density distribution by rearranging equation (79) so that

$$\Phi'_{i,j} = \frac{1}{a_3}(S_{i,j} - a_1\Phi_{i+1,j} - a_2\Phi_{i,j+1} - a_4\Phi_{i-1,j} - a_5\Phi_{i,j-1}) . \quad (86)$$

We only perform a solution of the Poisson equation when

$$|\Phi'_{i,j} - \Phi_{i,j}|/\Phi_{i,j} > \epsilon , \quad (87)$$

where $\epsilon \sim 10^{-5}$. In practice, this technique can reduce the frequency of solving the Poisson equation to once every tenth cycle, with substantial savings in CPU time.

4.8. *Program Control and Management*

Having described each of the components of the ZEUS-2D code, we can now summarize how these pieces are linked together in the actual code. Figure 7 is a schematic flow chart of the code. After initialization and problem set up, the main loop of the code is entered. Each successive cycle of the main loop representing integration of the PDEs forward in time by one time step includes solution of the Poisson equation to compute the gravitational potential (eq. [79]), the source step (three substep solution of the difference eqs. [30]–[31], [35]–[37], and [40]), the transport step (directionally split solution of the advection equations, represented by the difference eqs. [51]–[72]) followed by calculation of new grid velocities (if the grid is moving), and calculation of a new time step to satisfy the CFL condition. Execution of the main loop is then repeated until one of the several stopping criteria is reached. The main loop also contains data output options, and an *interrupt checker,* a subroutine which provides control of the

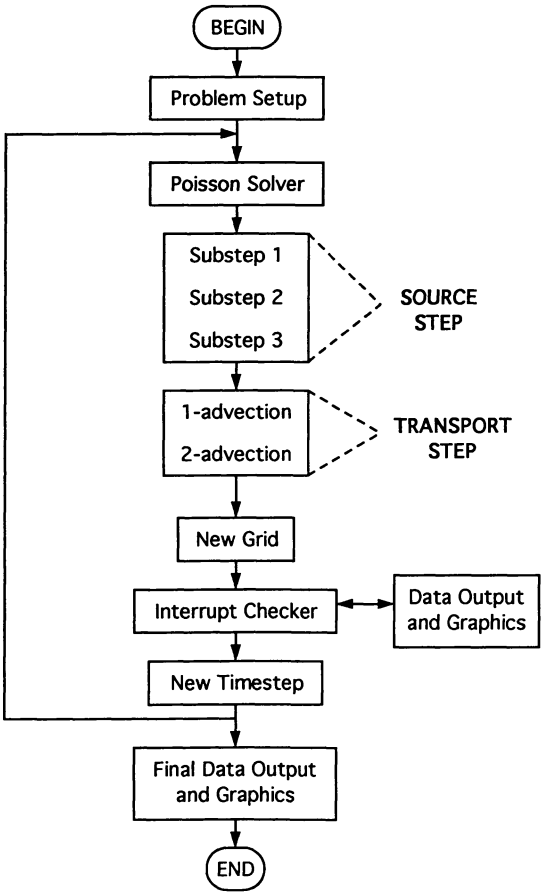

FIG. 7.—A schematic flow chart of the ZEUS-2D code including only the hydrodynamical algorithms. All of the major steps in the solution procedure, as described in the text, are shown.

program during execution. By typing any one of several dozen different commands at any time during execution, one can initiate a data dump, modify the values of control flags and variables, or pause, stop, or restart execution.

The driving philosophy of the code development and implementation has been to make the code as modular and portable as possible. In implementing the actual algorithms in FORTRAN, we have followed closely the paradigm established by Winkler & Norman (1986) for efficient and bug-free coding. We have followed the FORTRAN 77 standard as closely as possible to maintain portability. We have also used graphics software which is widely available; vector graphics are generated using NCAR graphics, raster image files are generated internally for animation by a variety of postprocessing software which runs on local workstations (e.g., Imagetool, Datascope, etc.), while floating point data is dumped in the HDF file format to interface with software developed at NCSA. All of these methods use software in the public domain for which the source code is available, so that ZEUS-2D and any of its dependent graphic library routines can be installed, compiled, and run on any machine which contains a FORTRAN and C compiler.

Whenever a code grows to a size longer that a few thousand lines, then efficient management (e.g., editing and compiling) of the source code becomes a problem (without common block substitution, the full ZEUS-2D code including the MHD and RHD algorithms currently amounts to over 30,000 lines of source). With ZEUS-2D, we use the precompiler CPP (which is part of the UNIX operating system), combined with the UNIX directory structure and utilities, to manage the editing and compilation steps. One useful feature of CPP is its ability to allow one to define various macros which will cause CPP to either include or suppress all portions of the code labeled by that macro during compilation. Thus, one can generate FORTRAN code which is optimized for the problem at hand. For example, by not defining the MHD macro at compilation, the CPP precompiler will produce an executable which is optimized for HD calculation with no reference to MHD either through unused (and wasted) array storage space or unreferenced subroutines. CPP macros are used to control all of the many physics, geometry, and data output modes in ZEUS-2D, greatly simplifying the code management procedure. Indeed, having used CPP, we feel that developing and applying a code as complex as ZEUS-2D would be very difficult without a precompiler.

5. THE HYDRODYNAMIC TEST PROBLEMS

The ZEUS-2D code represents an entirely new implementation of the algorithms found in the ZEUS code. In addition, new capabilities have been added such as a covariant formalism and PPA (third-order) interpolation for all variables. We therefore must run a complete battery of HD tests for this new code. Initially, such tests are designed to catch bugs in the coding. However, with careful and structured coding, such bugs in the implementation should be quickly traced down. Thereafter, tests provide the much more important task of benchmarking the algorithms. By testing the code on many different problems, we can gain an appreciation of the effectiveness, and the limitations, of the algorithms implemented. Such insight is essential to differentiating when a characteristic of a simulation is due to the physics, or is potentially an artifact of the numerics.

In this section, we present the results of the HD tests run on the ZEUS-2D code. Our philosophy of testing had been to isolate and test the numerical solution of individual terms in the difference equations, and slowly build more and more complicated problems until all portions of the code are in use. With this philosophy, we first present results from simple one-dimensional advection problems, next we show results from a one-dimensional shock tube problem, and finally we give results from two-dimensional tests involving strong shocks.

5.1. *Cartesian Advection Tests*

By far the most complicated step in the solution procedures is the transport step, wherein the dependent variables are advected across the grid. We therefore start by testing this portion of the code. The results presented here will only be for the mass density variable ($d_{i,j}$); we have checked that we obtain identical results for the other dependent variables. In addition, all one-dimensional tests have been performed independently in each orthogonal direction.

Advection tests are easily constructed by ignoring the source terms in the difference equations and assuming a time-independent velocity field. Analytic solutions to the differential equations with these restrictions are trivial to obtain. The simplest test we have performed is the one-dimensional advection of a square pulse in Cartesian geometry. Initially, the pulse is 50 zones wide, and centered at $x = 30$. The test involves advecting the pulse for a distance of 5 times its width. Figure 8 shows the results using the four choices of advection algorithms in ZEUS-2D: donor cell, van Leer, PPA, and PPA with the steepener. The results show that the donor cell method is the most diffusive: the pulse has completely lost its shape and no longer has the correct amplitude. The van Leer result is considerably better: the discontinuities in the variable are kept to about 14 zones, and the pulse has the correct amplitude. The PPA result is even better: discontinuities are now only 6 zones wide; while PPA with steepener keeps the discontinuities to only 2 zones wide. This test demonstrates the great range in accuracy of the methods for advecting discontinuities. Not surprisingly, the most sophisticated algorithm (PPA with steepener) obtains the best result for this problem; indeed it would be difficult to improve it. The PPA and van Leer results are good, while the donor cell result is unacceptable for simulations. We note, however, that due to the conservative differencing scheme used in the transport step, the area under the pulse in all cases is identical to the initial value to within one part in 10^{7-8}, regardless of the scheme used.

A more quantitative comparison of the advection algorithms can be made by comparing the convergence rates of the different schemes for a one-dimensional advection test. The convergence rate is computed by comparing the numerical solution at a certain

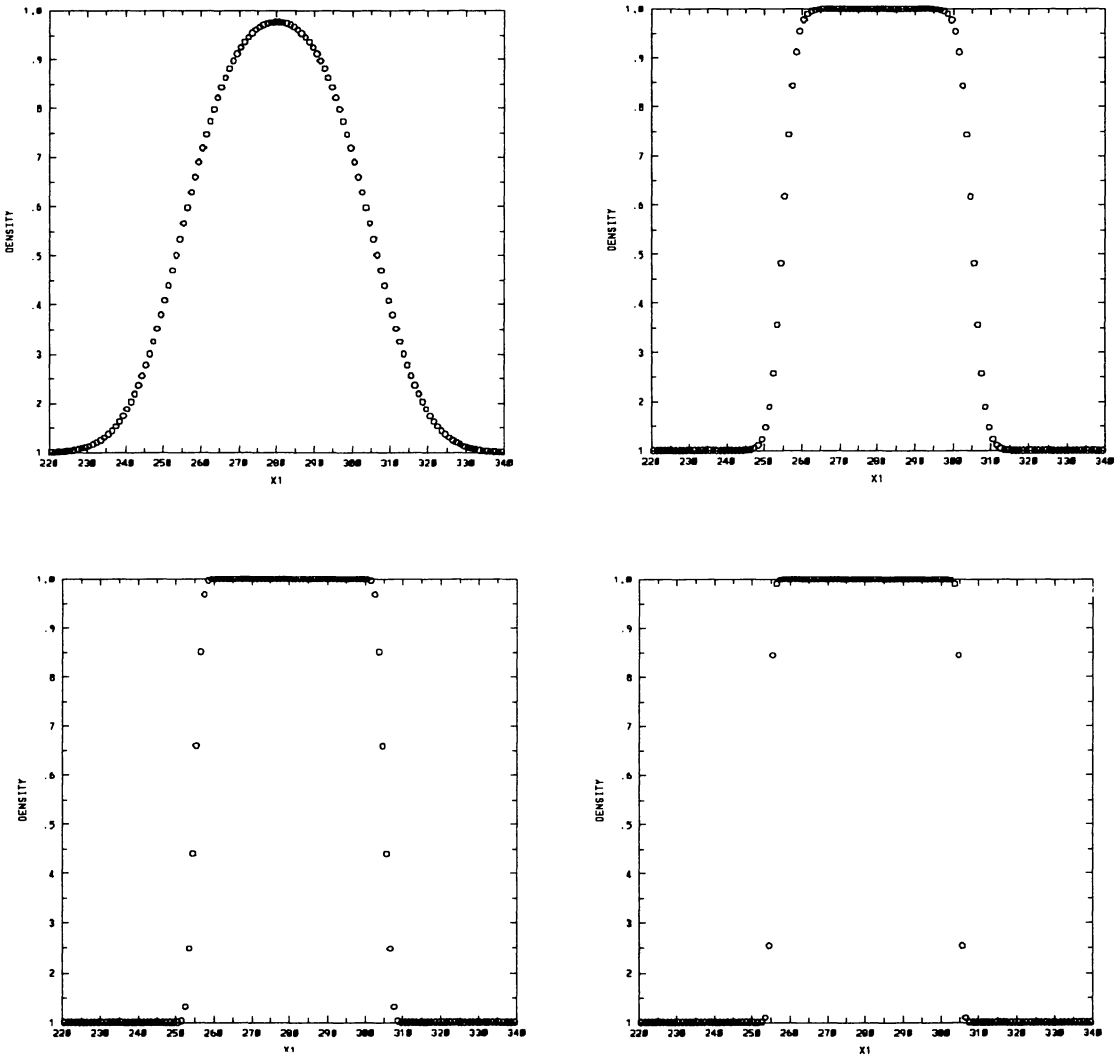

FIG. 8.—Results for the Cartesian advection of a square pulse of density, originally 50 zones wide, a distance of 5 times its width using the donor cell (*top left*), van Leer (*top right*), PPA (*bottom left*), and PPA with a steepener (*bottom right*) advection algorithms. Analytically, the edges of the pulse are located at $x = 255$ and 305.

time to the expected analytic solution. Using an L_1 error norm, the error in the numerical solution is

$$\epsilon = \left(\sum_{i=1}^{N} |q_i - \tilde{q}| \right) \Big/ N , \tag{88}$$

where N is the number of gridpoints, q_i is the numerical solution, and $\tilde{q}$ the analytic solution. For different resolutions, this error should decrease as $\epsilon \sim \Delta x^r$ where r is the convergence rate. The convergence rate is therefore found by computing the L_1 error norm at several different resolutions, and finding the slope of the log-log graph of error versus resolution. In principle, a first-order scheme should converge at a rate of 1, a second-order at a rate of 2, and a third-order at a rate of 3.

Figure 9 shows the errors and convergence rate for advecting a Gaussian pulse a distance of 10 times its width using donor cell, van Leer, and PPA. The square pulse is unsuitable for this test as it involves discontinuities which can never be resolved, regardless of the number of gridpoints. The error norms show that PPA gives the most accurate result at every resolution, over three orders of magnitude better than donor cell at the highest resolution. The van Leer method, though not as accurate as PPA, still gives errors which are two orders of magnitude smaller than donor cell at high resolution. The convergence rates for this problem are 0.6 for donor cell, 2.1 for van Leer, and 3.0 for PPA. This result agrees quite closely with what we expect for a one-dimensional advection test.

Although many other Cartesian advection tests can be designed using a variety of different waveforms (Clarke 1988; Woodward 1986), these are generally used to compare various advection algorithms. Instead, we have used the Cartesian advection tests primarily to check for bug-free coding and therefore have confined ourselves to a very limited set of waveforms. A more detailed

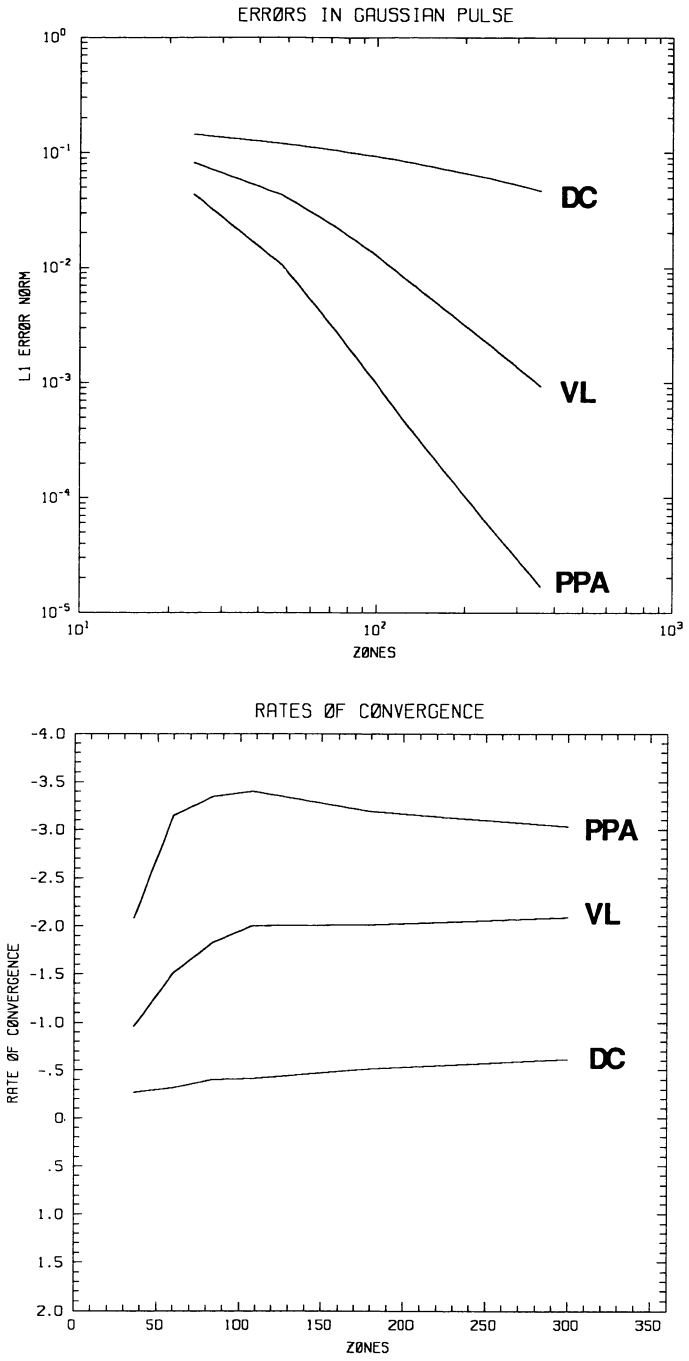

FIG. 9.—Relative errors (*top*) and convergence rates (*bottom*) of the donor cell (DC), van Leer (VL), and PPA advection algorithms for the Cartesian advection of a Gaussian pulse of density a distance of 10 times its width.

description of the performance of the advection algorithms implemented in ZEUS-2D for advecting various waveforms can be found in the above references.

5.2. *Non-Cartesian Advection Tests*

To demonstrate the covariant nature of ZEUS-2D, we can also perform advection tests in non-Cartesian geometries. A suitable one-dimensional test is a "relaxation" problem, wherein a constant density and a velocity field proportional to r ($v_r = v_0 r$) is initialized, and the density is allowed to decay away. For this problem, the analytic solution to the continuity equation is $\rho(t) = \rho_0 e^{-2v_0 t}$ in cylindrical geometry and $\rho(t) = \rho_0 e^{-3v_0 t}$ in spherical geometry. Thus, at both $v_0 t = 6$ in cylindrical geometry and $v_0 t = 4$ in spherical geometry, the analytic solution gives the same value for the density: $\rho = 6.14 \times 10^{-6} \rho_0$. The numerical solution computed

by ZEUS-2D for this problem gives density values of $\rho = 5.78\times10^{-6}$ in cylindrical geometry and $\rho = 5.60\times10^{-6}$ in spherical geometry. Thus, even though the density has decreased by nearly six orders of magnitude, the error in the amplitude is only 5.9% in cylindrical and 8.8% in spherical geometry. In addition, the numerical solution remains perfectly flat in both cases, a powerful confirmation of the geometry-dependent terms.

5.3. *Pressure-Free Collapse of a Sphere*

A final one-dimensional advection test is the gravitational collapse of a homogeneous, pressure-free sphere. An analytic solution describes the collapse of every mass shell (Hunter 1962),

$$(r/r_0) = \cos^2\beta, \qquad (\rho/\rho_0) = \cos^{-6}\beta\,, \tag{89}$$

where β is given by

$$\beta + \frac{1}{2}\sin 2\beta = t\sqrt{\frac{8\pi}{3}G\rho(0)}\,. \tag{90}$$

The free-fall time, defined as the time at which every mass shell reaches the origin ($r = 0$) simultaneously, occurs when $\beta = \pi/2$, or $\tau_{\rm ff} = \sqrt{(3\pi)/[32G\rho(0)]}$.

To run this test, we set up a homogeneous sphere in spherical geometry with $\rho = G = 1$ initially. The free-fall time is thus $\tau_{\rm ff} = 0.543$. We stop the evolution when $t = 0.535$, when the density has increased by nearly three orders of magnitude, and compare the numerical and analytic solutions. Since for a homogeneous sphere an exact numerical solution for the gravitational forces is obtained, this problem is an excellent test of radial advection.

The results as computed by ZEUS-2D are shown in Figure 10. We find that for a nonmoving grid, an anomalous spike occurs in the density in the first few zones near the origin, although the rest of the zones agree well with the analytic solution. As recently demonstrated by Mönchmeyer & Müller (1989), the error at the origin is a result of using a coordinate-centered staggered mesh. It can be entirely eliminated by using a volume-centered staggered mesh (or equivalently, using the notion of coordinate regularization in its full generality). This solution, however, requires substantial and cumbersome changes to the difference equations and the transport step. We have found this error can be substantially reduced by using a moving mesh. Thus, if we allow the mesh to collapse with the cloud by setting the grid velocity vectors equal to the fluid velocity at every gridpoint (except for the boundaries), then the advection errors are minimized, and the solution at the origin is substantially improved. This test demonstrates the usefulness of the moving mesh formulation. Furthermore, choosing the correct numerical method can be important to getting the most suitable result.

5.4. *Sod Shock-Tube Test*

We can increase the sophistication of our test problems by designing a one-dimensional problem which tests all the transport and source terms (including artificial viscosity). In HD, the quintessential test has become the shock-tube problem, first used by Sod (1978), and more recently by Hawley et al. (1984a, b), to benchmark numerical HD algorithms. The shock-tube problem involves setting up two discontinuous states, a hot dense gas on the left and a cool rarefied gas on the right, at $t = 0$, and letting them interact. Nonlinear waves are generated at the discontinuity and propagate into each state; a shock wave into the right state and a rarefaction fan into the left. The analytic solution to this problem (called a Riemann problem) gives a description of the propagation of these nonlinear waves for $t > 0$. As a test of a numerical algorithm, the shock-tube problem demonstrates whether the code can give the correct shock jump conditions and velocity in a stable manner. Although this is a one-dimensional Cartesian test, it can be run in both directions, and along the z-axis in cylindrical geometry.

The Sod shock-tube problem is a specific set of choices for the initial conditions in the left and right states. For a $\gamma = 1.4$ gas, the pressure and density are 1.0 in the left state, while in the right state the pressure is 0.1 and the density is 0.125. Initially, the discontinuity is located at $x = 0.5$ at a cell interface. The velocity is initially zero everywhere. Hawley et al. (1984a) describe how the analytic solution to this problem is generated for $t > 0$.

Figure 11 shows the results of the Sod shock-tube problem as computed by ZEUS-2D using 100 zones and the van Leer advection algorithm. The analytic solution (solid lines) shows the shock wave at far right, followed by a contact discontinuity and the rarefaction fan at left. The numerical solution (open circles) traces these features well. The shock and contact discontinuity are resolved by about 4 zones, consistent with the artificial viscosity and second-order advection scheme used. The foot and head of the rarefaction fan, which represent discontinuities in slope rather than the variables themselves, are also slightly smeared due to the diffusivity of the scheme. Using PPA on this problem does not significantly reduce this smearing. Small overshoots in the numerical solution near regions where the slope of a variable changes discontinuously from zero are also noticeable. Using a highly developed one-dimensional adaptive mesh code, Winkler & Norman (1986) have shown that such overshoots are in fact the correct solution to the *viscous* finite difference equations as $\Delta x \rightarrow 0$. These overshoots are therefore a sign of a scheme with a low diffusivity. We also note that the amplitude of the specific energy is incorrect behind the shock front; this is likely due to the fact that the fundamental variables in the shock-tube problem are density and pressure rather than density and specific internal energy, as are evolved with

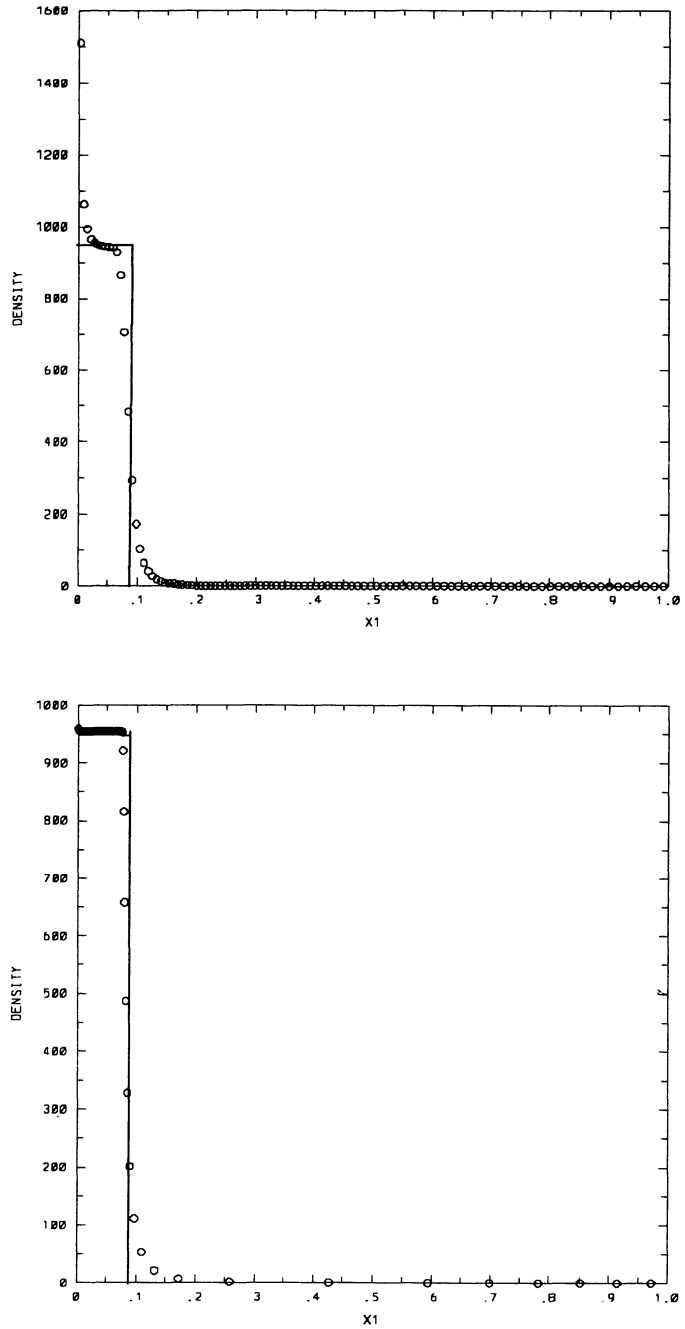

FIG. 10.—The density distribution for the pressure-free collapse of a homogeneous sphere at $t = 0.985$ free-fall times using a stationary (*top*) and a moving (*bottom*) radial mesh that collapses with the sphere. The analytic solution is denoted by the solid line in both cases.

ZEUS-2D using the consistent transport scheme. A code which does not use consistent transport would be slightly better suited to this problem. Hawley et al. (1984b) has provided extensive comparisons of commonly used numerical algorithms in astrophysics to this problem, including those used in ZEUS-2D. We find exact agreement to these published results. Hawley et al. conclude that for the algorithms that were tested for this problem, the algorithms we have implemented give the most accurate results.

5.5. *Tests with Strong Shocks*

Woodward & Colella (1984) have provided an extensive comparison of HD algorithms to three difficult test problems involving strong shocks in one and two dimensions. We have performed these tests on ZEUS-2D to benchmark our algorithms with these other codes. We describe our results below.

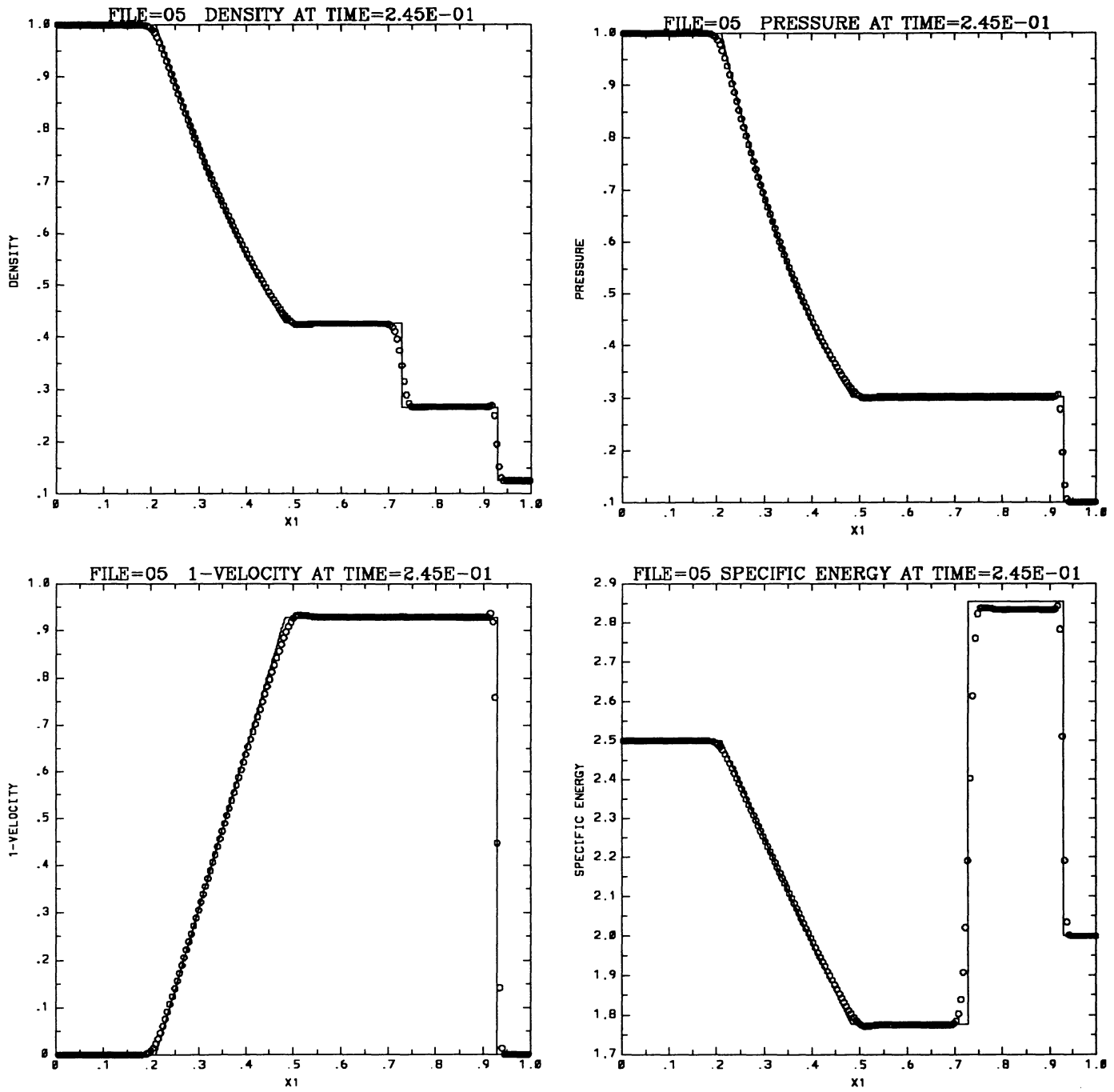

FIG. 11.—Results for the density (*top left*), pressure (*top right*), velocity (*bottom left*), and specific internal energy e/d (*bottom right*) for the Sod shock-tube problem computed using ZEUS-2D. A grid of 100 zones is used. In each case, the analytic solution is plotted as a solid line.

5.5.1. *Two Interacting Blast Waves*

The first problem described by Woodward & Colella (1984) is the interaction of two strong blast waves in one dimension. The problem is set up using three constant states of a gamma-law gas ($\gamma = 1.4$) which is at rest between two reflecting walls on the domain $x \in [0, 1]$. The density is everywhere set to 1.0, and from $x = 0$ to 0.1 the pressure is 1000, from $x = 0.9$ to 1.0 the pressure is 100, and in between it is 0.01. The discontinuities in pressure generate two strong shocks which reflect off the boundaries and interact with each other, producing a complicated pattern of shocks and discontinuities for $t > 0$. Although no analytic solution is given, Woodward & Colella present the results of a very high-resolution PPM calculation, to which the results of all other codes are compared. In Figure 12 we give the results obtained at $t = 0.038$ using ZEUS-2D with 1200 zones and both the van Leer and the PPA schemes, to be compared directly with the results given in Woodward & Colella.

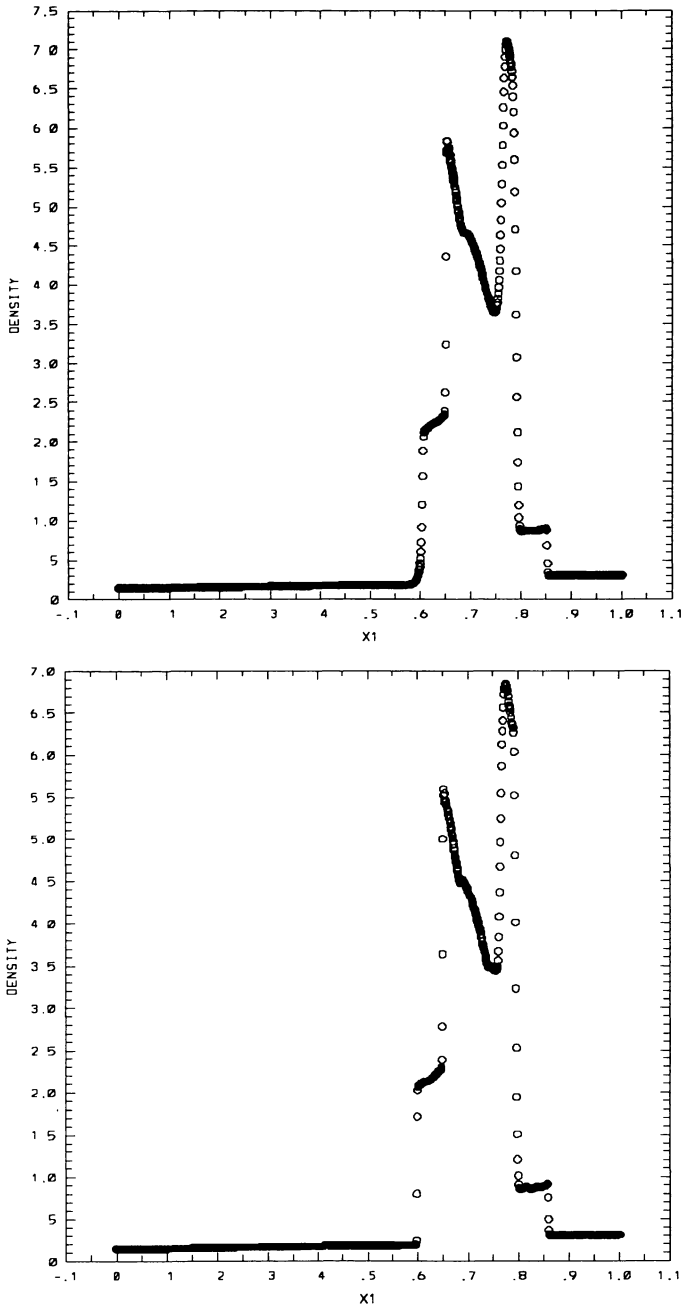

FIG. 12.—Density resulting from the interaction of two blast waves computed using 1200 zones and the van Leer (*top*) and PPA (*bottom*) advection algorithms.

Qualitatively, the ZEUS-2D solutions are very good. All of the shocks and contact discontinuities are present in their proper positions on the grid as given by the PPM solution. Comparing the solutions to each other, we see the PPA result is superior. Not only does the PPA solution have sharper contact discontinuities, but also the amplitude of the spike at $x = 0.76$ is closer to the best PPM value of ~ 6.5. Since this test was originally designed to demonstrate the relation between the accuracy of the overall solution to the thickness of discontinuities on the mesh, it is not surprising the PPA method gives better results. Though methods which keep shocks thinner on the grid do better (e.g., PPM or adaptive mesh methods; Winkler & Norman 1986), the simple algorithms in ZEUS-2D do quite well on this difficult problem.

5.5.2. *Mach 3 Wind Tunnel with a Step*

In this two-dimensional test, a uniform Mach 3 flow is set up in a wind tunnel containing a step. The tunnel is three units long and one high, and the step is 0.2 units high and located 0.6 units from the left side of the tunnel. Reflecting boundary conditions are used

for the top and bottom of the tunnel, flow in on the left and flow out on the right. The density is initially everywhere set to 1.4, the pressure is 1.0 ($\gamma = 1.4$ is used), and therefore for a uniform Mach 3 flow the velocity is everywhere 3.0. As described by Woodward & Colella (1984), we modify the flow quantities in the zones near the corner of the step to try to reduce numerical errors generated at this singular point. Thus, in the first four zones above and just to the right of the step, the density and magnitude (but not direction) of the velocity is reset so that the entropy and sum of the enthalpy and kinetic energies per unit mass are identical to the values of these quantities in the zone immediately to the left and below the corner. The same conditions are applied to the first two zones in the row above. In a staggered mesh, this reset is very difficult to implement properly. The zone-centered velocity is an average of the velocities at the interfaces; thus reseting the velocities within a zone requires modifying the interface values, and thus the velocities in the neighboring zones as well. As a result, the solutions presented here are marred by numerical errors generated by the incorrect boundary conditions at the corner of the step, especially at low resolution. Nonetheless, this problem provides a self-consistent test for comparing the advection algorithms within ZEUS-2D, as well as qualitative comparisons to other schemes at high resolution.

Figure 13 shows the resulting density profiles (plotted with 30 equally spaced contours) using the van Leer advection scheme at time $t = 4.0$ and resolutions of $\Delta x = \Delta y = \frac{1}{20}$, $\Delta x = \Delta y = \frac{1}{40}$, and $\Delta x = \Delta y = \frac{1}{80}$. Figure 14 shows the same results using the PPA method at the same time and resolutions. The effect of increasing the resolution is dramatically demonstrated in these plots. Not only are discontinuities thinner in the high-resolution runs, but also the shock positions are closer to the best PPM values. In the lowest resolution, the Mach stem at the upper boundary has not yet formed. The numerical error generated at the corner of the step is clearly demonstrated by the anomalous Mach stem at the lower reflection, caused by the interaction of the shock and an anomalous boundary layer along the surface of the step generated at the corner. This error certainly affects the overall flow, particularly the positions of the shocks and reflections. Thus, only the highest resolutions have a structure close to the best solution given by Woodward & Colella (1984).

Comparison of the PPA and van Leer results shows that PPA gives a better result at every resolution. The positions of the shocks and the length of the Mach stem are closer to the best PPM values, and the contact discontinuity generated at the triple point is sharper, as witnessed by the sharper kink in the density contours. The increased accuracy of the PPA result is bought at the expense of CPU time, however. Table 1 gives the CPU time for the solution at each resolution using both the van Leer and PPA methods.

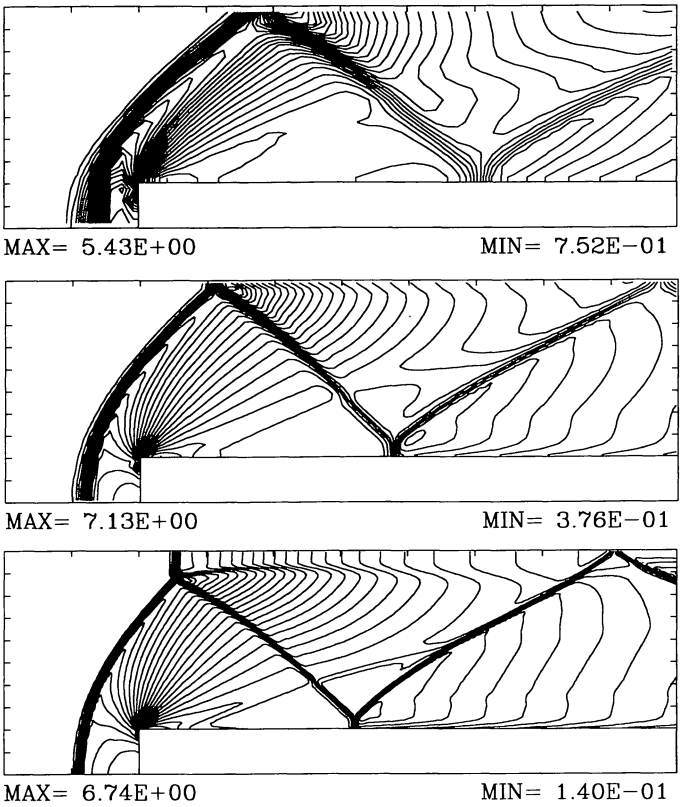

FIG. 13

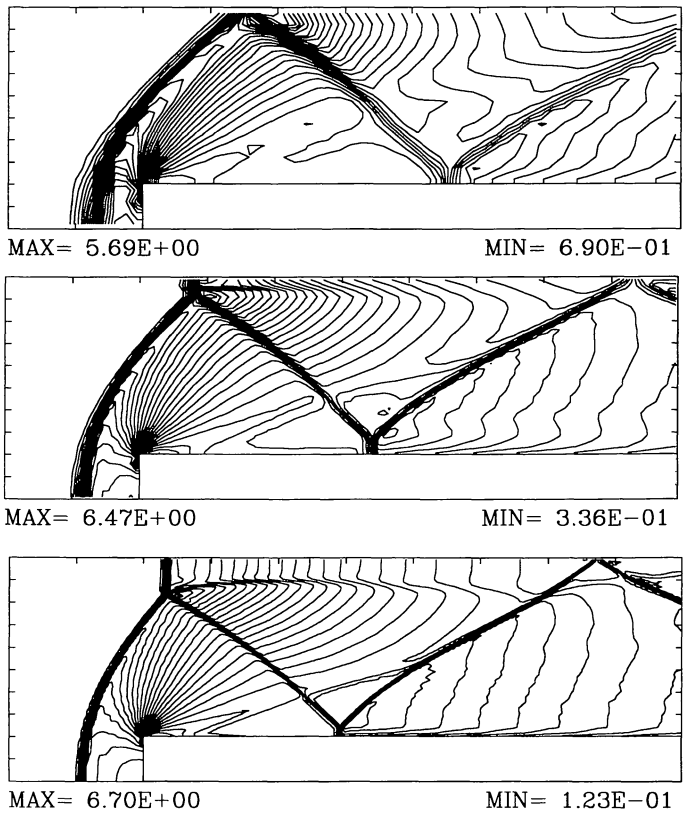

FIG. 14

FIG. 13.—Density contours resulting from the Mach 3 wind tunnel with a step using the van Leer algorithm and grids of 20×80 zones (*top*), 40×160 zones (*middle*), and 80×320 zones (*bottom*). Thirty equally spaced contours between the maximum and minimum are used.

FIG. 14.—Density contours resulting from the Mach 3 wind tunnel with a step using the PPA algorithm and grids of 20×80 zones (*top*), 40×160 zones (*middle*), and 80×320 zones (*bottom*). Thirty equally spaced contours between the maximum and minimum are used.

TABLE 1

EXECUTION TIMES FOR MACH 3 WIND TUNNEL WITH A STEP TEST ON A CRAY-XMP

Resolution	Algorithm	Cycles	CPU Time (s)
20 × 80	van Leer	1149	14.7
40 × 160	van Leer	2310	80.5
80 × 320	van Leer	4764	546
20 × 80	PPA	1227	28.1
40 × 160	PPA	2510	166
80 × 320	PPA	5114	1190

Note that the grids used for this problem cover a domain which has an axial ratio of 1:4, whereas we have used only the leftmost 1:3 portion of the domain for analysis. This avoids any numerical errors which may be associated with the outflow boundary condition on the right-hand side of the grid. For this two-dimensional test, PPA takes significantly longer, nearly 2.2 times longer at the highest resolution.

5.5.3. *Double Mach Reflection of a Strong Shock*

The final test problem described in Woodward & Colella (1984) is the double Mach reflection of a Mach 10 shock from an inclined plane. To reduce the difficulty of modeling an inclined reflecting boundary on a rectangular mesh, the shock is initially introduced at an angle to the mesh and allowed to reflect from the lower boundary. Thus, the problem is set up using a rectangular domain, one unit high and three units long, filled with a uniform gas at rest with density 1.4 and pressure 1.0 ($\gamma = 1.4$ is used). A planar shock is initialized at an angle of 60° to the x-axis, starting at $x = \frac{1}{6}$ and extending to the top at $y = 1$. The flow conditions behind the shock are given from the jump conditions using the preshock values and the shock speed. Thus, behind the shock, the density is 8, the pressure is 116.5, and the velocity is 8.25 normal to the shock front. The left boundary and the lower boundary from $x = 0$ to 1/6 are always set to the postshock conditions. From $x = \frac{1}{6}$ to 3 the lower boundary is reflecting, while the right boundary is flow out, and the upper boundary is modified to describe the exact motion of the shock. By initializing the shock at an angle to the grid, a small numerical error is introduced which is apparent in all of the plots given here as isolated contour lines stretching from the upper right corner back to the reflected shock. This same error also appears in the results given by Woodward & Colella and cannot be easily eliminated.

Figure 15 shows the resulting density profiles (plotted using 30 equally spaced contours) using the van Leer algorithm at time $t = 0.2$ and resolutions of $\Delta x = \Delta y = \frac{1}{30}$, $\Delta x = \Delta y = \frac{1}{60}$, and $\Delta x = \Delta y = \frac{1}{120}$. Figure 16 shows the same results using the PPA algorithm at the same time and resolutions. Again, the influence of resolution is dramatic in these plots. However, since there are no difficult boundary conditions which need be applied in this problem, we find much better agreement between the solutions at every resolution and the best PPM results for this problem. At high resolution, the shocks are much thinner, and the complicated shock structure at the double Mach reflection, including a small jet along the lower boundary bounded by a contact discontinuity, is captured much better.

At every resolution, the PPA results are once again superior. Indeed, at the highest resolution, there is little difference between the ZEUS-2D result using PPA and the best PPM result; the shocks in the latter case are thinner and smoother. Once again, however, the PPA result is much more computationally expensive to achieve, as demonstrated by the timing information given in Table 2. Note that once again, we have used a 1:4 computational domain, but have plotted the results only for the leftmost 1:3 portion of the grid. At the highest resolution, PPA is 2.1 times slower than van Leer.

5.6. *Tests of the Poisson Equation Solver*

Tests of the Poisson solver are fairly easy to design, as many analytic solution for the gravitational potential are possible. Even without analytic solutions, consistency checks can be performed by initializing a general density distribution on the grid, computing the numerical potential, and then differencing this potential and checking that the original density distribution is recovered. In this work, we have concentrated on testing with analytic solutions. This provides a check of the numerical gradient of the potential to the analytic solution, which is important since it is the gradient of the potential which appears in the evolution equations. We have used three analytic solutions:

1. A homogeneous sphere of radius R and density ρ_0:

$$\Phi(r) = \begin{cases} 2\pi G\rho_0(R^2 - r^2/3) & \text{if} \quad r \le R \\ \frac{4}{3}\pi G\rho_0 R^3/r & \text{if} \quad r > R \end{cases}, \tag{91}$$

$$\nabla\Phi(r) = \begin{cases} -\frac{4}{3}\pi G\rho_0 r & \text{if} \quad r \le R \\ -\frac{4}{3}\pi G\rho_0 R^3/r^2 & \text{if} \quad r > R \end{cases}; \tag{92}$$

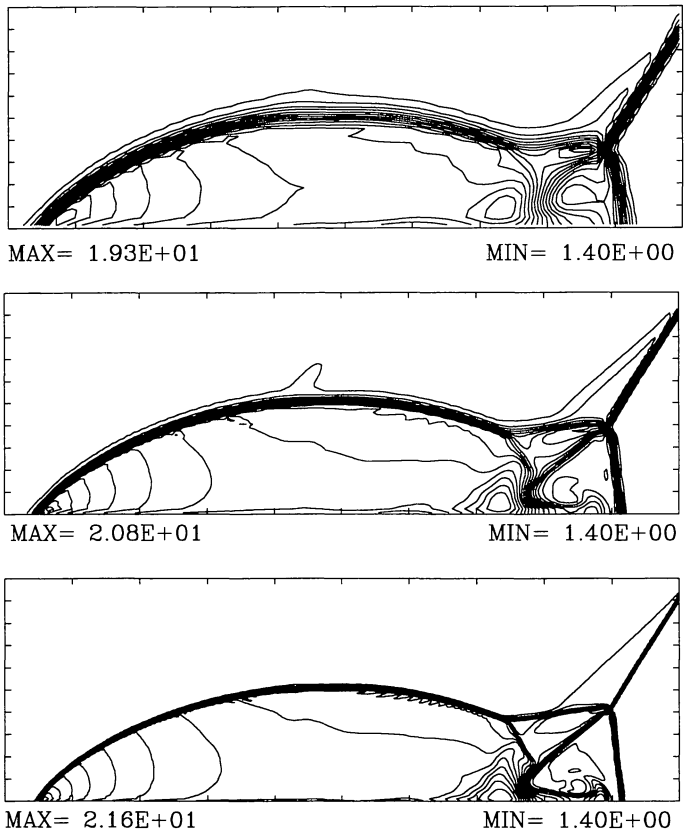

FIG. 15

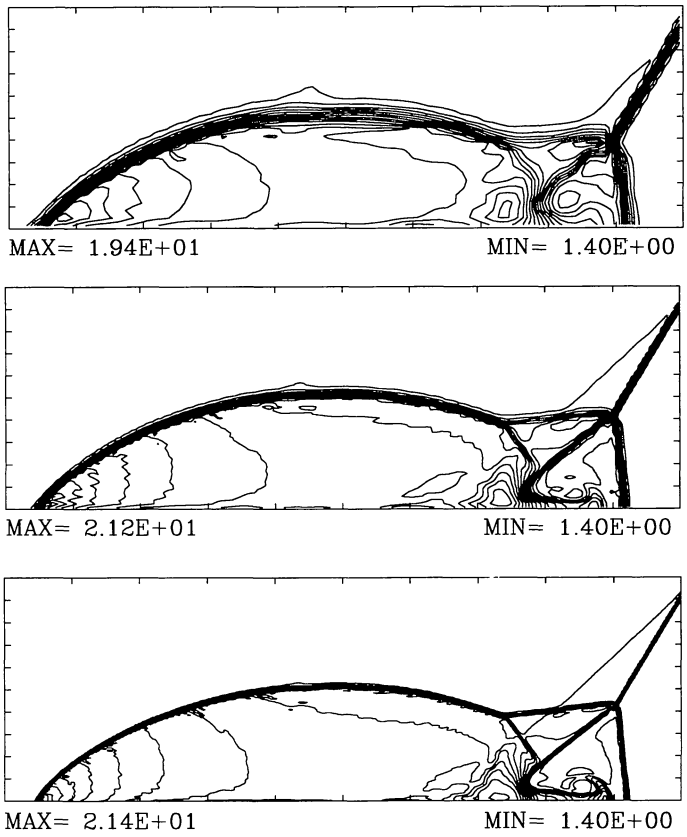

FIG. 16

FIG. 15.—Density contours resulting from the double Mach reflection of a strong shock using the van Leer algorithm and grids of 30×120 zones (*top*), 60×240 zones (*middle*), and 120×480 zones (*bottom*). Thirty equally spaced contours between the maximum and minimum are used.

FIG. 16.—Density contours resulting from the double Mach reflection of a strong shock using the PPA algorithm and grids of 30×120 zones (*top*), 60×240 zones (*middle*), and 120×480 zones (*bottom*). Thirty equally spaced contours between the maximum and minimum are used.

2. A centrally condensed sphere of radius R and density distribution

$$\rho(r) = \begin{cases} \dfrac{\rho_c}{1 + (r/r_c)^2} & \text{if} \quad r \le R \\ 0 & \text{if} \quad r > R \end{cases} \tag{93}$$

so that

$$\Phi(r) = \begin{cases} 4\pi G\rho_c r_c^2 \left[\dfrac{\arctan\,(r/r_c)}{r/r_c} + \dfrac{1}{2} \ln\left(\dfrac{1 + (r/r_c)^2}{1 + (R/r_c)^2} \right) - 1 \right] & \text{if} \quad r \le R \\ -4\pi G\rho_c r_c^3 / r[R/r_c - \arctan\,(R/r_c)] & \text{if} \quad r > R \end{cases}, \tag{94}$$

$$\nabla\Phi(r) = \frac{GM(r)}{r^2} = \begin{cases} 4\pi G\rho_c r_c^3 / r^2 [r/r_c - \arctan\,(r/r_c)] & \text{if} \quad r \le R \\ 4\pi G\rho_c r_c^3 / r^2 [R/r_c - \arctan\,(R/r_c)] & \text{if} \quad r > R \end{cases}; \tag{95}$$

3. And finally a homogeneous ellipsoid with a density distribution

$$\rho(r, z) = \begin{cases} \rho_0 & \text{if} \quad \dfrac{r^2}{a_1^2} + \dfrac{z^2}{a_3^2} \le 1 \\ 0 & \text{if} \quad \dfrac{r^2}{a_1^2} + \dfrac{z^2}{a_3^2} > 1 \end{cases}, \tag{96}$$

TABLE 2

EXECUTION TIMES FOR DOUBLE MACH REFLECTION TEST ON A CRAY-XMP

Resolution	Algorithm	Cycles	CPU Time (s)
30 × 120	van Leer	415	10.5
60 × 240	van Leer	853	64.2
120 × 480	van Leer	1734	456
30 × 120	PPA	434	19.7
60 × 240	PPA	888	129
120 × 480	PPA	1802	970

where a_1 and a_3 are the semimajor axes. From Chandrasekhar (1969, p. 42), the analytic solution for such an object is

$$\Phi(r,z)=\begin{cases}\pi G\rho_0(a_1^2A_1+a_2^2A_2+a_3^2A_3-A_1r^2-A_3z^2) & \text{if } \dfrac{r^2}{a_1^2}+\dfrac{z^2}{a_3^2}\le 1\\[2ex] \pi G\rho_0 a_1^2a_3\left\{\left(1+\dfrac{r^2}{2(a_3^2-a_1^2)}-\dfrac{z^2}{a_3^2-a_1^2}\right)I_1\right. & \\ \left.-r^2\dfrac{\sqrt{a_3^2+\lambda}}{(a_3^2-a_1^2)(a_1^2+\lambda)}-z^2\left[\dfrac{2}{(a_1^2+\lambda)\sqrt{a_3^2+\lambda}}-\dfrac{2\sqrt{a_3^2+\lambda}}{(a_3^2-a_1^2)(a_1^2+\lambda)}\right]\right\} & \text{if } \dfrac{r^2}{a_1^2}+\dfrac{z^2}{a_3^2}>1\end{cases}, \tag{97}$$

$$\nabla\Phi(r,z)=\begin{cases}2\pi G\rho_0(A_1^2r^2+A_3^2z^2)^{1/2} & \text{if } \dfrac{r^2}{a_1^2}+\dfrac{z^2}{a_3^2}\le 1\\[2ex] 2\pi G\rho_0 a_1^2a_3\left[\left[\dfrac{rI_1}{2(a_3^2-a_1^2)}-\dfrac{r\sqrt{a_3^2+\lambda}}{(a_3^2-a_1^2)(a_1^2+\lambda)}\right]^2\right. & \\ \left.+\left\{\dfrac{zI_1}{a_3^2-a_1^2}+z\left[\dfrac{2}{(a_1^2+\lambda)\sqrt{a_3^2+\lambda}}-\dfrac{2\sqrt{a_3^2+\lambda}}{(a_3^2-a_1^2)(a_1^2+\lambda)}\right]\right\}^2\right]^{1/2} & \text{if } \dfrac{r^2}{a_1^2}+\dfrac{z^2}{a_3^2}>1\end{cases}, \tag{98}$$

where for an *oblate* spheroid ($a_1=a_2>a_3$)

$$A_1=A_2=[(1-e^2)^{1/2}/e^3]\arcsin e-(1-e^2)/e^2\,, \tag{99}$$

$$A_3=2e^{-2}-2[(1-e^2)/e^3]\arcsin e\,, \tag{100}$$

$$e=[1-(a_3/a_1)^2]^{1/2}\,, \tag{101}$$

TABLE 3

ERRORS FOR TESTS OF THE POISSON EQUATION SOLVER

Object[a]	Symmetry[b]	Geometry[c]	Relative Error in Φ	Relative Error in ∇Φ
HS	E	RZ	2.23×10^{-4}	6.04×10^{-3}
CC	E	RZ	3.80×10^{-4}	2.21×10^{-3}
HE	E	RZ	2.42×10^{-3}	1.96×10^{-2}
HS	O	RZ	1.56×10^{-3}	1.91×10^{-2}
CC	O	RZ	4.29×10^{-3}	2.22×10^{-2}
HE	O	RZ	8.37×10^{-3}	6.44×10^{-2}
HS	E	RT	4.02×10^{-5}	1.11×10^{-4}
CC	E	RT	5.60×10^{-4}	9.98×10^{-4}
HE	E	RT	3.95×10^{-3}	1.86×10^{-2}
HS	O	RT	1.31×10^{-4}	9.02×10^{-4}
CC	O	RT	5.71×10^{-4}	9.99×10^{-4}
HE	O	RT	5.83×10^{-3}	1.74×10^{-2}

[a] HS = homogeneous sphere; CC = centrally condensed sphere; HE = homogeneous ellipsoid.
[b] E = even (equatorial) symmetry; O = odd (no equatorial) symmetry.
[c] RZ = cylindrical geometry; RT = spherical polar geometry.

$$I_1 = \frac{\pi}{\sqrt{a_1^2 - a_3^2}} - \frac{2}{\sqrt{a_1^2 - a_3^2}} \arctan \sqrt{\frac{a_3^2 + \lambda}{a_1^2 - a_3^2}}, \tag{102}$$

$$\lambda = [(r^2 + z^2 - a_1^2 - a_3^2) + \sqrt{(a_1^2 + a_3^2 - r^2 - z^2)^2 - 4(a_1^2 a_3^2 - r^2 a_3^2 - z^2 a_1^2)}]/2 , \tag{103}$$

and for a *prolate* spheroid ($a_1 = a_2 < a_3$)

$$A_1 = A_2 = e^{-2} - (1 - e^2)/(2e^3) \ln [(1 + e)/(1 - e)] , \tag{104}$$

$$A_3 = (1 - e^2)/e^3 \ln [(1 + e)/(1 - e)] - 2(1 - e^2)/e^2 , \tag{105}$$

$$e = [1 - (a_1/a_3)^2]^{1/2} , \tag{106}$$

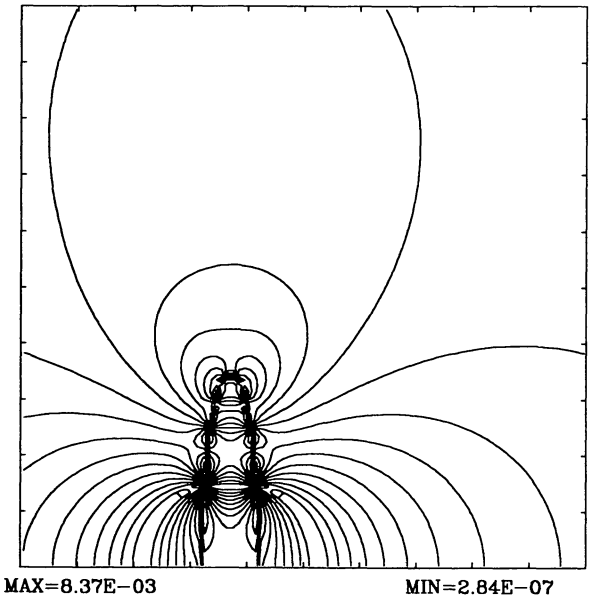

FIG. 17.—Contours of the error in the potential (*top*) and gradient of the potential (*bottom*) for a homogeneous ellipsoid in cylindrical geometry with no assumed equatorial symmetry. The axial coordinate Z is plotted horizontally, the radial coordinate R vertically. Sixteen equally spaced contours between the maximum and minimum are used.

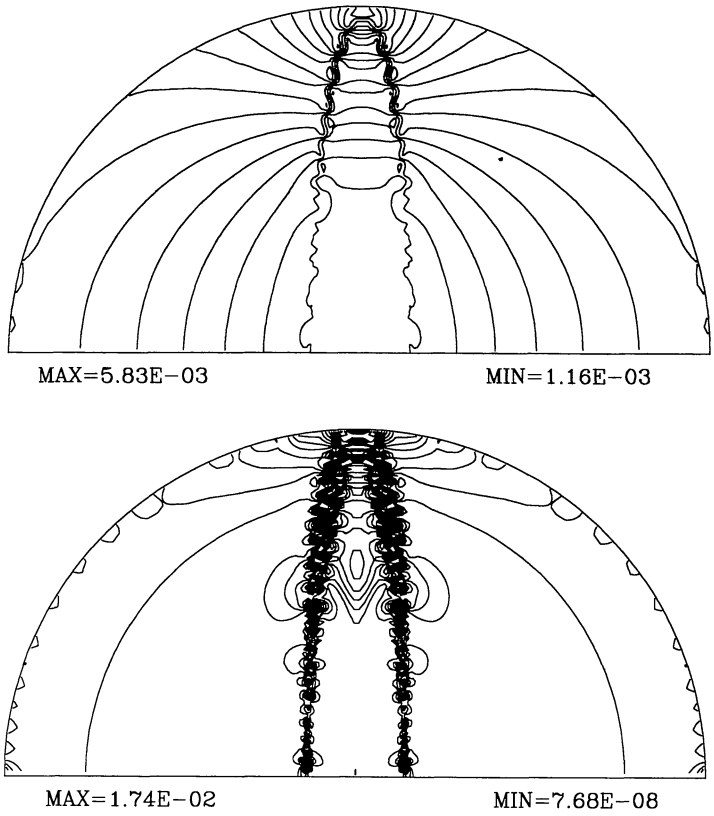

FIG. 18.—Contours of the error in the potential (*top*) and gradient of the potential (*bottom*) for a homogeneous ellipsoid in spherical geometry with no assumed equatorial symmetry. Sixteen equally spaced contours between the maximum and minimum are used.

$$I_1 = \frac{-1}{\sqrt{a_3^2 - a_1^2}} \ln\left[\frac{(\sqrt{a_3^2 + \lambda} - \sqrt{a_3^2 - a_1^2})^2}{a_1^2 + \lambda}\right], \tag{107}$$

$$\lambda = [(r^2 + z^2 - a_1^2 - a_3^2) + \sqrt{(a_1^2 + a_3^2 - r^2 - z^2)^2 - 4(a_1^2 a_3^2 - r^2 a_3^2 - z^2 a_1^2)}]/2\,. \tag{108}$$

For the tests, we have initialized each of the above configurations in spherical and cylindrical geometry both with and without equatorial symmetry, and compared the numerical potential and gradient to the above analytic results. Table 3 lists the results of these tests.

In Figures 17 and 18 we give contour plots of the errors in the potential and the gradient of the potential for a homogeneous ellipsoid in cylindrical and spherical geometry with no equatorial symmetry assumed (the largest errors occur for this case). We find the bulk of the error is confined to the surface on the object where the density drops discontinuously to zero. Resolution strongly affects the solution here, as the numerical representation of the smooth boundary is very approximate. Even so, from Table 3, we find the largest error in the gradient is 6.4%, in cylindrical geometry, and this is dramatically reduced to 1.74% if we use spherical coordinates which are better able to describe the ellipsoid. The contour plots also show that even when no equatorial symmetry is assumed, the error contours are symmetric and smooth across the equator.

6. SUMMARY

In this paper we have given a detailed description of the HD algorithms in the ZEUS-2D code. These algorithms form the foundation of the more complex MHD and RHD algorithms described in Papers II and III, respectively.

In addition to the description of the algorithm, we have also presented the results from the HD test problems performed on ZEUS-2D. These tests are intended not only to demonstrate the correct implementation of the algorithms, but also to gain insight into the performance of the methods in obtaining solutions to physical problems. This insight may be invaluable in avoiding potential problems caused by numerics rather than the physics being simulated.

A great variety of test problems (a "test suite") have been used to calibrate ZEUS-2D. The advection tests demonstrate the range of accuracy inherent to the various advection algorithms implemented in ZEUS-2D. The PPA algorithm has the lowest intrinsic

diffusion and highest rate of convergence and would therefore be the clear choice for pure advection problems. The one-dimensional shock-tube tests recover exactly the results published previously for the algorithms we have implemented. In addition, we have also performed the series of tests involving strong shocks described by Woodward & Colella (1984) and find that qualitatively, our results compare well with the best PPM results for the same problems. Although Godunov-type codes such as PPM produce superior results to the HD methods implemented in ZEUS-2D for problems involving strong shocks, it is very difficult to incorporate new physical effects (such as RMHD) into these schemes in a self-consistent fashion, which precludes their use here. Comparisons using van Leer and PPA advection on these problems show that PPA always gives the most accurate result. However, PPA requires a greater computational expense; typically PPA is twice as slow as van Leer advection for two-dimensional problems. On the other hand, donor cell advection is clearly too diffusive for any practical applications and is therefore never used in simulations. Finally, tests of the Poisson equation solver implemented in ZEUS-2D with analytic solutions demonstrate the small errors present in the gradient of the potential can be primarily attributed to the lack of resolution of the density distribution.

We thank David Clarke, Chuck Evans, John Hawley, Dimitri Mihalas, and Louis Wicker for many useful discussions. J. S. would like to thank Dimitri Mihalas, the Department of Astronomy, and the National Center for Supercomputing Applications (NCSA) at the University of Illinois for financial support during this work. The computations were performed on the Cray X-MP at the NCSA.

APPENDIX A
COVARIANT EXPRESSIONS FOR VECTOR AND TENSOR OPERATORS

In this appendix we give coordinate independent expressions for all vector and tensor operators used in this work. These expressions can be derived directly from the covariant forms for the operators, and the relations between the components of abstract covariant or contravariant tensors and the components of the associated physical tensors. Such formulae can be found in virtually any text on tensor calculus, as well as Mihalas & Mihalas (1984).

For brevity, we shall consider only orthogonal coordinate systems (although the method can be extended to nonorthogonal coordinate systems as well), so that the metric tensor g_{ij} which characterizes the space is diagonal and can be written as

$$g_{ij} = \begin{pmatrix} h_1^2 & 0 & 0 \\ 0 & h_2^2 & 0 \\ 0 & 0 & h_3^2 \end{pmatrix} . \tag{109}$$

The determinant g of the metric tensor, which appears in many tensor formulae, is then simply $g = h_1^2 h_2^2 h_3^2$. The h_i are the scale factors of the metric and in general are functions of all of the coordinates, $h_i = h_i(x_1, x_2, x_3)$. In this work we are primarily interested in three coordinate systems, *Cartesian* coordinates for which

$$(x_i, x_2, x_3) = (x, y, z), \qquad (h_1, h_2, h_3) = (1, 1, 1) , \tag{110}$$

cylindrical coordinates for which

$$(x_1, x_2, x_3) = (z, r, \phi), \qquad (h_1, h_2, h_3) = (1, 1, r) , \tag{111}$$

and *spherical polar* coordinates for which

$$(x_1, x_2, x_3) = (r, \theta, \phi), \qquad (h_1, h_2, h_3) = (1, r, r \sin\theta) . \tag{112}$$

In these cases, x_3 is always the ignorable coordinate for two-dimensional calculations. As well, the scale factors are particularly simple functions of the coordinates, which has implications for the efficient implementation of the coordinate independent expressions. However, in what follows we make no restrictions on the number of dimensions or on the functional form of the scale factors. This ensures the expressions can be applied to three-dimensional calculations, as well as more complex orthogonal coordinate systems.

For a diagonal metric, the relations between the components of a *covariant* vector a_i or tensor T_{ij} and the components of the corresponding physical vector $a_{(i)}$ or tensor $T_{(i,j)}$ are

$$a_i = h_i a_{(i)}, \qquad T_{ij} = h_i h_j T_{(i,j)} , \tag{113}$$

while the relations between the components of a *contravariant* vector a^i or tensor T^{ij} and the components of the corresponding physical vector $a_{(i)}$ or tensor $T_{(i,j)}$ are

$$a^i = a_{(i)}/h_i , \qquad T^{ij} = T_{(i,j)}/h_i h_j . \tag{114}$$

The coordinate independent expressions for various operators are then as follows.

A1. GRADIENT OF A SCALAR, ∇f

The covariant expression for the gradient of a scalar is $f_{,i} = \partial f/\partial x_i$. Converting the components of the resulting covariant vector to physical components gives $(\nabla f)_{(i)} = f_{,i}/h_i$, or writing it out

$$(\nabla f) = \left(\frac{1}{h_1}\frac{\partial f}{\partial x_1}, \frac{1}{h_2}\frac{\partial f}{\partial x_2}, \frac{1}{h_3}\frac{\partial f}{\partial x_3}\right). \tag{115}$$

A2. DIVERGENCE OF A VECTOR, $\nabla \cdot \boldsymbol{a}$

The divergence of a contravariant vector is $a^i_{;i} = g^{-1/2}(g^{1/2}a^i)_{,i}$. Converting the components of the contravariant vector to physical components gives $(\nabla \cdot \boldsymbol{a}) = a^i_{;i} = g^{-1/2}(g^{1/2}a_{(i)}/h_i)_{,i}$ which can be written out as

$$\nabla \cdot \boldsymbol{a} = \frac{1}{h_1h_2h_3}\left[\frac{\partial}{\partial x_1}(h_2h_3a_1) + \frac{\partial}{\partial x_2}(h_1h_3a_2) + \frac{\partial}{\partial x_3}(h_1h_2a_3)\right]. \tag{116}$$

A3. LAPLACIAN OF A SCALAR, $\nabla^2 f$

The Laplacian of a scalar is most easily derived from the vector identity $\nabla^2 f = \nabla \cdot \nabla f$. Thus, using the previous two results gives $(\nabla^2 f) = g^{-1/2}(g^{1/2}g^{ij}f_{,i})_{,i}$. Expanding this result and noting that $g_{ij} = 0$ for $i \neq j$ gives

$$\nabla^2 f = \frac{1}{h_1h_2h_3}\left[\frac{\partial}{\partial x_1}\left(\frac{h_2h_3}{h_1}\frac{\partial f}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\frac{h_1h_3}{h_2}\frac{\partial f}{\partial x_2}\right) + \frac{\partial}{\partial x_3}\left(\frac{h_1h_2}{h_3}\frac{\partial f}{\partial x_3}\right)\right]. \tag{117}$$

A4. CURL OF A VECTOR, $\nabla \times \boldsymbol{a}$

The covariant expression for the curl of a vector is $(\nabla \times \boldsymbol{a})^i = g^{-1/2}(a_{k;j} - a_{j;k})$. Converting the components of this vector to physical components, and replacing the covariant vector a_i with physical components, gives $(\nabla \times \boldsymbol{a})_{(i)} = h_i(\nabla \times \boldsymbol{a})^i = h_i g^{-1/2}[(h_k a_{(k)})_{,j} - (h_j a_{(j)})_{,k}]$, or writing each component out:

$$(\nabla \times \boldsymbol{a})_1 = \frac{1}{h_2h_3}\left[\frac{\partial}{\partial x_2}(h_3a_3) - \frac{\partial}{\partial x_3}(h_2a_2)\right], \tag{118}$$

$$(\nabla \times \boldsymbol{a})_2 = \frac{1}{h_1h_3}\left[\frac{\partial}{\partial x_3}(h_1a_1) - \frac{\partial}{\partial x_1}(h_3a_3)\right], \tag{119}$$

$$(\nabla \times \boldsymbol{a})_3 = \frac{1}{h_1h_2}\left[\frac{\partial}{\partial x_1}(h_2a_2) - \frac{\partial}{\partial x_2}(h_1a_1)\right]. \tag{120}$$

A5. GRADIENT OF A VECTOR, $\nabla \boldsymbol{a}$

The covariant expression for the gradient of a vector is $a_{i;j} = a_{i,j} - \{^k_{ij}\}a_k$, where $\{^k_{ij}\}$ is the Christoffel symbol of the second kind. The result is a rank-two covariant tensor. Converting it, and the components of the covariant vector a_i to physical components, we find $(\nabla \boldsymbol{a})_{(ij)} = (h_ih_j)^{-1}a_{i;j} = (h_ih_j)^{-1}[(h_ia_{(i)})_{,j} - \{^k_{ij}\}h_ka_{(k)}]$. Now, for an orthogonal coordinate system, the only nonzero Christoffel symbols are

$$\left\{\begin{matrix} i \\ ii \end{matrix}\right\} = \frac{1}{2}(\ln g_{ii})_{,i} = \frac{1}{2g_{ii}}\frac{\partial g_{ii}}{\partial x_i}, \tag{121}$$

$$\left\{\begin{matrix} i \\ jj \end{matrix}\right\} = -\frac{1}{2}(g_{jj})_{,i}\Big/ g_{ii} = -\frac{1}{2g_{ii}}\frac{\partial g_{jj}}{\partial x_i}, \tag{122}$$

$$\left\{\begin{matrix} i \\ ij \end{matrix}\right\} = \left\{\begin{matrix} i \\ ji \end{matrix}\right\} = \frac{1}{2}(\ln g_{ii})_{,j} = \frac{1}{2g_{ii}}\frac{\partial g_{ii}}{\partial x_j}. \tag{123}$$

Thus, for example, a representative diagonal element of the tensor is

$$\begin{aligned}\nabla \boldsymbol{a}_{(11)} &= \frac{1}{h_1^2}\left[\frac{\partial}{\partial x_1}(h_1a_1) - \frac{h_1a_1}{2h_1^2}\frac{\partial}{\partial x_1}(h_1^2) + \frac{h_2a_2}{2h_2^2}\frac{\partial}{\partial x_2}(h_1^2) + \frac{h_3a_3}{2h_3^2}\frac{\partial}{\partial x_3}(h_1^2)\right] \\ &= \frac{1}{h_1}\frac{\partial a_1}{\partial x_1} + \frac{a_2}{h_1h_2}\frac{\partial h_1}{\partial x_2} + \frac{a_3}{h_1h_3}\frac{\partial h_1}{\partial x_3},\end{aligned} \tag{124}$$

and a representative off-diagonal element is

$$\nabla \boldsymbol{a}_{(12)} = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial x_1}(h_2 a_2) - \frac{h_1 a_1}{2h_1^2}\frac{\partial}{\partial x_2}(h_1^2) - \frac{h_2 a_2}{2h_2^2}\frac{\partial}{\partial x_1}(h_2^2)\right]$$

$$= \frac{1}{h_1}\frac{\partial a_2}{\partial x_1} - \frac{a_1}{h_1 h_2}\frac{\partial h_1}{\partial x_2}. \tag{125}$$

All other components can be generated using a similar analysis, so that

$$\nabla \boldsymbol{a} = \begin{Bmatrix} \nabla \boldsymbol{a}_{(11)} & \nabla \boldsymbol{a}_{(12)} & \nabla \boldsymbol{a}_{(13)} \\ \nabla \boldsymbol{a}_{(21)} & \nabla \boldsymbol{a}_{(22)} & \nabla \boldsymbol{a}_{(23)} \\ \nabla \boldsymbol{a}_{(31)} & \nabla \boldsymbol{a}_{(32)} & \nabla \boldsymbol{a}_{(33)} \end{Bmatrix}, \tag{126}$$

where

$$\nabla \boldsymbol{a}_{(11)} = \frac{1}{h_1}\frac{\partial a_1}{\partial x_1} + \frac{a_2}{h_1 h_2}\frac{\partial h_1}{\partial x_2} + \frac{a_3}{h_1 h_3}\frac{\partial h_1}{\partial x_3},$$

$$\nabla \boldsymbol{a}_{(12)} = \frac{1}{h_1}\frac{\partial a_2}{\partial x_1} - \frac{a_1}{h_1 h_2}\frac{\partial h_1}{\partial x_2},$$

$$\nabla \boldsymbol{a}_{(13)} = \frac{1}{h_1}\frac{\partial a_3}{\partial x_1} - \frac{a_1}{h_1 h_3}\frac{\partial h_1}{\partial x_3},$$

$$\nabla \boldsymbol{a}_{(21)} = \frac{1}{h_2}\frac{\partial a_1}{\partial x_2} - \frac{a_2}{h_1 h_2}\frac{\partial h_2}{\partial x_1},$$

$$\nabla \boldsymbol{a}_{(22)} = \frac{1}{h_2}\frac{\partial a_2}{\partial x_2} + \frac{a_1}{h_1 h_2}\frac{\partial h_2}{\partial x_1} + \frac{a_3}{h_2 h_3}\frac{\partial h_2}{\partial x_3},$$

$$\nabla \boldsymbol{a}_{(23)} = \frac{1}{h_2}\frac{\partial a_3}{\partial x_2} - \frac{a_2}{h_2 h_3}\frac{\partial h_2}{\partial x_3},$$

$$\nabla \boldsymbol{a}_{(31)} = \frac{1}{h_3}\frac{\partial a_1}{\partial x_3} - \frac{a_3}{h_1 h_3}\frac{\partial h_3}{\partial x_1},$$

$$\nabla \boldsymbol{a}_{(32)} = \frac{1}{h_3}\frac{\partial a_2}{\partial x_3} - \frac{a_3}{h_2 h_3}\frac{\partial h_3}{\partial x_2},$$

$$\nabla \boldsymbol{a}_{(33)} = \frac{1}{h_3}\frac{\partial a_3}{\partial x_3} + \frac{a_1}{h_1 h_3}\frac{\partial h_3}{\partial x_1} + \frac{a_2}{h_2 h_3}\frac{\partial h_3}{\partial x_2}.$$

A6. LAPLACIAN OF A VECTOR, $\nabla^2 \boldsymbol{a}$

The easiest way to derive the Laplacian of a vector is to use the vector identity $\nabla^2 \boldsymbol{a} = \nabla(\nabla \cdot \boldsymbol{a}) - \nabla \times \nabla \times \boldsymbol{a}$. Combining the coordinate independent expressions (eqs. [115], [116], and [118]–[120]) above gives

$$(\nabla^2 \boldsymbol{a})_{(1)} = \frac{1}{h_1}\frac{\partial}{\partial x_1}\left\{\frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial x_1}(h_2 h_3 a_1) + \frac{\partial}{\partial x_2}(h_1 h_3 a_2) + \frac{\partial}{\partial x_3}(h_1 h_2 a_3)\right]\right\}$$
$$- \frac{1}{h_2 h_3}\left[\left[\frac{\partial}{\partial x_2}\left\{\frac{h_3}{h_1 h_2}\left[\frac{\partial(h_2 a_2)}{\partial x_1} - \frac{\partial(h_1 a_1)}{\partial x_2}\right]\right\} - \frac{\partial}{\partial x_3}\left\{\frac{h_2}{h_1 h_3}\left[\frac{\partial(h_1 a_1)}{\partial x_3} - \frac{\partial(h_3 a_3)}{\partial x_1}\right]\right\}\right]\right], \tag{127}$$

$$(\nabla^2 \boldsymbol{a})_{(2)} = \frac{1}{h_2}\frac{\partial}{\partial x_2}\left\{\frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial x_1}(h_2 h_3 a_1) + \frac{\partial}{\partial x_2}(h_1 h_3 a_2) + \frac{\partial}{\partial x_3}(h_1 h_2 a_3)\right]\right\}$$
$$- \frac{1}{h_1 h_3}\left[\left[\frac{\partial}{\partial x_3}\left\{\frac{h_1}{h_2 h_3}\left[\frac{\partial(h_3 a_3)}{\partial x_2} - \frac{\partial(h_2 a_2)}{\partial x_3}\right]\right\} - \frac{\partial}{\partial x_1}\left\{\frac{h_3}{h_1 h_2}\left[\frac{\partial(h_2 a_2)}{\partial x_1} - \frac{\partial(h_1 a_1)}{\partial x_2}\right]\right\}\right]\right], \tag{128}$$

$$(\nabla^2 \boldsymbol{a})_{(3)} = \frac{1}{h_3}\frac{\partial}{\partial x_3}\left\{\frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial x_1}(h_2 h_3 a_1) + \frac{\partial}{\partial x_2}(h_1 h_3 a_2) + \frac{\partial}{\partial x_3}(h_1 h_2 a_3)\right]\right\}$$
$$- \frac{1}{h_1 h_2}\left[\left[\frac{\partial}{\partial x_1}\left\{\frac{h_2}{h_1 h_3}\left[\frac{\partial(h_1 a_1)}{\partial x_3} - \frac{\partial(h_3 a_3)}{\partial x_1}\right]\right\} - \frac{\partial}{\partial x_2}\left\{\frac{h_1}{h_2 h_3}\left[\frac{\partial(h_3 a_3)}{\partial x_2} - \frac{\partial(h_2 a_2)}{\partial x_3}\right]\right\}\right]\right]. \quad (129)$$

A7. DIVERGENCE OF A RANK-TWO TENSOR, $\nabla \cdot \mathbf{T}$

The divergence of a contravariant rank-two tensor is $T^{ij}_{;j} = g^{-1/2}(g^{1/2}T^{ij})_{,j} + \{^{i}_{jk}\}T^{kj}$, which produces a covariant vector. Converting everything to physical components gives $(\nabla \cdot \mathbf{T})_{(i)} = h_i g^{-1/2}(g^{1/2}T_{(ij)}/h_i h_j)_{,j} + \{^{i}_{jk}\}h_i T_{(kj)}/h_k h_j$. The only nonzero Christoffel symbols for an orthogonal coordinate system are given by equations (121)–(123). Thus, after much algebra, we can write out the components of the divergence of a tensor as

$$(\nabla \cdot \mathbf{T})_{(1)} = \frac{1}{h_2 h_3}\left[\frac{\partial}{\partial x_1}\left(\frac{h_2 h_3}{h_1}T_{11}\right) + \frac{\partial}{\partial x_2}(h_3 T_{12}) + \frac{\partial}{\partial x_3}(h_2 T_{13})\right]$$
$$- \frac{T_{11}}{h_1^2}\frac{\partial h_1}{\partial x_1} - \frac{T_{22}}{h_1 h_2}\frac{\partial h_2}{\partial x_1} - \frac{T_{33}}{h_1 h_3}\frac{\partial h_3}{\partial x_1} + \frac{(T_{12}+T_{21})}{h_1 h_2}\frac{\partial h_1}{\partial x_2} + \frac{(T_{13}+T_{31})}{h_1 h_3}\frac{\partial h_1}{\partial x_3}, \quad (130)$$

$$(\nabla \cdot \mathbf{T})_{(2)} = \frac{1}{h_1 h_3}\left[\frac{\partial}{\partial x_1}(h_3 T_{21}) + \frac{\partial}{\partial x_2}\left(\frac{h_1 h_3}{h_2}T_{22}\right) + \frac{\partial}{\partial x_3}(h_1 T_{23})\right]$$
$$- \frac{T_{11}}{h_1 h_2}\frac{\partial h_1}{\partial x_2} - \frac{T_{22}}{h_2^2}\frac{\partial h_2}{\partial x_2} - \frac{T_{33}}{h_2 h_3}\frac{\partial h_3}{\partial x_2} + \frac{(T_{12}+T_{21})}{h_1 h_2}\frac{\partial h_2}{\partial x_1} + \frac{(T_{23}+T_{32})}{h_2 h_3}\frac{\partial h_2}{\partial x_3}, \quad (131)$$

$$(\nabla \cdot \mathbf{T})_{(3)} = \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial x_1}(h_2 T_{31}) + \frac{\partial}{\partial x_2}(h_1 T_{32}) + \frac{\partial}{\partial x_3}\left(\frac{h_1 h_2}{h_3}T_{33}\right)\right]$$
$$- \frac{T_{11}}{h_1 h_3}\frac{\partial h_1}{\partial x_3} - \frac{T_{22}}{h_2 h_3}\frac{\partial h_2}{\partial x_3} - \frac{T_{33}}{h_3^2}\frac{\partial h_3}{\partial x_3} + \frac{(T_{13}+T_{31})}{h_1 h_3}\frac{\partial h_3}{\partial x_1} + \frac{(T_{23}+T_{32})}{h_2 h_3}\frac{\partial h_3}{\partial x_2}. \quad (132)$$

Note that for a symmetric tensor, $T_{ij} = T_{ji}$, and the final terms in equations (130)–(132) sum in pairs. For an antisymmetric tensor, $T_{ij} = -T_{ji}$, and $T_{ii} = 0$, so that the final terms cancel in pairs, and all diagonal elements are identically zero.

APPENDIX B
TENSOR ARTIFICIAL VISCOSITY

The tensor artificial viscosity can be generated by analogy to the general form of the stress tensor for molecular viscosity of a Newtonian fluid (e.g., Mihalas & Mihalas 1984, § 25),

$$\sigma = \mu[\nabla \boldsymbol{v} - \tfrac{1}{3}(\nabla \cdot \boldsymbol{v})\mathbf{e}], \quad (133)$$

where μ is the coefficient of dynamical viscosity, $\nabla \boldsymbol{v}$ is the *symmetrized* velocity gradient tensor ($\nabla \boldsymbol{v} = [v_{i,j} + v_{j,i}]/2$), and $\mathbf{e}$ is the unit tensor. In writing equation (133), we have made the usual assumption that the coefficient of bulk viscosity for the fluid is zero (Stoke's hypothesis). Note also that by construction, the viscosity tensor is traceless (since $\mathbf{Tr}(\nabla \boldsymbol{v}) = \nabla \cdot \boldsymbol{v}$). Physically, this implies that the viscosity tensor is zero for a fluid which dilates symmetrically, which one can argue to be true on intuitive physical grounds.

The primary difference between the real molecular viscosity of a fluid and artificial viscosity is the magnitude of the coefficient of dynamic viscosity μ. For artificial viscosity, the value of this coefficient must be chosen to satisfy the following constraints:

1. The artificial viscosity must broaden shocks over several zones according to the local mesh size.
2. The artificial viscosity must be sensitive only to compression.
3. The artificial viscosity must be large in shocks, but negligibly small elsewhere.

With these constraints, and by analogy to the coefficient of the nonlinear von Neumann & Richtmyer (1950) formulation, TW write

$$\mathbf{Q} = \begin{cases} l^2 \rho \nabla \cdot \boldsymbol{v}[\nabla \boldsymbol{v} - \frac{1}{3}(\nabla \cdot \boldsymbol{v})\mathbf{e}] & \text{if} \quad \nabla \cdot \boldsymbol{v} < 0 \\ 0 & \text{otherwise} \end{cases}, \quad (134)$$

where l is a constant with dimensions of length, typically chosen to be a few zone widths. The key advantage to the tensor formulation over the von Neumann & Richtmyer approach is the use of the divergence, an invariant scalar, rather than the velocity gradient, a rank-two tensor which reduces to a single number only in one-dimensional Cartesian geometry.

TW give explicit expressions for **Q** applicable to one-dimensional, spherical geometry. Although the extension to general multidimensional problems is straightforward, we emphasize the following points: (1) In multidimensional problems, the value of l must be chosen to be a single number, rather than being different in each coordinate direction. The latter choice would result in a nonisotropic tensor and would destroy the property $\mathbf{Tr}\,(\mathbf{Q}) = 0$. In practice, we choose l using the maximum grid spacing in either direction. (2) In this work, we drop the off-diagonal (shear) components of the artificial viscosity tensor. We therefore choose not to broaden large gradients of shear, as we desire the artificial viscosity to smooth only compressive shock fronts.

With these considerations, and using the covariant forms for the tensor operators (Appendix A), we can write the zone-centered, nonzero (diagonal) components of the tensor artificial viscosity in finite difference form as

$$Q11_{i,j} = l_{i,j}^2 d_{i,j}(\nabla\cdot\boldsymbol{v})_{i,j}[(\nabla\boldsymbol{v}_{(11)})_{i,j} - \tfrac{1}{3}(\nabla\cdot\boldsymbol{v})_{i,j}]\,, \tag{135}$$

$$Q22_{i,j} = l_{i,j}^2 d_{i,j}(\nabla\cdot\boldsymbol{v})_{i,j}[(\nabla\boldsymbol{v}_{(22)})_{i,j} - \tfrac{1}{3}(\nabla\cdot\boldsymbol{v})_{i,j}]\,, \tag{136}$$

$$Q33_{i,j} = l_{i,j}^2 d_{i,j}(\nabla\cdot\boldsymbol{v})_{i,j}[(\nabla\boldsymbol{v}_{(33)})_{i,j} - \tfrac{1}{3}(\nabla\cdot\boldsymbol{v})_{i,j}]\,, \tag{137}$$

where $l_{i,j} = \max(C_2 dx1a_i, C_2 dx2a_j)$, and $(\nabla\cdot\boldsymbol{v})_{i,j}$, $(\nabla\boldsymbol{v}_{(11)})_{i,j}$, $(\nabla\boldsymbol{v}_{(22)})_{i,j}$, and $(\nabla\boldsymbol{v}_{(33)})_{i,j}$ are all zone-centered scalars, explicit finite difference expressions for which are given in Appendix C. The artificial viscous stress and dissipation terms in the momentum and energy equations are $-\nabla\cdot\mathbf{Q}$ and $-\mathbf{Q}:\nabla\boldsymbol{v}$, respectively (TW), which can be written in the following covariant form,

$$(\nabla\cdot\mathbf{Q})_{(1)} = \frac{1}{g_3^2 g_{31}}\frac{\partial}{\partial x_1}(g_2^2 g_{31} Q_{11})\,, \tag{138}$$

$$(\nabla\cdot\mathbf{Q})_{(2)} = \frac{1}{g_2 g_{32}^2}\frac{\partial}{\partial x_2}(g_{32}^2 Q_{22}) + \frac{Q_{11}}{g_2 g_{32}}\frac{\partial g_{32}}{\partial x_2}\,, \tag{139}$$

$$\mathbf{Q}:\nabla\boldsymbol{v} = l^2\rho\nabla\cdot\boldsymbol{v}\{\tfrac{1}{3}[(\nabla\boldsymbol{v}_{(11)} - \nabla\boldsymbol{v}_{(22)})^2 + (\nabla\boldsymbol{v}_{(11)} - \nabla\boldsymbol{v}_{(33)})^2 + (\nabla\boldsymbol{v}_{(33)} - \nabla\boldsymbol{v}_{(22)})^2]\}\,, \tag{140}$$

where we have used the fact that **Q** is traceless to eliminate Q_{33} in favor of the other two diagonal elements. Note that in Cartesian geometry, these expressions reduce to the von Neumann & Richtmyer (1950) formulation. TW emphasize that representing the viscous energy generation term as a sum of squares as in equation (140) is an absolute necessity; by experience all other formulations will eventually produce instabilities.

In ZEUS-2D, the diagonal components of tensors are zone centered. Thus, equations (138)–(140) can be differenced straightforwardly as

$$(\nabla\cdot\mathbf{Q})_{1,i,j} = \frac{(g2b_i^2 g31b_i Q11_{i,j} - g2b_{i-1}^2 g31b_{i-1} Q11_{i-1,j})}{g2a_i^2 g31a_i}\,, \tag{141}$$

$$(\nabla\cdot\mathbf{Q})_{2,i,j} = \frac{(g32b_j^2 Q22_{i,j} - g32b_{j-1}^2 Q22_{i,j-1})}{g2b_i g32a_j^2} + \frac{(Q11_{i,j} + Q11_{i,j-1})}{2g2b_i g32a_j}\left(\frac{\partial g32a_j}{\partial x_2}\right), \tag{142}$$

$$(\mathbf{Q}:\nabla\boldsymbol{v})_{i,j} = l_{i,j}^2 d_{i,j}(\nabla\cdot\boldsymbol{v})_{i,j}\{\tfrac{1}{3}\{[(\nabla\boldsymbol{v}_{(11)})_{i,j} - (\nabla\boldsymbol{v}_{(22)})_{i,j}]^2 + [(\nabla\boldsymbol{v}_{(11)})_{i,j} - (\nabla\boldsymbol{v}_{(33)})_{i,j}]^2 + [(\nabla\boldsymbol{v}_{(33)})_{i,j} - (\nabla\boldsymbol{v}_{(22)})_{i,j}]^2\}\}\,. \tag{143}$$

These difference equations are solved in place of equations (35)–(37) when the tensor artificial viscosity is used.

Finally, the viscous timestep limit implied by the tensor artificial viscosity is

$$\delta t = \frac{\Delta x^2}{4\nu} = \frac{\Delta x^2}{4l^2\nabla\cdot\boldsymbol{v}}\,. \tag{144}$$

This time-step limit replaces the δt_4 discussed in § 4.6.

APPENDIX C
SOME FINITE DIFFERENCE FORMULAE

In this appendix we give explicit finite difference formula for some vector and tensor operators used in ZEUS-2D:

$$(\nabla\cdot\boldsymbol{v})_{i,j} = \frac{(g2a_{i+1} g31a_{i+1} v1_{i+1,j} - g2a_i g31a_i v1_{i,j})}{dvl1a_i} + \frac{(g32a_{j+1} v2_{i,j+1} - g32a_j v2_{i,j})}{g2b_i dvl2a_j}\,, \tag{145}$$

$$(\nabla\boldsymbol{v}_{(11)})_{i,j} = (v1_{i+1,j} - v1_{i,j})/dx1a_i\,, \tag{146}$$

$$(\nabla \boldsymbol{v}_{(22)})_{i,j} = \frac{(v2_{i,j+1} - v2_{i,j})}{g2b_i\, dx2a_j} + \frac{(v1_{i,j} + v1_{i+1,j})}{2g2b_i}\left(\frac{\partial g2b_i}{\partial x_1}\right), \tag{147}$$

$$(\nabla \boldsymbol{v}_{(33)})_{i,j} = \frac{(v1_{i,j} + v1_{i+1,j})}{2g31b_i}\left(\frac{\partial g31b_i}{\partial x_1}\right) + \frac{(v2_{i,j} + v2_{i,j+1})}{2g2b_i\, g32b_j}\left(\frac{\partial g32b_j}{\partial x_2}\right). \tag{148}$$

REFERENCES

Berger, M. J., & Oliger, J. 1984, J. Comput. Phys., 53, 484
Black, D. C., & Bodenheimer, P. 1975, ApJ, 199, 619
Chandrasekhar, S. 1969, Ellipsoidal Figures of Equilibrium (New Haven: Yale Univ. Press)
Clarke, D. A. 1988, Ph.D. thesis, Univ. New Mexico
Clarke, D. A., Norman, M. L., & Burns, J. O. 1986, ApJ, 311, L63
———. 1989, ApJ, 342, 700
Clarke, D. A., Stone, J. M., & Norman, M. L. 1990, BAAS, 22, 801
Colella, P., & Woodward, P. R. 1984, J. Comput. Phys., 54, 174
Courant, R., & Friedrichs, K. O. 1948, Supersonic Flow and Shock Waves (New York: Springer)
Evans, C. 1986, in Dynamical Spacetimes and Numerical Relativity, ed. J. Centrella (Cambridge: Cambridge Univ. Press), 3
Finn, S., & Hawley, J. F. 1989, unpublished report
Godunov, S. K. 1959, Matematicheskii Sbornik, 47, 271
Hawley, J. F., Smarr, L. L., & Wilson, J. R. 1984a, ApJ, 277, 296
———. 1984b, ApJ, 55, 211
Hunter, C. 1962, ApJ, 136, 594
Jackson, J. D. 1975, Classical Electrodynamics (New York: Wiley)
Lapidus, A. 1967, J. Comput. Phys., 2, 154
Larson, D. J., Hewett, D. W., & Anderson, D. V. 1989, preprint
Mihalas, D., & Mihalas, B. W. 1984, Foundations of Radiation Hydrodynamics (Oxford: Oxford Univ. Press)
Mönchmeyer, R., & Müller, E. 1989, A&A, 217, 351
Norman, M. L. 1980, Ph.D. thesis, Univ. California Davis, LLNL report UCRL-52946
Norman, M. L., Smarr, L. L., Wilson, J. R., & Smith, M. D. 1981, ApJ, 247, 52
Norman, M. L., Smarr, L. L., Winkler, K.-H., & Smith, M. D. 1982, A&A, 113, 285
Norman, M. L., Wilson, J. R., & Barton, R. 1980, ApJ, 239, 968
Norman, M. L., & Winkler, K.-H. 1986, in Astrophysical Radiation Hydrodynamics, ed. K.-H. Winkler & M. L. Norman (Dordrecht: Reidel), 187
Richtmyer, R. D., & Morton, K. W. 1957, Difference Methods for Initial-Value Problems, 2d ed. (New York: Wiley Interscience),
Sod, G. A. 1978, J. Comput. Phys., 27, 1
Stone, J. M. 1990, Ph.D. thesis, Univ. Illinois, Urbana-Champaign
Stone, J. M., Mihalas, D., & Norman, M. L. 1992, ApJS, 80, 819 (Paper III)
Stone, J. M., & Norman, M. L. 1992, ApJS, 80, 791 (Paper II)
Strang, W. G. 1968, SIAM J. Numer. Anal., 5, 506
Thompson, K. W. 1987, J. Comput. Phys., 68, 1
Tscharnuter, W.-M., & Winkler, K.-H. 1979, Comput. Phys. Comm., 18, 171
van Leer, B. 1977, J. Comput. Phys., 23, 276
von Neumann, J., & Richtmyer, R. D. 1950, J. Appl. Phys., 21, 232
Wicker, L. 1990, preprint
Winkler, K.-H., & Norman, M. L. 1986, in Astrophysical Radiation Hydrodynamics, ed. K.-H. Winkler & M. L. Norman (Dordrecht: Reidel), 71
Woodward, P. R. 1986, in Astrophysical Radiation Hydrodynamics, ed. K.-H. Winkler & M. L. Norman (Dordrecht: Reidel), 245
Woodward, P. R., & Colella, P. 1984, J. Comput. Phys., 54, 115