A 3D MHD code for Parallel Computers

Å. Nordlund & K. Galsgaard
NBIfAFG, Astronomical Observatory
Øster Voldgade 3
1350 Copenhagen K
Denmark

August 15, 1995

Abstract

We describe a new 3D MHD code, written in High Performance Fortran (HPF). Although developed specifically for massively parallel computers, the code also performs well on vector-supercomputers and scalar computers. The code uses staggered meshes and conserves mass, momentum, energy, and magnetic field divergence. Artificial diffusion and discontinuity capturing are used to achieve high effective resolution for a given number of mesh points.

Key words: MHD, Algorithms, High Performance Fortran

1 Introduction

Massively parallel supercomputers offer a route to scalable performance, beyond the limitations of traditional vector-supercomputers. Even though the initial enthusiasm for the massively parallel processing (MPP) concept has receded somewhat, all current supercomputers are parallel to some extent, and the development towards an increasing degree of parallelism is inevitable.

Our original motivation for developing this code was the Danish acquisition of a Thinking Machines Corporation CM-200 computer, a Single Instruction Multiple Data (SIMD) machine with 8192 processors (256 floating point units) connected in a hyper-cube structure. Each of the 8k one-bit processors has 128kb of local memory, for a total of 1 Gb. The hyper-cube structure gives a maximum communication length of 13 CPU transitions for any two bits.

In the "slice-wise" mode of operation, one may consider the machine to consist of 256 floating point CPUs, each with 4 Mb of local memory. Arrays with more than 256 elements are wrapped around the hyper-cube, with a number of array elements residing in the same local memory. This reduces the communications cost, and leads to a favorable scaling of performance with problem size. We have subsequently used the code on larger CM-5 machines, with up to 512 Sparc CPUs. For any given number of CPUs, a larger problem size results in a smaller relative amount of time being spent on communications.

The CM-200/CM-5 class of machines supports CM Fortran, a dialect of Fortran 90 and a precursor to High Performance Fortran (HPF). If certain CM particular constructs are avoided, a program written in CM Fortran also compiles successfully on a standard Fortran 90 compiler. We have taken care to follow this philosophy and—apart from I/O routines particular to use of the CM "data vault", and optional routines that utilise CMSSL-library versions of shift operators—the code is written in generic Fortran 90. The Fortran 90 array syntax, and in particular the use of array-valued functions to represent mathematical operators, results in a very compact code, with a close correspondence to the mathematical formulation.

In order to be able to use the code on computers that only have Fortran 77, we have subsequently translated the code to Fortran 77. The f77 version of the part of the code that implements the MHD partial differential equations is over three times longer than the corresponding Fortran 90 code. Because of the compactness and clarity of the HPF version of the code, it is less cumbersome to maintain, easier to debug, and allows for a much quicker implementation of new pieces of physics and alternative boundary conditions.

SIMD computers such as the Connection Machines are optimal for problems that only have local information passing between closely neighboring grid points. For programs to take maximal advantage of this type of computer the problem needs to be highly parallel, and the methods used must result in a suitably low ratio between communication and computations. The type of 3D MHD problems that we address fulfill the first criterion, when solved by an explicit method. The second criterion is fulfilled because partial derivatives are determined locally, and only involve a small number of neighboring grid points.

2 Equations

We solve the standard MHD equations in the following form:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{u}, \tag{1}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \tag{2}$$

$$\mathbf{E} = -(\mathbf{u} \times \mathbf{B}) + \eta \mathbf{J}, \tag{3}$$

$$\mathbf{J} = \nabla \times \mathbf{B}, \tag{4}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla \cdot (\rho \mathbf{u}\mathbf{u} + \underline{\underline{\tau}}) - \nabla P + \mathbf{J} \times \mathbf{B} - g\rho, \tag{5}$$

$$\frac{\partial e}{\partial t} = -\nabla \cdot (e\mathbf{u}) - P\nabla \cdot \mathbf{u} + Q_{cool} + Q_{visc} + Q_{Joule}, \tag{6}$$

where ρ, $\mathbf{u}$, $\mathbf{B}$, $\mathbf{E}$, η, $\mathbf{J}$, $\underline{\underline{\tau}}$, e, g, $P = (\gamma - 1)e = \frac{2}{3}e$, $T = p/\rho$, Q_{cool}, Q_{visc} and Q_{Joule} are the density, velocity, magnetic field, electric field, electric resistivity, electric current, viscous stress tensor, internal energy, constant of gravity, gas pressure, temperature, cooling term, viscous dissipation, and Joule dissipation, respectively. The exact form of the cooling term varies with the experiment, but a simple form

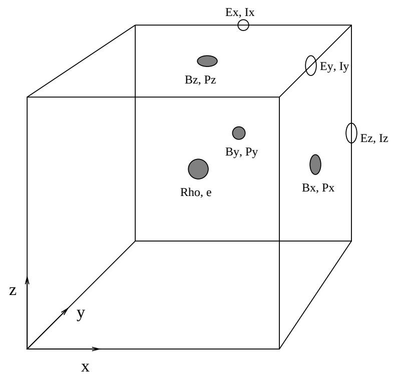

Figure 1: The positions of the physical quantities in a unit mesh volume. ρ and e are volume centered, $\mathbf{B}$ and $\rho\mathbf{u}$ are face centered, and $\mathbf{E}$ and $\mathbf{J}$ are edge centered.

used in many experiments is

$$Q_{cool} = -\rho(T - T_{ref})/t_{cool}, \tag{7}$$

where T_{ref} is a reference temperature and t_{cool} is a cooling time.

Note that we have non-dimensionalised the equations by setting the magnetic permeability $\mu_0 = 1$, the gas constant $R_0 = \mu$ (the mean molecular weight). Dimensional quantities may be obtained from

$$\begin{aligned}
u_0 &= \ell_0/t_0, & (8)\\
e_0 &= \rho_0 u_0^2, & (9)\\
B_0 &= u_0\sqrt{\mu_0\rho_0}, & (10)\\
T_0 &= \mu u_0^2/R_0, & (11)\\
E_0 &= u_0 B_0, & (12)\\
j_0 &= B_0/(\mu_0\ell_0) = \sqrt{(\rho/\mu_0)}/t_0, & (13)
\end{aligned}$$

where ℓ_0 is the unit of length, t_0 is the unit of time, etc., with $\mu_0^{SI} = 4\pi 10^{-7}$ $\mathrm{V\,s\,A^{-1}m^{-1}}$, and $R_0^{SI} = 8.3143$ $\mathrm{J\,mol^{-1}K^{-1}}$.

We often use initial conditions where $\langle\rho\rangle = 1$ and $\langle B^2\rangle = 1$, in a cubic volume of unit size, and in these cases another way of expressing the non-dimensionality is to say that we are measuring lengths in units of the size of the box, times in units of the Alfvén travel time, mass in units of the mass in the box, energy densities in units of twice the initial magnetic energy density, and so on.

2.1 Staggered Meshes, Discretization

The variables are represented on staggered meshes, where—in terms of a unit cube where ρ and e are volume centered—$\mathbf{B}$ and $\rho\mathbf{u}$ are face centered, and $\mathbf{E}$ and $\mathbf{J}$ are edge centered, as illustrated in Fig. 1.

We use a sixth order accurate method—involving the three nearest neighbor points on each side—for determining partial derivatives. The result is returned at a point which is shifted half a grid point up or down relative to the input values. This is indeed often the point where the result is needed, and this circumstance is the main advantage with the staggered grid method in the first place. Coefficients, variables and derivatives are sometimes needed at other positions. We use a fifth order accurate interpolation method, again involving the same six neighboring points, to "shift" the result into place. The interpolation and differentiation operators are computationally equivalent, and differ only in the scalar coefficients applied at the six grid points. The two classes of operators, the three directions in space, and the two shift directions result in a total of twelve numerical operators; $\partial^{\pm}_{,[xyz]}()$ and $T^{\pm}_{[xyz]}()$. Eq. (14) and Eq. (16) show examples of the $T^{+}_{x}()$ and the $\partial^{+}_{,x}()$ interpolation and derivative operators, that return the values shifted $+\frac{1}{2}$ a grid point in the x direction. Other operators are obtained through simple permutations of the indices.

$$
\begin{aligned}
T^{+}_{x}(f_{i,j,k}) &= f_{i+\frac{1}{2},j,k} = && (14)\\
&\quad a(f_{i,j,k}+f_{i+1,j,k}) + b(f_{i-1,j,k}+f_{i+2,j,k}) + c(f_{i-2,j,k}+f_{i+3,j,k}), &&\\
a &= \tfrac{1}{2} - b - c, \quad b = -\tfrac{1}{16} - 3c, \quad c = \tfrac{3}{256}, && (15)\\
\partial^{+}_{,x}(f_{i,j,k}) &= f'_{i+\frac{1}{2},j,k} = && (16)\\
&\quad \frac{a}{\Delta x}(f_{i,j,k}-f_{i+1,j,k}) + \frac{b}{\Delta x}(f_{i-1,j,k}-f_{i+2,j,k}) + \frac{c}{\Delta x}(f_{i-2,j,k}-f_{i+3,j,k}), &&\\
a &= 1 - 3b + 5c, \quad b = -\tfrac{1}{24} - 5c, \quad c = \tfrac{3}{640} && (17)
\end{aligned}
$$

(setting $c = 0$, or $b = c = 0$ produces 3rd or 1st order accurate interpolation, and 4th or 2nd order accurate derivatives, respectively).

The choice of a fifth order interpolation and a sixth order derivative algorithm (instead of methods with lower order and less communication) was based on tests that showed an enhanced noise level at wavenumbers close to the Nyquist wavenumber with lower order operators. To suppress this noise requires higher diffusion coefficients, resulting in an effectively lower resolution. On balance, using the higher order expressions is more efficient, both in terms of computer time, memory, and disc space.

The straight forward way of coding these routines requires that data is communicated between neighbouring grid points 9 times. By using two temporary arrays, holding variables shifted $\pm$ one grid point in each direction, the required number of communications is reduced to 5. Such optimizations may be of significant importance for machine with distributed memory (such as the CM). However, for large enough data sets, most of the communication is internal to the local CPU and the importance of of these optimizations diminishes.

2.2 Boundary Conditions

For non-periodic models, boundary conditions must be applied to handle the physical truncation of the model. The SIMD and hyper-cube architectures have important implications for the implementation of boundary conditions, and we focus on these implications in the following paragraphs. Note that optimizing the handling of boundary conditions for these architectures does not necessarily imply compromising the performance on scalar or vector machines; we find that the adopted methods still perform well on these more traditional machines.

The wrapping of data points that corresponds to the the hyper-cube memory architecture implies that periodic boundary conditions come "for free", while other boundary conditions must be implemented in ways that are consistent with the periodic data structure. The SIMD architecture implies that the same formulae must be used at all points. In our case, this means that the boundary conditions should be "built into" the interpolation and differentiation formulae. This may be accomplished either by using space dependent coefficients (a, b and c in Eqs. (14) and (16)), or by using constant coefficients, and instead add ghost zones outside the boundaries, to prevent the periodic wrapping from interfering with the boundaries.

We chose to introduce ghost zones, for reasons of simplicity, and to conserve memory. The fraction of memory occupied by ghost zone data shrinks with increasing problem size, whereas a number of spatially dependent coefficient would require a constant fraction of the memory. With the use of ghost zones it becomes straight forward to implement boundary conditions in the form of symmetry conditions across the boundaries.

Using symmetry conditions to implement boundary conditions is analogous to the method of reflecting charges and / or currents in electrodynamics; a superconducting surface may be represented by using mirror images of the currents, for example. In general, we need to be able to impose conditions on the variable values at the boundaries, or on partial derivatives of variables across the boundaries.

If we let the x coordinate measure the distance from the boundary, the required conditions and the corresponding symmetry conditions are the following:

condition at $x_b = 0$	symmetry condition
$f = 0$	antisymmetric f
$f = f_b$	antisymmetric $f - f_b$
$\frac{\partial f}{\partial x} = 0$	symmetric f
$\frac{\partial f}{\partial x} = f_b'$	symmetric $f - x f_b'$

The function values in the ghost zones are determined in such a way that the given boundary conditions are fulfilled when the standard expressions (14) and (16) are evaluated. The number of nearest neighbours that enter into the derivative and interpolation formulae determine how thick the ghost zones have to be. For our choice of fifth and the sixth order accurate methods, and with half-point values $f_{i-\frac{1}{2},j,k}$ stored at index (i, j, k), the boundaries should be at the full-point indices 4 and $n_x - 4$ (for one-indexed arrays). Fig. 2 shows how full and half-point centered variables are positioned relative to the x boundaries.

The boundary algorithms are organized into a small number of routines that operate on either full-point or half-point centered variables. The $f(x) = 0$ condition, f.ex., is handled by making the function antisymmetric with respect to the boundary. This is done by mirroring values from inside the boundaries into the ghost zone, and change their sign, e.g.:

$$f_{b-i} \quad = \quad -f_{b+i}; \qquad i = 1, ..., b-1, \tag{18}$$

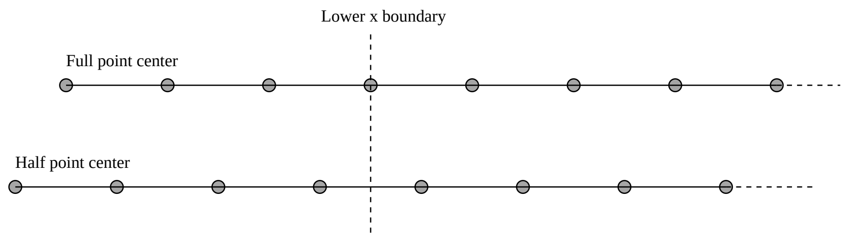

Figure 2: A 1D illustration of the placement of full-point and half-point centered variables relative to the boundaries.

$$\begin{aligned} f_b &= 0, \\ f_{b-\frac{1}{2}-i} &= -f_{b-\frac{1}{2}+i}; \qquad i = 0, ..., b-1, \end{aligned} \tag{19}$$

where $b(=4)$ is the boundary index value (cf. Fig. 2).

The $\frac{\partial f}{\partial x} = 0$ condition is implemented by making the function symmetric across the boundary;

$$f_{b-i} = f_{b+i}; \qquad i = 1, ..., b-1, \tag{20}$$

$$f_{b-\frac{1}{2}-i} = f_{b-\frac{1}{2}+i}; \qquad i = 0, ..., b-1. \tag{21}$$

When a function should continue monotonically across the boundary, the function values are extrapolated into the ghost zone. If the boundary value is known, this is achieved by antisymmetrising the function around the boundary value:

$$f_{b-i} = 2f_b - f_{b+i}; \qquad i = 1, ..., b-1, \tag{22}$$

where f_b can either be a constant value or a function of position. Other cases are handled analogously.

A non-trivial case is when a half-point centered function needs to be extrapolated, and the boundary value is not known a priori. We have tried several possibilities and found that the most reliable way to determine the half-point centered extrapolated ghost zone values in this case is to estimate the derivative at the boundary, f'_b, by using a second order polynomial defined by the three points just inside the boundary, and then symmetrise $f(x) - xf'_b$:

$$f_{b-\frac{1}{2}-i} = f_{b-\frac{1}{2}+i} - (i - \tfrac{1}{2})\, 2f'_b \Delta x; \qquad i = 0, ..., b-1. \tag{23}$$

Gravity contributes an external force in the momentum equation, $-\rho g$, where g in our calculations is taken to be constant. We use the force of gravity (and other external forces if present) to determine the pressure gradient at the boundary,

$$p_{b-i} = p_{b+i} + i\, 2g\rho_b \Delta x; \qquad i = 1, ..., b-1, \tag{24}$$

where Δx is the signed grid spacing, being positive at the bottom boundary and negative at the top and ρ_b is the density at the boundary layers.

Several experiments with slightly modified boundary conditions relative to the ones discussed here were made. Most of the effort was put into trying other methods for extrapolation of variables. Experiments with second and third order polynomial extrapolations all failed tests where different types of MHD waves were expected to be reflected off impenetrable walls, while the simple "mirroring" of function values works well.

All of the boundary condition formulae imply communications over several layers in the x-direction. The same optimization considerations as mentioned earlier (Section 2.1) apply; i.e., the communications cost may be minimized by caching intermediate values. For a SIMD architecture, the handling of boundary layers is as costly as the handling of the interior, and such optimizations are therefore quite important.

2.3 Boundary Driving

In some of the applications of this code, we wish to perform work on a magnetic field that penetrates a boundary (Galsgaard & Nordlund 1995a; Galsgaard & Nordlund 1995b; Galsgaard 1995). The boundary driving is accomplished by specifying the velocity fields in the x boundary planes, $v_{y,z}(y,z)$. In principle, this determines the evolution of the perpendicular component of the magnetic field at the boundary, by specifying the components of the electric field parallel to the boundary. In practice, because of the non-local nature of the derivative and staggering operators, it is necessary to couple the velocity field in the mesh plane next to the boundary to the boundary velocity, to avoid introducing numerical ringing at the boundary. We do this by passing a third order polynomial through the boundary point and the next two points inside. This gives the following relation for the velocity in the 5th layer (index 4 is the boundary),

$$\mathbf{u}_5 = \frac{4}{9}\mathbf{u}_4 + \frac{6}{9}\mathbf{u}_6 - \frac{1}{9}\mathbf{u}_7. \tag{25}$$

This algorithm was found by testing various polynomial representations and choosing the one that gave the smoothest driving.

3 Artificial Diffusion

The disparity between the smallness of diffusive effects such as viscosity, heat conduction and electrical resistivity in real astrophysical systems, and the size of the corresponding diffusive effects in numerical models of these systems has often been a cause of concern and confusion. A number of concepts, such as "turbulent diffusion", "artificial diffusion", "hyper diffusion", or "sub-grid scale diffusion" have been introduced to deal with the situation, with justifications ranging from mathematical turbulence closure theories, over empirical measurements of transport in turbulent fluids, to more or less arbitrary numerical considerations such as monotonicity. Distinctions are often made between Direct Navier Stokes (DNS) (or even Euler) simulations, and Large Scale Eddy (LSE) simulations.

DNS simulations represent the extreme conservative point of view; i.e., one solves, as accurately as possible, the correct Navier-Stokes equations, with constant diffusion coefficients (viscosity and conduction) large enough for the solution to remain everywhere well resolved at all times. Purely numerical techniques, such as the SHASTA method by Boris and Book, represent the other extreme, where one attempts to "tame" the numerical behavior, such that undesirable numerical behavior is prevented.

Each of these extremes has disadvantages. The DNS approach is wasteful in numerical resolution, even for incompressible or compressible sub-sonic cases, and can only handle transonic flows and shocks if the diffusive coefficients are made very large. The purely numerical techniques, on the other hand, sometimes produce incorrect solutions, or artifacts, in situations that differ from the ones the methods were originally designed to handle.

A successful line of approach has been to utilize "exact", analytical or numerical solutions, e.g. Riemann solvers, to evolve a local approximation to the solution (Godunov 1959, van Leer 1977, van Leer 1979, Colella & Woodward 1984a, Woodward & Colella 1984), often using knowledge of the local characteristics (describing the zones of influence of various wave modes) and upwind differencing. A disadvantage with these methods is that, since new types of wave modes and characteristics are introduced when additional forces are included in the simulations, new Riemann solvers have to be developed for every new situation (Dai & Woodward 1994a; Dai & Woodward 1994b). Versions have yet to be developed that can use generalized Reimann solvers in situations with simultaneous action from gas pressure, gravity, rotation and magnetic fields; with all ingredients being of comparable significance, as—for example—in the solar convection zone.

The upwind differencing is often combined with monotonicity constraints, to prevent new maxima from developing in the neighborhood of discontinuities. The net effect is to produce a behavior that is locally, in the neighbourhood of shocks, very dissipative, while at the same time highly accurate differencing operators allow very low diffusion to be used in regions with smooth behaviour.

The approach we take is to make explicit this use of highly localized diffusivities, while basically retaining the original form of the partial differential equations. We select a number of model situations, where the physically correct solution is obtainable from first principles, discuss the qualitative behavior of numerical solutions, and, based on this discussion, conclude what the appropriate form of the numerical diffusion is in each particular case. We then combine these diffusive effects into a scheme that (ideally) can handle any conceivable situation without developing numerical instabilities, and yet is as non-diffusive as possible in situations that only require very weak dissipative effects.

3.1 Sound Waves

The model situation here is a constant temperature ideal gas, with propagating, small amplitude sound waves of varying wavelength. We assume that the real viscosity and heat conduction are negligible. The isothermal sound speed is given by

$$c_s^2 = \gamma P/\rho, \tag{26}$$

and the solutions are constant amplitude waves, travelling without attenuation.

The staggered mesh is particularly well suited for this situation, with the staggered derivative of the pressure delivering the acceleration with maximum accuracy at half-integer points, and with the resulting momentum (i.e. mass flux) entering in the divergence term of the continuity equation, thus causing a correctly (integer) centered time rate of change of the mass density, feeding back into the equation of motion through the pressure.

The closed causal loop results, effectively, in the numerical equation

$$\frac{\partial^2 \rho}{\partial t^2} = c_s^2 \partial^+_{,x}(\partial^-_{,x}(\rho)). \tag{27}$$

Because of the sixth order accuracy of the differentiation operators, the right hand side is

$$c^2 \partial^+_{,x}(\partial^-_{,x}(\rho)) = -c_s^2 k^2 \rho + \mathrm{O}((k\Delta x)^7). \tag{28}$$

Thus, for anything but very short wavelengths ($\lambda/\Delta x \approx$ a few), truncation errors are negligible, and the propagation speed is accurate (assuming that the time stepping method is sufficiently exact).

On the other hand, the truncation error does grow very rapidly when k approaches the Nyquist wavenumber, $k_N = \pi/\Delta x$, with a progressively smaller and smaller result returned from $\partial^+_{,x}(\partial^-_{,x}(\rho))$, relative to the correct result $k^2\rho$. Thus, the numerical propagation speed decreases rapidly as k approaches k_N. This corresponds to a *phase error* in the evolution. One can show that the amplitude error is negligible compared to the phase error; a short wavelength wave propagates with little damping, but with an incorrect phase speed (a "saw-tooth" wave with $k = k_N$ has no $sin(kx)$ representation, and hence does not propagate at all).

A non-sinusoidal sound wave, for example a numerical approximation to a square wave, has non-vanishing amplitudes in all of the discrete wavenumbers. An incorrect propagation speed for the shortest wavelengths becomes visible as a "ringing" at the corners of the square wave, because of the numerical dispersion of the various wavelengths. To counter this effect, the amplitude of the wavenumbers with incorrect wave speeds must be damped at a rate that is comparable to the growth of the phase error.

Viscous second order diffuse terms, whether numerical or physical, introduce a damping of the travelling waves, with an e-folding distance inversely proportional to the viscosity;

$$\ell_e = \frac{c}{\nu k^2} = \frac{c\lambda^2}{4\pi^2\nu} = \frac{\lambda}{2\pi\upsilon}\left(\frac{\lambda}{2\pi\Delta x}\right), \tag{29}$$

if ν is given in terms of a dimensionless coefficient υ,

$$\nu = \upsilon\ c\Delta x. \tag{30}$$

Thus, in terms of wavelengths, the damping length is proportional to λ/υ. This immediately shows that a DNS method is far from ideal in the treatment of sound waves. For, if the damping of waves with wavenumbers close to the Nyquist wavenumber is to be significant (a damping length of four wavelengths at $\lambda = 3$, say), then the damping is also noticeable for well resolved wavelengths.

For example, in an experiment with 128 grid points (a non-trivial experiment in three dimensions), a wave with a wavelength equal to a quarter of the box is then damped by about 10% while travelling across the box. In contrast, the intrinsic phase error due to the truncation error of the sixth order derivatives is completely negligible for $\lambda = 30\Delta x$, and such a wave should in principle be allowed to travel virtually undamped.

The discussion above illustrates the need to use a diffusion operator that decreases more rapidly with wavelength. The next possible operator that has a qualitatively correct behavior is a fourth order diffusion operator, resulting from taking the divergence of a viscous flux proportional to the third order derivative of the velocity. Such a term would have dropped with two orders of magnitude more in the example discussed above; i.e., the damping of a $\lambda = 30\Delta x$ wave would only be 0.1% across the box, if the damping of a $\lambda = 3\Delta x$ wave was retained at one e-folding per four wavelengths.

The diffusion coefficient corresponding to $\ell_e = 4\lambda$ for $\lambda = 3$ is, according to Eq. (29),

$$\nu = \frac{c\lambda^2}{4\pi^2\ell_e} = \frac{3}{16\pi^2}c\Delta x = v\ c\Delta x, \tag{31}$$

where $v = 3/16\pi^2 =\approx 0.02$ is a dimensionless viscosity. The damping length with fourth order diffusion is

$$\ell_4 = \frac{c}{\nu_4 k^4} = \frac{c\lambda^4}{16\pi^4\nu_4} = \frac{\lambda}{2\pi v_4}\left(\frac{\lambda}{2\pi\Delta x}\right)^3, \tag{32}$$

where v_4 is the corresponding dimensionless diffusion coefficient.

Thus, with fourth order diffusion and the same dimensionless diffusion coefficient as for second order diffusion, waves with wavelengths shorter than $2\pi\Delta x$ are damped more, while longer waves are damped less. For well resolved waves, the result is a significant reduction of the damping.

We conclude that a high order diffusion operator is desirable, but postpone until a later section a discussion of exactly how to choose this operator in order to ensure a physically correct behavior for arbitrary wave packets. For now, we merely introduce the notation $d^+_{,x}()$ for the operator, and refer to it loosely as a "hyper-diffusive" operator. The corresponding contribution in the momentum equation is

$$\frac{\partial(\rho u_x)}{\partial t} = -\partial^-_{,x}(... - \rho v_1 c\Delta x d^+_{,x}(u_x)) - ... \tag{33}$$

3.2 Fast Mode Waves

Magneto-acoustic (fast mode) waves are analogous to sound waves, with an additional compressional force arising from the compression of magnetic field lines perpendicular to the direction of propagation. In our numerical scheme, magnetic compression contributes terms similar to

$$\frac{\partial(\rho u_x)}{\partial t} = ... - T_x^-(B_y)\partial_{,x}^-(B_y) + ... \tag{34}$$

in the equations of motion, and

$$\frac{\partial B_y}{\partial t} = ... - \partial_{,x}^+(u_x T_x^-(B_y)) \tag{35}$$

in the induction equation. The complete causal loop for small amplitude fast mode waves thus contains the terms

$$\begin{aligned} \frac{\partial^2 u_x}{\partial t^2} &= c_A^2 \partial_{,x}^-(\partial_{,x}^+(u_x)) + c^2 \partial_{,x}^-(\partial_{,x}^+(u_x)) + ... \\ &= c_f^2 \partial_{,x}^-(\partial_{,x}^+(u_x)) + ... \end{aligned} \tag{36}$$

where

$$c_A = \left(\frac{B^2}{\rho}\right)^{\frac{1}{2}} \tag{37}$$

is the Alfvén speed, and

$$c_f = \left(\frac{B^2 + \gamma P}{\rho}\right)^{\frac{1}{2}} \tag{38}$$

is the fast mode speed.

Thus, also from the point of view of our numerical scheme, fast mode waves are completely analogous to sound waves. The analysis of the previous section carries over, with the fast mode speed replacing the sound speed.

3.3 Advection

A complementary limiting case is *advective transport* of properties. The material motion may be slower or faster than the local wave speed, depending on the frame of reference. In the post shock region of a shock travelling into a stationary medium, for example, the material speed exceeds the local wave speed.

The material transport of small amplitude scalar fluctuations, for example the translation of temperature and density fluctuations at constant pressure with constant speed, is described numerically by the continuity equation

$$\frac{\partial \rho}{\partial t} = -\partial_{,x}^+(T_x^-(\rho)u_x) - ... \tag{39}$$

and the energy equation

$$\frac{\partial e}{\partial t} = -\partial_{,x}^+(T_x^-(e)u_x) - ... \tag{40}$$

In the simplest case, with constant velocity u_x and constant pressure P, the energy equation is redundant, since $e = 3P/2$ is constant as well.

The complete causal loop for small amplitude fluctuations in this case is thus

$$\frac{\partial \rho}{\partial t} = -u_x \partial^+_{,x}(T^-_x(\rho)) - ... \tag{41}$$

Because of the presence of the $T^+_x()$ interpolation operator, which is only fifth order accurate, this equation has amplitude errors that are comparable to the phase errors (which again are comparable to the phase errors in the case of compressional waves).

A discussion analogous to the one for compressional waves shows that the hyper-diffusion contribution should scale as

$$\frac{\partial \rho}{\partial t} = -\partial^+_{,x}(... - v_2 \Delta x |u_x| d^-_{,x}(\rho)) - ..., \tag{42}$$

where v_2 is a dimensionless constant similar in size to v_1.

3.4 Shocks

The third type of archetype situation to consider is the region of strong compression in a shock. As with compressional waves, we first discuss the purely hydrodynamical case, and then generalize to the MHD case.

The numerical representation of a shock should satisfy the same conditions as the idealised shock it corresponds to: It should be a near-discontinuity, across which the individual variables change rapidly but certain invariants are conserved. The Rankin-Hugoniot relations (Landau & Lifshitz (1987)) are consequences of the conservation of mass, momentum and energy, and are thus automatically satisfied (as a consequence of the conservation of these quantities) in our numerical scheme.

It remains to ensure that shocks do not generate numerical artifacts; e.g., fluctuations in the post shock material. Such fluctuations are easily generated if the dissipation in the shock is insufficient to keep the shock numerically resolved.

To estimate the required diffusion, consider the equation of motion in a region around a shock travelling into a stationary medium. The numerical form of the equation for the x-momentum is

$$\frac{\partial (\rho u_x)}{\partial t} = -\partial^-_{,x}(\rho (T^+_x(u_x))^2 + P - v_3 \rho \Delta x U \partial^+_{,x}(u_x)) - ..., \tag{43}$$

where v_3 is a dimensionless diffusion constant, and U is an as yet unspecified factor with the dimension of velocity. We know that the solution should be a profile $\rho u_x = \psi(x - u_s t)$, travelling with constant speed (the shock speed u_s), such that

$$\frac{\partial (\rho u_x)}{\partial t} = -u_s \frac{\partial \psi}{\partial x}. \tag{44}$$

Because of the jumps in both u_x and P across the shock, the viscous contribution must be of the order of $\rho \Delta u_x^2 \approx \rho \Delta u_x \Delta x \partial u_x / \partial x$, in order to balance the contributions from the momentum flux ρu_x^2 and the pressure jump (which is of the order of ρu_x^2 as well). We are here assuming, as is appropriate, that the steepness of the numerical shock front is such that $\Delta x \partial u_x / \partial x$ is of the order of u_x.

A viscous momentum flux of the type $-v_3 \rho \Delta x U \partial^{+}_{,x}(u_x)$ is thus able to balance the advection of momentum and the pressure gradient, provided $U \approx \Delta u_x$ and v_3 is of the order of unity. This corresponds to a local kinematic viscosity of the order of $\Delta u_x \Delta x$.

In order to both meet the requirement of a local kinematic viscosity of the order of $\Delta u_x \Delta x$ in shocks, and a much smaller viscosity elsewhere, we make the coefficient U proportional to the local rate of convergence, and obtain a shock viscosity

$$\nu^{(2)} = v_3 \Delta x_i^2 |\nabla \cdot \mathbf{u}|_{-}, \tag{45}$$

where $|...|_{-}$ denotes the absolute value of the negative part, and v_3 is a dimensionless coefficient of the order of unity.

Note that this local kinematic viscosity, which must be present in the shock, is much larger than either of the previously discussed viscosity contributions $v_1 c_s \Delta x$ and $v_2 u \Delta x$. The reason for this difference is that, whenever there is a negative divergence (positive convergence) of velocity, such as in a shock, the profile of any transported scalar (such as momentum per unit mass; i.e. velocity) tends to steepen with time, in proportion to the local rate of velocity convergence. It is indeed for this very reason that the shock viscosity should be proportional to the local rate of convergence. This effectively turns the shock viscosity on where it is needed, and nowhere else.

We can now appreciate why direct Navier Stokes (DNS) simulations must employ very large viscosities in order to cope with shocks. A constant kinematic viscosity would have to be everywhere equal to the values needed in the shocks, i.e., it would have to be of the order of $Mc\Delta x$ for a Mach M shock. In the non-shocking regions, this would be about two orders of magnitude more than we are using, even for small scale fluctuations. For large scale fluctuations, the ratio increases with another factor of $(\lambda/(2\pi\Delta x))^2$. In other words, with a constant kinematic viscosity, chosen as small as possible to cope with a Mach 2 shock, a $\lambda = 30\Delta x$ sound wave would have an e-folding distance of only about 0.4 waves (c.f. Eq. (29)). This shows that a DNS approach is completely out of the question for supersonic and / or super-Alfvénic flows.

3.5 Magnetic Diffusion

In analogy with the Eqs. (33) and (42), we also adopt a weak magnetic diffusivity

$$\eta^{(1)} = \frac{\Delta x}{\mathrm{Pr}_M}(v_1 c_f + v_2 |u|), \tag{46}$$

which multiplies a magnetic hyper-diffusion (detailed in the next sub-section). Pr_M is a magnetic Prandtl number, that scales the ratio of viscous to magnetic hyper-diffusion.

The expression for the magnetic shock diffusion accounts for the fact that the magnetic field is only influenced by the perpendicular part of the velocity field. The perpendicular velocity field transports the magnetic field basically as a passive scalar, subject to magnetic diffusivity. A generalization of the concepts discussed in the previous subsection thus leads us to adopt a magnetic shock resistivity proportional to the local convergence of the perpendicular velocity field;

$$\eta^{(2)} = \frac{v_3 \Delta x^2}{\mathrm{Pr}_M} |\nabla_\perp \cdot \mathbf{u}|_- . \tag{47}$$

Note that, while the hydrodynamic shock viscosity (45) only becomes significant in shocks, and vanishes in incompressible flows, (47) does not vanish everywhere in incompressible flows. Since the velocity along magnetic field lines in general is not constant, it follows that the divergence of the perpendicular velocity field in general is non-zero,

$$\nabla_\perp \cdot \mathbf{u} = \nabla \cdot \mathbf{u} - \hat{\mathbf{B}} \cdot \nabla \mathbf{u} \neq 0, \tag{48}$$

even for an incompressible flow.

The relatively large resistivity from this term is indeed appropriate, and has just the right size to halt the collapse of magnetic flux tubes near the limit of the numerical resolution. To see this, consider a a patch of magnetic flux that is initially well resolved, but is situated in a region of perpendicular convergence of the velocity field. As long as resistive effects are small, the magnetic field lines are carried passively by the perpendicular velocity field, and the field strength grows exponentially with time,

$$\frac{\partial B}{\partial t} = -B \, \nabla_\perp \cdot \mathbf{u}. \tag{49}$$

When the magnetic patch has concentrated to a size of a few numerical grid points, diffusion should become significant enough to balance the convergence. This requires

$$\eta k^2 B \approx B \, |\nabla_\perp \cdot \mathbf{u}|_- , \tag{50}$$

or

$$\eta \approx \Delta x^2 |\nabla_\perp \cdot \mathbf{u}|_- , \tag{51}$$

if we let $k \approx 1/\Delta x$ corresponds to a size of a few grid points. But Eq. (51) is the same as Eq. (47), apart from dimensionless constants of the order of unity.

3.6 Positive Definite Quenched Diffusion Operators

As discussed in sections 3.1 and 3.3 above, it would, in principle, be advantageous to use third order derivatives in the weakly diffusive terms that stabilize waves and advection. We must make sure, however, that the diffusion terms are physically meaningful, and—in particular—that the corresponding dissipation (of kinetic and magnetic energy) is positive definite.

The expression for the dissipation of kinetic energy into heat is obtained by taking the scalar product of the velocity with the equation of motion and integrating by parts,

$$Q_{diss} = \sum_{ij} \tau_{ij} \frac{\partial u_i}{\partial x_j} \tag{52}$$

where τ_{ij} is the viscous stress tensor. The hyper-diffusive terms contribute to the dissipation Q_{diss} with terms of the type $v_1\, c\, \Delta x_j \; d^{+}_{,j}(u_i)\, \partial^{+}_{,j}(u_i)$. These are positive definite if and only if the hyper-diffusive operators $d^{+}_{,j}(u_i)$ have the same sign as their first order derivative counterparts $\partial^{+}_{,j}(u_i)$. Similarly, the contributions to the Joule dissipation from the hyper-diffusive resistive terms are positive definite if and only if the resistive component of the electric field $E_i^{(\eta)}$ has the same sign as $\mathbf{J}_i$.

The hyper-diffusive operators that we employ fulfill these criteria, for the simple reason that each one is equal to the corresponding first order derivative times a positive factor. The positive definite factor is chosen so as to obtain the desired scaling with wavelength. The basic idea is to have a factor which is proportional to the absolute value of the third derivative, divided by the absolute value of the first derivative. This produces a result which has the sign of the first derivative (and hence guarantees positive definite dissipation), with the magnitude of the third derivative (thus drastically reducing the diffusion of resolved structures).

In practice one has to guard against the situation where the first derivative vanishes, or becomes very small, while the third derivative does not. We do this by taking maxima over the three nearest grid points; if the third difference is non-zero, then at least one of the nearest neighbor first differences must also be non-zero.

We thus choose as hyper-diffusive operators, for example

$$d^{+}_{,x}(f) = \frac{\max_{x\pm 1}|\Delta^3 f|}{\max_{x\pm 1}|\Delta f|}\partial^{+}_{,x}(f) \tag{53}$$

(the max is over three points, in the derivative direction). To make explicit the fact that these are proportional to the ordinary first order derivative operators, we write them as

$$d^{+}_{,x}(f) = q_x(\partial^{+}_{,x}(f))\partial^{+}_{,x}(f) \tag{54}$$

where

$$q_x(f) = \frac{\max_{x\pm 1}|\Delta^2 f|}{\max_{x\pm 1}|f|} \tag{55}$$

is a positive definite "quenching operator". This operator is of the order of unity for high wavenumber perturbations, and diminishes as k^{-2} for smaller wavenumbers.

Using these operators, the symmetrised viscous stress tensor is defined by

$$\tau_{ij} = \tfrac{1}{2}(e_{ij} + e_{ji}), \tag{56}$$

where

$$e_{ij} = \rho(\nu_j^{(1)} q_j(\partial^{+}_{,j} u_i) + \nu_j^{(2)})\partial^{+}_{,j} u_i, \tag{57}$$

and

$$\nu_j^{(1)} = \Delta x_j (v_1 c_f + v_2 |u_j|), \tag{58}$$

$$\nu_j^{(2)} = \Delta x_j^2 v_3 |\nabla \cdot \mathbf{u}|_- \tag{59}$$

(cf. Eq. (33) and Eq. (42)).

Similarly, we define the resistive part of the electric field as

$$
\begin{aligned}
E_x^{(\eta)} &= \left\{\tfrac{1}{2}(\eta_y^{(1)} q_y(J_x) + \eta_z^{(1)} q_z(J_x)) + \tfrac{1}{2}(\eta_y^{(2)} + \eta_z^{(2)})\right\} J_x, \\
E_y^{(\eta)} &= \left\{\tfrac{1}{2}(\eta_z^{(1)} q_z(J_y) + \eta_x^{(1)} q_x(J_y)) + \tfrac{1}{2}(\eta_z^{(2)} + \eta_x^{(2)})\right\} J_y, \\
E_z^{(\eta)} &= \left\{\tfrac{1}{2}(\eta_x^{(1)} q_x(J_z) + \eta_y^{(1)} q_y(J_z)) + \tfrac{1}{2}(\eta_x^{(2)} + \eta_y^{(2)})\right\} J_z,
\end{aligned} \tag{60}
$$

$$
\eta_j^{(1)} = \frac{\Delta x_j}{\mathrm{Pr}_M}(v_1 c_f + v_2 |u_j|), \tag{61}
$$

$$
\eta_j^{(2)} = \frac{\Delta x_j^2}{\mathrm{Pr}_M} v_3 |\nabla_\perp \cdot \mathbf{u}|_- \tag{62}
$$

The coefficients are symmetrised with respect to the directions perpendicular to the particular electric current component, in order for the quenching operators to monitor the behavior in the directions that contribute to the current.

4 Time Stepping

For advancing the solution of the MHD equations in time we use the explicit 3rd order predictor-corrector procedure by Hyman (1979), modified for variable time steps. The predictor is

$$
f_{n+1}^{(*)} = a_1 f_{n-1} + (1 - a_1) f_n + b_1 \dot{f}_n, \tag{63}
$$

and the corrector is

$$
f_{n+1} = a_2 f_{n-1} + (1 - a_2) f_n + b_2 \dot{f}_n + c_2 \dot{f}_{n+1}^{(*)}, \tag{64}
$$

where

$$
\begin{aligned}
a_1 &= r^2, && (65)\\
b_1 &= \Delta t_{n+1/2}(1 + r), && (66)\\
a_2 &= 2(1 + r)/(2 + 3r), && (67)\\
b_2 &= \Delta t_{n+1/2}(1 + r^2)/(2 + 3r), && (68)\\
c_2 &= \Delta t_{n+1/2}(1 + r)/(2 + 3r), && (69)\\
r &= \Delta t_{n+1/2}/\Delta t_{n-1/2}. && (70)
\end{aligned}
$$

The time truncation error is of fourth order in Δt, and is stabilizing.

The time step is limited by the requirement that information is only advanced over a fraction of the mesh size for each time step, and we thus require, for example, that

$$
(U + c_f)\Delta t/\Delta x \le C < 1 \tag{71}
$$

We find that the method is stable in simple test cases for $C \approx 0.6$–0.7. In actual simulations we typically use $C = 0.25$–0.33, to keep the time truncation error small and the time stepping well within the limits of stability.

In addition to the Courant condition (71), we also monitor similar conditions for the viscous and resistive diffusion, by requiring

$$\max(\nu, \eta) f \Delta t / \Delta x^2 \leq C, \tag{72}$$

where f is an factor that depends on the expression used for the diffusion quenching operator. For Eq. (55), $f = 20$.

5 Tests and Applications

The code has been verified by running a number of standard tests (Stein et al. 1994); advection with constant speed, propagation of sound waves and Alfvén waves, hydrodynamical 1D shock tubes and 2D cylindrical shocks, and the MHD shock tube problem of Brio & Wu (1988).

The code has henceforth been used in a number of applications, such as simulations of 3D turbulence, stellar convection and dynamo experiments (Nordlund et al. 1994), investigations of problems related to coronal heating (Galsgaard & Nordlund 1995a; Galsgaard & Nordlund 1995b; Galsgaard 1995), and 2D and 3D simulations of the buoyant rise of magnetic flux tubes (Dorch 1995).

6 Concluding Remarks

The development and application of this 3D MHD code illustrates several points of general interest with respect to the implementation on massively parallel computers:

- Fortran 90 / High Performance Fortran is an excellent language for the implementation of explicit methods of solving partial differential equations. The operator notation made available with the use of array-valued functions makes the implementation of complicated operator expressions straight-forward, and therefore less error prone.

- The floating point performance obtained on a massively parallel computer using this technique is close to optimal, when the operators involved are local or short range, for the simple reason that sufficiently interesting numerical solutions require so many mesh points that most nearest neighbors actually reside in the same local memory.

- The performance on mildly parallel vector-supercomputers such as the Cray C90 series and similar machines is also close to optimal with this coding technique.

- The access to visualizing tools, including visualizing debuggers and performance optimization tools is an essential productivity factor. With access to debuggers only capable of printing numerical values, it would have been much more difficult and time consuming to develop the code.

With regard to the actual numerical method, we have demonstrated:

- That an efficient numerical MHD code must, on the one hand, be able to dissipate efficiently when the physics forces structures (such as current sheets) to develop at the smallest resolved scales.
- On the other hand, the dissipation on well resolved scales should be kept as small as possible, to increase the effective numerical resolution of the code.

These two requirements may be combined by either utilizing a code (such as PPM; Colella & Woodward 1984b) that formally solves the dissipationless equations, and supplies dissipation on the smallest scales through finite numerical resolution effects, or by using a code (such as the present one), that uses explicit and significant dissipation where needed, and suppresses dissipation elsewhere.

Acknowledgments

This work was supported in part by the Danish Research Foundation, through its establishment of the Theoretical Astrophysics Center. We thank the Danish Center for Applied Parallel Processing (CAP) for supporting this work through a grant to KG, and through generous allotment of CM-200 computing time. Additional CM-200 time was provided by the Danish Natural Science Research Council

References

Brio, M., Wu, C. C. 1988, J. Comp. Phys., 75, 400
Colella, P., Woodward, P. R. 1984a, J. Comp. Phys., 54, 174
Colella, P., Woodward, P. R. 1984b, J. Comp. Phys., 54, 174
Dai, W., Woodward, P. R. 1994a, J. Comp. Phys., ??, ??
Dai, W., Woodward, P. R. 1994b, in J. D. B. et al. (ed.), Proc. of the Cornelius Lanczoz International Centenary Conference, Society for Industrial and Applied Mathematics, Philadelphia, p. 446
Dorch, S. 1995, *Master's thesis*, Copenhagen University
Galsgaard, K. 1995, J. Geophys. Res., (in preparation)
Galsgaard, K., Nordlund, Å. 1995a, J. Geophys. Res., (submitted)
Galsgaard, K., Nordlund, Å. 1995b, J. Geophys. Res., (in preparation)
Godunov, S. K. 1959, Math. Sb., 47, 271
Hyman, J. 1979, in R. Vichnevetsky, R. S. Stepleman (eds.), Adv. in Comp. Meth. for PDE's—III, 313
Landau, L., Lifshitz, E. M. 1987, Fluid Mechanics, Pergamon Press
Nordlund, Å., Galsgaard, K., Stein, R. F. 1994, in R. J. Rutten, C. J. Schrijver (eds.), Solar Surface Magnetic Fields, Vol. 433, NATO ASI Series
Stein, R. F., Galsgaard, K., Nordlund, Å. 1994, in J. D. B. et al. (ed.), Proc. of the Cornelius Lanczos International Centenary Conference, Society for Industrial and Applied Mathematics, Philadelphia, p. 440
van Leer, B. 1977, J. Comp. Phys., 23, 276
van Leer, B. 1979, J. Comp. Phys., 32, 101
Woodward, P. R., Colella, P. 1984, J. Comp. Phys., 54, 115