LCA

Laboratory for Computational Astrophysics

The ZEUS-2D Code for Astrophysical Fluid Dynamics Simulations

Michael L. Norman
NCSA
and
Department of Astronomy
University of Illinois

James M. Stone
Department of Astronomy
University of Maryland

LCA Preprint #14
May, 1994

National Center for Supercomputing Applications
Department of Astronomy
University of Illinois at Urbana-Champaign

NCSA Preprint P046

1

The ZEUS-2D Code for Astrophysical Fluid Dynamics Simulations

Michael L. Norman[1]
and James M. Stone[2]

ABSTRACT We describe ZEUS-2D: a two-dimensional Eulerian finite difference code for solving the equations of astrophysical fluid dynamics including the effects of magnetic fields, radiation transport, self-gravity and rotation. The finite difference equations are written in covariant form on an arbitrary moving orthogonal grid which enables the accurate simulation of different flow geometries under translations and changes of scale. In this paper the magnetohydrodynamic algorithms are presented and we describe how to use the code. The radiation hydrodynamic algorithms are presented in a companion paper by Stone. We describe the application of ZEUS-2D to the evolution of a supernova remnant in a strong magnetic field to illustrate several of its features.

1.1 Introduction

Astrophysical fluid dynamics (hereafter AFD) deals with the macroscopic evolution of astrophysical plasmas in a wide variety of circumstances and astronomical contexts. The scope of AFD is broad, encompassing such topics as stellar convection [1], pulsation and winds [2]; supernova explosions [3]; accretion onto compact objects [4,5]; dynamics of the interstellar medium [6]; star formation [7]; winds and jets from active galactic nuclei [8]; and the formation of cosmological large-scale structure [9]. This astonishing variety of phenomena is contained in the rather innocuous-looking equations of fluid dynamics suitably modified to incorporate additional sources and sinks of mass, momentum and energy as the problem dictates.

ZEUS-2D (hereafter ZEUS) is a program for simulating AFD in two space dimensions including the effects of shock waves, magnetic fields, ra-

[1]NCSA and Department of Astronomy, University of Illinois, 1002 W. Green St, Urbana, IL 61801

[2]Department of Astronomy, University of Maryland, College Park, MD 20742

diation transport, rotation and self-gravity. The central design principle of ZEUS is generality, which requires that it be based on robust numerical algorithms which are easily applied to the full range of processes listed above. Consequently, ZEUS is based on the methods of finite differences and finite volumes because these techniques are mature and their properties are quite well understood [10,11]. Furthermore, these methods can be extended straight-forwardly to other physical processes as required. Unlike many codes which are designed to solve a specific problem, ZEUS is designed to be a general-purpose solver which can be configured in different ways for different applications. The code is modular, so that additional physics can be easily incorporated.

covariant form

A powerful feature of ZEUS is that the finite difference equations are written in covariant form on an arbitrary moving orthogonal grid (Cartesian, cylindrical, or spherical polar) which enables the accurate simulation of different flow geometries under translations and changes of scale. ZEUS can be used as a hydrodynamics (HD) code, a magnetohydrodynamics (MHD) code, a radiation hydrodynamics code (RHD), or a radiation magnetohydrodynamics (RMHD) code. The numerical algorithms in ZEUS are detailed in a series of papers: [12],[13] (hereafter SNI & SNII respectively) and [14] (reprints available from the authors). Here we merely summarize the HD and MHD algorithms to illustrate the basic concepts. Our primary purpose in this paper is to describe the ZEUS-2D code included with this book, and how to use the code. The RHD algorithms are discussed in a companion paper by Stone. As an illustration of the MHD algorithms, we apply ZEUS to the evolution of a supernova remnant in a strong magnetic field.

This paper can be read in several ways. For readers who are not interested in the numerical algorithms but only how to use the code, we recommend reading §§1.2, 1.3, 1.7, and 1.8. For those only interested in the algorithms, we recommend reading §§1.2-1.6. And for those who wish to substantially modify and use the code, we recommend reading everything.

1.2 Moving Frame Equations of MHD

ZEUS solves the equations of nonrelativistic MHD including self- gravity, which in their familiar differential form are:

conservation of mass

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{v} = 0, \qquad (1.1)$$

$\frac{\partial}{\partial t} + V\cdot\nabla$

momentum

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p - \rho\nabla\Phi + \frac{1}{c}\mathbf{J}\times\mathbf{B}, \qquad (1.2)$$

energy

$$\rho\frac{D}{Dt}\left(\frac{e}{\rho}\right) = -p\nabla\cdot\mathbf{v}, \qquad (1.3)$$

there is no heat added ~~or lost~~ to the element

ex: $\rho\left\{\frac{De}{Dt} + p\left[\frac{D(\rho^{-1})}{Dt}\right]\right\} = -\nabla q_c$

$\nabla(\vec{v} \times \vec{B})$

magnetic flux

$$\frac{\partial \mathbf{B}}{\partial t} = -c\nabla\times\mathbf{E}, \tag{1.4}$$

where ρ, e and $\mathbf{v}$ are the fluid density, internal energy density and velocity, respectively; $\mathbf{B}$ is the magnetic flux density; and Φ is the gravitational potential. The $\frac{D}{Dt}$ denotes the Lagrangian or convective derivative,

def: ρ = v B Φ E

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla. \tag{1.5}$$

The electric field $\mathbf{E}$ is specified from Ohm's law

$$c\mathbf{E} = -\mathbf{v}\times\mathbf{B} + \mathbf{J}/\sigma. \tag{1.6}$$

1.1 - 1.4 plus 1.6 & 1.7 plus 1.8 check def and simple

where σ is the conductivity of the fluid, and the current $\mathbf{J}$ is given by

$$\frac{4\pi}{c}\mathbf{J} = \nabla\times\mathbf{B}. \tag{1.7}$$

? the Lagrangian

The fluid equations are closed by an equation of state $p = p(\rho, e)$, and the Poisson equation

? the Poisson eq

$$\nabla^2\Phi = 4\pi G\rho. \tag{1.8}$$

Equations 1.1-1.4 describe the conservation of mass, momentum, energy and magnetic flux in a self-gravitating fluid. Any finite difference approximation to the weak form of the conservation laws destroys the conservation properties inherent in the differential equations due to truncation errors. This leads to a loss of accuracy in the best case and to spurious results in the worst. To avoid this, we construct finite-difference approximations to the strong form of the conservation laws, which are derived by integrating equations 1.1-1.4 over an arbitrary control volume and its surface. This control volume corresponds to an individual cell, or zone, in the spatial grid. In ZEUS, we use a moving orthogonal grid, although this approach can be generalized to non-orthogonal coordinate systems.

Consider a moving finite control volume $V(t)$ with surface $S(t)$. Integrating equations 1.1-1.3 over $V(t)$ and utilizing the adaptive grid transport theorem [15], we get

$$\frac{d}{dt}\int_V \rho dV = -\oint_S \rho(\mathbf{v}-\mathbf{v}_g)\cdot d\mathbf{S}, \tag{1.9}$$

check def. and simple exc

$$\frac{d}{dt}\int_V \rho\mathbf{v}dV = -\oint_S \rho\mathbf{v}(\mathbf{v}-\mathbf{v}_g)\cdot d\mathbf{S} - \int_V (\nabla p + \rho\nabla\Phi - \frac{1}{c}\mathbf{J}\times\mathbf{B})dV, \tag{1.10}$$

? the Eulerian fram

$$\frac{d}{dt}\int_V e dV = -\oint_S e(\mathbf{v}-\mathbf{v}_g)\cdot d\mathbf{S} - \int_V p(\nabla\cdot\mathbf{v})dV, \tag{1.11}$$

where $\mathbf{v}_g$ is the grid velocity measured with respect to the Eulerian frame. The $\frac{d}{dt}$ denotes the total time derivative with respect to the moving frame,

v_g

$$\frac{d}{dt} \equiv \frac{\partial}{\partial t} + \mathbf{v}_g\cdot\nabla. \tag{1.12}$$

These equations merely state that the total time rate of change of mass, momentum and energy within the moving volume $V(t)$ is given by the net fluxes of these quantities integrated over the surface plus the volume integral of body forces and other source terms.

Likewise integrating equation 1.4 over a moving surface element $S(t)$ bounded by a moving circuit $C(t)$, one derives the general form of Faraday's law [16]

$$\frac{d}{dt}\int_S \mathbf{B}\cdot d\mathbf{S} = -c\oint_C \mathbf{E}'\cdot d\mathbf{l} \tag{1.13}$$

where $\mathbf{E}'$ is the electric field measured in the moving frame. We must express $\mathbf{E}'$ in terms of the rest frame quantities $\mathbf{E}$, $\mathbf{B}$, and $\mathbf{J}$. Denoting $\mathbf{E}_o$ as the electric field measured in the fluid frame we have, to first order in v/c, $\mathbf{E}' = \mathbf{E} + (\mathbf{v}_g \times \mathbf{B})/c$ and $\mathbf{E}_o = \mathbf{E} + (\mathbf{v}\times\mathbf{B})/c$. The fluid frame current density $\mathbf{J}_o$ is related to the fluid frame electric field through Ohm's law $\mathbf{J}_o = \sigma\mathbf{E}_o$. Using these relations and assuming charge neutrality (so that $\mathbf{J} = \mathbf{J}_o$) and a perfectly conducting ($\sigma = \infty$) fluid, equation 1.13 becomes

$$\frac{d}{dt}\int_S \mathbf{B}\cdot d\mathbf{S} = \oint_C (\mathbf{v}-\mathbf{v}_g)\times\mathbf{B}\cdot d\mathbf{l} \tag{1.14}$$

The grid velocity $\mathbf{v}_g$ is a free function subject to the constraints discussed in §1.3.5.

1.3 ZEUS Essentials

1.3.1 OPERATOR SPLITTING FORMALISM

Explicit multi-step schemes are generally more accurate than a single step that simply extrapolates forward in time on the basis of old data. Examples include the Lagrange-Remap formalism used in the PPM algorithm [17], Flux Corrected Transport [18], Lax-Wendroff [19], and other predictor-corrector algorithms. ZEUS solves the fluid equations 1.9-1.11 and 1.14 in two steps, called the *source* and *transport* steps. In the source step, we accelerate the fluid velocity due to pressure, gravity and Lorentz forces, and perform pressure work on the gas internal energy using finite-difference approximations to

$$\rho\frac{\partial \mathbf{v}}{\partial \mathbf{t}} = -(\nabla p + \rho\nabla\Phi) - \nabla\cdot\mathsf{Q} + \frac{1}{c}\mathbf{J}\times\mathbf{B} \tag{1.15}$$

$$\frac{\partial e}{\partial t} = -p\nabla\cdot\mathbf{v} - \mathsf{Q}:\nabla\mathbf{v}, \tag{1.16}$$

1) viscous stress tensor

2) tensor

where we have dropped the volume integration and introduced the artificial viscous stress tensor Q to treat shock waves. ZEUS implements several choices for Q which have advantages in different applications as described

below. In the transport step, fluid and magnetic flux is transported through the computational mesh conservatively using finite-difference approximations to

compare eqs 1.8–1.11 with {1.15, 1.16 and 1.17–1.20 ~~this~~ is this operator splitting formalism

$$\frac{d}{dt}\int_V \rho dV = -\oint_S \rho(\mathbf{v}-\mathbf{v}_g)\cdot \mathbf{dS}, \tag{1.17}$$

advection equations

$$\frac{d}{dt}\int_V \rho\mathbf{v} dV = -\oint_S \rho\mathbf{v}(\mathbf{v}-\mathbf{v}_g)\cdot \mathbf{dS} \tag{1.18}$$

$$\frac{d}{dt}\int_V e dV = -\oint_S e(\mathbf{v}-\mathbf{v}_g)\cdot \mathbf{dS} \tag{1.19}$$

$$\frac{d}{dt}\int_S \mathbf{B}\cdot \mathbf{dS} = \oint_C (\mathbf{v}-\mathbf{v}_g)\times \mathbf{B}\cdot \mathbf{dl} \tag{1.20}$$

1.3.2 COVARIANT FORMALISM

The finite-difference equations solved by ZEUS are derived by first expressing equations 1.8 and 1.15-1.20 in covariant form for an arbitrary orthogonal coordinate system with metric

? h_i metric coeffici
? S matric /
look eq. 1.22–1.24

$$ds^2 = h_1^2 dx_1^2 + h_2^2 dx_2^2 + h_3^2 dx_3^2 \tag{1.21}$$

and then setting all spatial derivatives in the ignorable x_3 direction to zero, *i.e.* $\partial/\partial x_3 \equiv 0$. Note that while this reduces the number of independent variables to two (x_1, x_2), all components of vectors and tensors are retained. This is sometimes referred to as the $2-1/2$D approximation, and is essential for simulating axisymmetric flows with azimuthal velocities and magnetic fields. Currently, ZEUS implements Cartesian (XY), cylindrical (RZ), and spherical polar (Rθ) geometries, but other orthogonal coordinate systems can easily be added by defining new metric coefficients.

In curvilinear coordinates, additional terms are generated by differential operators acting on the unit vectors. These terms, which appear as "pseudo-forces", are incorporated into the source step in ZEUS.

1.3.3 THE FINITE DIFFERENCE GRID

Subroutines: **ggen, newx1, newx2**

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 (see §1.4.3).

11.3 88.1

Spatial positions on the computational grid are defined using coordinate vectors. The positions of the zone boundaries in each direction is 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 $dx2b_i = x1b_i - x1b_{i-1}$, and similarly for the $dx2a_j$ and $dx2b_j$ (see Figure 2). Independent grid spacing allows for non-uniform 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). The metric scale factors, denoted g1, g2, g31 and g32, are defined on both the a-grid and b-grid (Fig. 2), and are related to the usual metric coefficients as follows:

metric scale factors
g1
g2
g31
g32

$$h_1 = 1 \equiv g_1, \qquad 1 \tag{1.22}$$

$$h_2 = f(x_1) \equiv g_2, \qquad r \tag{1.23}$$

$$h_3 = f(x_1)\, f(x_2) \equiv g_{31}\, g_{32}. \qquad r \sin\theta \tag{1.24}$$

The two-dimensional function h_3 has been expressed as the product of two one-dimensional functions to save computer memory. Without loss of generality, we can drop g_1 from the difference equations since its value is 1 in all geometries of interest. Table 1.1 gives the values for the remaining metric coefficients.

TABLE 1.1. Metric scale factors used in ZEUS-2D

Geometry	x1	x2	x3	g2	g31	g32
Cartesian	x	y	z	1	1	1
cylindrical	z	r	ϕ	1	1	r
spherical polar	r	θ	ϕ	r	r	$\sin\theta$

calculate ds^2 for a few simple examples to better understand a sense of g_i h_i

For some of the covariant equations, derivatives of the metric scale factors are also needed. For convenience, their values are given in Table 1.2.

TABLE 1.2. Metric coefficient derivatives and volume factors

Geometry	$\partial g_2/\partial x_1$	$\partial g_{31}/\partial x_1$	$\partial g_{32}/\partial x_1$	$dvl1$	$dvl2$
Cartesian	0	0	0	Δx	Δy
cylindrical	0	0	1	Δz	$\Delta(r^2/2)$
spherical polar	1	1	$\cos\theta$	$\Delta(r^3/3)$	$\Delta(-\cos\theta)$

Finally, in order to reduce discretization errors near coordinate singularities, we difference with respect to coordinate volume rather than coordinate distance (see SNI for more details). Thus, for example, we difference as follows: [12]

$$\frac{1}{h_2}\frac{\partial F}{h_3\,\partial x_1} \longrightarrow \frac{1}{g_{32}}\frac{\Delta F}{dvl1}, \qquad \frac{1}{h_3}\frac{\partial F}{\partial x_2} \longrightarrow \frac{1}{g_{31}}\frac{\Delta F}{dvl2}. \tag{1.25}$$

look at [12] (SNI) and try to understand why is so

where the volume factor variables are defined in Table 1.2.

1.3.4 CENTERING OF VARIABLES

Discrete values for each dependent variable are stored for each zone. ZEUS uses a staggered mesh so that scalars and the 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 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 Figure 3),

$$\begin{aligned}
\rho(x_1,x_2) &\longrightarrow d(x1b_i,x2b_j) = d_{i,j},\\
e(x_1,x_2) &\longrightarrow e(x1b_i,x2b_j) = e_{i,j},\\
v_1(x_1,x_2) &\longrightarrow v1(x1a_i,x2b_j) = v1_{i,j},\\
v_2(x_1,x_2) &\longrightarrow v2(x1b_i,x2a_j) = v2_{i,j},\\
v_3(x_1,x_2) &\longrightarrow v3(x1b_i,x2b_j) = v3_{i,j},\\
B_1(x_1,x_2) &\longrightarrow B1(x1a_i,x2b_j) = B1_{i,j},\\
B_2(x_1,x_2) &\longrightarrow B2(x1b_i,x2a_j) = B2_{i,j},\\
B_3(x_1,x_2) &\longrightarrow B3(x1b_i,x2b_j) = B3_{i,j},\\
\Phi(x_1,x_2) &\longrightarrow \Phi(x1b_i,x2b_j) = \Phi_{i,j}.
\end{aligned}$$

The location of the derived current density is also shown in Fig. 3 as dictated by straight-forward finite differencing of $\nabla\times\mathbf{B}$. 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.

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. Secondly, a staggered mesh reduces the number of interpolations needed for solving the

advection equations 1.17-1.19 in the transport step. Thus, the velocities, when centered on zone interfaces, naturally describe the flux of fluid into or out of a zone. The centering of the MHD field variables also leads to compact difference kernels, as we shall see below. 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.

1.3.5 THE GRID VELOCITY

Subroutines: **newgrid, newvg**

In ZEUS, the grid is constrained to remain orthogonal, but otherwise arbitrary. This implies that each component of the grid velocity $\mathbf{v}_g$ is a function of one independent variable only; i.e.,

$$v_{g,1}(x_1, x_2) \equiv v_{g,1}(x_1) \longrightarrow vg1(x1a_i) \equiv vg1_i \tag{1.26}$$

$$v_{g,2}(x_1, x_2) \equiv v_{g,2}(x_2) \longrightarrow vg2(x2a_j) \equiv vg2_j \tag{1.27}$$

The two 1-D arrays $vg1_i$ and $vg2_j$ control how the $x1a_i$ and $x2a_j$ grid lines move, respectively, and the user is free to specify them independently to suit the problem of interest. For example, setting $vg1_i = vg2_j = 0$ gives a fixed Eulerian grid. Setting $vg1_i \propto x1a_i$ and $vg2_j \propto x2a_j$ yields a homologously expanding or contracting grid, useful for simulating explosions or gravitational collapse. Setting $vg1_i$ or $vg2_j$ to a constant yields a uniformly translating grid. If ZEUS is used in 1D mode, setting $vg1_i = v1_i$ yields a Lagrangian grid. In 2D, a "pseudo- Lagrangian" grid can be defined for each axis as follows:

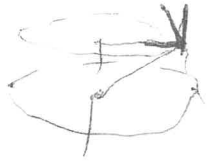
$$vg1_i = \frac{1}{jo - ji} \sum_{j=ji}^{jo} v1_{i,j} \tag{1.28}$$

$$vg2_i = \frac{1}{io - ii} \sum_{i=ii}^{io} v2_{i,j} \tag{1.29}$$

Together with the freedom to choose coordinate systems, the freedom to specify the grid velocity largely obviates the need for general adaptive coordinate systems, although a few extremely challenging applications still require them [7].

1.4 Hydrodynamic Algorithms

ZEUS can be configured to run as a pure hydrodynamics code (see §1.8). The hydrodynamics algorithms form the foundation of ZEUS to which other physics modules are added (MHD, RHD). Therefore, we summarize these



algorithms first. A complete listing of the difference equations are contained in SNI.

1.4.1 SOURCE STEP

Subroutines: **srcstep, pgas, stv1, stv2, viscus, tav, pdv**

In the source step we solve finite-difference approximations to the differential equations 1.15-1.16 suitably expressed in covariant form. The location of the quantities on the grid (Fig. 3) allow for simple second order-accurate centered differences and averages of the terms appearing on the right hand side. The source step itself is operator split into three substeps which improve the overall accuracy of the time integration (see Fig. 4). 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 apply the compressional heating term to the gas energy. 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 for each of these substeps are as follows.

n = start of timestep
n + a } update v due to P
n + b } substeps sbeoo + diss
n + c } compressional heating

substep 1: (physical accelerations and curvature terms)

$$\begin{aligned}\frac{v1^{n+a}_{i,j} - v1^{n}_{i,j}}{\Delta t} &= -\frac{p^n_{i,j} - p^n_{i-1,j}}{dx1b_i\ (d^n_{i,j} + d^n_{i-1,j})/2} - \frac{\Phi^n_{i,j} - \Phi^n_{i-1,j}}{dx1b_i} \\ &+ \frac{[(v2^n_{i,j} + v2^n_{i,j+1} + v2^n_{i-1,j} + v2^n_{i-1,j+1})/4]^2}{g2a_i}\left(\frac{\partial g2a_i}{\partial x_1}\right) \\ &+ \frac{[(v3^n_{i,j} + v3^n_{i-1,j})/2]^2}{g31a_i}\left(\frac{\partial g31a_i}{\partial x_1}\right), \end{aligned} \tag{1.30}$$

fourth term
$v3^n_{i,j} + v3^n_{i-1,j}$ ≠ $v3^n_{i,j}$ (at x1a(i,j))

$$\begin{aligned}\frac{v2^{n+a}_{i,j} - v2^{n}_{i,j}}{\Delta t} &= -\frac{p^n_{i,j} - p^n_{i,j-1}}{g2b_i\ dx2b_j\ (d^n_{i,j} + d^n_{i,j-1})/2} - \frac{\Phi^n_{i,j} - \Phi^n_{i,j-1}}{g2b_i\ dx2b_j} \\ &+ \frac{[(v3^n_{i,j} + v3^n_{i,j-1})/2]^2}{g2b_i\ g32a_j}\left(\frac{\partial g32a_j}{\partial x_2}\right). \end{aligned} \tag{1.31}$$

The pressure $p^n_{i,j}$ is computed at the beginning of the source step using the equation of state $p = p(\rho, e)$. Most commonly, we use an ideal gas law, so that $p^n_{i,j} = (\gamma - 1)e^n_{i,j}$, although more general equations of state can be added. The gravitational potential $\Phi^n_{i,j}$ is computed at the beginning of the source step from a solution of the Poisson equation 1.8 using the density $d^n_{i,j}$.

substep 2: (artificial viscous acceleration and heating)

ZEUS implements both a von Neumann and Richtmyer-like artificial viscosity and a tensor artificial viscosity to treat shocks. The former is simpler and works well in most circumstances. Thus, assuming the viscous stress tensor is diagonal, we solve

$\frac{\partial v}{\partial t} = -(\nabla Q)/\rho$

$$\frac{v1_{i,j}^{n+b} - v1_{i,j}^{n+a}}{\Delta t} = -\frac{q1_{i,j} - q1_{i-1,j}}{dx1b_i\ (d_{i,j}^n + d_{i-1,j}^n)/2}, \tag{1.32}$$

$\frac{g\,cm^2}{s^2}\frac{1}{cm^3}\frac{cm^3}{g\,cm} = \frac{cm}{s^2}$

$$\frac{v2_{i,j}^{n+b} - v2_{i,j}^{n+a}}{\Delta t} = -\frac{q2_{i,j} - q2_{i,j-1}}{g2b_i\ dx2b_j\ (d_{i,j}^n + d_{i,j-1}^n)/2}, \tag{1.33}$$

$\frac{\partial e}{\partial t} = -Q:\nabla v$

$$\frac{e_{i,j}^{n+b} - e_{i,j}^n}{\Delta t} = -q1_{i,j}\left(\frac{v1_{i+1,j}^{n+a} - v1_{i,j}^{n+a}}{dx1a_i}\right) - q2_{i,j}\left(\frac{v2_{i,j+1}^{n+a} - v2_{i,j}^{n+a}}{g2b_i\ dx2a_j}\right). \tag{1.34}$$

$\frac{g\,cm^2}{s^2}\frac{1}{cm^3}\cdot\frac{cm}{s}\frac{1}{cm}$

where

$$q1_{i,j} = \begin{cases} C_2 d_{i,j}(v1_{i+1,j}^{n+a} - v1_{i,j}^{n+a})^2 & \text{if } (v1_{i+1,j}^{n+a} - v1_{i,j}^{n+a}) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{1.35}$$

$\frac{g\,cm^2}{s^2}\frac{1}{cm^3}$

$$q2_{i,j} = \begin{cases} C_2 d_{i,j}(v2_{i,j+1}^{n+a} - v2_{i,j}^{n+a})^2 & \text{if } (v2_{i,j+1}^{n+a} - v2_{i,j}^{n+a}) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{1.36}$$

C_2 measures the number of zones over which the artificial viscosity will spread a shock, and is typically chosen to be $C_2 \approx 3$. For more demanding applications, such as gravitational collapse, the tensor formulation implemented in the subroutine `tav` is used. For more discussion of this, see SNI, Appendix B. The method by which the user chooses between the von Neumann and Richtmyer-like or tensor artificial viscosity is discussed in 1.7.3.

tav

substep 3: (apply pdV work)

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,

$$(e^{n+1} - e^n)/(\Delta t) = -p^{n+1/2}\nabla\cdot\mathbf{v}, \tag{1.37}$$

$e^{n+1} = -p^{n+1/2}\nabla v\Delta t + e^n$

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

$p = (\gamma - 1)e$

new

gamma law solver.

$$e_{i,j}^{n+c} = \left[\frac{1 - (\Delta t/2)(\gamma - 1)(\nabla\cdot\mathbf{v})_{i,j}}{1 + (\Delta t/2)(\gamma - 1)(\nabla\cdot\mathbf{v})_{i,j}}\right] e_{i,j}^{n+b}. \tag{1.38}$$

Note that $(\nabla\cdot\mathbf{v})_{i,j}$ is a zone centered scalar, an explicit finite-difference expression for which is given in SNI, Appendix C. This procedure can be

e^{n+c} - end of substep c.

e^{n+b} end of substep b.

used only for a gamma-law gas. For a more general equation of state, a predictor- corrector method is used

$$(e_{i,j}^{pred} - e_{i,j}^{n+b})/(\Delta t) = -p_{i,j}^{n}(\nabla\cdot\mathbf{v})_{i,j}, \tag{1.39}$$

$$(e_{i,j}^{n+c} - e_{i,j}^{n+b})/(\Delta t) = -\frac{1}{2}(p_{i,j}^{n} + p_{i,j}^{pred})(\nabla\cdot\mathbf{v})_{i,j}, \tag{1.40}$$

where p^{pred} is computed from the equation of state using e^{pred}.

1.4.2 TRANSPORT STEP

Subroutines: **transprt, tranx1, tranx2, momx1, momx2, vtos, stov, x1intzc, x1intfc, x2intzc, x2intfc**

In the transport step, we solve finite-difference approximations to the integral advection equations 1.17-1.19. A finite-difference approximation to these equations which is second order-accurate in space and time can be written schematically as

$$(q_{i,j}^{n+1}\ \tau_{i,j}^{n+1} - q_{i,j}^{n}\ \tau_{i,j}^{n})/(\Delta t) = -(\mathcal{F}_{i+1,j}^{1} - \mathcal{F}_{i,j}^{1} + \mathcal{F}_{i,j+1}^{2} - \mathcal{F}_{i,j}^{2})^{n+1/2}, \tag{1.41}$$

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

$$(\mathcal{F}_{i,j}^{k})^{n+1/2} = \langle (vk_{i,j} - vgk_i)\ q_{k,i,j}^{*}\ \tilde{A}_{k,i,j}\rangle, \tag{1.42}$$

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 angled brackets ("$\langle\ \rangle$") denote all quantities on the RHS are time centered. Note if q is zone centered, as are d, e, $v3$ and $b3$, then due to the staggered mesh, the relative velocities are naturally centered at the faces of the control volume. The control volumes for the face-centered quantities $v1$ and $v2$ are offset (cf. SNI, Fig. 5), therefore the relative velocity must be averaged to the faces. Exact conservation of the total quantity of the variable q 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 1.41 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 multi-dimensional 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 [20]. Thus, we use two

one-dimensional updates, starting with a sweep in the 1- direction,

$$\frac{q_{i,j}^{n+a}\tau_{i,j}^{n+a} - q_{i,j}^{n}\tau_{i,j}^{n}}{\Delta t} = -\left[\langle(v1_{i+1,j} - vg1_{i+1})q_{1,i+1,j}^{*}\tilde{A}_{1,i+1,j}\rangle - \langle(v1_{i,j} - vg1_{i})q_{1,i,j}^{*}\tilde{A}_{1,i,j}\rangle\right], \tag{1.43}$$

followed by a sweep in the 2-direction,

$$\frac{q_{i,j}^{n+1}\tau_{i,j}^{n+1} - q_{i,j}^{n+a}\tau_{i,j}^{n+a}}{\Delta t} = -\left[\langle(v2_{i,j+1} - vg2_{j+1})q_{2,i,j+1}^{*}\tilde{A}_{2,i,j+1}\rangle - \langle(v2_{i,j} - vg2_{j})q_{2,i,j}^{*}\tilde{A}_{2,i,j}\rangle\right]. \tag{1.44}$$

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. Therefore we alternate the sweep directions from one timestep to the next. The accuracy and stability of Eulerian advection hinges on how the quantity $\langle q^{*}\rangle$ is computed in equations 1.43-1.44.

Years of research by many have shown that stability is achieved through upwind monotonic interpolation of q whereas accuracy is achieved through higher order-accurate interpolation functions. In ZEUS, we have implemented three schemes for monotonic interpolation, the first order-accurate donor cell (DC) method, the second order-accurate van Leer (VL) method [21], and the third order-accurate piecewise parabolic advection (PPA) portion of the PPM method for gas dynamics [17]. These interpolation formulae are given in SNI, and are not repeated here. The user can switch between these interpolation schemes by setting an input parameter in the namelist **hycon** (see §1.8.5). We comment that DC advection should never be used because it is too diffusive; it is retained for pedagogical purposes only. PPA advection should be used judiciously because it is computationally expensive, although it yields the best results.

Regardless of the interpolation scheme used, time-centering of the advected quantity q is achieved by averaging over the domain of dependence traversed by fluid streamline from time level n to $n+1$. Thus, letting $q_{mono,i}(x)$ be the monotonic interpolation function for the distribution of q within zone i at time level n, and x the relative position within a zone, then

$$\langle q^{*}\rangle = \begin{cases} \frac{1}{\Delta\xi_i}\int_{1-\Delta\xi_i}^{1} q_{mono,i-1}(\xi')d\xi', & \Delta\xi_i \geq 0 \\ \frac{1}{|\Delta\xi_i|}\int_{0}^{|\Delta\xi_i|} q_{mono,i}(\xi')d\xi', & \Delta\xi_i < 0 \end{cases} \tag{1.45}$$

where

$$\Delta\xi_i = \begin{cases} (v_i - vg_i)\Delta t/\Delta x_{i-1}, & (v_i - vg_i) \geq 0 \\ (v_i - vg_i)\Delta t/\Delta x_i, & (v_i - vg_i) < 0 \end{cases} \tag{1.46}$$

Given $\langle q^{*}\rangle$, we can construct time centered fluxes and perform the advection using equations 1.43 and 1.44.

$\omega r^2 = const \quad \frac{r_1}{r} = 0.5$?

$\omega_1 r_1^2 = \omega r^2$

$\frac{\omega_1}{\omega} = \left(\frac{r}{r_1}\right)^2$

$\omega_1 = \left(\frac{r}{r_1}\right)^2 \omega \qquad \omega_1 = 4\omega$

$\omega r_1 = 4\omega\, 0.5 r$

We illustrate this for the mass density advection equation. Using directional splitting, the flux of density in the 1-direction is

$$\mathcal{F}^1_{i,j} = d^*_{1,i,j}\ (v1_{i,j} - vg1_i)\ g2a_i^{n+1/2}\ g31a_i^{n+1/2}\ dvl2a_j^n, \tag{1.47}$$

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

$$(d_{i,j}^{n+1}\ dvl1a_i^{n+1}\ dvl2a_j^n - d_{i,j}^n\ dvl1a_i^n\ dvl2a_j^n)/(\Delta t) = -(\mathcal{F}^1_{i+1,j} - \mathcal{F}^1_{i,j}), \tag{1.48}$$

where $dvl1a_i^n$ and $dvl2a_j^n$ are the volume factor variables discussed in §1.3.3, 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

$$\mathcal{F}^2_{i,j} = d^*_{2,i,j}\ (v2_{i,j} - vg2_j)\ g31b_i^n\ dx1a_i^n\ g32a_j^{n+1/2}, \tag{1.49}$$

$$(d_{i,j}^{n+1}\ dvl1a_i^n\ dvl2a_j^{n+1} - d_{i,j}^n\ dvl1a_i^n\ dvl2a_j^n)/(\Delta t) = -(\mathcal{F}^2_{i,j+1} - \mathcal{F}^2_{i,j}), \tag{1.50}$$

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.

Advection of the other HD variables (e, $v1$, $v2$, and $v3$) proceeds in a similar manner. However, there are two important differences. The first is the use of consistent transport, a notion first introduced by [22]. The second is the use of momentum control volumes which are centered on their respective components (cf. SNI, Fig. 5). Consistent transport attempts to minimize the effect of numerical diffusion on local conservation by tying the fluxes of e, $v1$, $v2$, and $v3$ to the flux of mass, rather than treating them as independent density fields. As demonstrated in [22] this can make a dramatic improvement to the local conservation of angular momentum in rotating gravitational collapse calculations. 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.

momentum
angular momentum
$\vec{r} \times \vec{v}$

Thus, with the face-centered mass flux in the 1-direction defined via

$$\dot{M}^1_{i,j} = d^*_{1,i,j}\ (v1_{i,j} - vg1_i), \tag{1.51}$$

the fluxes of the other variables are given by

$$\mathcal{F}^1_{i,j} = (e/d)^*_{1,i,j}\ \dot{M}^1_{i,j}\ g2a_i^{n+1/2}\ g31a_i^{n+1/2}\ dvl2a_j^n, \tag{1.52}$$

$$\mathcal{G}^1_{i,j} = v1^*_{1,i,j}\ \frac{1}{2}(\dot{M}^1_{i,j} + \dot{M}^1_{i+1,j})\ g2b_i^{n+1/2}\ g31b_i^{n+1/2}\ dvl2a_j^n, \tag{1.53}$$

$$\mathcal{H}^1_{i,j} = v2^*_{1,i,j}\ g2a_i\ \frac{1}{2}(\dot{M}^1_{i,j} + \dot{M}^1_{i,j-1})\ g2a_i^{n+1/2}\ g31a_i^{n+1/2}\ dvl2b_j^n, \tag{1.54}$$

$$\mathcal{I}^1_{i,j} = v3^*_{1,i,j}\ g31a_i\ g32b_j\ \dot{M}^1_{i,j}\ g2a_i^{n+1/2}\ g31a_i^{n+1/2}\ dvl2a_j^n, \tag{1.55}$$

where the quantities with asterix are interpolated as described above. These fluxes are used to update e, s1, s2, and s3 due to advection in the 1-direction according to,

$$(e^{n+1}_{i,j}\ dvl1a_i^{n+1}\ dvl2a_j^n - e^n_{i,j}\ dvl1a_i^n\ dvl2a_j^n)/(\triangle t) = -(\mathcal{F}^1_{i+1,j} - \mathcal{F}^1_{i,j}), \tag{1.56}$$

$$(s1^{n+1}_{i,j}\ dvl1b_i^{n+1}\ dvl2a_j^n - s1^n_{i,j}\ dvl1b_i^n\ dvl2a_j^n)/(\triangle t) = -(\mathcal{G}^1_{i,j} - \mathcal{G}^1_{i-1,j}), \tag{1.57}$$

$$(s2^{n+1}_{i,j}\ dvl1a_i^{n+1}\ dvl2b_j^n - s2^n_{i,j}\ dvl1a_i^n\ dvl2b_j^n)/(\triangle t) = -(\mathcal{H}^1_{i+1,j} - \mathcal{H}^1_{i,j}), \tag{1.58}$$

$$(s3^{n+1}_{i,j}\ dvl1a_i^{n+1}\ dvl2a_j^n - s3^n_{i,j}\ dvl1a_i^n\ dvl2a_j^n)/(\triangle t) = -(\mathcal{I}^1_{i+1,j} - \mathcal{I}^1_{i,j}). \tag{1.59}$$

Advection in the 2-direction proceeds in an entirely analogous way, and the formulae can be found in SNI.

1.4.3 BOUNDARY CONDITIONS

Subroutines: **bval{d,e,v1,v2,v3}**

Boundary conditions are applied by setting the values in the two rows of ghost zones along each of the four boundaries of the computational domain (Fig. 1). These boundaries are referred to as the "inner i boundary" (iib); "outer i boundary" (oib); "inner j boundary" (ijb); and "outer j boundary" (ojb). The values of the primary dependent variables are set according to the boundary condition type, of which there are five implemented in ZEUS. In practice, one sets the value of the integer arrays $niib_j$, $noib_j$, $nijb_i$, $nojb_i$ to one of $n = -1, 1, 2, 3, 4$ with the following effect:

1. **n=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 amongst the active zones. The normal component of velocity is set to zero on the boundary, and reflected for the second ghost zone.

2. **n=-1, 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. **n=2, Outflow Boundary Condition:** This is the most difficult boundary condition to implement properly. In ZEUS, 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 may 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 increasing the accuracy of the boundary condition (*e.g.*, by extrapolating along characteristics rather than gridlines, [23]).

4. **n=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.

5. **n=4, 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 should be computed from the difference equations.

1.4.4 TIMESTEP CONTROL

Subroutines: **nudt**

Since ZEUS is an explicit code, the timestep must satisfy the Courant-Friedrich-Lewy (CFL) stability and accuracy condition,

$$\Delta t \leq \min(\Delta x)/(|u| + C_a), \tag{1.60}$$

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. In addition, the use of artificial viscosity introduces a second timestep constraint for the time- explicit diffusion of momentum, which is

$$\Delta t \leq (\Delta x)^2/4\nu, \tag{1.61}$$

where ν is the effective coefficient of kinematic viscosity. The expression for ν can be determined by comparing the artificial viscosity terms to those in the Navier-Stokes equations, thus $\nu = l^2 \nabla \cdot \mathbf{v}$ where l is a constant with dimensions of length which describes the strength of the artificial viscosity.

Both of these constraints are conveniently implemented in ZEUS using

$$\Delta t = C_o/[\max(\delta t_1^{-2} + \delta t_2^{-2} + \delta t_3^{-2} + \delta t_4^{-2}]^{1/2}. \tag{1.62}$$

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

$$\delta t_1 = [\min(\Delta x_1, \Delta x_2)]/C_a, \tag{1.63}$$

$$\delta t_2 = \Delta x_1/(v_1 - vg_1), \qquad \delta t_3 = \Delta x_2/(v_2 - vg_2). \tag{1.64}$$

$$\delta t_4 = \min\left(\frac{(\Delta x_1)^2}{4l^2(\Delta v_1/\Delta x_1)}, \frac{(\Delta x_2)^2}{4l^2(\Delta v_2/\Delta x_2)}\right) = \min\left(\frac{\Delta x_1}{4C_2\Delta v_1}, \frac{\Delta x_2}{4C_2\Delta v_2}\right), \tag{1.65}$$

where C_2 is the dimensionless coefficient of artificial viscosity defined through $C_2 = l/\Delta x$ (see §1.4.1). In addition, the timestep is not allowed to increase by more than 25% per timestep.

1.5 Magnetohydrodynamic Algorithms

1.5.1 FUNDAMENTAL DIFFICULTIES AND APPROACH

There are two principal difficulties associated with the numerical solution of the MHD equations as compared to the HD equations. The first is that Maxwell's equations require the magnetic field satisfy the divergence-free constraint ($\nabla\cdot\mathbf{B} = 0$) at all times. Unless this constraint is built into the numerical evolution equations, accumulation of discretization errors can ultimately lead to gross violation of the constraint equation, and this will force termination of the calculation. This difficulty is overcome in ZEUS through the use of the constrained transport (CT) method of Evans and Hawley [EH,24]. EH showed that solving the magnetic induction equation in integral form (eq. 1.20) conserves magnetic flux to machine round-off, provided the electromotive force (EMF) $\varepsilon = (\mathbf{v} - \mathbf{v}_g) \times \mathbf{B}$ is evaluated only once over the whole grid and then used to update all field components in a single step. This means that if the magnetic field is divergence free initially, it will remain so.

To demonstrate this, consider the discrete representation of eq. 1.20 for the two poloidal field components (B_1, B_2) solved in ZEUS.

$$\begin{aligned}(\Phi 1^{n+1}_{i,j} - \Phi 1^{n}_{i,j})/\Delta t &= -\varepsilon 2_{i,j}\Delta x2 - \varepsilon 3_{i,j}\Delta x3 + \varepsilon 2_{i,j}\Delta x2 + \varepsilon 3_{i,j+1}\Delta x3 \\ &= +(\varepsilon 3_{i,j+1} - \varepsilon 3_{i,j})\Delta x3, \end{aligned} \tag{1.66}$$

$$\begin{aligned}(\Phi 2^{n+1}_{i,j} - \Phi 2^{n}_{i,j})/\Delta t &= +\varepsilon 1_{i,j}\Delta x1 - \varepsilon 3_{i+1,j}\Delta x3 - \varepsilon 1_{i,j}\Delta x1 + \varepsilon 3_{i,j}\Delta x3 \\ &= -(\varepsilon 3_{i+1,j} - \varepsilon 3_{i,j})\Delta x3. \end{aligned} \tag{1.67}$$

Here $\Phi 1$ and $\Phi 2$ are the magnetic flux components piercing the faces of the control volume created by extending a zone in the x_3 direction (Fig.

5), and $\varepsilon 1$, $\varepsilon 2$, and $\varepsilon 3$ are the edge-centered EMF's. The arrows in Fig. 5 denote the correct direction for the contour integration resulting in equations 1.66 and 1.67. Note that for two-dimensional calculations, the 1- and 2-components of the EMF cancel from the evolution equations by symmetry. By adding equations 1.66 and 1.67 together, we can calculate the new total flux through the surfaces of the control volume. Since the contribution from the EMF along each edge appears once in each equation with opposite signs, all EMF terms will cancel in the full sum for the total flux. Note also that the toroidal component of the magnetic field is automatically divergence-free by symmetry. Thus, we see that if the old flux obeyed the constraint equation (*i.e.* $\int_S \Phi \cdot dS = 0$ by Gauss' Law) then the new flux will continue to do so. This will be true regardless of how the EMF's are evaluated. In fact, the CT scheme does not specify how the EMFs are to be computed. This situation is completely analogous to solving eqs. 1.17-1.19 in conservative form, which insures global conservation through the cancellation of fluxes regardless of how the fluxes are computed. The key to local accuracy and stability, however, is the method by which the EMF's are calculated (see below).

This brings us to the second difficulty of numerical MHD, namely the stable and accurate propagation of all MHD wave modes. These are of two types: (1) longitudinal, compressive (fast and slow magnetosonic); and (2) transverse, non-compressive (Alfvén waves). The existence of these two types of wave modes can be displayed by expanding the Lorentz force using the vector identity:

$$(\nabla \times \mathbf{B}) \times \mathbf{B} = -\nabla(B^2/8\pi) + \mathbf{B} \cdot \nabla \mathbf{B} \tag{1.68}$$

4π

The equation of motion for the i-th velocity component, ignoring other forces, is then

$$\rho \frac{\partial v_i}{\partial t} = -\nabla_i (B^2/8\pi) + \frac{1}{4\pi}(\mathbf{B} \cdot \nabla) B_i \tag{1.69}$$

The first term on the RHS is the gradient of the magnetic pressure, and is clearly a longitudinal force. This term is handled in ZEUS using centered-space, forward-time (CSFT) difference algorithms (§1.5.2). The second term on the RHS represents magnetic tension in curved field lines, and is transverse to the gradient of $\mathbf{B}$. This term couples to the magnetic induction equation to produce Alfvén waves, which is where the real difficulty arises.

In ideal MHD, Alfvén waves can exhibit discontinuities (rotational, transverse) at current sheets. Unlike hydrodynamical shocks, these structures are not dissipative, which rules out the use of dissipative numerical algorithms to model them. In addition, Alfvén waves tightly couple the evolution equations for the velocity and magnetic field components perpendicular to the direction of propagation. This rules out operator splitting these components. Finally, we need an algorithm that can be combined with CT to give

both divergence-free transport of fields and correct local dynamics. This will be achieved if the EMFs used in the CT scheme contain information about all wave modes, which for stability, must be appropriately upwinded.

These multiple requirements can be met using the Method of Characteristics (MOC) to compute the EMFs (SNI, [25]). Ordinarily, MOC is applied to the full set of dynamical equations, and characteristic difference equations are derived which govern the propagation of all wave modes [26] along their respective characteristic hypersurfaces. Tangential discontinuities propagate along Alfvén wave characteristics and are treated exactly in MOC. Unfortunately, traditional MOC is not conservative. We circumvent this problem by using MOC only to evaluate the EMFs used in CT. Schematically, the method can be written as

$$\varepsilon 3_{i,j}^{n+1/2} = v1_{i,j}^{*,n+1/2} B2_{i,j}^{*,n+1/2} - v2_{i,j}^{*,n+1/2} B1_{i,j}^{*,n+1/2} \tag{1.70}$$

where $\mathbf{v}_{i,j}^{*,n+1/2}$ and $\mathbf{B}_{i,j}^{*,n+1/2}$ represent time centered values for these variables resulting from the solution of the characteristic equations at the zone corners (in 2D) where $\varepsilon 3$ is located. A second element of our approach is that we apply MOC to the Alfvén waves only, as the longitudinal modes are adequately handled in a previous step by finite-difference methods.

To illustrate our method, we begin by writing down the magnetic induction equation for the i-th component B_i in differential form:

$$\frac{\partial B_i}{\partial t} = (\mathbf{B} \cdot \nabla) v_i - \nabla \cdot (B_i \mathbf{v}) \tag{1.71}$$

In 1-D, equations 1.69 and 1.71 reduce simply to

$$\frac{\partial v}{\partial t} = \frac{B_x}{\rho} \frac{\partial B}{\partial x} - \frac{\partial}{\partial x}(v_x v), \tag{1.72}$$

$$\frac{\partial B}{\partial t} = B_x \frac{\partial v}{\partial x} - \frac{\partial}{\partial x}(v_x B), \tag{1.73}$$

where we have used the divergence-free constraint in one dimension (which implies $\partial B_x / \partial x \equiv 0$) and the non-compressive nature of Alfvén waves which implies $\partial v_x / \partial x \equiv 0$. We can rewrite the coupled equations 1.72 and 1.73 in characteristic form by multiplying equation 1.73 by $\rho^{-1/2}$ and then adding and subtracting them, yielding

$$\frac{Dv}{Dt} \pm \frac{1}{\rho^{1/2}} \frac{D\ B}{Dt} = 0. \tag{1.74}$$

The plus sign denotes the characteristic equation along the forward facing characteristic C^+, while the minus sign denotes the characteristic equation along the backward facing characteristic C^-. The comoving derivative used in equation 1.74 is defined as

$$D/Dt = \partial/\partial t + (v_x \mp B_x/\rho^{1/2}) \partial/\partial x, \tag{1.75}$$

where the minus (plus) sign is taken for the comoving derivative along the C^+ (C^-) characteristic. Note that the coefficient of the second term in the comoving derivative is just the Alfvén velocity in the moving fluid, $v_x \pm v_A$. Physically, equations 1.74 state that along characteristics, which are straight lines in spacetime with slopes of $v_x \pm v_A$, the changes in the velocity and magnetic field components in each direction are not independent.

The finite-difference equations used to solve the characteristic equations 1.74 can be generated as follows. Consider the one dimensional space-time diagram centered at the positions of the EMF, where we require the partially updated values $v_i^{*,n+1/2}$ and $B_i^{*,n+1/2}$ (see Figure 6). Extrapolating back in time along the characteristics C^+ and C^- to time level n defines the "footpoints". By using upwind interpolation at time level n (*e.g.* donor cell, van Leer or PPA), we can compute the time averaged values for these variables in each domain of dependence. For both the velocities and the magnetic fields the characteristic speed $v_x \pm v_A$ are used to compute upwind values. Given values for the variables at the footpoints, we difference the characteristic equations along C^+ or C^- as

$$(v_i^{*,n+1/2} - v_i^{+,n}) + (B_i^{*,n+1/2} - B_i^{+,n})/(\rho_i^+)^{1/2} = 0. \tag{1.76}$$

$$(v_i^{*,n+1/2} - v_i^{-,n}) - (B_i^{*,n+1/2} - B_i^{-,n})/(\rho_i^-)^{1/2} = 0, \tag{1.77}$$

In order to approximately time center the denominator of the second term in each of these equations, we set $\rho_i^+ = \rho_{i-1}^n$ and $\rho_i^- = \rho_i^n$. These two linear equations for the two unknowns $v_i^{*,n+1/2}$ and $B_i^{*,n+1/2}$ are then solved directly.

For our multidimensional calculations, we solve the characteristic equations in a directionally split fashion. In order to evaluate the four starred quantities appearing in eq. 1.70, we first solve at every grid point the pair of characteristic equations which represent the propagation of Alfvén waves in the 2-direction, namely

$$\frac{Dv_1}{Dt} \pm \frac{1}{\rho^{1/2}} \frac{D\ B_1}{Dt} = 0, \tag{1.78}$$

where

$$D/Dt = \partial/\partial t + (v_2 \mp B_2/\rho^{1/2})/h_2\ \partial/\partial x_2. \tag{1.79}$$

This yields $v1_{i,j}^{*,n+1/2}$ and $B1_{i,j}^{*,n+1/2}$ at every cell corner. We then solve, at every grid point, a similar pair of characteristic equations which represent the propagation of Alfvén waves in the 1-direction, namely

$$\frac{Dv_2}{Dt} \pm \frac{1}{\rho^{1/2}} \frac{D\ B_2}{Dt} = 0, \tag{1.80}$$

where

$$D/Dt = \partial/\partial t + (v_1 \mp B_1/\rho^{1/2})\partial/\partial x_1. \tag{1.81}$$

This yields $v2^{*,n+1/2}_{i,j}$ and $B2^{*,n+1/2}_{i,j}$ at every cell corner. We now have all the starred quantities required to evaluate the corner-centered EMF via eq. 1.70. The poloidal field is updated using eqs. 1.66-1.67. A more complete discussion of the derivation of the hybrid MOC-CT method, and the limitations of the present implementation, is given in [25].

The toroidal component of the magnetic field is divergence-free from symmetry. Therefore it is not evolved using CT. Nonetheless, we must still use MOC to handle torsional Alfvén waves properly. Table 1.3 summarizes the various steps in evolving the poloidal and toroidal magnetic fields in ZEUS. The difference equations are given in the following sections.

TABLE 1.3. Summary of MHD Algorithms in ZEUS-2D[1]

	Poloidal field	Toroidal field
Source step	1) accelerate v1 and v2 due to magnetic pressure gradients (CSFT) 2) accelerate v1 and v2 due to poloidal Alfvén waves[2] (MOC)	1) evolve v3 and b3 due to torsional Alfvén waves (MOC)
Transport step	1) compute EMFs (ε3) using MOC 2) transport B1, B2 (CT)	1) compute upwind flux of B3 (MUA) 2) transport B3

[1] CSFT = centered-space, forward-time; MOC = method of characteristics; CT = constrained transport; MUA = monotonic upwind advection [2] performed at the beginning of transport step

1.5.2 SOURCE STEP

Subroutines: **lorentz, stv3b3**

The numbering on the following MHD source substeps refer to the hydrodynamic substeps described in §1.4.1. Fig. 4 shows the sequence of MHD operations within the overall program flow.

Substep 1a: (longitudinal Lorentz acceleration)

Expanding the vector operators in the equation of motion 1.15, we can write the Lorentz force terms in covariant form as

$$\frac{\partial v_1}{\partial t} = \frac{B_2}{\rho\, h_2}\frac{\partial B_1}{\partial x_2} - \frac{B_2}{\rho\, h_2}\frac{\partial}{\partial x_1}(h_2 B_2) - \frac{B_3}{\rho\, h_3}\frac{\partial}{\partial x_1}(h_3 B_3), \tag{1.82}$$

$$\frac{\partial v_2}{\partial t} = \frac{B_1}{\rho\, h_2}\frac{\partial}{\partial x_1}(h_2 B_2) - \frac{B_1}{\rho\, h_2}\frac{\partial B_1}{\partial x_2} - \frac{B_3}{\rho\, h_2\, h_3}\frac{\partial}{\partial x_2}(h_3 B_3). \tag{1.83}$$

We identify the first term in these equations as those which, when coupled to the induction equation, describe the propagation of Alfvén waves, while

the latter terms represent changes in momentum due to gradients of magnetic pressure. We split the update for the Lorentz force terms into two parts. The first partial update is performed in the source step using the finite-difference equations

$$\frac{v1_{i,j}^{n+a} - v1_{i,j}^{n}}{\Delta t} = - \frac{\overline{B2_{i,j}}}{(d_{i-1,j} + d_{i,j})} \left(\frac{g2b_i(B2_{i,j} + B2_{i,j+1}) - g2b_{i-1}(B2_{i-1,j} + B2_{i-1,j+1})}{g2a_i \; dx1b_i} \right) - \frac{(B3_{i-1,j} + B3_{i,j})}{(d_{i-1,j} + d_{i,j})} \left(\frac{g31b_i B3_{i,j} - g31b_{i-1} B3_{i-1,j}}{g31a_i dx1b_i} \right), \tag{1.84}$$

$$\frac{v2_{i,j}^{n+a} - v2_{i,j}^{n}}{\Delta t} = - \frac{\overline{B1_{i,j}}}{(d_{i-1,j} + d_{i,j})} \left(\frac{(B1_{i,j} + B1_{i+1,j}) - (B1_{i,j-1} + B1_{i+1,j-1})}{g2b_i \; dx2b_j} \right) - \frac{(B3_{i,j-1} + B3_{i,j})}{(d_{i,j-1} + d_{i,j})} \left(\frac{g32b_j B3_{i,j} - g32b_{j-1} B3_{i,j-1}}{g2b_i \; g32a_j dx2b_j} \right), \tag{1.85}$$

$$\overline{B1_{i,j}} = (B1_{i,j} + B1_{i+1,j} + B1_{i,j-1} + B1_{i+1,j-1})/4, \tag{1.86}$$

$$\overline{B2_{i,j}} = (B2_{i,j} + B2_{i,j+1} + B2_{i-1,j} + B2_{i-1,j+1})/4. \tag{1.87}$$

Substep 1b: (transverse Lorentz acceleration)

The second partial update for the Lorentz force terms that result in Alfvén wave motion is actually performed at the beginning of the transport step to reduce computational overhead, but conceptually belongs in the source step. The finite-difference equations for this update utilize the time advanced variables $B1_{i,j}^{\prime,n+1/2}$ and $B2_{i,j}^{\prime,n+1/2}$ computed from a MOC solution of the directionally split characteristic equations in a frame *co-moving* with the fluid, so that

$$\frac{v1_{i,j}^{n+1} - v1_{i,j}^{n+a}}{\Delta t} = \frac{2\overline{B2_{i,j}}}{(d_{i-1,j} + d_{i,j})} \left(\frac{B1_{i,j+1}^{\prime,n+1/2} - B1_{i,j}^{\prime,n+1/2}}{g2a_i \; dx2a_j} \right), \tag{1.88}$$

$$\frac{v2_{i,j}^{n+1} - v2_{i,j}^{n+a}}{\Delta t} = \frac{2\overline{B1_{i,j}}}{(d_{i-1,j} + d_{i,j})} \left(\frac{g2a_{i+1} \; B2_{i+1,j}^{\prime,n+1/2} - g2a_i \; B2_{i,j}^{\prime,n+1/2}}{g2b_i \; dx1a_i} \right). \tag{1.89}$$

Lorentz

To compute $B_1^{\prime,n+1/2}$ and $B_2^{\prime,n+1/2}$ we solve equations equations 1.78 and 1.80 except now the characteristic curves are defined in the Lagrangian frame:

$$D/Dt = \partial/\partial t \mp (B_2/\rho^{1/2})/h_2 \; \partial/\partial x_2, \tag{1.90}$$

for Alfvén waves propagating in the 2-direction (eq. 1.78), and

$$D/Dt = \partial/\partial t \mp (B_1/\rho^{1/2})\partial/\partial x_1, \tag{1.91}$$

for Alfvén waves propagating in the 1-direction (eq. 1.80). The reason for this is that the momentum equation is operator split into source and transport steps, so that the source step is effectively in the Lagrangian frame. See SNII for a more complete discussion of this point.

read it

Substep 4: (torsional Alfvén waves)

For two-dimensional calculations, the 3-components of the equation of motion and induction equation can be written in full as

$$\frac{\partial v_3}{\partial t} = -\frac{1}{h_2 h_3}\frac{\partial}{\partial x_1}(h_2 h_3^2 v_1 v_3) - \frac{1}{h_2 h_3}\frac{\partial}{\partial x_2}(h_3^2 v_2 v_3) + \frac{B_1}{\rho\ h_3}\frac{\partial}{\partial x_1}(h_3 B_3) + \frac{B_2}{\rho\ h_2\ h_3}\frac{\partial}{\partial x_2}(h_3 B_3), \tag{1.92}$$

$$\frac{\partial B_3}{\partial t} = -\frac{1}{h_2}\frac{\partial}{\partial x_1}(h_2 v_1 B_3) - \frac{1}{h_2}\frac{\partial}{\partial x_2}(v_2 B_3) + B_1\ h_3\frac{\partial}{\partial x_1}(v_3/h_3) + B_2\ h_3/h_2\frac{\partial}{\partial x_2}(v_3/h_3). \tag{1.93}$$

The leading two terms in each of these equations are transport terms, accounting for advection of v_3 or B_3 respectively due to fluid motion in each direction, while the other two terms are source terms. Following the operator splitting formalism described in Paper I, we split the update of v_3 and B_3 due to source and transport terms into two separate steps. In the source step, we solve finite-difference approximations to

$$\frac{\partial v_3}{\partial t} = \frac{B_1}{\rho\ h_3}\frac{\partial}{\partial x_1}(h_3 B_3) + \frac{B_2}{\rho\ h_2\ h_3}\frac{\partial}{\partial x_2}(h_3 B_3), \tag{1.94}$$

$$\frac{\partial B_3}{\partial t} = B_1\ h_3\frac{\partial}{\partial x_1}\left(\frac{v_3}{h_3}\right) + \frac{B_2\ h_3}{h_2}\frac{\partial}{\partial x_2}\left(\frac{v_3}{h_3}\right), \tag{1.95}$$

These source terms are responsible for torsional Alfvén waves. We solve eqs. 1.94-1.95 using directional splitting; i.e., we first update v_3 and B_3 due to torsional Alfvén waves propagating in the 1-direction, followed by an identical step in the 2-direction. The centered-space, centered-time, fully upwind difference equations are

$$\frac{v3_{i,j}^{n+a} - v3_{i,j}^{n}}{\Delta t} = \frac{(B1_{i,j} + B1_{i+1,j})}{2\ g31b_i\ d_{i,j}\ dx1a_i}(\Phi_{i+1,j}^{*,n+1/2} - \Phi_{i,j}^{*,n+1/2}), \tag{1.96}$$

$$\frac{B3_{i,j}^{n+a} - B3_{i,j}^{n}}{\Delta t} = \frac{(B1_{i,j} + B1_{i+1,j})\ g31b_i}{2\ dx1a_i}(\Omega_{i+1,j}^{*,n+1/2} - \Omega_{i,j}^{*,n+1/2}). \tag{1.97}$$

followed by

$$\frac{v3_{i,j}^{n+1} - v3_{i,j}^{n+a}}{\Delta t} = \frac{(B2_{i,j} + B2_{i,j+1})}{2\ g32b_j\ d_{i,j}\ g2bi\ dx2a_j}(\Phi_{i,j+1}^{*,n+1/2} - \Phi_{i,j}^{*,n+1/2}), \tag{1.98}$$

$$\frac{B3_{i,j}^{n+1} - B3_{i,j}^{n+a}}{\Delta t} = \frac{(B2_{i,j} + B2_{i,j+1})\ g32b_j}{2\ g2b_i\ dx2a_j}(\Omega_{i,j+1}^{*,n+1/2} - \Omega_{i,j}^{*,n+1/2}). \tag{1.99}$$

where $\Omega = v_3/h_3$ and $\Phi = h_3 B_3$. The quantities $\Omega_{i,j}^{*,n+1/2}$ and $\Phi_{i,j}^{*,n+1/2}$ are time-centered and upwind in the Alfvén wave characteristic, and are computed from the characteristic equations

$$(\Omega_i^{*,n+1/2} - \Omega_i^{+,n}) + (\Phi_i^{*,n+1/2} - \Phi_i^{+,n})/(h_3^2\rho^{1/2})_i^+ = 0, \tag{1.100}$$

$$(\Omega_i^{*,n+1/2} - \Omega_i^{-,n}) - (\Phi_i^{*,n+1/2} - \Phi_i^{-,n})/(h_3^2\ \rho^{1/2})_i^- = 0, \qquad (1.101)$$

where the $\Omega_i^{-,n}$ and $\Phi_i^{-,n}$ are the upwind interpolated values at the old time level at the footpoints of the C^- Alfvén characteristic, $\Omega_i^{+,n}$ and $\Phi_i^{+,n}$ are the upwind interpolated values at the old time level at the footpoints of the C^+ Alfvén characteristic (as in Fig. 6). See SNII for a derivation of eqs. 1.96-1.99.

1.5.3 TRANSPORT STEP

Subroutines: **ct, emfs**

We have already described in §1.5.1 the hybrid MOC-CT algorithm we employ for evolving the poloidal magnetic field so that it is divergence-free and upwind in the various MHD wave modes. The toroidal magnetic field component is transported conservatively by differencing the integral equation for toroidal flux advection

$$\frac{d}{dt}\int_A B_3\ dA = -\int_{\partial A} B_3(\mathbf{v} - \mathbf{v_g}) \cdot dl. \qquad (1.102)$$

This is done in a directionally split fashion by first advecting the magnetic flux in the 1-direction using

$$\mathcal{F}_{i,j}^1 = \dot{M}_{i,j}^1\ (B3/d)_{1,i,j}^*\ g2a_i^{n+1/2}, \qquad (1.103)$$

$$(B3_{i,j}^{n+1}\ g2b_i^{n+1}\ dx1a_i^{n+1} - B3_{i,j}g2b_i^n\ dx1a_i^n)/\Delta t = -(\mathcal{F}_{i+1,j}^1 - \mathcal{F}_{i,j}^1), \qquad (1.104)$$

where $(B3/d)_{1,i,j}^*$ represents the face centered and upwinded value resulting from one of the three interpolation algorithms in ZEUS (donor cell, van Leer or PPA). We then advect the magnetic flux in the 2-direction using

$$\mathcal{F}_{i,j}^2 = \dot{M}_{i,j}^2\ (B3/d)_{2,i,j}^*, \qquad (1.105)$$

$$(B3_{i,j}^{n+1}\ g2b_i^n\ dx2a_j^{n+1} - B3_{i,j}g2b_i^n\ dx2a_j^n)/\Delta t = -(\mathcal{F}_{i,j+1}^2 - \mathcal{F}_{i,j}^2). \qquad (1.106)$$

Note that by using the mass flux variables $\dot{M}_{i,j}^1$ and $\dot{M}_{i,j}^2$, we have used consistent transport (cf. §1.4.2) to minimize the diffusion of the toroidal magnetic field relative to the mass, as required by flux-freezing.

1.5.4 BOUNDARY CONDITIONS

Subroutines: **bvalemf,bvalb3**

To completely specify boundary conditions for an MHD simulation, one needs to specify boundary conditions for the hydrodynamic variables as well as for the poloidal and toroidal magnetic field. The hydrodynamic boundary conditions are described in §1.4.4. In order to maintain divergence-free poloidal fields, boundary conditions are applied to the EMFs used in the CT

scheme. The values of the EMFs are set in the two rows of ghost zones along each of the four boundaries of the computational domain (Fig. 1) according to the boundary condition type, of which there are six implemented in ZEUS. In practice, one sets the value of the integer arrays $miib_j$, $moib_j$, $mijb_i$, $mojb_i$ to one of the values $m = 1 - 6$ with the following effect:

1. **m=1, Reflecting Boundary Condition:** The normal components of velocity and magnetic field are reflected, while the tangential components are continuous. This implies the EMF in the ghost zones are equal to the negative of the EMF in the equivalent active zones.

2. **m=2, Outflow Boundary Condition:** The EMF in each ghost zone are set equal to the EMF in the first active zone.

3. **m=3, Inflow Boundary Condition:** The EMF in the ghost zones are set to some predetermined value, which may be allowed to vary in time.

4. **m=4, Periodic Boundary Condition:** The EMF in the ghost zones are set equal to the EMF in the corresponding active zones on the opposite side of the grid.

5. **m=5, Continuous B Boundary Condition:** The velocity is reflected, while both components of the magnetic field are continuous.

6. **m=6, Equatorial Symmetry Boundary Condition:** The velocity is reflected, while the normal component of the magnetic field is continuous and the tangential component is reflected. This condition is designed for symmetry planes such as equatorial symmetry.

For the unconstrained toroidal magnetic field components, we can apply boundary conditions directly. Thus, we use the same HD boundary conditions with one important exception. For outflow boundaries, perfectly transparent boundary conditions for outgoing Alfvén waves can be achieved by removing the incoming characteristic from the evolution equation at the boundary [23]; [27]. Numerically, this can be achieved by setting the gradients of all quantities along the incoming characteristic to zero. Thus, for outflow boundaries, we project the angular velocity and toroidal magnetic field to achieve perfect transparent boundaries for outgoing torsional Alfvén waves. In addition, at an equatorial plane of symmetry the toroidal field should be antireflected while the rotational velocity is merely reflected.

1.5.5 TIMESTEP CONTROL

Subroutine: **nudt**

For explicit MHD simulations, the CFL condition must be modified so that

$$\Delta t \leq \min(\Delta x)/(|u| + C_f), \tag{1.107}$$

where u is the local fluid velocity and C_f is the fast magnetosonic speed ($C_f^2 = v_A^2 + C_a^2$), where v_A and C_a are the Alfvén and sound speeds respectively). Thus, since $C_f > C_a$, the HD CFL condition may not be a sufficient stability criterion for strong fields. This is handled by including a fifth timestep, that due to Alfvén waves, into the sum of inverse square timesteps in eq. 1.62.

$$\delta t_5 = [\min(\Delta x_1, \Delta x_2)]/v_A, \tag{1.108}$$

1.6 Poisson Equation

Subroutines: **gravity, phibv, giccg, iccgaf**

The gravitational potential is determined by solving the Poisson equation subject to appropriate boundary conditions. We require a general-purpose solver that can operate in different coordinate geometries with general boundary conditions. This rules out Fourier transforms. Therefore we solve the sparse matrix which results from finite-differencing the Poisson equation. The gravitational potential is zone centered, thus using the covariant form of the Laplacian operator, the finite-difference equation can be written as

$$a_1\Phi_{i+1,j} + a_2\Phi_{i,j+1} + a_3\Phi_{i,j} + a_4\Phi_{i-1,j} + a_5\Phi_{i,j-1} = S_{i,j}, \tag{1.109}$$

where

$$\begin{aligned}
a_1 &= (g2a_{i+1}\ g31a_{i+1}\ dvl2a_j)/dx1b_{i+1} \\
a_2 &= (g32a_{j+1}\ dvl1a_i)/(g2b_i^2\ dx2b_{j+1}), \\
a_3 &= \frac{dvl1a_i}{g2b_i^2}\left(\frac{g32a_{j+1}}{dx2b_{j+1}} + \frac{g32a_j}{dx2b_j}\right) + dvl2a_j\left(\frac{g2a_{i+1}\ g31a_{i+1}}{dx1b_{i+1}} + \frac{g2a_i\ g31a_i}{dx1b_i}\right), \\
a_4 &= (g2a_i\ g31a_i\ dvl2a_j)/dx1b_i, \\
a_5 &= (g32a_j\ dvl1a_i)/(g2b_i^2\ dx2b_j), \\
S_{i,j} &= dvl1a_i\ dvl2a_j\ 4\pi G\ d_{i,j}.
\end{aligned}$$

To avoid discretization errors at the origin in curvilinear coordinates, we have used the volume differencing described in §1.3.3. Expressing eq. 1.109 as a matrix equation results in a symmetric sparse banded matrix with three main diagonals and two outriggers whose elements are functions of the grid only. This system is solved using a conjugate gradient technique employing incomplete Cholesky preconditioning. For each boundary of the computational 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. Neumann boundary conditions simply modify the diagonal elements of the row in which they are applied, while Dirichlet boundary conditions modify the RHS.

For the case of Dirichlet boundaries we compute Φ_B using a multipole expansion formula [16],

$$\Phi_B = G \sum_l P_l(\mu_B) r_B^{-(l+1)} M_l, \tag{1.110}$$

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

$$M_l = \int_V \rho(r) r^l P_l(\mu) dV. \tag{1.111}$$

This method only applies to curvilinear coordinates for which there is an axis of symmetry. Equations 1.110-1.111 are discretized as

$$\Phi_{B,i,j} = G \sum_l P_l(\mu_{B,i,j}) r_{B,i,j}^{-(l+1)} M_l, \tag{1.112}$$

where $$M_l = \sum_i \sum_j d_{i,j} r_{i,j}^l P_l(\mu_{i,j})\ dvl1a_i\ dvl2a_j, \tag{1.113}$$

In practice, when the problem boundaries are far from most of the mass distribution, then the series in equation 1.112 converges rapidly and only a few moments are needed. However, when the density becomes significant near the boundary, the series converges slowly and many moments must be taken to achieve an accurate solution. As implemented in ZEUS, we continue to add higher moments until Φ_B has converged to one part in 10^3, up to a maximum of 100 terms.

Because the solution of the Poisson equation is very costly, we do not solve it every hydrodynamic timestep unless the density distribution is changing rapidly. To decide when to solve it, we monitor the relative error in the currently stored gravitational potential compared to the current density distribution by rearranging equation 1.109 so that

$$\Phi'_{i,j} = \frac{1}{a_3}\left(S_{i,j} - a_1\Phi_{i+1,j} - a_2\Phi_{i,j+1} - a_4\Phi_{i-1,j} - a_5\Phi_{i,j-1}\right). \tag{1.114}$$

We only perform a solution of the Poisson equation when

$$|\Phi'_{i,j} - \Phi_{i,j}|/\Phi_{i,j} > \epsilon, \tag{1.115}$$

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.

1.7 The ZEUS-2D Code

The ZEUS-2D code is implemented in portable Fortran77 with cpp precompiler extensions which are used to include macrofiles, substitute macronames and configure the code for the application and target machine of choice. The procedure for using the code is given in the next section. Here we summarize the files provided with this book and how they are organized.

1.7.1 ZEUS TARFILE

To install ZEUS, simply untar the file named **zeus2d.tar** on a UNIX machine. You will see a directory called **zeus2d_v2.0**. Inside are several subdirectories: (1) **exe** - contains the ZEUS executable **zeus2d.exe** after make has been run, as well as the input file **z2dinput**; (2) **plt** - contains source files for a graphical post-processor utility based on NCAR graphics which reads ZEUS output files; (3) **src** - contains source files, header files and Makefile for the ZEUS-2D code; (4) **tst** - contains the ZEUS test suite for program verification. The filenames follow the UNIX convention wherein source code files end in the suffix **.src**, header files end in the suffix **.h**, and the makefile is called **Makefile**. The main program is contained in the file **zeus2d.src**. These files are described in more detail below.

1.7.2 THE .H FILES

Each module in the ZEUS program includes header files which contain the parameters and data structures needed by the code. Table 1.4 summarizes these header files, their function, and the common blocks they contain.

TABLE 1.4. ZEUS header files

file	common blocks	function
bndry.h	bndryr, bndryi	declare boundary value arrays
cons.h		declare physical and astronomical constants
cvmgt.h		emulate Cray vector merge function
field.h	fieldr	declare field variable arrays
gravity.h	gravcomr, gravcomi	declare control parameters for gravity algorithms
grid.h	gridvarr, gridvari	declare grid arrays
options.h		defines macros for startup banner (zeus2d.src)
param.h		declare Fortran parameters
radiation.h	radr	declare radiation field arrays and parameters
root.h	rootr, rooti, chroot1, chroot2	declare scalar variables and control parameters
scratch.h	scratch, iccgscr	declare scratch arrays

1.7.3 THE ZEUS2D.DEF FILE

The file **zeus2d.def** contains macro definitions which control which physics modules are compiled, how the Fortran arrays are to be dimensioned, identifies the target machine, and assigns names to the user-supplied subroutines. This file is included at the beginning of every source module and is processed by the cpp precompiler as a part of the MAKE. Table 1.5 summarizes the action of the macronames when they are defined. The last column indicates whether the macroname is required or optional.

TABLE 1.5. ZEUS-2D Macronames in zeus2d.def

Macroname	Type	Action	r/o[1]
XY	geometry	defines Cartesian coordinates	r
RZ	geometry	defines cylindrical coordinates	r
RT	geometry	defines spherical polar coordinates	r
GRAV	physics	compile self-gravity modules and arrays	o
ROTATE	physics	compile rotation modules and arrays	o
MHD	physics	compile MHD modules and and arrays	o
RAD	physics	compile radiation hydro modules and arrays	o
NUM_OF_I_ZONES	array	dimensions arrays in 1-direction	r
NUM_OF_J_ZONES	array	dimensions arrays in 2-direction	r
MAX_OF_IN_OR_JN	array	larger of NUM_OF_I_ZONES and NUM_OF_J_ZONES	r
A_SMALL_NUMBER	numeric	single precision small number (system specific)	r
A_BIG_NUMBER	numeric	single precision big number (system specific)	r
ABSORPTION	numeric	default value of κ used in RHD	r
D_ABSORPTION_DT	numeric	default value of $\partial\kappa/\partial T$	r
SCATTERING	numeric	default value of σ used in RHD	r
SUN	OS	target machine Sun workstation	r
SGI	OS	target machine Silicon Graphics workstation	r
CONVEX	OS	target machine Convex supercomputer	r
CRAY	OS	target machine Cray supercomputer	r
VISCOSITY	usermod	filename of artificial viscosity routine	r
PROBLEM	usermod	filename of problem generator	r
SPECIAL	usermod	filename of routine called in main loop	o
USERDUMP	usermod	filename of routine called during data i/o step	o

[1] r=required, o=optional

1.7.4 THE ZEUS2D.SRC FILE

Each subroutine in the ZEUS code is contained in a separate file with name *subroutine_name*.**src**. The main program is contained in the file **zeus2d.src**.

1.7.5 EXTERNAL LIBRARIES

ZEUS incorporates source code from the LINPACK library as well as a sparse matrix solver called **iccgaf** in order to achieve portability. ZEUS requires only one external library, namely the HDF (Hierarchical Data File) I/O library, which is used to write the output files for analysis and visualization (cf. §1.8.7). HDF is a set of portable file formats and utilities developed by the National Center for Supercomputing Applications, University of Illinois which is widely used and disseminated. The HDF library exists for most high performance workstations and supercomputers, and is available via anonymous ftp at *ftp.ncsa.uiuc.edu*; in the public directory `HDF`.

1.7.6 TEST SUITE

ZEUS has been extensively tested against a large suite of challenging problems as documented in [12]-[14], [28]. The problem generators, macro definition files, and input decks for this test suite are contained in directory `zeus2d_v2.0/tst` with this book. We highly recommend trying out these test problems, particularly those relevant to the application area of interest, in order to build experience with the code as well as to verify it is working as it should.

1.8 Using ZEUS-2D

The following sections give a step-by-step procedure for configuring, customizing, compiling and running ZEUS. It is assumed the user has untarred the ZEUS tar file `zeus2d.tar` (cf. §1.7.1). Building the executable is done within subdirectory `src`. Executing the code is done within subdirectory `exe`. We will illustrate with a specific example: a strong point explosion in a uniform magnetized medium. The reader should inspect the sample files discussed below to see a worked example.

1.8.1 MACRO DEFINITIONS

The first step is to specify the macro definitions in file `zeus2d.def` found in directory `zeus2d_v2.0/src` (see §1.7.3 for list of macros). Inspecting the file `zeus2d.def`, we see that macronames RZ and MHD have been defined, which means the problem will be computed in cylindrical coordinates including magnetic fields. The macronames NUM_OF_I_ZONES and NUM_OF_J_ZONES each have the value 105, which means that we are using a grid of 100x100 active zones. The additional five array elements per dimension are needed for two rows of ghost zones on each edge of the grid. The macroname SUN means that we will run the code on a Sun worksta-

the problem generator routine. Thus, the user is responsible for initializing the field variables in the active zones only, except for the magnetic field (see point 10 below).

9. **ifdef's:** geometry-specific or physics-specific blocks of code must be set off with the appropriate cpp **ifdef** statements. In our example, the spherical bubble is set up using different code blocks depending upon whether the calculation is done in spherical (**#ifdef RT**) or cylindrical coordinates (**#ifdef RZ**). The code which references the magnetic field arrays is set off using **#ifdef MHD**.

10. Divergence-free fields: the user is responsible for initializing magnetic fields which are divergence-free over the entire computational domain including the ghost zones. The ZEUS MHD algorithms will then evolve these fields in a divergence-free manner (cf. §1.5.1) everywhere, including the ghost zones. Constant magnetic fields are trivially divergence free, and can be initialized by assigning the magnetic field arrays $b1$, $b2$, $b3$ directly. Spatially varying magnetic fields will be numerically divergence-free only if they are constructed from differencing a vector potential. In this case, the magnetic field arrays $b1$ and $b2$ are computed from the 3-component of the vector potential $a3$. This procedure is illustrated in sample problem generator **doncox**.

11. Initial timestep: ZEUS automatically computes the initial timestep from the initial data. In some extreme circumstances it may be necessary to set the initial timestep to some small value from within the problem generator to maintain numerical stability in the early iterations of the code.

1.8.3 USER-SUPPLIED ROUTINES

Many applications require adding problem-specific subroutines to various phases of the calculation, such as within the timestep iteration loop, and during data input/output. Of course, this can always be accomplished by the editing the source files directly. A much simpler approach is to use the macro facility built into ZEUS via the macronames SPECIAL and USERDUMP in file **zeus2d.def**. The procedure is identical to that described for the problem generator above. By defining the macroname SPECIAL to be the name of a diskfile, the subroutine(s) contained within that file are called in the main iteration loop subsequent to the transport step. Likewise, the subroutine(s) continued within the file defined to be macroname USERDUMP will be called from subroutine **dataio** with time interval **dtusr** which is read from the input file.

1.8.4 RUNNING MAKE

The ZEUS executable **zeus2d.exe** is created by running MAKE in directory **zeus2d_v2.0/src**. with the following syntax:

make compile PROB=*probfile* [SPEC=*specfile*] [USER=*usrfile*]

```
$pcon  tlim=1.00e14 $
$hycon qcon=2.0,courno=0.5 $
$ggen1 nbl=100,x1min=0.,x1max=3.086e20,igrid=1,x1rat=1.,lgrid=.true.   $
$ggen2 nbl=100,x2min=0.,x2max=3.086e20,igrid=1,x2rat=1.,lgrid=.true.   $
$iib niib=105*1, miib=105*5 $
$oib noib=105*2, moib=105*2 $
$ijb nijb=105*-1,mijb=105*-1 $
$ojb nojb=105*2, mojb=105*2 $
$grvcon $
$radcon $
$eqos gamma=1.6666667 $
$pgen  radius=5., n0=0.1, T0=1.e4, b0=5.e-6, Msn=2., Esn=5.e50 $
$gcon $
$iocon dthdf=1.0e12 $
```

Comments:

1. rescon: `irestart=0` means start from scratch; `iid="ba"` appends two character identifier `ba` to all output files from this run.
2. pcon: `tlim` is problem termination time.
3. hycon: `qcon` is coefficient of artificial viscosity; `courno` is Courant number. Advection method is controlled here; default is second order-accurate van Leer.
4. ggen1: The x_1 grid lines are set up in blocks; each block can have a different number of zones (`nbl`) distributed in various ways (`igrid`, `x1rat`) between block boundaries `x1min` and `x1max`. Here we create a block of 100 active zones in the 1-direction, uniformly distributed (`x1rat=1`) on the domain $0 \leq x_1^{\bullet} \leq 100$ pc. The logical `lgrid=.true.` terminates the block loop.
5. ggen2: Same as ggen1, except for x_2 grid lines.
6. iib: specifies boundary type along inner-i boundary, which in RZ geometry, is the equatorial plane. `niib=1` specifies the reflection symmetry boundary condition for the hydrodynamic variables (cf. §1.4.3), and `miib=5` specifies the equatorial symmetry boundary condition on the magnetic field (cf. §1.5.4).
7. oib: specifies boundary type along outer-i boundary. `noib=2` and `moib=2` specifies outflow (i.e. zero gradient) boundary conditions on the hydrodynamic and magnetic field variables, respectively.
8. ijb: specifies boundary type along inner-j boundary, which in RZ geometry is the axis of symmetry. Consequently, `nijb=-1` and `mijb=-1` (cf. §§1.4.3, 1.5.4).
9. ojb: identical to 8, except for outer-j boundary.
10. grvcon: left blank because gravity option not used
11. radcon: left blank because radiation option not used
12. eqos: `gamma` is adiabatic constant
13. pgen: input parameters for problem generator `doncox`

Restart files (filename=**res***nnnid*); used for checkpointing the run for later restart. They are written by subroutine **msave** in native word single precision every **dtdump** wallclock seconds. To restart the code from an already existing restart dump be sure to use an executable which is identical to the executable that created the restart dump. Then modify the namelist parameter **irestart=1** and **resfile='***resfile***'** in **rescon** in the input deck **z2dinput**. When the code is started, it will then restart from the specified file instead of executing the **setup** routine. Note this causes many namelists in **z2dinput** to be skipped: on some machines the skipped namelists must be deleted from **z2dinput**. Any special coding the user wants to perform at the start of a restarted job should be placed in the routine **restart**.

HDF files (filename=**hdf***nnnid*); these contain all the 2D field arrays defined in the problem, as well as associated header information, in HDF SDS (Scientific Data Set) format (see §1.7.5). These files are written out every **dthdf** interval of problem time, and can be analyzed and visualized using many excellent software tools now available commercially or in the public domain, all of which read HDF files: Spyglass TRANSFORM™ (email *info@spyglass.com*); IDL; NCSA Image, NCSA Collage (*ftp.ncsa.uiuc.edu*). In addition, these files are read by the graphical post-processor **pltz2d** included with this distribution in directory **zeus2d_v2.0/plt**, which outputs NCAR plots.

History files (filename=**hst***nnnid*); these contain time series data of scalar quantities which are calculated during the run in subroutine **hdump**, such as integrated mass and energy, timestep, maxima and minima, etc. These files are written out every **dthist** interval of problem time in ASCII format, and can be analyzed by any plotting package, including Spyglass PLOT™, Kaleidagraph, CricketGraph, etc.

Userdump files (filename=**usr***nnnid*); these files contain whatever data are written out by the user-supplied subroutine defined with macroname USERDUMP in file **zeus2d.def** (cf. §1.8.3). They are written out every **dtusr** interval of problem time in whatever format the user chooses.

In the filenames given above, *nnn* is a three character numeric sequence 000-999, and *id* is the two letter problem identification the user assigns to the run in namelist **rescon** in the input deck.

1.9 Example: Magnetic SNR

To illustrate the use of ZEUS-2D, Fig. 7a shows a contour plot of the magnetized blast wave at time $\sim t = 1.58 \times 10^{13}$ s = 0.5 Myr. This calculation was done with a stationary grid of 100 x 100 zones in cylindrical RZ coordinates. By this time the remnant has expanded to NN pc and has begun to deform due to the magnetic stresses in the ambient medium. For comparison, Fig. 7b shows the remnant at the same dynamical age computed on a stationary grid of 100 radial by 30 angular zones in spherical polar

coordinates. Although the remnant shapes are the same, which is reassuring, it is clear that the RZ grid provides better resolution in the shell than the RT calculation. By contrast, the RT calculation does a better job of resolving the early phases of expansion because of the finer resolution near the explosion origin. This example illustrates a general feature of numerical simulations: there is no unique "best grid"; the choice of grid depends on what you want to study.

Acknowledgements

MLN would like to thank Drs. Jim Wilson and Jim LeBlanc of the Lawrence Livermore National Laboratory who first introduced him to the fascination of computational astrophysics. JMS thanks Dimitri Mihalas for his enthusiastic support. This work is partially supported by NSF grant AST-920113.

References

[1] Xie, X., & Toomre, J. 1993. Ap. J., 405, 747.
[2] Simon, R. H., & Toshiki, I. 1986. Ap. J., 304, 249.
[3] Mayle, R., Wilson, J. R., & Schramm, D. N. 1987, 318, 288.
[4] Fryxell, B. A., Taam, R. E., & McMillan, S. W. 1987. Ap. J., 315, 536.
[5] Hawley, J. F. 1986. In Radiation Hydrodynamics in Stars and Compact Objects, eds. D. Mihalas & K.-H. Winkler, Springer-Verlag.
[6] MacLow, M.-M., McCray, R., & Norman, M. L. 1989. Ap. J., 337, 141.
[7] Fiedler, R. A., & Mouschovias, T. Ch. 1993. Ap. J., 415, 680.
[8] Balsara, D. S., & Krolik, J. H. 1993. Ap. J., 402, 109.
[9] Cen., R., & Ostriker, J. P. 1993. Ap. J., 417, 415.
[10] Richtmyer, R. D., & Morton, K. W. 1967. Difference Methods for Initial Value Problems, Second Edition, Wiley: Interscience.
[11] Bowers, R. L., & Wilson, J. R. 1991. Numerical Modeling in Applied Physics and Astrophysics, (Jones & Bartlett: Boston).
[12] Stone, J. M., & Norman, M. L. 1992. Ap. J. Supp., 80, 759. SN I
[13] Stone, J. M., & Norman, M. L. 1992. Ap. J. Supp., 80, 791. SN II
[14] Stone, J. M., Mihalas, D. M., & Norman, M. L. 1992. Ap. J. Supp., 80, 819.
[15] Winkler, K.-H. A., Norman, M. L., & Mihalas, D. 1984. J. Quant. Spectrosc. Radiat. Transfer, 31, No. 6, 473.
[16] Jackson, J. D. 1975. Classical Electrodynamics, New York: Wiley.
[17] Colella, P., & Woodward, P. R. 1984. J. Comp. Phys., 54, 174.
[18] Oran, E. S., & Boris, J. P. 1987. Numerical Simulation of Reactive Flow, Elsevier: New York.
[19] Sod, G. A. 1985. Numerical Methods in Fluid Dynamics, Cambridge University Press: Cambridge, p. 179.
[20] Strang, W.G. 1968. SIAM J. Numer. Anal., 5, 506.
[21] van Leer, B., 1977. J. Comp. Phys., 23, 276.
[22] Norman, M. L., Wilson, J. R., & Barton, R. 1980. Ap. J., 239, 968.
[23] Thompson, K. W. 1987. J. Comp. Phys., 68, 1.
[24] Evans, C. R. & Hawley, J. F. 1988. Ap. J., 332, 659.

[25] Hawley, J.F., & Stone, J.M. 1994. Preprint.
[26] Lou, Y. Q., Rosner, R., & Ulmschneider, P. 1987, Ap. J., 315, 349.
[27] Vanajakshi, T. C., Thompson, K. W., & Black, D. C. 1989, J. Comp. Phys., 84, 343.
[28] Stone, J.M., Hawley, J.F., Evans, C.R, & Norman, M.L. 1992. Ap. J. 388, 415.

Figure Captions

Figure 1. Discretization of the computational grid used in ZEUS-2D. The thick line denotes the boundary of the computational domain. Two additional rows of ghost zones are needed to apply boundary conditions.

Figure 2. Coordinate grid definitions on the staggered grid: solid lines="a-grid"; dashed lines="b-grid".

Figure 3. A typical zone, showing the centering of the various primary and derived variables.

Figure 4. Schematic flow chart of the ZEUS-2D code. The items in rectangles denote the hydrodynamic algorithms. The items in rounded rectangles indicate the changes required to add the MHD algorithms.

Figure 5. The MHD control volume is produced by extending a 2D zone in the 3-direction. Here we specify the variables and contour integration used in Eqs. 1.66 and 1.67.

Figure 6. Space-time diagram for Alfven wave propagation showing the location of variables appearing in characteristic equations 1.76 and 1.77

Figure 7. Adiabatic magnetic blast wave at t=500,000 yr simulated using ZEUS-2D in two coordinate geometries: a) cylindrical RZ, 100x100 zones; b) spherical polar RT, 100x30 zones. Gas density and magnetic field lines are plotted as solid and dashed contours, respectively.

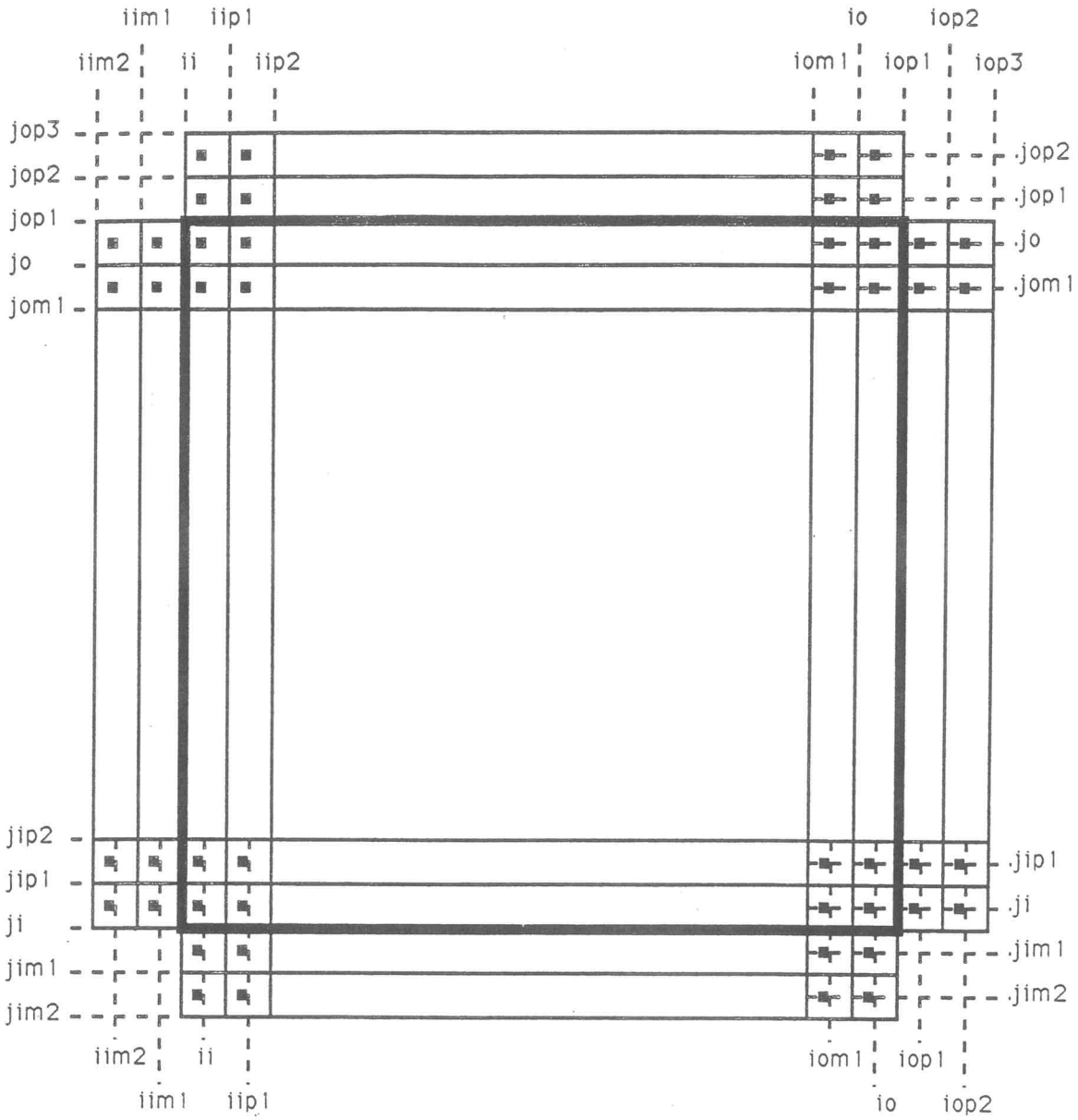

iim1
iip1
io
iop2
iim2
ii
iip2
iom1
iop1
iop3
jop3
jop2
jop1
jo
jom1
jip2
jip1
ji
jim1
jim2
jop2
jop1
jo
jom1
jip1
ji
jim1
jim2
iim2
ii
iom1
iop1
iim1
iip1
io
iop2

Figure 1

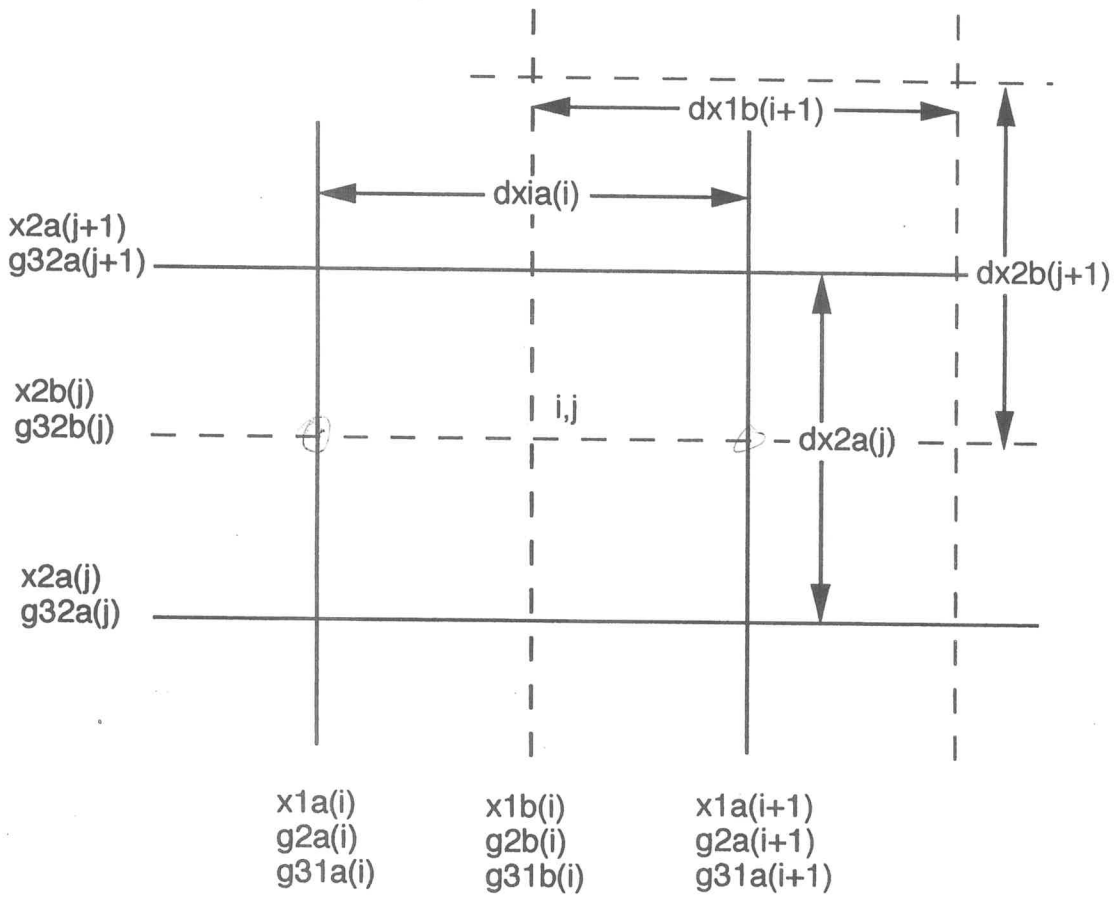

dx1b(i+1)
dxia(i)
x2a(j+1)
g32a(j+1)
dx2b(j+1)
x2b(j)
g32b(j)
i,j
dx2a(j)
x2a(j)
g32a(j)
x1a(i)
g2a(i)
g31a(i)
x1b(i)
g2b(i)
g31b(i)
x1a(i+1)
g2a(i+1)
g31a(i+1)

fig. 2

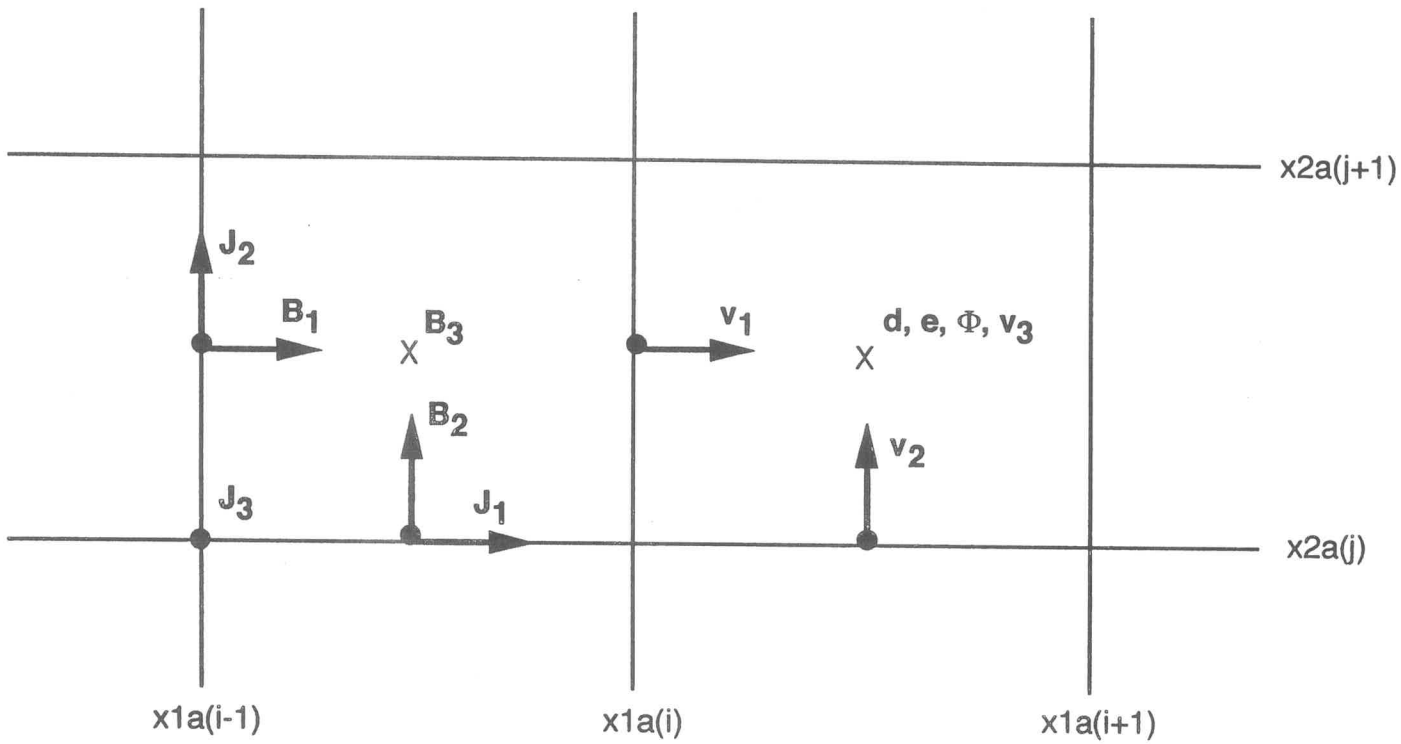


Fig 3

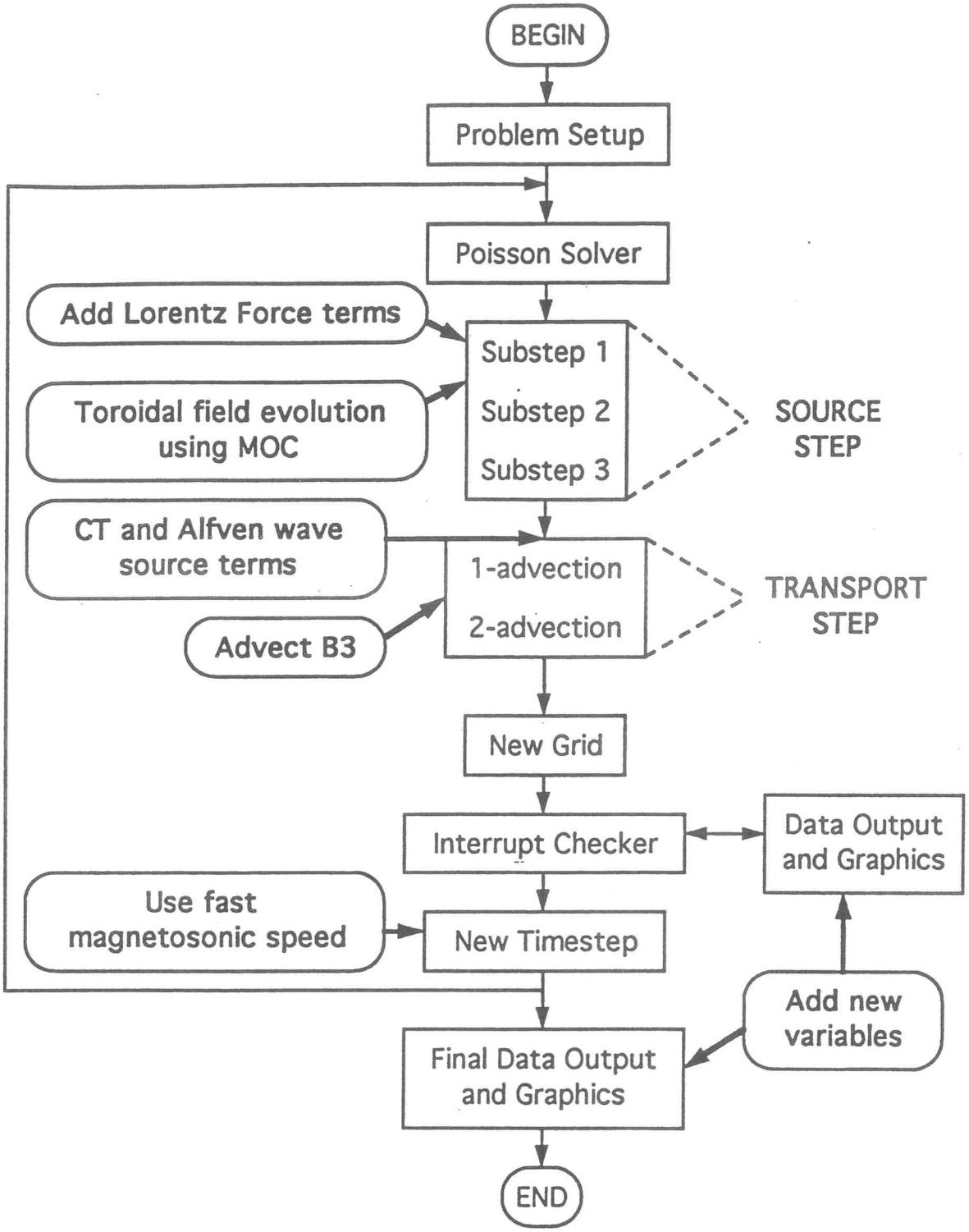

BEGIN
Problem Setup
Poisson Solver
Add Lorentz Force terms
Substep 1
Substep 2
Substep 3
SOURCE STEP
Toroidal field evolution using MOC
CT and Alfven wave source terms
1-advection
2-advection
TRANSPORT STEP
Advect B3
New Grid
Interrupt Checker
Data Output and Graphics
Use fast magnetosonic speed
New Timestep
Add new variables
Final Data Output and Graphics
END

Fig. 4

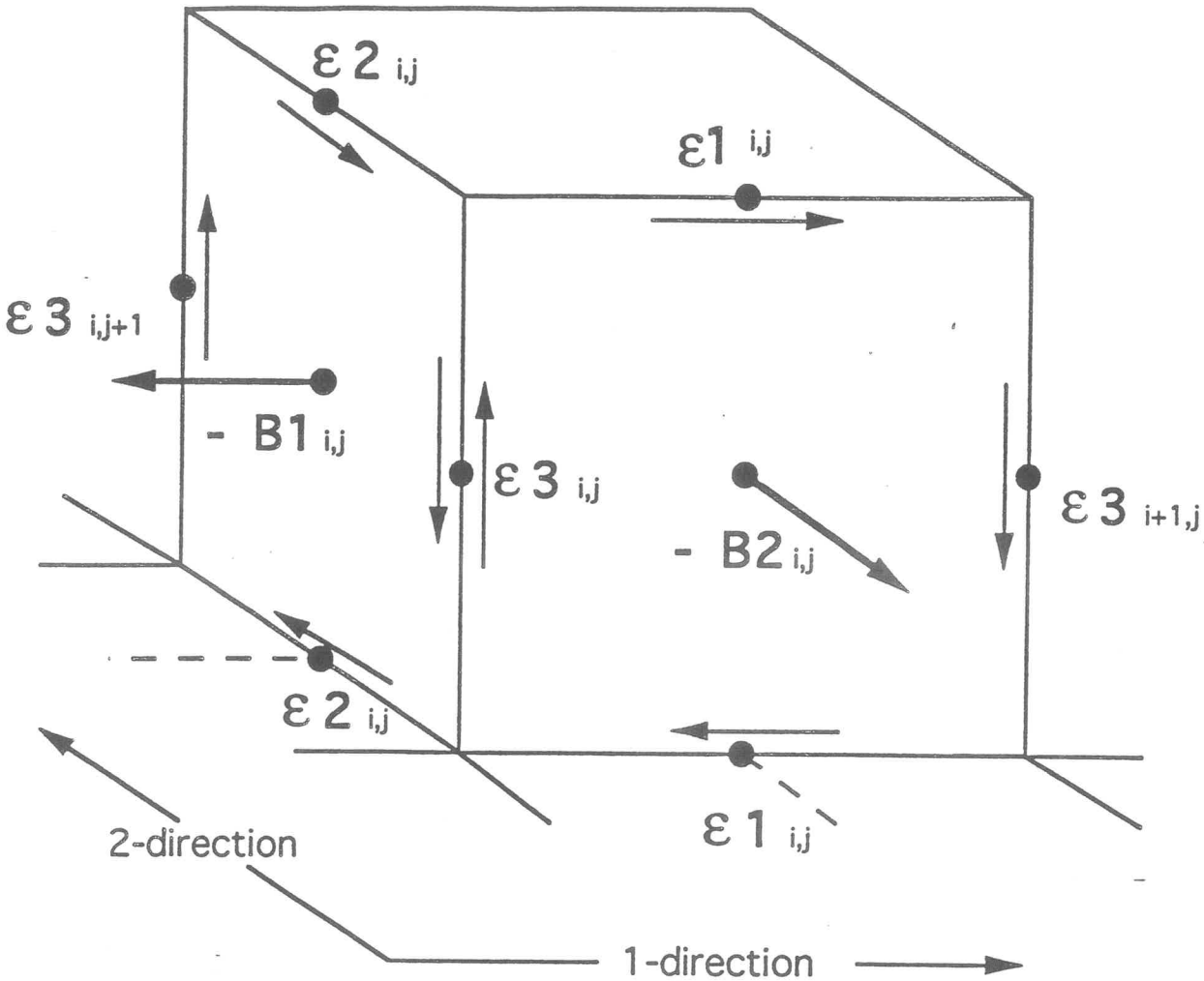

ε 2 i,j
ε1 i,j
ε 3 i,j+1
- B1 i,j
ε 3 i,j
- B2 i,j
ε 3 i+1,j
ε 2 i,j
ε 1 i,j
2-direction
1-direction

Fig. 5

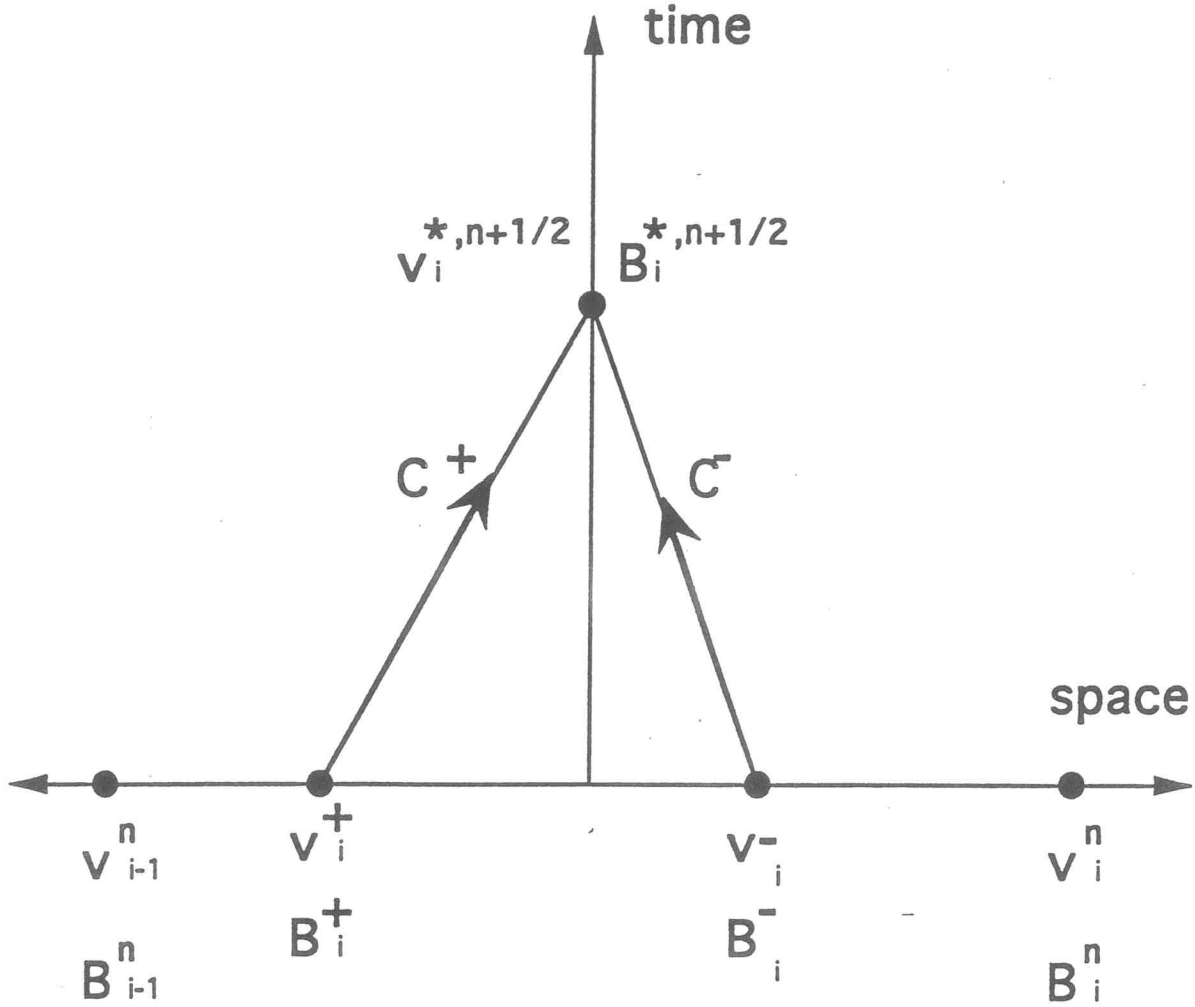
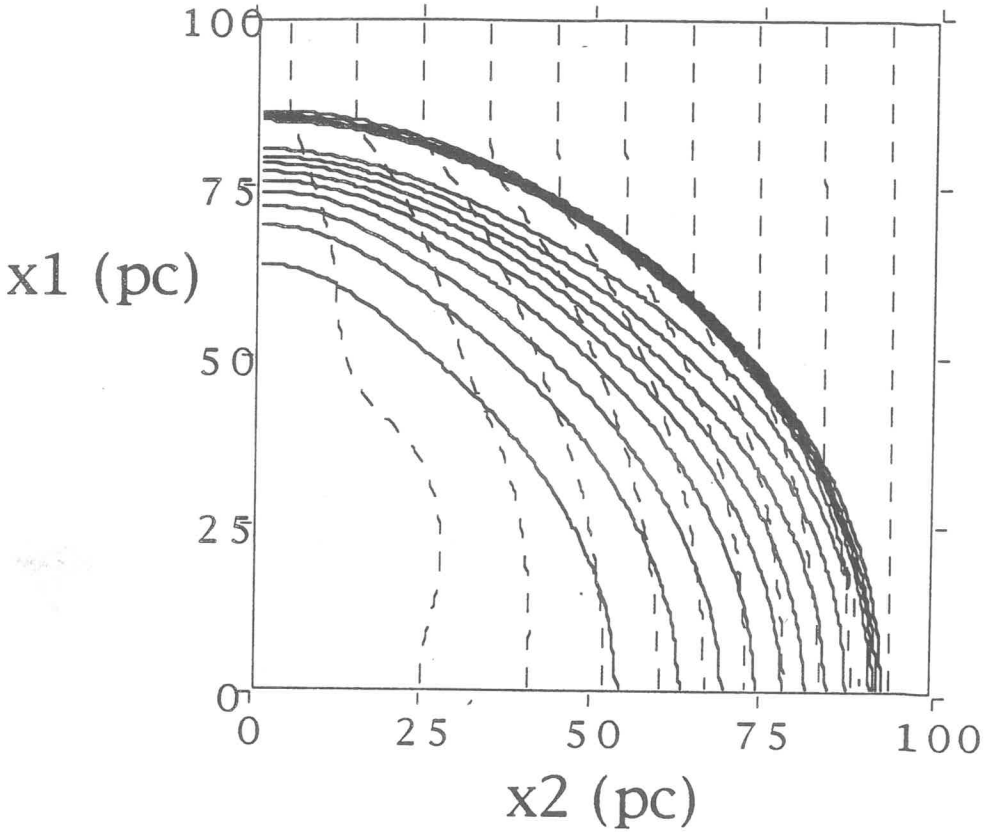


Fig. 6

a)


100
75
50
25
0
x1 (pc)
0
25
50
75
100
x2 (pc)

b)

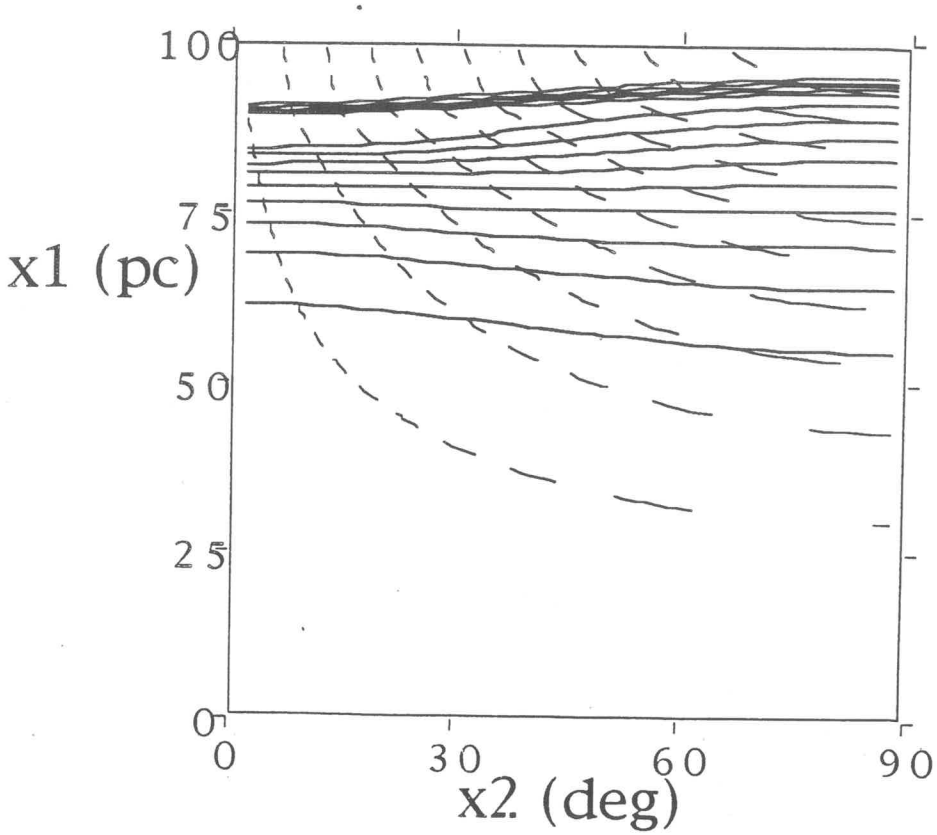

100
75
50
25
0
x1 (pc)
0
30
60
90
x2 (deg)

Fig. 7