COMPUTATIONAL MAGNETOHYDRODYNAMICS

Notes for an introductory level course

Gábor Tóth

Dept. of Atomic Physics, Eötvös University,
Puskin u. 5-7, 1088 Budapest, Hungary,
`gtoth@hermes.elte.hu`,
`http://hermes.elte.hu/~gtoth/`

Porto, June 15-19, 1998

Contents

Chapter 1

Introduction to Computational Magnetohydrodynamics

1.1 Motivation

There can be several reasons to do hydrodynamic (HD) and magnetohydrodynamic (MHD) computer simulations.

In astronomy, in contrast with physics, the observer usually cannot influence the object of the investigation, in other words, one cannot do experiments, only observations. Analytical calculations and numerical simulations provide the only substitute for experiments. Due to the complexity of the physical phenomena analytical calculations are limited to the simplest cases and many approximations have to be made. Numerical simulations have their own limitations too, but they can be used together with observations and analytical calculations in a complementary way which can lead to a deeper understanding of the examined phenomena.

Even in cases when experiments are possible to do, computer simulations may turn out to be more efficient and less expensive. This is the situation in aerospace engineering, where wind tunnel experiments are replaced by simulations for economical reasons. In fact, most of the modern numerical hydrodynamic schemes were invented and developed by researchers working in the aerospace industry. Magnetohydrodynamics has fewer engineering applications than hydrodynamics, although the thermonuclear fusion research also requires computer simulations of magnetized plasma.

Finally, computer scientists and the computer industry are always searching for applications that can prove the usefulness of their software and hardware. The memory and CPU required by three-dimensional MHD calculations are still a great challenge even for the latest parallel super computers. Massively parallel computers, on the other hand, require new numerical algorithms to be developed.

1.2 Equations

The equations of magnetohydrodynamics can be derived from the hydrodynamic equations, the Lorentz force and Ohm's law, and the Maxwell equations. The most important assumption is that the macroscopic velocity $\mathbf{v}$ of the plasma is much less than the speed of light c. The displacement current $\partial\mathbf{E}/\partial t$ can be ignored and the current can be expressed from the magnetic field $\mathbf{B}$. The electric field $\mathbf{E}$ can be expressed from the Lorentz force and the conductivity σ. This way electro-magnetic waves are eliminated, which greatly eases the numerical solution. Both the total pressure p_{tot} and total energy density e will have contributions from the magnetic field. The following equations hold:

$$\begin{aligned}
\mathbf{J} &= \frac{1}{\mu}\nabla\times\mathbf{B} && (1.1)\\
\mathbf{E} &= \mathbf{J}/\sigma - \mathbf{v}\times\mathbf{B} && (1.2)\\
p_{tot} &= p + \frac{\mathbf{B}^2}{2\mu} && (1.3)\\
e &= \frac{p}{\gamma-1} + \frac{\rho v^2}{2} + \frac{\mathbf{B}^2}{2\mu} && (1.4)
\end{aligned}$$

where ρ, $\mathbf{v}$, and p are the mass density, velocity and thermal pressure, respectively. Furthermore, γ is the adiabatic index for an ideal gas and σ is the electric conductivity. Below, the resistivity $\eta = 1/\sigma$ will be used, and we will choose units of the magnetic field such that $\mu = 1$.

After eliminating the electric field and the current, the magnetized plasma can be fully described by the *primitive variables* ρ, $\mathbf{v}$, p, and $\mathbf{B}$, which are all functions of time t and three (if no simplifying symmetry assumption is made) spatial coordinates $\mathbf{x}$. For ideal MHD $\eta = 0$, while in resistive MHD $\eta > 0$ is not negligible at least in some parts of the flow.

The MHD equations can be expressed in various mathematical forms, however, for numerical models the conservative form is often preferred: the equations (1.5–1.8) explicitly represent the conservation of mass, momentum, total energy, and induction of magnetic field. This is especially important if *weak solutions* containing discontinuities are of interest. The *conservative variables* are ρ, $\rho\mathbf{v}$, e, and $\mathbf{B}$. In terms of these variables the partial differential equations (PDE) of resistive MHD are

$$\begin{aligned}
\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) &= 0 && (1.5)\\
\frac{\partial\rho\mathbf{v}}{\partial t} + \nabla\cdot(\mathbf{v}\rho\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_{tot} &= 0 && (1.6)\\
\frac{\partial e}{\partial t} + \nabla\cdot(\mathbf{v}e + \mathbf{v}p_{tot} - \mathbf{B}\mathbf{B}\cdot\mathbf{v} - \mathbf{B}\times\eta\mathbf{J}) &= 0 && (1.7)\\
\frac{\partial\mathbf{B}}{\partial t} + \nabla\cdot(\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) + \nabla\times(\eta\mathbf{J}) &= 0 && (1.8)
\end{aligned}$$

The *initial condition* should satisfy

$$\nabla\cdot\mathbf{B} = 0 \qquad (1.9)$$

The exact solution of the MHD equations (1.5)-(1.8) keeps $\nabla\cdot\mathbf{B} = 0$ indefinitely. In multidimensional numerical calculations, however, the discretization errors may produce a finite divergence of the magnetic field unless the scheme is specifically designed to keep the discretized form of $\nabla \cdot \mathbf{B}$ zero (see chapter 5).

There are several physical terms that can be added to the right hand side of the momentum and energy equations (1.6, 1.7), such as the effects of gravity, viscosity, thermal conduction, radiative cooling or heating, etc. As long as these are small terms, the numerical methods valid for pure conservation laws can be generalized easily to the full equations. Stiff source terms, however, may require special numerical techniques.

1.3 Basic Properties

The resistive MHD equations are *parabolic* due to terms like $\nabla \times (\eta \mathbf{J}) = \eta \nabla^2 \mathbf{B}$ (for uniform η) in (1.8), which describe *diffusion* of the magnetic field. When no resistivity is present ($\eta = 0$), the ideal MHD equations are *hyperbolic*, which means that the equations have wave-like solutions that propagate without dissipation. As long as the diffusion time scale related to resistivity is not too short, the numerical techniques applicable to the ideal MHD equations work for the resistive equations as well. For this reason we will concentrate on the properties of the ideal MHD equations in this section.

The ideal MHD equations are *not strictly hyperbolic*, since some of the wave speeds can be equal. Due to the *non-linear* terms like $\nabla \cdot \mathbf{v}\rho\mathbf{v}$ in (1.6), discontinuities can also form spontaneously. Another interesting property of the ideal MHD equations is *nonconvexity*, which allows the existence of compound waves consisting of shocks and rarefaction waves. In a real physical system, however, these compound waves split due to deviations of the magnetic field from the rather special necessary conditions.

Three different waves exist in ideal MHD: *slow, Alfvén*, and *fast* waves. The Alfvén waves propagate perturbations in the transverse components of velocity and the magnetic field at a speed

$$c_x^{\text{alfven}} = \frac{|B_x|}{\sqrt{\rho}} \tag{1.10}$$

relative to the fluid velocity $\mathbf{v}$. The Alfvén wave is linearly degenerate like the contact discontinuity, thus non-linear circularly polarized Alfvén waves and rotational discontinuities (of the transverse field) can propagate at this speed without dispersion. The Alfvén wave propagates fastest along the field lines at a speed $c_{\text{alfven}} = |\mathbf{B}|/\sqrt{\rho}$.

In contrast with Alfvén waves, fast and slow waves involve compression of the plasma, therefore they are related to ordinary sound waves, which propagate with the sound speed $c_{\text{sound}} = \sqrt{\gamma p/\rho}$ in the absence of magnetic fields. The fast and slow magnetosonic wave speeds relative to the plasma are

$$c_x^{\text{fast,slow}} = \frac{1}{\sqrt{2}}\left[c_{\text{sound}}^2 + c_{\text{alfven}}^2 \pm \sqrt{\left(c_{\text{sound}}^2 + c_{\text{alfven}}^2\right)^2 - \left(2c_{\text{sound}}c_x^{\text{alfven}}\right)^2}\right]^{1/2} \tag{1.11}$$

where the plus sign corresponds to the fast magnetosonic wave speed. The fast and slow waves can steepen into fast and slow MHD shock waves respectively.

In general the following relationship holds

$$c_x^{\text{fast}} \geq c_x^{\text{alfven}} \geq c_x^{\text{slow}} \tag{1.12}$$

but depending on the orientation and magnitude of the magnetic field, the speeds can be equal. For example, when the direction x is parallel to the magnetic field, all three wave speeds coincide, which is called the *triple umbilic point.*

The largest wave speed by which information can propagate parallel to the x axis is

$$c^{\max} = |v_x| + c_x^{\text{fast}} \tag{1.13}$$

1.4 Generalizations of the MHD Description

The isothermal MHD equations can be regarded as a generalization (or rather simplification) of the full MHD equations. The physical idea is that some fast cooling and/or heating mechanism keeps the temperature T of the gas constant, thus the thermal pressure is simply

$$p = \frac{k_B}{M}\rho T \tag{1.14}$$

where k_B and M are the Boltzmann constant and the average mass of the particles (including free electrons!), respectively. One needs to solve equations (1.5, 1.6, 1.8) only, since the energy equation (1.7) is replaced by the isothermal equation of state (1.14).

In certain astrophysical problems the plasma may contain more than one weakly coupled ion species beside electrons (strongly coupled ions can be approximated as a single fluid with averaged mass, momentum, and energy densities). There are several levels of approximations to describe such a multicomponent plasma. The most general description requires the solution of the full set of hydrodynamic equations including magnetic, electric, and frictional terms for each species together with the full set of Maxwell equations. When one of the species is much lighter than the others, typically the electrons, the electric field can be expressed from the momentum equation of these particles. Quasi-neutrality can also be assumed, which defines the density of the light particles as a simple algebraic equation. The velocity of the light particles can be derived from the current. This procedure leads to the multi-ion MHD equations.

A weakly coupled mixture of neutral and ionized plasma can be described by the two-fluid equations [7, 8], which contain the hydrodynamic equations for neutrals and the MHD equations for the plasma. The two sets of equations are coupled by frictional and heating source terms which may require a semi-implicit treatment [37]. For small fractional ionization further simplifications can be made. The simplest form of partially ionized plasma describe ambipolar diffusion, which assumes a slow drift of neutrals relative to the ions.

Chapter 2

Spatial Discretization

In general a system of conservation laws with some extra source terms can be written as

$$\partial_t \vec{U} + \partial_i \vec{F}_i(\vec{U}) = \vec{S}(\vec{U}) \tag{2.1}$$

where $\vec{U}$, $\vec{F}_i$, and $\vec{S}$ contain the conservative variables, the fluxes, and the source terms respectively, and ∂_i represents the spatial derivative in direction i, and a summation is implied over $i = 1, 2, 3$.

In this section a very concise overview of the different approaches towards spatial discretization is given. The interested reader may look at standard text books (e.g. [11, 14]) describing the standard discretization techniques and read some more recent articles [20, 45, 21] on the relatively new compact schemes, and a very recent paper on the discontinuous Galerkin method [4].

2.1 Finite Differences

The simplest discrete representation of spatial derivatives is by the finite difference method (FDM). The flow variables are given as point-wise values U_j at locations $\mathbf{x}_j$ as

$$U_j(t) = U(\mathbf{x}_j, t) \tag{2.2}$$

Difference formulae of a given order of accuracy can be derived from Taylor expansion around the grid points. On a 1D uniform grid with grid spacing Δx, for example, a first order spatial derivative can be approximated by the centered finite difference formula

$$\begin{aligned} \frac{U_{j+1} - U_{j-1}}{2\Delta x} &= \frac{1}{2\Delta x}\left[U_j + \Delta x \partial_x U + \frac{(\Delta x)^2}{2!}\partial_{xx} U + \ldots\right] \\ &- \frac{1}{2\Delta x}\left[U_j - \Delta x \partial_x U + \frac{(\Delta x)^2}{2!}\partial_{xx} U + \ldots\right] \\ &= \partial_x U + \mathrm{O}\left((\Delta x)^2\right) \end{aligned} \tag{2.3}$$

where $U(x_{j\pm1}) = U(x_j \pm \Delta x)$ was expanded around x_j. Using the centered difference formula above, and a one-sided difference formula for the temporal discretization, the finite difference form of the mass conservation equation (1.5)

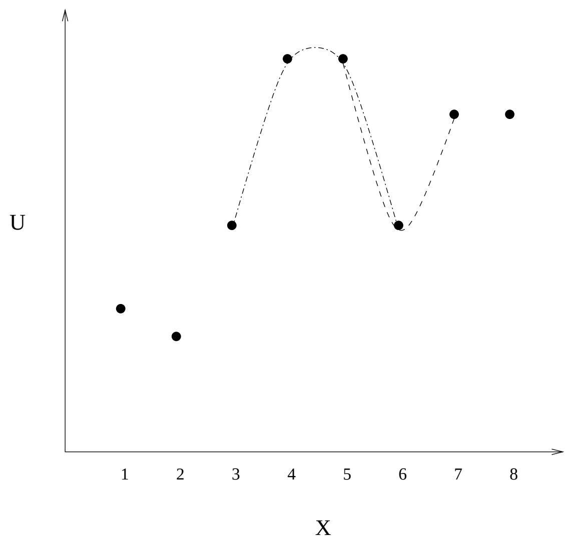

Figure 2.1: Finite difference representation of 1D data. The curves show polynomial fits that can be used for calculating spatial derivatives. There is no unique representation of U between grid points.

in 1D can be written as

$$\rho_j^{n+1} = \rho_j^n - \Delta t \frac{(\rho v)_{j+1}^n - (\rho v)_{j-1}^n}{2\Delta x} \tag{2.4}$$

where the n superscript indicates the discrete time level at $t = n\Delta t$. The main advantage of finite differences lies in its simplicity; it can be implemented easily and efficiently. The main disadvantage is that there is no unique way of defining FDM on unstructured grids, and there is no guarantee for conservation of quantities on non-uniform grids. On structured curvilinear grids one can use *generalized coordinates* to generalize the finite difference formulae valid on Cartesian grids.

2.2 Finite Volume

In the finite volume discretization space is divided into grid cells, and the cell-averages

$$U_j(t) = \frac{1}{V_j} \int_{V_j} U(\mathbf{x}, t) d\mathbf{x} \tag{2.5}$$

are known for each cell. The differential equations are discretized in their integral form, i.e. the fluxes through the cell interfaces are added to and subtracted from the cell-averages (see section 3.3 for details). This method automatically leads to a conservative discretization on arbitrary structured or unstructured grids, and it is simpler than the finite element method or the finite difference method with generalized coordinates. The disadvantage of finite volumes is in the difficulty to go higher than second order spatial accuracy and the relatively complicated representation of second derivatives in space.

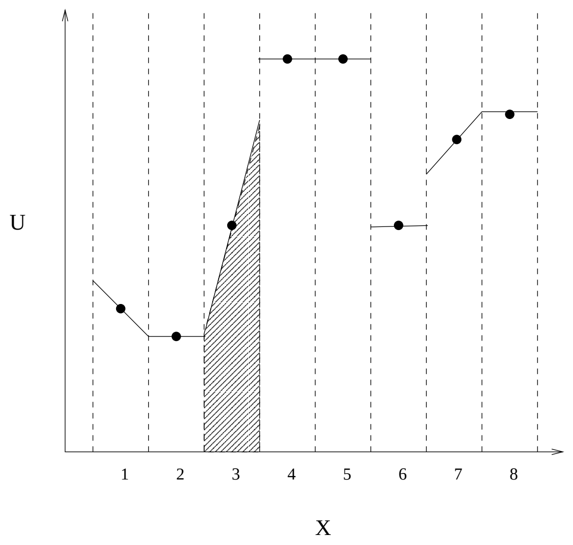

Figure 2.2: Finite volume representation of 1D data. The grid cells are bounded by cell interfaces (dashed lines). The dots represent the cell averages of U. The solid lines show possible extrapolations of U to the cell interfaces. Slope limiters (section 4.5) are necessary to maintain a monotonic profile.

2.3 Finite Elements

In the finite elements method (FEM) the nodal values U_j are given at the x_j points (nodes), while at other locations U is approximated by a linear combination of localized interpolating polynomials f_j as

$$\overline{U}(x,t) = \sum_j U_j(t) f_j(\mathbf{x}) \tag{2.6}$$

i.e. the approximate solution consists of finite elements. For the *Galerkin* FEM, the discretization of the partial differential equations happens by requiring that the integral of the residual of the approximate solution multiplied by the *same* f_j polynomials

$$\int \left[\partial_t \overline{U} + \partial_i \vec{F}_i(\overline{U}) - \vec{S}(\overline{U})\right] f_j d\mathbf{x} = 0 \tag{2.7}$$

must vanish for every j. The advantage of the finite element discretization lies in its systematic approach to high-order accuracy on arbitrary grids by use of high order interpolation polynomials. This explains the popularity of the FEM in engineering. On the other hand, the FEM is rather complicated, and even with explicit time integration schemes it requires the inversion of huge linear systems. Discontinuities are also a problem for standard finite element methods.

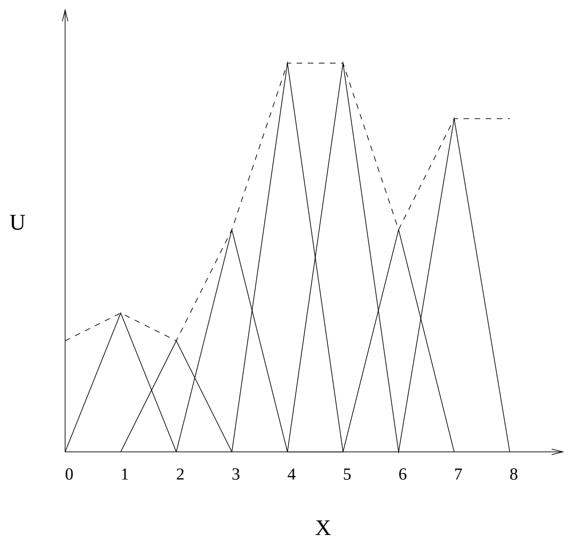

Figure 2.3: Finite element representation of 1D data with linear elements.

2.4 Pseudo-Spectral Methods

A further step from the local description towards a global description is approximating U by a linear combination of some orthogonal functions f_j

$$\overline{U} = \sum_j a_j(t) f_j(\mathbf{x}) \tag{2.8}$$

e.g. by Fourier components, Bessel functions, Chebyshev polynomials, etc. Again the method of weighted residuals (2.7) can be applied to obtain the discretized equations for the $a_j(t)$ amplitudes. In case of the finite elements method, it was the locality of the interpolating functions that simplified the discretization, here the orthogonality of the f_j functions reduces the number of terms.

In practice, however, non-linear terms in the equations have to be evaluated in physical space since the corresponding convolutions in Fourier space would be very expensive. This combination of spectral and physical space is called the pseudo-spectral method. The transformation between these two discrete sets of variables can be efficiently done by fast Fourier transforms (FFT). Pseudo-spectral methods can be extremely accurate with only a few modes representing U, this is referred to as *exponential convergence*, since the discretization error decreases exponentially with the increasing number of modes. This high accuracy makes the pseudo-spectral method ideal for studying turbulence, for example. Disadvantages of this method include the problems with using non-regular grids, satisfying arbitrary boundary conditions, and accurate representation of discontinuous solutions. The pseudo-spectral method may be used in combination with other methods, e.g. in cylindrical coordinates the periodic φ direction may be discretized by Fourier components, while the radial direction can be discretized by finite differences [16].

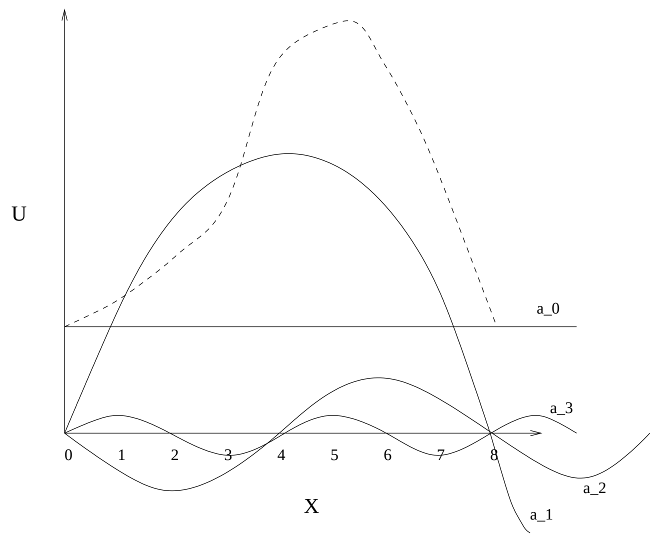

Figure 2.4: Spectral representation of 1D data (dashed line) with four modes (solid lines). The figure is schematic only.

2.5 Compact Schemes

In the computational fluid dynamics research there is a new trend of developing compact schemes with high order accuracy, but localized description of data. There are several approaches to obtain such a scheme: Padé discretization of derivatives (implicit finite differences); using the multi-dimensional information around a grid point rather than extending the stencil in one spatial direction (multi-D finite difference); combining the ideas of finite elements with finite volume method by using high order approximation of U localized for the grid cells (discontinuous Galerkin method). Although these schemes are very promising they are still in an experimental stage at the moment.

Chapter 3

Finding Weak Solutions

This chapter introduces the basic theorems of computational modeling and discusses algorithms suitable for numerically obtaining weak, i.e. discontinuous, solutions. The finite volume discretization uses the same integral form of the equations as the analytical definition of weak solutions, therefore it is especially suited for obtaining such solutions.

3.1 Consistency, Stability, and Convergence

A discretization of the analytical partial differential equations is *consistent* if the local error vanishes as the spatial and temporal resolution goes to infinity, i.e. $\Delta\mathbf{x}, \Delta t \to 0$. The local error can be calculated by expanding the numerical solution around a point in the discrete space-time both in $\mathbf{x}$ and t as

$$\overline{U}(\mathbf{x}, t) = U(\mathbf{x}_j, t_n) + (\mathbf{x} - \mathbf{x}_j)\frac{\partial U}{\partial \mathbf{x}} + (t - t_n)\frac{\partial U}{\partial t} + \dots \tag{3.1}$$

and by substituting $\overline{U}$ into the discretized algebraic equations. The resulting partial differential equation should contain all terms of the original equation (2.1) plus the local error consisting of terms proportional to $\Delta\mathbf{x}^k, \Delta t^k$ $(k \geq 1)$. The order of the scheme is defined as the order k of the lowest order local error term. For example, inserting (3.1) into the finite difference representation (2.4) of the continuity equation leads to

$$0 = \frac{\partial \overline{\rho}}{\partial t} + \partial_x \overline{\rho v} + \frac{\Delta t}{2!}\partial_{tt}\overline{\rho} + \frac{(\Delta x)^2}{3!}\partial_{xxx}\overline{\rho v} + \dots \tag{3.2}$$

Comparing this equation with the analytic equation (1.5), we can conclude that the finite difference equation (2.4) is first order accurate in time and second order accurate in space.

The numerical scheme is *stable* if errors introduced by the discretization do not grow unbounded. There are conditionally stable schemes, where there is some restriction on $\Delta\mathbf{x}$ and Δt, unconditionally stable schemes, and unconditionally unstable schemes. The latter ones are obviously useless for the particular equation. Proving stability of a scheme is a non-trivial task, one can usually prove stability in the linear sense by applying a von Neumann analysis. The idea is that the solution of the linearized equations can be written as a Fourier series. The linearized equations are stable if the amplitude of any Fourier component grows slower than constant times Δt in a single time step. For linear equations one can simply take the ansatz

$$\vec{U}(\mathbf{x}_j, t_n) = G^n \vec{U}_0 \exp(i\mathbf{k}\mathbf{x}_j) \tag{3.3}$$

and substitute it into the discretized equations. The resulting eigenvalue problem can be solved for the eigenvector $\vec{U}_0$ and the eigenvalue G, which gives the amplification factor G^{n+1}/G^n. Numerical stability requires that the magnitude of all eigenvalues $|G| \le 1$ for physically stable problems and $|G| \le 1 + \mathrm{O}(\Delta t)$ for physically unstable equations for any wave number $\mathbf{k}$. For example, the von Neumann analysis of the finite difference formula (2.4) for the continuity equation with a constant velocity v results in

$$|G| = \left|1 - v\frac{\Delta t}{2\Delta x}\left(e^{ik\Delta x} - e^{-ik\Delta x}\right)\right| = \left|1 - v\frac{\Delta t}{\Delta x} i \sin(k\Delta x)\right| > 1 \tag{3.4}$$

which means that this simple centered finite difference representation is unconditionally unstable, and consequently useless. Note that the coefficient $\Delta t/\Delta x$ does not decrease with increasing resolution $\Delta t \propto \Delta x \to 0$, and in any case, the continuity equation is physically stable. In chapters 4 and 6 we shall discuss methods that are consistent and (conditionally) stable for the full set of MHD equations. For hyperbolic equations, the condition is usually in terms of the dimensionless Courant number

$$C = c^{\max}\frac{\Delta t}{\Delta x} \tag{3.5}$$

which gives the distance traveled at the fastest wave speed $c^{\max}$ in time Δt as a fraction of the cell size Δx.

The most important requirement for a scheme is **convergence**, which simply means that the global error of the numerical solution should converge to zero as $\Delta\mathbf{x}, \Delta t \to 0$. It is usually very difficult to prove directly that a scheme is convergent, however the **Lax equivalence theorem states that a consistent and stable finite difference discretization is convergent** for properly posed linear initial value problems. The theorem also holds for finite element and finite volume methods. The MHD equations are not linear and astrophysical problems are usually posed with boundary conditions. Still, consistency and stability are always necessary and often sufficient conditions for convergence.

We shall discuss a practical way of establishing numerical convergence in section 7.6.

3.2 Weak Solutions and Conservative Discretization

A weak solution of hyperbolic partial differential equations contains some discontinuities, such as contact or rotational discontinuity or shock waves. The solution satisfies the PDE at the smooth parts, and it satisfies the jump conditions across discontinuities. These jump conditions can be derived from the integral form of the PDE.

The above discussion of consistency, stability, and convergence was only valid for smooth solutions. Therefore we cannot expect an arbitrary numerical scheme to correctly approximate weak solutions even if it converges to all smooth solutions correctly. It is easy to construct examples when a (non-conservative) numerical scheme converges to incorrect shock speed or jump condition. Lax and Wendroff [19] showed that the limit solution of a finite difference scheme in a conservation form satisfies the jump conditions across a discontinuity. Conservation form means that the discretized quantities are conserved in a numerical sense, for example on a uniform grid

$$\sum_j U_j^{n+1} = \sum_j U_j^n + \frac{\Delta t}{\Delta x} \sum_{\text{bound.}} F \tag{3.6}$$

or on a general grid

$$\sum_j V_j U_j^{n+1} = \sum_j V_j U_j^n + \Delta t \sum_{\text{bound.}} \mathbf{n}\cdot\mathbf{F} \tag{3.7}$$

where U is any of the conservative variables of the conservation law (2.1) with $\vec{S}=0$, and the summations for "bound." represent the contribution of fluxes crossing the boundaries. For the general grid, V_j and $\mathbf{n}$ are the cell volume and the normal vector to the cell interface, respectively. In case of finite volume schemes condition (3.7) is automatically satisfied, since fluxes are added and subtracted at the cell interfaces. For other type of discretizations one must check carefully whether the conservation property is satisfied by adding up the discretized equations.

It is important to note that weak solutions are not uniquely determined by the initial conditions, and in general, an *entropy condition* is needed to pick the right solution. It is not always trivial to show that a numerical scheme is consistent with the physical entropy condition (i.e. entropy cannot decrease with time), and some numerical schemes can converge to non-physical weak solutions (containing "expansion shocks" for example).

3.3 Finite Volume Discretization

Now we revisit the finite volume discretization already introduced in section 2.2 in more detail. If the conservation law (2.1) with $\vec{S}=0$ is volume averaged over the cell j, we get

$$\partial_t \vec{U}_j + \frac{1}{V_j}\int_{V_j} \partial_i \vec{F}_i d\mathbf{x} = 0 \tag{3.8}$$

where $\vec{U}_j$ is defined in the finite volume sense (2.5). There can be physically different terms in $\vec{F}$: gradients, divergences, curls, or Laplace operators. The formulae transforming the volume integral of gradient, divergence, and curl of scalar and vector quantities to surface integrals are

$$\int_{V_j} \nabla p d\mathbf{x} = \sum_l \mathbf{n}^l \bar{p}^{\,l} \tag{3.9}$$

$$\int_{V_j} \nabla\cdot\mathbf{v} d\mathbf{x} = \sum_l \mathbf{n}^l \cdot \bar{\mathbf{v}}^{\,l} \tag{3.10}$$

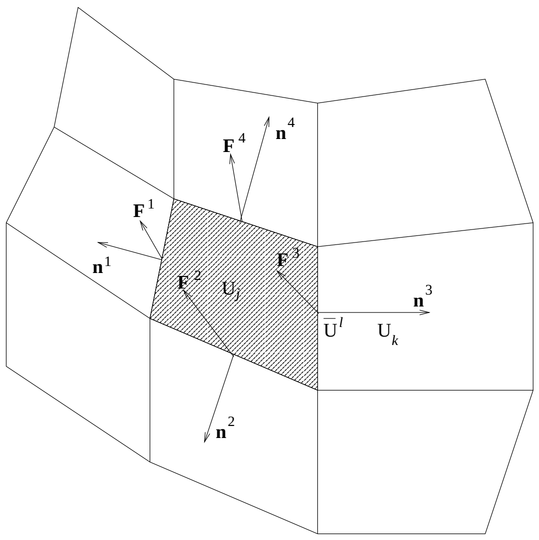

Figure 3.1: Finite volume discretization in 2D. The $\mathbf{n}^l$ normal vectors and the fluxes calculated from the interface averaged $\bar{U}$ are shown.

$$\int_{V_j} \nabla \times \mathbf{B} d\mathbf{x} \quad = \quad \sum_l \mathbf{n}^l \times \bar{\mathbf{B}}^l \tag{3.11}$$

where l runs over all interfaces of cell j and $\mathbf{n}^l$ is the normal vector for the l-th interface. Up to second order spatial accuracy one can approximate the interface averaged $\vec{\bar{U}}^l$ by some interpolation of the cell averaged values $\vec{U}_j$ and $\vec{U}_k$ sharing the interface l. Higher order accuracy would require a Gaussian quadrature along the interface.

The Laplace operator is a bit more difficult to approximate, since it requires the first order derivatives to be calculated for the cell interfaces. This can be done by doing a surface integral on a control volume defined around the midpoint of the cell interface by introducing a *dual grid*. A much simpler approach is, however, to calculate the volume averaged gradient according to the prescription (3.9) first, then to apply an averaging for the cell interface, and to use (3.10) successively for taking the divergence of the gradient. This simple two-step method has a somewhat larger stencil than the more elaborate "dual grid" approach, but that is usually acceptable, since terms with first order derivatives also require comparable stencils due to the need for slope limiters (see section 4.5).

3.4 Axial Symmetry

When the physical system has a cylindrical symmetry, e.g. a jet or an accretion disk, it is quite common to do 2D simulations in the r–z plane assuming axial symmetry in the φ direction. The usual approach is writing the equations into cylindrical coordinates and discretizing them in 2D. An alternative approach is to imagine that the full 3D simulation is done, but neighbouring cells in the φ direction contain the same values as cell j except for the rotation of

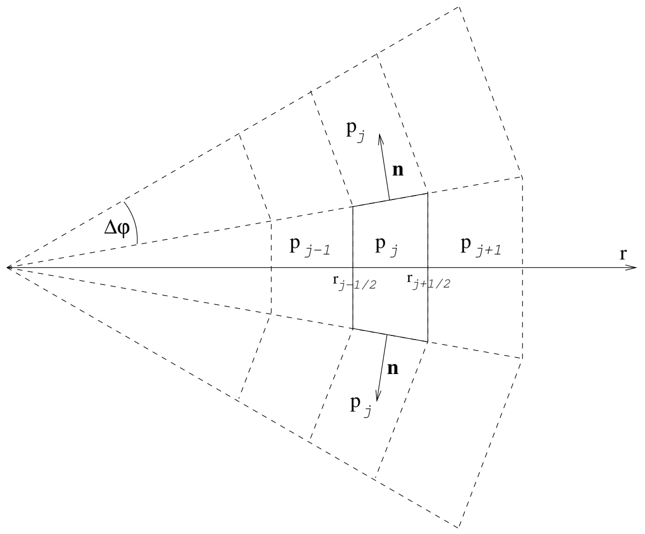

Figure 3.2: Axial symmetry in an imaginary 2D finite volume grid. The cell interface areas in the r direction are proportional to $r_{j\pm1/2}$, while in the φ direction the normal vectors $\mathbf{n}$ are misaligned by $\Delta\varphi$ which results in the term $-V_j p_j/r$ in (3.14).

vector variables by $\pm\Delta\varphi$. The 3 dimensional cell interfaces and the cell volume is $r\Delta\varphi$ times the 2 dimensional edge length and cell area, respectively. The azimuthal resolution $\Delta\varphi$ cancels from all expressions, of course, thus one can simply redefine the normal vectors and the cell volume to

$$\mathbf{n}^l = r^l \mathbf{n}^{2D} \tag{3.12}$$
$$V_j = r_j V_j^{2D} \tag{3.13}$$

and modify the formulae for the r component of the gradient and the φ component of the curl

$$\int_{V_j} \nabla_r p d\mathbf{x} = \sum_l \mathbf{n}_r^l \bar{p}^{\,l} - V_j \frac{p_j}{r_j} \tag{3.14}$$
$$\int_{V_j} (\nabla \times \mathbf{B})_\varphi d\mathbf{x} = \sum_l (\mathbf{n}^l \times \bar{\mathbf{B}}^{\,l})_\varphi - V_j \frac{B_z}{r_j} \tag{3.15}$$

and all other components and the divergence formula remain identical with (3.9-3.11). It is interesting to note that in axial symmetry, even for a uniform Cartesian grid in the r–z plane, the finite volume discretization (3.14) differs from the usual finite difference discretization by a higher than second order term.

Axial symmetry introduces new terms to the equations, e.g. p/r to the radial component of the momentum equation, which is obviously not in a conservation form. The reason is quite trivial: the volume integral of radial momentum is *not conserved* by the MHD equations, only the Cartesian components. Still, the axial symmetric description is identical with the full 3D description, where the source terms are replaced by the contributions from the neighbouring cells in the φ direction, thus we may expect the weak solutions to be correct.

On the other hand, there are *new conserved quantities* in axial symmetry, like the angular momentum $r\rho v_\varphi$. To have an exact conservation of angular momentum one may rewrite the equation into conservation form by using angular momentum $r\rho v_\varphi$ as the conservative variable instead of the momentum ρv_φ which is not conserved.

Chapter 4

Total Variation Diminishing Type Schemes

Some of the most popular methods for solving a hyperbolic system of PDE-s are the total variation diminishing (TVD) type numerical schemes. Although these schemes were developed and used for compressible hydrodynamics starting the 1980-s [13, 25, 42, 43], their application to MHD is relatively recent (e.g. [27, 39]). There are several variations and generalizations of TVD schemes, like total variation bounded (TVB), essentially non-oscillatory (ENO) etc. methods, but these are not discussed here. Even within the TVD family there are dozens of variants, therefore we concentrate on the simplest versions.

4.1 The Concept of Total Variation Diminishing

For a linear system of hyperbolic equations or for a single non-linear hyperbolic PDE (both in one spatial dimension) one can show that the total variation of the analytical solution

$$TV(U) = \sup \sum_{j=1}^{N-1} |U(x_{j+1}) - U(x_j)| \tag{4.1}$$

does not increase in time. The supremum is taken over all possible subdivisions of the x coordinate $x_1 < x_2 < \ldots < x_N$.

The total variation diminishing (TVD) schemes ensure that the discrete equivalent of (4.1) does not increase from one time level to the next one, i.e.

$$\sum_j \left| U_{j+1}^{n+1} - U_j^{n+1} \right| \le \sum_j \left| U_{j+1}^n - U_j^n \right| \tag{4.2}$$

Although the system of MHD equations are non-linear and we are usually interested in multi-dimensional simulations, schemes based on the TVD property are found to behave well near discontinuities; no or little spurious oscillations are produced. The price of the TVD property is the spatially first order representation of smooth local maxima and minima. TVB and ENO schemes can maintain uniformly second order accuracy for smooth solutions but they allow some oscillations near discontinuities.

4.2 TVD Lax-Friedrichs scheme

The simplest TVD type scheme is based on the first order Lax-Friedrichs scheme, which discretizes a conservation law according to

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2} - F_{j-1/2}\right) + \frac{1}{2}\left(\Phi_{j+1/2} - \Phi_{j-1/2}\right) \tag{4.3}$$

where

$$\begin{aligned} F_{j+1/2} &= \frac{F_j + F_{j+1}}{2} && (4.4) \\ \Phi_{j+1/2} &= U_{j+1} - U_j && (4.5) \end{aligned}$$

Clearly, the last two terms in (4.3) add numerical diffusion which corresponds to a term of the form $\nu\nabla^2 U$ with the diffusion coefficient $\nu \propto (\Delta x)^2/\Delta t$, thus the Lax-Friedrichs scheme is only first order accurate. One can also show that the scheme is conditionally stable for Courant number $C < 1$. The numerical diffusion can be reduced by using a diffusive flux

$$\Phi_{j+1/2} = \frac{\Delta t}{\Delta x} c^{\max}_{j+1/2}(U_{j+1} - U_j) \tag{4.6}$$

where the local Courant number (3.5) is used as a coefficient for the artificial diffusion. The scheme described by (4.3, 4.4, 4.6) is the first order TVD Lax-Friedrichs (TVDLF1) scheme. Another way to look at the numerical flux terms Φ is to realize that they modify the centered flux difference formula to a one-sided *upwinded* difference formula, at least for a single linear equation like the continuity equation (1.5). Indeed, for a fixed velocity, the maximum wave speed is $c^{\max} = |v|$, and

$$\begin{aligned} \rho_j^{n+1} &= \rho_j^n - \frac{\Delta t}{2\Delta x}[(\rho v)_{j+1} - (\rho v)_{j-1}] + \frac{\Delta t}{2\Delta x}|v|(\rho_{j+1} - 2\rho_j + \rho_{j-1}) \\ &= \rho_j^n - \frac{\Delta t}{\Delta x}\begin{cases} (\rho v)_j - (\rho v)_{j-1} & \text{for } v \ge 0 \\ (\rho v)_{j+1} - (\rho v)_j & \text{for } v < 0 \end{cases} \end{aligned} \tag{4.7}$$

The upwinded difference formula is very appropriate for the advection equation, and in general for hyperbolic equations, since physically information should propagate from the upstream direction. For a system of equations the upwinding is only approximate in the TVDLF scheme, of course, but in section 4.3 we will see how this can be improved by using characteristic variables.

Second order spatial accuracy can be achieved by a linear approximation of U and the corresponding fluxes at the boundary interfaces. The value of U at the interface at $x_{j+1/2}$ can be linearly extrapolated from the left and right cell center values as

$$\begin{aligned} U^L_{j+1/2} &= U_j^n + \frac{1}{2}\overline{\Delta U}_j^n \\ U^R_{j+1/2} &= U_{j+1}^n - \frac{1}{2}\overline{\Delta U}_{j+1}^n \end{aligned} \tag{4.8}$$

where the limited slopes $\overline{\Delta U}$ will be defined in section 4.5. The fluxes at the cell interface are calculated as

$$\begin{aligned} F_{j+1/2} &= \frac{F(U^L_{j+1/2}) + F(U^R_{j+1/2})}{2} && (4.9) \\ \Phi_{j+1/2} &= \frac{\Delta t}{\Delta x} c^{\max}_{j+1/2}(U^R_{j+1/2} - U^L_{j+1/2}) && (4.10) \end{aligned}$$

The diffusive Φ flux has been greatly reduced, since the difference between the left and right extrapolation is proportional to $(\Delta x)^2$ for a smoothly varying U, still, it provides proper upwinding for the flux difference formula. The maximum propagation speed can be defined as $c^{\max}_{j+1/2} = \max[c^{\max}(U^R), c^{\max}(U^L)]$ or $c^{\max}_{j+1/2} = c^{\max}(U^{LR})$, where U^{LR} is some symmetric average of U^L and U^R. Equations (4.3,4.8–4.10) define a spatially second order TVDLF scheme.

For explicit time stepping, temporally second order accuracy can be achieved by some two step Runge-Kutta discretization, or a predictor-corrector scheme. Hancock's predictor step is probably the best choice. First a time centered

$$U_j^{n+1/2} = U_j^n - \frac{1}{2}\frac{\Delta t}{\Delta x}\left[F(U_j^n + \frac{1}{2}\overline{\Delta U}_j^n) - F(U_j^n - \frac{1}{2}\overline{\Delta U}_j^n)\right] \tag{4.11}$$

is calculated and then it is used for calculating the linear extrapolations

$$\begin{aligned} U^L_{j+1/2} &= U_j^{n+1/2} + \frac{1}{2}\overline{\Delta U}_j^n \\ U^R_{j+1/2} &= U_{j+1}^{n+1/2} - \frac{1}{2}\overline{\Delta U}_{j+1}^n \end{aligned} \tag{4.12}$$

Definitions (4.3,4.9–4.12) define a spatially and temporally second order TVDLF scheme which is stable for Courant number $C < 1$. Contributions of source terms can be added easily to the right hand sides of (4.11) and (4.3), namely $(\Delta t/2)S(U_j^n)$ in the predictor step and $\Delta t S(U_j^{n+1/2})$ in the full step. In multidimensional simulations the U^L and U^R extrapolations should be determined for each interface, and the flux contributions can be added at the same time, however this usually requires the Courant number to be reduced to about $C < 0.5$. An alternative to this fully multidimensional unsplit approach is the use of operator splitting (see section 6.2).

4.3 TVD-MUSCL scheme

The TVD-MUSCL scheme (MUSCL stands for Monotonic Upstream Scheme for Conservation Laws) differs from the TVDLF scheme in that the upwinding is applied for *characteristic variables* rather than the conservative variables. The characteristic variables $\vec{r}^k$ are certain linear combinations of the conservative variables that form the right eigenvectors of the matrix $\partial\vec{F}/\partial\vec{U}$, i.e.

$$\frac{\partial\vec{F}}{\partial\vec{U}}\vec{r}^k = c^k\vec{r}^k \tag{4.13}$$

where c^k is the eigenvalue corresponding to the k-th eigenvector. For a linear system of hyperbolic PDE-s, the characteristic waves consist of components $\vec{r}^k$ that travel at a speed c^k. Hyperbolicity ensures that the eigenvectors and eigenvalues are real and a complete orthogonal basis can be formed from them. The normalized left eigenvectors $\vec{l}^k$ are related to the right eigenvectors by the orthogonality relation $\vec{l}^k\cdot\vec{r}^m = \delta_{k,m}$.

Now we may modify the numerical diffusive flux vector $\vec{\Phi}$ to be

$$\vec{\Phi} = \frac{\Delta t}{\Delta x}\sum_k \vec{r}^k\,|c^k|\,\vec{l}^k\cdot(\vec{U}^R - \vec{U}^L) \tag{4.14}$$

where $\vec{r}^k, c^k, \vec{l}^k$ are calculated for the averaged $U^{LR}_{j+1/2}$ state. The scalar product of the k-th left eigenvector with $\vec{U}^R - \vec{U}^L$ determines the *jump* in the k-th characteristic variable, while the multiplication by $\vec{r}^k$ transforms the result back to the conservative variables. The gain relative to the much simpler TVDLF flux (4.10) is the use of the eigenvalue c^k instead of the largest eigenvalue $c^{\max}$. Therefore the upwinding is accurate for each characteristic variable, which means less numerical diffusion. On the other hand, the left and right eigenvectors need to be calculated for each cell interface, which is rather expensive for the MHD equations.

The temporally second order explicit TVD MUSCL scheme can use the same Hancock predictor step (4.11) as the TVDLF scheme. Contributions of source terms and/or fluxes in multidimensional problems should be added in the predictor and the full step (4.3) similarly to the TVDLF method, and the same conditions apply for stability.

4.4 One Step TVD scheme

The one step TVD method [13] is based on a properly limited Lax-Wendroff scheme applied to each characteristic variable separately. Second order temporal accuracy is achieved by adding terms proportional to $(\Delta t c^k)^2$ which represent the second order term in the Taylor expansion $U^{n+1} = U^n + \Delta t \partial_t U + (\Delta t^2/2)\partial_{tt} U + \ldots$. Several versions of the one step TVD schemes are described in detail in [39].

This scheme is usually even less diffusive than the TVD-MUSCL discretization and it requires slightly less CPU time per time step. On the other hand, temporally second order accurate source terms cannot be directly included, operator splitting has to be used. In case of source terms related to axial symmetry the situation becomes even more complicated [28]. In multi-dimensional calculations the flux components cannot be added at the same time for stability reasons, and dimensional splitting has to be applied (see section 6.2).

4.5 Slope Limiters

In the second order TVDLF and TVD-MUSCL schemes the ΔU slopes of the conservative variables are limited by slope limiters denoted by $\overline{\Delta U}$. The slope limiter is required to ensure the TVD property for the schemes. There are many versions of slope limiters that satisfy both the conditions for TVD property and second order accuracy for smooth solutions. Here we define only three: the *minmod* limiter

$$\overline{\Delta U}_j = \mathrm{minmod}\left(\Delta U_{j-1/2}, \Delta U_{j+1/2}\right) \tag{4.15}$$

the *Woodward* or *monotonized central difference* limiter (MC-limiter) originally by van Leer

$$\overline{\Delta U}_j = \mathrm{minmod}\left(2\Delta U_{j-1/2}, 2\Delta U_{j+1/2}, \frac{1}{2}\Delta U_{j-1/2} + \frac{1}{2}\Delta U_{j+1/2}\right) \tag{4.16}$$

and the *superbee* limiter by Roe

$$\overline{\Delta U}_j = s \max\left[0, \min\left(2|\Delta U_{j+1/2}|, s\Delta U_{j-1/2}\right), \min\left(|\Delta U_{j+1/2}|, 2s\Delta U_{j-1/2}\right)\right] \tag{4.17}$$

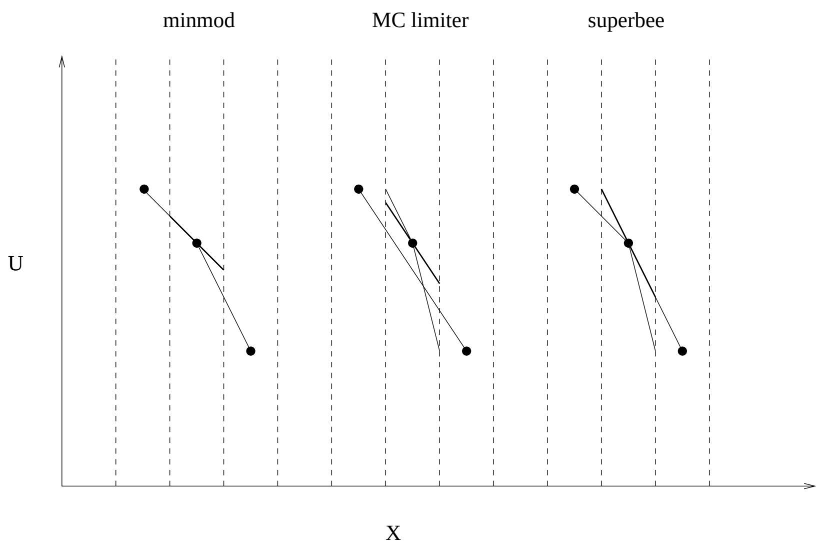

Figure 4.1: Three different slope limiters applied to the same configuration. Each limiter considers a few different approximate slopes (thin lines) and selects one of them (thick line).

where $\Delta U_{j+1/2} = U_{j+1} - U_j$ and $s = \mathrm{sgn}(\Delta U_{j+1/2})$. The generalized minmod function for $n > 1$ arguments is defined as

$$\mathrm{minmod}(w_1, w_2, \ldots, w_n) = \mathrm{sgn}(w_1)\max[0, \min(|w_1|, \mathrm{sgn}(w_1)w_2, \ldots \mathrm{sgn}(w_1)w_n)] \tag{4.18}$$

In words, the minmod function takes the argument with the smallest modulus when all argumnets have the same signs and otherwise it is zero. The minmod limiter is rather diffusive, while the superbee limiter can sharpen smooth waves into discontinuities. In fact, the minmod and the superbee limiters are the most and least diffusive of all acceptable symmetric two-variable slope limiters, respectively. The Woodward/MC limiter lies in between, and it is often found to be a good compromise.

Using slope limiters requires information from the two neighbouring cells in all directions (a 5 point stencil in 1D). This may lead to complications on unstructured or strongly deformed structured grids.

4.6 Riemann Solvers

The Riemann problem is the following: given the spatially constant left and right states U^L and U^R separated by a discontinuity, determine the solution after some time Δt. For hydrodynamics this problem can be solved analytically. The general solution consists of two shock and/or rarefaction waves, a contact discontinuity, and shear waves for the transverse velocity components. In case of a finite volume discretization we are interested in the total flux $\Delta t F_{j+1/2}(U^L, U^R)$ of the conserved quantities that enter and exit the cells j and $j+1$ through the cell interface at $x_{j+1/2}$ from time t to $t + \Delta t$.

Solving the full Riemann problem is fairly difficult and expensive even for the equations of hydrodynamics, and it is even more so for the more complicated MHD equations. It is quite questionable whether it is worthwhile to make such an effort, since the accuracy of the overall scheme is second order at best. There are several approximations to the Riemann problem, but here we will only discuss the simplest and most popular one, which is Roe's approximate Riemann solver [25]. The idea is to linearize the the governing equations and to find a proper average state U^{LR} for which the linearized equation

$$\partial_t \vec{U} + \frac{\partial \vec{F}}{\partial \vec{U}} \vec{U} = 0 \tag{4.19}$$

can be solved. As we have indicated above, in terms of the characteristic variables the linearized Riemann problem becomes simply the advection equation, which can be solved by upwinded derivatives.

The only question that remains is the proper averaging procedure between the left and right states. Roe's original prescription for the *Roe average* requires that the linearized problem is exact for a single shock wave. It is quite difficult to fulfill this requirement for the MHD Riemann problem, and in practice one can use the simple arithmetic average for the primitive variables $\vec{V} = (\rho, \mathbf{v}, p, \mathbf{B})$ to get the averaged state

$$\vec{V}^{LR} = \frac{\vec{V}^R + \vec{V}^L}{2} \tag{4.20}$$

The advantage of averaging primitive variables instead of conservative variables is twofold: negative pressure cannot occur due to the averaging and the eigenvalues and eigenvectors are simpler expressions in terms of $\vec{V}$ than in $\vec{U}$.

The eigenvalues are rather simple: $c^1, c^2 \ldots c^7$ are

$$v_x - c_x^{\text{fast}} < v_x - c_x^{\text{alfven}} < v_x - c_x^{\text{slow}} < v_x < v_x + c_x^{\text{slow}} < v_x + c_x^{\text{alfven}} < v_x + c_x^{\text{fast}} \tag{4.21}$$

where the wave speeds relative to the fluid have been defined by (1.10–1.11). The 4th wave speed $c^4 = v_x$ corresponds to the entropy wave or a contact discontinuity.

The properly normalized (non-singular) left and right eigenvectors $\vec{l}^k$ and $\vec{r}^k$ has been first given for the ideal MHD equations by Brio and Wu [3]. The expressions for the eigenvectors have been greatly simplified by Roe and Balsara [26]. Still, they are too complicated to be repeated here, the interested reader is referred to their article. For multidimensional MHD, Powell [23, 12] has introduced the *8-wave* Riemann solver, which adds a *divergence wave* that propagates with the fluid velocity v. The addition of this wave removes a singularity of the approximate Roe solver that occurs if no care is taken for the numerically generated divergence of the magnetic field (see chapter 5).

Riemann solver formulations always assume that the discontinuity is in the x direction and the vector variables are given with their x, y, z components. On a finite volume grid the cell interface is not aligned with any of the axes in general. The solution is to *rotate* the vector variables into a frame aligned with the interface. In 2D the rotation matrix is unique, in 3D it is not. Once the Riemann problem is solved, the resulting vector components should be rotated back to the x, y, z coordinate system with the inverse matrix.

4.7 Entropy Fixes

Roe's approximate Riemann solver (used by the TVD-MUSCL scheme and the one step TVD scheme) can produce non-physical expansion shocks that violate the entropy condition. The physically correct solution is a rarefaction wave. The TVDLF scheme, on the other hand, is not prone to such errors.

For the TVD-MUSCL scheme the magnitude of the eigenvalue $|c^k(U^{LR})|$ in the diffusive flux definition (4.14) is replaced by

$$\psi(c^k) = \begin{cases} |c^k| & \text{if } |c^k| \geq c^k_{\text{min}} \\ \frac{(c^k)^2+(c^k_{\text{min}})^2}{2c^k_{\text{min}}} & \text{otherwise} \end{cases} \tag{4.22}$$

Essentially, the sharp $|.|$ function is smoothed by a parabola near 0. The parameter c^k_{min} can be determined in several ways. The original entropy fix by Harten and Hyman uses the wave speeds $c^{k,L}$ and $c^{k,R}$ corresponding to the left and right states U^L and U^R, respectively:

$$c^k_{\text{min}} = \max(0, c^k - c^{k,L}, c^{k,R} - c^k) \tag{4.23}$$

They prove that this fix avoids the violation of the entropy condition near transonic points. Powell suggests a somewhat more diffusive (larger c^k_{min}) entropy fix

$$c^k_{\text{min}} = \max[0, 2(c^{k,R} - c^{k,L})] \tag{4.24}$$

while Yee simply takes a constant value for c^k_{min} for each k. See [17] for a detailed discussion of the entropy condition and fixes.

In principle an entropy fix is only required for the non-linear waves like the left and right moving slow and fast magnetosonic waves. In practice, however, numerical difficulties can often be overcome by adding a small amount of diffusion. Therefore entropy fixes are sometimes applied to all characteristic waves with good results [24].

4.8 Positivity Fixes

Beside the conservation of the conservative variables $\vec{U}$, there are other physical restrictions on the numerical solution. In hydrodynamics already, it is difficult to maintain the positivity of thermal pressure in supersonic flows. The reason is that the total energy e is dominated by the kinetic energy, and a small error in either may lead to a numerically negative thermal energy. The situation is even worse for MHD, where the magnetic energy can also dominate over thermal energy. The simplest fix is that negative pressure values calculated from equation (1.4) are replaced by a small positive value. This may work in some cases, but not always. The next step can be adding some diffusion in form of an entropy fix, or by applying more diffusive slope limiters (like minmod), or sometimes reducing the time step may help. A more drastic solution is to increase the total energy where the pressure is negative to a value which makes it positive again. Such a change violates the conservation of energy, but in some cases, e.g. when a stationary solution is sought for, this can be acceptable.

The best solution is to locally recalculate $\vec{U}$ by a *positive scheme* like the HLLE scheme [9], which guarantees positivity. The reason of not using a positive scheme everywhere is that such schemes are only first order accurate. Tests by Quirk [24] show that such a composite scheme can be quite robust against many failings of the approximate Riemann solvers in computational hydrodynamics. It is not clear how the HLLE scheme works for MHD.

Chapter 5

Keeping Divergence of the Magnetic Field Zero

The MHD equations have a special property which does not occur in the compressible hydrodynamic equations, namely the requirement that the magnetic field should remain divergence free (1.9). This is automatically satisfied in 1D simulations where B_x =const, and $\partial_y = \partial_z = 0$, but the standard discretization methods do not guarantee $\nabla \cdot \mathbf{B} = 0$ in multi-dimensional simulations. There are several ways to deal with this problem. Many of these approaches have close analogues applied to the equations of *incompressible hydrodynamics*, where the velocity field $\mathbf{v}$ should remain divergence free [18]. This analogy, however, is not realized by many practitioners of computational MHD.

The simplest approach is to increase the spatial resolution until the divergence of $\mathbf{B}$ become small and the solution starts to converge. This may not work if the scheme is unstable due to the errors in $\nabla\cdot\mathbf{B}$, which is the case for some Riemann type solvers.

Another way to keep $\nabla\cdot\mathbf{B}$ exactly zero is to rewrite the MHD equations by using the *vector potential* $\mathbf{A}$ instead of $\mathbf{B} = \nabla \times \mathbf{A}$. The disadvantage of this approach is that the order of spatial derivatives increases by one, which reduces the order of accuracy by one; the equations are not in a conservation form any longer; the equations become complicated in 3D; and the boundary conditions on the vector potential may not be physically intuitive.

The methods presented in the following sections are used in many modern MHD codes.

5.1 Non-conservative Formulation

One can modify the equations (1.6-1.8) to

$$\frac{\partial \rho\mathbf{v}}{\partial t} + \nabla\cdot(\mathbf{v}\rho\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_{tot} = -(\nabla\cdot\mathbf{B})\mathbf{B} \qquad (5.1)$$

$$\frac{\partial e}{\partial t} + \nabla\cdot(\mathbf{v}e + \mathbf{v}p_{tot} - \mathbf{B}\mathbf{B}\cdot\mathbf{v} - \mathbf{B}\times\eta\mathbf{J}) = -(\nabla\cdot\mathbf{B})\mathbf{B}\cdot\mathbf{v} \qquad (5.2)$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla\cdot(\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) + \nabla\times(\eta\mathbf{J}) = -(\nabla\cdot\mathbf{B})\mathbf{v} \qquad (5.3)$$

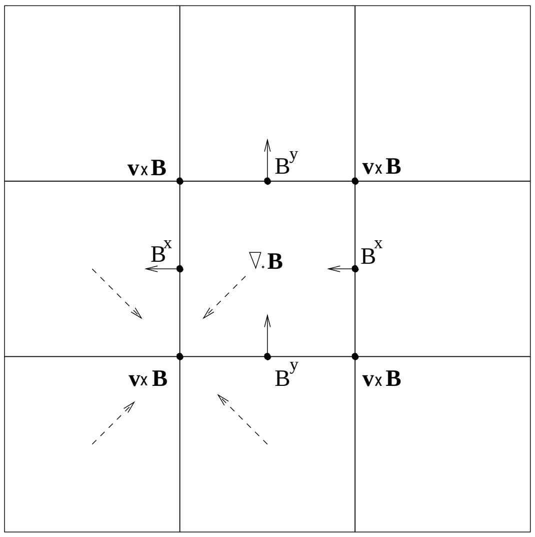

Figure 5.1: The staggered grid version of Constrained Transport. Combined with a finite volume approach, some interpolation is required (dashed arrows).

by adding source terms to the right hand sides. All of these terms are proportional to $\nabla\cdot\mathbf{B}$ thus they should always remain zero analytically.

It was found by Powell [23, 12] that writing the MHD equations in the above form together with the 8-wave Riemann solver is stable unlike the usual form in which these "corrective source terms" are omitted. Tóth and Odstrčil [39] found that Powell's source terms are beneficial for the flux-corrected transport (FCT) and TVDLF schemes as well. Note, that these terms are not in a conservation form, but they are usually small.

The non-conservative source terms eliminate the instability, but not the numerical error in divergence $\mathbf{B}$. For some problems the use of Powell's source terms seems to be sufficient, the errors in $\nabla\cdot\mathbf{B}$ remain small, and the conservation of quantities is satisfactory. In other problems, especially for long time integrations, the error may grow to unacceptable values.

5.2 Constrained Transport

For finite difference schemes the *constrained transport* method by Evans and Hawley [10] offers a simple and efficient solution by using staggered grids. The magnetic field components are represented on the cell interfaces, while density, momentum and energy in the cell centers. In 2D the B^x component is located at $x_{j+1/2}, y_k$ while the B^y component is at $x_j, y_{k+1/2}$. The $\mathbf{v}\times\mathbf{B}$ term is calculated at the cell corners $x_{j+1/2}, y_{k+1/2}$ and the induction equation (1.8) is applied using simple finite differences along the cell edges, e.g.

$$B^{x,n+1}_{j+1/2,k} = B^{x,n}_{j+1/2,k} + \Delta t \frac{(\mathbf{v}\times\mathbf{B})_{j+1/2,k+1/2} - (\mathbf{v}\times\mathbf{B})_{j+1/2,k-1/2}}{\Delta y} \qquad (5.4)$$

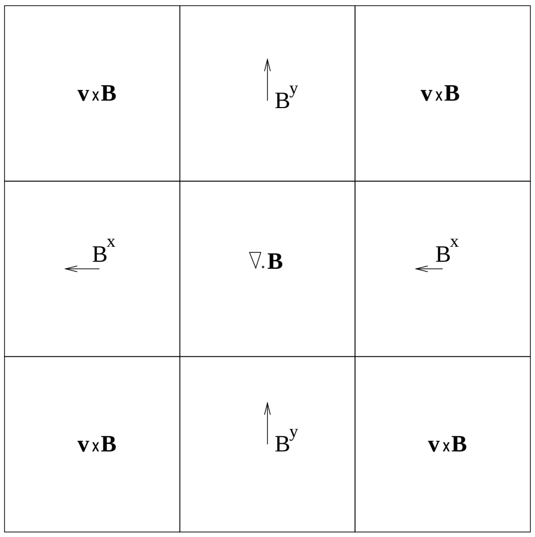

Figure 5.2: The centered difference version of Constrained Transport. Combined with a finite volume approach, no spatial interpolation is needed.

It is easy to show that the numerical divergence of B

$$(\nabla\cdot\mathbf{B})_{j,k} = \frac{B^x_{j+1/2,k} - B^x_{j-1/2,k}}{\Delta x} + \frac{B^y_{j,k+1/2} - B^y_{j,k-1/2}}{\Delta y} \tag{5.5}$$

does not change due to perfect cancellation of terms, i.e. if $\nabla\cdot\mathbf{B}^n = 0$ then $\nabla\cdot\mathbf{B}^{n+1} = 0$ to round off errors. The constrained transport method works in 3D in a similar fashion.

Recently this idea was applied to Godunov type schemes by Dai and Woodward [5]. After a full step by a scheme like TVDLF or TVD-MUSCL, interpolation in space and time is used to obtain $\mathbf{v}\times\mathbf{B}$ at the cell corners at time level $n+1/2$. Afterwards the $\mathbf{B}^*$ field centered at the cell interfaces is updated according to (5.4). The components of the cell centered $\mathbf{B}$ are obtained by averaging $\mathbf{B}^*$, and they are used in the next step. Although Dai and Woodward regard $\mathbf{B}^*$ as the primary variable, in fact it is only used to obtain $\mathbf{B}$, thus one can equally regard $\mathbf{B}$ as the primary variable which eliminates the need for a staggered grid and corresponding boundary conditions. Of course, the numerical divergence of the cell centered $\mathbf{B}$ is not zero, but it should remain close to zero to second order accuracy.

A further step in the direction of simplifying the idea of constrained transport is to use simple central differencing (like eq.2.3) for the induction equation on the original grid using cell centered and time centered $(\mathbf{v}\times\mathbf{B})^{n+1/2}_{j,k}$. Preliminary experiments by the author show that the results are just as good as with the staggered approach, and the formulation is simpler. Furthermore, the cell centered magnetic field is divergence free in the

$$(\nabla\cdot\mathbf{B})_{j,k} = \frac{B^x_{j+1,k} - B^x_{j-1,k}}{2\Delta x} + \frac{B^y_{j,k+1} - B^y_{j,k-1}}{2\Delta y} \tag{5.6}$$

numerical sense, which is more natural than having a staggered $\mathbf{B}^*$ with zero divergence.

The main advantage of the constrained transport method is its simplicity and efficiency. There are a few draw backs as well: the method only works for orthogonal finite volume grids; the stability of the scheme requires the Courant number to be $C \leq 0.6$; the interpolations introduce diffusion and increase the stencil.

5.3 Projection Scheme

The most general fix to the problem is the *projection scheme* originally proposed by Brackbill and Barnes [2], which is a correction to the magnetic field after the time step is completed by some arbitrary numerical scheme. The name comes from the idea that the $\mathbf{B}$ field is projected to a divergence free $\mathbf{B}'$ field which is as close as possible to $\mathbf{B}$. To achieve this, one needs to solve a Poisson equation for the potential ϕ

$$\nabla^2\phi = \nabla\cdot\mathbf{B} \tag{5.7}$$

and then to correct the magnetic field to

$$\mathbf{B}' = \mathbf{B} - \nabla\phi \tag{5.8}$$

It is easy to show that the correction does not affect the vector potential, the current density or the integrated magnetic flux. In fact one can prove that

$$\int dx(\mathbf{B}' - \mathbf{B})^2 \tag{5.9}$$

is minimal with the constraint $\nabla\cdot\mathbf{B}' = 0$ and ϕ is the Lagrange multiplier for this constrained minimalization problem. In other words, the projection scheme makes the smallest possible correction that removes the unphysical part of the numerical solution without affecting the physically meaningful quantities. The price to pay for these nice properties is the Poisson problem (5.7), but that can be solved efficiently with either direct or iterative solvers. The numerical divergence (like eq.5.6) will be exactly zero only if the Laplace operator is evaluated in two steps as a divergence of the gradient (see section 3.3) with the same difference operators as used for calculating $\nabla\cdot\mathbf{B}$ in (5.7). One should realize, however, that the numerical scheme does not actually use this numerical representation of the divergence B directly, so its exact cancellation may not be crucial.

Direct Poisson solvers work for relatively special cases, e.g. uniform Cartesian grid with periodic boundary conditions, only, and they require about 30% of the total CPU time [27]. For this particular Poisson problem (5.7), iterative solvers are not just flexible, but also surprisingly efficient. The numerical errors in $\nabla\cdot\mathbf{B}$ usually arise as local errors of opposite signs (short wavelength) which are removed by the Conjugate gradient type solvers (section 6.6 rather efficiently. The very small long wavelength errors do not have to be removed at all, since an approximate solution of the Poisson problem is quite acceptable. By applying a few iterations of the linear solver, we can reduce the numerically generated divergence of the magnetic field sufficiently. In my experience, the projection scheme with an iterative solver requires about 15% of the total execution time on Cartesian grids (more efficient than direct solvers), and about

30% on general structured grids. One draw back of solving the Poisson equation approximately is the resulting numerical noise that can spoil the convergence to a steady state.

It is also important to choose good boundary conditions for ϕ in the Poisson equation 5.7 so that the corrected magnetic field $\mathbf{B}'$ satisfies the boundary conditions for the physical problem.

Chapter 6

Temporal Discretization

6.1 Explicit Time Stepping

In explicit schemes the fluxes and the sources are calculated at the n-th time level and their contribution is added to the current value of $\vec{U}^n$. The general conservation law (2.1) can be discretized in time as

$$\vec{U}^{n+1} = \vec{U}^n + \Delta t[-\partial_i \vec{F}_i(\vec{U}^n) + \vec{S}(\vec{U}^n)] \tag{6.1}$$

This *forward Euler* scheme is temporally first order accurate only, but it is easy to form higher order Runge-Kutta or predictor-corrector type schemes, e.g. a two step Runge-Kutta scheme is

$$\begin{aligned} \vec{U}^{n+1/2} &= \vec{U}^n + \frac{\Delta t}{2}[-\partial_i \vec{F}_i(\vec{U}^n) + \vec{S}(\vec{U}^n)] & (6.2) \\ \vec{U}^{n+1/2} &= \vec{U}^n + \Delta t[-\partial_i \vec{F}_i(\vec{U}^{n+1/2}) + \vec{S}(\vec{U}^{n+1/2})] & (6.3) \end{aligned}$$

Explicit time integration is simple and fast, and it is easily parallelized for parallel computers. The main disadvantage is that the time step Δt is limited by numerical stability requirements. For the ideal MHD equations, the stability condition requires that the time step is shorter than the crossing time of the grid cells by the fastest wave

$$\Delta t < \frac{\Delta x_i}{c_i^{\max}} \tag{6.4}$$

for all grid cells and all directions $i = 1, 2, 3$. This is the famous **Courant-Friedrich-Levy (CFL) condition** valid for explicit time integration of arbitrary hyperbolic PDE-s. In terms of the Courant number (3.5), the CFL condition simply states that $C < 1$ is required everywhere for stability. In a numerical code one needs to calculate the smallest value of $\Delta x_i / c_i^{\max}$ in every time step and modify Δt such that the CFL condition (6.4) is fulfilled.

In cases when fast waves are really present and the time evolution should be accurately simulated, the stability limitation coincides with the accuracy limitation, since one cannot calculate the interaction of fast waves without representing them at the discrete time levels in each cell they are passing through. In some time accurate calculations, however, the fast waves that set the limit on the time step are not actually present at all, thus the accuracy would not require short time steps, and the explicit scheme may be very inefficient. This is also the case in steady state calculations, where the transients leading to the final state are of no interest.

For the parabolic resistive MHD equations an additional constraint comes from the diffusion time scale, which implies that

$$\Delta t < \frac{(\Delta x_i)^2}{\eta} \tag{6.5}$$

should hold for all grid cells and all directions $i = 1, 2, 3$. It depends on the resistivity η and the grid resolution $\Delta\mathbf{x}$ whether the CFL condition or the diffusion time limit is more restrictive.

6.2 Operator Splitting

The easiest way to generalize a 1D scheme to multidimensional problems is via an operator splitting, originally by Strang [32]. Below a simpler but equally accurate version is described, which is usually used in practice.

Suppose that L_x, L_y and L_z are some temporally and spatially second order operators that evolve the one dimensional equations

$$\begin{aligned} \partial_t\vec{U} + \partial_x\vec{F}_x &= \vec{S}/3 \\ \partial_t\vec{U} + \partial_y\vec{F}_y &= \vec{S}/3 \\ \partial_t\vec{U} + \partial_z\vec{F}_z &= \vec{S}/3 \end{aligned} \tag{6.6}$$

from time t to $t + \Delta t$, respectively. Then applying these 1D operators one by one in an alternating order gives

$$\vec{U}^{n+2} = L_x L_y \; L_y L_x \vec{U}^n \tag{6.7}$$

in 2D and

$$\vec{U}^{n+2} = L_x L_y L_z \; L_z L_y L_x \vec{U}^n \tag{6.8}$$

in 3 spatial dimensions. Using Taylor expansions in time, one can show that $\vec{U}^{n+2}$ is a spatially and temporally second order accurate solution of the sum of the split equations (6.6) which is the same as the full PDE (2.1). Without the alternations the scheme would only be first order accurate in time. The splitting of the source term into equal parts in 6.6 minimizes the splitting error. When calculation of the source term $\vec{S}$ is expensive, one can put all of it into one of the 1D equations. Source terms arising due to axial symmetry (3.14–3.15) should be added at the same time as the radial fluxes $\vec{F}_r$.

In a similar fashion the PDE can be split into a homogeneous multi dimensional part and the equation for the source terms

$$\begin{aligned} \partial_t\vec{U} + \partial_i\vec{F}_i &= 0 \\ \partial_t\vec{U} &= \vec{S} \end{aligned} \tag{6.9}$$

Although the formal accuracy of the split formulation is the same as the accuracy of the unsplit discretization, the two approaches are not always interchangeable. For steady state calculations, for example, the alternating order of the operators will introduce some oscillations between even and odd time steps. This can be fixed by keeping the order of the operators fixed. A more serious problem is that the splitting error introduces a non-linear term in Δt even if the individual operators are linear in Δt. Linearity in the time step is recommended for steady state problems, since it ensures that the solution satisfies the discretized form of $\partial_i \vec{F}_i + \vec{S} = 0$ exactly independent of Δt. Nonlinearity in Δt can lead to spurious steady states, oscillations, or even chaotic behavior [44].

6.3 Fully Implicit and Semi-Implicit Time Stepping

Stability of the time integration can be improved by *fully implicit* time integration. For example the *backwards Euler* scheme

$$\vec{U}^{n+1} = \vec{U}^n + \Delta t[-\partial_i \vec{F}_i(\vec{U}^{n+1}) + \vec{S}(\vec{U}^{n+1})] \tag{6.10}$$

is unconditionally stable (at least according to the linear stability analysis). Note that the fluxes and the sources are evaluated at time level $n+1$ thus one needs to solve an *implicit* equation for U^{n+1}. This simple scheme is only first order accurate in time, but that can be improved easily for time accurate calculations (6.12).

While the stability of the implicit time discretization is a great improvement over explicit schemes, one has to solve the implicit equations for the unknown $\vec{U}^{n+1}$ which requires linearization and the inversion of the $[I - \Delta t \partial(\partial_i \vec{F}_i)/\partial \vec{U}]$ matrix. In 1D there are efficient direct solvers, while in multidimensional cases iterative and multigrid methods need to be used. This can be costly both in terms of CPU and storage requirements. It is also important to realize that the linear stability does not guarantee stability for non-linear equations. The time step is usually limited by non-linear stability condition which can only be found by experimentation.

Semi-implicit methods try to combine the stability of implicit methods with the efficiency of explicit methods. The idea is to treat only that part of the equations implicitly, which causes the stability problems. For example, in resistive MHD when the diffusion time $(\Delta x)^2/\eta$ becomes the limiting time for Δt, one can treat the resistive terms implicitly, i.e. calculating them at the next time level, while all the other terms explicitly. There is a lot gained relative to the fully implicit method: the size of the matrix to be inverted decreased since only $\mathbf{B}$ and e are implicit, the matrix elements are much simpler, and the structure of the matrix is symmetric and diagonally dominated, which is much easier to solve with iterative methods than the advection dominated fully implicit problem. The main problem with semi-implicit time integration is that the scheme is equation and problem dependent, and the non-linear stability condition is usually more restrictive than that of fully implicit time stepping.

6.4 Some Implicit and Semi-implicit Schemes

The rest of the sections in this chapter are taken directly from [15, 38] with minor modifications.

For sake of simple notation, we rewrite the PDE system (2.1) into an even more compact form

$$\partial_t \vec{U} = \vec{R}(\vec{U}) \tag{6.11}$$

where the *residual* $\vec{R} = -\sum_i \partial_i \vec{F}_i + \vec{S}$.

There are three popular schemes for the (semi-)implicit time integration of (6.11): the temporally first order *backward Euler* scheme (6.10) which can be recommended for steady state calculations, the second order *trapezoidal* method for implicitly treated parabolic source terms, and the second order *Backward Differentiation Formula* (BDF2) for advection dominated time accurate integration. All of these methods can be regarded as a special case of the following general two parameter time discretization

$$\begin{aligned}\vec{U}^{n+1} &= \vec{U}^n + \Delta t_n \vec{R}(\vec{U}^n) \\ &\quad +\alpha \Delta t_n \left[\frac{\vec{U}^n - \vec{U}^{n-1}}{\Delta t_{n-1}} - \vec{R}(\vec{U}^n) \right] \\ &\quad +\beta \Delta t_n \left[\vec{R}_{\rm impl}(\vec{U}^{n+1}) - \vec{R}_{\rm impl}(\vec{U}^n) \right], \end{aligned} \tag{6.12}$$

Here $\vec{R}$ is a conservative high-order discretization (TVD, TVD-MUSCL, or TVDLF), $\vec{R}_{\rm impl}$ contains all the implicitly treated terms, and the α and β parameters may vary between 0 and 1. The scheme is three-level whenever the parameter $\alpha \neq 0$, while for $\alpha = 0$, it only uses time levels n and $n+1$. The backward Euler scheme corresponds to $\alpha = 0, \beta = 1$, and the trapezoidal scheme to $\alpha = 0, \beta = 1/2$. The BDF2 method with a fixed time step requires $\alpha = 1/3, \beta = 2/3$, (for varying time step $\alpha = \Delta t_n/(\Delta t_n + 2\Delta t_{n-1}), \beta = 1 - \alpha$).

In the first time step, when $\vec{U}^{n-1}$ is not available, one has to use the backward Euler or the trapezoidal method. It can be showed [15] that second order spatial accuracy can be maintained even if a first order discretization is used for $\vec{R}_{\rm impl}$ in (6.12). Different semi-implicit options, i.e. when only some of the variables, or some of the terms are treated implicitly are also discussed in detail in [15].

To solve Eq. (6.12) for $\vec{U}^{n+1}$, we linearize it by

$$\begin{aligned}\vec{R}_{\rm impl}(\vec{U}^{n+1}) &= \vec{R}_{\rm impl}(\vec{U}^{n+1}_{\rm expl}, \vec{U}^n_{\rm impl}) \\ &+ \frac{\partial \vec{R}_{\rm impl}}{\partial \vec{U}_{\rm impl}} (\vec{U}^{n+1}_{\rm impl} - \vec{U}^n_{\rm impl}) + \mathcal{O}(\Delta t^2)\end{aligned} \tag{6.13}$$

where $\vec{U}_{\rm impl}$ contains the implicitly treated variables. The explicitly treated variables $\vec{U}_{\rm expl}$, if any, are advanced by an explicit time stepping scheme before Eq. (6.13) is applied.

The linearized fully implicit backward Euler scheme, for example, results in the linear system

$$\left[\frac{\hat{I}}{\Delta t} - \frac{\partial \vec{R}_{\rm impl}}{\partial \vec{U}} \right] \left(\vec{U}^{n+1} - \vec{U}^n \right) = \vec{R}(\vec{U}^n), \tag{6.14}$$

which can be solved by one of the linear system solvers discussed in section 6.6.

6.5 Evaluation of the Jacobian matrix

The *Jacobian matrix*, $\partial\vec{R}_{\mathrm{impl}}/\partial\vec{U}_{\mathrm{impl}}$ in (6.13), can be evaluated in several ways. One may attempt to calculate the partial derivatives analytically, however, this only works if the partial differential equations and their discretization to $\vec{R}_{\mathrm{impl}}$ are relatively simple. In case of the MHD equations even a trivial discretization (like centered differences) will result in rather complicated elements of the Jacobian matrix. Usually a simple shock capturing scheme (like the first order TVDLF1 or TVD-MUSCL1) are used, and the analytical approach becomes very involved even for hydrodynamics [42, 43].

Another approach is to calculate the Jacobian matrix elements *numerically*. We discuss here three options following [15, 38] and describe their basic properties and their applicability. For sake of brevity we use $\vec{R}$ instead of $\vec{R}_{\mathrm{impl}}$.

The *matrix-free* evaluation does not calculate the elements of the Jacobian matrix, only the action of the Jacobian on a vector $\Delta\vec{U}$, when this is needed by the iterative schemes. We simply take the directional derivative of $\vec{R}$ with respect to $\vec{U}$ by taking a difference

$$\frac{\partial\vec{R}}{\partial\vec{U}}\Delta\vec{U} = \frac{\vec{R}(\vec{U}^n + \epsilon\Delta\vec{U}) - \vec{R}(\vec{U}^n)}{\epsilon} \tag{6.15}$$

where ϵ is an appropriately chosen small parameter. The matrix-free method is independent of the spatial discretization and requires very little storage. Unfortunately, the matrix vector multiplication 6.15 can be computationally expensive, and direct linear solvers and preconditioners cannot be used, since the matrix elements are not available. Even the iterative solvers may fail to converge due to the fact that the matrix is effectively perturbed at each iteration by the error in the numerical evaluation of the matrix vector product. The matrix-free approach can be applied for treating parabolic source terms (e.g. resistivity in MHD) implicitly and in combination with the approximately implicit strategy (see section 6.7).

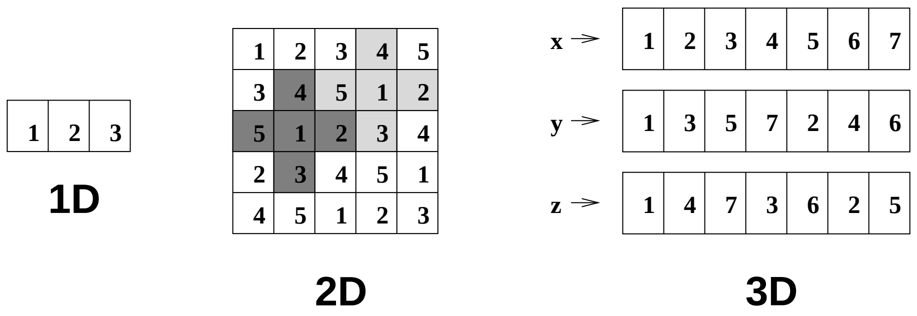

Figure 6.1: Grid masks used for the calculation of the Jacobian matrix. Cells with the same number have non-overlapping stencils.

The *grid masking algorithm* calculates the individual matrix elements numerically in a fairly general way. The idea is to perturb each implicitly treated variable w of $\vec{U}$ in certain spatial patterns, and then to read off the matrix elements from the numerical evaluation of $\vec{R}$

$$\frac{\partial R_j^u}{\partial U_k^w} = \frac{R_j^u(\vec{U} + \epsilon\delta_k^w) - R_j^u(\vec{U})}{\epsilon}. \tag{6.16}$$

In this expression, the superscript u denotes the component of $\vec{R}$ considered, and the grid cell j belongs to the stencil of the perturbed cell k. If we restrict ourselves to spatial discretizations that involve the neighbouring grid cells only, all matrix elements can be determined in $1 + (2D + 1)N_{\mathrm{impl}}$ evaluations of $\vec{R}$, where D is the number of spatial dimensions and N_{impl} is the number of implicitly treated variables.

A more *efficient*, but less general algorithm calculates the Jacobian matrix elements directly from the partial derivatives $\partial\vec{F}_i/\partial\vec{U}$ and $\partial\vec{S}/\partial\vec{U}$ obtained numerically in every cell according to the general rule

$$\frac{\partial f}{\partial U^w} = \frac{f(\vec{U} + \epsilon\delta^w) - f(\vec{U})}{\epsilon}. \tag{6.17}$$

The partial derivatives of $\vec{F}_i$ and $\vec{S}$ are combined according to the spatial discretization scheme, for example TVDLF1. Here some of the work is done analytically, i.e. the influence of boundary conditions, non-local source terms, numerical fluxes etc. on the matrix elements have to be taken into account explicitly, however, the efficient method can calculate the Jacobian matrix about $2D+1$ times faster than the general grid masking algorithm. If even $\partial\vec{F}_i/\partial\vec{U}$ and $\partial\vec{S}/\partial\vec{U}$ were calculated analytically, we would arrive back to the fully analytical approach.

Once the Jacobian matrix elements are calculated, which is a computationally expensive step, direct solvers, preconditioners, and iterative schemes can be applied efficiently.

6.6 Linear System Solvers

In one dimensional calculations the Jacobian matrix will be block tridiagonal, assuming that the scheme used for $\vec{R}_{\mathrm{impl}}$ in (6.12) has a three point stencil. Such a linear system can be efficiently solved by a *direct block tridiagonal solver*. This requires that the Jacobian matrix elements are calculated directly with the grid masking or the efficient algorithm described in section 6.5.

In more than one dimensions, or in combination with the matrix-free evaluation of the Jacobian, the linear system has to be solved by iterative methods. For symmetric positive definite matrices, e.g. resistive source terms on a Cartesian grid, the Conjugate Gradient (CG) scheme is the most efficient, both in terms of memory and CPU cost. For all other types of matrices, the Stabilized Bi-Conjugate Gradient (Bi-CGSTAB) algorithm (van der Vorst [41]) is found to be robust and efficient with a relatively small amount of storage requirement.

In multi-dimensional advection dominated problems preconditioning is vital to accelerate the convergence of the iterative schemes. The Modified Block Incomplete *LU* (MBILU) preconditioner [40] can efficiently precondition penta-diagonal and hepta-diagonal matrices resulting from the nearest neighbour discretization in 2 and 3 dimensions, respectively.

6.7 Approximately Implicit Method

When direct computation of the Jacobian elements is not desirable (e.g. due to high computational cost), we can use the matrix-free approach (see section 6.5) in combination with Minimum Residual Approximated Implicit (MRAI) time integration strategy introduced by Botchev et. al. [1].

In essence, to solve the equation in $\vec{U}^{n+1}$ for a given implicit scheme, the MRAI method uses a restricted and fixed number of iterations with a minimum residual method, such as the Generalized Minimal RESidual (GMRES) algorithm [29]. This allows the time step to be much greater than for an explicit scheme. For time accurate calculations, the initial vector for the GMRES process is a solution obtained with a second order explicit scheme, and the implicit scheme to be approximated is also of the second order, e.g. BDF2. For steady state calculations, the initial vector is taken simply as $\vec{U}^n$, and the implicit scheme is Backward Euler. The order of the MRAI scheme is the minimum of those of the initial vector and the basic implicit scheme, and it does not depend on how accurately the linear system is solved.

Since the MRAI scheme can be seen as a stabilized explicit scheme, it is not unconditionally stable, and it is important to adjust the time step correctly. This is achieved by extracting the spectral information delivered by minimal residual iterations, and requires a negligible amount of extra work. For details see [1].

Chapter 7

Principles of Code Design and Numerical Simulations

In the previous sections a great number of options are listed for both space and time discretization. It is not practical to implement all these schemes and just see which one works best for the problem we want to solve. Instead, one needs to go through a number of steps to make a choice. There are three options: using existing general codes, modifying existing research codes, or writing a new software from scratch. In the latter case the code should be well designed and structured. Doing numerical simulations is not a trivial task, it requires careful planning and testing.

7.1 General Considerations for the Numerical Scheme

First, the class of problems to be solved should be considered. Even if at the beginning it seems that we are interested in doing a single simulation only, unavoidably we will want to solve other related problems later, especially with all the efforts put into the software development. The selected class of problems should be written in a general mathematical form, e.g. like equation (2.1). Using general notation in the software is preferred, since the different variables can be handled in a uniform manner, and changes to the equation can be done easily.

In the selection of the discretization technique several things should be taken into account. The geometry of the computational domain determines if a Cartesian grid is sufficient, or the more general structured, or even unstructured grids are necessary. If we expect discontinuities in the solution, only shock capturing methods should be used, while smooth solutions can be best approximated with high order methods. The time scales determine if explicit time integration schemes are efficient or not. The hardware to be used depends on the size of the problem and of course on the availability of super computers. If parallel machines are needed, the numerical techniques need to parallelise well.

7.2 Using Existing Codes

Before one starts to develop a new code from scratch it is worthwhile to check whether an existing code can be used. It is possible that there already is a working software which needs only minor modifications to solve our problems. Of course, codes written for special problems by other researchers are usually difficult to get, to understand, and to modify. Another option is to use a general software that covers the problems we want to solve. There are a few such general HD and MHD codes publicly available:

- the ZEUS code is a traditional finite difference code with a staggered mesh by Stone and Norman [30] at
  http://zeus.ncsa.uiuc.edu:8080/lca_home_page.html,

- the CLAWPACK and AMRCLAW software pakages by R. J. LeVeque contain TVD-type shock capturing schemes (AMRCLAW has adaptive mesh refinement) mainly for hydrodynamics at
  http://www.amath.washington.edu/~rjl,

- the Versatile Advection Code (VAC) by Tóth [33, 35], described in some detail in chapter 8, at
  http://www.phys.uu.nl/~toth

7.3 Structuring a New Code

Numerical codes become huge with time. It is important to properly structure a code from the beginning, since it will ease the modification and extension at later stages. The two main principles of structuring: no program parts should be repeated and any overly complex part should be split into smaller pieces even if there is no repetition.

Structuring should be done on all scales starting from the use of parameters, functions, and subroutines, to the use of modules and separate programs for separate tasks. The files containing different type of informations should also be well organized.

At the highest level one may have a code INIT for setting up the initial conditions and another code EVOL for evolving the solution in time. Such a separation is desirable for many reasons: the initial condition is stored as a file in the same format as the output data, thus the code can be restarted from an intermediate stage; different evolutions can be done for the same initial data without redoing the calculations involved in the initialization; initial data can be produced by other codes as long as the data format is compatible. Both INIT and EVOL should read parameters from easily editable parameter files. Changing parameters in the source files is not practical, because it requires recompilation, and the earlier parameter settings are lost. On the other hand, the small parameter files can be stored for all past runs and they can be modified without wasting expensive CPU time.

At the level of modules, a good example is the input-output (I/O) module that can be shared by the codes EVOL and INIT, therefore the data formats

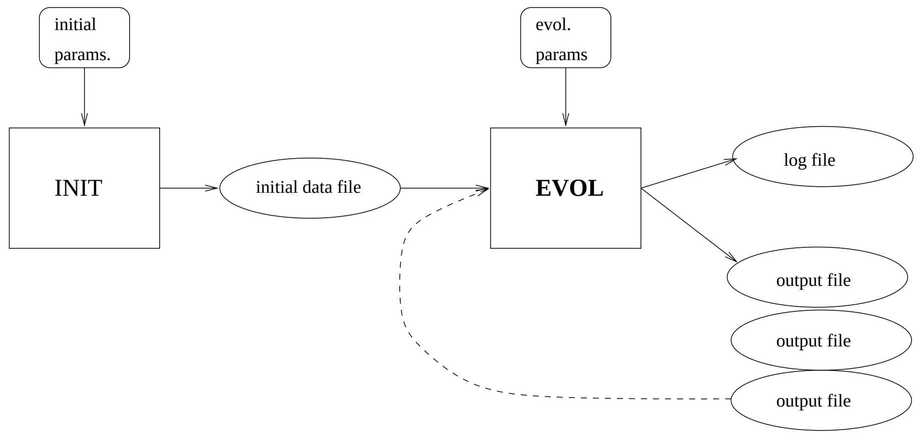

Figure 7.1: Information flow diagram of a well structured code. See discussion in the text.

can be kept consistent easily. Another module may contain all the subroutines that are directly linked to the equations. This allows to use the same numerical techniques on different equations by simply replacing the equation module. Temporal and spatial discretization, grid related subroutines and boundary conditions should also be separated in modules for sake of clarity.

A typical subroutine for the MHD equation can be GETP(U,P) that calculates thermal pressure from the conservative variables U. The thermal pressure is required in several places in the numerical algorithms for different U values, thus it would be foolish to repeat the complicated formulae every time. Having such a subroutine makes it easier to change the equations from full MHD to isothermal MHD, for example. Notice that the subroutine GETP should return an *array* for the full grid rather than a scalar value for a single grid cell, since the overhead in the subroutine or function call would become large. Also, calling functions from a loop is usually difficult to vectorize and parallelize on super computers, while a call to a subroutine with a huge loop inside vectorizes and parallelizes easily. Using similar subroutines to calculate fluxes, source terms, eigenvalues and eigenvectors, the physics can be separated from the numerical scheme.

On the level of parameters the grid size NX, NY, or the number of variables NU should be named constants that can be changed in a single declaration line. This avoids unnecessary frustration when some of these basic parameters should be changed. There is no computational cost involved, since constant parameters are replaced by their values at compilation time. Ideally one should avoid using any numbers in the source code except for very trivial constants like 2 in an averaging. Constant parameters can also be used to make the general notation for the variables easier to read. For example defining the constants IRHO, IRHOVX, IRHOVY, IE, IBX, IBY to be the integers 1,…,NU, one can refer to the density as U(IRHO) in the equation module, while a loop can be done for all variables U(I) in the other modules, where their physical meaning does not matter.

```
Example_mhd22
0 0.0E+00  2  2  6
10 10
1.4E+00  0.0E+00
x y rho rhovx rhovy e bx by   gamma eta
0.5E+00 0.5E+00 2.0E+00 0.0E+00 0.0E+00 3.0E+00 1.0E+00 0.0E+00
1.5E+00 0.5E+00 2.0E+00 0.0E+00 0.0E+00 3.0E+00 1.0E+00 0.0E+00
...
9.5E+00 0.5E+00 2.0E+00 0.0E+00 0.0E+00 3.0E+00 1.0E+00 0.0E+00
0.5E+00 1.5E+00 2.0E+00 0.0E+00 0.0E+00 3.0E+00 1.0E+00 0.2E+00
...
...
```

Figure 7.2: A formatted 2D MHD initial data file for the Versatile Advection Code. The 5 header lines contain the header string; the parameters IT, T, NDIM, NPAR, NU; the grid size NX, NY; the equation parameters GAMMA ETA; and the variable name string. The header is followed by the coordinates X and variables U for the NX$\times$NY cells. In the real file 10 decimals are saved.

7.4 Data Format

As it has already been mentioned, the initial data and the output data should be in the same format for sake of easy restart. The data file should contain as a minimum the grid size NX, NY, the coordinates X, Y, the number of conservative variables NU and their value U, the physical time T and the time step counter IT. If there are adjustable equation parameters, like γ and η in the MHD equations, there number NPAR and values PAR should also be given. It is very useful to include strings describing the problem, and the names of the variables and equation parameters. Having the sizes of the arrays in the data file makes it easy to read the arrays in one block. If the output file contains several snapshots at different times, the size of the snapshots can be calculated from the array sizes and one can jump directly to the position of a given snapshot.

The best format of the data files depends on the purpose. For an accurate restart one should save the current state in the binary format of the machine. The advantage of binary format is accuracy, economic disk space usage, fast reading and writing. The disadvantage is that the file cannot be ported between different machines unless they have compatible binary representation; it is difficult to read the file directly; and the visualization software may not be able to read arbitrary binary data formats. The other extreme is storing the data in ASCII files. The obvious advantage is portability between machines and softwares, the disadvantage is some loss in accuracy and slow input and output. On parallel super computers reading and writing formatted data files can become a bottleneck of performance. ASCII data files tend to be huge, although their size can be reduced by standard compression programs. There are compromises between the two extremes, such as the XDR format, which is portable between machines having the XDR libraries, and still economical. Ideally one should have a choice in the code for the format of the data files.

Due to limitations in available disk space and the CPU time spent on I/O, one can usually save at most a few dozen or hundred snapshots from a simulation

```
data/example.ini
Example_mhd22
x y rho rhovx rhovy e bx by
domain
10     10              -nx ny
 0.     0.             -xmin ymin
10.    10.             -xmax ymax
uniform
2. 0. 0. 3. 1. 0.      -rho rhovx rhovy e bx by
eqpar
1.4 0.                 -gamma eta
save
```

Figure 7.3: An example parameter file for the initial data shown in figure 7.2. The `domain, eqpar, uniform, save` strings are interpreted as *actions* by the initialization code, and the appropriate subroutine reads the consequtive parameters. The Fortran READ command ignores the comments following numerical values, which can be used for documentation.

that may consist of ten or hundred thousands of time steps. The progress of the computation may be best followed by writing out some overall information, like the physical time, the time step, some interesting global quantities (e.g. total kinetic energy for an instability or the residual for steady state calculations) much more frequently into a *logfile*. Usually the logfile should be viewed directly, so it should be in ASCII format. The logfile should also have a header that identifies the problem and describes the quantities saved into it. One may also save partial information for visualization purposes more frequently than the full data is saved but less frequently than the logfile is written.

Parameter files should always be in easily editable ASCII format. Fortran provides the rather flexible *namelists* that allow setting parameters in arbitrary order by using their names. Unfortunately, the exact format of namelists has not been standardized in Fortran 77, and even some of the existing Fortran 90 compilers fail to be consistent with the standard. One can, of course, mimic the behavior of namelists, but that can result in a rather lengthy program if there are many parameters to set. The alternative is to use a fixed order of reading parameters.

The parameter file for the initial condition code INIT should define the file name for the initial data, the string describing the problem, the grid resolution and the coordinate values, the parameters for setting the initial condition for U, and possibly the intended equation parameters. The parameter file for the EVOL code should contain the input, output and log filenames, the frequency of saving information, and parameters for stopping condition, for the numerical methods, and for the boundary conditions. It is important to use expressive parameter *strings* to select among several options. The use of numbers should be restricted to parameters of truly numerical nature, such as the maximum number of time steps.

Data files and parameter files belonging to the same problem should have similar names, or should be in a separate directory.

```
&filelist
       filenameini='data/example.ini',
       filename=   'data/example.log',
                   'data/example.out'
/

&savelist
        ditsave=1,10
/

&stoplist
        itmax=100
/

&methodlist
       typefull=       6*'tvdlf'
       typelimiter=    6*'minmod'
       eqpar(2)=       0.01
       dimsplit=       .false.
       constrainB=     .true.
/

&boundlist
       typeB= 24*'periodic'
/

&paramlist
        courantpar=0.4
/
```

Figure 7.4: An example parameter file for the Versatile Advectom Code that can evolve the initial data shown in figure 7.2. Namelists are used to set the filenames, file saving frequency, stopping condition, solution method, boundary conditions, and time step size. Note the flexibility of namelists in setting full arrays and array elements.

7.5 Doing Simulations

New or modified codes should be thoroughly tested on simple problems with known solutions before an attempt is made to do the simulations we are interested in. If possible, the first simulations should be done at a lower dimension using some symmetry of the simplified problem. Doing a simulation in 1 or 2 spatial dimensions does not mean that the vector variables $\rho\mathbf{v}$ and $\mathbf{B}$ should have only 1 or 2 components respectively. When all 3 components of the vector variables are represented, but only two spatial coordinates are used, it is commonly called a *2.5D simulation.*

The initial condition should be carefully checked, it is very easy to make a mistake in setting it up. It is also worthwhile to do a few time steps at a low grid resolution and check whether the results make sense, and in particular, to check the boundary conditions. The number of time steps, the grid resolution, and the dimensionality of the problem can be increased once we are confident about the early results. If possible, check the code in higher dimension against the lower dimension results by doing physically equivalent simulations.

After all these steps, we may proceed to solve the real problem for which the solution is not known. The results of the simulations should be handled with extreme care. They should be checked against physics (e.g. conservation of quantities not built explicitly into the numerical scheme), against other numerical methods (if available), and most importantly, against grid refinement, i.e. *the solution should converge as the spatial and temporal resolution is increased.*

7.6 Numerical Convergence

To verify numerical convergence, the same physical problem should be solved with at least three different resolution for the the same physical time. The global difference between the high, middle, and coarse resolution solutions U^H, U^M, U^C can be calculated as

$$E^2_{HM} = \sum_j (U^H_j - U^M_j)^2 \qquad (7.1)$$

$$E^2_{CM} = \sum_j (U^M_j - U^L_j)^2 \qquad (7.2)$$

The differences should be taken at the grid points j corresponding to the coarsest resolution. Let us assume that the solutions are related to the unknown analytical solution U as

$$U^{H,M,C}_j = U_j + (\Delta x_{H,M,C})^k E_j \qquad (7.3)$$

where E_j is the local numerical error for $\Delta x = 1$ and k is the convergence rate. Substituting (7.3) into (7.1) two equations are obtained for two unknowns: the convergence rate k and the global error $\sum E^2_j$.

The procedure outlined above has been taken from Stone et. al. [31].

Chapter 8

Versatile Advection Code

The Versatile Advection Code (VAC) [33, 35] is a general tool for solving hydrodynamic and magnetohydrodynamic problems arising in astrophysics. The software package has been developed by the author since 1994 as part of the project on 'Parallel Computational Magneto-Fluid Dynamics', funded by the Dutch Scientific Research Foundation (NWO) Priority Program on Massively Parallel Computing. Starting from 1996 other researchers of the project joined the development: R. Keppens joined the general software development and maintainance efforts and A. van der Ploeg and M. Botchev contributed significantly to the linear system solvers and the implicit time integration schemes.

VAC is available via registration at the *http://www.phys.uu.nl/~toth* home page. The latest version VAC 3.0 has been distributed to 30 users.

8.1 Equation Modules

VAC aims to solve a set of conservation laws with source terms of the form equation (2.1). The actual equations are separated in modules.

At the moment five equation modules are implemented: the simple transport equation (1.5) for test purposes, the Euler equations of compressible hydrodynamics with adiabatic or full energy equation (eqs.1.5–1.7 with $\mathbf{B} = 0$), and the resistive MHD equations with isothermal or full energy equation (1.5–1.8). Source terms for external gravity, heat conduction, and viscosity can be included and modified as library subroutines. Other source terms can be defined in a user written subroutine in the VACUSR module. The equation modules can be combined with the various source terms, e.g. the Euler equations with the viscosity source terms are the compressible Navier-Stokes equations.

All equation-modules are written in the dimension independent notation (see section 8.5), thus the number of spatial dimensions $1 \le D \le 3$ and the number of components $D \le C \le 3$ for the vector variables (e.g. the momentum and the magnetic field) may have 6 different combinations.

8.2 Grid and Boundary Conditions

The equations can be solved on a 1, 2 or 3D structured grid with conservative finite volume discretization. A structured grid can be thought of as a continuous mapping from a Cartesian mesh. Structured grids include Cartesian, polar and spherical grids as trivial examples, but they can be used in more complicated geometries as well. In 1D and 2D both slab and cylindrical symmetry may be assumed for the ignored dimension(s).

Boundary conditions are implemented by two layers of *ghost cells* around the mesh. The values in the ghost cells are updated in every time step, thus the discretized equations can be applied to the cells inside the physical mesh without making special cases for the edges. This way the boundary conditions are fully specified in the subroutines updating the ghost cells, and the boundary conditions become independent of the choice of the numerical schemes. For each boundary and each variable the type of the boundary can be defined. At present there are seven predefined boundary types: *periodic, symmetric, antisymmetric, fixed, fixed-gradient* and *continuous.* The continuous boundary condition is an approximate representation of an open boundary by putting the gradients normal to the boundary to zero in the ghost cells. The fixed-gradient type is similar, except that the gradient can be different from zero. Time dependent or more complicated boundary conditions can be realized via the `special` boundary type which results in a call of a user defined subroutine.

8.3 Spatial Discretization

At present four algorithms are available for solving the differential equations. The Flux Corrected Transport scheme (FCT), two Total Variation Diminishing schemes with a Roe-type Riemann solver (TVD and TVD-MUSCL), and a Lax-Friedrichs type TVD scheme (TVDLF) with no Riemann solver. See [39] for a description of these schemes and the references provided there.

All these schemes are second order in space and they are able to simulate shocks and other discontinuities as well as smooth flows. Multidimensional equations are solved either by Strang type dimensional splitting, or by adding the fluxes from all directions at the same time. Source terms may be included in a time split fashion or added at the same time as the fluxes.

The well-known FCT schemes solve each equation separately. Artificial diffusion is added only where it is needed for numerical stability. The FCT schemes are good in resolving contact discontinuities, they can handle shocks well, but close to the discontinuities some numerical noise might be generated. The TVD schemes on the other hand are very efficient in maintaining monotone profiles by appropriately limiting the slopes of the variables. The TVD schemes with the approximate Riemann solver can resolve discontinuities as well as or better than FCT without the spurious oscillations, but the Riemann solver is specific to the equation, and the TVD and TVD-MUSCL schemes are not as robust as the TVD Lax-Friedrichs scheme, which does not need a Riemann solver at all. The TVDLF scheme is somewhat more diffusive than the other two.

Both the projection scheme and the constrained transport methods are implemented to eliminate the numerically generated divergence $\mathbf{B}$ when the MHD equations are solved. The numerical instability related to $\nabla \cdot B$ can also be controlled by Powell's MHD Riemann solver and the corrective source terms.

8.4 Temporal Discretization

Second order time discretization is achieved by a 2-step Runge-Kutta scheme for FCT, and a predictor-corrector method, using a Hancock predictor step, for the TVDLF and TVD-MUSCL methods. The single-step TVD scheme is second order in time by making use of the eigenvalues, however for this method source terms are difficult to include with second order accuracy. Higher order Runge-Kutta time integration schemes are also implemented, for certain problems they can be numerically more stable than the two-step schemes.

Fully implicit and semi-implicit time integration schemes [15, 38] are also available, both for time accurate and steady state problems. The Jacobian matrix is evaluated numerically. The resulting linear system is solved by a block tridiagonal solver in 1D, while in multidimensional simulations preconditioned [40] iterative schemes are used.

8.5 Software Design

The philosophy behind VAC is using a single versatile software with options and switches for various problems, rather than developing a different method or version for each problem separately. The advantage of such a general approach is a reduction of overall time for software development, easier maintenance, compatibility of different parts, automatic extension of new features to all existing applications. The price of the general approach is some added complexity in the source code.

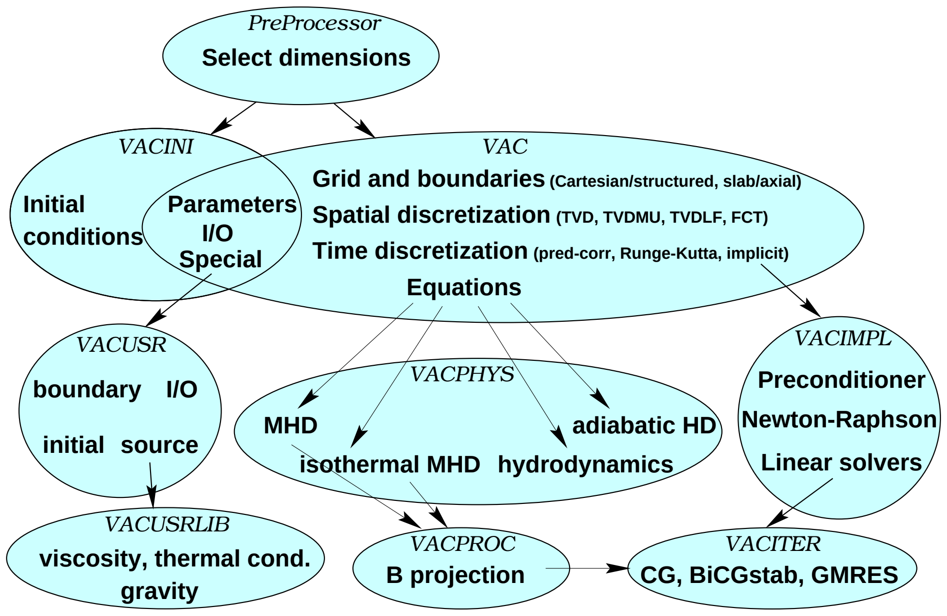

Figure 8.1: The structure of the Versatile Advection Code

VAC is written in a dimension independent notation using the Loop Annotation SYntax (LASY) [34] which is translated by a special-purpose preprocessor to Fortran 90, and it can be further translated to Fortran 77 or High Performance Fortran (HPF). The preprocessor and the translator scripts are written in the Perl language. The Perl interpreter is freely available from the net, and it is already installed on most UNIX systems. The source code expresses data parallelism clearly, which is important for vector and parallel computers as well, it can be compiled on any parallel machine with an HPF compiler. VAC has been installed and is being used on Pentium PC-s under LINUX, on a number of workstations (DEC, SGI, IBM, SUN, HP), on the vector super computers Cray C90 and Cray J90, and on the parallel machines Cray T3D, Cray T3E, IBM SP, and Connection Machine CM5. Good performance [36] has been found on all platforms.

The source code is divided into smaller units, called modules. Some modules have several versions, for example the VACPHYS module which contains the equation specific information: fluxes, sources, the eigenvalues and eigenvectors for the approximate Riemann solver, etc. Each system of equations is specified by its respective VACPHYS.EQUATION module. The boundary conditions and the initial conditions are often problem specific, these special subroutines are collected in the VACUSR.PROBLEM modules. This is the only part of the source code which the user is expected to modify, and it can be written in the dimension independent notation as well as in Fortran 90 or Fortran 77.

8.6 Software Environment

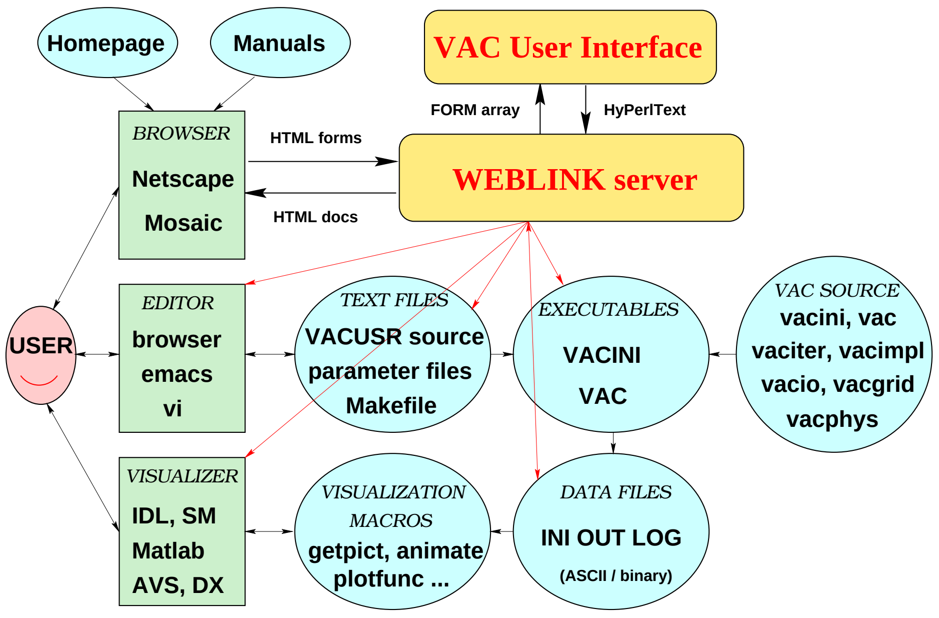

Figure 8.2: Information flow in the VAC software package

The documentation, about 120 pages, is written in hypertext markup language (HTML), which can be viewed and/or printed from the standard web browsers, e.g. Netscape or Mosaic. There is a user interface written in Perl, which lets the user to configure, edit, compile, and run VAC, and visualize results through the same web browser. Of course, one can do all these steps from the UNIX shell as well. The manual pages and a demo version of the interface can be looked at the VAC home page.

The software package contains visualization macros that can read, process, and visualize the ASCII/binary data files for the visualization softwares IDL, Matlab and SM (SuperMongo). Conversion for Gnuplot, and to AVS and DX file formats is also supported.

Chapter 9

Some Astrophysical Applications

In this chapter, a few astrophysical applications of computational magnetohydrodynamics, which the author has been involved with recently, are described. Each section starts with the authors and the abstract of the paper or manuscript. The abstract is followed by comments on the numerical techniques used.

9.1 Nonlinear MHD Simulations of Wave Dissipation in Flux Tubes

Stefaan Poedts, Sander Beliën & Gábor Tóth,
paper appeared in Solar Physics [22]

The phase-mixing and resonant dissipation of Alfvén waves is studied in both the 'closed' magnetic loops and the 'open' coronal holes observed in the hot solar corona. The resulting energy transfer from large to small length scales contributes to the heating of these magnetic structures. The non-linear simulations show that the periodically varying shear flows that occur in the resonant layers are unstable. In coronal holes, the *phase-mixing* of running Alfven waves is *speeded-up* by the 'flaring-out' of the magnetic field lines in the lower chromosphere.

The closed loop equations are studied by the HEating by Resonant Absorption (HERA) [16] code solving the 3D MHD equations in cylindrical coordinates. The numerical scheme uses finite differences in the radial direction and a pseudo-spectral discretization along the φ and z coordinates. Semi-implicit time stepping is used.

The open loop problem is studied by VAC solving the MHD equations on a 2.5D generalized grid assuming axial-symmetry in the 3rd dimension. The initial condition and the grid for a flaring loop was produced by a finite element code, PARIS. Alfvén waves were induced by shaking the field lines at the chromosphere and their propagation into the corona is calculated by VAC.

9.2 Numerical Simulations of Prominence Oscillations

N. A. J. Schutgens and G. Tóth,
manuscript submitted to Astronomy & Astrophysics

We present numerical simulations, obtained with the Versatile Advection Code, of the oscillations of an inverse polarity prominence. The internal prominence equilibrium, the surrounding corona and the inert photosphere are well represented. Gravity and thermodynamics are not taken into account, but it is argued that these are not crucial. The oscillations can be understood in terms of a solid body moving through a plasma. The mass of this solid body is determined by the magnetic field topology, not by the prominence mass proper. The model also allows us to study the effect of of the ambient coronal plasma on the motion of the prominence body. Horizontal oscillations are damped through the emission of sound waves while vertical oscillations are damped through the emission of Alfvén waves.

The isothermal MHD equations are solved on a 2.5D non-uniform Cartesian grid with the FCT algorithm. The initial conditions for the arcade and the prominence are obtained by a sophisticated relaxation procedure. Then the stationary prominence is kicked, and the the resulting damped oscillations are calculated by the code. The non-reflecting coronal boundary conditions are achieved by the strongly stretched grid, which places the boundaries far away. It is more difficult to represent the photosphere, which behaves like a solid wall with frozen in field lines that make an angle.

9.3 Simulations of Small-Scale Explosive Events on the Sun

D. E. Innes and G. Tóth,
manuscript accepted by Solar Physics

Small-scale explosive events or microflares occur throughout the chromospheric network of the Sun. They are seen as sudden bursts of highly Doppler shifted spectral lines of ions formed at temperatures in the range $2 \times 10^4 - 5 \times 10^5$ K. They tend to occur near regions of canceling photospheric magnetic fields and are thought to be directly associated with magnetic field reconnection. Recent observations have revealed that they have a bi-directional jet structure reminiscent of Petschek reconnection. In this paper compressible MHD simulations of the evolution of a current sheet to a steady Petschek, jet-like configuration are computed using the Versatile Advection Code. We obtain velocity profiles that can be compared with recent ultraviolet line profile observations. By choosing initial conditions representative of magnetic loops in the solar corona and chromosphere, it is possible to explain the fact that at Sun center the jet flowing outward into the corona is more extended and seen before the jet flowing towards the chromosphere. Although this model can reproduce the high Doppler shifted components of the line profiles, the brightenings at low velocities, near the center of the bi-directional jet, cannot be explained by this simple MHD model.

The resistive MHD equations are solved on a 2D non-uniform Cartesian grid with the TVDLF method. Radiative cooling source terms are used in the energy equation. The initial conditions are relatively simple. Reconnection of antiparallel field lines is induced by a localized anomalous resistivity. The time evolution of the jet formation can be best followed by explicit time integration, while the final steady state can be more efficiently calculated with a fully implicit scheme. Line profiles of the emission coming from the explosive event are calculated from the simulation data saved by VAC.

9.4 Growth and Saturation of the Kelvin-Helmholtz Instability with Parallel and Anti-Parallel Magnetic Fields

R. Keppens, G. Tóth, R. H. J. Westermann, J. P. Goedbloed,
manuscript accepted by the Journal of Plasma Physics

We investigate the Kelvin-Helmholtz instability occurring at the interface of a shear flow configuration in 2D compressible magnetohydrodynamics (MHD). The linear growth and the subsequent non-linear saturation of the instability are studied numerically. We consider an initial magnetic field aligned with the shear flow, and analyze the differences between cases where the initial field is unidirectional everywhere (uniform case), and where the field changes sign at the interface (reversed case). We recover and extend known results for pure hydrodynamic and MHD cases with a discussion of the dependence of the non-linear saturation on the wavenumber, the sound Mach number, and the Alfvénic Mach number for the MHD case.

A reversed field acts to destabilize the linear phase of the Kelvin-Helmholtz instability compared to the pure hydrodynamic case, while a uniform field suppresses its growth. In resistive MHD, reconnection events almost instantly accelerate the buildup of a global plasma circulation. They play an important role throughout the further non-linear evolution as well, since the initial current sheet gets amplified by the vortex flow and can become unstable to tearing instabilities forming magnetic islands. As a result, the saturation behaviour and the overall evolution of the density and the magnetic field is markedly different for the uniform versus the reversed field case.

The 2D MHD (both ideal and resistive) equations are solved on a uniform Cartesian grid with the one step TVD scheme using dimensional splitting. The results are compared with the purely hydrodynamic case as well.

9.5 On the Azimuthal Stability of Shock Waves around Black Holes

D. Molteni, G. Tóth, O. A. Kuznetsov,
manuscript submitted to the Astrophysical Journal

Earlier analytical studies and numerical simulations of time dependent axially symmetric flows onto black holes have shown that it is possible to produce stationary shock waves with a stable position both for ideal inviscid and for moderately viscous accretion disks.

We perform several two dimensional numerical simulations of accretion flows in the equatorial plane to study shock stability against non-axisymmetric azimuthal perturbations. We find that a very small perturbation can produce an instability as it crosses the shock. After some small oscillations, the shock wave suddenly transforms into a partially spiral, but closed pattern, and it stabilizes with a finite radial extent.

The main characteristics of the final flow are: 1) The deformed shock rotates steadily without any damping. It is a permanent feature and the thermal energy content and the emitted energy vary periodically with time. 2) This behavior is also stable against further perturbations. 3) The shock is still very strong and well defined. The average radial distance of the deformed shock is somewhat greater than that of the axially symmetric circular shock. 4) The instability occurs in a wide range of parameters, thus it may have relevant observational consequences, like (quasi) periodic oscillations, for the accretion of matter onto black holes. Typical time scales for the periods are 0.01 and 1000 seconds for black holes with 10 and 10^6 solar mass, respectively.

Here we solve the HD equations with pseudo-relativistic gravitational source terms that approximate the relativistic effects. The axially symmetric initial conditions can be best obtained by a fully implicit 1.5D integration. The 1D result is rotated around the symmetry axis and perturbed by a non-axisymmetric perturbation. The instability is evolved in time on a 2D polar grid with an explicit time integration using the one step TVD scheme with operator splitting.

Chapter 10

Closure

Numerical simulation of MHD flows is not a simple problem. There is no one "perfect" numerical scheme that could solve, or which would be optimal for, all possible problems. It is important to know the available methods, their strengths and weaknesses, before one starts to write a program. Due to the complexity of the numerical algorithms it may be worthwhile to use or modify existing software. This allows the researcher to concentrate on the simulations and not on the software development. Numerical codes and the results of the simulations should always be thoroughly tested and verified. On the other hand, well designed numerical simulations can provide a wealth of information on the physical processes which we could otherwise only observe.

Acknowledgements. The author receives a postdoctoral fellowship (D 25519) from the Hungarian Science Foundation (OTKA). He thanks Prof. Duarte for his kind invitation to give a series of lectures on the subject of Computational MHD at the University of Porto. The presentation in these notes has gained a lot from the excellent notes on "Nonlinear Conservation Laws and Finite Volume Methods" by R. LeVeque [18].

Bibliography

[1] Botchev M. A., Sleijpen G. L. G., van der Vorst H. A. 1997, Department of Mathematics, Utrecht University, Preprint 1043, December, also at http://math.ruu.nl/publications/.

[2] Brackbill J. U., Barnes D. C. 1980, J. Comput. Phys. 35, 426

[3] Brio M., Wu C. C. 1988, J. Comput. Phys. 75, 400

[4] Cockburn B., Shu C. W. 1998, J. Comput. Phys. 141, 199

[5] Dai W., Woodward P. R. 1998, Astrophys. J. 494, 317

[6] DeVore R. C. 1991, J. Comput. Phys. 92, 142

[7] Draine B. T. 1980, Astrophys. J. 241, 1021

[8] Draine B. T., McKee C. F. 1993, Ann. Rev. Astr. Ap. 31, 373

[9] Einfeldt B., Munz C. D., Roe P. L., Sjögreen B. 1991, J. Comput. Phys. 92, 273

[10] Evans C. R., Hawley J. F. 1988, Astrophys. J. 332, 659

[11] Fletcher C. A. J. 1991, *Computational Techniques for Fluid Dynamics*, (Springer, Berlin)

[12] Gombosi T. I., Powell K. G., De Zeeuw D. L. 1994, J. Geophys. Res. 99, 21525

[13] Harten A. 1983, J. Comput. Phys. 49, 357

[14] Hirsch C. 1990, *Numerical Computation of Internal and External Flows* (Wiley, New York)

[15] Keppens R., Tóth G., Botchev M. A., van der Ploeg A. 1997, Implicit and Semi-Implicit Schemes in the Versatile Advection Code: algorithms, submitted for publication to the Int. J. Num. Meth. in Fluids

[16] Keppens R., Poedts S., Goedbloed J. P. 1998, Lecture Notes in Comput. Sci. 1401, 233

[17] LeVeque R. J. 1992, *Numerical Methods for Conservation Laws* (Birkhäuser-Verlag)

[18] LeVeque R. J. 1997, in *Computational Methods for Astrophysical Fluid Flow*, Saas-Fee Advanced Course 27 Lecture Notes, eds. O. Steiner and A. Gautschy (Springer-Verlag)

[19] Lax P. D., Wendroff B. 1960, Commun. Pure Appl. Math. Vol. 13, No. 140, 848

[20] Lele S. K. 1992, J. Comput. Phys. 103, 16

[21] Lowrie R. B., Roe P. L.S. K., van Leer B. 1995, AIAA paper 95-1658

[22] Poedts S., Tóth G., Goedbloed J. P., Beliën A. J. C. 1997, Solar Physics 172, 45

[23] Powell K. G. 1994, ICASE Report No 94-24, Langley, VA

[24] Quirk J. J. 1994, Int. J. Num. Meth. Fluids. 18, 555

[25] Roe P. L. 1981, J. Comput. Phys. 43, 357

[26] Roe P. L., Balsara D. 1996, SIAM J. Appl. Math. 56, 57

[27] Ryu D., Jones T. W., Frank A. 1995, Astrophys. J. 452, 785

[28] Ryu D., Brown G. L., Ostriker J. P., Loeb A. 1995, Astrophys. J. 452, 364

[29] Saad Y., Schultz M. H. 1986, SIAM J. Sci. Stat. Comput. 7(3), 856

[30] Stone J. M., Norman M. L. 1992, Astrophys. J. Suppl. 80, 753

[31] Stone J. M., Hawley J. F., Evans C. R., Norman M. L. 1992, Astrophys. J. 388, 415

[32] Strang G. 1968, SIAM J. Numer. Anal. 5, 506

[33] Tóth G. 1996, Astrophys. Lett. & Comm. 34, 245

[34] Tóth G. 1997, J. Comput. Phys. 138, 981

[35] Tóth G. 1997, Lecture Notes in Comput. Science 1225, 253

[36] Tóth G. 1998, Lecture Notes in Comput. Science 1241, 368

[37] Tóth, G. 1994, Astrophys. J. 425, 171

[38] Tóth G., Keppens R., Botchev M. A. 1997, Astron. & Astrophys. 332, 1159

[39] Tóth G., Odstrčil D. 1996, J. Comput. Phys. 128, 82

[40] van der Ploeg A., Keppens R., Tóth G. 1997, Lecture Notes in Comput. Science 1225, 421

[41] van der Vorst H. A. 1992, SIAM J. Sci. Statist. Comput. 13, 631

[42] Yee H. C. 1989, NASA Tech. Mem. 101088

[43] Yee H. C. 1990, J. Comput. Phys. 88, 31

[44] Yee H. C., Sweby P. K. 1997, RIACS Tech. Mem., also J. Comput. Phys. 38

[45] Yee H. C. 1996, Explicit and Implicit Compact High-Resolution Shock-Capturing Methods I: Formulation, accepted for publication by J. Comput. Phys.