Chapter 4

A finite volume ideal MHD code

Numerical simulations of MHD flows with shocks have been performed for many decades, but most of the time rather primitive numerical techniques like Lax-Wendroff [90] or Flux Corrected Transport [28] were used. Numerical codes typically either were excessively diffusive or produced spurious oscillations near discontinuities. In the mean time robust and accurate shock-capturing numerical techniques of the so-called high-resolution type have been developed for Computational Fluid Dynamics (CFD) simulations. Only since a decade or so, these new numerical methods have been introduced in MHD codes [12, 157, 158, 183, 23, 24, 128, 129, 1, 118, 127, 162, 6, 7, 99, 36, 26, 111]. The MHD system has certain peculiar properties which impeded the straight translation of the CFD techniques, but it seems that the main obstacles have been removed in recent work by Roe and Balsara (1996) [127], who provided a well-behaved eigensystem decomposition of the MHD Jacobian, and by Powell (1995) [118], who provided a consistent and efficient way to deal numerically with the $\nabla \cdot \vec{B} = 0$ constraint via a source term (the right hand side term proportional to $\nabla \cdot \vec{B}$ in Eq. 2.6).

In this Chapter we report on our development of the PAR-MA (PARallel MAgnetohydrodynamics) code, which is a massively parallel shock capturing MHD code. Using the proven CFD concepts present in the von Karman institute multi-block solver [106], and incorporating the essential contributions of Roe and Powell [1], a new robust and accurate high-resolution MHD flow solver was developed. After a general introduction which illustrates some of the basic concepts of numerical simulation of flow problems with shocks, we give a brief but complete description of the MHD flow solver we have developed. This is the numerical code used for the simulations presented in this dissertation.

Numerical techniques similar to the one described in this Chapter are currently being used in several application codes [118, 162, 124, 117]. Not much can be found in the literature on rigorous validation of this approach, however. We have carried out our MHD code development to the point of its validation by a set of careful grid convergence studies making use of characteristic theory. These studies are reported in Chap. 5 which should be read as a complement to Chap. 4.

4.1 Upwind discretization of conservation laws in one space dimension

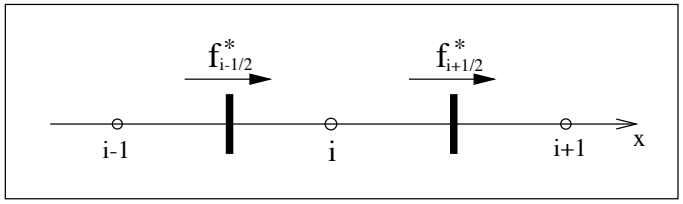

Figure 4.1: *The x-axis is divided into cells of equal length Δx for the numerical computation of the value u_i^n in every cell which approximates the solution $u(x,t)$ at discrete position $x = x_i$ and time $t = t_n$.*

In this Section we introduce some of the basic concepts used in the design of numerical schemes for equations which allow for discontinuous solutions. A good introduction to this subject can be found in [90]. It is beyond the scope of this Section to give a rigorous and complete account of this subject, but we want to give an intuitive introduction for the reader who is unfamiliar with this kind of numerical techniques, and at the same time we may remind the more knowledgeable reader of some of the basic concepts. The reader will note a general conceptual analogy between the discussion in this Section and the discussion of waves and characteristics in hyperbolic systems presented in Sec. 3.2.1. Most of the discussion in the present Section is situated in the context of *finite difference* numerical discretizations. By the end of the Section, we make the link to *finite volume* discretizations.

4.1.1 Scalar conservation laws

A *scalar* conservation law is described by

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{4.1}$$

in terms of scalar u and flux function $f(u)$. The *linear* scalar conservation law or linear advection equation is given by

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{4.2}$$

with linear flux function $f(u) = a\,u$ for some constant a. Let us assume for the moment that $a > 0$, such that advection proceeds from left to right. The left direction is thus called the *upwind* direction. Obtaining solutions of this equation on a computer requires *discretization* of the equation. We divide both the x and t axes in equal intervals and look for values u_i^n which approximate the solution $u(x,t)$ at discrete position $x = x_i = i\,\Delta x$ and time $t = t_n = n\,\Delta t$ (Fig. 4.1).

A natural way to discretize Eq. 4.2 is to approximate the temporal and spatial derivatives by *finite differences* in the following way

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0, \tag{4.3}$$

The temporal derivative has been taken *forward* in time, such that a value at time $t = t_{n+1}$ can be calculated from values at time $t = t_n$, leading to an *explicit* numerical scheme. In the rest of this Section we only consider explicit time integration. The *central* spatial discretization in Eq. 4.3 is an approximation of the derivative which is accurate up to second order in Δx. Stability analysis shows, however, that this central discretization results in a numerical scheme which is *numerically unstable*.

Alternatively, we can use the following *upwind* discretization for the spatial derivative

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \tag{4.4}$$

The derivative is now approximated by a finite difference chosen on the *upwind* side of point i. Stability analysis shows that this discretization is numerically stable under the condition that

$$\Delta t < \frac{\Delta x}{a}, \tag{4.5}$$

which is called the Courant-Friedrichs-Lewy (CFL) condition. This spatial discretization is, however, accurate only to first order in Δx.

More generally, if the sign of a is not specified in advance, then the upwind discretization can be written as

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a^+\frac{u_i^n - u_{i-1}^n}{\Delta x} + a^-\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0, \tag{4.6}$$

with the definitions: $a^+ = max(0, a)$ and $a^- = min(0, a)$.

This upwind discretization can be written in *conservative form*

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{f_{i+1/2}^{n*} - f_{i-1/2}^{n*}}{\Delta x} = 0, \tag{4.7}$$

with the *numerical flux function* f^{n*} (Fig. 4.1) given by

$$f_{i+1/2}^{n*} = a^+ u_i^n + a^- u_{i+1}^n = \frac{a\,u_{i+1}^n + a\,u_i^n}{2} - \frac{1}{2}|a|(u_{i+1}^n - u_i^n). \tag{4.8}$$

The conservative form of the equations reflects on the discrete level the conservation law form of the PDE, and guarantees *discrete conservation*: $\sum_i u_i^{n+1} = \sum_i u_i^n$. This property turns out to be crucial for the numerical simulation of problems with discontinuities. Indeed, the *Lax-Wendroff theorem* guarantees that a conservative numerical scheme converges to a weak solution of the conservation law, if it converges. This is an important property, because non-conservative numerical schemes can for instance converge to solutions with shocks which propagate at wrong speeds.

The upwind discretization is attractive, but it is too *diffusive*, due to the fact that it is only first order accurate. This can be seen by rewriting Eq. 4.6 in the form

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - \frac{|a|\Delta x}{2}\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = 0, \tag{4.9}$$

which can be interpreted to be a second order accurate discretization of the *linear advection-diffusion equation*

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = \eta\frac{\partial^2 u}{\partial x^2}, \tag{4.10}$$

with diffusion coefficient $\eta_{num} = |a|\Delta x/2$. This *numerical diffusion* vanishes in first order in Δx, and thus makes Eq. 4.9 a first order accurate discretization of the linear advection equation 4.2. If one solves the ideal conservation law Eq. 4.1 numerically, one naturally wants to minimize the amount of numerical diffusion in the discretization. However, a certain amount of numerical diffusion is necessary to obtain a stable scheme: the second order accurate central discretization, which does not introduce any numerical dissipation, is numerically unstable. This allows us to interpret the expression for the numerical flux function in Eq. 4.8 as the sum of a term corresponding to a central second order accurate discretization without numerical diffusion, and a term proportional to $|a|$ which represents numerical diffusion.

It is also interesting to remark that the discretization Eq. 4.9 of the scalar advection-diffusion equation 4.10 is only numerically stable under

the condition

$$\Delta t < \frac{(\Delta x)^2}{2\eta_{num}}. \tag{4.11}$$

The upwind scheme thus introduces the *maximal* numerical diffusion $\eta_{num} = |a|\Delta x/2$ which is dissipatively stable under the constraint of the CFL condition, in the sense that adding more (numerical) dissipation would make the dissipative time step limitation more stringent than the CFL condition.

Spurious oscillations are another point of concern. They do not arise for numerical schemes which have the property of *positivity* [7]. We discretize the spatial derivative of Eq. 4.1 on a three–point stencil and obtain as a general form

$$\frac{\partial u_i}{\partial t} = c_{i-1}\, u_{i-1} + c_i\, u_i + c_{i+1}\, u_{i+1}. \tag{4.12}$$

Because of consistency with the differential equation, it holds that $c_{i-1} + c_i + c_{i+1} = 0$, so we can rearrange Eq. 4.12 to

$$\frac{\partial u_i}{\partial t} = c_{i-1}\,(u_{i-1} - u_i) + c_{i+1}\,(u_{i+1} - u_i). \tag{4.13}$$

It is clear that a local extremum — for instance a local maximum with $u_i > u_{i-1}$ and $u_i > u_{i+1}$ — will be damped out when the coefficients c_{i-1} and c_{i+1} are both positive. A spatial discretization is called positive when the coefficients of the points in the stencil — except c_i — are positive. It can be seen from Eq. 4.6 that the upwind scheme is positive, such that spurious oscillations cannot be formed at discontinuities. Positive numerical schemes are related to *total variation diminishing* (TVD) schemes [90]. The positivity concept carries over automatically to multiple space dimensions, while the TVD concept is only well-defined in one space dimension.

The upwind scheme can be made more accurate by reducing the numerical diffusion such that $\eta_{num} \approx (\Delta x)^2$, which would mean that the scheme is second order accurate. This can be done through the technique of *linear reconstruction*, as explained in Sec. 4.2. However, *Godunov's theorem* states that a second order scheme which is linear, cannot be positive. To conserve the property of positivity, one has to consider *nonlinear schemes* which are second order away from discontinuities, but which through the action of a *nonlinear limiter* automatically add more dissipation $\eta_{num} \approx \Delta x$ at discontinuities.

The upwind scheme can be generalized to *nonlinear* scalar conservation laws by defining the numerical flux function in analogy with Eq. 4.8 as

$$f^{n*}_{i+1/2} = \frac{f(u^n_{i+1}) + f(u^n_i)}{2} - \frac{1}{2}|f'^{n*}_{i+1/2}|(u^n_{i+1} - u^n_i), \tag{4.14}$$

where $f'^{n*}_{i+1/2}$ is some approximation for the derivative of the flux at the location $x = x_{i+1/2}$. For instance, one can make the simple choice $f'^{n*}_{i+1/2} = f'((u^n_i + u^n_{i+1})/2)$, but other choices with advantageous properties are possible.

4.1.2 Systems of conservation laws

A hyperbolic *system* of conservation laws

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0, \tag{4.15}$$

can be discretized in a way analogous to the scalar case. In case of a *linear* system with $\mathbf{F}(\mathbf{U}) = \mathsf{A} \cdot \mathbf{U}$ and $\mathsf{A} = \mathsf{R}\,\Lambda\,\mathsf{L}$ the eigenvector decomposition of A, we can decouple the equations and apply the upwind discretization technique to every decoupled equation separately, which leads to the numerical flux function

$$\mathbf{F}^{n*}_{i+1/2} = \frac{\mathsf{A}\,\mathbf{U}^n_{i+1} + \mathsf{A}\,\mathbf{U}^n_i}{2} - \frac{1}{2}|\mathsf{A}|(\mathbf{U}^n_{i+1} - \mathbf{U}^n_i), \tag{4.16}$$

where $|\mathsf{A}| = \mathsf{R}\,|\Lambda|\,\mathsf{L}$. This adds the numerical dissipation $\eta^{(k)}_{num} = |\lambda^{(k)}|\,\Delta x/2$ to (decoupled) equation k.

For a *nonlinear* system, we have to specify an approximation for $\mathsf{A}^n_{i+1/2} = \mathbf{F}'^n_{i+1/2}$, which is the Jacobian $\partial\mathbf{F}/\partial\mathbf{U}$ of the flux at the location $x = x_{i+1/2}$. For instance, one can again make the simple choice $\mathsf{A}^{n*}_{i+1/2} = \mathsf{A}((\mathbf{U}^n_i + \mathbf{U}^n_{i+1})/2)$. The numerical flux function then reads

$$\mathbf{F}^{n*}_{i+1/2} = \frac{\mathbf{F}(\mathbf{U}^n_{i+1}) + \mathbf{F}(\mathbf{U}^n_i)}{2} - \frac{1}{2}|\mathsf{A}^{n*}_{i+1/2}|(\mathbf{U}^n_{i+1} - \mathbf{U}^n_i), \tag{4.17}$$

with $|\mathsf{A}^{n*}_{i+1/2}| = \mathsf{R}_{i+1/2}\,|\Lambda|_{i+1/2}\,\mathsf{L}_{i+1/2}$. This expression is closely related to the Roe flux function [127]. The CFL time step limitation now becomes

$$\Delta t < \frac{\Delta x}{\max_{k,i}(|\lambda^{(k)}_{i+1/2}|)}, \tag{4.18}$$

with $\max_{k,i}(|\lambda^{(k)}_{i+1/2}|)$ the largest wave speed in the system. This suggests, however, that we could simplify the numerical scheme considerably by applying the largest numerical diffusion $\eta_{num,i+1/2} = \max_k(|\lambda^{(k)}_{i+1/2}|)\,\Delta x/2$ uniformly to *all* the equations at location $x = x_{i+1/2}$. This still guarantees numerical stability and positivity of the discretization, although it adds some excessive dissipation to some of the characteristic waves. The numerical flux function simplifies to

$$\mathbf{F}^{n*}_{i+1/2} = \frac{\mathbf{F}(\mathbf{U}^n_{i+1}) + \mathbf{F}(\mathbf{U}^n_i)}{2} - \frac{1}{2}\max_k(|\lambda^{(k)}_{i+1/2}|)(\mathbf{U}^n_{i+1} - \mathbf{U}^n_i). \tag{4.19}$$

The resulting numerical scheme is much simpler because no eigenvector decomposition is necessary, and it also turns out to be much more robust. It is called the *(local) Lax-Friedrichs scheme*.

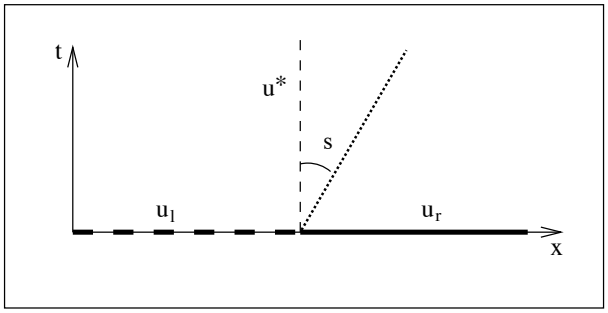

Figure 4.2: *Riemann problem for a scalar equation. In this example the solution is a shock which connects u_l with u_r and propagates with speed s.*

We can now make the link to *finite volume* discretizations [90]. To this end we first have to introduce the concept of a *Riemann problem* [90]. For simplicity we discuss the scalar case. As is shown in Fig. 4.2, a Riemann problem is an initial value problem for which the initial state is a discontinuous jump between two constant states. We have shown in Sec. 3.3.5 that the exact solution of a scalar Riemann problem is a shock (propagating with shock speed s, see Fig. 4.2) or a rarefaction — or possibly a compound shock for non-convex flux functions. For a $k \times k$ system, the solution to the Riemann problem generally consists of k shocks and/or rarefactions which propagate at different speeds. An important property of Riemann problem solutions is that they are *self-similar* in the variable x/t, such that the state $u^*(t)$ at the x-location of the initial discontinuity between u_l and u_r, is *constant in time*. In a linear system, there are only shocks and no rarefactions, and below we consider *linearized* Riemann problems.

We divide the x and t axes in *finite volume* cells with constant lengths Δx and Δt. We integrate Eq. 4.1 over a finite volume cell (Fig. 4.3) and obtain

$$(\overline{u}_i^{n+1} - \overline{u}_i^n)\,\Delta x + (\overline{\overline{f}}(u_{i+1/2}^{n*}) - \overline{\overline{f}}(u_{i-1/2}^{n*}))\,\Delta t = 0, \tag{4.20}$$

where the overbar denotes a spatial average, the double overbar is a temporal average and $u_{i+1/2}^{n*}$ is the exact solution at $x = x_{i+1/2}$. The cell average $\overline{u}_i^n$ is stored in the center of cell i. Eq. 4.20 is exact and can be interpreted as an evolution equation for the cell averages, which change when there is a net flux through the cell interfaces. We can

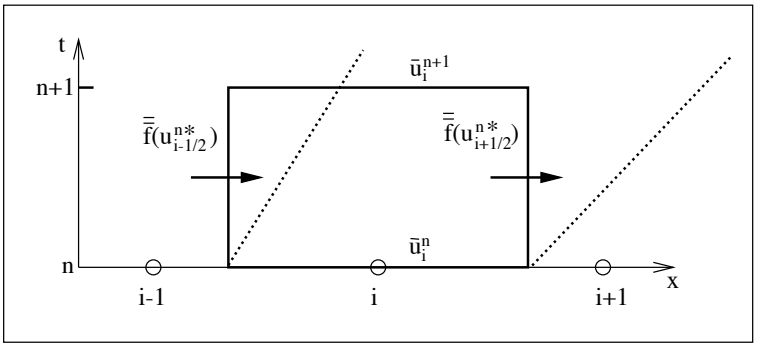

Figure 4.3: *Finite volume integration.*

now approximate $u(x,t)$ by evolving the cell averages in time using Eq. 4.20 with a certain approximation $f_{i+1/2}^{n*}$ for the flux $\bar{\bar{f}}(u_{i+1/2}^{n*})$. $f_{i+1/2}^{n*}$ is again called the *numerical flux function*. At a given time $t = t_n$, we have thus to consider the evolution of a piecewise-constant profile, which in fact is equivalent to the solution of many Riemann problems at every cell interface. The solutions u^* of these Riemann problems at the interfaces are constant in time, so we can drop the double overbar in Eq. 4.20. We can solve these Riemann problems approximately, and can for instance choose the linearized approximate Riemann solver $f_{i+1/2}^{n*}$ defined by

$$f_{i+1/2}^{n*} = \frac{f(u_{i+1}^n) + f(u_i^n)}{2} - \frac{1}{2}|f'^{n*}_{i+1/2}|(u_{i+1}^n - u_i^n), \tag{4.21}$$

where we have been inspired by the flux function used in our upwind finite difference scheme Eq. 4.14.

The reader can note the equivalence between conservative finite difference schemes (*point values* are updated using finite difference approximations of derivatives) and finite volume schemes (*cell averages* are updated using differences of fluxes through cell interfaces). For the first order schemes considered in this Section the two approaches are completely equivalent. For instance, the CFL condition can be interpreted as the condition that waves from neighboring Riemann problems do not interfere. *Upwind discretization* is equivalent to solving *Riemann problems* to determine the numerical finite volume flux. Higher order discretization of derivatives, however, is not automatically equivalent to higher order discretization of fluxes through interfaces.

4.2 Spatial discretization: finite volume schemes

In this Section we describe the spatial discretization approach adopted in our code, which leads to schemes which are first and second order accurate in space (for smooth flow). Here we only describe discretization of the *ideal* equations. It is not too difficult to discretize the dissipative terms as well [106], and the 2D numerical code used for the simulations presented in this dissertation actually is capable of performing dissipative simulations [155], but this aspect of the code is not described here. For simplicity, we start with an equi-distant spatial discretization in 1D (grid size Δx), and subsequently discuss 2D, 3D and 2D axi-symmetrical discretizations. The ideal MHD equations 3.12 can be written in the following abstract conservation law form

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{U}) = \mathbf{S}^{Pow}, \tag{4.22}$$

with $\mathbf{U}$ the vector of state variables which are conserved quantities, $\vec{\mathbf{F}}$ the flux vector, and $\mathbf{S}^{Pow}$ the Powell source term (which equals zero on the PDE level because it is proportional to $\nabla \cdot \vec{B}$).

4.2.1 One–dimensional space

In every cell (i) we define the cell average of the state variable $\mathbf{U}$ as

$$\overline{\mathbf{U}}_i = \left(\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t)\, dx \right) / \Delta x. \tag{4.23}$$

This cell average is stored in the center of cell (i) — leading to a *cell-centered* approach. We obtain the following time evolution equation for this average after integration of Eq. 4.22 in space over the *finite volume* cell with label i

$$\frac{\partial \overline{\mathbf{U}}_i}{\partial t} + 1/\Delta x \, (\mathbf{F}^*_{i+1/2} - \mathbf{F}^*_{i-1/2}) = 0. \tag{4.24}$$

In 1D the Powell source term is also identically zero numerically, because B_x is a constant in 1D space. $\mathbf{F}^*_{i+1/2}$ is a numerical approximation for the flux through the interface between cell (i) and cell $(i+1)$ (the *numerical flux function*).

There are many possible choices for the form of the numerical flux function, corresponding to the choice of an (approximate) Riemann solver. In Sec. 4.2.5 we discuss some possible choices and describe the numerical flux function used for the simulations presented in this paper.

Many numerical flux functions, including the Roe and Lax-Friedrichs flux functions, can be cast in the following form

$$\mathbf{F}^*(\mathbf{U}_l, \mathbf{U}_r) = \frac{\mathbf{F}(\mathbf{U}_l) + \mathbf{F}(\mathbf{U}_r)}{2} + D(\mathbf{U}_l, \mathbf{U}_r), \tag{4.25}$$

with $\mathbf{F}$ being the 1D MHD flux function of Eq. 3.76, and $\mathbf{U}_l$ and $\mathbf{U}_r$ the state variables to the left and to the right of the interface. The third term $D(\mathbf{U}_l, \mathbf{U}_r)$ is in general proportional to $\mathbf{U}_r - \mathbf{U}_l$.

Let us focus on the flux through the interface between cells i and $i+1$. A first order accurate spatial discretization is obtained if we take the left and the right state used to calculate the numerical flux, to be the cell-averages to the left and the right of the interface, viz. $\mathbf{U}_l = \overline{\mathbf{U}}_i$ and $\mathbf{U}_r = \overline{\mathbf{U}}_{i+1}$. In this picture, the solution is thought to be *piecewise-constant* in every cell. The first two terms in the right-hand side (RHS) of Eq. 4.25 correspond to a second-order accurate central space discretization of the flux terms without dissipation, and would lead to a numerically unstable scheme without the addition of dissipative terms. The third term $D(\mathbf{U}_l, \mathbf{U}_r)$ precisely adds this (numerical) dissipation with a dissipation coefficient which vanishes in first order in Δx. There are many choices for the exact form of this dissipative term, corresponding to the choice of an (approximate) Riemann solver. This is discussed in Sec. 4.2.5.

The scheme can be enhanced to second order spatial accuracy by considering *piecewise-linear* variation in a cell. For instance, $\mathbf{U}_l$ can be calculated — or *reconstructed* — as $\mathbf{U}_l = \overline{\mathbf{U}}_i + 1/2\,(\overline{\mathbf{U}}_i - \overline{\mathbf{U}}_{i-1})$, or possibly also as $\mathbf{U}_l = \overline{\mathbf{U}}_i + 1/2\,(\overline{\mathbf{U}}_{i+1} - \overline{\mathbf{U}}_i)$ or any convex combination of these two. This kind of reconstruction would lead to second order accurate schemes — the dissipation coefficient of the term $D(\mathbf{U}_l, \mathbf{U}_r)$ would vanish in second order —, but the solution near discontinuities would contain spurious oscillations. To discard these spurious oscillations, a nonlinear *slope limiter* L can be used to determine the slope of the linear reconstruction, with

$$\mathbf{U}_l = \overline{\mathbf{U}}_i + 1/2\, L(\overline{\mathbf{U}}_i - \overline{\mathbf{U}}_{i-1}, \overline{\mathbf{U}}_{i+1} - \overline{\mathbf{U}}_i). \tag{4.26}$$

Again, there are many possible choices for such a limiter [90], and we use the *minmod* limiter for our simulations. The component-wise action of the minmod function is given by

$$l(x,y) = \mathrm{sign}(x)\ \ \max(\,0\,,\, \min(\mathrm{abs}(x), \mathrm{sign}(x)\, y)\,). \tag{4.27}$$

It can easily be verified that this function selects the least steep slope for the reconstruction if the sign of the slopes is the same, and vanishes (leading to piecewise constant reconstruction and first order accuracy) when the signs of the two slopes are different, which typically is the case

at discontinuities. It can be proven rigorously that this limiter function (and other similar limiter functions) effectively discards spurious oscillations [90] because it ensures positivity of the scheme. In practice we do the second order reconstruction using the primitive variables **W**, and not the conservative variables **U**. Experience shows that this improves the robustness and steady state convergence properties of the scheme.

4.2.2 Two–dimensional space

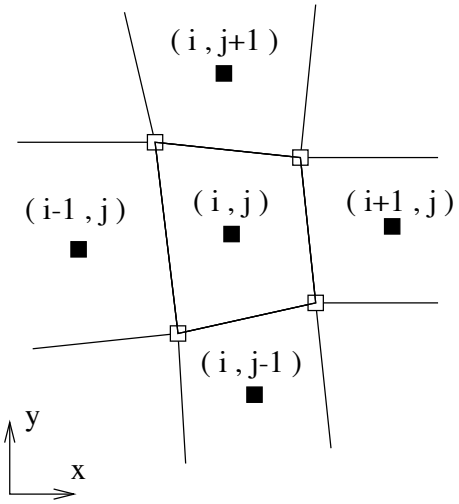

Figure 4.4: *Finite volume grid cell with label* (i, j) *and its nearest neighbor cells. The variables stored in the centers of a cell represent the averages of the state variables over the cell.*

Now we describe how these concepts can be extended to spatial discretization on a 2D structured body-fitted grid. We divide the computational domain in a logically rectangular structured grid of quadrilaterals (Fig. 4.4). The solution of the flow is sought in the *physical* cells, with indices in the computational domain ranging from 1 to n_i for index i, and from 1 to n_j for index j. This physical domain is surrounded by two layers of *ghost cells*, which allow for a simple implementation of boundary conditions (see Sec. 4.2.6). The cell interfaces are not constrained to be parallel to a Cartesian axis, which for instance allows to fit the grid to a curved rigid body. We integrate Eq. 4.22 formally over the *finite volume* cell with label (i, j), and obtain the following discretized

equation

$$\frac{\partial \overline{\mathbf{U}}_{i,j}}{\partial t} + 1/\Omega_{i,j} \sum_{k=1}^{4} \vec{\mathbf{F}}_k^* \cdot \vec{n}_k \, \Delta l_k = \overline{\mathbf{S}}_{i,j}^{Pow}, \tag{4.28}$$

for the time evolution of the average of the state variable over cell (i,j)

$$\overline{\mathbf{U}}_{i,j} = \left(\int\int \mathbf{U}(x,y,t) \, dx \, dy \right) / \Omega_{i,j}, \tag{4.29}$$

which is stored in the center of cell (i,j). The coordinates of the cell centers are calculated as the arithmetic average of the corner point coordinates. Here $\Omega_{i,j}$ is the surface of cell (i,j), and the summation in Eq. 4.28 extends over the four sides or interfaces of cell (i,j). $\vec{\mathbf{F}}_k^*$ is a numerical approximation for the flux vector through the interface k, Δl_k is the length of interface k, and $\vec{n}_k$ is the outside unit vector normal to side k. $\overline{\mathbf{S}}_{i,j}^{Pow}$ is the cell average of the Powell source term in cell (i,j).

For each interface, we can apply the above described 1D technique to calculate the normal numerical flux $\mathbf{F}^* = \vec{\mathbf{F}}^* \cdot \vec{n}$. The 1D formulas remain valid in a coordinate system aligned with the interface. For instance, to calculate the flux between cell (i,j) and cell $(i+1,j)$, a first order accurate scheme is obtained by taking $\mathbf{U}_l = \overline{\mathbf{U}}_{i,j}$ and $\mathbf{U}_r = \overline{\mathbf{U}}_{i+1,j}$. Appropriate rotations of vector components have to be carried out then to calculate $\mathbf{F}^*$ using Eq. 4.25. A second order accurate scheme results from taking $\mathbf{U}_l = \overline{\mathbf{U}}_{i,j} + 1/2\, L(\overline{\mathbf{U}}_{i,j} - \overline{\mathbf{U}}_{i-1,j}, \overline{\mathbf{U}}_{i+1,j} - \overline{\mathbf{U}}_{i,j})$. In practice we do the second order reconstruction using the primitive variables $\mathbf{W}$, and with vector components in the coordinate system aligned with the interface. This alignment reduces errors of the scheme at boundaries. This simple dimension-by-dimension approach turns out to work well, and it can be proven that the scheme remains second order accurate if the grid is not too much distorted [164]. In the next Chapter we present grid convergence results which confirm this. A more sophisticated approach would be to do reconstruction using estimates of gradients based on 2D interpolation [7].

We use the following discretization for the source term $\overline{\mathbf{S}}_{i,j}^{Pow}$ in cell (i,j). $\nabla \cdot \vec{B}$ is discretized as

$$(\nabla \cdot \vec{B})_{i,j} = 1/\Omega_{i,j} \sum_{k=1}^{4} \vec{B}_k \cdot \vec{n}_k \, \Delta l_k, \tag{4.30}$$

with

$$\vec{B}_k = (\vec{B}_l + \vec{B}_r)/2 \tag{4.31}$$

the average of the (reconstructed) magnetic fields on the left and the right of the interface. This discretization of $\nabla \cdot \vec{B}$ is then multiplied

with the appropriate cell-averaged state variables (stored in the center of cell (i,j)) to obtain a discretization for the source term in Eq. 4.28. Experience has shown that it is important to use the *reconstructed* values of $\vec{B}$ in the discretization of $\nabla \cdot \vec{B}$ for the second order scheme, because this results in a more robust scheme.

4.2.3 Three–dimensional space

It is rather straightforward to extend the above described techniques to spatial discretization to a 3D structured body-fitted grid. We divide the computational domain in a logically rectangular structured grid of hexahedrals (cells with eight corner points and six boundary interfaces). We introduce a third index k ranging from 1 to n_k in the computational space. The physical domain is again surrounded by two layers of ghost cells.

We integrate Eq. 4.22 formally over the hexahedral *finite volume* cell with label (i,j,k), and obtain the following discretized equation

$$\frac{\partial \overline{\mathbf{U}}_{i,j,k}}{\partial t} + 1/\Omega_{i,j,k} \sum_{p=1}^{6} \vec{\mathbf{F}}_p^* \cdot \vec{n}_p \, \Delta A_p = \overline{\mathbf{S}}_{i,j,k}^{Pow}, \tag{4.32}$$

for the time evolution of the average of the state variable over cell (i,j,k)

$$\overline{\mathbf{U}}_{i,j,k} = \left(\iiint \mathbf{U}(x,y,z,t) \, dx \, dy \, dz \right) / \Omega_{i,j,k}, \tag{4.33}$$

which is stored in the center of cell (i,j,k). Here $\Omega_{i,j,k}$ is the volume of cell (i,j,k), and the summation in Eq. 4.32 extends over the six sides or interfaces of cell (i,j,k). $\vec{\mathbf{F}}_p^*$ is a numerical approximation for the flux vector through the interface p, ΔA_p is the surface of interface p, and $\vec{n}_p$ is the outside unit vector normal to side p. In general the four points defining a cell interface are *not* co-planar, so care has to be taken to define the cell volumes and the interface normals and surfaces in a consistent way. We have used a technique which subdivides the hexahedron in three five-sided pyramids pointing at the same vertex, as described in [106].

As in the 2D case, we can apply the above described 1D technique to calculate the normal numerical flux $\mathbf{F}^* = \vec{\mathbf{F}}^* \cdot \vec{n}$ for each interface. The 1D formulas remain valid in a coordinate system aligned with the interface. The source term is discretized in a way completely analogous to the 2D case.

4.2.4 Axial symmetry

The conservative form of the ideal MHD equations with a source term Eq. 3.12 can be written in cylindrical coordinates (x,r,θ), and by assuming

rotational or axial symmetry around the x-axis ($\partial/\partial\theta \equiv 0$), one obtains

$$\frac{\partial(r\mathbf{U})}{\partial t} + \frac{\partial(r\mathbf{F}_x(\mathbf{U}))}{\partial x} + \frac{\partial(r\mathbf{F}_r(\mathbf{U}))}{\partial r} = \mathbf{S}^{geo}(\mathbf{U}) + r\mathbf{S}^{Pow}. \tag{4.34}$$

Here $\mathbf{U} = (\rho, \rho v_x, \rho v_r, \rho v_\theta, B_x, B_r, B_\theta, e)$ is the vector of conservative variables in the cylindrical coordinate system. $\mathbf{F}_x(\mathbf{U})$ and $\mathbf{F}_r(\mathbf{U})$ are MHD flux functions (as in Eq. 3.76) in the x and the r directions, for instance

$$\mathbf{F}_x(\mathbf{U}) = \begin{bmatrix} \rho\, v_x \\ \rho\, v_x^2 + p + B^2/2 - B_x^2 \\ \rho\, v_x v_r - B_x B_r \\ \rho\, v_x v_\theta - B_x B_\theta \\ 0 \\ B_r v_x - B_x v_r \\ B_\theta v_x - B_x v_\theta \\ (e + p + B^2/2) v_x - B_x\, (\vec{v} \cdot \vec{B}) \end{bmatrix}. \tag{4.35}$$

$\mathbf{S}^{geo}(\mathbf{U})$ is a geometrical source term reading

$$\mathbf{S}^{geo}(\mathbf{U}) = \begin{bmatrix} 0 \\ 0 \\ (\rho v_\theta^2 - B_\theta^2) + (p + B^2/2) \\ -(\rho v_r v_\theta - B_r B_\theta) \\ 0 \\ 0 \\ v_r B_\theta - v_\theta B_r \\ 0 \end{bmatrix}, \tag{4.36}$$

and $r\mathbf{S}^{Pow}$ is the Powell source term given by

$$r\mathbf{S}^{Pow} = \mathbf{D}^{Pow}\, \mathbf{W}^{Pow}(\mathbf{U}), \tag{4.37}$$

with

$$\mathbf{D}^{Pow} = \left(\frac{\partial(rB_x)}{\partial x} + \frac{\partial(rB_r)}{\partial r} \right), \tag{4.38}$$

and

$$\mathbf{W}^{Pow}(\mathbf{U}) = \begin{bmatrix} 0 \\ B_x \\ B_r \\ B_\theta \\ v_x \\ v_r \\ v_\theta \\ \vec{B} \cdot \vec{v} \end{bmatrix}. \tag{4.39}$$

We have derived these expressions following the technique proposed in [13]. We discretize these equations on a 2D structured body-fitted grid.

We integrate Eq. 4.34 formally over the cell with label (i,j), and after some approximations regarding the averaging of state variables and geometrical quantities, we obtain the following discretized equation

$$\Omega_{i,j} r_{i,j}^{center} \frac{\partial \overline{\mathbf{U}}_{i,j}}{\partial t} + \sum_{k=1}^{4} r_k^{middle} \vec{\mathbf{F}}_k^* \cdot \vec{n}_k \Delta l_k = \Omega_{i,j} (\mathbf{S}^{geo}(\overline{\mathbf{U}}_{i,j}) + \overline{\mathbf{D}}_{i,j}^{Pow} \mathbf{W}^{Pow}(\overline{\mathbf{U}}_{i,j})), \tag{4.40}$$

for the time evolution of the average of the state variable over cell (i,j)

$$\overline{\mathbf{U}}_{i,j} = \left(\int \int \mathbf{U}(x,r,t)\, dx\, dr \right) / \Omega_{i,j}, \tag{4.41}$$

which is stored in the center of cell (i,j). Here $\Omega_{i,j}$ is the surface of cell (i,j). $\vec{\mathbf{F}}_k^*$ is a numerical approximation for the flux vector through the interface k, Δl_k is the length of interface k, and $\vec{n}_k$ is the outside unit vector normal to side k. $r_{i,j}^{center}$ is the distance between the center of cell (i,j) and the axis of rotational symmetry, and r_k^{middle} is the distance between the middle point of the interface segment with label k and the symmetry axis. The term $\overline{\mathbf{D}}_{i,j}^{Pow}$ is discretized as

$$\overline{\mathbf{D}}_{i,j}^{Pow} = 1/\Omega_{i,j} \sum_{k=1}^{4} r_k^{middle} \vec{B}_k \cdot \vec{n}_k \Delta l_k, \tag{4.42}$$

with

$$\vec{B}_k = (\vec{B}_l + \vec{B}_r)/2 \tag{4.43}$$

the average of the (reconstructed) magnetic fields on the left and the right of the interface. The same numerical flux functions $\vec{\mathbf{F}}^*$ and the same second order reconstruction can be used as for the case of the 1D, 2D and 3D schemes. Actually, the 2D code based on the scheme described in Sec. 4.2.2 for planar symmetry ($\partial/\partial z \equiv 0$) can be used for axi-symmetrical simulations by a simple re-definition of the geometrical quantities $\Omega' = r^{center}\Omega$ and $\Delta l' = r^{middle}\Delta l$, and by adding the geometrical source term.

4.2.5 Numerical flux functions

Throughout the years, many interesting MHD numerical flux functions have been proposed that can be used in the type of finite volume discretization described above, most of them based on some kind of approximate Riemann solver at the interface between two cells [12, 183, 23, 24, 128, 129, 1, 118, 127, 162, 6, 7, 99, 36, 26, 111]. Many of those flux functions are designed to produce as sharp as possible shock transitions and

tangential discontinuities. The Roe scheme for example, which has been very popular for hydrodynamic applications, has been extended to MHD [127, 7]. This scheme tries to minimize the numerical dissipation by decomposing the difference $\mathbf{U}_r - \mathbf{U}_l$ present in $D(\mathbf{U}_l, \mathbf{U}_r)$ in the space of the eigenvectors of the Jacobian and by applying the minimum amount of numerical dissipation to every characteristic wave separately. There remain, however, several serious problems of local numerical instability with this scheme, including the carbuncle phenomenon, as for example described by Quirk [121]. Probably because of these problems the Roe scheme is not so much used in MHD simulations these days [100, 150, 82].

Several approaches have been proposed to remedy these problems. First, different types of nonlinear flux functions are being investigated, e. g. Linde's HLLE-based solver for MHD [99], which is based on some other form for the numerical dissipation, or solvers derived from kinetic descriptions [100, 99]. These new flux functions seem to remedy some of the problems associated with the Roe solver, but more investigation is necessary to see if they can solve all the stability problems. Second, it is sometimes argued that much of the problems with finite volume schemes on structured grids are inherent to the dimension-by-dimension approach, and that many of the pathological instabilities could be removed by considering truly multi-dimensional schemes on unstructured grids [167, 22]. Third, the failure of the Roe scheme can probably be related to the fact that it is not *entropy stable* [7]. New entropy stable schemes formulated in symmetrized *entropy variables* are being developed [7] and it can be expected that they will lead to more stable numerical schemes. Also in this area a lot of research is still going on.

This short discussion indicates that there are certainly many unsolved issues regarding the choice of numerical flux functions and numerical schemes in general.

The Lax-Friedrichs flux function

The (local) Lax-Friedrichs (LF) flux function [90, 162, 6] is given by

$$\mathbf{F}^*(\mathbf{U}_l, \mathbf{U}_r) = \frac{\mathbf{F}(\mathbf{U}_l) + \mathbf{F}(\mathbf{U}_r)}{2} - (|v_n^*| + c_n^{f*}) \frac{\mathbf{U}_r - \mathbf{U}_l}{2}, \qquad (4.44)$$

with $|v_n^*| + c_n^{f*}$ the largest wave speed in the direction normal to the interface, determined from the arithmetic average $(\mathbf{U}_l + \mathbf{U}_r)/2$ on the interface. This is sometimes also called the Rusanov flux function. The Lax-Friedrichs flux function applies to all characteristic waves the same numerical dissipation, determined by the maximum wave speed, which makes it more dissipative than the Roe scheme for instance, but much more robust and less prone to local numerical instabilities. The Lax-Friedrichs flux function is probably the most robust numerical flux func-

tion. Because it is simple and robust, its use in applications is often advocated [162, 6, 150, 82]. Stationary shock profiles are actually captured surprisingly well with this Lax-Friedrichs scheme [6, 162], but tangential discontinuities are smeared out. We show in the next Chapter that we obtain satisfactory grid convergence results using this numerical flux function. For most of the simulations to be described in this dissertation, we use the LF flux function.

The Roe flux function

The Roe scheme has been extended to MHD, and the details of the eigenvector decomposition can be found in [127, 7]. We have certainly tried to use the Roe scheme for our simulations of bow shock flows, but the Roe scheme has turned out to be not reliable enough. For instance, the carbuncle phenomenon [121] was often encountered at the nose of bow shocks. We illustrate this in Chap. 9.

The MHD HLLE flux function

Linde's extension of the HLLE flux function to MHD [99] seems to be more reliable than the Roe scheme. The carbuncle phenomenon did not seem to show up in our 2D simulations when we used this flux function, but we observed other instabilities which prevented convergence to a steady state, as is illustrated in Chap. 9. In 2D we were able to choose the resolution of the grid high enough to get good results with the LF scheme. In 3D, however, we sometimes had to use Linde's HLLE scheme to obtain flow solutions with reasonable detail.

4.2.6 Boundary conditions

Most of the simulations presented in this dissertation were performed using boundary conditions implemented with two layers of *ghost cells* (e.g. [91]). This allows us to use the same algorithm to update all the cells in the physical simulation domain, also the ones close to the boundary. Superfast inflows and outflows and perfectly conducting walls can easily be modeled by this technique. For subfast inflows and outflows, however, we have used boundary conditions based on direct specification of the numerical flux on the boundary. We discuss the boundary conditions for the case of two spatial dimensions. The 3D case is completely analogous.

Boundary conditions using ghost cells

The following three types of boundary conditions can easily be specified using the ghost cell approach. Let us focus on a boundary to the right of cell ($i = n_i, j = j^*$), which is the last *physical* cell in the i direction

with the second coordinate j equal to a certain j^* in the computational space.

First, at perfectly conducting walls, the magnetic and the velocity fields have to be tangent to the wall. This is implemented by copying the cell-averaged state of (n_i,j^*) into the first ghost cell (n_i+1,j^*), except for the components of the magnetic and velocity fields perpendicular to the interface between cells (n_i,j^*) and (n_i+1,j^*), which are copied with a change in sign. For the second order scheme, the cell-averaged state of (n_i-1,j^*) is copied into the second ghost cell (n_i+2,j^*), except again for the components of the magnetic and velocity fields perpendicular to the interface between cells (n_i,j^*) and (n_i+1,j^*), which are copied with a change in sign. This procedure guarantees that the value of $(\mathbf{U}_l + \mathbf{U}_r)/2$ for the normal components of the fields vanishes exactly on that interface, also for the second order scheme with minmod reconstruction in a coordinate system aligned with the interface.

Second, at free outflows, where the normal outward plasma velocity is larger than the normal fast MHD wave speed and all the characteristic information thus propagates outward of the physical domain, the state variables of the last two physical cells are used to extrapolate linearly into the two layers of ghost cells, for instance $\overline{\mathbf{U}}_{n_i+2,j^*} = \overline{\mathbf{U}}_{n_i,j^*} + 2\,(\overline{\mathbf{U}}_{n_i,j^*} - \overline{\mathbf{U}}_{n_i-1,j^*})$.

Third, at free inflows, where the normal inward plasma velocity is larger than the normal fast MHD wave speed and all the characteristic information thus propagates into the physical domain, we impose the value of $\mathbf{U}_{in} = (\mathbf{U}_l + \mathbf{U}_r)/2$ on the boundary interface, and use this and the state of the last physical cell to calculate the value in the first ghost cell. In our example, $\overline{\mathbf{U}}_{n_i+1,j^*} = 2\,\mathbf{U}_{in} - \overline{\mathbf{U}}_{n_i,j^*}$. The value in the second ghost cell $\overline{\mathbf{U}}_{n_i+1,j^*}$ follows from linear extrapolation of $\overline{\mathbf{U}}_{n_i,j^*}$ and $\overline{\mathbf{U}}_{n_i+1,j^*}$. For the free inflows and outflows, an even simpler but often satisfactory approach is to use constant extrapolation of the solution.

Although these boundary conditions are based on a simple dimension-by-dimension extrapolation, they turn out to work well, as is proven in the next Chapter.

Flux boundary conditions

Characteristic boundary conditions [7] are implemented by directly specifying the flux as follows. At the boundary interface to the right of cell ($i = n_i, j = j^*$), one determines the state $\mathbf{U}^*$ which is the approximate solution of the Riemann problem with left state the interpolated state $\overline{\mathbf{U}}_l = \overline{\mathbf{U}}_{n_i,j^*} + 1/2\,(\overline{\mathbf{U}}_{n_i,j^*} - \overline{\mathbf{U}}_{n_i-1,j^*})$, and right state a specified *free flow state* $\mathbf{U}_r = \mathbf{U}_\infty$. In bow shock flows, this free flow state is typically the state far upstream of the object. $\mathbf{U}^*$ can for instance be determined using the Roe linearized Riemann decomposition. The numerical flux is

then given by $\mathbf{F}^* = \mathbf{F}(\mathbf{U}^*)$. For superfast inflow and outflow, this results in a boundary condition which is very close to the above described ghost cell approach. For subfast flow, however, some characteristic information flows into the physical domain, and other information flows out. It is generally hard to translate this to specifying some of the primitive variables externally, and extrapolating the others from the physical domain, because there is a complicated nonlinear and state-dependent relationship between the characteristic variables carrying the wave information and the primitive variables. The solution of the Riemann problem *automatically* selects the necessary amount of information from the state in the physical domain and the externally specified free flow state [7].

4.3 Temporal discretization: explicit Runge-Kutta schemes

In this Section we discuss the time integration strategy used in our simulation code. We situate the discussion in the context of the 2D code, but the 1D and 3D cases are completely analogous. An equation for the time evolution of the cell average in cell (i, j) can be written in the following general form

$$\frac{\partial \overline{\mathbf{U}}_{i,j}}{\partial t} = \mathbf{R}_{i,j}^{(1),(2)}(\overline{\mathbf{U}}), \tag{4.45}$$

with $\mathbf{R}_{i,j}^{(1),(2)}$ a first or second order accurate discretization of the *residual* in cell (i, j), containing the spatial discretization of the divergence term in Eq. 4.22, and the discretization of the source term. An explicit first-order accurate time integration is obtained by simply approximating the time derivative by a forward finite difference,

$$\overline{\mathbf{U}}_{i,j}^{t+\Delta t} = \overline{\mathbf{U}}_{i,j}^{t} + \mathbf{R}_{i,j}^{(1)}(\overline{\mathbf{U}}^t)\,\Delta t. \tag{4.46}$$

An explicit second-order accurate time integration is obtained by applying the following two-stage Runge-Kutta integration,

$$\begin{aligned} \overline{\mathbf{U}}_{i,j}^{*} &= \overline{\mathbf{U}}_{i,j}^{t} + \mathbf{R}_{i,j}^{(2)}(\overline{\mathbf{U}}^t)\,\Delta t/2 \\ \overline{\mathbf{U}}_{i,j}^{t+\Delta t} &= \overline{\mathbf{U}}_{i,j}^{t} + \mathbf{R}_{i,j}^{(2)}(\overline{\mathbf{U}}^*)\,\Delta t. \end{aligned} \tag{4.47}$$

The time step Δt is derived from the following CFL-like time step limitation [106]

$$\Delta t = c_{CFL}\ \min_{i,j}\left[\frac{\Omega_{i,j}}{\sum_{k=1}^{4} \max(0, (\vec{v}_{i,j}\cdot\vec{n}_k + c^f_{k,i,j}))\,\Delta l_k}\right], \tag{4.48}$$

with $c^f_{k,i,j}$ the fast MHD wave speed in the direction of $\vec{n}_k$ calculated with the cell-averaged state values stored in cell (i,j). The constant c_{CFL} has to be chosen smaller than one for the first order scheme, and may be chosen slightly larger than one for the second order scheme. We generally use $c_{CFL} = 0.8$ for first order calculations, and $c_{CFL} = 1.2$ for second order simulations.

We use these general time-accurate integration methods to calculate the stationary flow solutions to be described further on in this dissertation. In general, we start from a uniform initial flow condition and we evolve the flow in time until vanishing of the residuals $\mathbf{R}$ shows that a steady state has been reached. We use the following quantity based on the density residual to measure the convergence to a steady state at iteration m

$$\overline{R}(m) = \log\left(c_{norm}\sqrt{\frac{\sum_{i,j}(R^{\rho}_{i,j}(m))^2}{n_i n_j}}\right) \tag{4.49}$$

with n_i and n_j the number of cells in the i and the j direction, and the normalization constant c_{norm} chosen in such a way that $\overline{R}(0) = 0$. We routinely achieve convergence of 15 orders of magnitude, which means that the residuals are driven to machine zero. Throughout this dissertation we mean the base 10 logarithm when we use 'log' in convergence measures and on plots.

It would be possible to obtain convergence to a steady state more efficiently, and a whole scala of convergence acceleration methods of varying complexity could be tried out, ranging from simple local time stepping over implicit residual smoothing and multigrid, to fully implicit time integration [106, 161, 83, 22]. We have, however, not attempted any convergence acceleration technique for the simulations presented in this dissertation.

4.4 Enforcing the $\nabla \cdot \vec{B}$ constraint in MHD

The numerical enforcement of the $\nabla \cdot \vec{B}$ constraint is an important and much debated problem for numerical MHD codes. The $\nabla \cdot \vec{B} = 0$ condition is an initial condition, which is exactly preserved in time by the partial differential equations (PDEs) of MHD, but which is not always exactly preserved after discretization of the equations. Various strategies have been proposed to deal with the $\nabla \cdot \vec{B}$ constraint. We do not intend to give a full discussion of this subtle subject here, but we find it useful to give a brief overview of the various approaches to this problem.

The equations can be formulated in terms of a vector potential, which automatically assures divergence free magnetic fields. This approach is

not always very practical and leads to difficulties near sharp gradients due to the presence of second order derivatives of the vector potential in the equations (see e.g. [33, 158]). Staggered grid approaches store different state variables on different positions of the grid, in such a way that the discrete time evolution of the magnetic field automatically conserves the divergence free condition (e.g. [33, 28, 158, 26]). They require, however, extra interpolations to be performed, and the interpolated magnetic field is not necessarily divergence free. Projection scheme approaches solve an elliptic equation in every time step and add a computed correction to the magnetic fields in order to achieve divergence free fields [10, 183, 129, 162]. Solution of the elliptic equation may take considerable computing time, and spurious oscillations can be generated, also in supersonic regions that should not be affected because they can theoretically not be reached by wave perturbations in the hyperbolic system.

Recently a new approach has been presented by Powell [118]. He proposes to add a source term proportional to $\nabla \cdot \vec{B}$ to the conservative form of the MHD equations (Eq. 3.12). Discretization of this Galilean invariant symmetrizable form of the equations with a source term [52, 118, 7] (see Sec. 4.2.2) leads to a stable numerical scheme. The $\nabla \cdot \vec{B}$ constraint is not enforced strongly and $\nabla \cdot \vec{B}$ can sometimes be substantially different from zero, but because of the presence of the source terms the dynamical effect of the $\nabla \cdot \vec{B}$ errors is largely neutralized and $\nabla \cdot \vec{B}$ errors can be shown to be advected away with the plasma flow. Although this approach seems to work well [118, 162, 100, 99, 124, 149, 151, 7, 22] and has several conceptual advantages over other techniques because of its simplicity and consistency with the hyperbolic nature of the MHD equations, not much can be found in the literature about rigorous validation of this approach, and consequently this approach is still heavily disputed in discussions often dominated by dogmatic arguments.

In this dissertation we generally employ the Powell source term technique to control $\nabla \cdot \vec{B}$. In the Sections above, we have described how we discretize the Powell source term. We have found that this technique leads to consistent solutions, and in the next Chapter we prove by formal grid convergence studies of magnetic flux conservation and other flow quantities, that this approach is valid, at least for the class of stationary flow problems that we consider. It is sometimes argued that for some problems, for which the conservation of magnetic flux up to very high accuracy is crucial, the source term technique could turn out to be insufficient. We are not aware of any clear examples which would confirm this, however.

For some flow problems, we have explicitly compared the Powell source term technique with a projection scheme technique. In the projection scheme approach, one calculates a correction $\vec{B}_{corr} = -\nabla\Phi$ to the magnetic field after every time step, such that $\nabla \cdot (\vec{B} + \vec{B}_{corr}) = 0$

or $\nabla \cdot (\nabla \Phi) = \nabla \cdot \vec{B}$. Discretization of this Poisson equation leads to a large linear system, and care has to be taken to formulate the boundary conditions of this Poisson equation consistently with the boundary conditions of the hyperbolic MHD system. We have discretized the Poisson equation as proposed in [129, 161]. As is shown in Chap. 9, the Powell approach and the projection approach produce very similar results. Only near a stagnation point have we found the Powell technique to be less accurate than projection. The Powell technique is much faster and deals with the $\nabla \cdot \vec{B}$ constraint in a more consistent hyperbolic way, whereas the projection technique introduces contamination of upstream flow regions, which in the case of our projection scheme implies upstream oscillations due to *mode decoupling*. The projection scheme technique is not easily parallelizable, because it involves the global solution of a linear system. For all these reasons we prefer to use the Powell technique for most of our simulations.

4.5 Implementation on massively parallel computers

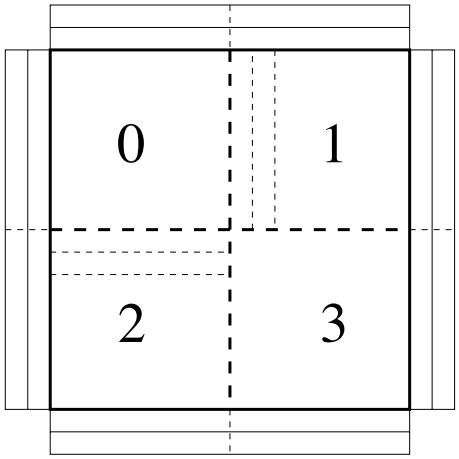

Figure 4.5: *Parallelization strategy to solve a flow problem in a given physical domain (thick solid) on four processors by dividing the domain in four sub-domains (thick dashed). The thin lines represent ghost cell layers.*

The algorithm described above has been implemented to run on *massively parallel* computers with *distributed memory*. Such computers con-

sist of several processors connected by a fast network, and every processor has its own memory which can not directly be accessed by the other processors. The processors can communicate by sending *messages*. It is relatively straightforward to implement our numerical scheme with explicit time integration on a parallel computer, for instance using the Message Passing Interface (MPI) communication library [60]. Indeed, the algorithm is *inherently parallel* because for the update of a given cell only the values stored in its immediate neighbor cells are needed.

We illustrate our approach for the 2D case in Fig. 4.5. Suppose we want to calculate a flow problem in a given physical domain (thick solid) on four processors. We cover the physical domain by a logically rectangular structured grid, and add two layers of ghost cells (thin solid). We divide the domain in four equally-sized sub-domains (thick dashed) and associate one processor to every sub-domain. We store the flow variables of all the cells in sub-domain 0 in the memory of processor 0, and add two layers of ghost cells (thin dashed). Only the ghost cells for processor 0 are shown on the Figure. In every time step, the internal cells of sub-domain 0 can be updated by processor 0 with information available in the memory of processor 0. For the cells close to the boundary (thick dashed), we use the information in the ghost cells. After every time step, the ghost cells which correspond to physical boundaries (thin solid), are updated using the physical boundary conditions. The ghost cells corresponding to sub-domain boundaries, however, are updated by exchanging messages between neighboring processors. Every processor thus executes the same algorithm, which is very similar to the serial algorithm, except for the update of the ghost cells. Only the time step Δt has to be communicated globally in every iteration.

For an inherently parallel algorithm it is expected that a given problem executes two time faster on $2n$ processors than on n processors. We say that the theoretical *speedup* is 2 in this case. In practice, however, this theoretical speedup is rarely achieved, because for an increasing number of processors, more and more information has to be exchanged via messages for a given problem size, such that the *communication overhead* may become significant enough to reduce the speedup. The relative communication overhead is generally smaller for larger problems. Also, the influence of *serial bottlenecks* — small portions of the program which are not parallelized— becomes more significant for high processor numbers. In some cases the theoretical speedup can be exceeded, for instance because of hardware cache effects.

It is difficult to measure the execution speed of parallel programs in a consistent way, for instance because of variability in system load. In Fig. 4.6 we show the speedup for the simulation of a 2D bow shock flow with our parallel code. The theoretical speedup is given by the solid line. The dashed line with asterisks shows the speedup curve for

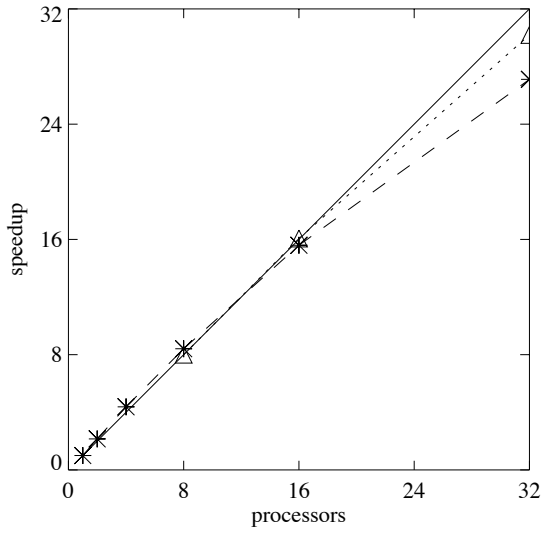

Figure 4.6: *Speedup for the simulation of a 2D bow shock flow with our parallel code on a relatively coarse grid (*80×80*, dashed with asterisks) and on a slightly finer grid (*160×160*, dotted with triangles). The theoretical speedup is given by the solid line.*

simulation on a relatively coarse grid (80×80). For $n_{proc} = 32$ the communication overhead starts to become important. For a slightly finer grid (160×160, dotted with triangles), however, the theoretical speedup is almost achieved. These curves show that our parallel code scales satisfactorily with the number of processors used. Good parallel scaling is an advantage of the explicit time integration strategy, and parallel codes based on implicit time integration scale generally less favorably [167] because matrix inversion is a global process.