

A FREE, FAST, SIMPLE, AND EFFICIENT TOTAL VARIATION DIMINISHING MAGNETOHYDRODYNAMIC CODE

UE-LI PEN,¹ PHIL ARRAS,² AND SHINGKWONG WONG^{1,3}

Canadian Institute for Theoretical Physics, University of Toronto, 60 St. George Street, Toronto, ON, M5S 3H8 Canada;
arras@cita.utoronto.ca, pen@cita.utoronto.ca

Received 2003 May 7; accepted 2003 July 23

ABSTRACT

We describe a numerical method to solve the magnetohydrodynamic (MHD) equations. The fluid variables are updated along each direction using the flux-conservative, second-order, total variation diminishing (TVD), upwind scheme of Jin & Xin. The magnetic field is updated separately in two-dimensional advection-constraint steps. The electromotive force (EMF) is computed in the advection step using the TVD scheme, and this same EMF is used immediately in the constraint step in order to preserve $\nabla \cdot \mathbf{B} = 0$ without the need to store intermediate fluxes. Operator splitting is used to extend the code to three dimensions, and Runge-Kutta is used to get second-order accuracy in time. The advantages of this code are high-resolution per grid cell, second-order accuracy in space and time, enforcement of the $\nabla \cdot \mathbf{B} = 0$ constraint to machine precision, no memory overhead, speed, and simplicity. A three-dimensional FORTRAN implementation less than 400 lines long is made freely available. We also implemented a fully scalable message-passing parallel MPI version. We present tests of the code on MHD waves and shocks.

Subject headings: methods: numerical — MHD

On-line material: source code

1. INTRODUCTION

Astrophysical fluids in which the magnetic field plays an important role are common in nature. As a few examples, consider magnetized interstellar gas, accretion disks, molecular clouds, and jets. With the advent of high-speed computers and ever-improving MHD codes, considerable theoretical progress has been made through numerical simulation of otherwise intractable problems.

A major challenge to solving flux-conservative systems of equations, such as ideal fluids and MHD, is the spontaneous development of shock discontinuities. Finite differencing across discontinuities leads to divergences and instabilities. Modern codes implement various aspects of “flux limiters” (Harten 1983) to achieve stability near shocks and second-order accuracy away from shocks. Recently, several shock capturing methods that solve the MHD equations in flux-conservative form with upwind finite differencing have been developed. Enforcing the $\nabla \cdot \mathbf{B} = 0$ constraint is key to the accuracy of these codes. Near shock fronts, derivatives are ill defined, and the divergence constraint can be maximally violated. Evans & Hawley (1988) first noted that this “constrained transport” (CT) can easily be enforced to machine precision by (1) defining the magnetic field at cell faces instead of centers and (2) using the same EMF, computed on cell corners, to update the magnetic flux through each neighboring face. Using CT and various shock capturing schemes, several groups (Dai & Woodward 1998; Balsara & Spicer 1999; and see, e.g., Tóth (2000) for a review of different methods) have now produced robust, efficient MHD codes.

The detailed algorithm used for the finite differencing varies between the different groups. We do not give an exhaustive review of the literature but rather compare only with widely used codes or those similar to ours. The Zeus code (Stone & Norman 1992) partially updates certain fluid and magnetic field quantities along Alfvén, but not magnetosonic, characteristics to avoid short length scale instabilities in shear Alfvén waves. Ryu et al. (1998) use Harten’s TVD method, which evolves the fluid along all the characteristics by constructing the linearized eigenvectors. Dai & Woodward (1998) used approximate Riemann solvers (Dai & Woodward 1994) to compute the fluxes. Common to these two methods is the need to first compute the EMF over the whole grid, then perform a spatial averaging of the EMF, and then update the magnetic field in a divergence-free form.

In this paper we implement the divergence constraint in a slightly different way than previous investigators. We show that individual pieces of the EMF can be used in advection-constraint steps, *without the need to store the computed EMFs over the whole grid*. This gives us a sizeable savings in memory than if the EMFs were stored. Furthermore, by using Jin & Xin’s (1995) “symmetric”⁴ method of computing TVD fluxes, we can reduce the operations count relative to codes that manifestly evolve the fluid along characteristics.

In § 2 we review the MHD equations. In § 3 we describe our numerical method. Tests of the code are presented in § 4. Section 5 contains a discussion of the merits and drawbacks of the code. Section 6 contains the conclusions. We briefly

¹ Canadian Institute for Theoretical Astrophysics, University of Toronto.

² Kavli Institute for Theoretical Physics.

³ Physics Department, National Taiwan University.

⁴ This method is called symmetric since it decomposes each fluid quantity into left and right moving parts, each of which can be differenced in an upwind fashion.

review the Jin & Xin (1995) method for solving one-dimensional advection equations in an Appendix.

2. EQUATIONS

The MHD equations expressing conservation of mass, momentum, and energy, as well as magnetic flux freezing, are (Landau & Lifshitz 1984)

$$\partial_t \rho + \nabla(\rho \mathbf{v}) = 0, \quad (1)$$

$$\partial_t(\rho \mathbf{v}) + \nabla(\rho \mathbf{v} \mathbf{v} + P_* \boldsymbol{\delta} - \mathbf{b} \mathbf{b}) = \rho \mathbf{a}, \quad (2)$$

$$\partial_t e + \nabla[(e + P_*) \mathbf{v} - \mathbf{b} \mathbf{b} \cdot \mathbf{v}] = \rho \mathbf{v} \cdot \mathbf{a}, \quad (3)$$

$$\partial_t \mathbf{b} = \nabla \times (\mathbf{v} \times \mathbf{b}), \quad (4)$$

$$\nabla \cdot \mathbf{b} = 0, \quad (5)$$

$$P_* = p + \frac{b^2}{2}, \quad (6)$$

$$e = \frac{\rho v^2}{2} + \frac{p}{\gamma - 1} + \frac{b^2}{2}. \quad (7)$$

Here ρ and e are the mass and (total) energy densities, \mathbf{v} is the velocity, P_* is the total pressure, p is the gas pressure, $\mathbf{b} = \mathbf{B}/\sqrt{4\pi}$ is the magnetic field in terms of $\sqrt{4\pi}$, \mathbf{a} is an externally imposed acceleration, and $\boldsymbol{\delta}$ is the Kronecker delta symbol. In equation (7) we have used an ideal gas equation of state with internal energy $\varepsilon = p/(\gamma - 1)$, where γ is the ratio of specific heats. The infinite conductivity limit has been used so that the electric field is $\mathbf{E} = -\mathbf{v} \times \mathbf{B}/c$. The electric force has been ignored since it is assumed that charge separation is negligible on the scales of interest.

3. NUMERICAL METHOD

First we describe the update of the magnetic field in two-dimensional advection-constraint steps. We then briefly review the update of the fluid variables along one dimension. Next we discuss how operator splitting and Runge-Kutta can be used to make the code second-order-accurate in space and time. Finally, we discuss boundary conditions, fine tunings of the code, and the parallel implementation.

3.1. Solution of the Induction Equation in Two Dimensions

We use operator splitting to reduce the problem into a series of smaller decoupled equations. Alternating the order of operators in the correct fashion allows one to achieve net second-order accuracy. In this prescription, we hold the fluid variables fixed to update the magnetic field. The magnetic field is defined on cell faces (see Fig. 1) in order to satisfy the $\nabla \cdot \mathbf{b} = 0$ to machine precision. Let the cell centers be denoted by $(i, j, k) \equiv (x_i, y_j, z_k)$, and faces by $(i \pm 1/2, j, k)$, $(i, j \pm 1/2, k)$, and $(i, j, k \pm 1/2)$, etc. For convenience, let the cells have unit width. The magnetic field is then stored in arrays:

$$\begin{aligned} b_x(i, j, k) &= b_{i-1/2, j, k}^x, \\ b_y(i, j, k) &= b_{i, j-1/2, k}^y, \\ b_z(i, j, k) &= b_{i, j, k-1/2}^z. \end{aligned} \quad (8)$$

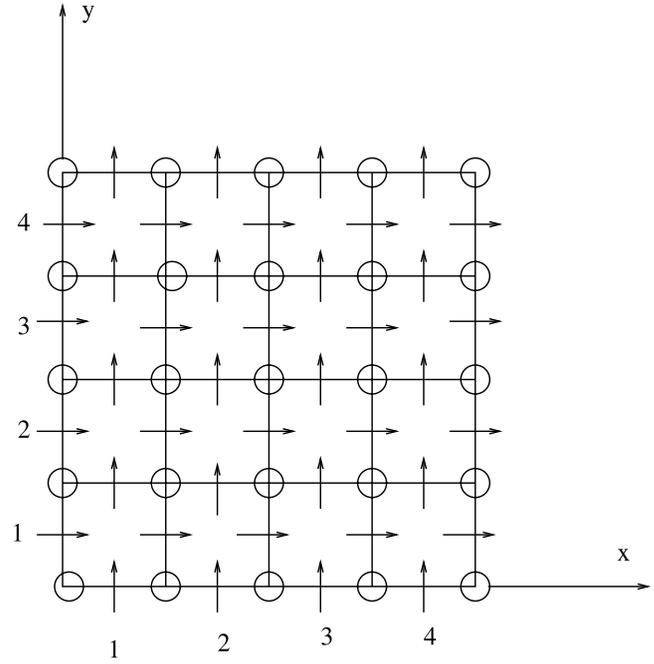


FIG. 1.—Position of variables on the grid. The vertical (horizontal) arrows represent $b_y(b_x)$, respectively. The circles denote the position of the electromotive force.

The flux out of cell (i, j, k) is then

$$\begin{aligned} \left(\int d^3 \mathbf{x} \nabla \cdot \mathbf{b} \right)_{ijk} &= b_{i+1/2, j, k}^x - b_{i-1/2, j, k}^x \\ &\quad + b_{i, j+1/2, k}^y - b_{i, j-1/2, k}^y \\ &\quad + b_{i, j, k+1/2}^z - b_{i, j, k-1/2}^z. \end{aligned} \quad (9)$$

Since the magnetic field is defined on cell faces, where the magnetic flux in a cell is evaluated, it is possible to enforce $\nabla \cdot \mathbf{b} = 0$ to machine precision (Evans & Hawley 1988). If we defined the magnetic field at some other location, the divergence could only be kept to zero to truncation error arising from interpolation to the faces. The truncation errors are necessarily large near shock discontinuities. Next we describe a method to evolve the field that preserves $\nabla \cdot \mathbf{b} = 0$, if it is so initially.

When the induction equation is written out in spatial components, it is apparent that the terms involving $\mathcal{E} \equiv \mathbf{v} \times \mathbf{b}$ come in six pairs. For instance, the terms involving $v_y b_x$ are

$$\partial_t b_x + \partial_y(v_y b_x) = 0, \quad \partial_t b_y = \partial_x(v_y b_x). \quad (10)$$

The first equation is just the advection of b_x along the y direction, the second equation is a *constraint* that enforces $\nabla \cdot \mathbf{b} = 0$. The key point to note here is that to enforce $\nabla \cdot \mathbf{b} = 0$ we must use the same EMF computed in the advection step during the constraint step; otherwise $\nabla \cdot \mathbf{b} = 0$ will only be zero up to truncation error. We accomplish this by finding a second-order-accurate, upwind EMF $v_y b_x$ for the advection step to update b_x and then immediately use this same EMF for the constraint step to update b_y . Ryu et al. (1998) first store the EMFs over the entire three-dimensional grid, then average the EMFs, and then update

the field. We construct the EMF using Jin & Xin's (1995) TVD method, which is described in the Appendix. Note that the velocity v_{ijk}^y must be interpolated to the same position as the magnetic field $b_{i-1/2,j,k}^x$ with second-order accuracy.

Jin & Xin's (1995) symmetric method introduces a "flux freezing speed," c , which must be greater than or equal to the maximum speed at which information can travel. Since we are holding the fluid variables fixed, the flux freezing speed for the advection-constraint equation is just $c = |v_y|$.

3.2. Solution of the Fluid Equations in One Dimension

Now we briefly describe the fluid update. A more complete discussion is given in Trac & Pen (2003a). The magnetic field is held fixed and interpolated to grid centers with second-order accuracy. Let $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5) = (\rho, \rho v_x, \rho v_y, \rho v_z, e)$ represent the volume-averaged quantities positioned at the center of each cell. For advection along the x direction, the Euler, continuity, and energy equations can be written in flux-conservative form as

$$\partial_t \mathbf{u} + \nabla_x \mathbf{F} = 0, \quad (11)$$

where the flux vector is given by

$$\mathbf{F} = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + P_* - b_x^2 \\ \rho v_x v_y - b_x b_y \\ \rho v_x v_z - b_x b_z \\ (e + P_*) v_x - b_x \mathbf{b} \cdot \mathbf{v} \end{pmatrix},$$

and the pressure is determined by $p = (\gamma - 1) \times (e - \rho v^2/2 - b^2/2)$ and $P_* = p + b^2/2$. We hold the magnetic field fixed during the fluid update and interpolate \mathbf{b} to cell centers for second-order accuracy.

Equation (11) can be solved by symmetric TVD, described in the Appendix. The flux freezing speed is taken to be $c = |v_x| + (\gamma p/\rho + b^2/\rho)^{1/2}$, which is the maximum speed information can travel.⁵

3.3. Extension to Three Dimensions

Let the fluid update step for a time Δt along x be denoted by "fluidx," and update of b_x along y by "bxalongy." Operator splitting requires us to apply each operator first in forward, and then in reverse order to advance by two time steps. We implemented two versions. In one, we advance forward using the sequence of operations: fluidx, byalongx, bzalongx, fluidy, bxalongy, bzalongy, fluidz, bxalongz, byalongz, and then the reverse byalongz, bxalongz, fluidz, bzalongy, bxalongy, fluidy, bzalongx, byalongx, fluidx. A second implementation, used in the public version of this code, is to transpose the fluid variables and spatial dimensions (see, e.g., Press et al. 1996, p. 984). This is only easily done if two of the dimensions are equal. Transposing has the benefit of high efficiency on cache-based computers, where we only need to read data in column order. It is convenient to implement the three-dimensional code by using a single routine for the fluid update along the x direction and a single routine for advection of b_y along x , and constraint of b_x along y . The order of the spatial indices is transposed

to take account of the other directions. The advantage of having one subroutine is that it can be heavily optimized.

Other operators, such a gravity, heating or cooling, can be added into the sweep sequence, as described in Trac & Pen (2003a). One just has to keep in mind that the reverse sweep has the reverse ordering of operators. If a routine is particularly computationally expensive (for example some gravity solvers), one can put it in between the forward and reverse sweeps and apply it once with twice the time step $2\Delta t$.

The time step is set by the fastest speed at which information travels over the grid. Since the fluid update is more restrictive, the time step is set to be

$$\Delta t = \text{cfl} [\max(|v_x|, |v_y|, |v_z|) + (\gamma p/\rho + b^2/\rho)^{1/2}]^{-1} \quad (12)$$

where $\text{cfl} \lesssim 1$ is generally set to $\text{cfl} \simeq 0.7$ for stability.

3.4. Boundary Conditions

The standard boundary conditions such as periodic, continuous, or reflecting can be easily enforced by specifying values of the variables in "ghost zones" adjacent to the physical grid. These ghost zones are needed when interpolating \mathbf{v} to cell faces, \mathbf{b} to cell centers, and in the one-dimensional advection routines.

We have implemented the boundary conditions in two different ways. The first method is to pad the grid with a large number ($\sim 6-10$) of extra cells at each boundary. Both the on-grid and off-grid variables are evolved in time, but so many extra cells are used that the boundary cells only need be updated once per double time step. This method is useful for parallel implementations in which buffer zones are used to represent a small number of cells in adjacent regions. The second method to implement the boundary conditions requires that one write specific routines for interpolation or derivatives that specify the off-grid values. This method requires less computation and is preferable for serial applications. We find it convenient to evolve "extra" values of the magnetic field variables. That is, we evolve $b_{i-1/2,j,k}^x$ for $j = 1, \dots, n_y$ and $k = 1, \dots, n_z$ but $i = 1, \dots, n_x + 1$. This is useful for three reasons. First, this allows $\nabla \cdot \mathbf{b}$ to be computed over all cells. Second, to update $b_{n_x,j+1/2,k}^y$ the fluxes $v_y b_x$ are needed at in the boundary cells (see Fig. 1) with $i = n_x + 1/2$. Third, \mathbf{b} can be interpolated to cell centers without the need to specify off-grid values.

3.5. Fine Tunings of the Code

Since the TVD limiters are nonlinear, sinusoidal waveforms are not guaranteed to remain that way. With aggressive limiters, such as superbee or the van Leer limiter used in our tests, initial sine waves can tend to become "clipped," or boxy-looking (Trac & Pen 2003a). We find that these nonlinear distortions can be minimized by using constant flux freezing speed, set to be the maximum along that advection line. Additional stability can be gained by multiplying the flux freezing speed by a constant multiplicative factor, although this increases the number of time steps needed and makes the code more diffusive. Empirically, we find that smoothing the velocity field that advects the magnetic field can lead to less damping of the slow mode.

In production runs, we have found an occasional failure of the code when the Courant condition is pushed too close

⁵ We have taken the maximum speed of the fast MHD wave over all directions.

to the limit. In the operator split approach, the time step is fixed at the beginning of a double time step and determined from the Courant condition at the beginning. During the time step, this condition may change, leading to an instability if it exceeds the initial constraint. Our solution has been to be sufficiently conservative using a choice of $\text{cfl} \lesssim 0.7$. A more efficient procedure would be to measure the change in the Courant condition during the sweeps and use this as an indicator in subsequent time steps. And should a given sweep step be unstable, one can always break it into two sub-steps. An additional tuning that helps maintain high Courant condition is to increase the freezing speed slightly (20%) or to use a larger Alfvén speed for CFL purposes.

3.6. Parallel Implementation

We have implemented a fully distributed version in MPI. After a full set of operators in one dimension, we update the buffer zones. For hydrodynamics, only three buffer cells are required. The magnetic field requires interpolation, and we use 16 buffer cells for magnetized simulations. A full three-dimensional domain decomposition is implemented, where we update the buffers in the appropriate direction after each dimensional operator. Since only large faces are communicated, latency of communication is negligible, but significant bandwidth is required to move the buffer zones. The communication is performed asynchronously, and computations proceed during the communication stage. Within each node, OpenMP is used to utilize multiple processors in a node without the overhead of buffer cells and communications.

We tested the parallel implementation on the CITA McKenzie beowulf cluster. The main cluster has 256 nodes of dual Intel Pentium-4 Xeon processors running at 2.4 Ghz, 1 GB of RAM, dual gigabit ethernet, and 160 GB of disk. The networking consists of bristles of 16 machines with one gigabit port connected to a switch. The second gigabit port is used to interconnect the bristles in a cubical layout. The nominal bisection bandwidth is 128 Gbit s⁻¹. For cubical problems, the usable bandwidth is higher when the domain decomposition is matched to the cube as it is for our runs. To minimize communications overhead, we mapped the computational grid layout to coincide with the physical network interconnect. The largest problem that we have been able to run in memory is 1400³ grid zones, which takes about 40 s per double time step. The fine grained OpenMP parallelism within each nodes allows the code to benefit from the intranode hyperthread speedup. The code also fully vectorizes for the SSE2 parallel execution units. Due to the large number of buffer cells required, about 1/3 of the computation and memory are used by these buffers. The operation count of the van Leer limiter relaxing TVD algorithm is 33 floating point operations per variable per time step. The flux computation takes an additional seven operations averaged per variable, for a total count of 40. Each double time step consists of six sweeps of eight variables, or about 2000 floating point operations. Our execution speed corresponds to a sustained rate of over 200 Gflop on the cluster, which is about 5% of theoretical peak speed of 4.8 Tflop in single precision.

4. TESTS

In this section we present tests of the code on MHD waves and shocks.

For all the tests we use $\gamma = 5/3$ and set the box size L to be equal in all directions. Periodic boundary conditions were used for the wave tests, and continuous boundary conditions (all variables continuous across the boundary) in the shock tests. The wave tests are two dimensional, while the shock tests are along one dimension. The van Leer limiter (see, e.g., Trac & Pen 2003a) and a constant freezing speed and $\text{cfl} = 0.9$ (see § 3.5) were used throughout.

4.1. Torsional Alfvén Waves

Torsional Alfvén waves are exact nonlinear solutions of the compressible MHD equations. Unlike linearly polarized waves, these circularly polarized torsional waves involve no density change, and thus no compression, but can test the magnetic parts of the code. In the absence of any perturbations or noise, they should propagate without steepening, making them a good test for numerical codes.⁶

We perform tests for four different resolutions $n_x = n_z = 8, 16, 32, 64,$ and 128 , where $n_{y,z}$ are the number of grid points in the y and z directions. Different fluid pressures are used corresponding to low and high $\beta = c_s^2/c_A^2 = 0.1$ and 10 . The exact solution we input to the code is $\rho = 1$, $e = p/(\gamma - 1) + 0.5 + A^2$,

$$b_x = A \cos[k(y+z) - \omega t], \quad (13)$$

$$b_y = \frac{1}{\sqrt{2}} \{1 + A \sin[k(y+z) - \omega t]\}, \quad (14)$$

$$b_z = \frac{1}{\sqrt{2}} \{1 - A \sin[k(y+z) - \omega t]\}, \quad (15)$$

$$v_x = -A \cos[k(y+z) - \omega t], \quad (16)$$

$$v_y = -\frac{A}{\sqrt{2}} \sin[k(y+z) - \omega t], \quad (17)$$

$$v_z = \frac{A}{\sqrt{2}} \sin[k(y+z) - \omega t]. \quad (18)$$

The wavenumber and frequency are $k = 2\pi/L$ and $\omega = \sqrt{2}k$ for the lowest-order mode. We set the ratio of wave to background field to be $A = 0.1$ for the low gas pressure test, and $A = 1$ for the high gas pressure test. As long as the thermal energy is larger than the kinetic energy of the wave, we can also test very nonlinear waves.⁷

The waves were propagated for one wave period. The result is read along the z -axis and plotted against the exact solution in Figure 2.

The second-order convergence is apparent from the figures: as one doubles the number of grid cells, the errors goes down by a factor of 4.

Even the significantly nonlinear solutions are well behaved and also converge close to quadratically. To quantify the rate of convergence, we measured the rms error

⁶ In the presence of any infinitesimal noise, Alfvén waves are unstable to decay into three other waves (Goldstein 1978; Derby 1978). For large-amplitude waves the growth time becomes comparable to the Alfvén wave period. Hence, care must be used in applying this test to very nonlinear waves as noise arising from truncation error or the nonlinear flux limiters may grow exponentially.

⁷ For sufficiently small $\beta \ll 1$ and large-amplitude $A \sim 1$, we found the code is susceptible to a short length scale instability. The timescale over which this instability develops depends strongly on β and A , and also the number of grid points. It is unclear to us whether or not this is the physical decay instability of Alfvén waves due to perturbations seeded by the truncation error.

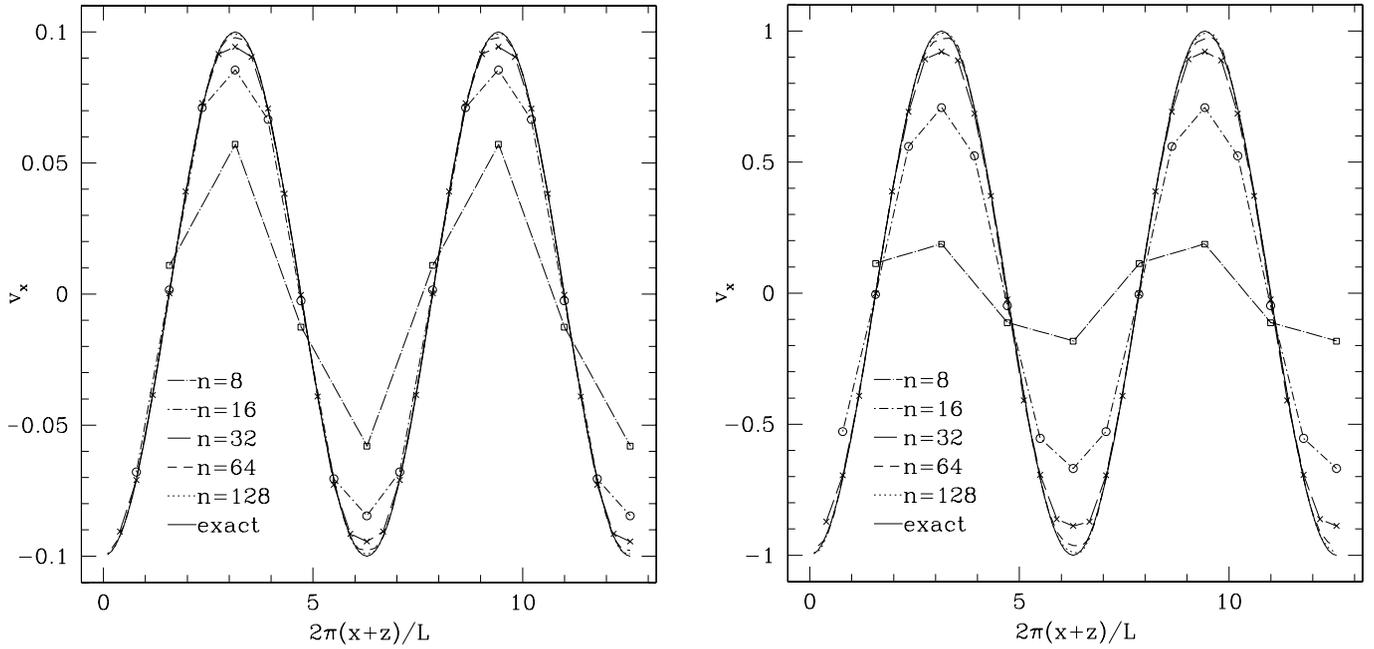


FIG. 2.—Torsional Alfvén wave with amplitude $v_{\text{wave}} = 0.1 v_{\text{Alfvén}}$ and $\beta = 0.1$ (left) and $v_{\text{wave}} = 1 v_{\text{Alfvén}}$ and $\beta = 10$ (right) propagated for one wave period. The different lines correspond to the analytic solution in eq. (18) and the code output for different numbers of grid points.

in the y component of the velocity field compared with the exact solution, normalized to a wave of unit amplitude. The results are shown in Figure 3. As expected, the fast wave has the smallest error, with the slow wave containing the largest error. The slow wave propagates slowly compared with the other characteristics and is thus subject to more time steps of diffusion. The convergence rate, however, is the same. We

note that all waves are at a 45° angle to the grid axes, and all waves nontrivially involve several magnetic field components. All operator split codes are at most second-order accurate. The TVD limiter reduces the order of the code to first order near extrema, so we expect the total order of convergence to be slightly worse than second. We have plotted a line with slope -1.75 for reference, which appears to describe the convergence well.

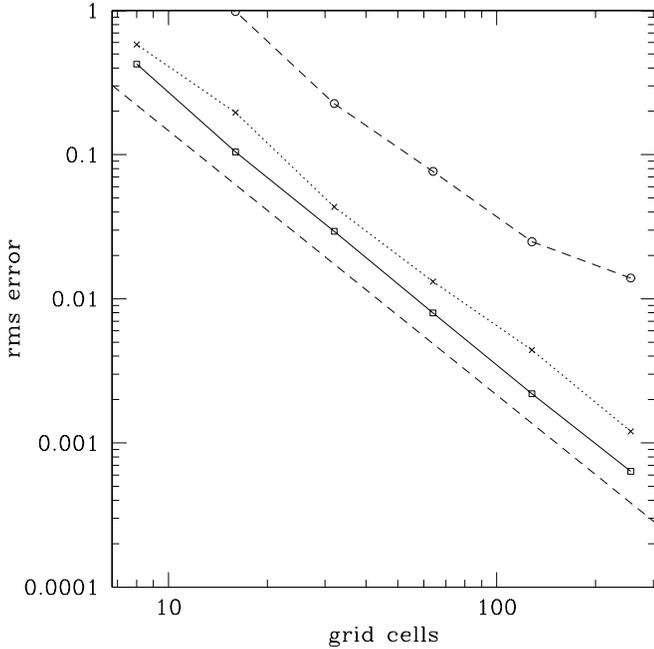


FIG. 3.—Convergence rate of various wave families. The solid line with boxes is for the fast wave at low β , the dotted line with crosses is the large-amplitude Alfvén wave, and the dashed line is the slow wave at large β , which has the largest prefactor on the convergence. The dotted straight line is drawn at a slope of -1.75 for reference.

4.2. Fast and Slow Waves

We have tested the fast and slow MHD waves in the linear regime $\delta p \ll p$ for $\beta = 0.1$ and 10 . The background magnetic field is $\mathbf{b} = \mathbf{e}_z$, background pressure is $p = \beta/\gamma$, and the wavevector is $\mathbf{k} = k(\mathbf{e}_x + \mathbf{e}_z)$, where $k = 2\pi/L$ and $L = L_y = L_z$. The exact solution we plug into the code is

$$\begin{aligned} \delta p &= pA \cos[k(x+z) - \omega t], \\ \rho &= 1 + \frac{\delta p}{c_s^2}, \\ \delta \mathbf{v} &= \frac{\omega}{\omega^2 - \omega_b^2} \left(\mathbf{k} - \frac{\omega_b^2}{\omega^2} \hat{\mathbf{b}} \hat{\mathbf{b}} \cdot \mathbf{k} \right) \frac{\delta p}{\rho}, \\ \delta \mathbf{b} &= \frac{k^2}{\omega^2 - \omega_b^2} \left(\mathbf{b} - \hat{\mathbf{k}} \hat{\mathbf{k}} \cdot \mathbf{b} \right) \frac{\delta p}{\rho}, \end{aligned} \quad (19)$$

where $c_s^2 = \gamma p/\rho$, $\omega_b^2 = k^2 b^2/\rho$, and $A \ll 1$ is the wave amplitude. The fast and slow mode frequencies are given by $\omega_{f,s}^2 = .5(\omega_b^2 + \omega_s^2) \pm 0.5[(\omega_b^2 + \omega_s^2)^2 - 4\omega_s^2 \omega_a^2]^{1/2}$, where $\omega_s^2 = c_s^2 k^2$ and $\omega_a^2 = (\mathbf{k} \cdot \mathbf{b})^2/\rho$. We evolved the waves for one period. The results for the fast and slow waves are shown in Figures 4 and 5, respectively. The slow wave is subject to substantial diffusion as compared with the fast wave since its frequency is so much lower for these extreme values of β . We also see phase errors that arise when the amplitude errors are large.

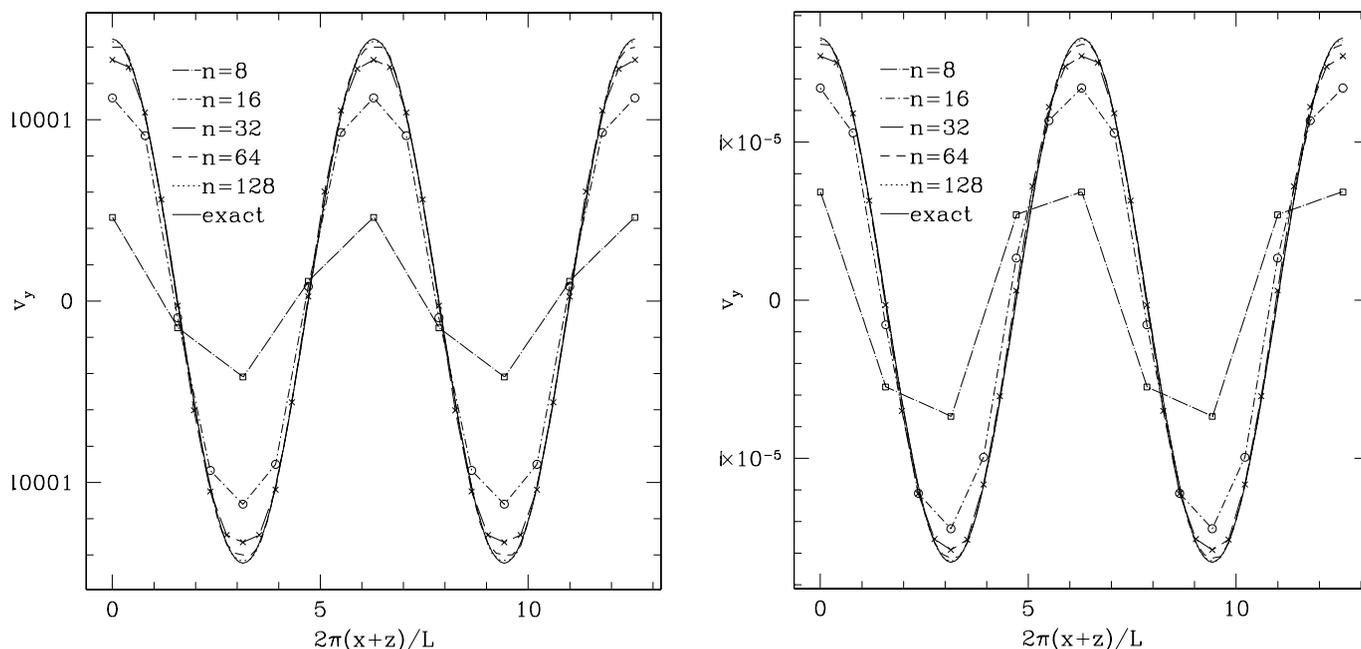


FIG. 4.—Fast MHD wave with $k_x L = k_z L = 2\pi$ and $\beta = 10$ and 0.1 in the left and right panels, respectively, evolved for one wave period

We pause to summarize that we have tested all MHD wave families and that second-order convergence is observed in all instances, even when the wave and magnetic field vectors are not aligned with the numerical grid directions. Our algorithm has not explicitly decomposed the problem into individual waves, but the algebraic convergence still holds.

4.3. The MHD Riemann Problem

These one-dimensional tests of the code involve a shock tube along the x -axis, as in Figures 2a and 2b of Ryu et al.

(1998). We used continuous boundary conditions and 1024 grid points for both tests.

The initial conditions for Figure 6 are

$$(\rho, v_x, v_y, v_z, p, b_x, b_y, b_z) = (1, 10, 0, 0, 20, 5/\sqrt{4\pi}, 5/\sqrt{4\pi}, 0)$$

for the left side and

$$(1, -10, 0, 0, 1, 5/\sqrt{4\pi}, 5/\sqrt{4\pi}, 0)$$

for the right side. The code is run for a time $0.08L$. The

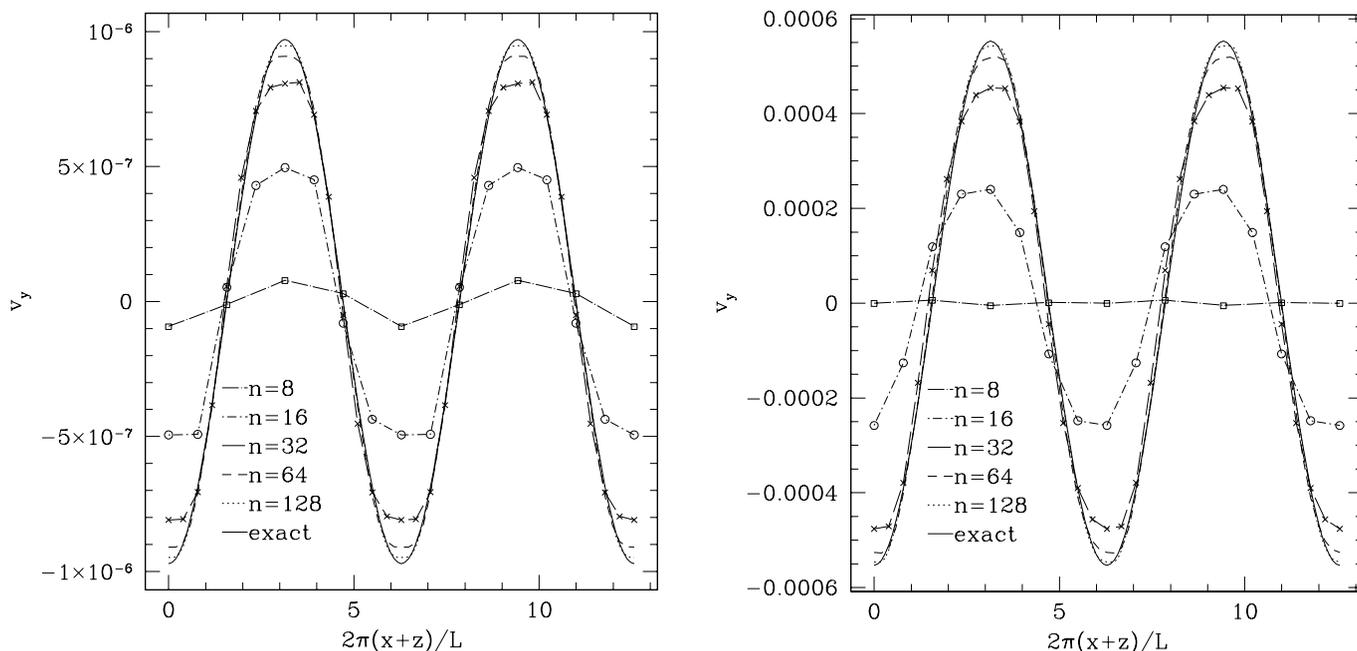


FIG. 5.—Slow MHD wave with $k_x L = k_z L = 2\pi$ and $\beta = 10.0$ and 0.1 in the left and right panels, respectively, evolved for one wave period

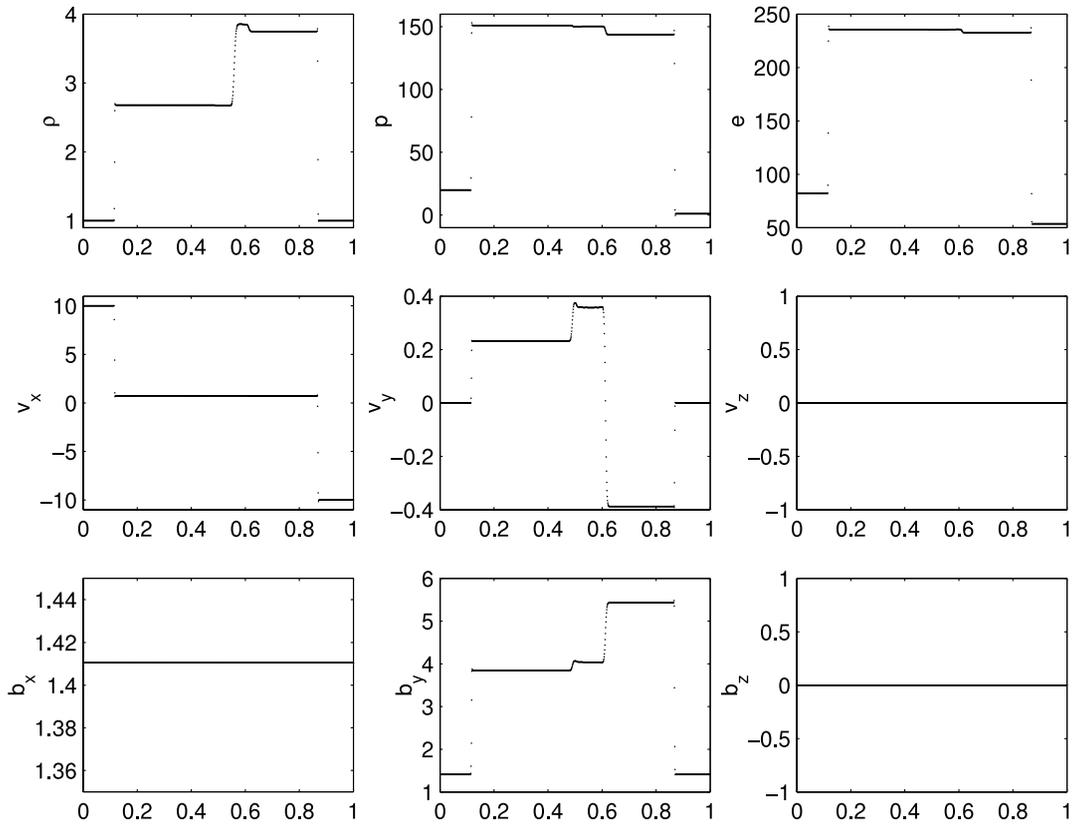


FIG. 6.—Shock tube test along the x -axis with velocities and magnetic field along the x - and y -directions

result agrees well with Figure 2a of Ryu et al. (1998). The following features can be seen. The steep discontinuities at $x \sim 0.1$ and $x \sim 0.85$ are fast shock fronts in which the incoming flow converts its kinetic energy into thermal energy and compresses the transverse field b_y . As new matter falls on, this shock is regenerated and maintains its steep profile as it moves outward. At $x \simeq 0.6$ and $x \simeq 0.5$ are a slow shock and slow rarefaction, respectively. The slow shock again compresses the fluid but decreases the transverse field. At $x \simeq 0.55$ the two phases of the initial gas configuration with different entropies form a contact discontinuity. Pressure, magnetic field, and velocity are continuous, while density and thermal energy experience a discontinuity. This discontinuity moves rightward across the grid, and the TVD advection of such discontinuities results in some smearing or diffusion of the structure. No physical mechanism steepens this contact discontinuity once it smears, and a slow numerical diffusion is visible in this, and all generic TVD codes that do not introduce explicit contact steepeners. There are no significant oscillations. Both our solution and that of and Ryu et al. (1998) have a slight overshoot in some variables in the first postshock cell, but this effect does not persist onto subsequent cells.

The initial condition for Figure 7 has velocity and magnetic field components in all directions, and hence exhibits additional structures such as rotational discontinuities. The values are

$$\begin{aligned}
 &(\rho, v_x, v_y, v_z, p, b_x, b_y, b_z) \\
 &= (1.08, 1.2, 0.01, 0.5, 0.95, 2/\sqrt{4\pi}, 3.6/\sqrt{4\pi}, 2/\sqrt{4\pi})
 \end{aligned}$$

on the left-hand side and

$$(1, 0, 0, 0, 1, 2/\sqrt{4\pi}, 4/\sqrt{4\pi}, 2/\sqrt{4\pi})$$

on the right-hand side. The code is run for a time $0.2L$. The results again agree with Ryu et al. (1998). The following features may be seen: fast shocks at $x \simeq 0.3$ and 0.9 , rotational discontinuity at $x \simeq 0.53$ right next to a slow shock at $x \simeq 0.55$, contact discontinuity at $x \simeq 0.6$, slow shock and rotational discontinuity at 0.68 and 0.70 , respectively.

The shock tests have two basic types of structures: self-steepening shock fronts and contact and rotational discontinuities. For the active shock fronts, Figures 6 and 7 show the shocks with a nominal shock resolution of around two grid cells. The internal contact discontinuities arise from the discontinuities in the initial conditions, and diffuse numerically as they advect over the grid. At discontinuities, the solution is nondifferentiable, which is in general a challenge to numerical schemes. In the TVD approach, the scheme drops to first-order accuracy, with some associated diffusivity. This is discussed in more detail in Trac & Pen (2003a). The shock fronts are self-steepening, so the first-order diffusivity is less noticeable.

5. DISCUSSION OF THE MERITS AND DRAWBACKS OF OUR NUMERICAL SCHEME

The code described in this paper differs from its predecessors (TVD method for shock capturing, enforcement of $\nabla \cdot \mathbf{b} = 0$ to machine precision) in two main respects. First, we solve the induction equation in two-dimensional

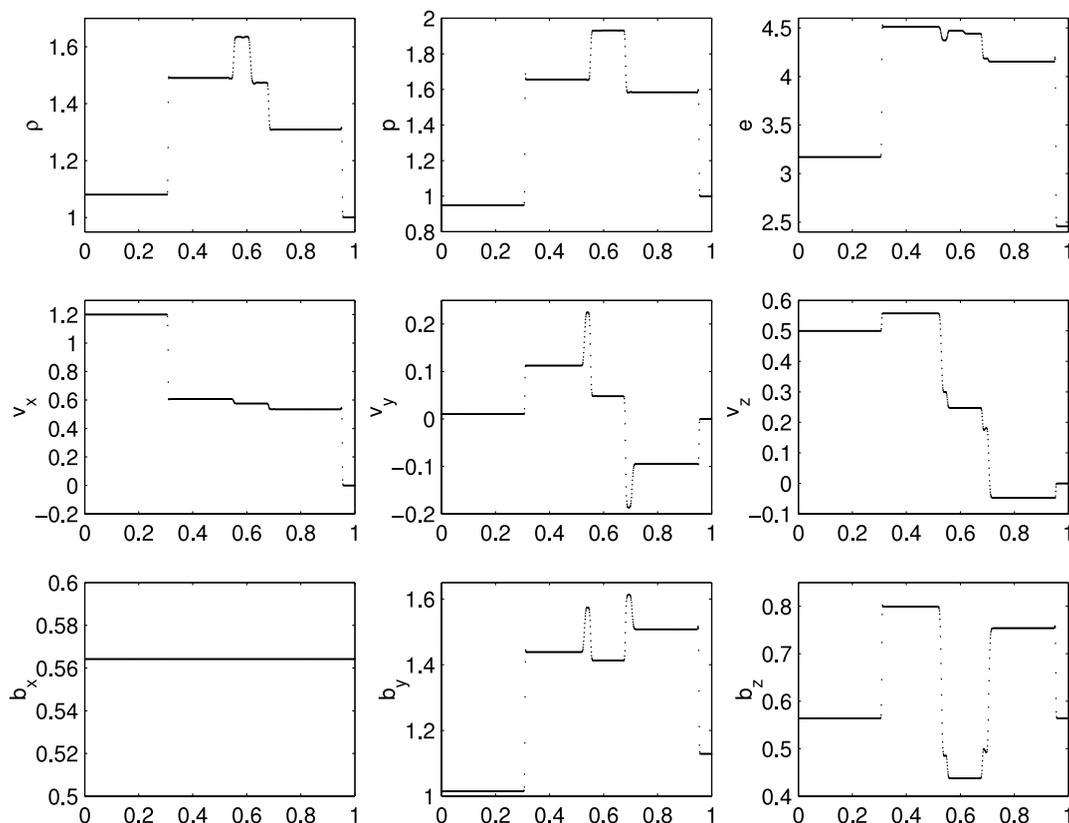


FIG. 7.—Shock tube test along the x -axis with velocities and magnetic field in all three directions

advection-constraint steps without storing intermediate fluxes over the entire grid. Storage of the fluxes would require a $3n^3$ array, nearly half the memory used for the basic variables. Second, we use Jin & Xin's (1995) method to implement TVD, which requires only ~ 30 floating point operations per grid cell per time step per variable. The benefits of these two methods are high-resolution per grid cell, low operations count, and simplicity of coding (the public version of the three-dimensional code is only 400 lines long).

The price of separating the fluid and magnetic field updates into two steps, rather than updating all variables at once, is that the coupling between the velocity and magnetic field may be relatively weak compared with other methods. It was this point which led Stone & Norman (1992) to use an update along Alfvén characteristics. However, our tests on both linear and nonlinear waves, as well as shocks, seem to indicate that in most circumstances the code performs rather well and no instabilities arise.

There are regimes in which we know the current code to be inaccurate or unstable. The generic setting is one in which the characteristic families have very different velocities. These can occur at low β for large-amplitude Alfvén waves, or highly supersonic flow with weak embedded shocks. For such regimes, customized modifications to the algorithm may be necessary (for the high Mach number regime see, e.g., Trac & Pen 2003; for the low- β case see Turner et al. 2003). Also, since Jin & Xin's (1995) method does not explicitly evolve the fluid variables along characteristics, our code may be more diffusive for low-frequency waves when the ratio of fast and slow wave speeds is large (either

large or small β). As can be seen in Figure 3, the slow waves have the largest errors. They still converge at second-order accuracy.

6. CONCLUSIONS

We have presented the algorithm and tests for a simple and robust MHD code that incorporates all features of modern high-resolution shock capturing. It is second-order-accurate away from extrema, requires no memory overhead beyond storing the fluid variables, optimizes easily to many computer architectures, and offers simplicity in the coding. We have tested this code on linear and nonlinear MHD waves as well as shocks. The single-processor source code can be freely downloaded from the electronic edition of the *Astrophysical Journal Supplement* or at our Web site.⁸ We have also implemented a parallel version, which scales well on very large commodity beowulf clusters.

We thank Neal Turner for several useful discussions. Phil Arras is an NSF Astronomy and Astrophysics Postdoctoral Fellow. This material is based upon work supported by the National Science Foundation under grant 0201636. Computing resources were provided by the Canada Foundation for Innovation. S. W. is supported by the Taiwan NSC.

⁸ See <http://www.cita.utoronto.ca/~pen/MHD>.

APPENDIX

ADVECTION IN ONE DIMENSION

Here we review Jin & Xin's (1995) solution of the advection equation. We will focus on a scalar equation, but extension to a vector equation is straightforward. A more detailed discussion as well as code has recently been published by Trac & Pen (2003a).

The advection equation for a quantity u with flux f is

$$\partial_t u + \partial_x f = 0. \quad (\text{A1})$$

Jin & Xin's (1995) symmetric method is to define the new variable $w = f/c$ and equations for u and w

$$\begin{aligned} \partial_t u + \partial_x (cw) &= 0, \\ \partial_t w + \partial_x (cu) &= 0. \end{aligned} \quad (\text{A2})$$

These equations can be written in terms of left and right moving variables by defining $u_r = (u + w)/2$ and $u_l = (u - w)/2$. These variables satisfy the equations

$$\begin{aligned} \partial_t u_r + \partial_x (cu_r) &= 0, \\ \partial_t u_l - \partial_x (cu_l) &= 0, \end{aligned} \quad (\text{A3})$$

which describe information propagating to the right and left, respectively.

To solve equation (A3) over a full time step with second-order accuracy, we first advance u_r and u_l over a half time step using the first-order upwind donor cell formula. These values are then used to construct a second-order-accurate upwind flux using any of the known nonlinear TVD limiters such as minmod, van Leer, or superbee. Finally, given the updated values for u_r and u_l , we reconstruct $u = u_r + u_l$.

For stability, the value of the flux freezing speed c must be chosen larger than the speed at which information propagates. As discussed in the text, we set $c = |v|$ when advecting the magnetic field, and $c = \text{cfl}[|v| + (\gamma p/\rho + b^2/\rho)^{1/2}]^{-1}$.

How can one relate TVD to the "artificial viscosity" schemes? These schemes add in a nonlinear viscosity term in order to prevent instabilities, as well as damp away oscillations that may occur near discontinuities. However, this viscosity tends to prevent the formation of discontinuities on scales of order one cell, severely degrading the resolution of the simulation. TVD may be viewed as a strongly nonlinear flux limiter that adds just enough diffusion to prevent numerical instabilities. TVD can often capture shocks in only one or two cells. Away from discontinuities, maxima, or minima, TVD is second order in space, but at a maxima it is only first order.

REFERENCES

- Balsara, D., & Spicer, S. S. 1999, *J. Comput. Phys.*, 149, 270
 Dai, W., & Woodward, P. R. 1994, *J. Comput. Phys.*, 111, 354
 ———. 1998, *J. Comput. Phys.*, 142, 331
 Derby, N. F. 1978, *ApJ*, 224, 1013
 Evans, C. R., & Hawley, J. F. 1988, *ApJ*, 332, 659
 Goldstein, M. L. 1978, *ApJ*, 219, 700
 Harten, A. 1983, *J. Comput. Phys.*, 49, 357
 Jin, S., & Xin, Z. 1995, *Commun. Pure Appl. Math.*, 48, 235
 Landau, L. D., & Lifshitz, E. M. 1984, *Electrodynamics of Continuous Media* (Oxford: Pergamon)
 Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1996, *Numerical Recipes in FORTRAN 90: The Art of Parallel Scientific Computing* (2d ed.; Cambridge: Cambridge Univ. Press)
 Ryu, D., Miniati, F., Jones, T. W., & Frank, A. 1998, *ApJ*, 509, 244
 Stone, J. M., & Norman, M. L. 1992, *ApJS*, 80, 791
 Tóth, G. 2000, *J. Comput. Phys.*, 161, 605
 Trac, H., & Pen, U. 2003, *PA&P*, 115, 303
 ———. 2003b, *NewA*, Submitted (astro-ph/0309599)
 Turner, N. J., Stone, J. M., Krolik, J. H., & Sano, T. 2003, *ApJ*, 593, 992