THE UNIVERSITY OF
WARWICK

AUTHOR: **Thomas Goffrey** DEGREE: **Ph.D.**

TITLE: **A Cylindrical Magnetohydrodynamic Arbitrary Lagrangian Eulerian Code**

DATE OF DEPOSIT: .....

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I **agree** that the thesis may be photocopied (single copies for study purposes only).

Theses with no restriction on photocopying will also be made available to the British Library for microfilming. The British Library may supply copies to individuals or libraries. subject to a statement from them that the copy is supplied for non-publishing purposes. All copies supplied by the British Library will carry the following statement:

> "Attention is drawn to the fact that the copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's written consent."

AUTHOR'S SIGNATURE: .....

USER'S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE SIGNATURE ADDRESS

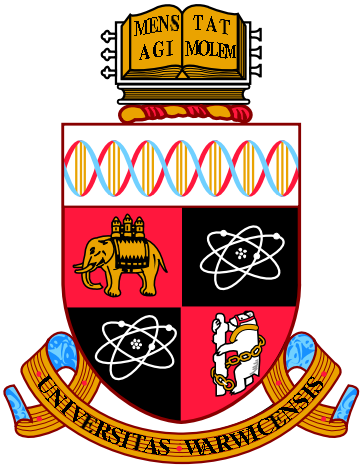
.....

.....

.....

.....

.....



A Cylindrical Magnetohydrodynamic Arbitrary Lagrangian Eulerian Code

by

Thomas Goffrey

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Physics

January 2014

THE UNIVERSITY OF WARWICK

Contents

List of Figures

Acknowledgments

Thanks go first and foremost to my supervisor professor Tony Arber who has provided excellent support and advice throughout this project. I must also thank Dr Chris Brady for providing useful discussions throughout this work, and for his work in extending the capabilities of Odin. Dr Keith Bennett has also provided valuable insights into numerical methods.

I would also like to thank Dr. Andrew Barlow for his advice on arbitrary Lagrangian Eulerian methods, and whose thesis provided the starting point for my own research in the field.

On a more personal note, I would like to thank my family for their encouragement over the years, without which I would not be writing this thesis. Finally I would like to thank Olivia, her guidance and patience were integral to the completion of this work.

This work acknowledges the financial support of AWE.

Declarations

I declare that this thesis has not been submitted for a degree at another university. This thesis describes the development of a numerical code, and as such borrows methods from previously published work, I declare that where such methods have been used references to the original work has been provided.

The methods to extend the hydrodynamical remap to second order, described in chapter 9 were implemented by Dr. C. S. Brady, although planned and tested in conjunction with the author. The description and development of these methods have been included for completeness as they are used in generating the results for the final results. All other methods described were implemented by the author.

Abstract

Arbitrary Lagrangian Eulerian methods are methods which seek to take advantage of the strengths of Eulerian and Lagrangian methods, whilst circumventing the weaknesses. This thesis discusses the development of such a code ,Odin, in two dimensions, for both Cartesian and cylindrical coordinates. Odin is capable of handling shocks through the addition of shock viscosity to the Euler equations. Furthermore the hydrodynamical scheme is expanded to include magnetohydrodynamics.

Chapter 1

Introduction

1.1 Introduction

Whilst this thesis covers the development of a two dimensional arbitrary Lagrangian Eulerian (ALE) code, it is worthwhile discussing the merits and disadvantages of more basic methods of modelling fluid flow, before introducing the ALE methodology. The most basic methodical distinction is between pure Lagrangian and Eulerian methods. Increasing in complexity, the next method is Lagrangian remap codes, which is a hybrid method somewhere between Lagrangian and Eulerian methods, and is in fact a limiting case of ALE codes.

1.2 Timestep Control

In analysing the relative advantages and disadvantages of particular numerical schemes for hydrodynamics the question of time-step control and efficiency will be a recurrent issue, as such it is prudent to explain briefly how a timestep is chosen for (explicit) hydrodynamics. The Courant-Friedrichs-Lewy (CFL, [1]) essentially states from a physical perspective that in a given time-step no information should be able to cross more than one grid cell. For example in a one dimensional Lagrangian calculation the time-step should be calculated according to,

$$\Delta t \leq \frac{C}{\Delta x}, \tag{1.1}$$

where C is the sound speed, and Δx is the width of the cell. For a complicated grid the choice of length becomes more complicated and it is common to take some simplification, and run with the time-step at some reduced factor of the maximum allowed by the CFL condition. However, the important result remains that the time-

step is (neglecting higher order schemes not considered here) inversely proportional to the minimum grid spacing.

1.3 Eulerian Methods

The most basic method of modelling fluid flow is the Eulerian method (e.g. [2],[3]). In discretising the fluid in an Eulerian code, the computational cells remain fixed in space, and allow fluid to flow through it. It is important to make a distinction here, between a pure Eulerian method, in which the grid never moves, and Eulerian mesh motion, in which at the end of the time step the grid is returned to its original position.

The main advantage in using an Eulerian method is robustness. As shall be discussed Lagrangian methods often struggle to complete computations when the flow becomes complex, this is not a problem for Eulerian methods. The physics within an Eulerian code is usually simpler to expand than their Lagrangian counterparts, due to the fact that the grid is known, and orthogonal. This simplicity of the grid means that at first glance the computational cost of a single time-step should be cheaper than Lagrangian codes, however in practice Eulerian codes can be dimensionally split (where fluxes along each coordinate direction are calculated and applied individually rather than simultaneously), so this advantage can be reversed.

Eulerian codes do also have disadvantages. As the grid is fixed in space they experience some numerical diffusion. This can damage the accuracy of the solution, in particular shocks may become smeared across a large number of zones, however this can also reduce numerical oscillations around shock fronts.

Providing the required resolution can also be problematic for Eulerian methods. As the grid is fixed, it is necessary to provide resolution in all required areas at all times. However for a number of applications the local resolution requirement may change during the simulation; different areas may be interesting at different times. This can increased the required number of cells by orders of magnitude for Eulerian codes, thus rendering them potentially very computationally expensive.

1.4 Riemann Solvers on Eulerian Grids

Riemann solvers (e.g. [4]) work by treating the computational domain as piecewise constant across cells, and solving the individual Riemann problem at each cell interface to advance the solution to the next time step. The solution is then averaged across each cell to return the domain to a piecewise constant state. The first of

such schemes [5] is only first order but higher order schemes have been developed. Whilst due to their intrinsic nature such schemes are able to capture shocks, without added complications such as shock viscosity, some Riemann solvers (both exact and approximate) do encounter difficulties in modelling shock reflections and additional dissipative methods are needed [6]. Finally, such schemes have traditionally been used for Eulerian codes, although recent efforts have seen them adapted for arbitrary grids [7]. However such schemes are computationally expensive and have further complications (such as the requirement of accurate sound speed which can be problematic) and are not considered further in this work.

1.5 Lagrangian Methods

In contrast to Eulerian methods Lagrangian methods (e.g. [8]) have a mesh which is attached to the fluid. This means that no fluid flows through cell edges during the computation, the grid moves with the fluid. This has the result that the code is less diffusive, due to the fact that the grid moves with the fluid, rather than smoothing out features during advection. This grid motion also means that the method will naturally provide time dependent resolution where it is needed. As the fluid moves in one direction or piles up in an area of the domain the grid will follow it there, providing the necessary resolution.

However these advantages come at a cost, particularly robustness. Should the flow become complex the grid may begin to twist. As the grid twists (or indeed piles up in a specific location) the distance across a cell can decrease by several orders of magnitude, which consequently reduces the time step by a proportional amount. Also the grid is completely arbitrary, so implementing increasingly complex physics can become complicated.

Comparing run time and cost between Eulerian and Lagrangian methods is tricky. Lagrangian methods do not need to be directionally split, but due to the potentially complicated nature of the grid they lack simplifications that can be made in Eulerian methods. Running with identical numbers of cells a Lagrangian method will almost certainly require more time steps to complete a calculation than Eulerian methods, due to the grid concentrating itself in areas of interest. To make a fairer comparison a higher resolution Eulerian simulation should be run. In general if Lagrangian simulations are able to complete, their results arrive quicker and more accurately, but it is a big if.

Finally, although this thesis concerns itself (in the majority) with single material methods it is worthwhile considering multi-material affects. Lagrangian methods do

not allow mass to flow between cells, and it is possible to set up the initial conditions such that cells are only one material, i.e. the grid is aligned with material interfaces. Eulerian codes will usually employ some form of interface reconstruction (e.g. [9]), but despite this, Eulerian codes are still prone to artificial mixing.

1.6 Lagrangian Remap Codes

Lagrangian remap codes (e.g. [10]) represent the middle ground between Eulerian and Lagrangian methods, and attempt to circumvent the respective problems, whilst keeping the advantages. The idea is relatively simple, carry out a single Lagrangian step, before carrying out a remap step to return the grid back to its original position. This grid motion shall be referred to as Eulerian grid motion throughout this thesis. The Lagrangian phase of a Lagrangian remap code is exactly that, and thus inherits its advantages of reduced numerical diffusion and better estimation of mixing. However as the grid is returned to its original position at the end of the time step a number of simplifications/approximations can be made without too large a drop in accuracy, thus reducing the numerical cost of such a time step.
However the remap phase does still produce numerical diffusion. The remap step comes at added computational cost, which is increased by the fact that it is often directionally split, this obviously trades off against the reduction in complexity of the Lagrangian phase.
Of course Lagrangian remap codes, due to their Eulerian grid motion do not inherit the natural resolution of Lagrangian codes, but they do inherit the robustness of pure Eulerian codes.

1.7 Arbitrary Lagrangian Eulerian Methods

Arbitrary Lagrangian Eulerian (ALE) methods (e.g. [11]) step beyond the complexity of Lagrangian remap codes. The essential difference between ALE methods and Lagrangian remap methods is that the remap is both optional, and no longer constrained to return the grid to its original position; the grid is arbitrary hence the name of the method. ALE codes attempt to run for as long as possible in Lagrangian mode, thus inheriting all the advantages of Lagrangian scheme, but when the point at which a pure Lagrangian simulation would halt is reached a remap is triggered, thus enabling the computation to continue. ALE codes are by far the most complex to develop, in that the Lagrangian scheme has the associated complications of an arbitrary grid, but now the remap may become equally, or more complicated as it

too has to cope with an arbitrary grid. The question of directionally splitting the remap is discussed in later chapters.

1.8 Thesis Outline

This thesis describes the development of a single material two-dimensional arbitrary Lagrangian Eulerian MHD code, Odin. The thesis first begins with a basic description of the discretisation and derivation of the Lagrangian phase, in Cartesian coordinates. Chapter 3 then introduces shock viscosities, and tests two popular methods. Chapter 4 discusses the necessary changes to be made to enable Odin to run in cylindrical coordinates as well as Cartesian coordinates. Chapter 5 introduces the problem of hourglass modes, and assess a widely used method to suppress them, subzonal pressures. In practice subzonal pressures are not used in Odin.
Chapter 6 introduces a number of different remapping strategies, and discusses the implementation of a first order remap method with an ALE code. The following chapter explains how to implement ideal magnetohydrodynamics (MHD) within an ALE code, both in its Lagrangian phase, and within the context of a first order remap, before chapter 8 briefly covers how to adapt such a scheme to cylindrical coordinates.
Chapter 9 then expands the remap to second order for ideal MHD, and discusses directionally splitting such a remap, before chapter 10 which presents results of the code running implosion problems with an imposed B-field. Finally chapter 11 briefly outlines direction for further work.

Chapter 2

Governing Equations

2.1 Continuous Description

2.1.1 Euler Equations, Eulerian Form

The equations governing the evolution of an ideal fluid, the Euler equations, given in their Eulerian form are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \vec{u} = 0, \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \nabla \cdot (\vec{u} \otimes (\rho \vec{u})) + \nabla P = 0, \tag{2.2}$$

$$\frac{\partial}{\partial t}(\rho e_T) + \nabla \cdot (\vec{u}(\rho e_T + P)) = 0, \tag{2.3}$$

where ρ is the density, $\vec{u}$ the velocity vector, P the (thermodynamic) pressure and $e_T = e_i + 1/2u^2$ is the specific total energy. e_i is the specific internal energy, which for an ideal gas has the form $e_i = P/\rho(\gamma - 1)$.These are simply statements of the conservation of mass, momentum and energy in an Eulerian frame. These are closed by an equation of state linking density, energy and pressure.

2.1.2 Tensor Review

A basic discussion of simple tensor calculus is included in appendix A.1. However some key results are repeated here. The dyadic is defined as,

$$\boldsymbol{A} = \vec{a} \otimes \vec{b}, \tag{2.4}$$

where,

$$A^{ij} = a^i b^j. \tag{2.5}$$

A generalised dot product is,

$$\vec{n} \cdot \left(\vec{a} \otimes \vec{b} \right) = \left(\vec{n} \cdot \vec{a} \right) \vec{b}, \tag{2.6}$$

and,

$$\left(\vec{a} \otimes \vec{b} \right) \cdot \vec{n} = \left(\vec{n} \cdot \vec{b} \right) \vec{a}. \tag{2.7}$$

The divergence of a rank two tensor is,

$$\nabla \cdot \boldsymbol{T} = \frac{\partial T_{ji}}{\partial x_j}, \tag{2.8}$$

so that the divergence of a dyadic is,

$$\nabla \cdot \left(\vec{a} \otimes \vec{b} \right) = \left(\nabla \cdot \vec{a} \right) \vec{b} + \left(\vec{a} \cdot \nabla \right) \vec{b}. \tag{2.9}$$

Finally the divergence theorem for tensors is,

$$\int_V \frac{\partial T^{ij\ldots k\ldots m}}{\partial x_k} dV = \oint_s T^{ij\ldots k\ldots m} n_k dS, \tag{2.10}$$

where n_k is the rank one covariant tensor associated with the face, and dS is the magnitude of that face area. Applying (2.10) to a dyadic,

$$\int_V \nabla \cdot \left(\vec{a} \otimes \vec{b} \right) = \oint_s \left(\vec{a} \cdot \vec{n} \right) \vec{b}. \tag{2.11}$$

2.1.3 Euler Equations, Lagrangian Form

As an ALE code evolves the equations in a Lagrangian frame the Euler equations must be rewritten using the Lagrangian derivative given by,

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \vec{u} \cdot \nabla f, \tag{2.12}$$

for a general scalar f. Expanding (2.1) and using the definition of the Lagrangian derivative, (2.12),

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \vec{u}, \tag{2.13}$$

which is a mass conservation equation in the Lagrangian frame. Considering the momentum equation, (2.2), and using (2.9),

$$\frac{\partial}{\partial t} \left(\rho \vec{u} \right) + \vec{u} \left(\nabla \cdot \rho \vec{u} \right) + \rho \vec{v} \cdot \nabla \vec{u} = -\nabla P. \tag{2.14}$$

Next, expanding the time derivative,

$$\rho\frac{\partial\vec{u}}{\partial t}+\vec{u}\frac{\partial\rho}{\partial t}+\vec{u}\left(\nabla\cdot\rho\vec{u}\right)+\rho\vec{v}\cdot\nabla\vec{v}=-\nabla P. \tag{2.15}$$

However, by (2.1), the second and third terms cancel, yielding our momentum equation in Lagrangian form,

$$\rho\frac{D\vec{u}}{Dt}=-\nabla P. \tag{2.16}$$

For a scalar and a vector, the product rule states,

$$\nabla\cdot\left(f\vec{a}\right)=\vec{a}\cdot\nabla f+f\nabla\cdot\vec{a}, \tag{2.17}$$

so it is possible to expand and rewrite (2.3),

$$\frac{\partial}{\partial t}\left(\rho e_T\right)+\rho e_T\left(\nabla\cdot\vec{u}\right)+\vec{u}\cdot\nabla\left(\rho e_T\right)=-\nabla\left(\vec{u}P\right), \tag{2.18}$$

and by using (2.12),

$$\frac{D\rho e_T}{Dt}=-\nabla\cdot\left(\vec{u}P\right)-\rho e_T\nabla\cdot\vec{u}. \tag{2.19}$$

Once again, expanding the derivative,

$$\rho\frac{De_T}{Dt}+e_T\frac{D\rho}{Dt}=-\nabla\cdot\left(\vec{u}P\right)-\rho e_T\nabla\cdot\vec{u}, \tag{2.20}$$

noting (again by using (2.13)) that the second terms on each side cancel, an equation for the Lagrangian evolution of total energy is obtained,

$$\rho\frac{De_T}{Dt}=-\nabla\cdot\left(\vec{u}P\right). \tag{2.21}$$

It is possible to recast (2.21) in terms of specific internal energy, e_i,

$$\rho\left[\frac{De_i}{Dt}+\frac{1}{2}\frac{Du^2}{Dt}\right]=-\nabla\cdot\left(\vec{u}P\right), \tag{2.22}$$

and then applying the product rule (in the form of (2.17)) and the chain rule,

$$\rho\frac{De_i}{Dt}+\rho\vec{u}\cdot\frac{D\vec{u}}{Dt}=-\vec{u}\cdot\nabla P-P\nabla\cdot\vec{u}, \tag{2.23}$$

and finally using (2.16) to cancel the second term on the left hand side with the first on the right a final energy equation in the Lagrangian frame is acquired,

$$\rho\frac{De_i}{Dt}=-P\nabla\cdot\vec{u}. \tag{2.24}$$

2.1.4 Reynolds Transport Theorem

Reynolds Transport Theorem for fluid parcels is a statement of the Leibniz integral rule. It is given by [12],

$$\frac{D}{Dt}\int\limits_{\Omega(t)} \vec{f}dV = \int\limits_{\Omega(t)} \frac{\partial \vec{f}}{\partial t}dV + \int\limits_{\partial\Omega(t)} (\vec{v}\cdot\vec{n})\,\vec{f}dA, \tag{2.25}$$

where the integration is carried out over a fluid volume, $\Omega(t)$ bounded by a surface, $\partial\Omega(t)$. The vector, $\vec{n}$ is the unit normal vector to that surface. The velocity $\vec{v}$ need not be the fluid velocity, it is simply the velocity of the bounding surface. Before using the theorem on the Lagrangian form of the Euler equations, this can be manipulated into a more useful form. Using first the generalised dot product, (2.6) (which has been shown in appendix A.1 to be equivalent to the contraction (A.7)),

$$\frac{D}{Dt}\int\limits_{\Omega(t)} \vec{f}dV = \int\limits_{\Omega(t)} \frac{\partial \vec{f}}{\partial t}dV + \int\limits_{\partial\Omega(t)} \vec{n}\cdot\left(\vec{v}\otimes\vec{f}\right)dA, \tag{2.26}$$

Now using the divergence theorem for tensors (2.10), and the definition of the divergence of a rank two (contravariant) tensor, (2.8),

$$\frac{D}{Dt}\int\limits_{\Omega(t)} \vec{f}dV = \int\limits_{\Omega(t)} \frac{\partial \vec{f}}{\partial t}dV + \int\limits_{\Omega(t)} \nabla\cdot\left(\vec{v}\otimes\vec{f}\right)dV. \tag{2.27}$$

Expanding this using (2.9),

$$\frac{D}{Dt}\int\limits_{\Omega(t)} \vec{f}dV = \int\limits_{\Omega(t)} \left[\frac{\partial \vec{f}}{\partial t} + \vec{v}\cdot\nabla\vec{f} + \vec{f}(\nabla\cdot\vec{v})\right]dV. \tag{2.28}$$

Using (2.12) yields our final form of the Reynolds Transport Theorem,

$$\frac{D}{Dt}\int\limits_{\Omega(t)} \vec{f}dV = \int\limits_{\Omega(t)} \left[\frac{D\vec{f}}{Dt} + \vec{f}(\nabla\cdot\vec{v})\right]dV. \tag{2.29}$$

2.1.5 Integral form of the Euler Equations

It is now possible to integrate the Euler equations in their Lagrangian form to gain the final evolution equations of the Lagrangian scheme within Odin. Integrating

(2.13):

$$\int\limits_{\Omega(t)} \frac{D\rho}{Dt} dV = -\int\limits_{\Omega(t)} \rho \nabla \cdot \vec{u} dV, \tag{2.30}$$

From (2.29) it follows,

$$\frac{D}{Dt} \int\limits_{\Omega(t)} \rho dV = 0. \tag{2.31}$$

It will be useful to define volume averaged quantities as,

$$\bar{f} = \frac{1}{V} \int\limits_{\Omega(t)} f dV. \tag{2.32}$$

Similarly to (2.32) a mass averaged quantity will be introduced,

$$\tilde{f} = \frac{1}{M} \int_{\Omega(t)} f \rho dV, \tag{2.33}$$

Using (2.32), (2.31) can be rewritten as,

$$\frac{D}{Dt} \bar{\rho} V = \frac{D}{Dt} M = 0, \tag{2.34}$$

where $M = \bar{\rho} V$ is the cell mass. This is a familiar result, that the mass in a Lagrangian cell is constant. Moving onto (2.16), and integrating,

$$\int\limits_{\Omega(t)} \rho \frac{D\vec{u}}{Dt} dV = -\int\limits_{\Omega(t)} \nabla P dV. \tag{2.35}$$

Considering first the left hand side,

$$\begin{aligned} \int\limits_{\Omega(t)} \rho \frac{D\vec{u}}{Dt} dV &= \int\limits_{\Omega(t)} \left(\frac{D}{Dt} (\rho \vec{u}) - \vec{u} \frac{D\rho}{Dt} \right) dV \\ &= \int\limits_{\Omega(t)} \left(\frac{D}{Dt} (\rho \vec{u}) + \vec{u} \rho (\nabla \cdot \vec{u}) \right) dV \\ &= \frac{D}{Dt} \int\limits_{\Omega(t)} \rho \vec{u} dV, \end{aligned}$$

where in the last step (2.29) has been used. Use (2.33) this can be rewritten as,

$$\frac{D}{Dt} \int_{\Omega(t)} \rho \vec{u} dV = \frac{D}{Dt} M \tilde{\vec{u}}. \tag{2.36}$$

and by applying the divergence theorem to the right hand side and using (2.36) the final integral equation for the momentum update in Cartesian coordinates is obtained,

$$M\frac{D\tilde{\vec{u}}}{Dt} = -\int\limits_{\partial\Omega(t)} P d\vec{S}. \tag{2.37}$$

It is desirable to apply the same method to define a model for updating the energy equation. However as will be shown such a simple form cannot be obtained without some questionable steps. In fact Odin uses a compatible energy update [13] in place of such a model. The discussion that follows is provided for completeness. As with the momentum equation it is possible to rearrange the left hand side of the integral of the energy equation, (2.24),

$$M\frac{D\tilde{e_i}}{Dt} = -\int\limits_{\Omega(t)} P\nabla\cdot\vec{u}dV. \tag{2.38}$$

The pressure relates to density and specific internal energy through the equation of state. As finite volume codes only ever model the volume/mass average of such quantities, $\bar{\rho}$ and $\tilde{e_i}$, it follows that only a volume average pressure, $\bar{P}$ is ever known. Thus it is possible to remove the pressure from the integral, now being explicit that only an average pressure value is known, to yield an energy update of the form,

$$M\frac{D\bar{e_i}}{Dt} = -\bar{P}\int\limits_{\Omega(t)} \nabla\cdot\vec{u}dV = -\bar{P}\int\limits_{\partial\Omega(t)} \vec{u}\cdot d\vec{S}. \tag{2.39}$$

However in place of (2.39) a compatible energy update is used to ensure exact energy conservation. This method shall be explained in a later section. In this section $\bar{f}$ has been used to denote a volume averaged quantity and $\tilde{f}$ has been used to denote a mass averaged quantity as defined by (2.32) and (2.33) respectively. It is these quantities that a finite volume code works in terms of, however for the sake of brevity this notation shall be dropped, and the variables being considered shall be assumed to be mass (or in the case of density, volume) averaged quantities unless stated otherwise.

2.2 Discrete Description

2.2.1 Hydrodynamical Variable Placement

Before discussing the details of the numerical scheme within Odin it is necessary to define the placement of each variable on the grid. Odin employs a staggered grid hydrodynamic scheme, where variables such as density, volume and internal energy are defined at cell centres, and variables such as velocity and position are defined at nodes. This is demonstrated by figure 2.1.

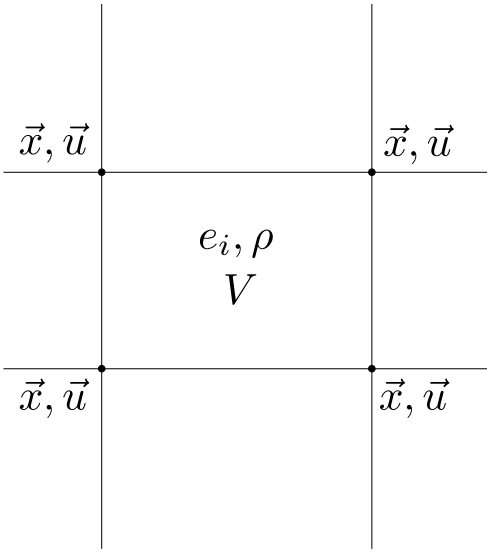


Figure 2.1: Hydrodynamical variable placement on a staggered grid.

A cell (or zone) is a quadrilateral formed by four nodes, nodes are intersections of the grid, marked by a dot in figure 2.1. Any derived variables will be defined in the same position as the relevant primary variables, so pressure will be defined at the cell centre, as will cell mass. There is also a nodal mass, from which momentum and kinetic energy can be derived, which are nodal values. The definition of the nodal mass is described in detail in the next section. Figure 2.1 shows that each cell (ir, iz) has four velocities associated with it, the indexing is demonstrated in figure 2.2. This same indexing is used when associating position variables with cells, and the four relevant nodal positions are averaged to define a cell centred position. The use of (ir, iz) rather than (ix, iy) is to aid comparison with the code, which was originally conceived as a cylindrical code with variable names that reflect that. This convention will be used at various stages of this work, as a result (ir, iz) and (ix, iy) should be seen only as labels, not necessarily associated with any particular

coordinate system. In order to calculate forces for each node, the four contributions

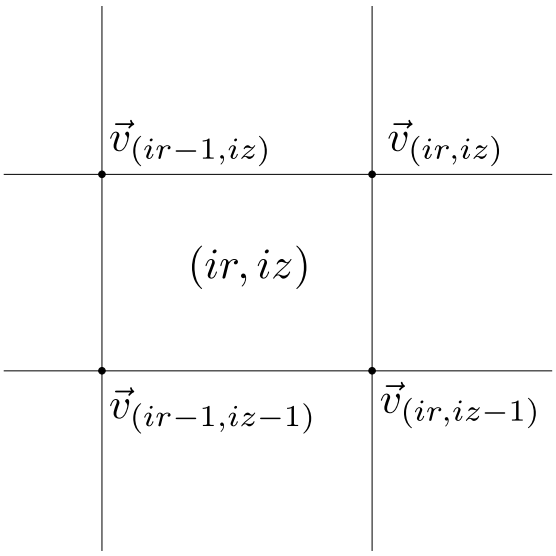


Figure 2.2: Indexing for the four velocity vectors associated with each cell (ir,iz).

from each of its four connected cells will need to be considered. When doing so it is useful to define and retain an indexing. The four cells associated with each node are numbered 1-4 in an anticlockwise manner, starting bottom left, this is demonstrated by figure 2.3. A similar indexing is used for the nodes associated with each cell, this

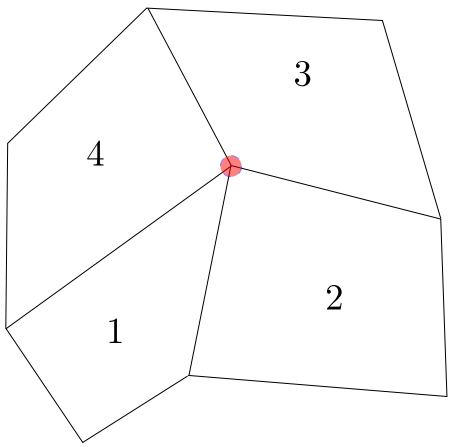


Figure 2.3: Indexing used for the cells associated with a node (highlighted in red).

is demonstrated by figure 2.4. This diagram also provides the indexing of the edges

of each cell, which once again are defined in an anticlockwise manner, this time starting at the bottom edge.

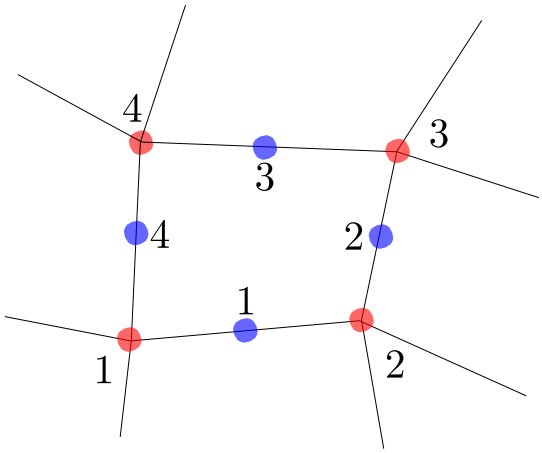


Figure 2.4: Indexing used for the four nodes associated a cell. The nodes are highlighted in red. The indexing for the cell edges is also denoted by numbers, and the cell edge midpoints have been highlighted blue for clarity.

2.2.2 Compatible Energy Update

A compatible energy update, [13], may be viewed as choosing a discretisation which always analytically conserves energy. Essentially compatibility involves the calculation of the forces used for the update of the momentum equation, then replacing the normal energy update equation with a combination of those forces and the half time step velocities, in such a way that the total energy is conserved over the full timestep.

Definition of Corner Masses

In order to explain the full compatible energy update it is necessary to introduce the concept of a corner mass. In (staggered grid) Lagrangian methods the mass of the zone is considered constant, however the mass associated with a node, the nodal mass is often recalculated at the beginning of each time step. This change in nodal mass leads to a change in momentum and kinetic energy, which requires an extra term in the momentum equation and is usually neglected. Caramana et al [13] employed a stronger Lagrangian formulation in which both the zonal mass, M_z,

and the nodal mass, M_P, are considered constant.
The result of this is that there are two sets of boundaries over which no mass flows, the boundaries of the cells (the primary mesh) and the boundaries of the nodes (or the median mesh). The median mesh is the mesh defined by connecting the midpoints of each cell edge. Both meshes are demonstrated by figure 2.5.

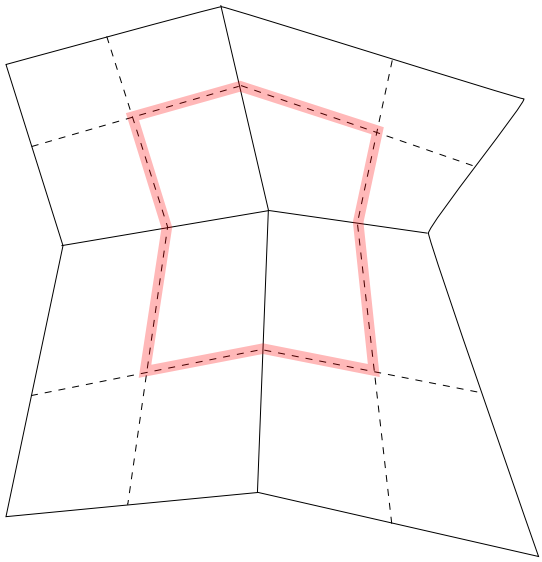


Figure 2.5: The two meshes are illustrated. The primary mesh (solid lines) is defined by the connection of nodes with their nearest (in logical space) neighbours. The median mesh (dotted lines) is defined by the connection of the midpoints of cell edges. The median mesh segments used to define the volume associated with single node have been highlighted red. As both nodal and cell centred masses are assumed to constant, no mass can flow through either mesh.

Thus two sets of constant masses are defined, however their sum must be equal,

$$\sum_p M_p = \sum_z M_z, \tag{2.40}$$

where the first summation in (2.40) is carried out over all nodes (or points using the nomenclature of [13]), and the second over all cells (zones). Given that these two boundaries through which no mass flows intersect it is natural to assume that the subzones formed by these intersections also have constant mass. It is these subzonal

masses that are referred to as corner masses, and shall be denoted m_j^i, where the indexing is explained in the following section.

It should be stressed that in fixing the subzonal masses as constant masses associated with the intersection of the primary and median mesh a finite volume numerical model is being defined. The centre of a zone is not itself a Lagrangian point, and the assumption that no mass flows across the median mesh is again simply a step in defining a numerical model. The finite volume model will converge to the correct limit, but is not one derivable from the fluid equations. This is why the step to use a compatible energy update in place of the integral energy equation is important. As will be shown this numerical model does exactly conserve mass, energy and momentum. To calculate the mass of a zone or node the following summations (keeping for now the notation of [13]) are carried out,

$$M^z = \sum_p m_p^z, \tag{2.41}$$

and,

$$M^p = \sum_z m_z^p. \tag{2.42}$$

where in (2.41) the summation is carried out over all points associated with the zone (see figure 2.4) and in (2.42) over all zones associated with a point (see figure 2.3). To be clear, $m_p^z = m_z^p$, but summations are always carried out with respect to the lower index. To summarise the mass of a zone is calculated by summing over all corner masses with that fixed label z, and to calculate the nodal mass the summation is carried out over fixed label p.

Clearly if there are $nr \times nz$ cells there will be $2nr \times 2nz$ corner masses. Within Odin the corner masses (m) are given two indices, i and j, so

$$M_{Cell}(ir, iz) = \sum_{i=2ir-1}^{2ir} \sum_{j=2iz-1}^{2iz} m_{i,j}, \tag{2.43}$$

and,

$$M_{Node}(ir, iz) = \sum_{i=2ir}^{2ir+1} \sum_{j=2iz}^{2iz+1} m_{i,j}. \tag{2.44}$$

These summations are illustrated in figures 2.6 and 2.7, and are the implementations of (2.41) and (2.42).

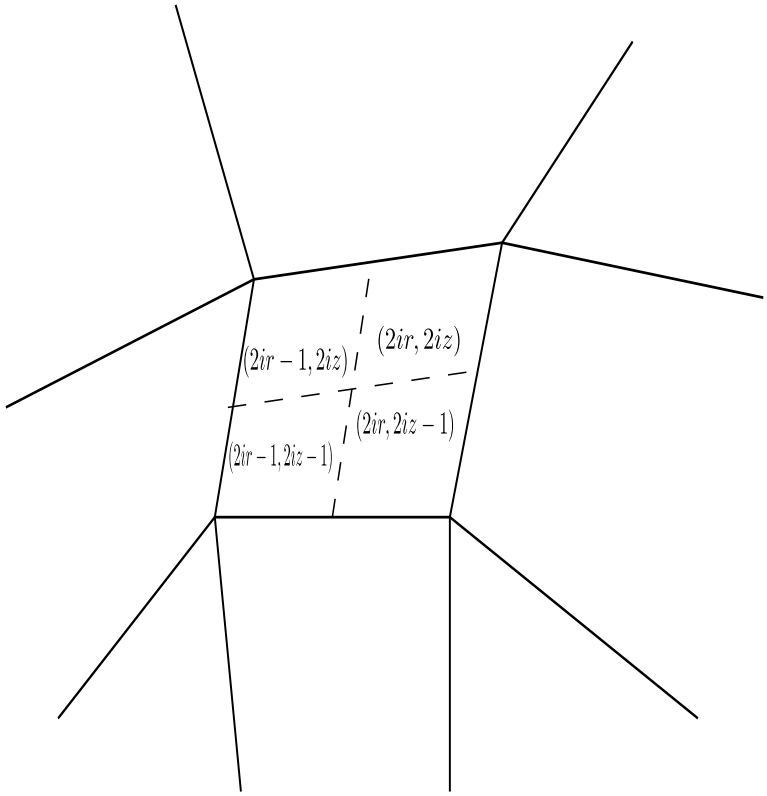


Figure 2.6: Indexing for the four corner masses contributing to the total mass of cell (ir,iz).

Compatible Formulation

Before using the corner masses described in the previous subsection, some grid vectors must be defined. There are two sets of grid vectors. For each cell there are associated eight primary grid vectors, two to each cell face. These are equal in magnitude to half the area of that cell face and point in an outward normal direction with respect to the cell and it's face. The indexing for these is illustrated by figure 2.8.

The median mesh is a secondary mesh defined by the line segments connecting the midpoints of the primary mesh. Each cell has two median mesh lines associated with it, and four corresponding median mesh vectors each equal in magnitude to half the area associated with its median mesh line. This area in a two dimensional Cartesian code is of course a length, but in different geometries this will not be the

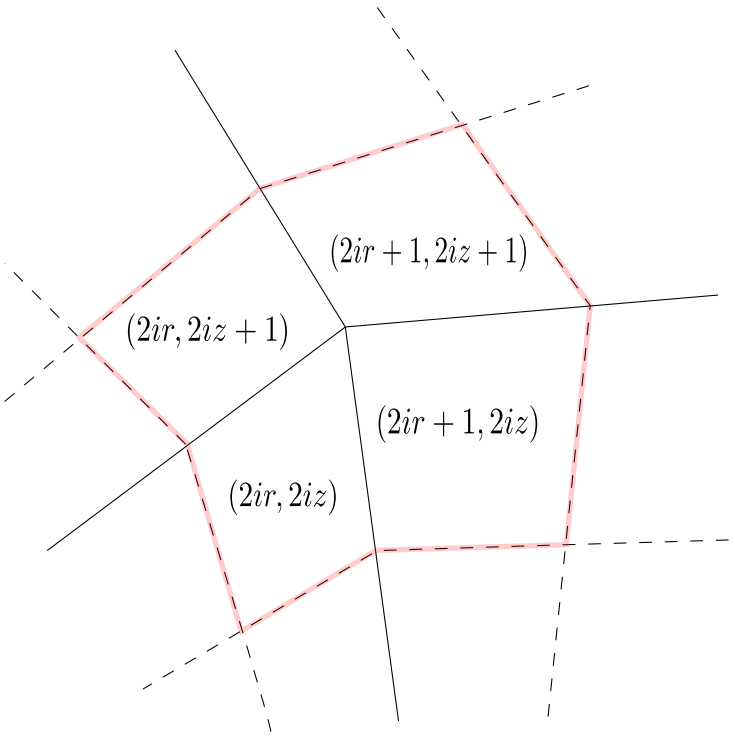


Figure 2.7: Indexing for the four corner masses contributing to the total mass of node (ir,iz). For clarity the line segments defining the nodal cell have been highlighted in red.

case, and in three dimensions it will be an area, as such it is referred to as an area. The direction and indexing of the median mesh vectors are defined by figure 2.9.For a Cartesian code $\vec{S}_1 = -\vec{S}_3$, however this is not the case for a cylindrical code, where the vectors may be defined at a different radial coordinate.

Consider the integral form of the momentum equation in Cartesian coordinates (2.37),

$$M_p \frac{D\vec{v}_p}{Dt} = -\int_V \nabla P dV = -\oint_{\delta S} P\vec{n} dS = \sum_z \vec{f}_z^p. \tag{2.45}$$

In the final step a similar notation to that of corner masses has been employed, where the summation is carried out over all (yet to be defined) corner forces associated with the node, p. In the context of (2.45) $\vec{f}_z^p$ is the force contribution due to pressure of a zone z, on its node p. In general this force need not just be derived from the

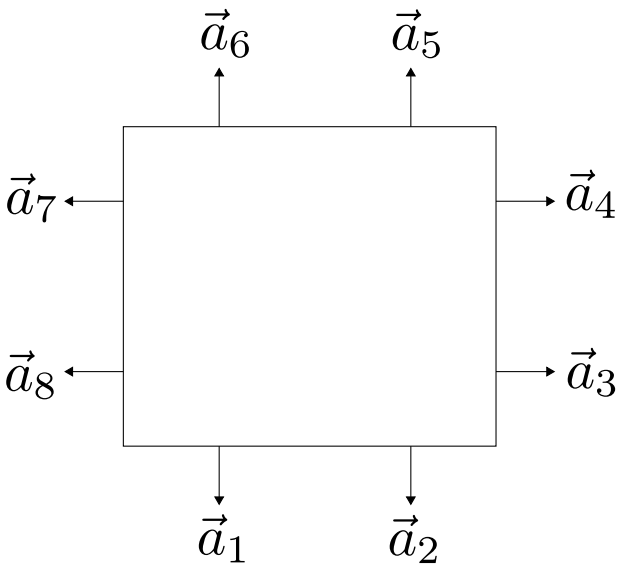


Figure 2.8: Indexing used for the primary mesh vectors, $\vec{a}_i$.

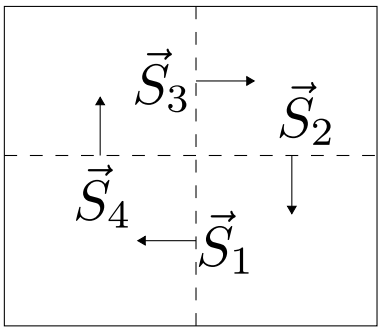


Figure 2.9: Indexing and orientation for median mesh vectors. The median mesh itself is shown by the dotted line. This diagram is used to define directions, so that $\vec{S}_1 = 0.5\,(y_1 - y_c, x_c - x_1)$ and $\vec{S}_3 = 0.5\,(y_3 - y_c, x_c - x_3)$, where x_c, y_c are the coordinates of the cell centre, and x_i, y_i are coordinates of cell edges, with indexing defined by figure 2.4. In cylindrical coordinates these vectors will have an additional radial weighting.

pressure gradient. Indeed a corner force is not the total force on a corner mass, but the total force on a single node from a single zone.

Corner forces are the four force contributions of a cell to each of its nodes. As forces are carried out by calculating the integral (2.37) around the median mesh the forces from each of the four cells associated with each node can be written as four discrete force contributions. These discrete forces are known as corner forces, and use the same indexing system as the corner masses.

To examine the total energy of the domain two sums must be carried out, over nodes and cells,

$$E_T = \sum_{Cells} M_z e_z + \sum_{Nodes} \frac{M_p v_p^2}{2}, \tag{2.46}$$

where e_z is the specific internal energy of the cell. The total energy across the domain is,

$$E_T = \sum_{z} M_z e_z + \sum_{p} M_p \frac{v_p^2}{2}. \tag{2.47}$$

Taking the total derivative with respect to time, and using the Lagrangian nature of both the cell masses, and the nodal masses,

$$\begin{aligned} \frac{D}{Dt} E_T &= \sum_z M_z \frac{De_z}{Dt} + \sum_p \sum_z \vec{f}_z^p \cdot \vec{v}_p \\ &= \sum_z \left(M_z \frac{De_z}{Dt} + \sum_p \vec{f}_z^p \cdot \vec{v}_p \right) + \sum_i \vec{f}_{bd,i} \cdot \vec{v}_{bd,i}. \end{aligned} \tag{2.48}$$

The final term in (2.48) represents contributions from boundary forces, which will be neglected for the remaining part of this discussion. Now in order for the contributions from the cells to total zero, the following condition is clear,

$$M_z \frac{D}{Dt} e_z = -\sum_p \vec{f}_p^z \cdot \vec{v}_p. \tag{2.49}$$

Caramana et al [13] went on to show that in fact compatibility is a natural consequence of control volume differencing in Cartesian coordinates. Using (2.49) for the

update of internal energy gives the following,

$$\begin{aligned} M_z \frac{De_z}{Dt} &= -\sum_p \vec{v}_p \cdot \vec{f}_p^z \\ &= -P_z \left[\left(\vec{S}_1 - \vec{S}_4 \right) \cdot \vec{v}_1 + \left(\vec{S}_2 - \vec{S}_1 \right) \cdot \vec{v}_2 + \left(\vec{S}_3 - \vec{S}_2 \right) \cdot \vec{v}_3 \right. \\ &\quad \left. + \left(\vec{S}_4 - \vec{S}_3 \right) \cdot \vec{v}_4 \right] \\ &= -P_z \left[(\vec{a}_8 + \vec{a}_1) \cdot \vec{v}_1 + (\vec{a}_2 + \vec{a}_3) \cdot \vec{v}_2 + (\vec{a}_4 + \vec{a}_5) \cdot \vec{v}_3 \right. \\ &\quad \left. + (\vec{a}_6 + \vec{a}_7) \cdot \vec{v}_4 \right] \\ &= -\oint_{\partial S} P_z \vec{n} \cdot \vec{v} dS \\ &= -\int_V P_z \nabla \cdot \vec{v} dV. \end{aligned} \tag{2.50}$$

In the first step of (2.50) vector addition has been used; it should be stressed that this equality is only true for Cartesian coordinates. The second step has carried out a piecewise constant boundary integral, and the final step is just the application of the divergence theorem.

Although (2.50) shows that compatible discretisation occurs naturally for Cartesian coordinates in any dimension this is not true for other coordinate systems, where the third equality does not hold. As such (2.51) is adopted as the canonical form of the internal energy update in both Cartesian and cylindrical coordinate systems,

$$M^z \frac{De_i}{Dt} = -\sum_{i=1}^{4} \vec{f}_i \cdot \vec{v_i}, \tag{2.51}$$

where the forces are the same forces used to update the velocity components of the nodes,

$$M^p \frac{Dv_x}{Dt} = \sum_{i=1}^{4} f_{x,i}. \tag{2.52}$$

The implementation of a compatible energy update can be viewed as the replacement of the continuous energy change, defined as the product of pressure and change in volume, with a discretised total work rate. The discrete forces are combined with the discrete velocities to form a discrete rate of work, and thus energy change of the cell.

The final step is to introduce time variation into the scheme described. Integrating

(2.48) over a single time step yields,

$$\sum_z M_z \Delta e_z + \sum_p M_p \Delta \frac{v_p^2}{2} = \sum_z M_z \Delta e_z + \sum_p M_p \vec{v}_p^{n+1/2} \cdot \Delta \vec{v}_p = 0, \tag{2.53}$$

where Δ represents the change in quantities between time steps n and n+1. The discrete (in both time and space) form of the momentum update is

$$M_p \Delta \vec{v}_p = \sum_z \vec{f}_z^{p(n+1/2)} \Delta t, \tag{2.54}$$

where $\vec{f}_z^{p(n+1/2)}$ represents a time centred force. Applying this and changing the order of the resulting double summation,

$$\sum_z \left[M_z \Delta e_z + \sum_p \vec{v}_p^{n+1/2} \cdot \vec{f}_p^{z(n+1/2)} \Delta t \right] = 0, \tag{2.55}$$

yields the final fully discretised form of the compatible energy update,

$$\Delta e_z = -\sum_p \vec{f}_p^z \cdot \vec{v}_p^{n+1/2} \Delta t / M_z. \tag{2.56}$$

Additional Forces

The compatible framework is a very flexible one that allows new (hydrodynamical) forces to be added in an energy conserving manner. Any force may be specified directly as a force on the nodes and (2.56) provides an energy conserving framework to calculate the resulting change in internal energy. This will prove a particularly useful tool in specifying shock viscosity forces and calculating the resulting heating. It is also worth noting that it is possible to reverse engineer such a process where some heating mechanism may be specified, and the necessary energy conserving forces may be calculated using the compatible framework.

There are a few notable exceptions. The first is the force resulting from the addition of a B-field which has no effect on the internal energy, the forces here provide a transfer of energy between the magnetic and kinetic energies. Secondly in the case of gravity (2.47) must be modified as,

$$E_{T,cell} = M_z e_z + \sum_p m_p^z \left(\frac{v_p^2}{2} + g y_p \right) \tag{2.57}$$

where y_p is the height of the node and g is the strength of the gravitational field, to account for the additional gravitational force,

$$\vec{f}_{grav} = -M_p g \hat{y}. \tag{2.58}$$

To show energy is still conserved, consider only the additional terms arising from the inclusion of a (constant) gravitational acceleration,

$$\begin{aligned} \frac{D}{Dt}\sum_p \frac{M_p \vec{v}_p^2}{2} + g y_p &= \sum_p M_p \vec{v}_p \cdot \frac{D}{Dt}\vec{v_p} + g M_p \frac{D}{Dt} y_p \\ &= \sum_p M_p v_{p,y} \left(-g\right) + M_p v_{p,y} \\ &= 0. \end{aligned} \tag{2.59}$$

Here it is assumed gravity acts in the negative y-direction, although a similar result is obtained for a general gravitational field, as long as the definition of gravitational potential is correctly adjusted.

2.3 Boundary Conditions

In order to calculate forces on nodes along the boundary of the domain, as well as to carry out a remap (see Chapter 6, Chapter 9) it is necessary to define two ghost values for each variable external to the domain. The values are set according to boundary conditions. Throughout this work the only boundary conditions applied will be periodic or reflective. For simplicity these will be explained in a one dimensional context, application to a second dimension is trivial. The grid size is assumed to equal n cells.

2.3.1 Hydrodynamical Variables

When using a periodic boundary condition, the density ghost values are applied as,

$$\begin{aligned} \rho_0 &= \rho_n, \\ \rho_{-1} &= \rho_{n-1}, \end{aligned} \tag{2.60}$$

for the left hand boundary, and,

$$\begin{aligned} \rho_{n+1} &= \rho_1, \\ \rho_{n+2} &= \rho_2, \end{aligned} \tag{2.61}$$

for the right hand boundary. The specific internal energy is applied in an analogous manner. For the velocity components the indexing is changed slightly, due to grid staggering,

$$\begin{aligned} \vec{v}_{-1} &= \vec{v}_{n-1}, \\ \vec{v}_{-2} &= \vec{v}_{n-2}, \end{aligned} \tag{2.62}$$

for the left hand boundary, and,

$$\begin{aligned} \vec{v}_{n+1} &= \vec{v}_1, \\ \vec{v}_{n+2} &= \vec{v}_2, \end{aligned} \tag{2.63}$$

for the right hand boundary. Finally it is important to enforce,

$$\vec{v}_n = \vec{v}_0. \tag{2.64}$$

For reflective boundary conditions, the important principle is that no material is permitted to flow through the boundary. Considering a reflective boundary in the x-direction,

$$v_{x,0} = v_{x,n} = 0. \tag{2.65}$$

The other variables are calculated to be consistent with this,

$$\begin{aligned} \rho_0 &= \rho_1, \\ \rho_{-1} &= \rho_2, \\ \rho_{n+1} &= \rho_n, \\ \rho_{n+2} &= \rho_{n-1}, \end{aligned} \tag{2.66}$$

for density, and similarly for energy. The y-velocity is calculated as,

$$\begin{aligned} v_{y,-1} &= v_{y,1}, \\ v_{y,-2} &= v_{y,2}, \\ v_{y,n+1} &= v_{y,n-1}, \\ v_{y,n+2} &= v_{y,n-2}. \end{aligned} \tag{2.67}$$

The y-velocity along the boundary is allowed to evolve as any other point, with forces calculated using the other boundary values. Finally the other boundary values for the x-component of the velocity are defined,

$$\begin{aligned} v_{x,-1} &= -v_{x,1}, \\ v_{x,-2} &= -v_{x,2}, \\ v_{x,n+1} &= -v_{x,n-1}, \\ v_{x,n+2} &= -v_{x,n-2}. \end{aligned} \tag{2.68}$$

This set of boundary conditions allow the first set of boundary values in each direction to be updated during the predictor step in a consistent manner, without the need for an additional application of the boundary conditions. For reflective boundary conditions in the y-direction the roles of the x and y components of the velocity are interchanged. The scalar quantities are unchanged.
It is also necessary to apply boundary conditions to the grid positions. For reflective boundary conditions the following equations should be used,

$$\begin{aligned} x_{-1} &= x_0 - (x_1 - x_0) = 2x_0 - x_1, \\ x_{-2} &= x_0 - (x_2 - x_0) = 2x_0 - x_2, \\ x_{n+1} &= x_n + (x_n - x_{n-1}) = 2x_n - x_{n-1}, \\ x_{n+2} &= x_n + (x_n - x_{n-2}) = 2x_n - x_{n-2}, \end{aligned} \tag{2.69}$$

and,

$$\begin{aligned} y_{-1} &= y_1, \\ y_{-2} &= y_2, \\ y_{n+1} &= y_{n-1}, \\ y_{n+2} &= y_{n-2}. \end{aligned} \tag{2.70}$$

For periodic boundary conditions the following boundary conditions should be used,

$$\begin{aligned} x_{-1} &= x_0 - (x_n - x_{n-1}), \\ x_{-2} &= x_0 - (x_n - x_{n-2}), \\ x_{n+1} &= x_n + (x_1 - x_0), \\ x_{n+2} &= x_n + (x_2 - x_0), \end{aligned} \tag{2.71}$$

and,

$$\begin{aligned} y_{-1} &= y_{n-1}, \\ y_{-2} &= y_{n-2}, \\ y_{n+1} &= y_1, \\ y_{n+2} &= y_2. \end{aligned} \tag{2.72}$$

finally, being sure to enforce,

$$y_0 = y_n, \tag{2.73}$$

and,

$$x_0 = x_n. \tag{2.74}$$

2.3.2 Polar Grids

Boundary conditions for a polar grid are required to be different than for a standard, or Cartesian grid. It is also no longer possible to restrict the discussion to one dimension. Consequently the grid is extended to be of size $n \times n$. A polar grid is formed by a set of degenerate grid points along what is logically the left hand side boundary. All of these points are forced to be at $(0, 0)$. This has the result that the boundary which is the logical up, or positive y, boundary becomes the left hand side boundary, the logical lower boundary, remains as such. The logical right hand boundary becomes an outer radial boundary, and the left hand becomes an inner radial boundary. The treatment of scalars remains unchanged, however velocity components and positional boundary conditions must be altered. For the upper and lower logical boundary the modified boundary conditions for the velocity

components are,

$$\begin{aligned} v_{y,i,-1} &= -v_{y,i,1}, \\ v_{y,i,-2} &= -v_{y,i,2}, \\ v_{y,i,n+1} &= v_{y,i,n-1}, \\ v_{y,i,n+2} &= v_{y,i,n-2}. \end{aligned} \tag{2.75}$$

Here it has been assumed the velocity (and later the position) is indexed (i, j). These statements apply for all i and j. The x-component is calculated as,

$$\begin{aligned} v_{x,i,-1} &= v_{x,i,1}, \\ v_{x,i,-2} &= v_{x,i,2}, \\ v_{x,i,n+1} &= -v_{x,i,n-1}, \\ v_{x,i,n+2} &= -v_{x,i,n-2}. \end{aligned} \tag{2.76}$$

Velocity components normal to the boundaries are forced to be zero,

$$v_{y,i,0} = 0, \tag{2.77}$$

and,

$$v_{x,i,n} = 0. \tag{2.78}$$

The velocity component tangential to the boundary are allowed to evolve as any other interior point. For the logically left hand boundary, the velocity components are given by,

$$\begin{aligned} v_{x,-1,j} &= -v_{x,1,j}, \\ v_{x,-2,j} &= -v_{x,2,j}, \\ v_{y,-1,j} &= -v_{y,1,j}, \\ v_{y,-2,j} &= -v_{y,2,j}. \end{aligned} \tag{2.79}$$

The velocities at the origin are forced to be equal to zero,

$$v_{x,0,j} = v_{y,0,j} = 0.0. \tag{2.80}$$

For the logically right, or outer radial boundary the following boundary conditions are applied,

$$\begin{aligned} v_{x,n+1,j} &= v_{x,n-1,j}, \\ v_{x,n+2,j} &= v_{x,n-2,j}, \\ v_{y,n+1,j} &= v_{y,n-1,j}, \\ v_{y,n+2,j} &= v_{y,n-2,j}, \end{aligned} \tag{2.81}$$

whilst velocities on the outer radial limit are calculated normally, using the values from other boundary conditions to calculate forces.
Grid positions are done in a similar, consistent manner. Firstly the points on the inner radial boundary are forced to remain at the origin,

$$x_{0,j} = y_{0,j} = 0.0 \tag{2.82}$$

Positions on the inner radial boundary are given as,

$$\begin{aligned} x_{-1,j} &= -x_{1,j}, \\ x_{-2,j} &= -x_{2,j}, \\ y_{-1,j} &= -y_{1,j}, \\ y_{-2,j} &= -y_{2,j}. \end{aligned} \tag{2.83}$$

Positions on the outer radial boundary are updated according to the respective velocity values, whilst beyond this the positions are given by,

$$\begin{aligned} x_{n+1,j} &= 2x_{n,j} - x_{n-1,j}, \\ x_{n+2,j} &= 2x_{n,j} - x_{n-2,j}, \\ y_{n+1,j} &= 2y_{n,j} - y_{n-1,j}, \\ y_{n+2,j} &= 2y_{n,j} - y_{n-2,j}. \end{aligned} \tag{2.84}$$

For the logically up boundary the positions are given as,

$$\begin{aligned} x_{i,n+1} &= -x_{i,n-1}, \\ x_{i,n+2} &= -x_{i,n-2}, \\ y_{i,n+1} &= y_{i,n-1}, \\ y_{i,n+2} &= y_{i,n-2}, \end{aligned} \tag{2.85}$$

whilst enforcing $x_{i,n} = 0.0$ and allowing $y_{i,n}$ to update according the velocity in that position. The positions for the logically lower boundary are given by,

$$\begin{aligned} x_{i,-1} &= x_{i,1}, \\ x_{i,-2} &= x_{i,2}, \\ y_{i,-1} &= -y_{i,1}, \\ y_{i,-2} &= -y_{i,2}, \end{aligned} \tag{2.86}$$

whilst enforcing $y_{i,0} = 0.0$ and updating $x_{i,0}$ according to the velocity in that position.

Finally, using periodic boundary conditions it is possible to model an entire (infinite) cylinder, by linking the logically upper and lower boundaries. The method for this remains the same as for periodic boundary conditions on a standard grid, except the positions are now given by,

$$\begin{aligned} x_{i,-1} &= x_{i,1}, \\ x_{i,-2} &= x_{i,2}, \\ y_{i,-1} &= y_{i,1}, \\ y_{i,-2} &= y_{i,2}, \end{aligned} \tag{2.87}$$

and,

$$\begin{aligned} x_{i,n+1} &= x_{i,n-1}, \\ x_{i,n+2} &= x_{i,n-2}, \\ y_{i,n+1} &= y_{i,n-1}, \\ y_{i,n+2} &= y_{i,n-2}, \end{aligned} \tag{2.88}$$

whilst enforcing,

$$x_{i,0} = x_{i,n}, \tag{2.89}$$

and,

$$y_{i,0} = y_{i,n}. \tag{2.90}$$

Chapter 3

Shock Viscosity

3.1 Introduction

In an ideal fluid a shock is a discontinuous jump in velocity, density and pressure. However in a non-ideal fluid dissipative effects act to smear the discontinuity over a finite distance. In numerical modelling shock viscosity acts in a dissipative manner to enable the numerical study of situations involving such phenomena. Early shock viscosities acted as a scalar pressure added only in cells which were judged to contain a shock. As such the momentum equation is modified to,

$$\rho \frac{D\vec{v}}{Dt} = -\nabla \left(P + Q \right), \tag{3.1}$$

where Q is the shock viscosity. Von Neumann and Richtmeyer [14] were the first to introduce such a concept whilst modelling the propagation of shocks in a one-dimensional, inviscid fluid. Often referred to as the quadratic viscosity term it had a general form of,

$$Q_{quad} = c_q^2 \rho \left(\Delta x \right)^2 \left(\frac{\partial \vec{v}}{\partial x} \right)^2, \tag{3.2}$$

where Δx is the width of the cell in question, and c_q is a dimensionless constant used to control the magnitude of the shock viscosity, which was set to zero when the velocity gradient was greater than or equal to zero (when the cell was expanding). Landshoff [15] introduced what is known as a linear viscosity term,

$$Q_{linear} = c_l \rho \Delta x C_s \left| \frac{\partial \vec{v}}{\partial x} \right|, \tag{3.3}$$

where here C_s is the local sound speed and c_l is another dimensionless constant used to control the magnitude of the linear shock viscosity. He recommended a

combination of the two terms,

$$Q_{Tot} = Q_{Linear} + Q_{Quad}. \tag{3.4}$$

The size of these constants remained arbitrary and problem dependent (although both must be positive during compression, and set to zero in expansion), until the work of Kuropatenko [16], later repeated by Wilkins [17]. Kuropatenko considered the Hugoniot relations and using an ideal gas equation of state derived an expression for the pressure jump across a shock:

$$P_1 - P_0 = \frac{\gamma+1}{4}\rho_0 \left(\Delta v\right)^2 + \rho_0 \left|\Delta v\right| \left[\left(\frac{\gamma+1}{4}\right)^2 \left(\Delta v\right)^2 + C_{s,0}^2\right]^{1/2}. \tag{3.5}$$

What is evident here is that the early a posteriori efforts of Von Neumann and Richtmeyer, and Landschoff, were acting to try and fulfil the requirements of shock physics; the early artificial (shock) viscosities were acting to provide a pressure jump across the shock. The pressure jump provided by (3.5) is of the form of (3.4), a combination of linear and quadratic forms. Kuropatenko also noted that in the limit of small and large $\left(\Delta v\right)^2$, relative to $C_{s,0}$, linear and quadratic terms like (3.2) and (3.3) respectively are recovered. The pressure jump given by (3.5) shall be used in the definition of shock viscosities and will be denoted,

$$q_{kur} = \frac{\gamma+1}{4}\rho_0 \left(\Delta v\right)^2 + \rho_0 \left|\Delta v\right| \left[\left(\frac{\gamma+1}{4}\right)^2 \left(\Delta v\right)^2 + C_{s,0}^2\right]^{1/2}. \tag{3.6}$$

3.2 Edge Based Shock Viscosity

Odin contains two (optional) types of shock viscosity, the first of which is the edge based shock viscosity developed by Caramana et al [18]. Following the methodology of Schulz [8], Caramana et al [18] set out a number of physical qualities that are desirable for a shock viscosity to possess. It is this set that shall be considered in this work.

3.2.1 Requirements of Shock Viscosity

The five requirements of shock viscosity outlined by Caramana, and now used as a standard set are,

1. Dissipativity

2. Galilean Invariance
3. Self-similar motion Invariance
4. Wave front Invariance
5. Viscous Force Continuity

Dissipativity is simply the requirement that the shock viscosity must always act to decrease kinetic energy and thus increase internal energy. This will sometimes be referred to as the viscous heating requirement. Shock viscosity in Odin is implemented within the compatible framework, so the sum of the viscous contributions to the change in internal energy must always be positive.
The remaining four requirements are all in some way connected to the identification of regions where the shock viscosity should act. In previous discussions the dimensionality of the problem has been limited to one, and thus identifying a shock has been limited to considering the (one dimensional) velocity gradient. When the velocity gradient across the cell is negative the cell is being compressed and the viscosity switches on, in regions of a positive gradient the viscosity is switched off.
Galilean invariance requires that the shock viscosity vanishes smoothly as the velocity becomes constant. Put more simply the viscosity must not change under the addition of a uniform velocity field. Such a requirement was also used by Schulz.
Self similar motion invariance requires the viscosity to not act in cells undergoing rigid rotation or uniform contraction. This condition is actually split into two conditions in Schulz's work. Uniform contraction (or expansion) over the entire medium is considered a reversible process. Most importantly (regardless of discussions of entropy change) shock viscosity should only account for shock heating, so a shock viscosity should be independent of such motion. Rigid rotation was Shultz's fourth requirement, and in fact was not fulfilled by his work. He linked this to his failure to conserve angular momentum. Such a motion is not a shock, and shock viscosity should not respond to it. Both rigid rotation and uniform contraction are considered jointly under the self similar motion invariance requirement.
Wave front invariance simply requires that the shock viscosity has no effect along a line of constant phase. Finally viscous force continuity requires that viscous forces go to zero and remain so in the transition from compression to expansion.

3.2.2 Definition of Edge Based Shock Viscosity

Edge based viscosity differs from the previous discussions in that, in conjunction with the aforementioned requirements, it was directly postulated as a force, rather

than as a scalar which when acted upon by the gradient operator would become a force. The form of the force is,

$$\vec{f}_i^{visc} = q_{kur} \times (1 - \psi) \left(\Delta\vec{v}_i \cdot \vec{S}_i \right) \hat{\Delta}\vec{v}_i, \tag{3.7}$$

on the condition that $\left(\Delta\vec{v} \cdot \vec{S} \right) \leq 0$ and zero otherwise. This condition needs expanding upon. Within the compatible framework, [13], forces are calculated around the median mesh of each node, and these forces are then used in place of a PdV energy update. As each median mesh segment of the cell is in contact with two nodal cells, to which it applies equal and opposite forces it is possible to re-write the compatible internal energy update, (2.51),

$$M \frac{De_i}{Dt} = -\sum_{i=1}^{4} \vec{f}_i' \cdot \vec{\Delta}v. \tag{3.8}$$

Here $\vec{f}_i' = P_z \vec{S}_i$ is the force associated with median mesh segment i, and $\vec{\Delta}v$ is the velocity difference along the cell edge to which the median mesh segment is connected so that,

$$\vec{\Delta}v_i = \vec{v}_i - \vec{v}_{i+1}. \tag{3.9}$$

The definition is cyclical around the cell so that,

$$\vec{\Delta}v_4 = \vec{v}_4 - \vec{v}_1. \tag{3.10}$$

As the compatible energy update is used in terms of a PdV work, what (3.8) yields is a compatible calculation of the change of volume of the cell,

$$dV = \sum_{i=1}^{4} \vec{S}_i \cdot \vec{\Delta}v_i dt. \tag{3.11}$$

This is the motivation for the dot product which triggers the shock viscosity of (3.7). The change in cell volume has been decomposed in terms of the four primary mesh cell edge contributions. It has already been stated that shock viscosity should act in areas of compression, and (3.11) has provided compression switches for each edge. Explicitly, when any of the four components of (3.11) are negative the edge in question is acting to reduce to volume of the cell and the viscosity along this edge should switch on, hence the concept of edge based viscosity. Thus numerically compression and expansion along an edge are defined.

Although this adequately defines a force, it is a force associated with an edge. In

fact it is illustrative to follow the method of [18] and visualise these forces as been associated with the triangles formed by the edges and cell midpoint. Each edge force acts as an equal and opposite force on each of it's nodes in order to oppose the compression of the edge.

The compatible methodology requires that forces are defined in terms of corner forces. In combining the edge based viscous force with the compatible method Caramana used the viscous heating requirement in conjunction with the fact that that $\vec{f} \sim -\hat{\Delta}\vec{v}$. Thus to ensure that the heating term is positive, the rate of viscous work was defined as $\vec{f}_{edge} \cdot -\vec{\Delta}v$ and is thus positive definite. In Odin, to bring edge viscosity fully in line with the compatible framework the viscous corner force is coded as:

$$\vec{f}_{visc}^{corner,i} = -\vec{f}_{visc}^{i-1} + \vec{f}_{visc}^{i}. \tag{3.12}$$

This has the same total effect as the original formulation, but is now explicitly defined in terms of corner force contributions. This force re-distribution is the reverse method used to decompose PdV work in terms of median mesh contributions, and has the effect that the heating from edge viscosity can be written in the form,

$$M\frac{De_i}{Dt}\bigg|_{visc} = -\sum_i \vec{f}_{visc}^{corner,i} \cdot \vec{v}^i. \tag{3.13}$$

3.2.3 Viscosity Limiters

In (3.7) ψ is used but not defined, it is the viscosity limiter. The purpose of the limiter function in one dimension is to act to turn off the viscosity where the second derivative of the velocity is zero, but without a direct calculation of the second derivative. This will fulfil the self similar motion requirement of the shock viscosity. In multiple dimensions Caramana showed that for a correct choice of viscosity limiter the third and fourth requirements outlined for shock viscosity are fulfilled. The limiter function for an edge, i, is defined as:

$$\psi_i = max\left[0, min\left(0.5\left(r_{l,i} + r_{r,i}\right), 2r_{l,i}, 2r_{r,i}, 1\right)\right], \tag{3.14}$$

where $r_{r,i}$ and $r_{l,i}$ are right and left velocity ratios, defined as:

$$r_{l,i} = \frac{\Delta\vec{v_{i+1}} \cdot \hat{\vec{\Delta v}}_i}{\Delta\vec{x_{i+1}} \cdot \hat{\vec{x}}_i} \Big/ \frac{|\Delta\vec{v_i}|}{|\Delta\vec{x_i}|}, \tag{3.15}$$

and,

$$r_{r,i} = \frac{\Delta \vec{v_{i-1}} \cdot \hat{\Delta \vec{v}_i}}{\Delta \vec{x_{i-1}} \cdot \hat{\vec{x}_i}} / \frac{|\Delta \vec{v_i}|}{|\Delta \vec{x_i}|}. \tag{3.16}$$

There are similar formulae for all edges. It is worth noting the suggestion from Caramana that should the effective edge CFL-like condition $\frac{|\Delta \vec{v_i}|}{|\Delta \vec{x_i}|}\Delta t$ be less than round off the both of the edge ratios should be set to unity, thus turning off the shock viscosity and reducing numerical noise.

3.3 Time step limiting in Conjunction with Shock Viscosity

The preceding section provides enough information to construct a shock viscosity and how to implement it within a finite volume compatible hydrodynamics scheme. However one subtle part, which is often overlooked remains unconsidered. As the shock viscosity alters the equations of the system it's clear that it should also alter the stable time step calculation of the system, however there has been little consensus over the exact manner of the alteration. All methods define a generalised sound speed, based upon the combination of thermodynamic pressure and an equation of state with (possibly part of) the shock viscosity. One such method [11]is:

$$C^2_{s,gen.} = C^2_{s,0} + \frac{2Q}{\rho}. \tag{3.17}$$

Where $C_{s,0}$ is the normal equation of state based sound speed, and Q is some scalar part of the shock viscosity. In the original paper outlining edge based viscosity Caramana mentioned the use of a generalised sound speed, but not it's exact form. However in a later paper [19] where a vorticity damping term is added to the original edge based viscosity the authors review the method, and although a simplified version of the edge viscosity is presented by comparison of that with (3.7) an equivalent generalised sound speed may be derived:

$$C_{s,gen.} = \left(C^2_{s,0} + C^2_{q,edge}\right)^{(1/2)}, \tag{3.18}$$

where,

$$C^2_{q,edge} = \frac{q_{kur}}{\rho}\left(1 - \psi\right). \tag{3.19}$$

This gives a generalised sound speed for each edge, and the most restrictive generalised sound speed for each cell is used to calculate a stable time step.
In fact, a number of other (published) options (e.g. [20])for controlling the time step

in the presence of shock viscosity exist, so it is useful to examine the problem more explicitly in the limiting case of cold (zero internal energy) shock compression. As a number of standard test cases for shock viscosity have the initial condition of zero internal energy, this is a relevant case to consider.

3.3.1 Cold Compression of a single cell

Consider a simplified test case on a uniform grid, density is set to unity everywhere and internal energy zero. As previously mentioned this is not dissimilar to a number of standard test cases. Without loss of generality we can assume the cell under consideration is undergoing compression due to the motion of just one of its edges. In the more general case the motion would be considered on an edge by edge basis, and similar to before the most restrictive Δt would be used. Let us further assume that the shock viscosity in use is that of (or similar to) the edge based viscosity previously defined. However it is useful to rewrite it as,

$$\vec{f} = -\frac{q_{Kur}}{|\Delta\vec{v}|}\left(1-\psi\right)\left(\vec{\Delta v}_i \cdot \vec{S}_i\right)\hat{\vec{\Delta v}}. \tag{3.20}$$

This can be written in a more simple form,

$$\vec{f} = -k\Delta\vec{v}, \tag{3.21}$$

where,

$$k = \frac{q_{Kur}}{|\Delta\vec{v}|}\left(1-\psi\right)\left(\Delta\vec{v}\cdot\vec{S}\right). \tag{3.22}$$

Assuming the edge being compressed is that defined by nodes 2 and 3, and using (3.8)the rate of work on the cell is given by,

$$W = k\left(\Delta\vec{v}_2\right)^2 = k\left(\vec{v_2}-\vec{v_3}\right)^2. \tag{3.23}$$

Clearly, this is positive definite. However, like in many other codes Odin only calculates the viscous force at the start of the time step, or the predictor level. So in the fully (both time and space) discretised situation the true rate of work is,

$$W = k\Delta\vec{v}_2^{\,n}\cdot\Delta\vec{v}_2^{\,n+1/2}. \tag{3.24}$$

This is positive if and only if,

$$SIGN\left(\Delta\vec{v}_2^{\,n}\right) = SIGN\left(\Delta\vec{v}_2^{\,n+1/2}\right). \tag{3.25}$$

In the limit of zero internal energy, we can rewrite the half time step velocity jump using the masses associated with the nodes in question, and the viscous forces present:

$$\Delta\vec{v_2^{n+1/2}} = \vec{v_2^n} - \frac{k\Delta t\Delta\vec{v_2^n}}{2M_2} - \vec{v_3^n} - \frac{k\Delta t\Delta\vec{v_2^n}}{2M_3}, \tag{3.26}$$

but in a uniform grid and density set up $M_2 = M_3 = \rho\Delta x\Delta y$ and (3.26) can be rewritten as,

$$\frac{\Delta v_2^{n+1/2}}{\Delta v_2^n} = \left(1 - \frac{k\Delta t}{M_2}\right). \tag{3.27}$$

In this final step the vector notation on the velocity jumps as been dropped. In a one dimensional case this step is trivial, in a multidimensional case a coordinate by coordinate sweep/combination would be necessary to carry out this time step consideration. Due to (3.25) the left hand side of (3.27) is required to be positive. k can be rewritten in a slightly different form:

$$k = \frac{q_{Kur}}{|\Delta v|}(1-\psi)\,|\vec{S}|\left(\Delta\hat{v}\cdot\hat{S}\right). \tag{3.28}$$

For this simple case the dot product evaluates to one. For other cases it may be smaller, but as shall be shown the case where it evaluates to unity provides the most stringent limit on the time step. Define,

$$k' = k\frac{\Delta x}{2}, \tag{3.29}$$

and enforcing the right hand side of (3.27) to be positive yields

$$\Delta t < \frac{2\rho\Delta y}{k'}, \tag{3.30}$$

This provides a constraint on Δt based on the requirement of viscous heating in a predictor corrector scheme. In the limit of zero energy it is easy to compare (3.17) and (3.30). In order to ensure the viscous heating is positive, (3.17) shows the requirement is,

$$\frac{2\rho|\Delta\vec{v}|}{q_{Kur}(1-\psi)} > \left(\frac{\rho}{2q_{Kur}(1-\psi)}\right)^{1/2}. \tag{3.31}$$

In the limit of zero internal energy q_{Kur} as the simple form,

$$q_{Kur} = \rho\,|\Delta\vec{v}|^2\left(\frac{\gamma+1}{2}\right)^{1/2}|\Delta\vec{v}|\,. \tag{3.32}$$

The requirement that (3.17) is more stringent than the viscous heating requirement is,

$$\frac{2\rho\left|\Delta v\right|}{q_{Kur}\left(1-\psi\right)} > \left(\frac{\rho}{2q_{Kur}\left(1-\psi\right)}\right)^{1/2}, \tag{3.33}$$

or simplifying,

$$\left|\Delta\vec{v}\right| > \frac{1}{2\sqrt{2}}\left(\frac{q_{Kur}\left(1-\psi\right)}{\rho}\right)^{1/2}. \tag{3.34}$$

Given the simplified form of q_{Kur} in the zero internal energy limit, this reduces to,

$$\left|\Delta\vec{v}\right| > \left|\Delta\vec{v}\right|\frac{1}{2\sqrt{2}}\left(\frac{\gamma+1}{2}\right)^{1/2}\left(1-\psi\right), \tag{3.35}$$

or,

$$1 > \frac{1}{2\sqrt{2}}\left(\frac{\gamma+1}{2}\right)^{1/2}\left(1-\psi\right). \tag{3.36}$$

Clearly (as $\gamma \leq 3$) (3.17) fulfils the viscous heating requirement.
This calculation lead to (3.17) being implemented in Odin with all types of shock viscosity, despite it being contrary to the original authors of edge based viscosity. The above example is only that, it is not a stability calculation, but a limitation on edge viscosity based on the ideal characteristics of shock viscosity as previously quoted. It's extension to two dimensions and a non-uniform grid is possible and gives a similar result. In practice Odin uses a CFL number of 0.75, and takes the most restrictive combination of grid spacing and viscosity magnitudes.

3.4 Edge Viscosity Results

3.4.1 Sod's Shock Tube Problem

A very simple test of a shock viscosity is Sod's shock tube problem [21], the initial conditions for which are given by,

$$\begin{aligned}(\rho, e_i) &= \begin{cases}(1, 2.5) & \text{if } x < 0.5\\ (0.125, 2) & \text{if } x > 0.5\end{cases}\\ \vec{v} &= 0.0\end{aligned} \tag{3.37}$$

With an ideal gas equation of state, $\gamma = 1.4$ this corresponds to a pressure ratio of 10 between the left and righthand states. The problem was run with $n_x = 100$, until t=0.15.

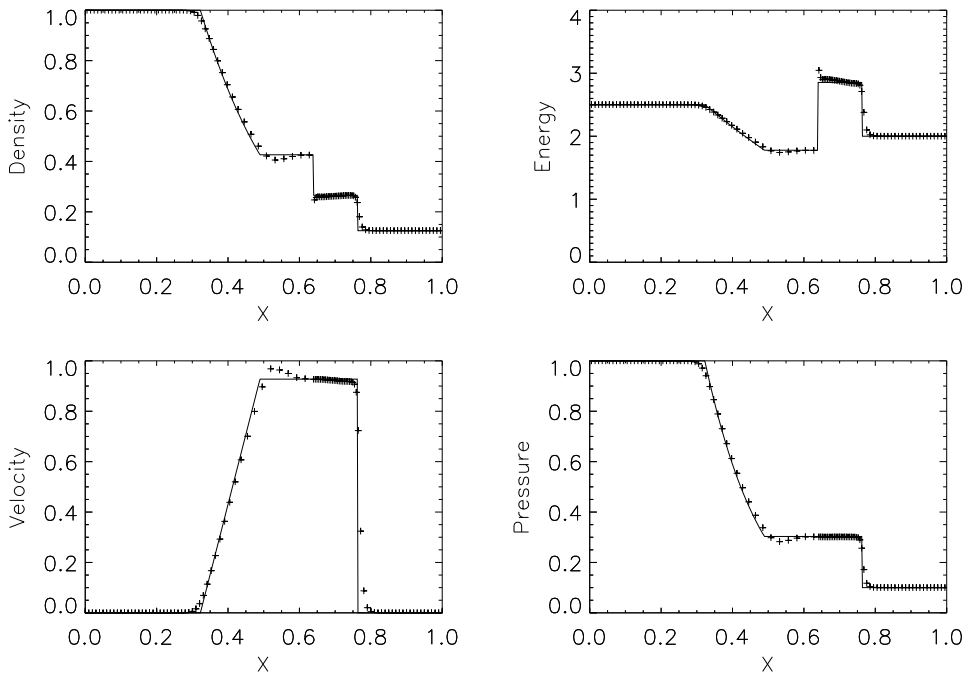


Figure 3.1: Results obtained for Sod's shock tube problem using edge based viscosity. The analytical solution is shown as a solid line, whilst the results obtained are shown as crosses.

3.4.2 Saltzman's Piston Problem

Saltzman's piston problem [22] is a one dimensional problem, run on a two dimensional grid. The problem consists of a piston moving in the positive x-direction with a speed of 1.0 with respect to the gas, compressing an initially cold (zero internal energy), unit density ideal gas, with $\gamma = 5.0/3.0$. The initial grid is perturbed by a sinusoidal perturbation, and is shown in figure 3.2. The problem was run until

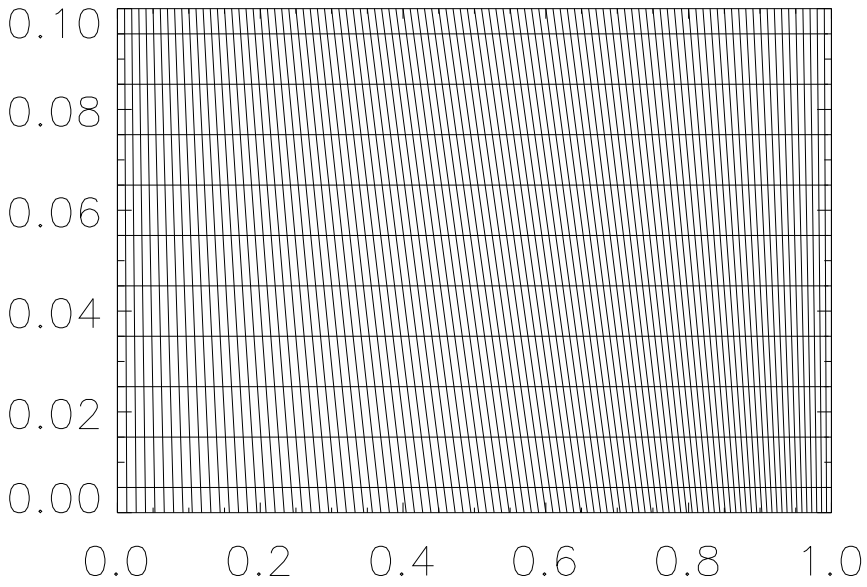


Figure 3.2: Initial grid for Saltzman's piston problem

t=0.8 in pure Lagrangian mode, at which point the shock wave launched by the pistons motion has bounced off the far wall. The analytic solution is a density of 4.0 in the singly shocked region, and 10.0 in the twice shocked region. Any deviations from a one dimensional solution show effects of the grid on the core solver. The grid for Saltzman's piston problem at $t = 0.8$ is shown in figure 3.3. The density is shown by both a contour plot, figure 3.4 and a scatter plot, figure 3.5. Some loss of symmetry is apparent in the contour plot, and examining the scatter plot highlights both a wall heating error near the piston, as well as significant overshoot at the shock front. The grid is also beginning to tangle, a problem which causes the time-step to collapse shortly after $t = 0.8$.

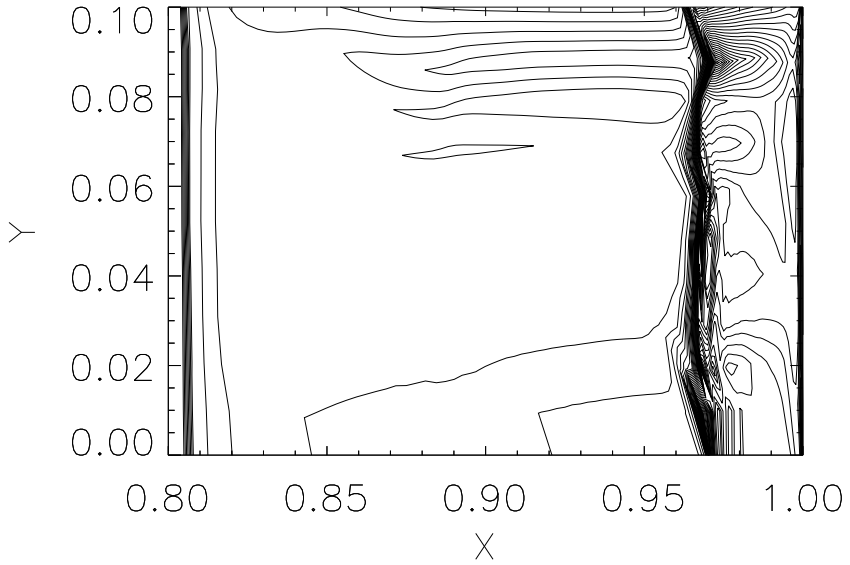


Figure 3.3: Density contour plot for Saltzman's piston problem at t=0.8 using edge viscosity.

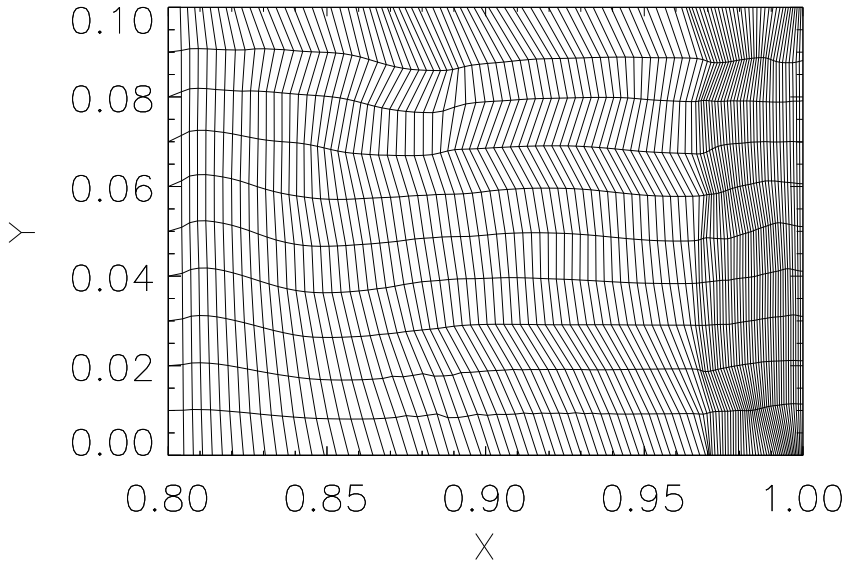


Figure 3.4: Grid for Saltzman's piston problem at t=0.8 using edge viscosity.

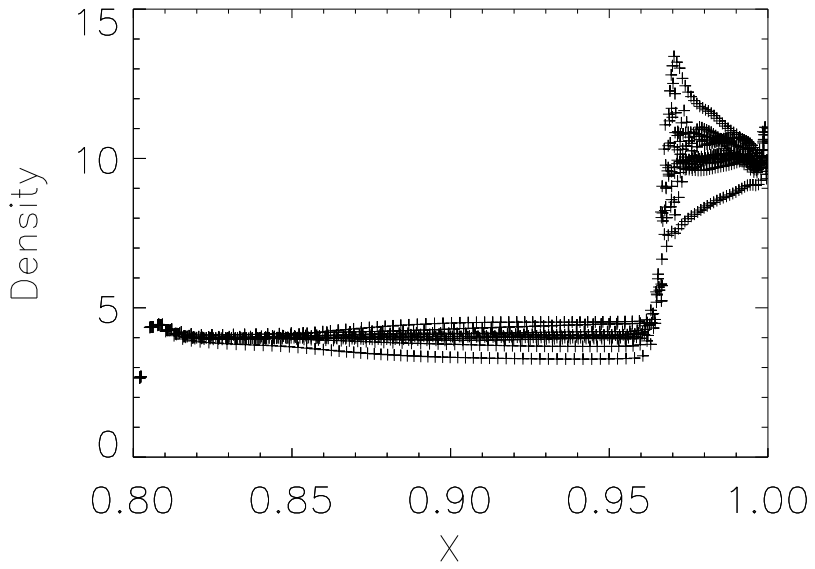


Figure 3.5: Density scatter plot for Saltzman's piston problem at t=0.8 using edge viscosity.

3.4.3 Noh's Problem

Noh's problem [23] in Cartesian coordinates consists of a cold ($e_i = 0$) cylinder of unit density, and initial velocity field of magnitude one, pointed inwards radially everywhere. At $t = 0$ a shock wave forms at the origin and moves outwards. The problem has a simple analytic solution and is a good test of a codes ability to handle spherical shock waves. The problem was run on two different grids. The first is a polar grid, with angular spacing of $\Delta\theta = \pi/20$, and radial spacing of $\Delta r = 0.02$. At $t = 0.6$ the shock should have reached $r = 0.2$.

As can be seen from figure 3.6 for a polar grid the edge based viscosity is successful at obtaining the analytic solution. The undershoot of the density at the origin is caused by the well known wall heating problem. The shock front is found to be in the correct position, and only spread out across 2/3 zones.

The same problem was rerun on a Cartesian mesh, with $\Delta x = \Delta y = 0.02$, again using edge based viscosity. Some loss of symmetry is apparent in figure 3.7, with jet like structures appearing along the axis. This is coupled with the grid distortion along both axes apparent in figure 3.8.

The loss of symmetry is most apparent when considering figure 3.9, as well as some overshoot at the shock front.

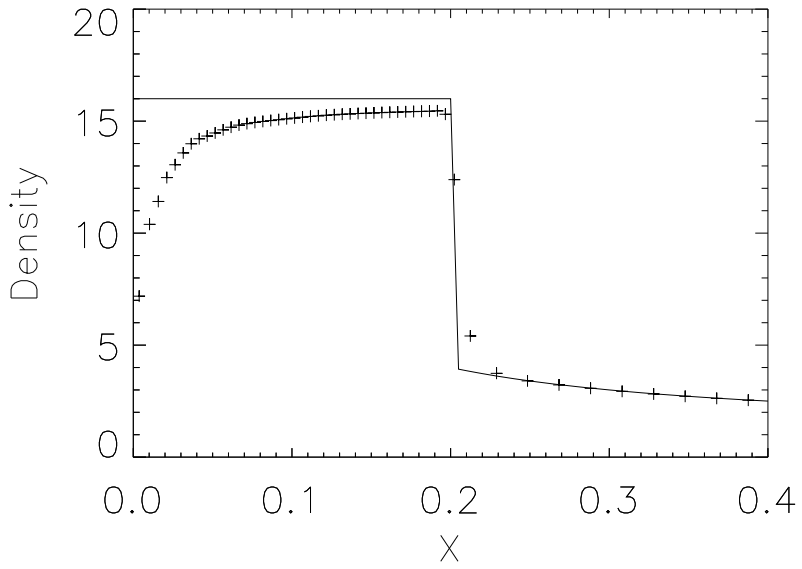


Figure 3.6: Results for Noh's problem run on a polar grid with edge based viscosity. Analytic solution is shown as a solid line, and results marked with crosses.

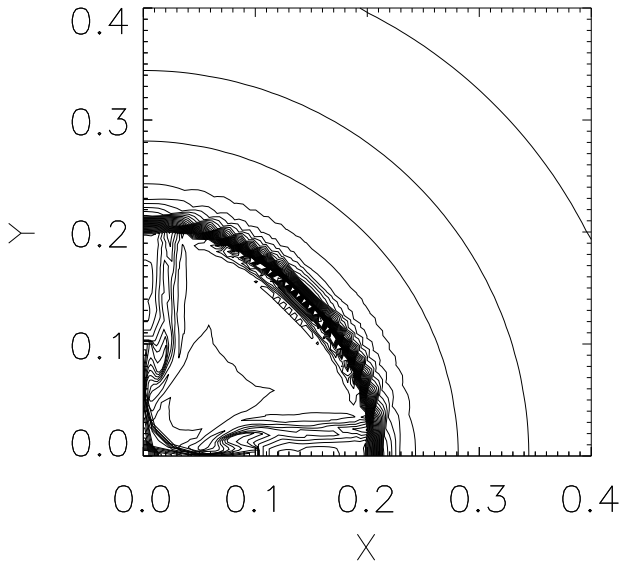


Figure 3.7: Density contour plot for Noh's problem run on a Cartesian grid with edge based shock viscosity.

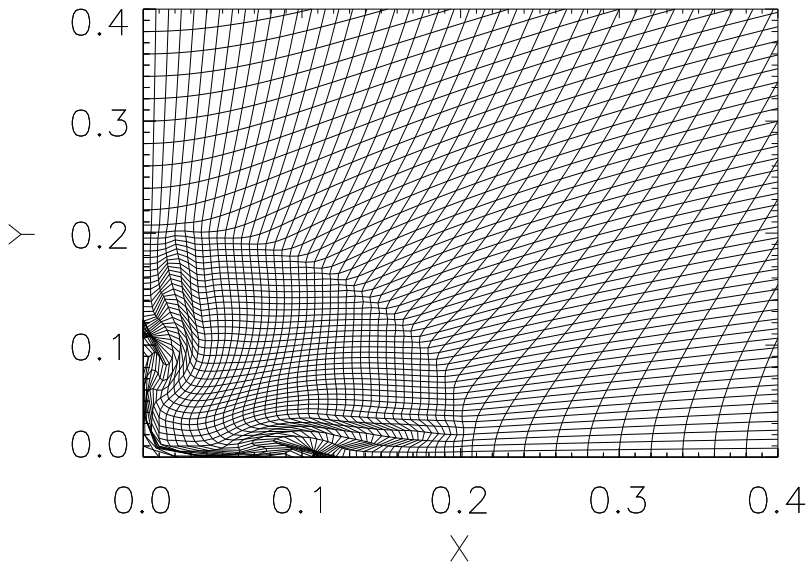


Figure 3.8: Grid for Noh's problem run with edge based shock viscosity on an initially Cartesian grid.

The results of these tests of the edge based viscosity show that whilst the method is capable of capturing shocks across a small number of zones it has problems when the grid is not aligned with with flow, and when run with a problem with significant grid distortion appears (especially in the case of Saltzman's piston problem) to increase the grid distortion, which has obvious problems for Lagrangian simulations, as such an improved shock viscosity was sought.

3.5 Tensor Shock Viscosity

The edge viscosity developed by Caramana et al. whilst not a scalar pressure term is not a true tensor viscosity. Whilst it is considered to be the force resulting from a postulated tensor, that tensor is never properly defined in a continuous sense. Taking the example of Noh's problem, on a polar grid the edge viscosity produces good results, however when using a Cartesian grid, large scale jets occur along the axes. This is a result of the edge viscosities dependence on the grid, due to the method of development. The force was developed with conditions, and implicitly a grid, in mind. Campbell and Shashkov first put this observation forward, and sought to improve on edge based results by devising a proper tensor viscosity, whilst

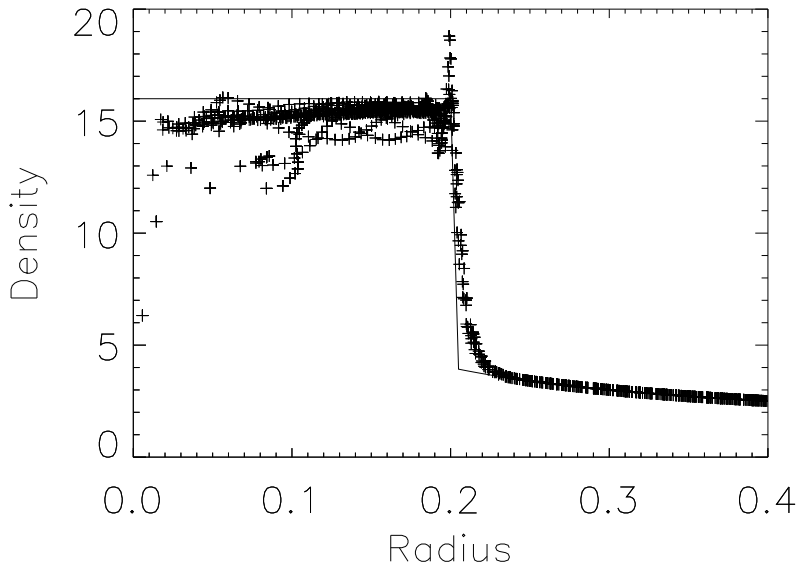


Figure 3.9: Scatter plot of density against radius for Noh's problem run with edge based shock viscosity.

keeping the same previously outlined qualities as edge viscosity. This is the second viscosity available in Odin .

3.5.1 Continuous Form of Shock Viscosity

Campbell and Shashkov [24] define a continuous form of shock viscosity in terms of the gradient of the velocity tensor, $\boldsymbol{G} = \nabla\vec{v}$ and a scalar coefficient, μ;

$$\boldsymbol{Q} = \mu\boldsymbol{G}^T, \tag{3.38}$$

where $G_{ij} = \partial v_i/\partial x_j$. Inclusion of a viscous stress tensor, $\boldsymbol{\tau}$ in the equations of motion for a compressible fluid (Navier-Stokes, neglecting heat conduction) yields,

$$\rho\frac{Du_i}{Dt} = -\frac{\partial}{\partial x_j}\left[P\delta_{ij} + \tau_{ji}\right], \tag{3.39}$$

and,

$$\rho\frac{De}{Dt} = -P\nabla\cdot\vec{u} + \frac{\partial}{\partial x_j}\left[u_i\tau_{ij}\right]. \tag{3.40}$$

Using the viscous tensor given by (3.38) and neglecting pressure yields,

$$\rho \frac{D\vec{u}}{Dt} = \nabla \cdot \boldsymbol{Q}^T = \nabla \cdot (\mu \boldsymbol{G}), \tag{3.41}$$

and,

$$\rho \frac{De}{Dt} = Q_{ij} G_{ji} = \mu G_{ij} G_{ij}. \tag{3.42}$$

Briefly reviewing the requirements of shock viscosity, consideration of (3.42) shows that to fulfil the viscous heating requirement, $\mu \geq 0$. Galilean invariance requires that μ is Galilean invariant, due to the tensor forms already fulfilling this requirement.

The viscous force continuity requirement is two-fold. Firstly μ must be equal to zero in expansion. Previously edge viscosity used a switch of the form $\vec{S}_i \cdot \vec{\Delta} v_i$ as a compression switch, but for the tensor viscosity an obvious choice is the divergence of the velocity. The edge viscosity uses the fact that this switch multiplies all other terms in order to fulfil the requirement that the force goes to zero smoothly. In order to fulfil this requirement using μ, would require the loss of the linear viscosity term. However with a quadratic only viscosity oscillations form behind the shock wave. As such viscous force continuity is not fully satisfied in order to remove these oscillations.

Self similar motion invariance is satisfied by viscosity limiters in an analogous way to edge viscosity, although they are defined in a slightly different manner, as explained in subsection 3.5.4. Finally, wavefront invariance is inherited by the tensor nature of the viscosity. Considering the case for two dimensional cylindrical coordinates,

$$\rho a_r = \frac{\partial Q_{rr}}{\partial r} + \frac{1}{r}\frac{\partial Q_{\theta r}}{\partial \theta} + \frac{1}{r}\left(Q_{rr} - Q_{\theta\theta}\right), \tag{3.43}$$

and,

$$\rho a_\theta = \frac{\partial Q_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial Q_{\theta\theta}}{\partial \theta} + \frac{1}{r}\left(Q_{r\theta} - Q_{\theta r}\right), \tag{3.44}$$

where a_r and a_θ are accelerations in the radial and angular direction respectively. For purely radial flow, $v_\theta = 0$ and $\partial v_r / \partial \theta = 0$, the only non-zero components of $\boldsymbol{Q}$ are the diagonal elements,

$$Q_{rr} = \mu \frac{\partial v_r}{\partial r}, \tag{3.45}$$

and,

$$Q_{\theta\theta} = \mu \frac{v_r}{r}. \tag{3.46}$$

As such, the two acceleration components are,

$$\rho a_r = \frac{\partial}{\partial r}\left(\mu \frac{\partial v_r}{\partial r}\right) + \frac{\mu}{r}\left(\frac{\partial v_r}{\partial r} - \frac{v_r}{r}\right), \tag{3.47}$$

and,

$$\rho a_\theta = 0, \tag{3.48}$$

as required.

3.5.2 Tensor Viscosity in a general Curvilinear system

In order to derive a discrete form of the continuous tensor viscosity it is necessary to consider the case in a general curvilinear system. The reason for this is that as the two connecting edges of a cell can never be parallel, and are thus linearly independent, they always represent the basis vectors of a curvilinear system. By considering the tensor viscosity in such a system, the continuous form of a proper tensor viscosity in the frame of reference of the cell is developed in terms of the (known) local Cartesian velocity field.

Using two edges as basis vectors the gradient of the velocity tensor is rewritten in terms of two vectors, formed by taking the dot product of the tensor with the unit basis vector. These two basis vectors in a general coordinate system,(ξ, η) are defined as,

$$\vec{e}_\xi = \begin{pmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{pmatrix}, \tag{3.49}$$

and,

$$\vec{e}_\eta = \begin{pmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \end{pmatrix}. \tag{3.50}$$

In a general coordinate system a (rank 2) tensor may be represented by the two vectors formed by contracting the tensor with each of the two basis vectors. These resulting vectors are given by,

$$G_i^\xi = G_{i,j}\hat{e}_{\xi,j}, \tag{3.51}$$

and,

$$G_i^\eta = G_{i,j}\hat{e}_{\eta,j}. \tag{3.52}$$

Components of the metric tensor are constructed as the relevant dot products of basis vectors,

$$g_{\xi\xi} = \vec{e}_\xi \cdot \vec{e}_\xi, \quad g_{\eta\eta} = \vec{e}_\eta \cdot \vec{e}_\eta, \quad g_{\eta\xi} = g_{\xi\eta} = \vec{e}_\xi \cdot \vec{e}_\eta. \tag{3.53}$$

It will be useful to calculate the determinant of the metric tensor,

$$\begin{aligned}
|g| &= g_{\xi\xi}g_{\eta\eta} - g_{\xi\eta}g_{\eta\xi} \\
&= \left[\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2\right]\left[\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2\right] \\
&\quad - \left[\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta}\right]^2 \\
&= \left(\frac{\partial x}{\partial \xi}\right)^2\left(\frac{\partial y}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2\left(\frac{\partial x}{\partial \eta}\right)^2 - 2\left(\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta}\right)
\end{aligned} \tag{3.54}$$

Rewriting the vectors defined by (3.51) and (3.52),

$$G^{\xi} = \frac{1}{\sqrt{g_{\xi\xi}}}\begin{pmatrix}\frac{\partial v_x}{\partial \xi}\\ \frac{\partial v_y}{\partial \xi}\end{pmatrix} = \begin{pmatrix}G^{\xi x}\\ G^{\xi y}\end{pmatrix}, \tag{3.55}$$

and,

$$G^{\eta} = \frac{1}{\sqrt{g_{\eta\eta}}}\begin{pmatrix}\frac{\partial v_x}{\partial \eta}\\ \frac{\partial v_y}{\partial \eta}\end{pmatrix} = \begin{pmatrix}G^{\eta x}\\ G^{\eta y}\end{pmatrix}. \tag{3.56}$$

Combining and reordering the two systems represented by (3.55) and (3.56) yield the full system,

$$\begin{pmatrix}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & 0 & 0\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & 0 & 0\\ 0 & 0 & \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi}\\ 0 & 0 & \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta}\end{pmatrix}\begin{pmatrix}\frac{\partial v_x}{\partial x}\\ \frac{\partial v_x}{\partial y}\\ \frac{\partial v_y}{\partial x}\\ \frac{\partial v_y}{\partial y}\end{pmatrix} = \begin{pmatrix}\sqrt{g_{\xi\xi}}G^{\xi x}\\ \sqrt{g_{\eta\eta}}G^{\eta x}\\ \sqrt{g_{\xi\xi}}G^{\xi y}\\ \sqrt{g_{\eta\eta}}G^{\eta y}\end{pmatrix} \tag{3.57}$$

The matrix on the left hand side of (3.57) can be block inverted. By noting that

$$\left[\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\right]^2 = |g|, \tag{3.58}$$

where the term in the square brackets in (3.58) is the determinant used in block inverting the matrix in (3.57), the following expressions for the Cartesian components of $\boldsymbol{G}$ are obtained in terms of the projections:

$$G_{xx} = \frac{\sqrt{g_{\xi\xi}}}{\sqrt{|g|}}\frac{\partial y}{\partial \eta}G^{\xi x} = \frac{\sqrt{g_{\eta\eta}}}{\sqrt{|g|}}\frac{\partial y}{\partial \xi}G^{\eta x}, \tag{3.59}$$

$$G_{xy} = \frac{\sqrt{g_{\eta\eta}}}{\sqrt{|g|}}\frac{\partial x}{\partial \xi}G^{\eta x} = \frac{\sqrt{g_{\xi\xi}}}{\sqrt{|g|}}\frac{\partial x}{\partial \eta}G^{\xi x}, \tag{3.60}$$

$$G_{yx} = \frac{\sqrt{g_{\xi\xi}}}{\sqrt{|g|}}\frac{\partial y}{\partial \eta}G^{\xi y} = \frac{\sqrt{g_{\eta\eta}}}{\sqrt{|g|}}\frac{\partial y}{\partial \xi}G^{\eta y}, \tag{3.61}$$

and,

$$G_{yy} = \frac{\sqrt{g_{\eta\eta}}}{\sqrt{|g|}}\frac{\partial x}{\partial \xi}G^{\eta y} = \frac{\sqrt{g_{\xi\xi}}}{\sqrt{|g|}}\frac{\partial x}{\partial \eta}G^{\xi y}. \tag{3.62}$$

It is now possible to derive a form for the divergence of the tensor in terms of these projections, using the following identity,

$$\int_V \nabla\vec{u} : \boldsymbol{T}\, dv = \oint_{\partial V} \vec{u}\cdot(\boldsymbol{T}\cdot\vec{n})\, ds - \int_V \vec{u}\cdot(\nabla\cdot\boldsymbol{T})\, dv, \tag{3.63}$$

where $\boldsymbol{G} : \boldsymbol{T}$ represents the contraction $G_{ij}T_{ij}$ and the concept of a generalised dot product from the previous chapter where $\boldsymbol{T}\cdot\vec{n}$ representing the contraction $T_{ij}n_j$ has been retained. Using (3.59) - (3.62) to rewrite the contraction on the right hand side of (3.63) gives,

$$\boldsymbol{G} : \boldsymbol{T} = G^{\xi}\cdot\left[T^{\xi}\frac{g_{\xi\xi}g_{\eta\eta}}{|g|} - T^{\eta}\frac{g_{\xi\eta}}{|g|}\sqrt{g_{\xi\xi}}\sqrt{g_{\eta\eta}}\right] + G^{\eta}\cdot\left[T^{\eta}\frac{g_{\xi\xi}g_{\eta\eta}}{|g|} - T^{\xi}\frac{g_{\xi\eta}}{|g|}\sqrt{g_{\xi\xi}}\sqrt{g_{\eta\eta}}\right]. \tag{3.64}$$

Finally using,

$$\frac{g_{\xi\xi}g_{\eta\eta}}{|g|} = \frac{1}{sin^2\theta}, \quad \frac{g_{\xi\eta}}{\sqrt{g_{\xi\xi}}\sqrt{g_{\eta\eta}}} = cos\theta, \tag{3.65}$$

where θ is the angle between the two edges used to construct the basis vectors. As such the left hand side of (3.63) can be written,

$$\begin{aligned}&\int\int \nabla\vec{u} : \boldsymbol{T}\sqrt{|g|}d\xi d\eta \\ &\quad = \int\int\left(\frac{\partial\vec{u}}{\partial\xi}\cdot\left[\frac{T^{\xi} - cos\theta T^{\eta}}{\sqrt{g_{\xi\xi}}sin^2\theta}\right] + \frac{\partial\vec{u}}{\partial\eta}\cdot\left[\frac{T^{\eta} - cos\theta T^{\xi}}{\sqrt{g_{\eta\eta}}sin^2\theta}\right]\right)\sqrt{|g|}d\xi d\eta.\end{aligned} \tag{3.66}$$

Integrating by parts, and neglecting the term that is equal to the surface integral in (3.63) yields,

$$\begin{aligned}&\int\int \vec{u}\cdot(\nabla\cdot\boldsymbol{T})\sqrt{|g|}d\xi d\eta \\ &\quad = \int\int\left(\vec{u}\cdot\frac{\partial}{\partial\xi}\left[\sqrt{|g|}\frac{T^{\xi} - cos\theta T^{\eta}}{\sqrt{g_{\xi\xi}}sin^2\theta}\right] + \vec{u}\cdot\frac{\partial}{\partial\eta}\left[\sqrt{|g|}\frac{T^{\eta} - cos\theta T^{\xi}}{\sqrt{g_{\eta\eta}}sin^2\theta}\right]\right)d\xi d\eta.\end{aligned} \tag{3.67}$$

The final expression for the divergence of a tensor is a general curvilinear system is,

$$\nabla \cdot \boldsymbol{T} = \frac{\partial}{\partial \xi}\left[\sqrt{|g|}\frac{T^{\xi} - cos\theta T^{\eta}}{\sqrt{g_{\xi\xi}}sin^2\theta}\right] + \frac{\partial}{\partial \eta}\left[\sqrt{|g|}\frac{T^{\eta} - cos\theta T^{\xi}}{\sqrt{g_{\eta\eta}}sin^2\theta}\right]. \tag{3.68}$$

3.5.3 Discrete form of Tensor Viscosity

The vectors defined by (3.55) and (3.56) in discrete form are calculated as projections of the tensor $\nabla\vec{v}$ on cell edges, so that the projection onto edge $p + 1/2$ is,

$$G^e_{p+1/2} = \frac{\vec{v}_{p+1} - \vec{v}_p}{l_{p+1/2}}, \tag{3.69}$$

where $l_{p+1/2}$ is the length of the cell edge. The support operator method [25],[26] first defines a prime operator and derives other operators from it. Shashkov and Campbell use the vector gradient as the prime operator and derive from it the divergence of a tensor. To do this a discrete form of (3.63) is used, and "for simplicity" the surface integral term is assumed to equal zero. This point will be returned to. This means that the divergence is the negative adjoint of the gradient,

$$(div) = -(grad)*, \tag{3.70}$$

or writing (3.63) in discrete form neglecting the surface integral terms,

$$\sum_z (\boldsymbol{G}, \boldsymbol{T})_z V_z = -\sum_p \left(\vec{v}, DI\vec{V}\boldsymbol{T}\right)_p V_p, \tag{3.71}$$

where (,) represents a scalar product. The discrete form of the tensor product on the left hand side of (3.71), defined in it's continuous form by (3.64) is,

$$\begin{aligned}(\boldsymbol{G}, \boldsymbol{T})_z = \sum_{p\in S(z)} \frac{W^z_p}{sin^2\theta^z_p}\Big(&G^e_{p-1/2}\cdot T^e_{p-1/2} + G^e_{p+1/2}\cdot T^e_{p+1/2} \\ &+cos\theta^z_p\left[G^e_{p-1/2}\cdot T^e_{p+1/2} + G^e_{p+1/2}\cdot T^e_{p-1/2}\right]\Big).\end{aligned} \tag{3.72}$$

The sign of the cosine has changed due to the convention adopted by Shashkov and Campbell [24]. The terms W^z_p are weights satisfying the following conditions,

$$W^z_p \geq 0, \quad \sum_{p\in S(z)} W^z_p = 1. \tag{3.73}$$

Campbell and Shashkov [24] defined the weights as half of the area of the triangle containing the angle θ^z_p divided by the volume of the zone. Inserting (3.72) into

(3.71) and collecting terms of $\vec{v}_p$ gives,

$$
\begin{aligned}
&-\sum_p \left(\vec{v}, DI\vec{V}\boldsymbol{T}\right)_p V_p = \\
&\sum_z \sum_{p\in S(z)} \vec{v}_p \cdot \left(\frac{W_p^z}{sin^2\theta_p^z} \left\{ \frac{T^e_{p-1/2}}{l_{p-1/2}} - \frac{T^e_{p+1/2}}{l_{p+1/2}} + cos\theta_p^z \left[\frac{T^e_{p+1/2}}{l_{p-1/2}} - \frac{T^e_{p+1/2}}{l_{p+1/2}} \right] \right\} \right. \\
&\left. \frac{W_{p-1}^z}{sin^2\theta_{p-1}^z} \left\{ \frac{T^e_{p-1/2}}{l_{p-1/2}} + cos\theta_{p-1}^z \frac{T^e_{p-3/2}}{l_{p-1/2}} \right\} - \frac{W_{p+1}^z}{sin^2\theta_{p+1}^z} \left\{ \frac{T^e_{p+1/2}}{l_{p+1/2}} + cos\theta_{p+1}^z \frac{T^e_{p+3/2}}{l_{p+1/2}} \right\} \right) V_z.
\end{aligned}
\tag{3.74}
$$

This means that the expression for the divergence of a tensor at a point is,

$$
\begin{aligned}
(DIV\boldsymbol{T})_p =& \frac{1}{V_p} \sum_{z\in S(p)} V_z \left[\frac{1}{l_{p+1/2}} \left\{ \frac{W_z^{p+1}}{\sin^2\theta_z^{p+1}} \left(T^e_{p+1/2} + \cos\theta_z^{p+1} T^e_{p+3/2} \right) \right\} \right. \\
&+ \frac{1}{l_{p+1/2}} \left\{ \frac{W_z^p}{\sin\theta_z^p} \left(T^e_{p+1/2} + \cos\theta_z^p T^e_{p-1/2} \right) \right\} \\
&- \frac{1}{l_{p-1/2}} \left\{ \frac{W_z^p}{\sin^2\theta_z^p} \left(T^e_{p-1/2} + \cos\theta_z^p T^e_{p+1/2} \right) \right\} \\
&\left. - \frac{1}{l_{p-1/2}} \left\{ \frac{W_z^{p-1}}{\sin^2\theta_z^{p-1}} \left(T^e_{p-1/2} + \cos\theta_z^{p-1} T^e_{p-3/2} \right) \right\} \right].
\end{aligned}
\tag{3.75}
$$

Returning to the assumption of the surface integral in (3.67) equalling zero it is worth nothing that the discrete forms of (3.41) and (3.42) are,

$$
m_p \frac{D\vec{v}_p}{Dt} = V_p \left(DIV\boldsymbol{Q}\right)_p = \sum_{z\in S(p)} \vec{f}_z^p, \tag{3.76}
$$

and

$$
M_z \frac{De_z}{Dt} = V_z \left(\boldsymbol{Q}, \boldsymbol{G}\right)_z = - \sum_{p\in S(z)} \vec{f}_p^z \cdot \vec{v}_p. \tag{3.77}
$$

Clearly (3.71) (and the association assumption that the surface integral is zero) is in fact a statement of energy conservation in the discrete system.

The final step is to define the scalar coefficient of the tensor viscosity. In order to satisfy dissipativity (see also, subsection 4.4.1) the scalar coefficient must be constant in a corner volume. Analogously to the edge viscosity the Kurapatenko pressure jump is used as a basis, and after dimensional considerations is multiplied by a length scale specific to the corner volume. For the velocity jump the divergence of the velocity is calculated for each subzone, and multiplied by a characteristic length

scale. These two length scales are equal to,

$$l_z^p = min(l_{p+1/2}, l_{p-1/2}), \tag{3.78}$$

the minimum of the lengths of the two cell edges connected to the subzone.
The choice of the velocity divergence as the basis for the velocity jump provides a compression switch for each viscous corner force, and has the added advantage that it will also act to suppress hourglass modes (see also, chapter 5), which change the volume of the subzone, but not the zone itself.

3.5.4 Velocity Limiters for Tensor Viscosity

The natural choice for constructing the limiters for tensor viscosity would be the subzonal volume changes calculated for the viscosity coefficients. This, as pointed out by Caramana and Shashkov [24] will give incorrect results, due to the fact that three of the four velocities used to calculate this quantity are interpolated. This error is most obvious when considering the one dimensional case, where a shock is moving aligned with the grid. Corner based limiters would turn off the viscosity for corners on the upstream side of the zone as a shock wave passes through, which is incorrect. Instead limiters based on the change of volume associated with each edge are used. This volume is defined by four points, the two nodes of the edge being considered, and the centre points of the two adjacent zones. For each edge, four ratios are constructed, the two previous ratios left and right, and two new ones, up and down, where each direction is easily defined on a logical grid.

$$r_{l,k} = \frac{(\nabla \cdot \vec{v})_l}{(\nabla \cdot \vec{v})_k}, \quad r_{r,k} = \frac{(\nabla \cdot \vec{v})_r}{(\nabla \cdot \vec{v})_k} \tag{3.79}$$

$$r_{u,k} = \frac{(\nabla \cdot \vec{v})_u}{(\nabla \cdot \vec{v})_k}, \quad r_{d,k} = \frac{(\nabla \cdot \vec{v})_d}{(\nabla \cdot \vec{v})_k}. \tag{3.80}$$

These four ratios give rise to two different limiter functions for each edge, and the minimum of these is taken to be the final edge limiter function,

$$\psi_k = min(\psi_1, \psi_2), \tag{3.81}$$

$$\psi_1 = max\left[0, min\left(0.5\left(r_{l,k} + r_{r,k}\right), 2r_{l,k}, 2r_{r,k}, 1\right)\right] \tag{3.82}$$

$$\psi_2 = max\left[0, min\left(0.5\left(r_{u,k} + r_{d,k}\right), 2r_{u,k}, 2r_{d,k}, 1\right)\right] \tag{3.83}$$

A limiter function for the subzone is then constructed, being the minimum of the two edge limiter functions connected to the subzone, so that

$$\psi_z^p = min\left(\psi_{p+1/2}, \psi_{p-1/2}\right). \tag{3.84}$$

3.5.5 Final Form of Tensor Shock Viscosity

All that remains is to combine the previous subsections, to obtain the following corner force representation of the tensor shock viscosity,

$$\begin{aligned}\vec{f}_z^p = V_z \Bigg[& \frac{1}{l_{p+1/2}} \left(\frac{W_z^{p+1}}{\sin^2\theta_z^{p+1}} \left(\mu_z^{p+1} \vec{G}^e_{p+1/2} + \cos\theta_z^{p+1} \mu_z^{p+1} \vec{G}^e_{p+3/2} \right) \right) \\ & + \frac{1}{l_{p+1/2}} \left(\frac{W_z^{p}}{\sin^2\theta_z^{p}} \left(\mu_z^{p} \vec{G}^e_{p+1/2} + \cos\theta_z^{p} \mu_z^{p} \vec{G}^e_{p-1/2} \right) \right) \\ & - \frac{1}{l_{p-1/2}} \left(\frac{W_z^{p}}{\sin^2\theta_z^{p}} \left(\mu_z^{p} \vec{G}^e_{p-1/2} + \cos\theta_z^{p} \mu_z^{p} \vec{G}^e_{p+1/2} \right) \right) \\ & - \frac{1}{l_{p-1/2}} \left(\frac{W_z^{p-1}}{\sin^2\theta_z^{p-1}} \left(\mu_z^{p-1} \vec{G}^e_{p-1/2} + \cos\theta_z^{p-1} \mu_z^{p-1} \vec{G}^e_{p-3/2} \right) \right) \Bigg]. \end{aligned} \tag{3.85}$$

To summarise here $l_{p-1/2}$ is the length of side $p-1/2$, θ_z^p is the interior angle of zone z between edges $p-1/2$ and $p+1/2$. The weighting functions, W_z^p is defined as half the area of the triangle containing θ_z^p divided by the cross sectional area of zone z. The vectors, $\vec{G}^e_{p+1/2}$ are defined by (3.69). Finally the scalars, μ_z^p are defined as,

$$\mu_z^p = q^p_{kur,z} l_z^p \left(1 - \psi_z^p\right), \tag{3.86}$$

where $q^p_{kur,z}$ is the Kurapatenko pressure jump for the subzone, calculated by using $|\nabla \cdot \vec{v}|\, l_z^p$ as the subzonal velocity jump. The divergence of the velocity vector is calculated using the divergence theorem. The length scale is defined using (3.78), and the limiter functions, ψ_z^p are defined in the previous subsection.

3.6 Tensor Shock Viscosity Results

3.6.1 Sod's Shock Tube Problem

Sod's shock tube problem was rerun with the tensor based viscosity. It is now worth examining and comparing the two types of viscosity presented in the limiting case of a one dimensional problem. For the edge viscosity the force in a one dimensional problem on the node i is,

$$f_{edge} = q_{kur}\left(1-\psi\right)\left(v_{i+1}-v_i\right)\Delta y. \tag{3.87}$$

The force due to tensor shock viscosity is given by,

$$f_{tensor} = \frac{V_z}{l_{p+1/2}}\left(\frac{v_{i+1}-v_i}{l_{p+1/2}}\left(W_z^{p+1}\mu_z^{p+1}+W_z^p\mu_z^p\right)\right). \tag{3.88}$$

For the one dimensional case, the weights, W_z^p are equal for fixed z, and given by,

$$W_z^p = \frac{l_{p+1/2}l_{p-1/2}}{2V_z}. \tag{3.89}$$

A second simplification can be made by noting that for a one dimensional problem the scalar coefficients are equal for all subzones within a cell, and given by,

$$\mu_z^p = q_{kur}\left(1-\psi\right)l_{min}, \tag{3.90}$$

where l_{min} is the minimum of the two edge lengths of the cell. The tensor force is therefore given by,

$$f_{tensor} = q_{kur}\left(1-\psi\right)l_{min}\frac{\Delta y}{\Delta x}. \tag{3.91}$$

As discussed in [24] for a one dimensional case the limiter functions for each type of viscosity are identical, thus the forces due to the two types of viscosity should have the same form as long as $l_{min} = \Delta x$. This condition should hold for all cells undergoing compression for Sod's problem, as long as the cells initially have aspect ratio of unity. However the two methods differ in their calculation of q_{kur}. Whilst the tensor viscosity calculates q_{kur} individually for each subzone, using subzonal density, and zonal sound speed, the edge viscosity calculates it for each edge. This is done by combining values for each point on the edge, which in turn uses values from each of the four cells the node is attached to. Thus the edge viscosity contains additional averaging which will change the magnitude of the force slightly. Comparing figure 3.1 and figure 3.10 there are small differences between the two sets of results, due to

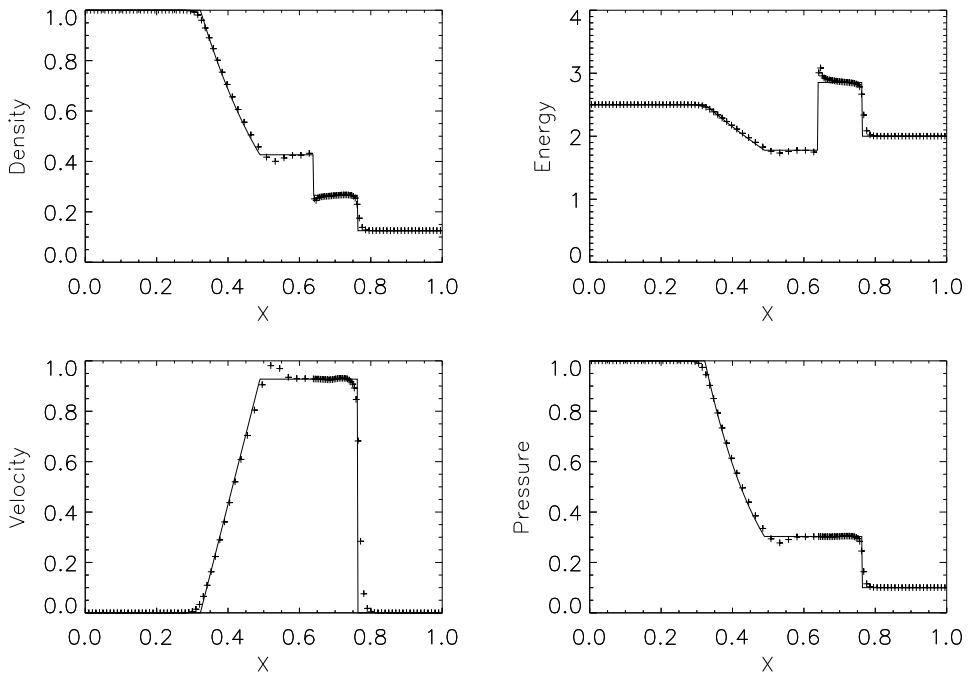


Figure 3.10: Results obtained for Sod's shock tube problem using tensor shock viscosity. The analytical solution is shown as a solid line, whilst the results obtained are shown as crosses.

the averaging procedure used by the edge viscosity.

3.6.2 Saltzman's Piston Problem

Saltzman's piston problem was also rerun using tensor based shock viscosity, with the same original grid as for the edge based viscosity. Although the solution is one dimensional, the grid is not, so this test represents a good chance to compare how the grid will impact on the solution when using different shock viscosities. As

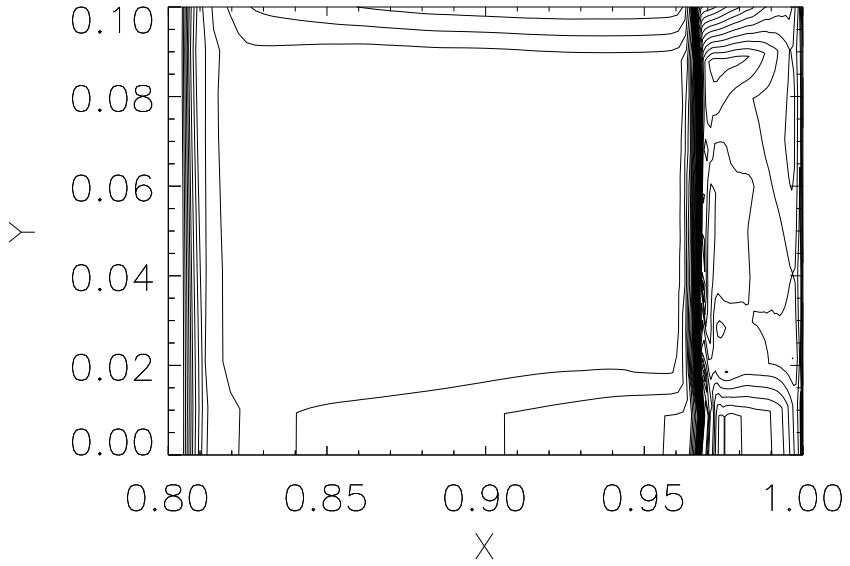


Figure 3.11: Density contour plot for Saltzman's piston problem at t=0.8 using tensor shock viscosity.

figure 3.11, figure 3.12 and figure 3.13 show the tensor shock viscosity maintains the one dimensional aspect of the solution much more than the edge based shock viscosity. There is also less grid distortion present. This enables the tensor viscosity to run to later times than the edge based viscosity. Both schemes suffer from similar amounts of the wall heating problem, this is to be expected as this is a one dimensional problem, occurring at a point where the grid is still well aligned with the flow.

3.6.3 Noh's Problem

As with the edge viscosity Noh's problem [23] was run with the tensor shock viscosity on both a polar grid and a Cartesian grid. The resolutions remained the same as

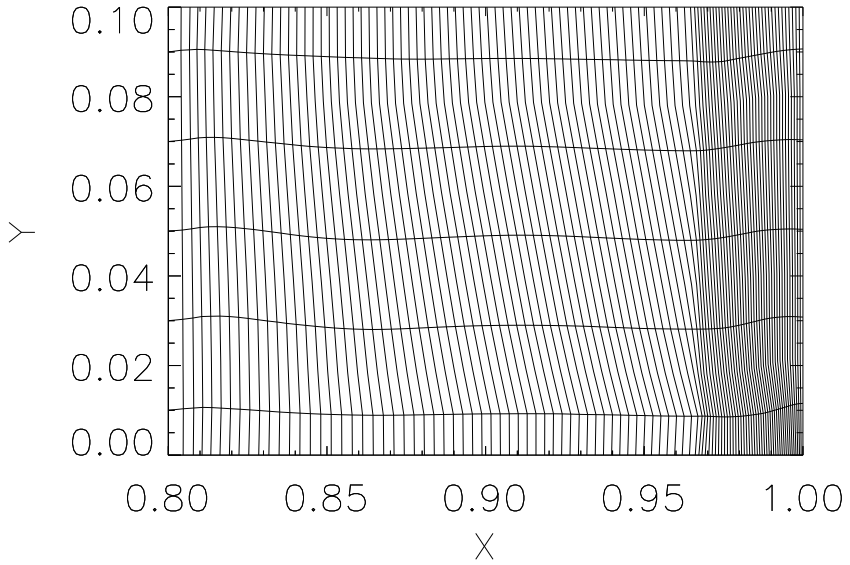


Figure 3.12: Grid for Saltzman's piston problem at t=0.8 using tensor shock viscosity.

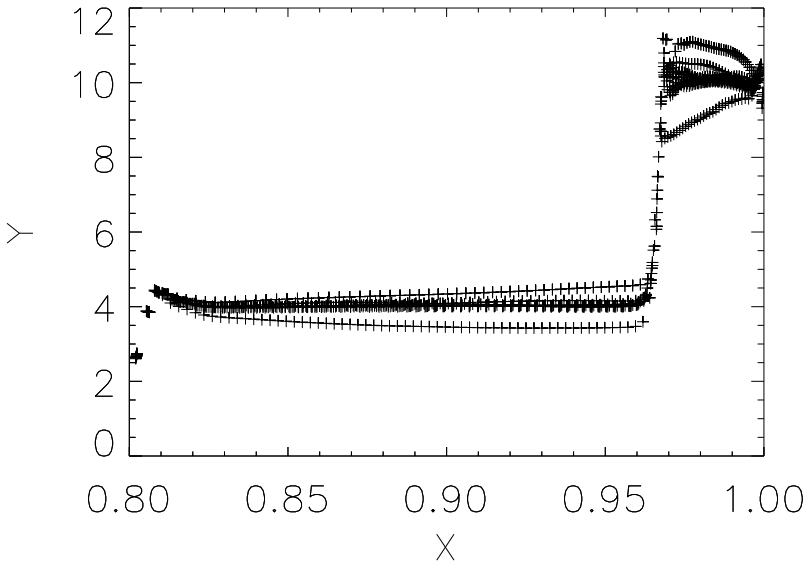


Figure 3.13: Density scatter plot for Saltzman's piston problem at t=0.8 using tensor shock viscosity.

the calculation for edge viscosity on a polar grid. When compared to the edge based

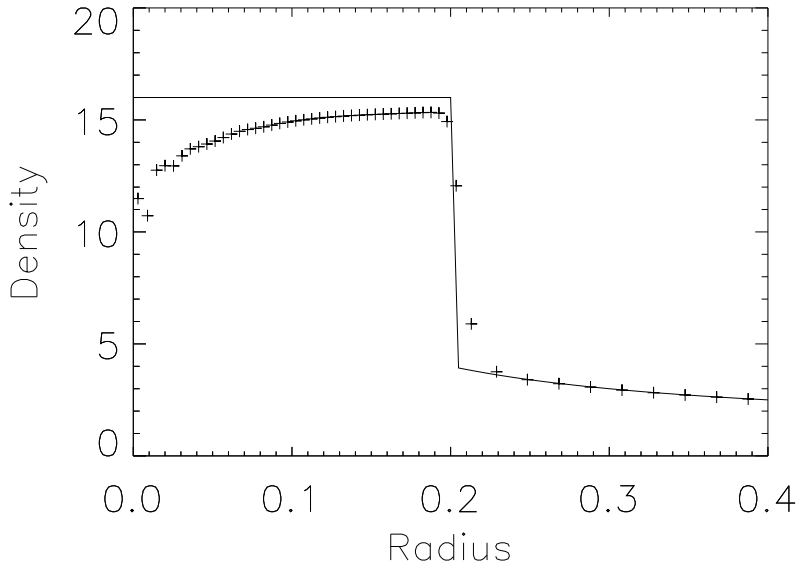


Figure 3.14: Results for Noh's problem run on a polar grid with tensor viscosity. Analytic solution is shown as a solid line, and results marked with crosses.

viscosity the tensor viscosity results are very similar. There is a reduction in the error due to wall heating, but at the same time a slight ripple is observed in the solution near the origin. The viscosities are now compared on the Cartesian grid.

Comparing contour plots of the edge based viscosity and tensor viscosity the first point to note is some apparent structure along the $\pi/4$ line that is not apparent when using edge based viscosity. However some reduction in the magnitude of the features on the axes is apparent in the case of tensor viscosity. Considering now the final grids for the two viscosities, tensor shock viscosity is the clear winner. The grid tangling seen along the axes for the edge viscosity is almost completely avoided, with only some slight tangling near the origin.

Some loss of symmetry is still apparent on the scatter plot for Noh's problem with the tensor shock viscosity, however the density is much more tightly packed around the true solution, and the overshoot at the shock front has been significantly reduced.

The results of these tests was that tensor viscosity is the preferred viscosity within Odin, although both methods are implemented in the current version. Unless stated otherwise the remaining plots in this thesis will use tensor shock viscosity.

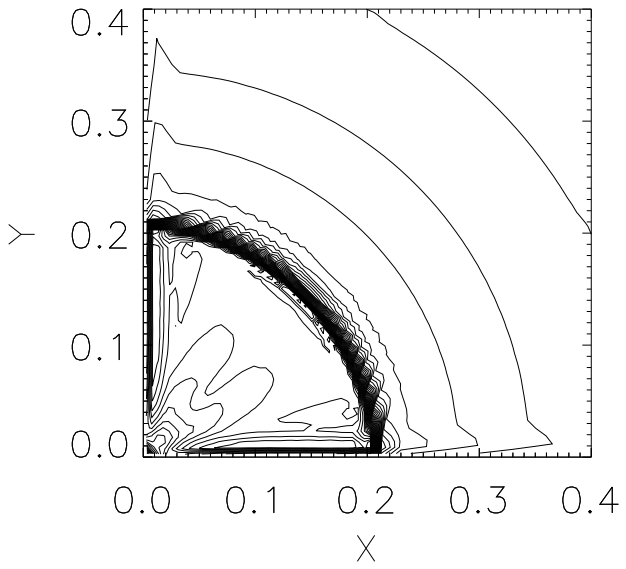


Figure 3.15: Density contour plot for Noh's problem run on a Cartesian grid with tensor shock viscosity.

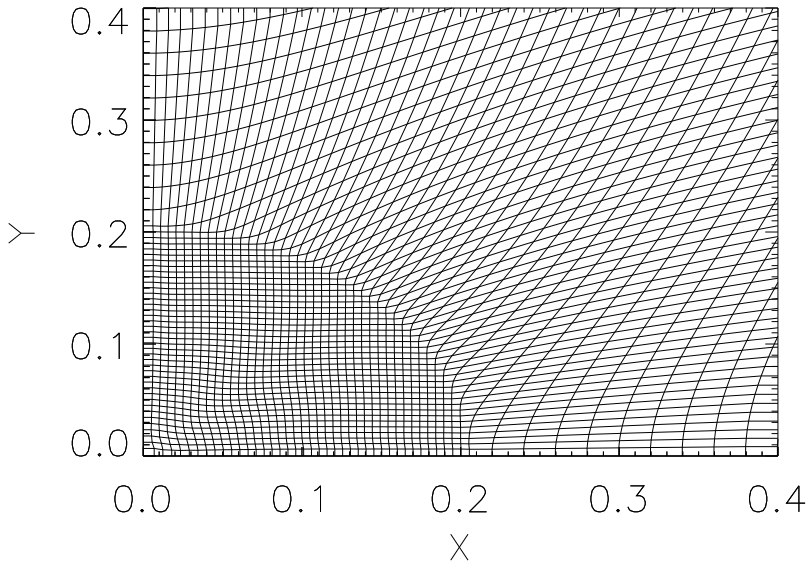


Figure 3.16: Grid for Noh's problem run with tensor shock viscosity on an initially Cartesian grid.

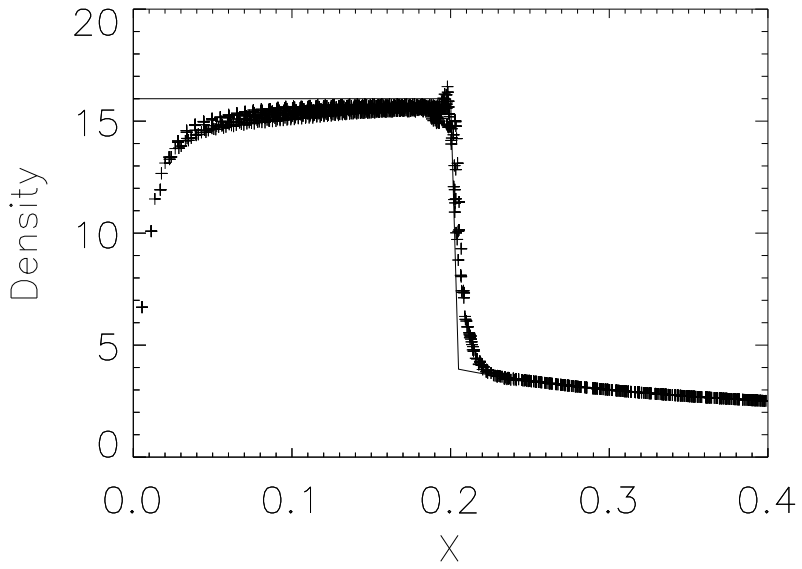


Figure 3.17: Scatter plot of density against radius for Noh's problem run with tensor shock viscosity.

3.6.4 Sedov Blast Problem

A final test problem to introduce is the Sedov blast wave problem [27]. In its normal form, the Sedov problem consists of a uniform density stationary fluid, with zero initial energy, except for the cell nearest the origin which has a high energy value, $e = 409.7$ for Cartesian coordinates. However running this test problem on a Cartesian mesh results in grid points piling up along the axis, and the time step collapsing. This failure is illustrated by the grid in fig 3.18. However, it is possible to run the problem with tensor shock viscosity on a polar mesh. In this variant of the problem the inner annulus of cells are injected with energy, such that the total energy in the initial conditions is the same so that the blast wave reaches $r = 1.0$ at $t = 1.0$. The results are shown in figure 3.19. Clearly it is desirable that such a problem could also be run on a Cartesian grid. It is possible, but only by splitting the hot cell at the centre into four cells. In order to run the original problem it is necessary to either carry out a remap, or to introduce some other strategy to mitigate grid motion, such as subzonal pressures, which will be described in a later chapter, and used to successfully run Sedov's problem on a Cartesian mesh.

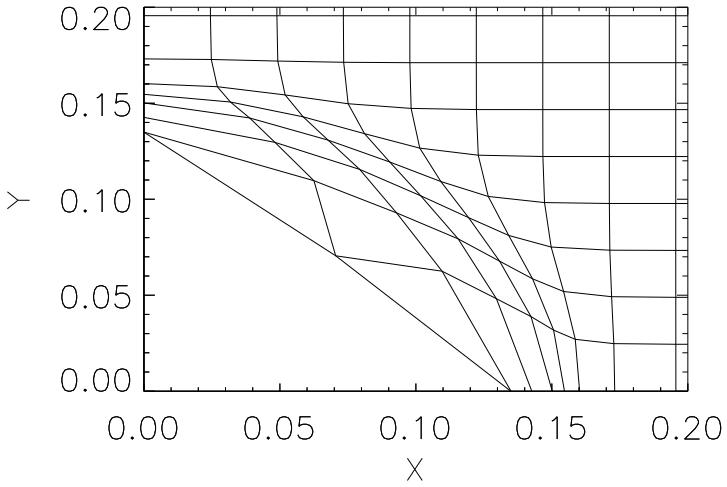


Figure 3.18: Grid resulting from Sedov's problem on a Cartesian grid.

3.7 Summary

Two types of shock viscosity have been described and implemented within Odin. Whilst the edge based shock viscosity was successful at modelling shocks aligned with the flow, difficulties occurred for problems where either the grid has been perturbed from being aligned with the shock such as in Saltzman's piston problem, or in cases where the geometry of the mesh does not match the shock such as Noh's problem on a Cartesian mesh. The second variant of shock viscosity, tensor shock viscosity was found to be an improved method for such cases, whilst not keeping the quality of solution seen for edge based shock viscosity on simpler problems. For this reason, it is referred to as being less grid dependent than the edge viscosity. However as seen for the case of Sedov's problem on a Cartesian mesh even the tensor viscosity can still run into difficulties, and as such the need for either a strategy to mitigate grid collapse, or an implementation of a remap phase has been made clear. Both of these options shall be explored in future chapters.

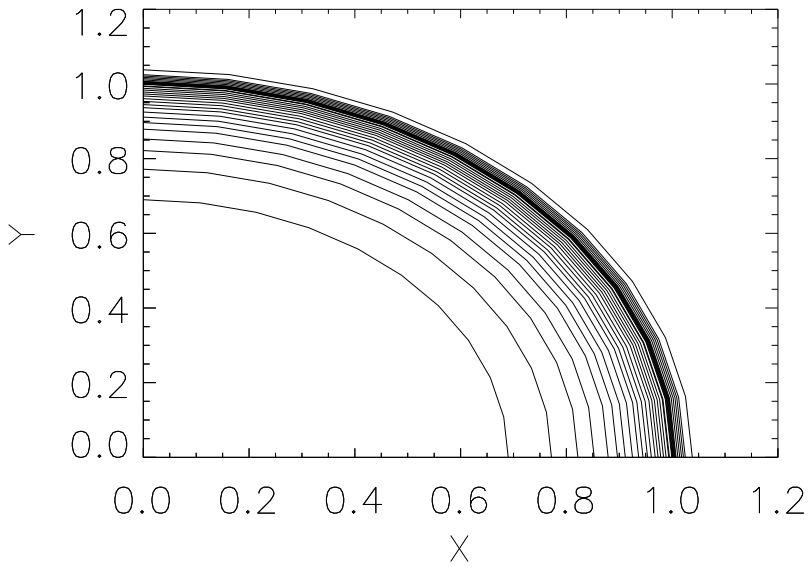


Figure 3.19: Density contour plot for Sedov's problem run on a polar grid with tensor shock viscosity.

Chapter 4

Cylindrical Coordinates

4.1 Introduction

The previous chapters have implicitly assumed a two dimensional Cartesian geometry. Whilst first conceived as a cylindrical (rz) code, Odin was originally written in Cartesian coordinates, before the necessary changes were made to enable it to run in either xy, or rz coordinates. There are in fact two (main) options for constructing a two dimensional cylindrical finite volume code, area weighted differencing, (AWD), or control volume differencing (CVD), each with their own advantages and disadvantages.

4.2 Control Volume Differencing

Control volume differencing, or volume weighted differencing (VWD), is the most natural scheme to adopt. The forces, as with the Cartesian case, are calculated by multiplying outward normal vectors, of magnitude equal to the surface area of the face, with the pressure; essentially this means multiplying the Cartesian forces by the average radial coordinate of the face. Volumes and masses are calculated in the normal way for cylindrical coordinates, the product of the cell area, the radial coordinate and the density, essentially the Cartesian volume has been multiplied by the average radial coordinate of the cell or subcell. The true physical masses and volumes are calculated by multiplying this value by 2π (or some other $d\phi$), but this is not included in the code, as it always cancels, and as such is dropped from following discussions for the sake of brevity.

4.2.1 Cylindrical Stability in CVD

Whilst the above description is the most natural and physically motivated choice of discretisations it fails to properly mimic the equations. This is clearly shown by considering a cylindrical collapse, with the grid aligned with the flow, as shown in 4.1.

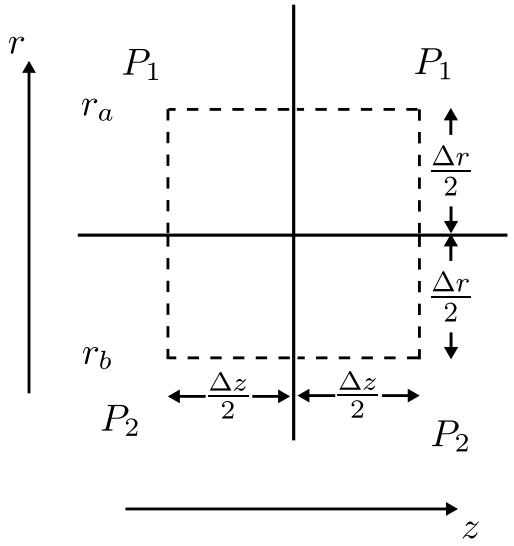


Figure 4.1: Set up for cylindrical Collapse

Denoting the velocity vector (u, v) the correct momentum update equations are,

$$\rho \frac{Du}{Dt} = -\frac{\partial P}{\partial r}, \quad \rho \frac{Dv}{Dt} = -\frac{\partial P}{\partial z}. \tag{4.1}$$

It is possible now to integrate directly (4.1) to obtain the forces in the radial and z-directions. Denoting $R = (r_a + r_b)/2$ the centre of the (nodal) cell and considering

only the radial force only,

$$\begin{aligned}
f_r &= -\int_{\Omega(t)} \frac{\partial P}{\partial r} dV \\
&= -\int_{\Omega(t)} \frac{\partial P}{\partial r} r d\phi dr dz \\
&= -2\pi \Delta z \int_{r_b}^{r_a} \frac{\partial P}{\partial r} r dr \\
&= -2\pi \Delta z \int_{r_b}^{r_a} \left(\frac{\partial}{\partial r} (rP) - P \right) dr \\
&= -2\pi \Delta z \left([rP]_{r_b}^{r_a} - \int_{r_b}^{r_a} P dr \right) \\
&= -2\pi \Delta z \left(r_a P_1 - r_b P_2 \right) - 2\pi \Delta z \int_{r_b}^{r_a} P dr \\
&= -2\pi \Delta z \left[\left(R + \frac{\Delta r}{2} \right) P_1 - \left(R - \frac{\Delta r}{2} \right) P_2 \right] - 2\pi \Delta z \int_{r_b}^{r_a} P dr \\
&= -2\pi \Delta z \left[R \left(P_1 - P_2 \right) + \frac{\Delta r}{2} \left(P_1 + P_2 \right) \right] - 2\pi \Delta z \int_{r_b}^{r_a} P dr. \qquad (4.2)
\end{aligned}$$

The final step is to evaluate the (piecewise constant) second integral to give,

$$2\pi \Delta z \int_{r_b}^{r_a} P dr = 2\pi \Delta z \left(P_1 + P_2 \right) \frac{\Delta r}{2}. \qquad (4.3)$$

Where in line with previous discussions the pressure in a cell is taken to be constant as only the average pressure is known. This cancels with the second term of (4.2) and gives the final force as:

$$f_r = -2\pi \Delta z R \left(P_1 - P_2 \right), \qquad (4.4)$$

This force is not equal to $\int_S P\vec{n}dS$ around the (nodal) cell boundary. To correctly calculate the force by integrating over the volume of the cell one must use

$$\begin{aligned}\int_V \left[\nabla \cdot (P\vec{a})\right] dV &= \int_V \left[P\left(\nabla \cdot \vec{a}\right) + \left(\vec{a} \cdot \nabla\right) P\right] dV \\ &= \int_S \left(P\vec{a}\right) \cdot \vec{n}dS. \end{aligned} \tag{4.5}$$

Rearranging,

$$\int_V \left(\vec{a} \cdot \nabla\right) PdV = \int_S P\vec{a} \cdot \vec{n}dS - \int_V P\nabla \cdot \vec{a}dV. \tag{4.6}$$

Setting $\vec{a}$ as the unit vector in a coordinate direction gives the force component in that direction. For the Cartesian case the second term on the right hand side is zero. However for cylindrical coordinates,

$$\nabla \cdot \hat{r} = \frac{1}{r}. \tag{4.7}$$

Hence,

$$M_p \frac{Du_r}{Dt} = -\int_S P\hat{r} \cdot \vec{n}dS + \int_V \frac{P}{r} dV. \tag{4.8}$$

Both (4.4) and (4.8) are correct volume averages of the equations.

4.2.2 Symmetry Preservation in CVD

In their paper on symmetry preservation in cylindrical coordinates Caramana and Whalen [28] use a different implementation of control volume differencing. Whilst this paper was focused on polar grids rather than general grids it is still worth consideration. In this implementation of volume weighted differencing the primary grid lines were used to calculate the pressure gradient forces, rather than the median mesh lines that actually form the boundary of the region in question.

This is simply a reversal of the previous compatible formulation. Previously it has been assumed that the forces would be calculated, and these used to define a compatible energy update. This scheme acts in reverse, defining forces associated with the energy update, and using these to update momentum (and thus, kinetic energy) in a compatible manner. In Cartesian coordinates of course, the two options are exactly equivalent, indeed, in any code they would be numerically equivalent. However in cylindrical coordinates this is not the case, due the to radial weighting of the areas in question.

For the limiting case of an orthogonal grid the scheme described by Caramana and Whalen [28] is actually equivalent to an area weighted scheme. This makes it a

good candidate for implementation of cylindrical coordinates in fixed grid codes. For an arbitrary grid however the two schemes are different. The volume weighted scheme described in [28] suffers from a loss of symmetry for spherical collapse in cylindrical coordinates due to the differing magnitudes of forces on neighbouring grid elements, due to different radial weightings. Whilst it may be possible to carefully derive the P/r term in such a way that this symmetry loss is reduced, or even entirely alleviated, the area weighted scheme does not require such additional considerations. It should be pointed out however, that in Cartesian coordinates when considering the collapse of a cylinder with uneven grid spacing a similar loss of symmetry occurs, and as such this error is inherited by the area weighted scheme in cylindrical coordinates.

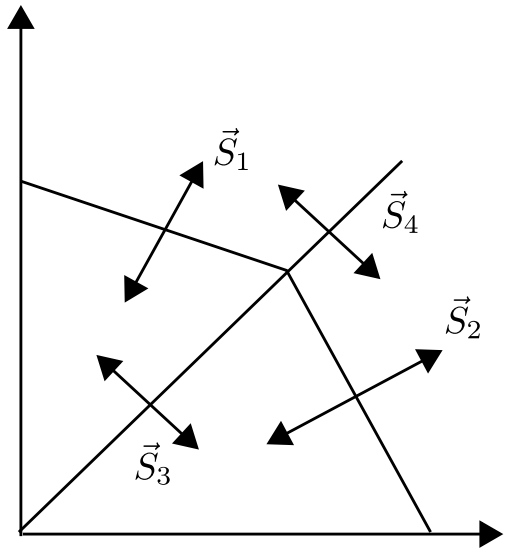


Figure 4.2: The vectors used to calculate CVD forces by Caramana et al.

The mainstay of [28] is a modification of the gradient operator that accounts for, and eliminates this problem. Essentially this method looks for some preferred directions in the volume weighting calculated forces, and removes the non-radial (in a spherical sense) component. However, whilst this allows cylindrical collapse in xy with an unequal angular zoning, and spherical collapse in rz with an equal angular zoning, it does not remedy the problem for unequal angular zoning in rz, the solution does not

work for an arbitrary grid. Secondly, it essentially tries to correct for an unnecessary problem. The method is looking for a symmetric solution, thus assuming some *a priori* knowledge of the solution, yet it is only useful when this a priori knowledge has been ignored in the setting up of the simulation. For these reasons, it was not included in the final version of Odin .

4.3 Area Weighted Differencing

As should be clear from the preceding discussion, whilst control volume differencing is the most natural physical choice it is not without its problems. Thus an alternative has been suggested and used in various forms, area weighted differencing [29]. The motivation for this is that the momentum equation for rz looks identical to the momentum equation for xy, with the variables changed appropriately. It is possible to rewrite (4.4) as,

$$f_r = r f_x, \tag{4.9}$$

where f_r is the radial component of the force, and f_x is the equivalent force calculated in the x-direction in Cartesian coordinates. As previously mentioned the continuous equations look identical in both coordinate systems, so it is desirable to write the mass as,

$$M_{node} = r A_{node} \rho, \tag{4.10}$$

as by doing this it enables the radial factors on either side of the equation to cancel, resulting in an identical numerical scheme to the Cartesian case, where $M = \rho A$. In both cases A is some cross sectional area associated with the node. It is necessary that the sum of the nodal masses and the cell centred masses sum to give the same total, as such each must be derived from the subzonal masses, where each subzonal mass contributes to one cell-centred mass and one nodal mass, thus enforcing the two totals to be the same.
The subzonal masses are calculated at $t = 0$ as

$$m_i^z = \rho_{cell} v_i^z, \tag{4.11}$$

where v_i^z is the i-th subzonal volume of the cell, z. In cylindrical coordinates the mass of a subzone is therefore,

$$m_i^z = \rho_{cell} a_i^z |r_i^z| 2\pi, \tag{4.12}$$

where here a_i^z is the cross sectional area of the subzone, and $|r_i^z|$ is the (average) radial coordinate of the subzone. However, as previously mentioned it is desirable to write the mass of a subzone as being proportional to the radial coordinate of its node, thus it is necessary to redistribute the subzonal masses within a cell. Combining the four subzones of a cell to give the total cell centred mass yields,

$$\begin{aligned} M_{cell} = \rho 2\pi \Big[& \frac{a_1}{4} \left(r_1 + 0.5(r_1 + r_2) + 0.5(r_1 + r_4) + r_z\right) \\ & + \frac{a_2}{4} \left(r_2 + 0.5(r_2 + r_3) + 0.5(r_2 + r_1) + r_z\right) \\ & + \frac{a_3}{4} \left(r_3 + 0.5(r_3 + r_4) + 0.5(r_3 + r_2) + r_z\right) \\ & + \frac{a_4}{4} \left(r_4 + 0.5(r_4 + r_1) + 0.5(r_4 + r_3) + r_z\right) \Big] . \end{aligned} \tag{4.13}$$

This form only holds at $t = 0$, after which the densities within the different subzones may no longer be equal. Here r_z is the radial coordinate of the cell given as,

$$r_z = \frac{1}{4} \left(r_1 + r_2 + r_3 + r_4\right) . \tag{4.14}$$

It is useful to note that the radial coordinate of the i-th subzone can be written as,

$$r_i^z = \frac{1}{16} \left(9r_i + 3r_{i+1} + 3r_{i-1} + r_{i-2}\right) , \tag{4.15}$$

where the nodal indexing is defined in the same cyclic manner as used in previous sections. Using this the sum of a cell's contributions to its nodal masses can be written as,

$$\begin{aligned} M_{cell} = \rho_{cell} 2\pi \Big[& \frac{a_1}{16} \left(9r_1 + 3(r_2 + r_4) + r_3\right) \\ & + \frac{a_2}{16} \left(9r_2 + 3(r_1 + r_3) + r_4\right) \\ & + \frac{a_3}{16} \left(9r_3 + 3(r_2 + r_4) + r_1\right) \\ & + \frac{a_4}{16} \left(9r_4 + 3(r_3 + r_1) + r_2\right) \Big] . \end{aligned} \tag{4.16}$$

Considering figure 4.3 and (4.16) the area weighted nodal contribution from the cell of volume $V = a_1|r_a| + a_2|r_b| + a_3|r_c| + a_4|r_d|$ is given by,

$$m_1^z = \rho_{cell} 2\pi \frac{1}{16} |r_1| \left(9a_1 + 3\left(a_2 + a_4\right) + a_3\right) . \tag{4.17}$$

Returning to the simple case of the cylinder, but now calculating the forces by

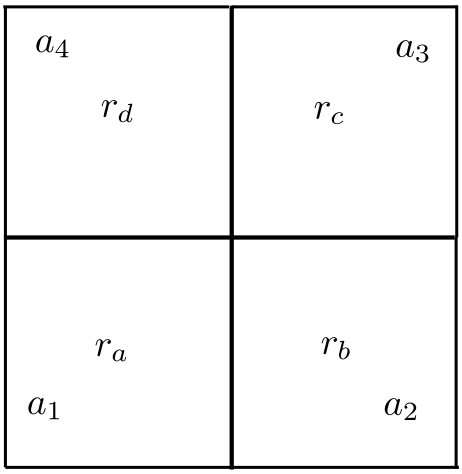


Figure 4.3: Areas used to calculate nodal masses in AWD. a_i are the areas associated with the subvolumes, r_a is the average radial coordinate associated with a_1, r_b with a_2, r_c with a_3 and r_d with a_4. The radial coordinates associated with the nodes are r_i with $i = 1 - 4$ where the numbering scheme defined in the same manner as the area numbering scheme

integrating about an area (as was effectively done in Cartesian coordinates),

$$\int_A \rho \frac{Dv_r}{Dt} dA = -\int_A \frac{\partial P}{\partial r} dA, \tag{4.18}$$

which results in a momentum update of the form,

$$\rho A \frac{Dv_r}{Dt} = -\left(P_1 - P_2\right) \Delta z, \tag{4.19}$$

or,

$$\frac{M_{node}}{r} \frac{Dv_r}{Dt} = \frac{-\left(P_1 - P_2\right)}{r} \Delta z. \tag{4.20}$$

To update the energy in a compatible form, the normal compatible energy update for Cartesian form is modified so that the dot product of the corner forces and the nodal velocity vectors is multiplied by the magnitude of the radial coordinate of the node, thus ensuring energy conservation in a proper sense. The final compatible

form of area weighted differencing is,

$$\vec{f}_{Compat,AWD} = \vec{f}_{Cart.}|r|, \tag{4.21}$$

and,

$$\Delta E_{Compat,AWD} = \vec{f}_{Cart.} \cdot \vec{u}|r|. \tag{4.22}$$

To summarise, this section has adapted a Cartesian numerical scheme to a cylindrical one. The key similarity is that the momentum update equation has remained unchanged. However the definition of subzonal (and thus cell centred and nodal) masses has changed, as well as the forces used for heating. It is these differences (effectively summarised as the addition of a radial weighting) which encompass the change of geometry from Cartesian to cylindrical. The fact that the momentum update scheme does not change has a key advantage in that it means that nodal masses retain the same inertia in cylindrical coordinates as would be calculated for a given density in Cartesian coordinates. This reduces numerical jetting sometimes seen in cylindrical simulations. This does however mean that momentum is no longer analytically conserved, due to the mass not exactly appearing in the momentum equation, however in practice this effect is found to be negligible, approximately one part in 10^{10}.

The name of the area weighting scheme is derived from the fact that such a scheme is arrived at by integrating about an area rather than a volume, for example, integrating the continuous momentum equation in cylindrical coordinates yields,

$$\int_A \nabla P dA = \vec{f}_{Cart}. \tag{4.23}$$

The left hand side of the momentum equation results in the inertia of the node being given as,

$$\int_A \rho dA = \rho A. \tag{4.24}$$

Both of these can be multiplied by the radial coordinate to then give the more obvious cylindrical scheme. The mass continuity requirement means that the subzonal masses must then be distributed in such a way that the mass of the node can be written as being proportional to the radial coordinate. This technique of integrating the continuous equations about an area shall be applied in the development of the cylindrical MHD scheme.

4.4 Shock Viscosity in Cylindrical Coordinates

As with the core hydrodynamic solver, both of the shock viscosities in Odin were originally formulated in Cartesian coordinates. As discussed previously it is desirable to carry over many qualities of the Cartesian scheme into cylindrical coordinates. Thus in line with previous discussions on area weighted schemes, the conversion of Cartesian viscosities into an area weighted scheme is simple, the Cartesian forces are calculated as normal, and multiplied by the radial value of the node the corner force is associated with,

$$\vec{f}_{visc,rz} = r\vec{f}_{visc,xy}. \tag{4.25}$$

There is however also the question of viscosity limiters and the scalar coefficient of the tensor viscosity to consider when converting to an area weighted scheme.

Considering the case of the tensor viscosity coefficient, the only possible area which may be changed is the calculation of the divergence of the velocity, and thus the velocity jump. If calculating the (integrated) quantity properly in rz, analogously with previous arguments in the control volume differencing section, an extra term would appear, $-u/|r|$, where u is the radial component of the velocity. The addition of this term gives incorrect results, this is made most obvious when returning to Noh's problem. This term would in fact cause uniform heating of the region, regardless of the position of the shock. As well as going against the analytical solution for this specific test case it also violates the self-similar motion invariance criterion, it is causing heating in the presence of uniform compression. Thus this term is not added, and the normal Cartesian form of the divergence is used to construct a velocity jump. For this reason, the scalar coefficient and thus the viscosity trigger in the tensor viscosity are left unchanged.

It is also worth considering the case of the tensor viscosity limiters. The question of the inclusion of the extra term here is best answered by comparing with the Cartesian case of uniform compression. What is required of the viscosity limiters is to turn off viscosity in cases where the velocity field is a linear function of the coordinates. The Cartesian test case has already shown the limiters to be very effective in doing this, and as the requirements are no different in cylindrical coordinates there is little reason to modify them.

During development a number of different combinations were tried within the tensor viscosity. Firstly the velocity jump was modified to include the extra term, but the limiters left unchanged. Secondly both the limiter and the viscosity jump were modified to include the extra term. Neither of the trials resulted in significant ($> 1\%$) differences in final solutions, as such, and due to the reasoning above, neither change

was implemented in the final version.
Results with the shock viscosity in area weighted schemes are not as successful as in the Cartesian case, most notably in Noh's test case. Here the magnitude of the wall heating is increased; the peak in post shock density is approximately 25% too low, compared with the undershoot in xy, which is $< 5\%$ too low. Secondly the shock speed in rz calculations is noticeably wrong, around 10% too fast. There are possible causes for these errors, either the viscosity is incorrect, or the lack of momentum conservation is causing shock calculations to be incorrect. These results are seen in multiple implementations of the Campbell Shashkov tensor viscosity, the majority of which use area weighted schemes. A noticeable exception is that of the astrophysical code BETHE-HYDRO [30], which uses the viscosity in a control volume differencing scheme. As a proper control volume scheme conserves momentum this is a good test of the cause of the errors in Noh's case. However, Campbell Shashkov viscosity is introduced into this numerical scheme in an area weighted sense only, in that the Cartesian forces are multiplied by $2\pi r$. This violates momentum conservation again. However, BETHE-HYDRO [30] does show improved accuracy in shock speed and post shock density when compared with full area-weight schemes, which is further improved by increasing resolution.

4.4.1 Dissipativity in Area Weighting Schemes

Another problem with shock viscosity in area weighted schemes, that is lacking consideration in the literature is that of dissapativity in rz. In the preceeding section, dissapitivity was assumed, based on the reasoning that as the continuous tensor, analytically had this quality the proper discretisation of it would inherit this property. It is enlightening to prove dissipativity based upon the discrete form alone. For simplicity an orthogonal grid will be assumed, thus reducing the number of terms to be considered. However this assumption can be relaxed, and dissipativity can be proven in a similar manner. Firstly considering the Cartesian case and once again restricting the discussion to a pair of nodes, p, and $p+1$. The respective viscous forces are:

$$\begin{aligned}\vec{f^p} = V_z \Bigg[&\frac{1}{l_{p+1/2}} \left(W^{P+1} \mu^{p+1} \vec{G}_{p+1/2} + W^P \mu^p \vec{G}_{p+1/2} \right) \\ &+ \frac{1}{l_{p-1/2}} \left(\mu^p \vec{G}_{p-1/2} + W^{p-1} \mu^{p-1} \vec{G}_{p-1/2} \right) \Bigg], \end{aligned} \tag{4.26}$$

and,

$$\vec{f^{p+1}} = V_z \left[\frac{1}{l_{p+3/2}} \left(W^{P+2} \mu^{p+2} \vec{G}_{p+3/2} + W^{p+1} \mu^{p+1} \vec{G}_{p+3/2} \right) + \frac{1}{l_{p+1/2}} \left(\mu^{p+1} \vec{G}_{p+1/2} + W^p \mu^p \vec{G}_{p+1/2} \right) \right]. \tag{4.27}$$

It is also important to recall,

$$\vec{G}_{p+1/2} = \frac{\vec{v_p} - \vec{v_{p+1}}}{l_{p+1/2}} \tag{4.28}$$

Now consider only the first two terms in (4.26) and the last two terms in (4.27). To consider all terms it would be required to in fact consider the entire cell, but using the cyclical definition of velocity difference vectors, (4.28), the result of this localised discussion can be applied to the entire cell. Grouping the terms under consideration, and remembering the work due to a viscous force is $W = -\vec{f} \cdot \vec{v}$ we obtain,

$$\begin{aligned} W &= -\frac{W^{p+1}}{l_{p+1/2}} \mu^{p+1} \vec{G}_{p+1/2} \left(\vec{v_p} - \vec{v_{p+1}} \right) - \frac{W^p}{l_{p+1/2}} \mu^p \vec{G}_{p+1/2} \left(\vec{v_p} - \vec{v_{p+1}} \right) \\ &= \left(\frac{W^{p+1} \mu^{p+1} + W^p \mu^p}{l_{p+1/2}^2} \right) \cdot \left(\vec{v_{p+1}} - \vec{v_p} \right)^2 . \end{aligned} \tag{4.29}$$

Which is positive definite as required. However repeating the calculation for an area weighted scheme yields,

$$\begin{aligned} W = &-\frac{W^{p+1}}{l_{p+1/2}} \mu^{p+1} \vec{G}_{p+1/2} \left(\vec{v_p} \left| r^p \right| - v_{p+1} \left| \vec{r}^{p+1} \right| \right) \\ &- \frac{W^p}{l_{p+1/2}} \mu^p \vec{G}_{p+1/2} \left(\vec{v_p} \left| r^p \right| - v_{p+1} \left| \vec{r}^{p+1} \right| \right) \\ = &\left(\frac{W^{p+1} \mu^{p+1} + W^p \mu^p}{l_{p+1/2}^2} \right) \cdot \left(\vec{v_{p+1}} - \vec{v_p} \right) \cdot \left(\vec{v_{p+1}} \left| r^{p+1} \right| - \vec{v_p} \left| r^p \right| \right) . \end{aligned} \tag{4.30}$$

This is no longer positive definite. The reason for this is simple, the area weighting discretisation of the tensor viscosity is not a proper discretisation of the tensor in rz coordinates, thus it no longer inherits it's continuous properties.

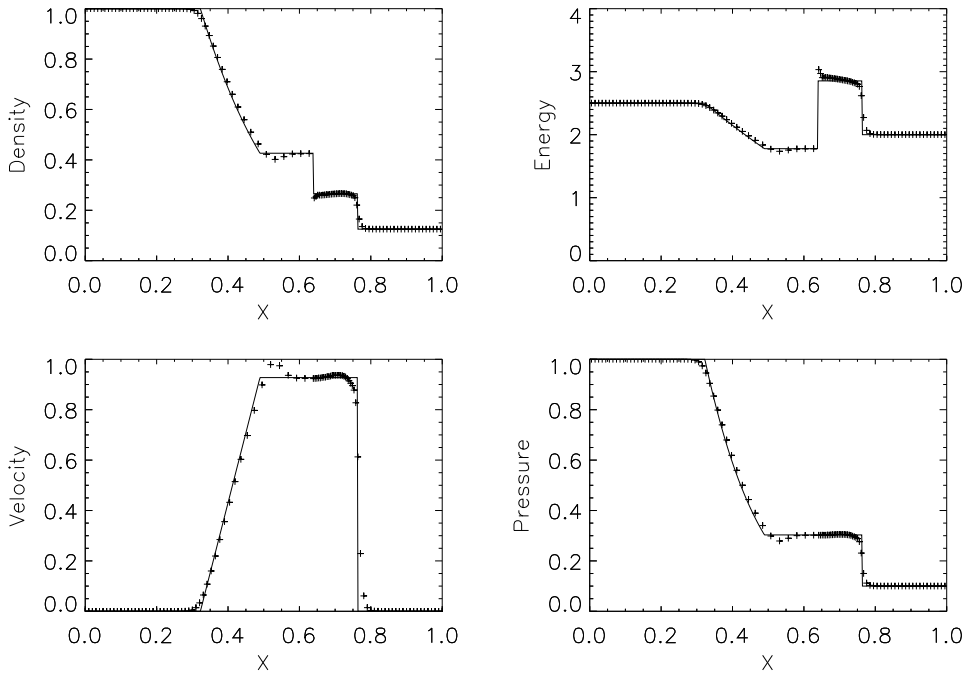


Figure 4.4: Results obtained for Sod's shock tube problem using tensor shock viscosity in cylindrical coordinates. The analytical solution is shown as a solid line, whilst the results obtained are shown as crosses.

4.5 Results

In order to validate the scheme some of the test problems from the previous section were rerun in cylindrical coordinates. In both the following test cases the velocity on axis is set equal to it's nearest of axis neighbour.

4.5.1 Sod's Problem

Sod's shock tube [21]is easily adapted to cylindrical coordinates by initialising a cylinder, with the interface at z=0.5, so that,

$$(\rho, e_i) = \begin{cases} (1, 2.5) & \text{if } z < 0.5 \\ (0.125, 2) & \text{if } z > 0.5 \end{cases} \tag{4.31}$$
$$\vec{v} = 0.0$$

with reflecting boundary conditions at $z = 0.0$ and $z = 1.0$. the results obtained are in figure 4.4.

4.5.2 Noh's Problem

Noh's problem [23] is also easily cast into cylindrical coordinates, now the problem consists of an imploding sphere rather than an imploding cylinder.

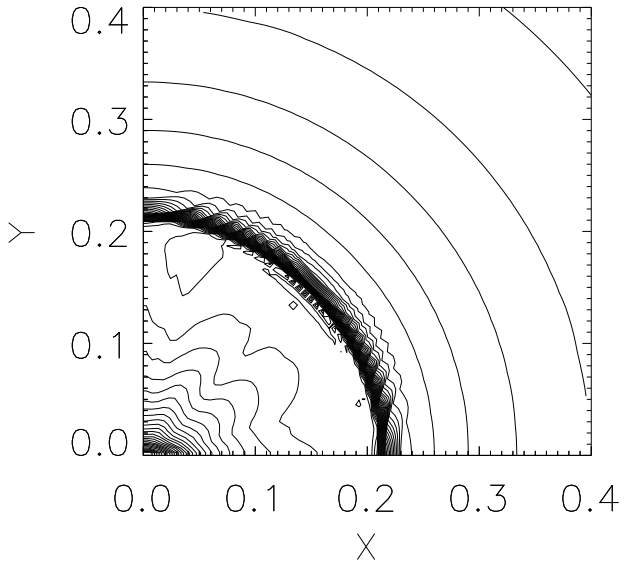


Figure 4.5: Density contour plot for Noh's problem run on a Cartesian grid with tensor shock viscosity, in cylindrical coordinates.

4.6 Summary

Two competing methods for discretising hydrodynamics in cylindrical coordinates have been described, area weighted, and volume weighted. This distinction does not occur in Cartesian coordinates, where the two methods are equivalent. The major disadvantage of volume weighted methods is the lack of symmetry preservation. This is not a disadvantage shared by the area weighted scheme, which inherits the same symmetry preservation qualities of the Cartesian scheme discussed previously. However, momentum conservation is no longer exact, although during testing the magnitude of this violation was found to be small. The area weighted scheme also has the advantage of ease of implementation; the Cartesian scheme requires very little modification to be adapted to run in cylindrical coordinates. This is not only beneficial for implementation, but also for code maintanance, as such an area weighted method was used to implement cylindrical hydrodynamics within Odin.

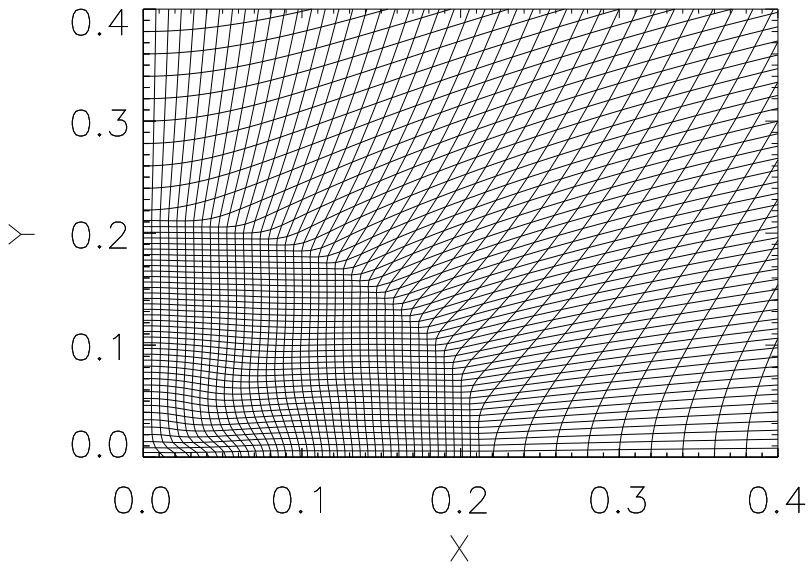


Figure 4.6: Grid for Noh's problem run with tensor shock viscosity on an initially Cartesian grid, in cylindrical coordinates.

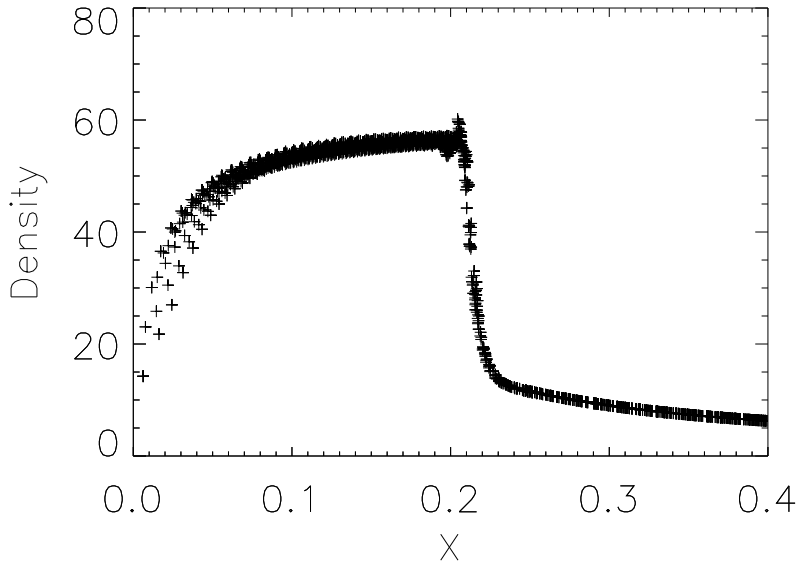


Figure 4.7: Scatter plot of density against radius for Noh's problem run with tensor shock viscosity, in cylindrical coordinates.

Chapter 5

Subzonal Pressures

5.1 Introduction

Lagrangian simulations have a major problem; grid tangling. Heavily distorted meshes, or mesh segments that have almost collapsed lead to order of magnitude reduction in time step, and eventually to the premature end of the simulation. There are many physically relevant motions that lead to time step reduction, such as turbulent motion, or the collapse of cells during compression, however there are also non-physical grid motions which can cause the simulation to end for non-physical reasons. Most commonly referred to as hourglass modes, or keystone motion, or on a larger scale as spurious vorticity these grid motions are the bane of Lagrangian simulations.

Caramana and Shaskov [31] split these two motions into two types, the first type being grid scale motion associated with hourglass modes, which occurs due to quadrilateral (hexahedral in three dimensions) cells being under-constrained with respect to the total numbers of degrees of freedom of the grid. The second type was the larger scale spurious vorticity, motion which was described as not relating to the physical solution. As pointed out by Caramana and Shaskov [31] this type of motion can only really be defined with respect to a known one-dimensional solution, where movement is seen to deviate away from the analytically expected motion. This type of motion is most clearly seen by attempting to solve Noh's problem on a quadrilateral mesh, with edge based viscosity. Caramana and Shaskov [31] explain such un-physical grid motion as being due to the solution gradients being mis-aligned with the grid. Taking a more critical point of view however, it is in reality caused by an over-reliance on the grid by the hydrodynamical solver. In attempt to alleviate these problems Caramana and Shashkov [31] introduced the concept

of subzonal pressures, which result from the assumption that subzonal masses are constant, and utilised these pressures to enable Lagrangian simulations to run to longer time scales, by limiting artificial grid motion.

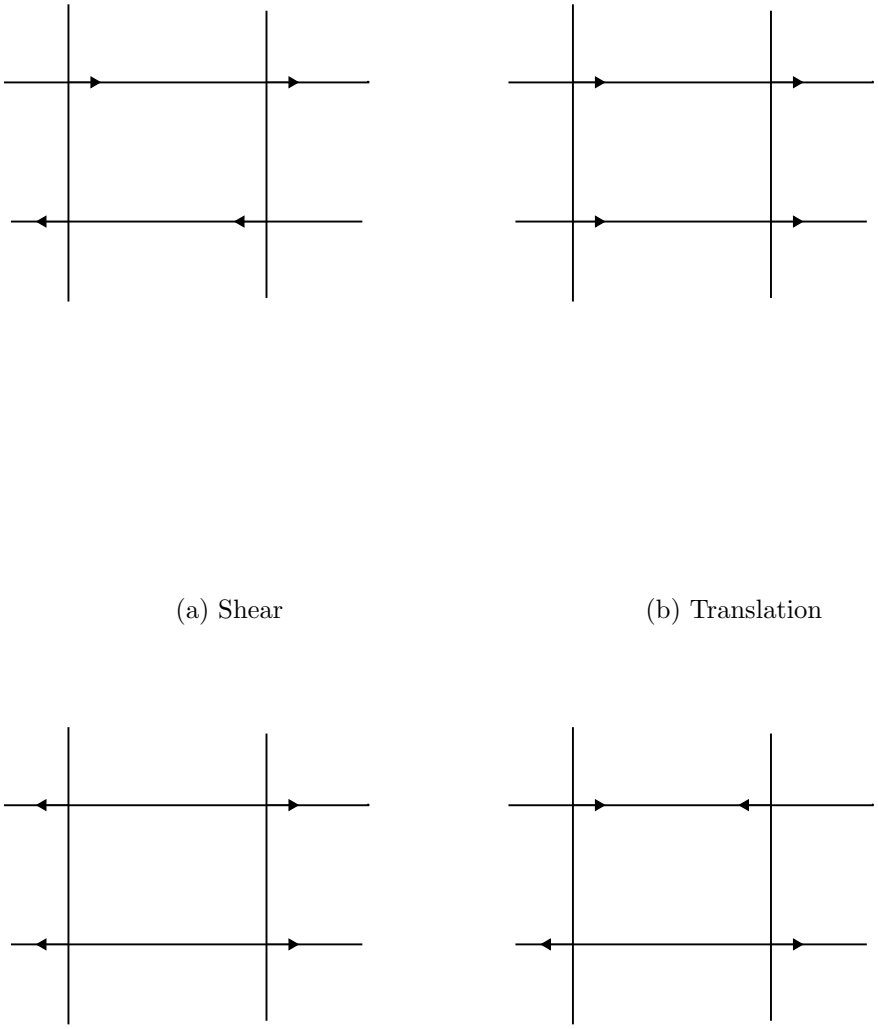
5.2 Modes of Grid Motion

For a two dimensional grid there are six physical degrees of freedom; translation, shear and compression/expansion, in each direction. Rotation is obtained by combination of shear in two dimensions. There is a fourth degree of freedom in each dimension, which is non-physical. Caused by checkerboard pattern in the components of the velocity, it is a situation where the sign of the velocity component changes at each node, and each row of nodes in the second direction is out of phase with it's two neighbours. These non-physical degrees of freedom lead to quadrilateral grids being called under-constrained. The four degrees of freedom for the x-direction are shown in figure 5.1.
Of the three physical degrees of freedom, only expansion/contraction causes a direct hydrodynamical response, due to the fact that in it's pure form it is the only one that causes a change of volume of the cell, thus changing pressure and changing the forces. However neither shear nor translation cause problems in regard to time step collapse. Hourglass modes also cause no change in volume of the cell, and thus illicit no hydrodynamical response, and as discussed previously do cause a timestep collapse. They have also been found to grow in time, and thus their eradication is imperative for Lagrangian hydrodynamical simulations.

5.3 Subzonal Masses and Pressures

Whilst hourglass modes do not change the total cell volume they do cause a change in the volume of the subcells, the same subcells used to define subzonal masses in the compatible formulation. At $t = 0$ all that is known is the cell centred density, and this density is prescribed to each of the subzones within the cell, and the subzonal mass is thus calculated. However during an hourglass mode (and it is worth pointing out that this is not a quality shared by the physical modes) the subzones change volumes at different rates, and their densities (due to their assumed constant mass) diverges. This change in density causes the pressures associated with each subcell to be different. These are calculated by

$$\delta P = \delta \rho c_s^2, \tag{5.1}$$



(a) Shear (b) Translation

(c) Expansion (d) Hourglass

Figure 5.1: Modes of grid motion

where c_s is the sound speed. The internal energy of the cell has been assumed to be constant throughout the cell, for now only forces arising from the subzonal masses being constant shall be considered. Although (5.1) is only an approximation, Caramana and Shashkov [31] found it to be an adequate one, and it avoids the need for four extra calls to equations of state routines which can become costly.

5.4 Calculation of Subzonal Forces

Having established the change in pressure due to constant subzonal masses it remains to calculate the forces due to these pressure perturbations. Caramana and Shashkov base their formulation around the conservation of momentum, and that discussion is mirrored here. Considering for now only the bulk pressure of a cell (i.e. neglecting subzonal pressure perturbations) conservation of momentum requires that the sum of forces acting on nodes must sum to zero. Taking this sum over the cells this requirement is,

$$\sum_z \sum_{i=1}^{8} \vec{f}_i^z = \sum_z \sum_{i=1}^{8} p_z \vec{a}_i = \sum_z p_z \sum_{i=1}^{8} \vec{a}_i = 0. \tag{5.2}$$

Given that the pressure in a zone is arbitrary this requirement is simply,

$$\sum_{i=1}^{8} \vec{a}_i = 0, \tag{5.3}$$

for each cell. This is achieved through the fact that each cell is a closed surface, so the sum of the outward normal vectors sums to zero. This reasoning doesn't change for subzonal pressure forces, but now the reasoning must be applied to the subzones, the sum of the forces from each subzone must sum to zero.

5.4.1 Dynamical and Non-Dynamical Points

At this stage it is important to differentiate between dynamical and non-dynamical points. Dynamical points are simply the nodes of each cell, points for which the velocity is directly calculated, and thus points over which momentum is summed. These points are illustrated by figure 5.2.Thus the requirement for subzonal momentum conservation may be stated as the requirement that the forces on dynamical points arising from subzonal pressures must sum to zero. Non-dynamical points are all other points throughout the domain. The position of these points changes with the motion of the fluid, but the velocity of such points is simply an average of the surrounding dynamical points, and thus the momentum of such points is not taken

into account in momentum considerations. Dynamical and non dynamical points are illustrated by figure 5.2

This distinction is particularly important in calculating subzonal pressure forces. As the subzones are quadrilaterals like the cells the forces on the corners of these subzones can be calculated in the same way as corners of cells. However this results in forces on non-dynamical points, specifically the centre of cells, and edge mid-points. Unless these forces are properly accounted for momentum conservation will be violated.

The simplest way of transferring these forces from non-dynamical to dynamical points is to prescribe to each node a fraction of the force on each of the non-dynamical points. This fraction is simply set equal to the fraction of the non-dynamical points velocity given by the node in question. So specifically, the force on the cell centre is prescribed to each node with a weighting of one quarter, and the force on each edge mid-point is prescribed to the two nodes on that edge with a fraction of one half. This method is illustrated by figure 5.3. This method is simple, and easily applied to subzones of arbitrary shape, however it is not a unique force differencing.

Caramana and Shashkov [31] go on to calculate what they call a more preferable force differencing using the following two step procedure. This is carried out by considering a rebound of forces onto neighbouring points. The original mesh forces are illustrated by figure 5.4. The first step is to rebound the forces on the cell centre associated with the median mesh back onto the midpoints of cell edges, whilst at the same time rebounding the initial force contributions associated with the half cell edge normals to their respective nodes. The force distribution after this step is illustrated by figure 5.5. The second stage is to then prescribe the intermediate cell edge midpoint forces onto the connected nodes with a weighting of one half, and a sign illustrated by figure 5.6. The final resulting force on node 1 can be written in the following three equivalent ways,

$$\delta\vec{f_1} = \delta P_1 \left(\vec{a}_1 + \vec{a}_8\right) + \frac{1}{2}\left[\left(\delta P_1 - \delta P_4\right)\vec{S}_4 + \left(\delta P_2 - \delta P_1\right)\vec{S}_1\right], \tag{5.4}$$

$$\delta\vec{f_1} = \delta P_1 \left(\vec{a}_1 + \vec{a}_8\right)/2 - \delta P_4\vec{S}_4/2 + \delta P_2\vec{S}_1/2, \tag{5.5}$$

and,

$$\delta\vec{f_1} = \left(\delta P_1 + \delta P_2\right)\vec{S}_1/2 - \left(\delta P_1 + \delta P_4\right)\vec{S}_4/2. \tag{5.6}$$

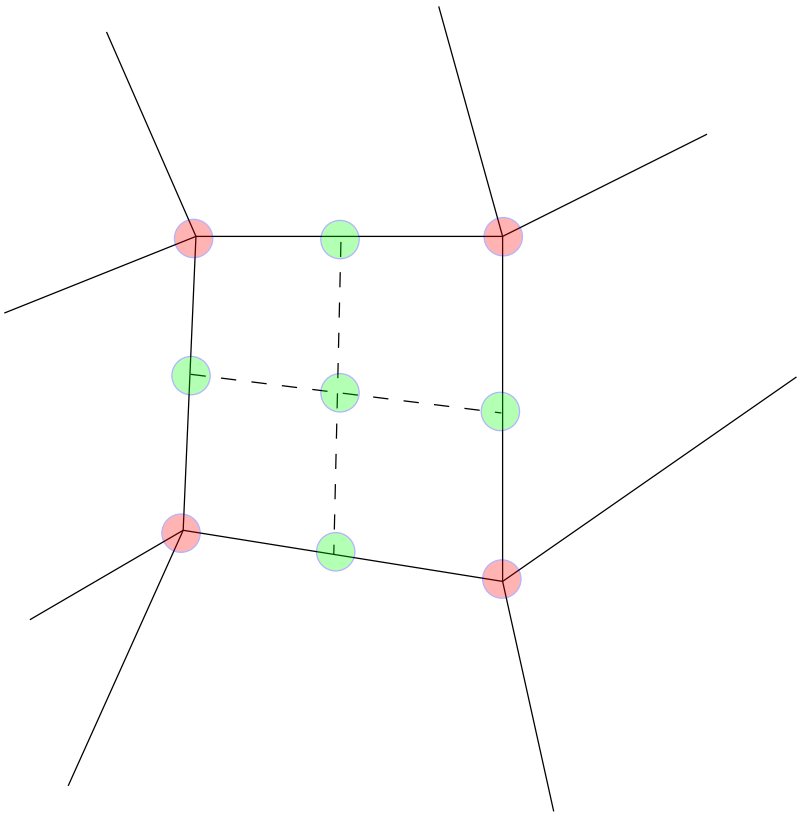


Figure 5.2: The dynamical (red) and non-dynamical points (green) of a cell.

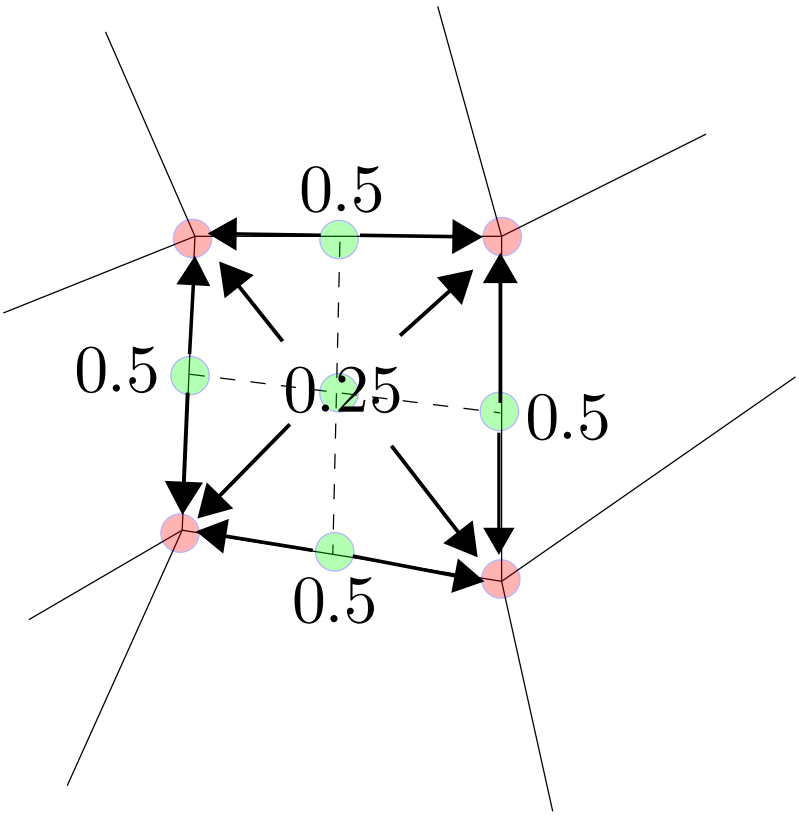


Figure 5.3: The redistribution with weightings of forces from non-dynamical points to dynamical points.

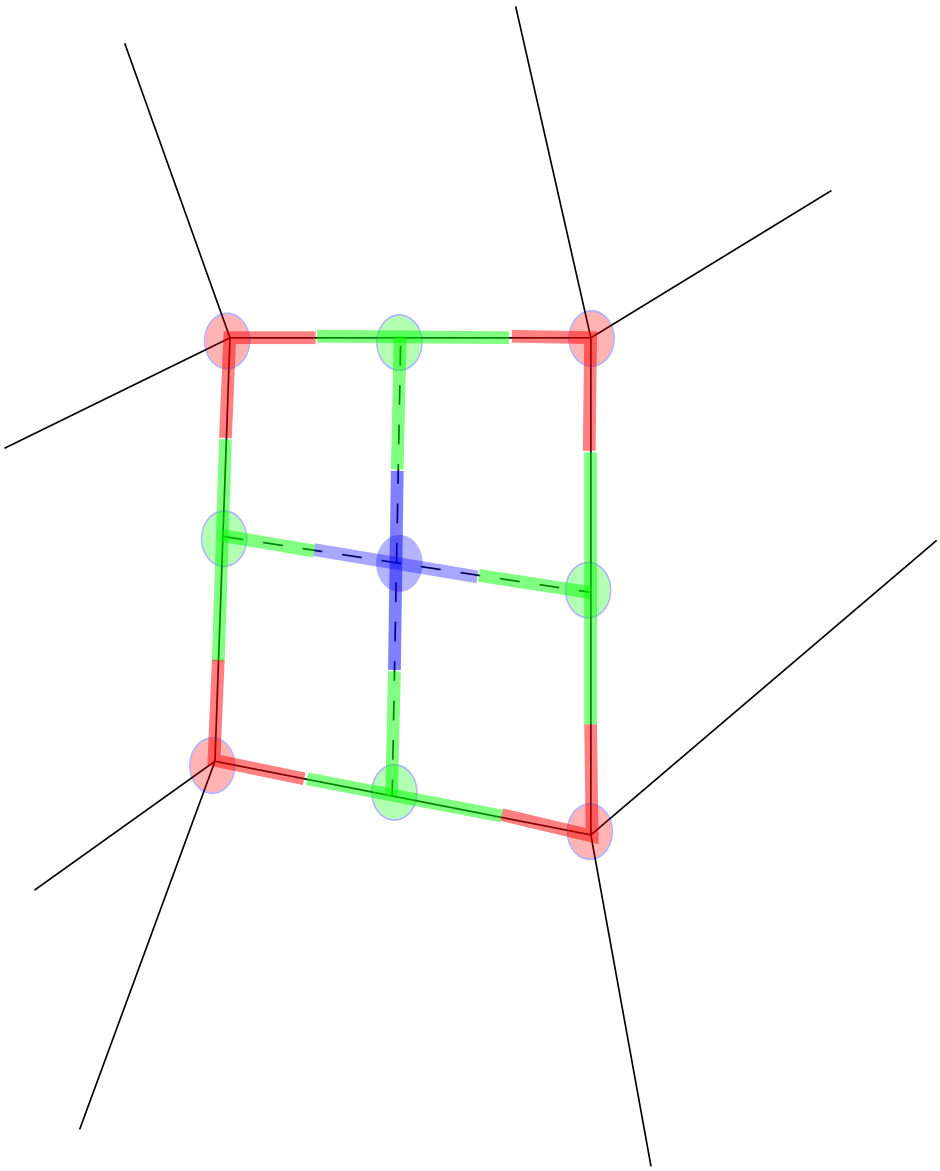


Figure 5.4: The initial forces calculated for subzonal pressures, before any redistribution of forces has occurred. The mesh segments contribution for forces on nodes are highlighted in red, forces on mesh midpoints green, and cell centres blue.

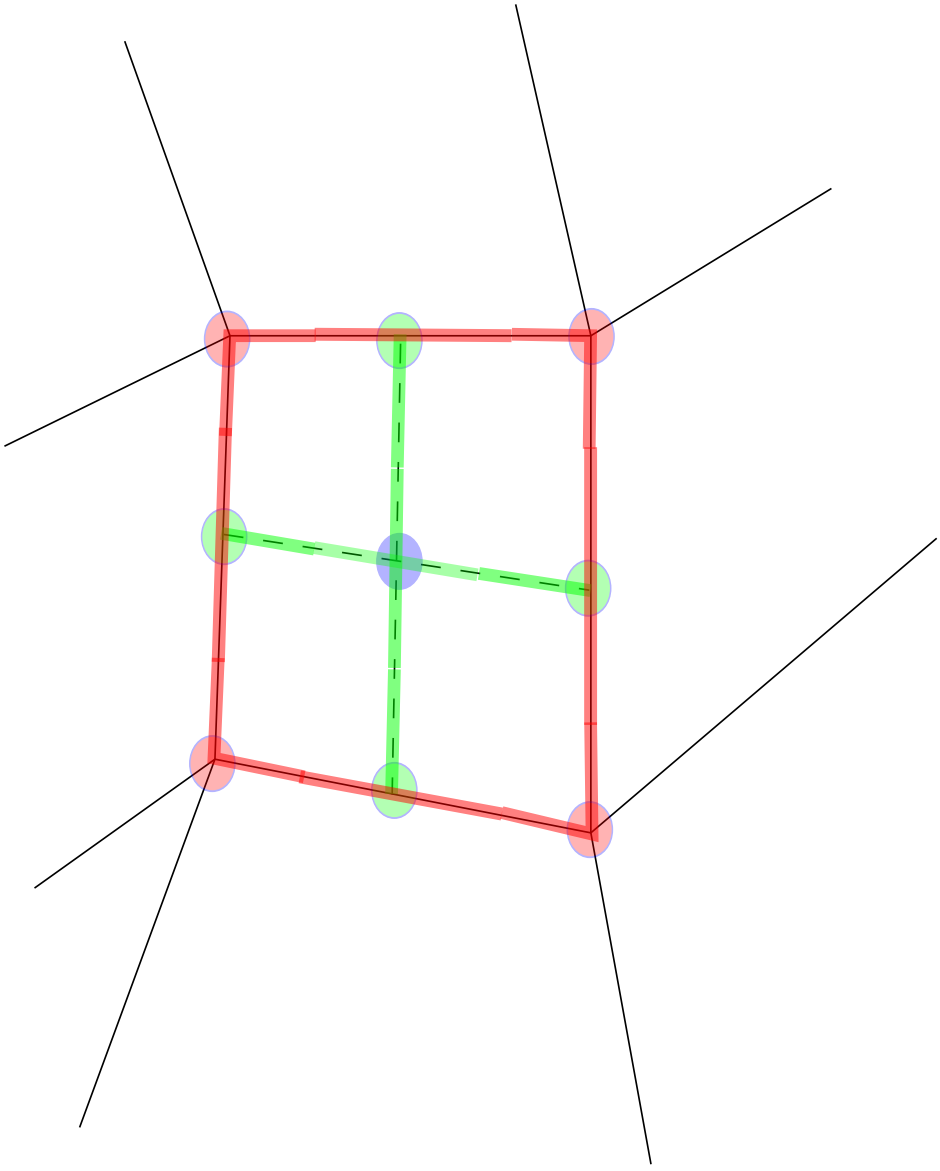


Figure 5.5: The intermediate forces in the rebounding method. Force contributions on the cell centre have been move to their nearest (logical) mesh midpoint neighbours. Force contributions on the mesh midpoint neighbours along the primary mesh have been moved to their nearest (logical) nodal neighbours.

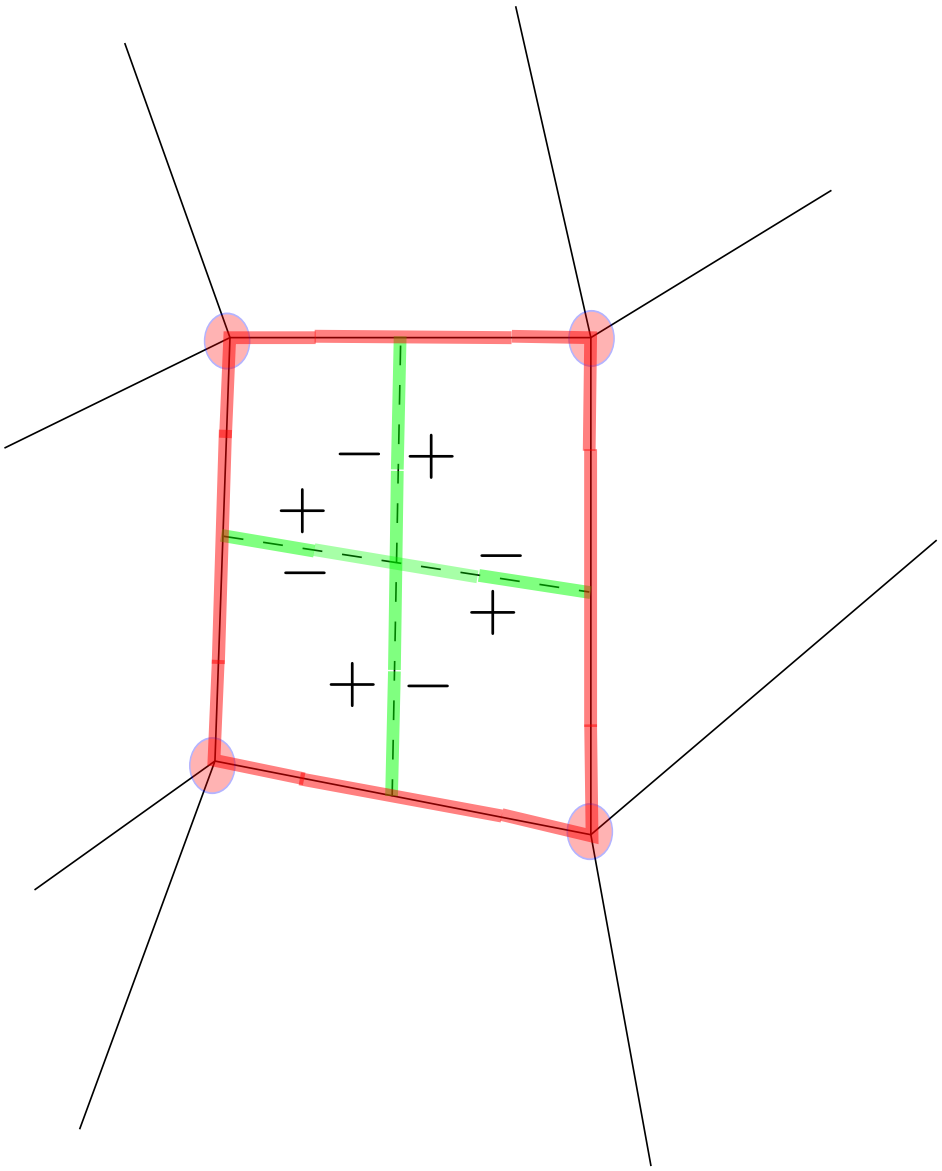


Figure 5.6: The final rebounding of forces to nodes. To be clear, the force along a mesh length is the pressure difference across it multiplied by its vector. For example the force at $\vec{S}_1$ is $(\delta P_2 - \delta P_1)\,\vec{S}_1$. The final rebounding from median mesh to nodes is with a weighting of one half and a sign shown by the $+$ and $-$ signs in the diagram. For example the force along $\vec{S}_2$ is rebounded to node 2 with a weighting of $+1/2$ and to node 3 with a weighting of $-1/2$.

5.4.2 Merit Factor

Carmana and Shashkov [31] state that the form given by (5.4) multiplied by two is used in their code. This multiplication does not appear justified, but this force is what they term as having a merit factor of unity. The introduction of the concept of a merit factor enables Caramana and Shashkov [31] to artificially reduce or increase the magnitude relative to (5.4) as deemed necessary. Some automation of this merit factor was discussed, centred around increasing the merit factor for higher fractional variation in density, presumably motivated by the idea that as subzonal density perturbations grow the grid is excessively tangling and thus stronger subzonal forces are desirable.

Finally it should be pointed out that (5.4), (5.5) and (5.6) are all equivalent in Odin. However Caramana and Shashkov employ the modification to the gradient vectors described in [28]. Under such a scheme (5.4), (5.5) and (5.6) are no longer equivalent. However as mentioned in previous chapters these modifications were not employed in Odin, and as such (5.4), (5.5) and (5.6) remain equivalent.

5.4.3 Subzonal Pressures-an alternative formulation

The discussion in the proceeding two sections describe Caramana and Shaskov's [31] derivation of subzonal pressure forces, however it is a discussion that diverged from physical intuition. An important observation is that the apparent problem of forces acting on non-dynamical points arises even for all subzonal densities (and thus pressures) being equal, should the pressure in neighbouring cells be different, as this yields a net force on the edge mid points. However this does not in itself cause a violation of momentum conservation.

A far simpler way of deriving subzonal pressure forces is to calculate pressure forces along the median mesh. The pressure at the median mesh is taken to be the average of the two subzonal pressures in each of the neighbouring subzones. The pressure force on each of the nodes is then calculated in the normal way, except that in place of the cell pressure the average pressure at the median mesh is used. This simple formulation has two desirable quantities. Firstly, as each median mesh force acts on two nodes with opposite signs it conserves momentum exactly, and secondly it is exactly equivalent to (5.6), disregarding the non-physical methods of merit factors and modifications to the gradient vector.

5.4.4 Subzonal Pressures within the Compatible Framework

Having shown that the subzonal pressure forces can be implemented in a way that conserves momentum, it is important now to consider energy conservation. Fortunately as the subzonal pressure forces have been conceived in terms of corner forces they are easily implemented within the compatible framework and thus do not pose any complications for energy conservation.

5.5 Temporary Triangular Subzones

The work by Caramana and Shashkov [31] was based around a number of previous efforts to reduce grid tangling in Lagrangian calculations, an important example of which was the work by Browne [32]. The work by Browne attempted to reduce time step reduction in calculations involving long thin zones which were prone to twisting, this was carried out by calculating forces due to triangular subzones, formed by the connection of two nodes and the cell centre. These forces have two origins, firstly the assumption that the mass within a triangular subzone is constant, and secondly by temporarily depositing the energy due to viscous heating into the triangular subzones.

The work of Carmana and Shashkov [31] resulted in forces due to quadrilateral subzones based around the assumption of a constant subzonal mass, however triangular subzones were also considered, and due to the lack of detail in the work of Browne [32] that discussion shall now be followed.

The previous method of redistributing forces on non-dynamical points to dynamical points for triangular subzones is simplified greatly by the fact that only one non-dynamical point is being considered, the cell centre. However, given that the previous discussion was simplified by already using the force calculation for quadrilaterals developed for the primary cells, it is useful to briefly summarise how those forces are calculated, and how this method can be applied for triangular zones. The pressure within a cell can be seen to push out on the cell edges, and these forces must be distributed to the nodes. This is done by simply equally dividing the force on each cell edge (in three dimensions, a cell face) to each of the two (four in three dimensions) nodes connected to the edge (face). For example the resulting force on node one is,

$$\vec{f}_z^1 = (\vec{a}_1 - \vec{a}_8)\, P_z. \tag{5.7}$$

In the previous chapter on compatible methods it was shown that this force arose from energy conservation considerations alone, and it was in fact shown to be equiv-

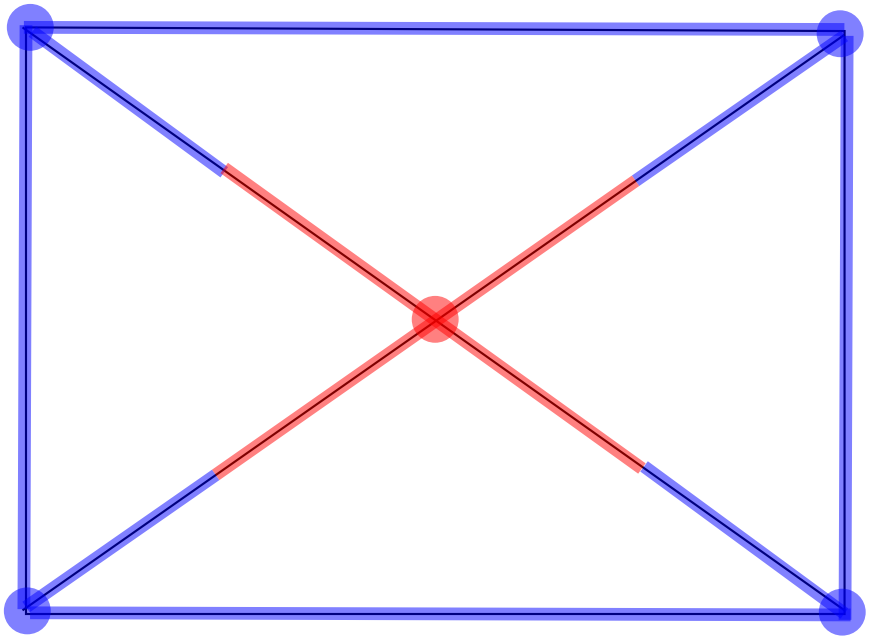


Figure 5.7: Original force segments calculated for pressure perturbations in triangular subzones.

alent to integrating directly around the median mesh, this of course is no longer true for volume weighted differencing, but in order to aid comparison with the work of Caramana and Shashkov [31] to calculate forces on dynamical and non-dynamical points the method of distributing outward pressure forces from edges to nodes shall be followed. For a quadrilateral subzoning these forces are illustrated by 5.4, whereas a similar approach for triangular subzones yields forces summarised by 5.7, and given by,

$$\vec{f_1'} = \delta p_1 \vec{a}_1 + \delta p_4 \vec{a}_8 + (\delta p_1 - \delta p_4)\, \vec{D}_1/2, \tag{5.8}$$

and

$$\vec{f_5'} = \frac{1}{2}\left[(\delta p_4 - \delta p_1)\, \vec{D}_1 + (\delta p_1 - \delta p_2)\, \vec{D}_2 + (\delta p_2 - \delta p_3)\, \vec{D}_3 + (\delta p_3 - \delta p_4)\, \vec{D}_4\right], \tag{5.9}$$

where the vectors are shown by figure 5.8 and the δp_is are defined at the centre of triangular subzones. The forces of the cell centre are distributed back to the nodes according to which node the relevant vector normal is associated with, for example the forces proportional to $\vec{D}_1$ are assigned to node 1, so that the final force on node 1 is given by,

$$\vec{f_1} = \vec{f_1'} + (\delta p_4 - \delta p_1)\, \vec{D}_1/2, \tag{5.10}$$

or simplifying,

$$\vec{f_1} = -\delta p_4 \vec{S}_4 + \delta p_1 \vec{S}_1. \tag{5.11}$$

This force redistribution is demonstrated by figure 5.9. Of course, the force given by (5.11) is simply the usual median mesh force differencing which can be calculated by directly integrating around the median mesh. An alternative formulation is given by redistributing the force on the cell centre to each of the four nodes with a weighting of one quarter so that,

$$\vec{f_1} = \vec{f_1'} + \frac{1}{4}\vec{f_5}. \tag{5.12}$$

This is demonstrated by figure 5.10. Clearly in calculating (5.12) more averaging is carried out than in calculating (5.11) so the forces due to a given value of δp_i will be reduced as they are spread out more over the nodes, although the total force remains the same of course. For the remainder of the discussion the form of triangular subzoning given by (5.11) shall be used, as it results from the more physical consideration of integrating about the nodal volume, a method notable for the fact that it will always conserve momentum. As the forces are once again calculated in terms of the affect on a node these forces are easily incorporated into a compatible

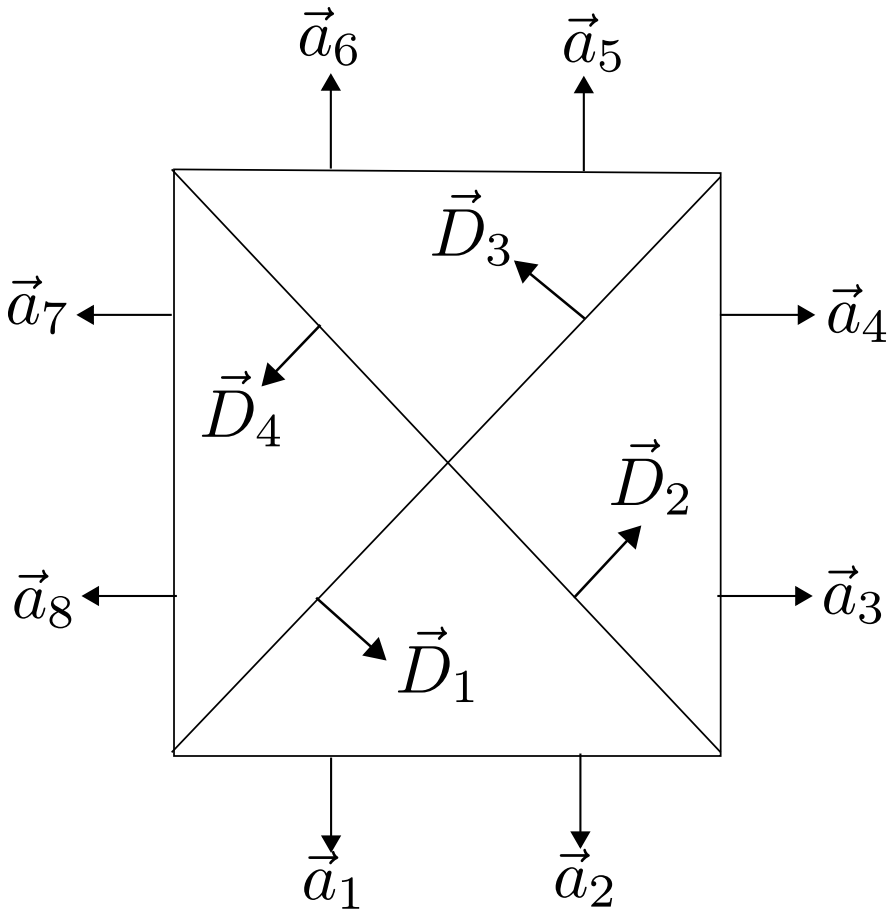


Figure 5.8: Vectors used to calculated forces arising from triangular subzonal pressures.

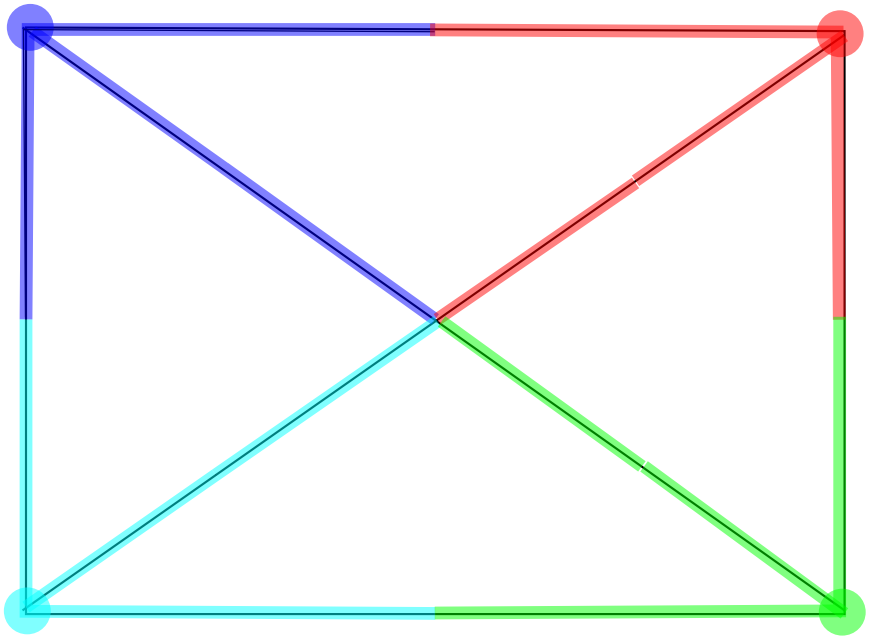


Figure 5.9: Redistribution of forces from central non-dynamical point to nodes.

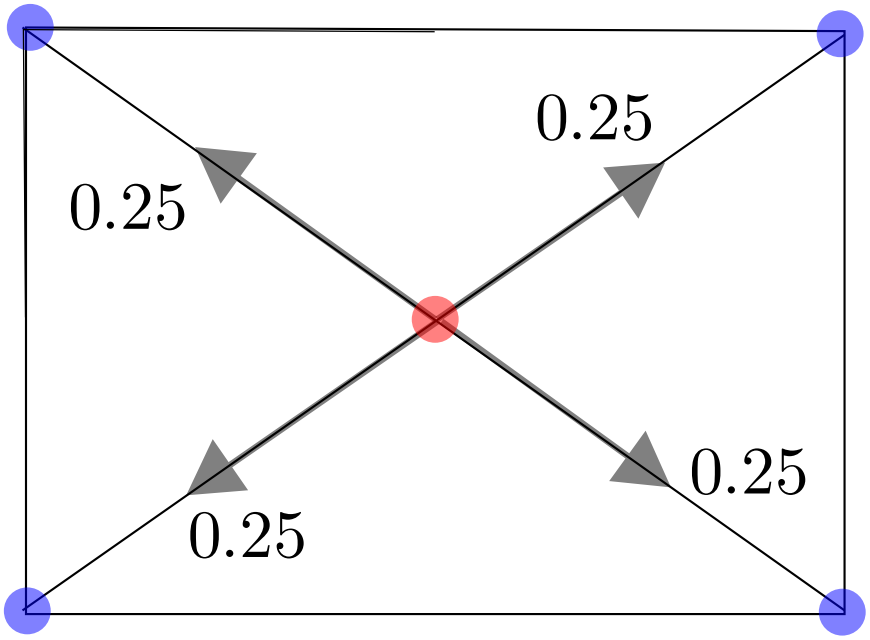


Figure 5.10: Averaging of central force to nodes.

formulation.
The original work by Browne [32] considered forces due to both constant triangular subzonal mass and temporary viscous heating in the subzones. Given that the assumption of constant quadrilateral subzonal mass was already present in Odin, only forces arising from temporary heating of triangular subzones shall be considered. Secondly it was also decided that within Odin, forces arising from constant quadrilateral subzonal mass, and temporary heating of triangular subzones would be considered separately, so the methods could be used in isolation, or together.
The temporary heating of Browne [32] was originally rejected by Caramana and Shashkov [31] due to the fact that there was no physical time scale over which the energy was assumed to equilibrate across the cell; the temporary heating was only applied at the predictor step, at the corrector the heating was applied to the entire cell. However such a method is beneficial in that it always forces the subzonal forces to be of order Δt less than the original forces, hopefully reducing some of the numerical artifacts seen in using subzonal pressures discussed in the last section of this chapter. Temporary triangular subzoning has increased in usage over the past decade or so and is often applied to calculations involving tensor shock viscosity, perhaps due to the fact that for an arbitrary grid these viscous effects are discretised into triangular subzones.

5.6 Results

5.6.1 Sedov's Problem

Although subzonal pressures are not used in production runs of Odin as a brief test of subzonal pressures, results obtained for Sedov's problem [27] in both Cartesian and cylindrical coordinates are presented. This case is of particular relevance as running in pure Lagrangian mode on a Cartesian mesh without subzonal pressures the time step collapses to such an extend the problem does not complete.

Cartesian Coordinates

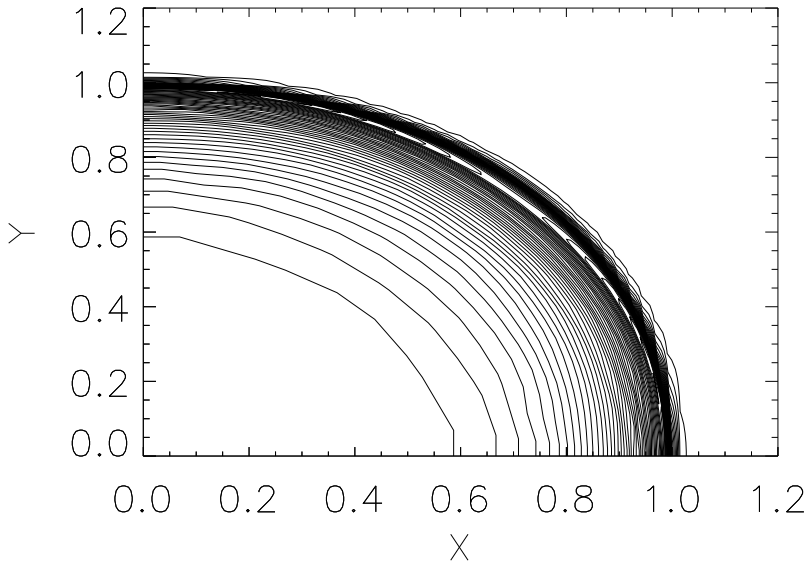


Figure 5.11: Density contour plot for Sedov's problem at $t = 1.0$ run on a Cartesian grid with tensor shock viscosity.

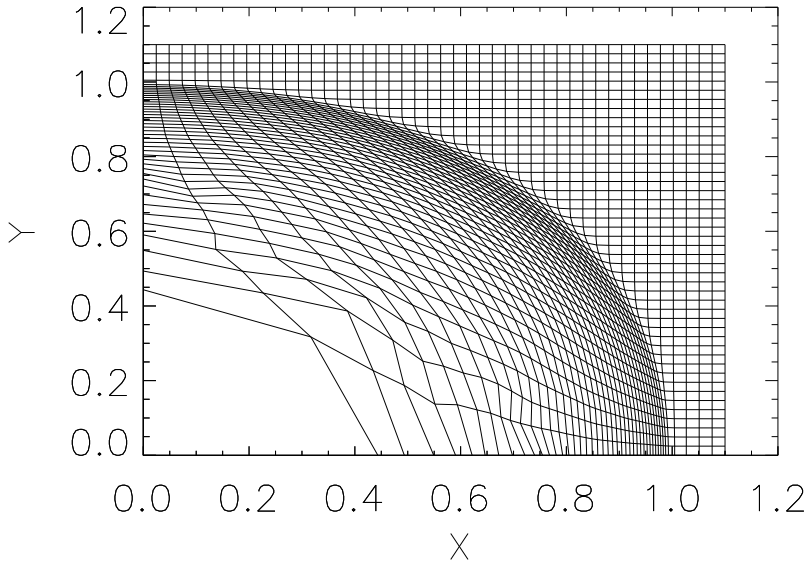


Figure 5.12: Grid for Sedov's problem at $t = 1.0$ run with tensor shock viscosity on an initially Cartesian grid.

Cylindrical Coordinates

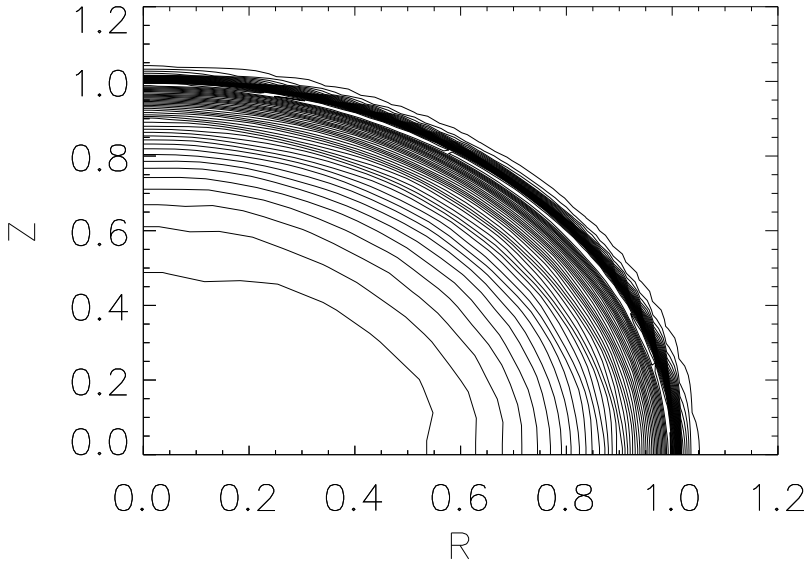


Figure 5.13: Density contour plot for Sedov's problem at $t = 1.0$ run on a Cartesian grid with tensor shock viscosity, in cylindrical coordinates.

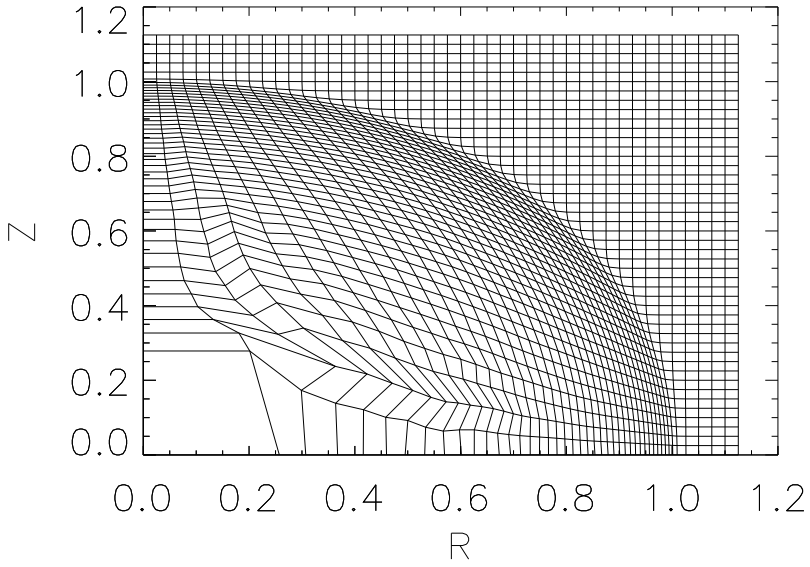


Figure 5.14: Grid for Sedov's problem at $t = 1.0$ run with tensor shock viscosity on an initially Cartesian grid, in cylindrical coordinates.

5.7 Summary

Grid distortion and tangling, and the prevention of such problems has been examined, the main method of prevention being subzonal pressures. Whilst this method has been shown to extend the range of problems that can be run in a pure Lagrangian mode, it is known to alter the growth rate of the Rayleigh Taylor instability, a key physical phenomenon to be examined by ALE codes. As such, whilst subzonal pressures have been implemented within Odin for both cylindrical and Cartesian coordinates, they are not used in production runs, due to concerns over the affect of the method on the validity of the answer.

Chapter 6

First Order Remapping Methods

6.1 Introduction

Whilst keeping a calculation purely Lagrangian for as long as possible is beneficial, for most problems of physical interest where flow will either from the onset or at some later stage be misaligned from the grid the need for a remap will eventually arrive. In order to simplify the introduction to remapping strategies and concepts the discussion in this chapter is limited to a donor cell, or first order basis.
Judging exactly when a remap should occur is a difficult, and nearly always problem dependent issue. The most basic criteria for remapping is based on time step considerations alone, should the time step fall to below some value then a remap can be triggered, here the decision to remap is centred on finishing the calculation. In other cases some prior knowledge of the physical problem might be useful, for example allowing the calculation for an implosion to remain Lagrangian until the radius is reduced by a certain factor. Other more geometric criteria could be used, having a maximum aspect ratio for cell, or a maximum and minimum value of internal angles of the cells. Within this thesis the time step criteria is used, as in general the other criteria are encompassed by this implicitly, and it is by far the simplest to implement, although it may require some trial test runs of the problem. This isn't always possible, in which case it may be useful to use other more complex criteria.
Having decided when to perform a remap it remains to calculate where to remap to. Here again a number of options exist, from simply locking the grid in the simplest case, to the more complex equipotential remapping. This is not always an obvious question, and it is addressed in a later section.

A multi-material remap is more complex and has additional considerations. The work described in this thesis is a single material ALE code, although Odin was further developed into a multi-material code by Dr C. S. Brady. Where relevant multi-material considerations will be mentioned in this and the later chapter on higher order remaps, but the discussion focuses on single material remaps.
To begin the discussion on first order remaps a new grid shall be assumed, as remapping methodology and mesh motion can be considered (almost) entirely separately.

6.2 General Remapping Methodology

There are two general remapping strategies, swept region based remaps, and intersection based remaps. Beyond these methodologies exist hybrid remapping methodologies. In order to explain and decide between remapping methodologies it is first beneficial to explain a basic remapping method, before making it specific to any of the aforementioned schemes.
Remapping is a purely geometric step, however basic physical ideas are involved, specifically through the concept of conserved quantities. Having identified a new mesh to which to move to, a decision must be made about how material is transferred between cells, it is this decision which is taken by the choice between swept region and intersection based remaps. However, in a one dimensional code these methods are the same, so it is within this context that a basic remap shall be explained. The discussion on intersection based remaps demonstrates how the scheme reduces to a swept region based remap in the limiting case of a one dimensional calculation.

6.2.1 One Dimensional Remap

Again for the sake of simplicity an Eulerian remapping mesh motion shall be assumed, although no loss of generality occurs; this discussion is easily adapted to any mesh motion strategy by replacing $\vec{v}\Delta t$ with a displacement vector defined by the difference between the old and new meshes.
Consider the case demonstrated by 6.1. Here the fluid, and thus the mesh has moved to the right by an amount $\vec{v}\Delta t$ and thus during the remap the mesh must move minus this amount. Clearly there is a volume of fluid that must be transported between the two cells being considered, given by,

$$V_{remap,i} = -\vec{v}\Delta t dy dz. \tag{6.1}$$

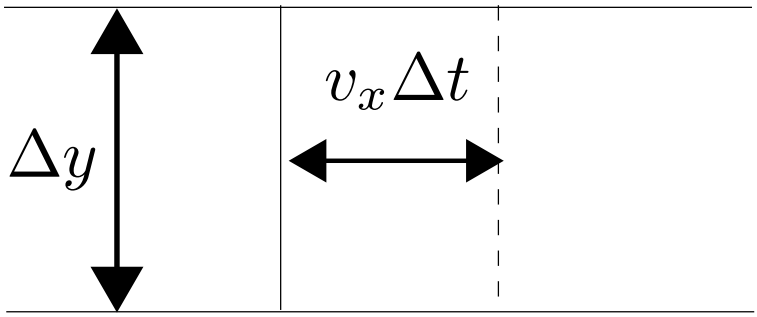


Figure 6.1: One dimensional remap with Eulerian grid motion. The displaced (post-Lagrangian grid is shown by the dashed line, the original/remapped grid shown by solid lines.

Note here the concept of a signed volume has been used, so that the post remap volumes, denoted V are calculated in terms of the remap volume and the pre-remap volume V' as follows,

$$V_i = V'_i - V_{remap,i}, \tag{6.2}$$

and,

$$V_{i+1} = V'_{i+1} + V_{remap,i}. \tag{6.3}$$

Expanding (6.2) and (6.3) to the case where the mesh has moved in multiple locations the following update for the volume of any cell is given by,

$$V_i = V'_i + V_{remap,i-1} - V_{remap,i}. \tag{6.4}$$

The sign convention in (6.4) is essentially arbitrary (although it is the one used in Odin), but it is used in deciding the sign of the volume as used by (6.1). Having made this sign convention, the first step is to calculate the transfer of mass due to the remap. For each remap volume a remap mass is calculated, dM_i,

$$dM_i = \rho_{remap} V_{remap}, \tag{6.5}$$

where V_{remap} is given by (6.1) and ρ_{remap} is some interpolated value of the density in the remap volume. As the present discussion is only considering first order remap,

ρ_{remap} is given by the donor cell method so that,

$$\rho_{remap} = \begin{cases} \rho_i \text{if } V_{remap,i} \leq 0 \\ \rho_{i+1} \text{if } V_{remap,i} > 0. \end{cases} \tag{6.6}$$

Having calculated the dM_i the post remap densities are calculated in a way such as to conserve mass,

$$\rho_i = \frac{(\rho V)'_i + dM_{i-1} - dM_i}{V_i}, \tag{6.7}$$

so that,

$$\sum_{cells} \rho V = \sum_{cells} \rho' V'. \tag{6.8}$$

The second step is to remap the internal energy of the cells. As this is also a cell centred variable the strategy is very similar to that of remapping density, except for the fact that rather than using volume as the independent variable, mass is now used. This means that to calculate remapped energies $d\epsilon_i$ the following formula should be used,

$$d\epsilon_i = dM_i \epsilon_{remap}, \tag{6.9}$$

where dM_i is given by (6.5). For a donor cell scheme ϵ_{remap} is calculated in a similar way to (6.6) such that,

$$\epsilon_{remap} = \begin{cases} \epsilon_i \text{if } dM_i \leq 0 \\ \epsilon_{i+1} \text{if } dM_i > 0. \end{cases} \tag{6.10}$$

The final update formula is again constructed through conservation considerations, specifically the conservation of internal energy,

$$\epsilon_i = \frac{M'_i \epsilon'_i + d\epsilon_{i-1} - d\epsilon_i}{M_i}, \tag{6.11}$$

where the masses, M_i are given by $M_i = \rho_i V_i$.
The final step is to remap the velocity, but given that velocity is a node centred variable, some intermediate steps must be carried out before following a similar method to that of remapping energy. In order to do this (in a momentum conserving manner) it is necessary to calculate the post remap nodal masses. Given that the requirement that the two sets of masses sum to give the same total over the domain,

$$\sum_{nodes} M = \sum_{cells} M, \tag{6.12}$$

this is in fact places a requirement on the recalculation of subzonal masses after remaps, to ensure consistent sums. At $t = 0$ the subzonal masses, m_i are calculated as,

$$m_i = \rho_{cell} w_i, \tag{6.13}$$

where w_i is the volume of the subzone, here denoted w to avoid confusion with the velocity. If (6.13) was applied after a remap the mass associated with the subzones, and consequently the nodal cells, may change even if the grid remains stationary. As such this method should not be used. This has introduced a requirement of the velocity remap, that a remap which does not move the nodes should not change the subzonal masses, this will be referred to as the passive remap requirement. This may seem an artificial requirement given the fact that a remap which does not move the grid at all should not need to be considered, but in cases where highly localised remaps are used it is important. The main reason this requirement is needed is that so when remapping momentum a passive remap does not change the velocities. Secondly, if subzonal masses were to be used, this would clearly change the resultant forces.

What may be apparent from considering parallels with the internal energy remap is that it is necessary to calculate dM_i values for the edge of nodal cells so that momentum can be transferred in a conservative manner. These nodal values of dM_i shall be denoted $d\tilde{M}_i$. Considering the case of a constant velocity field assuming some calculation of $d\tilde{M}_i$, the post remap velocity is given by,

$$\begin{aligned} v_i &= \frac{\tilde{M}'_i v'_i + d\tilde{M}_{i-1} v_{remap,i-1} - d\tilde{M}_i v_{remap,i}}{\tilde{M}_i} \\ &= \frac{v\left(\tilde{M}'_i + d\tilde{M}_{i-1} - d\tilde{M}_i\right)}{\tilde{M}_i}. \end{aligned} \tag{6.14}$$

Clearly this only reconstructs the constant velocity field correctly if and only if the following requirement is met,

$$\tilde{M}_i = \tilde{M}'_i + d\tilde{M}_{i-1} - d\tilde{M}_i. \tag{6.15}$$

This shall be referred to as the constant velocity field requirement. Perhaps the simplest method that passes both the passive remap and constant velocity remaps is shown below. However as the velocity is a nodal quantity, it is necessary to expand the one dimensional remap slightly, to a one dimensional remap in a two dimensional code. The first step is to redistribute the overlap masses to the edge of

nodal cells, such that,

$$d\tilde{M}_{i,j} = \frac{dM_{i,j} + dM_{i+1,j} + dM_{i,j+1} + dM_{i+1,j+1}}{4}, \tag{6.16}$$

and then to recalculate the nodal masses in a similar way to the cell centred masses (assuming only fluxes through the x-facing cell faces),

$$M_{node} = M'_{node} + d\tilde{M}_{i-1,j} - d\tilde{M}_{i,j}. \tag{6.17}$$

The subzonal masses are then calculated as,

$$m_{i,j} = \frac{m'_{i,j}}{M'_{node}} M_{node}, \tag{6.18}$$

so that the post remap subzonal mass carries the same fraction of the nodal mass post and pre remap. By construction (6.16) guarantees mass consistency between cells and nodes, and a simple application of (6.15) passes the constant velocity requirement. As for a passive remap all $d\tilde{M}_i = 0$ (6.18) passes the passive remap requirement as it leaves the subzonal masses unchanged.

The above strategy works well for a number of problems, and fulfils all the requirements previously set out. It does have the interesting side effect however that it no longer guarantees that the mass of a cell is the sum of it's subzonal masses. In some problems this scheme was found to cause subzonal masses in the same cell to vary by several orders of magnitude, subzones would preferentially acquire mass over it's neighbours, this in turn caused spikes in viscous forces which use subzonal densities. As such an improved scheme is used in Odin as follows. Firstly the mass fraction for each subzone with respect to it's cell is taken to be constant before and after remap,

$$m_{i,j} = \frac{m'_{i,j}}{M'_{cell}} M_{cell}. \tag{6.19}$$

This in fact implicitly gives a definition for the redistribution of the overlap masses to nodal cells,

$$d\tilde{M}_{i,j} = \frac{m^a_{i,j}}{M^a_{cell}} \left(dM_{i,j} + dM_{i+1,j}\right) + \frac{m^b_{i,j}}{M^b_{cell}} \left(dM_{i,j+1} + dM_{i+1,j+1}\right), \tag{6.20}$$

where the relevant cells and subcells are illustrated by figure 6.2.

Having used (6.20) to define the nodal cell remap masses each component of the velocity is remapped individually in a momentum conserving manner. For a first order scheme the remap velocity component is once again defined by the donor cell

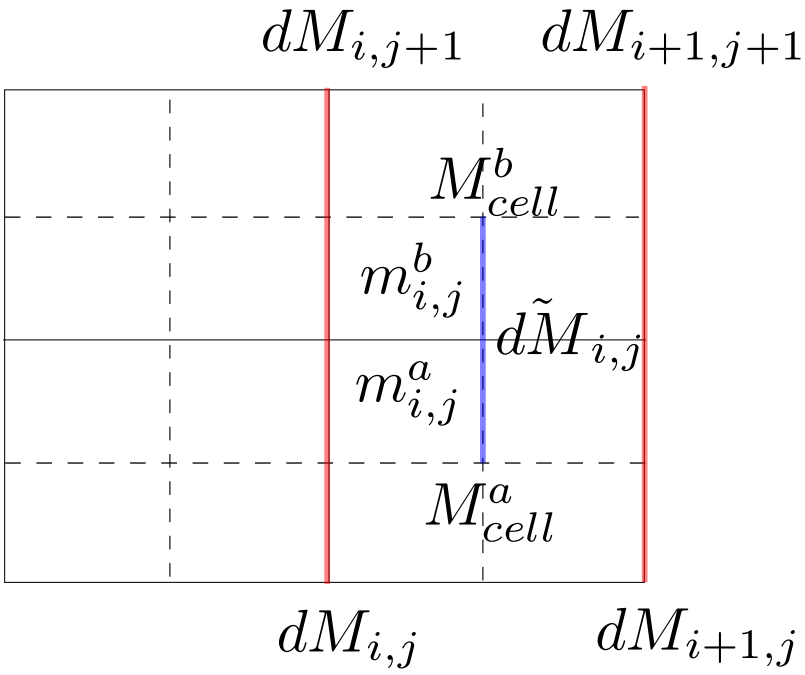


Figure 6.2: Indexing used for redistribution of remap masses to nodal cells.

method,

$$v_{x,remap} = \begin{cases} v_{x,i} \text{if } d\tilde{M}_i \leq 0 \\ v_{x,i+1} \text{if } d\tilde{M}_i > 0, \end{cases} \tag{6.21}$$

and similarly for the other components. The velocity is then updated as follows,

$$v_x = \frac{M'_{node} v'_x + dpx_{i-1} - dpx_i}{M_{node}}, \tag{6.22}$$

where dpx is the overlap x-component of the momentum, calculated as,

$$dpx_{i,j} = v_{x,remap} d\tilde{M}_{i,j} \tag{6.23}$$

6.2.2 Kinetic Energy Conservation

It is worth pointing out that the scheme described above whilst conserving momentum exactly, does not exactly conserve kinetic energy. It is possible to account for this loss of kinetic energy by calculating the total kinetic energy before and after the remap. This lost kinetic energy can then be deposited in the form of heat back into the fluid, thus restoring energy conservation, as implemented in [10] for example. This is not a problem unique to kinetic energy, a similar problem is encountered in

remapping the B-field, where although the remap is formulated to preserve the divergence (free nature) of the B-field some loss of magnetic energy is seen. A similar correction was used in [33]. These methods are not currently implemented in Odin.

6.3 Swept Region Based Remaps

Having explained a basic remapping strategy it now remains to decide firstly given a new grid how remap volumes, and thus remapped quantities are transferred between cells, and secondly how to calculate a new grid. The first, and simplest, option for transferring material is a swept region based remap.

A swept region based remap (e.g. [34]) transfers material between a cell and it's four neighbours with which it shares and edge, or face in three dimensions. The remap, or overlap, volumes are calculated by first calculating the area of the quadrilateral formed by two pairs of new and old node positions, thus defining an overlap area for each edge, which results in a volume by multiplying by either $dz = 1$ or 2π for Cartesian or cylindrical coordinates.

However, it remains to decide upon a sign convention. This is based upon the previous arguments. As the problem must now be considered as two dimensional, the overlap volumes are now labelled with a subscript of either x or y. This refers to the logical grid direction, not necessarily some alignment with the coordinate directions. However for the moment assume that the logical and coordinate directions are approximately aligned. Expanding the one dimensional case, the mass is now assumed to update as,

$$M_{i,j} = M'_{i,j} + dM^x_{i-1,j} - dM^x_{i,j} + dM^y_{i,j-1} - dM^y_{i,j}. \tag{6.24}$$

The signed area of a quadrilateral is calculated as,

$$A = \frac{1}{2}\left[(x_3 - x_1)(y_4 - y_2) - (x_4 - x_2)(y_3 - y_1)\right], \tag{6.25}$$

which will be positive if the nodes are arranged in a counterclockwise manner, or negative if arranged clockwise. Considering the case shown in 6.3. Here the post remap grid is slightly displaced from the pre-remap grid. Using the sign convention of (6.24) the nodes of the overlap volume in the x-direction are numbered as,

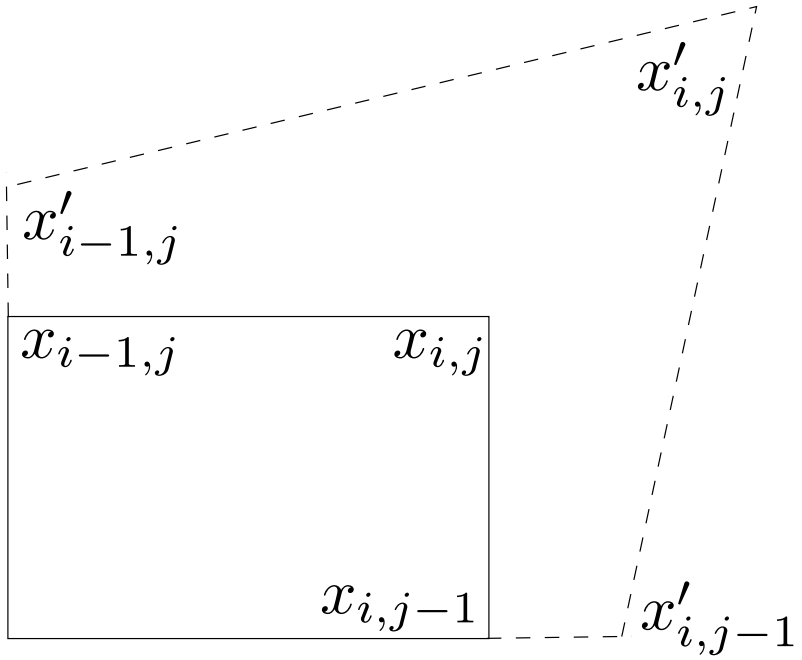


Figure 6.3: Two dimensional remap, the post-Lagrangian grid is shown by a dashed line, the post-remap grid by a solid line. x and y share the same indexing, however only x is shown for clarity.

$$\begin{aligned} x_1 &= x_{i,j-1} \\ x_2 &= x'_{i,j-1} \\ x_3 &= x'_{i,j} \\ x_4 &= x_{i,j} \end{aligned} \tag{6.26}$$

and for the y-direction overlap volume as,

$$\begin{aligned} x_1 &= x_{i,j-1} \\ x_2 &= x_{i,j} \\ x_3 &= x'_{i,j} \\ x_4 &= x'_{i,j-1}. \end{aligned} \tag{6.27}$$

Corresponding definitions apply for the y-coordinates. Positions denoted as x' represent node positions pre-remap and x post remap.

For some remaps that are slightly more complex than the remap denoted in fig-

ure 6.3 due to the simplified nature of a swept region based remap some regions get misplaced. However, these misplacements occur in pairs, and act to cancel, so that the volumes remain consistent. This does mean that the transfer of mass is not what might be expected by a simple overlaying of the grids, this is one of the key differences between swept region based remaps, and the intersection based remaps described in the following section. In the case demonstrated by figure 6.4 there are six regions which may be considered for transfer, as labelled. For a swept region based remap the following transfer of areas occur,

$$a_1 + a_5 \text{ Cell 1} \longrightarrow \text{Cell 3}, \tag{6.28}$$

$$a_2 + a_5 \text{ Cell 2} \longrightarrow \text{Cell 1}, \tag{6.29}$$

$$a_3 + a_6 \text{ Cell 4} \longrightarrow \text{Cell 3}, \tag{6.30}$$

and,

$$a_4 + a_6 \text{ Cell 2} \longrightarrow \text{Cell 4}. \tag{6.31}$$

In the first of these equations it can be seen from inspecting figure 6.4 that cell 1 loses too much area (volume), as pre-remap A_5 was in cell 2 rather than cell 1. A_5 however finishes correctly in cell 3. This apparent loss of volume is accounted for by the second exchange, where cell 1 gains too much area, as it gains $A_2 + A_5$ despite the fact that A_5 is in the post remap cell 3. A corresponding process occurs in exchanges three and four, where in the third exchange cell 4 incorrectly loses A_6, but then gains it back (again incorrectly) in exchange four.

This process occurs due to the fact that only cells which share edges exchange information. This limitation of swept region based remaps result in inaccurate results, particularly in the case of circular wave fronts, causing a squaring off process. This problem can be avoided in two ways. The simplest is a split remap, which was one of the improvements implemented by Dr. C. S. Brady and is explained in a later chapter. The second way is by use of an intersection based remap, which is explained in the following section.

6.4 Intersection Based Remaps

Intersection based remaps (e.g. see discussion in [35] are considerably more complicated than the previously described scheme. In this scheme, as well as exchanging information with the 2 neighbouring cells in each dimension, a cell also interacts with it's diagonal neighbours. In terms of a finite difference stencil, in two dimen-

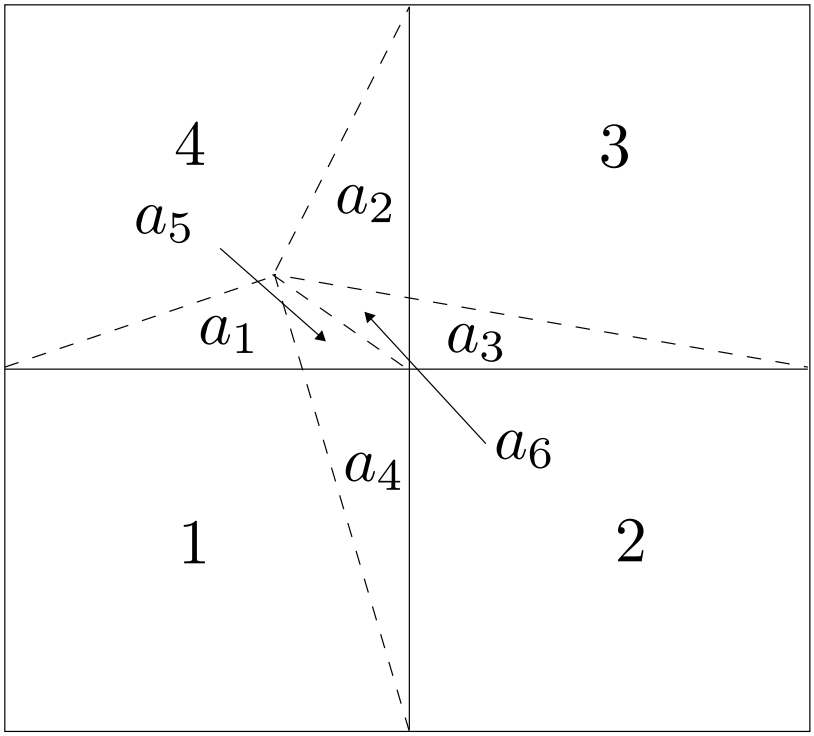


Figure 6.4: Remapping illustrating double counting of overlap areas for swept region based remap.

sions it increased from four cells to eight, and in three dimensions, from six cells to ten. Despite it's increased computational complexity, it is perhaps the more intuitive from initial consideration.

As is implied by the name, an intersection based remap is carried out by calculating the areas (or volumes) formed by the intersection of old and new grids. The overlap areas are formed by overlaying the old and new grids and transferring the polygons that are formed by the intersection of the two grids to the cell in which they lie in the new grid. Of course the immediate added computational cost of finding multiple intersection points is apparent, however no (incorrect) double counting of volumes occurs, as for the swept region based remap. For the case of a donor cell method, it is then necessary to calculate which cell the centre of the overlap volume was in, again this is more complicated than the donor cell scheme for swept region based remaps. Odin uses a swept region based remap, however it is insightful to compare the two methods in some simple geometries.

6.4.1 Equivalence to a Swept Region Based Remap in One Dimension

As mentioned in the introduction to this chapter, the two remapping schemes are equivalent in a one dimensional case. This is clearly seen by considering the diagram of such a remap; the volume for each remapping scheme is the same. This occurs for any remap with which the grid motion is aligned with the original grid. With this in mind, it is clear why swept region based remaps struggle along the 45 degree line of a radial flow, whereas intersection remaps do not. For a radial flow, along the axes the grid is aligned with the flow, and thus the two remapping schemes are equivalent. However they deviate in increasing amounts when moving away from the axes. At 45 degree the flow is completely misaligned with the grid, and there is a strong interaction between diagonal neighbours. As such here an intersection based remap is expected to perform much more strongly than a swept region based remap.

6.4.2 Hybrid Remapping Strategies

Intersection based remaps are often preferred for multi-material cases. This opinion is usually justified by the idea that an intersection based remap always allows remap volumes of single material to be constructed, assuming knowledge of the material interface. In principal at least, assuming the material interface lies on the mesh, this allows for the interface to be tracked exactly; no interface reconstruction method

is needed. The intersection of the old material interface with the new grid is automatically calculated during a multi-material intersection based remap, and this information of the interface can be retained. In practice this is quite an added computational cost, but an intersection based remap remains popular for multi-material methods, even with interface reconstruction.
Hybrid remapping strategies [36] [37] are remapping schemes which identify single material regions and remap these using the simplified swept-region based remap, whereas regions which are identified as having more than one material, are remapped using a intersection based remapping scheme.

6.5 Remapping within Odin

Odin uses a swept region based remapping scheme. An intersection based remap has two advantages, information transfer between diagonal neighbours, and some (possible) advantage for multi-material. However as discussed this comes at a significant extra computational cost, a cost which hybrid remapping schemes attempt to alleviate. This improvement for multi-material is unclear, it may even cause excessive stiffening of the interface. A swept region based remapping scheme was chosen for Odin on the basis that the diagonal information transfer problem could be simply solved by using a split remap. This was implemented by Dr C. S. Brady, and is described in a later chapter. With this in mind the intersection remap had only some (somewhat unproven) advantage for multi-material, still at greatly increased computational cost; swept region based remaps seemed an obvious choice.

6.5.1 Remapping Strategy

The topic of remapping strategy is two fold, when to remap, and where to remap to. The question of when to remap can be answered either by some automatic remapping criteria, or by a pre-defined remap strategy. Currently Odin employs the latter strategy where the user can add a pre-defined remap strategy after a certain time. Recently Dr. C. S. Brady extended this capability so that the user can add multiple, possibly different, remap strategies for different times. The former option usually depends on geometric considerations, for example limiting cell aspect ratios, or the internal angle of cells. This is discussed throughout the literature, for example see the discussion within [11]. The question of where to remap to is also a rather open ended question, with no particular, single best answer for all applications. However within Odin two simple grid movement strategies are implemented. The first is to simply calculate the new grid by subtracting a fraction of the product of

the velocity and time step,

$$x = x' - C\Delta t v_x, \tag{6.32}$$

where $0 \leq C \leq 1$ is a constant which controls the grid movement. Clearly for $C = 0$ the grid movement is purely Lagrangian, whereas for $C = 1$ the grid movement is purely Eulerian. Alternatively Odin also employs an equipotential remap, again a popular strategy within ALE codes. Developed in [38], it is a method to automatically calculate a smooth mesh. The method consists of assigning to potentials, ϕ and ψ to each mesh coordinate. A smooth grid is then calculated by solving,

$$\nabla^2 \psi = 0, \tag{6.33}$$

and,

$$\nabla^2 \phi = 0. \tag{6.34}$$

These equations can be inverted, to yield the following system,

$$\alpha \frac{\delta^2 x}{\delta \phi^2} - 2\beta \frac{\delta^2 x}{\delta \phi \delta \psi} + \gamma \frac{\delta^2 x}{\delta \psi^2} = 0, \tag{6.35}$$

and,

$$\alpha \frac{\delta^2 y}{\delta \phi^2} - 2\beta \frac{\delta^2 y}{\delta \phi \delta \psi} + \gamma \frac{\delta^2 y}{\delta \psi^2} = 0. \tag{6.36}$$

Here,

$$\alpha = \left(\frac{\delta x}{\delta \psi}\right)^2 + \left(\frac{\delta y}{\delta \psi}\right)^2, \tag{6.37}$$

$$\beta = \frac{\delta x}{\delta \psi}\frac{\delta x}{\delta \phi} + \frac{\delta y}{\delta \psi}\frac{\delta y}{\delta \phi}, \tag{6.38}$$

and,

$$\gamma = \left(\frac{\delta x}{\delta \phi}\right)^2 + \left(\frac{\delta y}{\delta \phi}\right)^2. \tag{6.39}$$

The next step is to make the following finite difference approximations,

$$\frac{\delta^2 x}{\delta \phi^2} = x_{\phi-1,\psi} - 2x_{\phi,\psi} + x_{\phi+1,\psi}, \tag{6.40}$$

$$\frac{\delta^2 y}{\delta \phi^2} = y_{\phi-1,\psi} - 2y_{\phi,\psi} + y_{\phi+1,\psi}, \tag{6.41}$$

$$\frac{\delta^2 x}{\delta \psi^2} = x_{\phi,\psi-1} - 2x_{\phi,\psi} + x_{\phi,\psi+1}, \tag{6.42}$$

$$\frac{\delta^2 y}{\delta\psi^2} = y_{\phi,\psi-1} - 2y_{\phi,\psi} + y_{\phi,\psi+1}, \tag{6.43}$$

$$\frac{\delta^2 x}{\delta\phi\delta\psi} = \frac{1}{4}\left(-x_{\phi+1,\psi-1} + x_{\phi+1,\psi+1} - x_{\phi-1,\psi+1} + x_{\phi-1,\psi-1}\right), \tag{6.44}$$

and,

$$\frac{\delta^2 y}{\delta\phi\delta\psi} = \frac{1}{4}\left(-y_{\phi+1,\psi-1} + y_{\phi+1,\psi+1} - y_{\phi-1,\psi+1} + y_{\phi-1,\psi-1}\right). \tag{6.45}$$

Define α, β, γ as,

$$\alpha = \left(x_{\phi,\psi+1} - x_{\phi,\psi-1}\right)^2 + \left(y_{\phi,\psi+1} - y_{\phi,\psi-1}\right)^2, \tag{6.46}$$

$$\begin{aligned}\beta = \frac{1}{2}\big(&\left(x_{\phi,\psi+1} - x_{\phi,\psi-1}\right)\left(x_{\phi+1,\psi} - x_{\phi-1,\psi}\right) \\ &+ \left(y_{\phi,\psi+1} - y_{\phi,\psi-1}\right)\left(y_{\phi+1,\psi} - y_{\phi-1,\psi}\right)\big),\end{aligned} \tag{6.47}$$

and,

$$\gamma = \left(x_{\phi+1,\psi} - x_{\phi-1,\psi}\right)^2 + \left(y_{\phi+1,\psi} - y_{\phi-1,\psi}\right)^2. \tag{6.48}$$

These finite difference approximations can be combined to yield equations for x and y,

$$\begin{aligned}x = \frac{1}{2\left(\alpha+\gamma\right)}\big[&\alpha\left(x_{\phi+1,\psi} + x_{\phi-1,\psi}\right) \\ &+ \gamma\left(x_{\phi,\psi+1} + x_{\phi,\psi-1}\right) \\ &+ \beta\left(x_{\phi+1,\psi-1} - x_{\phi+1,\psi+1} + x_{\phi-1,\psi+1} - x_{\phi-1,\psi-1}\right)\big],\end{aligned} \tag{6.49}$$

and,

$$\begin{aligned}y = \frac{1}{2\left(\alpha+\gamma\right)}\big[&\alpha\left(y_{\phi+1,\psi} + y_{\phi-1,\psi}\right) \\ &+ \gamma\left(y_{\phi,\psi+1} + y_{\phi,\psi-1}\right) \\ &+ \beta\left(y_{\phi+1,\psi-1} - y_{\phi+1,\psi+1} + y_{\phi-1,\psi+1} - y_{\phi-1,\psi-1}\right)\big].\end{aligned} \tag{6.50}$$

These equations can be applied iteratively to calculate a new grid,

$$\begin{aligned}x^{m+1} = \frac{1}{2\left(\alpha+\gamma\right)}\big[&\alpha\left(x^m_{\phi+1,\psi} + x^m_{\phi-1,\psi}\right) \\ &+ \gamma\left(x^m_{\phi,\psi+1} + x^m_{\phi,\psi-1}\right) \\ &+ \beta\left(x^m_{\phi+1,\psi-1} - x^m_{\phi+1,\psi+1} + x^m_{\phi-1,\psi+1} - x^m_{\phi-1,\psi-1}\right)\big],\end{aligned} \tag{6.51}$$

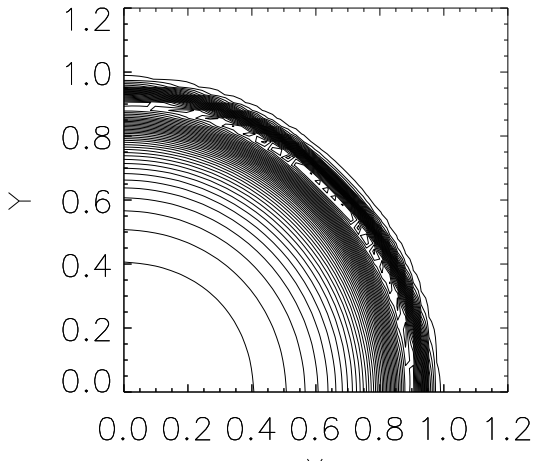


Figure 6.5: Density contour plot for Sedov's problem, using a first order remap.

and,

$$y^{m+1} = \frac{1}{2(\alpha+\gamma)}\left[\alpha\left(y^m_{\phi+1,\psi} + y^m_{\phi-1,\psi}\right)\right.$$
$$+ \gamma\left(y^m_{\phi,\psi+1} + y^m_{\phi,\psi-1}\right)$$
$$\left. + \beta\left(y^m_{\phi+1,\psi-1} - y^m_{\phi+1,\psi+1} + y^m_{\phi-1,\psi+1} - y^m_{\phi-1,\psi-1}\right)\right], \tag{6.52}$$

where in (6.51) and (6.52) α, β, γ are evaluated at iteration m. An important feature of this grid movement strategy is that β is proportional to the dot product of the mesh segments, ϕ and ψ, so $\beta = 0$ is the grid is orthogonal, and if the grid is nearly orthogonal this grid movement strategy will act to make the grid closer still to being orthogonal. Although this is an iterative method within Odin the default method is to apply a single iteration only, as this has been found to provide sufficient grid relaxation. As discussed in [11] it is possible to further refine this method by applying weights to specific grid points, to force the resolution to be greater in some regions, but this is not currently implemented within Odin.

6.6 Results

To briefly assess the implementation of the first order remap Sedov's problem shall be re-run. This problem is of particular interest as the pure Lagrangian scheme failed to complete this problem using a Cartesian mesh. The results are shown in figures 6.5 and 6.6. The contour plot, figure 6.5 shows a strong degree of

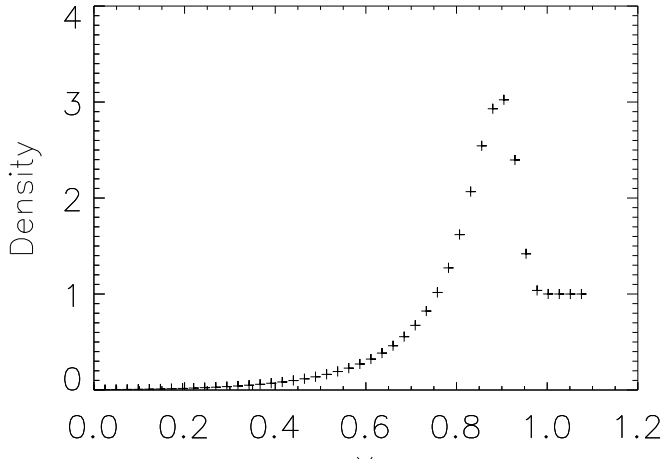


Figure 6.6: Line plot of the density obtained along $y = 0$ for Sedov's problem using a first oder remap.

symmetry maintained over the domain. However the line plot, figure 6.6, shows that the shock speed has been miscalculated, and the peak density has been spread over a larger number of cells than seen in the calculations using a polar grid, or subzonal pressures.

6.7 Summary

A first order remapping method has been discussed and implemented. Whilst the test results are promising, and show how the addition of a remapping step can increase the robustness of the code the answer lacks an acceptable level of accuracy, and as such higher order remapping methods shall be sought.

Chapter 7

Ideal MHD in Cartesian Coordinates

7.1 Introduction

There exists limited published material on the use of Arbitrary Lagrangian Eulerian codes for MHD simulations. When writing an MHD algorithm it is imperative to conserve the divergence free nature of the magnetic field. When extending this principle to ALE methods it amounts to constraining the magnetic field to be divergence free in both the Lagrangian step, and any potential remapping step. Initial conditions (and their evolution) that do not obey the solenoidal condition aren't physically correct; thus the requirement reduces to conservation of the divergence. Should the field be divergence free at $t = 0$, then it should remain so. Divergence is calculated as the sum of discrete fluxes through mesh segments. The most natural choice is to use the cell faces as these surfaces, thus making $\nabla \cdot \vec{B}$ a cell centred quantity. This mirrors the methodology of many (intrinsically divergence preserving) staggered grid codes. Driven partly by this, and partly to work in terms of conserved quantities it is desirable for the stored quantity within the code to be magnetic flux through faces.

However, in order to carry out the momentum update, it is necessary to calculate a cell centred value of the magnetic field, at the end of the predictor step. Thus an apparent contradiction appears, in that the momentum update requires a cell centred magnetic field, whereas conventional divergence conserving remap strategies, require face centred quantities.

It may appear logical to consider first the Lagrangian evolution of the full MHD equations, however the exact details of this process will be dictated by the place-

ment of the variables on the grid, recalling the need for a cell centred magnetic field only applies at predictor level. At integer time step levels, the position of magnetic field variables are decided by the choice of divergence conserving remapping strategy, and as such the question of the remap will be addressed first. It is also worth noting, that to keep this discussion separate from other numerical issues, for now the remap will be limited to first order only. Numerically there are two approaches to obeying the solenoidal condition; schemes which intrinsically keep the B-field divergence free, or those which seek to correct any violation. Constrained transport [39] is a remapping technique which intrinsically obeys the solenoidal condition. Alternatively a divergence cleaning scheme may be used. A divergence cleaning scheme (early examples include [40], [41])is applied at the end of a time step, and acts to remove the portion of the B-field which violates the solenoidal condition, and it is this technique that shall be considered first.

7.2 Divergence Cleaning Schemes

In an effort to examine differences between staggered grid and divergence cleaning schemes Balsara and Kim [42] conducted a review of divergence cleaning schemes. In order to assess the viability of divergence cleaning schemes their discussion and some key results will be summarised. Although their work was within the context of Godunov solvers, some of the conclusions made about divergence cleaning schemes are still relevant here.

Divergence cleaning schemes may be split into two distinct types, scalar and vector divergence cleaners. Balsara and Kim [42] look at two different scalar divergence cleaners, as well as a vector divergence cleaner. The first scalar divergence cleaner (SDC1) is formulated as follows, and is applied at the end of every step. Firstly write the modified (divergence free) B-field (B') in terms of the original B-field (B), and a scalar, ϕ,

$$\begin{aligned} B'_{x,i,j,k} &= B_{x,i,j,k} - \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}; \\ B'_{y,i,j,k} &= B_{y,i,j,k} - \frac{\phi_{i,j+1,k} - \phi_{i,j-1,k}}{2\Delta y}; \\ B'_{z,i,j,k} &= B_{z,i,j,k} - \frac{\phi_{i,j,k+1} - \phi_{i,j,k-1}}{2\Delta z}; \end{aligned} \tag{7.1}$$

It can then be shown that if B' satisfies the discrete divergence condition,

$$\frac{B'_{x,i+1,j,k} - B'_{x,i-1,j,k}}{2\Delta x} + \frac{B'_{y,i,j+1,k} - B'_{y,i,j-1,k}}{2\Delta y} + \frac{B'_{z,i,j,k+1} - B'_{z,i,j,k-1}}{2\Delta z} = 0, \tag{7.2}$$

if ϕ obeys,

$$\frac{\phi_{i+2,j,k} - 2\phi_{i,j,k} + \phi_{i-2,j,k}}{4\Delta x^2} + \frac{\phi_{i,j+2,k} - 2\phi_{i,j,k} + \phi_{i,j-2,k}}{4\Delta y^2} + \frac{\phi_{i,j,k+2} - 2\phi_{i,j,k} + \phi_{i,j,k-2}}{4\Delta z^2}$$
$$= \frac{B_{x,i+1,j,k} - B_{x,i-1,j,k}}{2\Delta x} + \frac{B_{y,i,j+1,k} - B_{y,i,j-1,k}}{2\Delta y} + \frac{B_{z,i,j,k+1} - B_{z,i,j,k-1}}{2\Delta z}. \tag{7.3}$$

Thus the problem of removing the solenoidal condition violating part of the B-field has been reduced to solving a Poisson equation, as defined by (7.3). Balsara and Kim [42] list a number of advantages and disadvantages of the scheme. The key advantage of this scheme is that it is exact. Whilst (7.3) represents a Poisson problem, it is one derived by the exact application of the solenoidal constraint. However it is hampered by odd even decoupling. It is clear from (7.3) that odd and even ϕ's are decoupled from each other, and in fact represent (for a three dimensional problem) eight $(N/2)^3$ solutions to Poisson problems. There is also a need for an all to all communication, and most importantly perhaps (as with any Poisson solve) the system is now non-local. Explicitly, should one region develop a non divergence free solution this will affect the solution everywhere; there is now effectively information propagating at an infinite speed. This divergence cleaning scheme may be summarised as exact, with odd-even decoupling.

Having summarised the first divergence cleaning scheme of Balsara and Kim it is now worthwhile summarising the underlying methodology behind divergence cleaners. Divergence cleaning is essentially taking a Hodge projection of the B-field. A vector field may be uniquely decomposed into two parts, a solenoidal part, and irrotational part. Thus the post Lagrangian B-field, $\vec{B}_L$, may be rewritten as,

$$\vec{B}_L = \vec{B}' + \nabla\phi, \tag{7.4}$$

where $\vec{B}'$ represents the solenoidal part (the part of the post Lagrangian field that obeys the solenoidal condition) and $\nabla\phi$ is the irrotational part as $\nabla \times \nabla\phi = 0$ for all scalars ϕ. Thus to secure a divergence free B-field all that remains is to solve for $\nabla\phi$. To do this take the divergence of (7.4),

$$\nabla \cdot \vec{B}_L = \nabla^2\phi, \tag{7.5}$$

which is a Poisson equation. All that remains is to solve (7.5) for ϕ and use (7.4) to solve for $\vec{B}'$.

As an aside it is worth pointing out that due to the definition provided by (7.4), $\nabla \times \vec{B}_L = \nabla \times \vec{B}'$ and as such divergence cleaning does not change the MHD current,

$\vec{j} = 1/\mu_o \nabla \times \vec{B}$, although it does still change the $\vec{j} \times \vec{B}$ force.
With this in mind it is possible to formulate different variations of the previously defined scheme. One such version is the second scalar divergence cleaning scheme described by Balsara and Kim, [42]. This scheme, SDC2, differs from the previous one in that (7.3) is replaced by,

$$\frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{\Delta x^2} + \frac{\phi_{i,j+1,k} - 2\phi_{i,j,k} + \phi_{i,j-1,k}}{\Delta y^2} + \frac{\phi_{i,j,k+1} - 2\phi_{i,j,k} + \phi_{i,j,k-1}}{\Delta z^2}$$
$$= \frac{B_{x,i+1,j,k} - B_{x,i-1,j,k}}{2\Delta x} + \frac{B_{y,i,j+1,k} - B_{y,i,j-1,k}}{2\Delta y} + \frac{B_{z,i,j,k+1} - B_{z,i,j,k-1}}{2\Delta z}. \tag{7.6}$$

The remaining equations from the first scheme remain unchanged. Clearly what defines a divergence cleaning method is the choice of finite difference template for $\nabla^2\phi$. This second scheme shares the advantages and disadvantages of the previous scheme, with the following exceptions. Clearly, from (7.6) this scheme does not suffer from odd-even decoupling. However it is no longer exact. This is because the finite difference template has not been derived to exactly obey the finite solenoidal condition. It is also worth pointing out that repeated applications of this scheme will result in different B-fields. This scheme may be summarised as non-exact, but without odd-even decoupling.
The remaining scheme describe by Balsara and Kim is a vector divergence cleaner. This scheme works by transferring the three components of the B-field into Fourier space, and correcting for non-zero divergence. This is easily done as the divergence operator in real space transforms to the dot product between $\vec{k}$ and $\vec{B}(k)$ in Fourier space, thus a divergence free (in Fourier space) field can be calculated,

$$B_i\left(\vec{k}\right) = \sum_{j=1}^{3} \left(\delta_{i,j} - \frac{k_i k_j}{\vec{k}^2}\right) B_j\left(\vec{k}\right). \tag{7.7}$$

Clearly the resultant field from (7.7) has zero component parallel to the wave vector, and is thus divergence free in Fourier space. To complete the divergence cleaning step this field is transformed back into real space.
The main advantage of this scheme over SDC1 is that it is not susceptible to odd-even decoupling. Unlike SDC2 this scheme is exact in Fourier space, which also adds the further advantage over SDC2 in that repeated applications of this scheme does not change the divergence in real space, so it does not suffer from the ambiguous nature of SDC2.
This scheme is though, like SDC2, not exact in physical space, due to the differing definitions of the divergence operator in Fourier and real space. It is also compu-

tationally expensive (even compared to SDC1 and SDC2), needing three times as many fast Fourier transforms as the scalar schemes. It should be stressed also that this scheme is still a Poisson solving one, and as such still suffers from the associated non-local solution. A further disadvantage is that it requires all-to-all communication in parallel, again requiring three times as much as SDC1 and SDC2. The most pertinent disadvantage to this discussion however is that the scheme cannot be extended to non-Cartesian domains. This is a problem that could be rectified in the context of ALE codes by transforming to an orthogonal space, such as the a-grid used in the Cauchy solution of the B-field (see subsection 7.5.2), but this would represent yet further computational cost. To summarise whilst multiple variants of divergence cleaners exist, each with their own individual advantages and disadvantages, there are some recurring themes. All methods involve a Poisson solve, and as such are computationally expensive, and introduce a non-physical non-local part to the system. This action at a distance causes a divergence free violation in one part of the domain to lead to action at a distance, thus changing the result elsewhere. This discussion reaffirms the previous point, that divergence cleaners are non-physical, computationally expensive methods used to correct for previous failings. Should those previous failings be avoidable, then divergence cleaners should not be seen as an option.

7.3 Conventional Remapping Strategies

Constrained transport, [39], is a widely used method which enforces a divergence conserving evolution of the magnetic field. Originally formulated for fixed grid codes, it has also been adapted for the remap phase of Lagrangian remap codes, [10], by using a remap velocity in place of the fluid velocity (relative to the grid) used by [39]. The original method involved relativistic considerations (not relevant here), and an arbitrary grid motion relative to the fluid, however it is most clearly explained by considering the case of a stationary (Eulerian) grid. It is also worth noting that in [39] a different staggering is used; velocity components are defined at face centres, and magnetic fluxes (and corresponding components) are stored at the midpoints of what is effectively the median mesh. The algorithm is easily switched to the staggering used in *Odin*, and it is that context which shall be used to explain the method.

The magnetic flux through a surface, $\vec{S}$ is given by:

$$\phi = \int_s \vec{B} \cdot d\vec{S}, \tag{7.8}$$

where S is the grid face being considered. Considering now the change in that quantity as the fluid moves through the grid, and using the Leibniz integral rule (for two dimensions),

$$\frac{d}{dt}\phi = \int_S \frac{\partial \vec{B}}{\partial t} \cdot d\vec{S} + \oint_{\partial S(t)} \vec{B} \cdot \vec{v}_c \times d\vec{r}, \tag{7.9}$$

where ∂S is the bounding contour of the surface, S, and $\vec{v}_c$ is the velocity of the line element $d\vec{r}$. Enforcing the grid to be Eulerian, thus setting $\vec{v}_c = 0$, and assuming ideal MHD

$$\frac{d}{dt}\phi = \int_S \frac{\partial \vec{B}}{\partial t} \cdot d\vec{S} = \int_S \nabla \times \left(\vec{v} \times \vec{B}\right) d\vec{S}. \tag{7.10}$$

Finally, the application of Stokes theorem yields,

$$\frac{d}{dt}\phi = \oint_{\partial S} \left(\vec{v} \times \vec{B}\right) \cdot d\vec{r}, \tag{7.11}$$

where ∂S is (once again) the bounding contour of the surface, S. In order to carry out this integral it is necessary to define where all points are on the grid, including in the ignorable direction. Thus, a third index, k, is needed. The positioning, and indexing of cells, nodes and edges are defined by figure 7.1. The change in flux due to (7.11) is calculated by carrying out the integral along the edges in right handed sense.

It is now possible to write the change in flux through a cell face as the sum of the emfs calculated along each edge, as defined by the discrete parts of (7.11).

$$\frac{d\phi_i}{dt} = \sum_{l=1}^{4} \epsilon_l. \tag{7.12}$$

These are not the full emfs (which always equals zero for ideal MHD, since $\epsilon = \int \left(\vec{E} + \vec{v} \times \vec{B}\right) .d\vec{l})$, just one contribution to it. However for the sake of brevity, and to aid comparison between the method described and Evans and Hawleys' [39] original method these discrete parts will be referred to as emfs.

Consider now, the left most face of figure 7.1, defined by $(i - 1/2, j, k)$. The two edge contributions from $(i - 1/2, j, k - 1/2)$ and $(i - 1/2, j, k + 1/2)$ are equal in magnitude due to symmetry (either in the z direction for Cartesian, or theta for cylindrical) but opposite in sign due to the opposite direction along which the integration is carried out, and thus cancel. Whilst demonstrated for a face with fixed i-coordinate, a similar cancellation of two (albeit different edges) occurs for

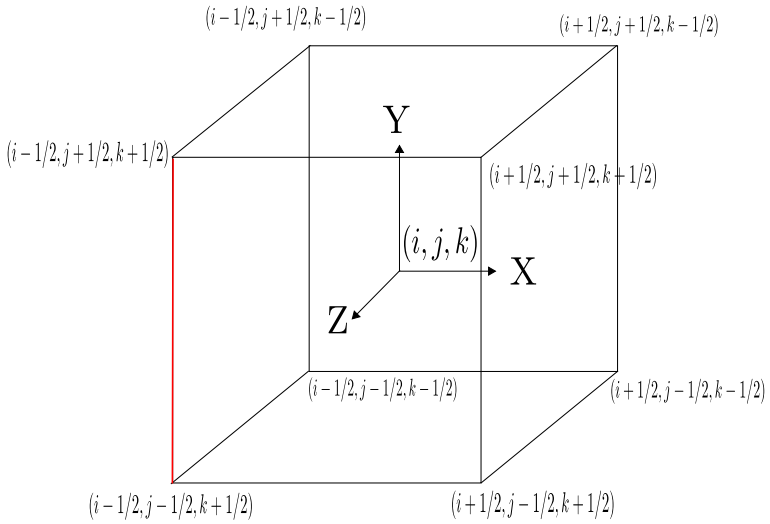


Figure 7.1: Indexing used for a three dimensional cell. The velocity is defined at the node. In order to carry out a constrained transport step it is necessary to interpolate the magnetic field and the velocity to the same point, although they may be stored in separate locations during the Lagrangian phase. Edges are defined by the midpoint of the varying index, for example the left most edge (highlighted red) in this diagram is denoted $(i - 1/2, j, k - 1/2)$.

faces with fixed j-coordinate. The emf contribution due to edge $(i - 1/2, j - 1/2, k)$ can be written as:

$$\epsilon_{i-1/2,j-1/2,k} = \int_{z(k+1/2)}^{z(k-1/2)} (v_x B_y - v_y B_x)\, dz. \tag{7.13}$$

As $\vec{v}$ and $\vec{B}$ do not vary along the ignorable direction the emf of that edge can be written as:

$$\epsilon_{i-1/2,j-1/2,k} = -\left(v_x \bar{B}_y - v_y \bar{B}_x\right) \Delta z, \tag{7.14}$$

where $\bar{B}_i$ represents an interpolated value for the B-field i-th component at the node. The minus sign in (7.14) has appeared due to the ordering of the integration limits in (7.13), and the actual displacement of the line over which the integral has been carried out is $-\Delta z$. Various ways exist to calculate $\bar{B}_i$, but the choice is not important for explaining the algorithm. Repeating this argument from the other

contributing edge, and dropping the (now obsolete) third index,

$$\left(\frac{d\phi}{dt}\right)_{i-1/2,j} = \left[-\left(v_x\bar{B}_y - v_y\bar{B}_x\right)^{i-1/2,j-1/2} + \left(v_x\bar{B}_y - v_y\bar{B}_x\right)^{i-1/2,j+1/2}\right]\Delta z. \tag{7.15}$$

This argument can be repeated for all faces of the cell, and it is easy to see, that due to the shared nodes but opposing directions of integration of each face that the total flux change through the faces of each cell is zero. It is worth examining constrained transport in a more physical sense to explain this.

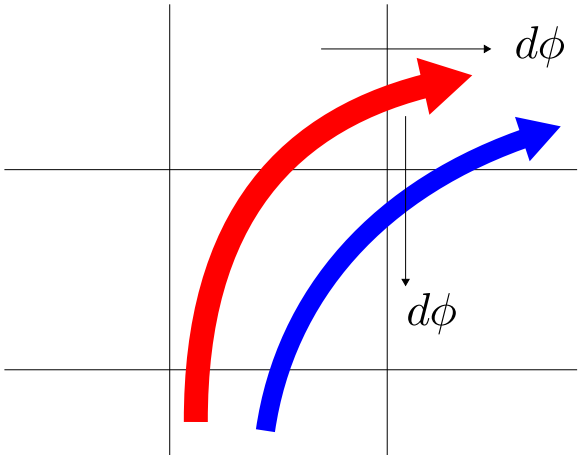


Figure 7.2: Flux tube moving through a stationary grid at $t = 0$ (red) and one time step later, blue.

Consider the layout as shown in figure 7.2, where initially there is a tube of flux, denoted by the red arrow, which during the time step moves to it's new position at the end of the time step, denoted by the blue arrow. Clearly as the flux tube has moved through the grid there must be a change in flux through the top of the main cell illustrated as that flux is transferred to it's neighbour in the positive x-direction, and a corresponding, and identical flux must be transferred in the y-direction as illustrated. It is clear that during this flux transport the divergence of the magnetic field has been conserved, and this is the physical process Evans and Hawley's constrained transport models [39]. The adaption of this method for a remap phase is simple; having calculated the new grid, calculate an effective velocity remap, where,

$$\vec{v}_{remap}\Delta t = \left(\vec{x}_{new} - \vec{x}_{old}\right). \tag{7.16}$$

Here $\vec{x}_{new}$ refers to the post remap position, and $\vec{x}_{old}$ the post Lagrangian velocity. In fact, the negative of this velocity must be used as it is now the grid which is

moving, not the fluid. There is of course a second consideration when adapting this algorithm for a remap phase. What has been implicit in the preceeding discussion is that the B-field needs to be interpolated to the point at which the velocity vector is defined, when using this scheme to evolve a B-field on a fixed grid, as with the original Evans and Hawley implementation, [39]. However when being used to remap a B-field, it is necessary to interpolate the B-field to the midpoint of the remap vector, i.e. to the midpoint of the old and new node positions.

7.3.1 Out of Plane Magnetic field Component

For a two-dimensional code remapping the ignorable direction component of the magnetic field requires no special treatment in terms of maintaining the divergence of the magnetic field. Due to this the third component of the magnetic field is usually stored as a cell centred variable, as such the evolution of the ignorable direction component is separated from the evolution of the two in plane components. As such it can be remapped in the same manner as any other cell centred variable, as shown in figure 7.3.
This scheme can, like previously described non-split remaps, be extended to higher order. However, as discussed in the chapter on second order remapping methods this method does have some difficulties and discrepancies, and as such for higher order remaps a directionally split remap is preferable.

7.4 Cell Centred Based Remaps

Both the original constrained transport algorithm [39] and the adaption described above use magnetic field components located at different points on the grid. Components in the non-ignorable direction are given at cell faces, the ignorable direction component is a cell centred quantity. Due to the requirement of a cell centred magnetic field for all components in the Lagrangian step (at least at predictor level) it is worth considering the possibility of a constrained transport remap that uses cell centred magnetic field components. Using such a method the most natural choice for the positioning of the divergence of the magnetic field would be to make it a nodal quantity and for it to be calculated in terms of fluxes through the median mesh, as previously defined by the connection of midpoints of the primary grid, as defined by the nodes. Constrained transport remaps are not limited to the primary grid, nor are they limited to quadrilateral grids. In this sense, constrained transport remaps are well suited to ALE schemes, in that as long as proper connectivity is maintained (i.e. cells are properly defined), a constrained transport remap may be

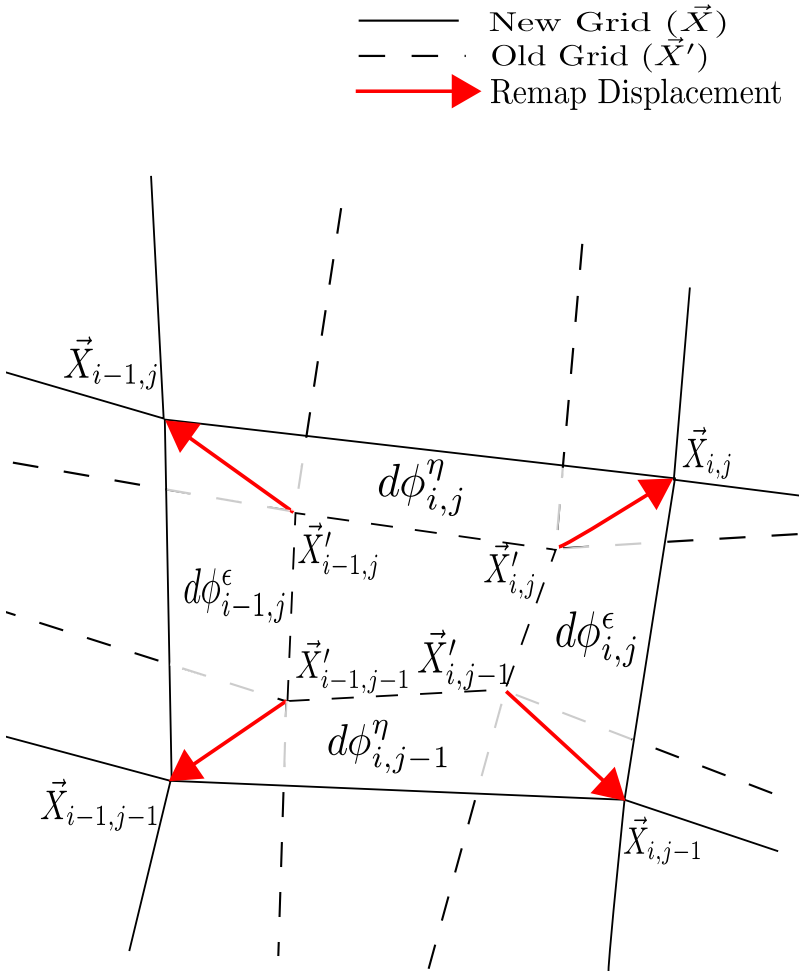


Figure 7.3: Change in flux in ignorable direction, as defined by a swept region based remap. The areas associated with the remap are calculated as the area of the quadrilateral formed by pairs of nodes of the new ($\vec{X}$) and old ($\vec{X}'$) grids, for example the area associated with $d\phi^{\epsilon}_{i,j}$ is formed by $\vec{X}_{i,j}, \vec{X}'_{i,j}, \vec{X}_{i,j-1}, \vec{X}'_{i,j-1}$. The $d\phi$'s (directed out of the page) are calculated by the product of this area and the interpolated value of the ignorable component of the magnetic field in the centre of that region.

implemented.

With this in mind it is possible to then carry out such a remap, once again calculating the flux transfer due to the motion of (as yet undefined) points of the median mesh, and applying them with either a positive or negative sign, once again based on (7.13). The question of which points to consider remains to be answered. If calculating the flux purely in terms of motion of the end points of the median mesh then only four points need be considered. However, with the nodal cell actually

being defined by eight points that would also appear to be an option.
Although there are multiple possible implementations of a cell centred constrained transport remap, they all share a common method of calculating the divergence of the B-field around the node. To calculate the divergence, discrete contributions must be summed around the median mesh. The flux through the median mesh is defined as the dot product of the median mesh vector and the cell centred magnetic field. The flux through the segments contributing to each of the nodal values of $\nabla \cdot \vec{B}$ is half of this. This is illustrated by figure 7.4.

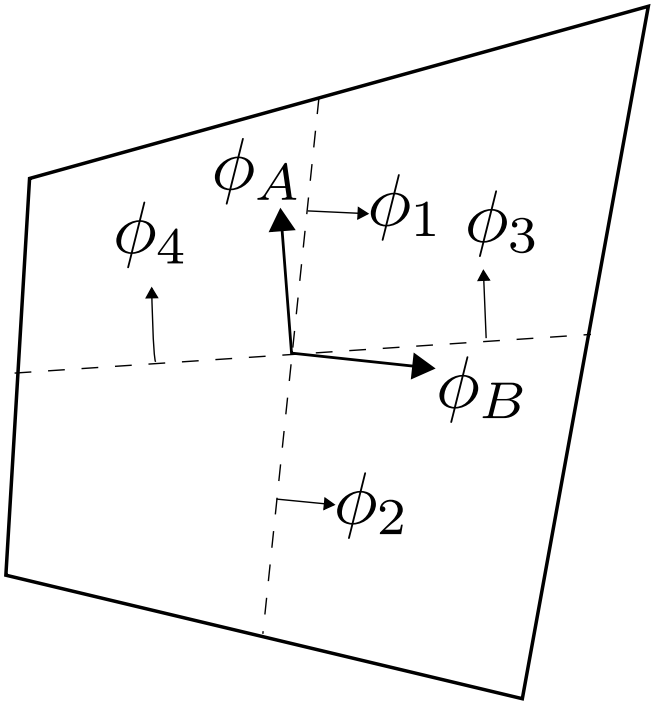


Figure 7.4: The initial fluxes used for a cell centred remap scheme, calculated at the end of the Lagrangian step. ϕ_A and ϕ_B are calculated from the dot product of the cell centred magnetic field and the median mesh vectors. The half-median mesh fluxes are set equal to half of the relevant full median mesh fluxes, $\phi_1 = \phi_2 = \phi_B/2$, $\phi_3 = \phi_4 = \phi_A/2$.

7.4.1 Eight Point Cell Centred Remap

Perhaps the most obvious cell centred constrained transport remap is the eight point cell centred remap. However this method is substantially more computationally expensive than the previously described constrained transport method in that it required twice as many flux transfers to be calculated.

The remap may be carried out (for the sake of transparency at least) on a node by node basis. For each node, eight flux transfers are calculated, one at each dynamic flux point defined by figure 7.6. Consider the point $(i + 1/2, j)$. The flux transfer is calculated as,

$$\epsilon_{(i+1/2,j)} = -\bar{v_x}\bar{B}_y + \bar{v_y}\bar{B}_x. \tag{7.17}$$

The velocity at this point is calculated as the average of the velocity of the two nodes of the edge. The magnetic field will need to be interpolated from the magnetic field in the neighbouring cells. So the change in ϕ_4 as defined by figure 7.7 can be written explicitly as:

$$\frac{d\phi_4}{dt} = \epsilon_{(i+1/2,j)} - \epsilon_{(i+1,j)}. \tag{7.18}$$

This process is carried out for each of the eight flux segments of the nodal mesh. Computational cost aside the problem with this method is at the end of the remap there is no guarantee that the two halves of a median mesh face within a cell will have equal fluxes, in fact, it is quite probable they won't. This process is illustrated in figure 7.5.

This presents the difficulty of constructing the cell averaged magnetic field for the following Lagrangian step. A simple (or weighted) averaging procedure would render the code with a cell centred quantity, however it is no longer guaranteed that this magnetic field is divergence free when calculated around the nodes at the end of the remap phase.

It may be possible to retain these subzonal fluxes and calculate subzonal magnetic fields. Subzonal pressure forces [31] however have been shown to modify the underlying physics of the problem being considered, changing physical measurements such as the growth rate of Rayleigh Taylor instabilities, as such subzonal magnetic fluxes (or indeed, fields) were not considered an option.

7.4.2 Four Point Cell Centred Remap

Mach2 [33] is an ALE code for MHD problems, which uses a cell centred remapping strategy. However, unlike the scheme described in the preceding section, only four

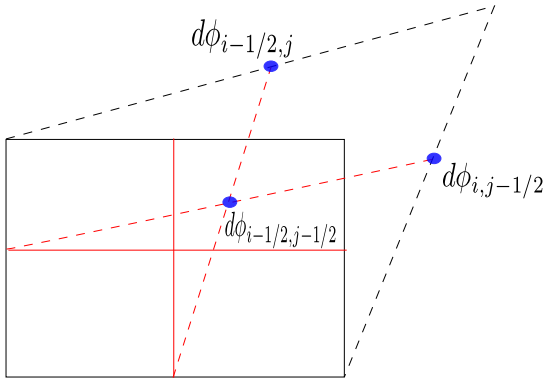


Figure 7.5: The development of subzonal fluxes occurring in an 8-point remap. The pre-remap grid is shown as a dash line, and the post-remap grid solid. The corresponding median meshes are shown in red. Using the notation of figure 7.4, $\phi_4 \rightarrow \phi_4 - d\phi_{i-1/2,j-1/2}$ $\phi_3 \rightarrow \phi_3 - d\phi_{i,j-1/2} + d\phi_{i-1/2,j-1/2}$, clearly post remap, $\phi_4 \neq \phi_3$. A similar argument may be applied to ϕ_1 and ϕ_2.

flux transfers are calculated for each node, at the intersection of the median mesh and the primary mesh, and as such this method does not suffer from the increased computational cost of the eight point method. Once again these change in fluxes are applied in a constrained transport sense, and thus divergence is conserved.
The four dynamic flux points of this scheme are defined by figure 7.8 but the nodal divergence retains the eight flux segments of figure 7.7. The flux transfer at $(i + 1/2, j)$ is as defined by (7.17) but there is no corresponding flux transfer at $(i+1, j)$, so,

$$\frac{d\phi_4}{dt} = \epsilon_{(i+1/2.j)}. \tag{7.19}$$

Figure 7.9 illustrates a single node, four point cell centred remap. It is clear this scheme also introduced subzonal magnetic fluxes. This could lead to the inference of subzonal B-fields; as the vectors associated with two differing flux segments ($\phi_3^{i-1,j}$ and $\phi_4^{i,j}$) are always equal, then the only possible conclusion is that the magnetic field is different on the two faces associated with these flux segments. This of course, would lead to subzonal magnetic fluxes acting on the nodes, which as discussed previously is undesirable. The authors do not describe how these are accounted for, either in terms of some smoothing procedure or subzonal magnetic fields. In fact, given the formulation presented, no subzonal magnetic fluxes are considered, and

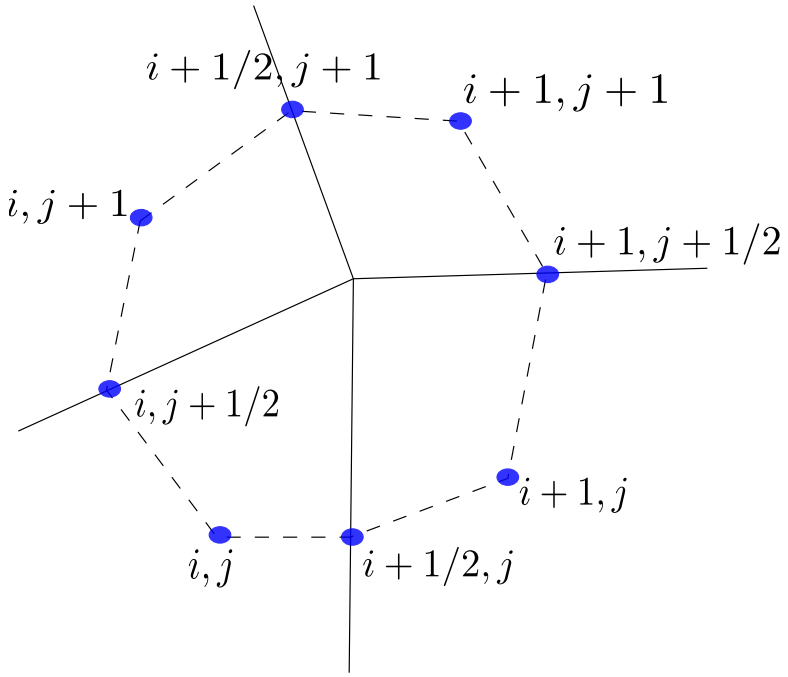


Figure 7.6: The dynamic flux points (indicated in blue) for an eight point cell centred remap. The velocities for the edge points and cell centres are calculated by averaging the relevant nodes. The magnetic field will have to be interpolated at all eight dynamic flux points to carry out a constrained transport remap.

thus it is unclear if the divergence of the magnetic field is conserved at all.

7.4.3 Remap Summary

The preceding discussion maybe be summarised by two main conclusions. A divergence cleaning scheme, whilst numerically viable, is a scheme which introduces a non-local effect at a considerable extra computational cost, added to correct for previous failures, and as such should really be used as a last resort. Secondly, a cell centred constrained transport remap, whilst conserving the divergence of the magnetic field, inevitably leads to subzonal magnetic fluxes, and the emergence of subzonal forces. Subzonal forces have been shown in the context of pure hydrodynamics to alter the properties of key physical results, which is clearly undesirable. With these facts in mind the development of the Lagrangian phase of the B-field evolution was carried out assuming that it would ultimately be necessary to either purely work in terms of flux through faces, or find some intermediate step, to al-

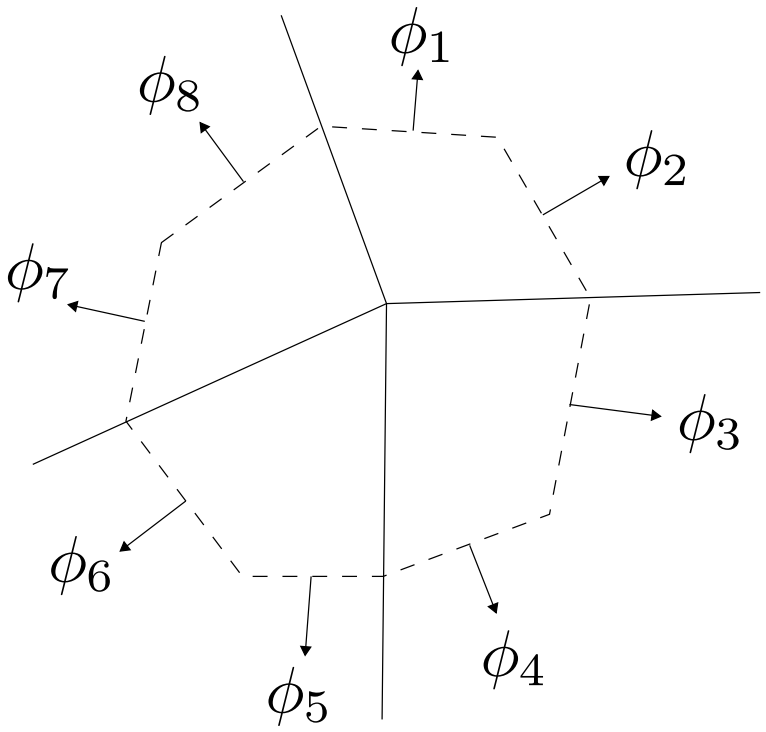


Figure 7.7: Eight-point cell centred remap. $\nabla \cdot \vec{B}$ is defined around the nodal cell, defined by the median mesh (dashed lines). There are eight flux contributions, ϕ_i, labelled by solid arrows. For the most general remap eight different $d\phi_i$s will need to be calculated.

low the coupling of the Lagrangian phase with a conventional constrained transport remap.

7.5 Lagrangian Phase

What should be clear from the introduction and the discussion of cell centred remapping schemes is that is is necessary to work in terms of fluxes through a cell face, calculate a cell centred magnetic field, advance this to a half time step value, and thus calculate the corrector level force needed to complete the Lagrangian phase.

7.5.1 Lagrangian Remap Codes

Lagrangian remap codes, like ALE codes, employ a staggered grid and store the magnetic field components at face centres. This allows for a simple update of the magnetic field at those locations by utilising the conservation of magnetic flux in

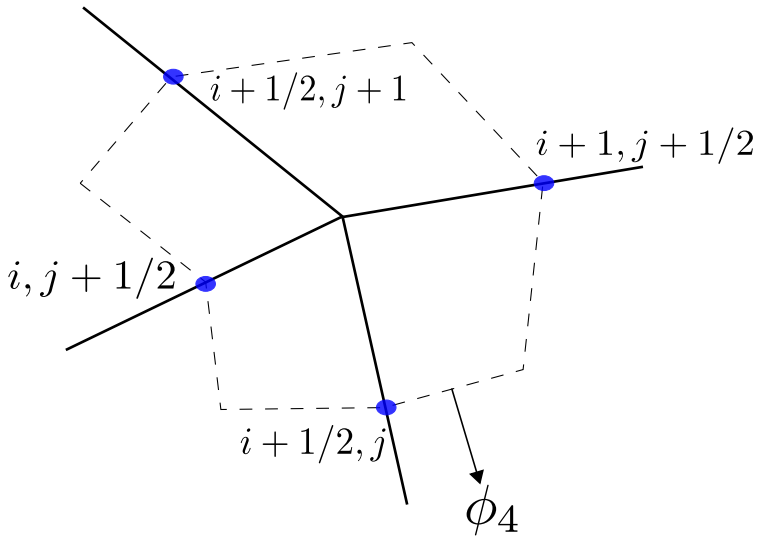


Figure 7.8: The dynamic flux points (indicated in blue) for a four point cell centred remap, once again velocities are averaged to obtain edge values, and the magnetic field components interpolated there. The other four vertices of the median mesh cell (the primary cell centres) do move during remap, however, no flux transfer is attributed to this movement.

ideal MHD. In order to carry out the calculation of the $\vec{J} \times \vec{B}$ force at the start of the predictor step the two non-ignorable direction components of the B-field are averaged to calculate a cell centred magnetic field. This can then be updated using the induction equation,

$$\frac{\partial \vec{B}}{\partial t} = \nabla \times \left(\vec{v} \times \vec{B} \right). \tag{7.20}$$

This allows for the cell averaged magnetic field to be advanced to the half time step level, which in turn is used to update the momentum. The magnetic field at the end of the time step is only needed for remapping, and is easily calculated at the necessary locations through conservation of magnetic flux.

Perhaps the most obvious method of extending the method of Lagrangian remap codes to ALE codes is to continue to store flux through the cell face, and modify somehow the method which calculates the cell centred magnetic fields. Before continuing this idea further it should be noted that this method has the advantage that it both keeps the inherent divergence conserving quality of Lagrangian remap codes' Lagrangian phases, and again, like Lagrangian remap codes, it is fully com-

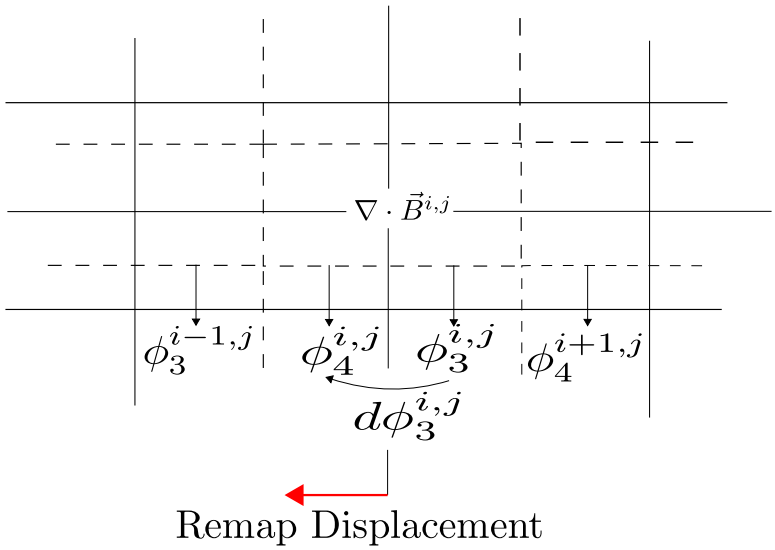


Figure 7.9: Four point cell centred remap, as used by Mach2. $\nabla \cdot \vec{B}$ is again defined around the nodal cell, defined by the median mesh (dashed lines). There are eight flux contributions, $\phi_i^{i,j}$, to the calculation of $\left(\nabla \cdot \vec{B}\right)^{i,j}$. Four such contributions (which are undergoing change in this example) are shown by solid arrows. Prior to the remap, $\phi_3^{i-1,j} = \phi_4^{i,j}$. In this remap, only flux changes calculated at the intersection of the primary and median mesh are calculated and transferred, as shown here, for the node $i, j-1$ moving in isolation (red arrow), giving rise to just one $d\phi_i$, and hence $\phi_3^{i-1,j} \neq \phi_4^{i,j}$ post remap.

patible with constrained transport. Furthermore, it is desirable that in modifying this method, that in the limiting case that the ALE code operates as a Lagrangian remap code, that the method reduces to that of the Lagrangian remap code.

ALE codes have the added complication that cell faces are no longer guaranteed to align with coordinate directions, but a magnetic field contribution can be recovered by realising that the flux is just the dot product of the B-field and the surface normal vector. Thus an orthogonal B-field at each cell face can always be recovered. This does not yet yield a cell centred B-field, nor will simply averaging these B-field contributions, simply because the method has half the information it requires; in order to average, both the parallel and perpendicular components at each cell face would be required.

As a further attempt to adapt the method used within Lagrangian remap codes it was proposed that averaging the perpendicular fields from each of the opposite edges in the cell, then summing the averages would yield a more accurate cell centred B-field. This was inspired by the idea that the two methods should reduce to

being identical in the perpendicular limit. Promising as this would appear, it has a very obvious shortcoming when used on the simple one dimensional MHD shock tube of Brio and Wu [43].
At the interface between the left and right states, at the first time step there is both an initial acceleration in the x-direction (due to the ∇P force), and an acceleration in the negative y, as demonstrated in figure 7.10. Applying the previously described

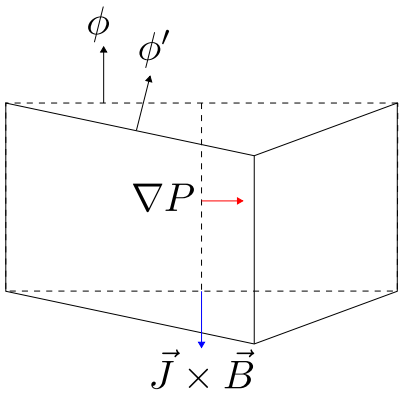


Figure 7.10: The initial forces at t=0 for the Brio and Wu MHD shock tube problem. The forces are shown by a red arrow (∇P) and a blue arrow ($\vec{J} \times \vec{B}$), on the old grid (dashed lines). The resulting (not to scale) grid displacement is shown by the solid lines. The flux is unchanged, $\phi = \phi'$.

scheme will produce clearly incorrect results. The two faces which were originally x-facing continue to do so, and provide correct (and unchanged) contributions to the cell centred B-field x-component.
However, the two originally y-facing components have had both their magnitude and direction perturbed, thus they will both now contribute to both the y-component and the x-component. Thus, the resulting cell centred B-fields has an x-component that varies with x, which cannot be so; it is a one dimensional problem, and as such such a variation will always violate the solenoidal condition. A more sophisticated approach is clearly needed.

7.5.2 Cauchy Solution

The Cauchy solution for the evolution of the B-field is based upon a method devised by Cauchy in the context of the vorticity equation for inviscid flow, but is explained in the context of the evolution of the B-field by Moffatt [44]. This was later extended by Craig and Sneyd [45] in studying the stability of coronal magnetic fields. This is a method of calculating the magnetic field based solely on the initial positions

of nodes, the initial magnetic field, and the deviation from the initial positions; there is no explicit time stepping. As will be shown, this is utilised to show that the divergence of the time advanced B-field is equal to that of the original B-field so, like constrained transport, this method is divergence conserving. This is in fact what is used in the current version of LareXd [10] to advance the magnetic field to the predictor level.

In order to explain the method, Craig and Sneyd's [45] original development is followed, and consequently the relevant sections of Moffatt [44]. It is necessary to introduce the following notation, $\vec{x}\left(\vec{X},t\right)$ which will denote the position of a Lagrangian point, at time t. $\vec{X}$ shall denote the initial position,

$$\vec{x}\left(\vec{X},0\right)=\vec{X}, \tag{7.21}$$

so that,

$$\frac{\partial x_i}{\partial X_j}=\delta_{ij}, \tag{7.22}$$

effectively $\vec{X}$ is our Lagrangian coordinate. Moffatt uses a different notation, $\vec{a}$ in place of $\vec{X}$, however here $\vec{a}$ is reserved for later use. Using this notation, a line element connecting two Lagrangian fluid parcels may be described as,

$$\delta x_i=\frac{\partial x_i}{\partial X_\alpha}\delta X_\alpha, \tag{7.23}$$

keeping Craig and Sneyds [45] original notation of summing over Greek indices and (7.22) is used to recover the desired result that $\delta x_i=\delta X_i$. Applying the Lagrangian derivative to this yields,

$$\frac{D}{Dt}\delta x_i=\frac{D}{Dt}\left(\frac{\partial x_i}{\partial X_\alpha}\right)\delta X_\alpha, \tag{7.24}$$

as $\delta\vec{X}$ is constant. This can be written in terms of the velocity vector:

$$\begin{aligned}\frac{D}{Dt}\delta\vec{x}&=\frac{\partial\vec{u}}{\partial X_\alpha}\delta X_\alpha\\&=\frac{\partial\vec{u}}{\partial x_\alpha}\frac{\partial x_\alpha}{\partial X_\beta}\delta X_\beta\\&=\left(\delta\vec{x}\cdot\nabla\right)\vec{u},\end{aligned} \tag{7.25}$$

where (7.23) has been used in the final simplification. This result is equally derivable from basic geometric considerations of a Lagrangian line element, where the rate of change of a component of the line element is given by the velocity difference of the

corresponding component along the line element.

As expected (from (7.23)) this is just the change in velocity component along the line element.

Now consider the evolution of the vector defined by the B-field, divided by the density.

$$\begin{aligned}\frac{D}{Dt}\left(\frac{\vec{B}}{\rho}\right) &= \frac{1}{\rho}\frac{D}{Dt}\left(\vec{B}\right) - \frac{\vec{B}}{\rho^2}\frac{D}{Dt}\left(\rho\right) \\ &= \frac{1}{\rho}\left(\vec{B}\cdot\nabla\vec{u} - \vec{B}\nabla\cdot\vec{u} + \frac{\vec{B}}{\rho}\rho\nabla\cdot\vec{u}\right) \\ &= \frac{\vec{B}}{\rho}\cdot\nabla\vec{u} \end{aligned} \tag{7.26}$$

Clearly (7.26) has the same form as (7.25), and so it is possible to write the vector $\vec{B}/\rho$ in terms of it's original value and the change in initial positions,

$$\frac{B_i^t}{\rho^t} = \frac{\partial x_i}{\partial X_\alpha}\frac{B_\alpha^0}{\rho^0}, \tag{7.27}$$

where the convention of summing over Greek indices has been retained, and superscripts refer to time levels. Time level 0 refers to initial values. Conservation of mass gives,

$$\frac{\rho_0}{\rho} = \frac{\partial\left(x_1, x_2, x_3\right)}{\partial\left(X_1, X_2, X_3\right)} = \Delta_0, \tag{7.28}$$

and the final equation of evolution of $\vec{B}/\rho$ is,

$$B_i = \frac{\partial x_i}{\partial X_\alpha} B_{0\alpha}\Delta_0^{-1}. \tag{7.29}$$

Whilst (7.29) is valid for any original grid, it is not a particularly convenient method for a grid that is not originally aligned with the coordinate system. Similarly to the case of (7.23), (7.29) is evolving the components of the B-field parallel to the original line elements used to define the elements of $\partial x_i/\partial X_j$. ALE codes are often used to model complex geometries, with computational domains that are not aligned with the global coordinate space, as such, in it's current form (7.29) will not (necessarily) yield the evolution of the B-field in terms of xy (or rz) components, unless line elements used to define the derivatives are aligned with the coordinate directions. However given that the derivation of (7.29) is done assuming a Lagrangian line element the (only fully) correct choice of points to define the derivatives are the

nodes. Of course, these directions do themselves provide a valid coordinate system but as Odin works in terms of the global coordinates, rather than the Lagrangian ones, it would be necessary to transform these components back into the global space. This would however require some knowledge of the original grid to be retained throughout, as well as extra computation, rendering the method as inconvenient for the current purposes.

As discussed ideally the line elements used to define the derivatives in (7.29) would be Lagrangian line elements, defined by two Lagrangian points. However given that what is actually being considered is the evolution of $\vec{B}/\rho$, the B-field components will be (co-)located at the centre of the cell. Thus it is required that $\partial x_i/\partial X_j$ will be cell centred also, so the median mesh is an obvious candidate for calculating these partial derivatives. It should be stated that these points are not true Lagrangian points themselves (see also discussion on definition of corner masses), there is a finite difference error present with such a method.

In a similar manner, having relaxed the criteria that the line elements be Lagrangian it is always possible to select some arbitrary points on the mesh such that the derivatives in (7.29) can be taken along line elements aligned with the coordinate system and result in B-field components in that system. In fact, (7.29) can be used to define B-field components in any direction (coordinate system) desired. Like the median mesh method, this is subject to a finite difference error, but is guaranteed to evolve the B-field components in the global coordinate system. However, like the suggestion of transforming the components back every time step it does required further storage and computation.

In order to find a more elegant solution to these problems Craig and Sneyd [45] introduced a third coordinate space, $\vec{a}$. They state that it is desirable that $\vec{a}$ is a uniform space. As discussed previously what is really desirable is a grid aligned with the coordinate system, and as will be shown it's both numerically and physically beneficial to make the $\vec{a}$ space an orthogonal coordinate space, with separation of unity between neighbouring points everywhere. This shall be referred to as a unit coordinate space. Before rewriting (7.29), it is useful to introduce some compact notation for the Jacobian determinants,

$$\frac{\partial\left(x_1, x_2, x_3\right)}{\partial\left(a_1, a_2, a_3\right)} = \Delta_{x,a} = \frac{1}{\Delta_{a,x}} \tag{7.30}$$

and correspondingly,

$$\frac{\partial\left(X_1, X_2, X_3\right)}{\partial\left(a_1, a_2, a_3\right)} = \Delta_{X,a} = \frac{1}{\Delta_{a,X}}. \tag{7.31}$$

Finally,

$$\Delta_{x,X} = \Delta_{x,a} \cdot \Delta_{a,X}, \tag{7.32}$$

so (7.29) can be first rewritten as,

$$B_i = \frac{\partial x_i}{\partial X_\alpha} B_{0\alpha} \Delta_{X,a} \Delta_{x,a}^{-1}. \tag{7.33}$$

Now application of the chain rule yields,

$$B_i = \frac{\partial x_i}{\partial a_\alpha} \frac{\partial a_\alpha}{\partial X_\beta} B_{0\beta} \Delta_{X,a} \Delta_{x,a}^{-1}. \tag{7.34}$$

Now, making the choice to make $\vec{a}$ a unit coordinate space no extra information (i.e. a metric) would need be stored to calculate Δ and the partial derivatives with respect to a_i, hence the previously mentioned numerical benefit. As an example,

$$\frac{\partial x_1}{\partial a_1} = x_i - x_{i-1}, \tag{7.35}$$

due to the uniform separation of points in $\vec{a}$ space. Collecting the terms in (7.34) which do not vary in time, (7.34) can be rewritten,

$$B_i = \frac{\partial x_i}{\partial a_\alpha} \bar{B}_\alpha \Delta^{-1}, \tag{7.36}$$

where,

$$\bar{B}_i = B_{0\alpha} \frac{\partial a_i}{\partial X_\alpha} \frac{\partial \left(X_1, X_2, X_3\right)}{\partial \left(a_1, a_2, a_3\right)}. \tag{7.37}$$

The field defined by (7.37) is what the original authors, [45], term the pre-initial magnetic field. Thus the evolution of the B-field has been written in terms of two separate coordinates, firstly the Lagrangian coordinates, X, and secondly the unit space, a. As with the formulation of the Cauchy solution of (7.36) evolves the B-field components parallel to the directions defined by the vectors $\vec{a}$. These have though been defined to be (unit) vectors aligned with the coordinate space so the full Cauchy solution as developed by Craig and Sneyd [45] evolves the B-field components in the global coordinate space.

Having derived the full expression for B_i at any time, the divergence of the evolved B-field should be considered. The chain rule gives,

$$\frac{\partial B_i}{\partial x_j} = \frac{\partial a_\alpha}{\partial x_j} \frac{\partial B_i}{\partial a_\alpha} = a_{\alpha,j} B_{i,\alpha}, \tag{7.38}$$

and substituting (7.36) gives,

$$\frac{\partial B_i}{\partial x_j} = a_{\alpha,j}\left(x_{i,\alpha\beta}\bar{B}_\alpha \Delta^{-1} + x_{i,\alpha}\bar{B}_{\beta,\alpha}\Delta^{-1} - x_{i,\beta}\bar{B}_\beta \Delta^{-2}\Delta_{,\alpha}\right). \tag{7.39}$$

For a nonsingular matrix $(m_{i,j})$,

$$\frac{\partial}{\partial m_{ij}} det(M) = (M^{-1})_{ji} det(M), \tag{7.40}$$

so that,

$$\begin{aligned}\Delta_{,i} &= x_{\alpha,\beta i}\frac{\partial}{\partial x_{\alpha,\beta}}\Delta \\ &= x_{\alpha,\beta i}a_{\beta,\alpha}\Delta,\end{aligned} \tag{7.41}$$

where the fact that $a_{i,j}$ is found from the inverse of $x_{i,j}$ has been used. Substituting in and multiplying through by Δ gives,

$$\Delta\frac{\partial B_i}{\partial x_j} = a_{\alpha,j}\left(x_{i,\alpha\beta}\bar{B}_\beta + x_{i,\beta}\bar{B}_{\beta,\alpha} - x_{i,\beta}\bar{B}_\beta x_{\gamma,\delta\alpha}a_{\delta,\gamma}\right), \tag{7.42}$$

and finally contracting on i and j to give the divergence yields,

$$\Delta\left(\nabla\cdot\vec{B}\right) = a_{\alpha,\gamma}x_{\gamma,\alpha\beta}\bar{B}_\beta + \nabla\cdot\bar{\vec{B}} - a_{\delta,\gamma}x_{\gamma,\delta\alpha}\bar{B}_\alpha = 0. \tag{7.43}$$

The first and third terms cancel exactly, leaving the familiar result that if that B-field was initially divergence free it will remain so.

This method describes the evolution of a B-field with components co-located on the grid. As previously suggested these will be co-located at the cell centre, which is also required as the location for the predictor B-field. It is worthwhile considering exactly what the pre-initial B-field means in this case, considering the x-component:

$$\bar{B}_x = \left(B_{0x}\frac{\partial a_x}{\partial X_x} + B_{0y}\frac{\partial a_x}{\partial X_y}\right)\frac{\partial\left(X_1, X_2, X_3\right)}{\partial\left(a_1, a_2, a_3\right)}, \tag{7.44}$$

where the fact that $\partial a_x/\partial X_z = 0$ always for a two dimensional code has been used. The coefficients of the matrix $\partial a_i/\partial X_j$ are calculated by the inverse of $\partial X_i/\partial a_j$, so it is possible to rewrite the x-component of the magnetic field as,

$$\bar{B}_x = B_{0x}\frac{\partial X_y}{\partial a_y} - B_{0y}\frac{\partial X_x}{\partial a_y}. \tag{7.45}$$

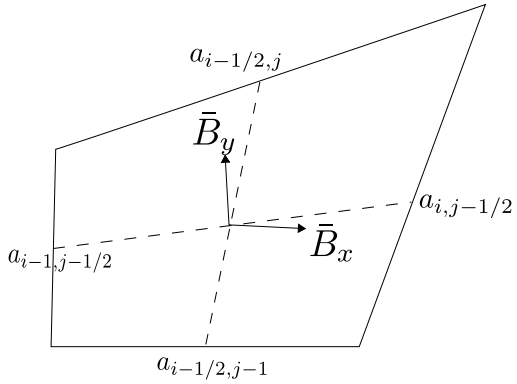


Figure 7.11: The pre-initial magnetic field can be visualised as the flux through the median mesh. $\partial X_i/\partial a_j$ for each cell is calculated numerically using the edge midpoint positions as indicated.

By making $\vec{a}$ a unit space, $\bar{B}$ is equal in magnitude to the flux through the median mesh; the same fluxes that were previously suggested for the cell centred remap. The apparently incorrect minus sign in the second term in (7.45) is accounted for by the fact that in the case that a positive B_y should make a positive contribution to the flux, $\partial X_x/\partial a_y$ is negative, as illustrated by fig 7.12. All the points needed to calculate both components of the pre-initial field are illustrated by figure 7.11.

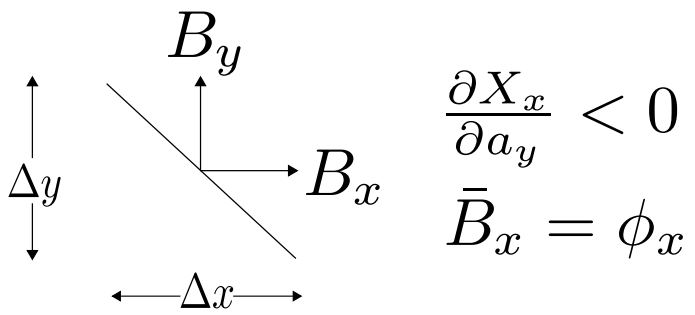


Figure 7.12: The pre-initial magnetic field is shown to be the correct calculation of the flux through the median mesh.

Unfortunately fluxes through the median mesh have been shown to be incompatible

with ALE schemes. Although $\bar{B}$ is equal to the flux, it is still a field, with units that reflect that, and with multiple components. It is a B-field defined on a unit, orthogonal grid space. As such, borrowing a method from fixed grid codes, the cell centred pre-initial B-field could be replaced with the average of the value of those components on the faces of the cell. More simply, rather than use the median mesh flux as the pre-initial B-field the average of the fluxes through the primary mesh faces is used instead. Explicitly the pre-initial magnetic field has been replaced by,

$$\bar{B}_x = \frac{1}{2}\left(\phi_{i-1/2,j} + \phi_{i+1/2,j}\right), \bar{B}_y = \frac{1}{2}\left(\phi_{i,j-1/2} + \phi_{i,j+1/2}\right). \tag{7.46}$$

This avoids the problems described in the previous section as $\vec{a}$ space is always orthogonal.

As the ϕ values in (7.46) are equal to the fluxes through the primary mesh (defined by nodes) they are compatible with a constrained transport remap, and as such the Cauchy solution, with the modification described can safely be implemented in an ALE scheme, whilst maintaining a divergence free magnetic field. Of course, this change in definition of $\bar{B}$ introduces a finite difference error, but does remove the previous finite difference error of the cell centred scheme by describing the evolution of the B-field in terms of movement of Lagrangian points.

7.5.3 Equivalence to Induction Equation

It is in fact possible to show that, given (7.46) as an (pre) initial magnetic field, that advancing it through either the induction equation, (7.20), or by the Cauchy solution, (7.34) are equivalent. First (7.20) should be written in integral form, using the definition of the Lagrangian derivative, and Reynolds Transport Theorem,

$$\frac{D}{Dt}\int_\Omega B d\Omega = \int_\Omega \left(\vec{B}\cdot\nabla\right)\vec{v} d\Omega \tag{7.47}$$

It is convenient to consider the product of the magnetic field and the control volume, BCV, rather than simply the magnetic field, without obstructing the validity of the proof, as volume is calculated consistently in the two methods. Now, writing the x-component of (7.47) in its discrete form,

$$B_x^{1/2} CV^{1/2} = B_x^0 CV^0 + \frac{\Delta t}{2}\left[\phi^\eta_{i,j}\bar{v}^b_x - \phi^\eta_{i-1,j}\bar{v}^d_x + \phi^\epsilon_{i,j}\bar{v}^c_x - \phi^\epsilon_{i,j-1}\bar{v}^a_x\right], \tag{7.48}$$

where the superscripts $0, (1/2)$ refer to the initial and predictor time levels respectively. $\bar{v}_x$ is the edge averaged x-component of the velocity, and the superscript

denotes the corresponding edge, as denoted by figure 7.13.

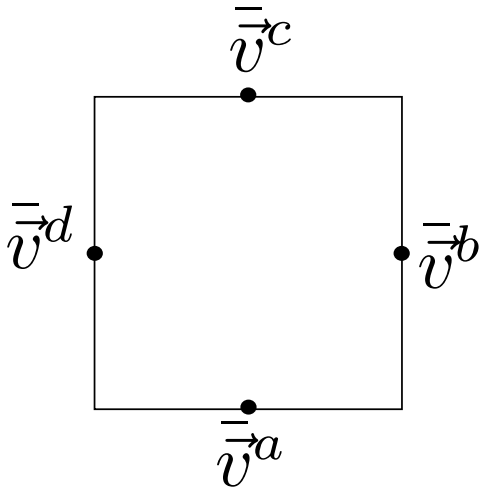


Figure 7.13: Notation for edge averaged velocities used for possible implementation of evolution through the induction equation.

Rearranging,

$$\Delta(BCV) = \frac{\Delta t}{2}\frac{1}{2}\Big[\left(\phi^{\eta}_{i,j} + \phi^{\eta}_{i-1,j}\right)\left(\bar{v}^{b}_{x} - \bar{v}^{d}_{x}\right) + \left(\phi^{\eta}_{i,j} - \phi^{\eta}_{i-1,j}\right)\left(\bar{v}^{b}_{x} + \bar{v}^{d}_{x}\right) + \left(\phi^{\epsilon}_{i,j} + \phi^{\epsilon}_{i,j-1}\right)\left(\bar{v}^{c}_{x} - \bar{v}^{a}_{x}\right) + \left(\phi^{\epsilon}_{i,j} - \phi^{\epsilon}_{i,j-1}\right)\left(\bar{v}^{c}_{x} + \bar{v}^{a}_{x}\right)\Big], \tag{7.49}$$

and finally grouping terms,

$$\Delta(BCV) = \frac{\Delta t}{2}\Big[\bar{v}_{x}\left(\phi^{\eta}_{i,j} - \phi^{\eta}_{i-1,j} + \phi^{\epsilon}_{i,j} - \phi^{\epsilon}_{i,j-1}\right) + \frac{1}{2}\left(\phi^{\eta}_{i,j} + \phi^{\eta}_{i-1,j}\right)\left(\bar{v}^{b}_{x} - \bar{v}^{d}_{x}\right) + \frac{1}{2}\left(\phi^{\epsilon}_{i,j} + \phi^{\epsilon}_{i,j-1}\right)\left(\bar{v}^{c}_{x} - \bar{v}^{a}_{x}\right)\Big]. \tag{7.50}$$

Here $\bar{v}_x$ is the cell average of the x component of the velocity.
Assuming that the magnetic field has been specified to be divergence free, the first term on the right hand side of (7.50) disappears. Noting that the velocity differences in the second terms on the right hand side constitute numerical derivatives in uniform grid space, and writing the averages of opposing fluxes as barred quantities, the final difference term for the explicit induction equation is,

$$\Delta(BCV) = \frac{\Delta t}{2}\left(\bar{\phi}^{\eta}\frac{\partial v_x}{\partial a_x} + \bar{\phi}^{\epsilon}\frac{\partial v_x}{\partial a_y}\right). \tag{7.51}$$

To calculate the equivalent term for the Cauchy solution consider first the magnetic field at $t = 0$,

$$B_x^0 CV^0 = \frac{1}{2}\left[\left(\phi^{\eta}_{i-1,j} + \phi^{\eta}_{i,j}\right)\frac{\partial x}{\partial a_x} + \left(\phi^{\epsilon}_{i,j-1} + \phi^{\epsilon}_{i,j}\right)\frac{\partial x}{\partial a_y}\right]. \tag{7.52}$$

Then by noting that,

$$\left.\frac{\partial x}{\partial a_x}\right|_{t=1/2} = \left.\frac{\partial x}{\partial a_x}\right|_{t=0} + \frac{\Delta t}{2}\left.\frac{\partial v_x}{\partial a_x}\right|_{t=0}, \tag{7.53}$$

it is clear that,

$$\Delta\left(BCV\right) = \frac{\Delta t}{2}\frac{1}{2}\left[\left(\phi^{\eta}_{i-1,j} + \phi^{\eta}_{i,j}\right)\frac{\partial v_x}{\partial a_x} + \left(\phi^{\epsilon}_{i,j-1} + \phi^{\epsilon}_{i,j}\right)\frac{\partial v_x}{\partial a_y}\right]. \tag{7.54}$$

Clearly, (7.54) and (7.51) are equivalent.

There is a caveat to the above proof however. The proposed method of implementing the induction equation has been deliberately simplified. As given above the method would (most likely) require the cell centred magnetic field, as well as the fluxes through the face to be stored. Other choices may exist, but this discussion is somewhat off topic. With this in mind, it must be stressed that what has actually been demonstrated is only equivalent evolution. The adaption of the Cauchy solution to ALE codes as outlined previously effectively carries out an averaging procedure at $t = 0$. Given that the induction equation may not carry out this step the initial magnetic field, as calculated by the code from the initial conditions may be different, and this of course will yield different results. The adapted Cauchy solution is still used despite this averaging, as it is not clear how to combine evolution through the induction equation with an arbitrary constrained transport remap, and it is considerably computationally cheaper.

A second inference from the above proof is that it makes it clear that the Cauchy solution does indeed involve time stepping, but only in the calculation of the partial derivatives of the position with respect to the Lagrangian coordinates. The fact that an ALE scheme would already have generated this information and carried out the necessary time stepping reinforces the computational efficiency of the scheme.

7.6 Coupling of Remap to Cauchy Solution

All that remains is to couple the modified Cauchy solution to the adaption of the constrained transport method for remapping. The remap for the ignorable direction

B-field component remains unchanged, and the following discussion is only relevant to the two in plane components. Should a remap be triggered the following steps are carried out.

The first step in a remap is to define a remap displacement vector as defined according to (7.16). As previously mentioned, to calculate the flux transfers during a remap it is necessary to interpolate the B-field to the midpoint of this vector. In order to do this the Cauchy solution is used to calculate the cell centred B-field at the end of the time step.

It is desirable to now upwind the B-field in order to carry out the remap. In a first order scheme this reduces to a donor cell scheme. However for an arbitrary grid at the end of the time step it is non-trivial to calculate which cell the midpoint of the displacement vector lies in. This is complicated further by the fact that it is possible this point lies along one of the edges itself.

Point in polygon schemes (such as the ray tracing algorithm) encounter difficulties when the point in question lies along the polygon (cell) edge. Solutions do exist where a specific cell could be chosen, or it may be suggested that an average of the two cells in question be taken.

For a displacement vector lying along an edge it would however be more correct to (for a first order scheme) calculate the flux transfer as a fraction of the flux through the full edge. Clearly this scheme is more complicated than the previously described hydrodynamic remap. For a higher order scheme complications grow. An isoparametric remap (chapter 9) greatly simplifies these problems, but is beyond the scope of this chapter, thus to demonstrate the basic scheme the average of the four surrounding cell centred B-fields was used.

It may be argued that a distance weighted average might be more appropriate, but as the aim is to simply demonstrate the scheme a simple average was used for the tests presented in this chapter. Having obtained a value for the B-field at the node, this is combined with the remap velocity as calculated from (7.16) and used to calculate and apply flux transfers in a constrained transport remap. This gives post remap fluxes through the cell faces which are then used for the pre-initial magnetic field as in (7.46) for proceeding Lagrangian steps.

7.7 Shock Capturing in ALE MHD

Shock viscosities (e.g. [14], [18], [24]) are needed to enable calculations involving shocks in ALE hydrodynamics codes, and are usually triggered by some form of compression switch. However in MHD problems shocks may also occur in cells that

are not undergoing compression. Therefore to enable shock capturing the mimetic tensor viscosity of Campbell and Shashkov, [24], was used, but with the compression requirement relaxed. The viscosity limiters were left on, as they ensure the viscosity will turn off smoothly in regions of smooth flow. The viscous heating, despite the relaxation of the compression switch remains positive always.
For pure hydrodynamics the magnitude of this viscosity is derived from the jump conditions across a shock, [16], [17]. However as pointed out in [10] no such derivation exists for MHD. For purely hydrodynamical cases this magnitude is dependent on the sound speed, and in adapting the tensor shock viscosity of [24], the sound speed is replaced by the fast speed. In practice running with the full Kuropatenko magnitude for the viscosity (with sound speed replaced with fast speed) resulted in excessive dissipation, and as such problems were run with reduced viscosity of $c_1 = 0.1$, $c_2 = 0.5$ unless stated otherwise.

7.8 Lorentz Force Term Calculation

Having demonstrated how to couple a constrained transport remap, and an adapted version of the Cauchy solution to an ALE code, and how to modify a shock viscosity to work in MHD problems, the final required step is how to calculate the Lorentz force terms. For ideal MHD in terms of normalised variables the acceleration is given by,

$$\begin{aligned} \rho\frac{D\vec{v}}{Dt} &= \vec{J}\times\vec{B} - \nabla p \\ &= \left(\nabla\times\vec{B}\right)\times\vec{B} - \nabla p. \end{aligned} \tag{7.55}$$

However in order to utilise the divergence theorem, (7.55) is rewritten in its conservative form,

$$\rho\frac{D\vec{v}}{Dt} + \nabla\cdot\left(\boldsymbol{I}p + \frac{\boldsymbol{I}\vec{B}^2}{2} - \vec{B}\otimes\vec{B}\right) = 0. \tag{7.56}$$

Treatment of the second term in (7.56) is straightforward; the magnetic pressure force is calculated in an analogous way to the thermodynamic pressure force, given that the half time step B-field is cell centred. The final term is a little more complex. Considering just the magnetic tension force,

$$\vec{F} = -\int_\Omega \nabla\cdot\left(\vec{B}\otimes\vec{B}\right)dV, \tag{7.57}$$

where the integration is carried out over the nodal cell. Applying the divergence theorem for tensors,

$$\begin{aligned}\vec{F} &= -\int_{\partial\Omega} \left(B^i n_i\right) \vec{B} dV \\ &= -\oint_{\partial\Omega} \vec{B}\left(\vec{B}\cdot\vec{n}\right) dS \\ &= -\sum_{i=1}^{8} \vec{B}\phi_i \end{aligned} \tag{7.58}$$

Thus the corner force resulting from the magnetic tension force is given by,

$$\vec{f}_i^z = -\vec{B}^z\phi, \tag{7.59}$$

where ϕ is the (half time-step) flux through the median mesh segment associated with the corner force.

7.9 Summary of a single MHD ALE time step

To summarise the development in this chapter the individual steps necessary in adding ideal MHD to an ALE code are presented.

0. The fluxes through the primary mesh faces are stored. This is either done through initial conditions if carrying out the first time step, or retained from a previous Lagrangian step if no remap has taken place, or modified by a remap step.

1. The half time step, cell centred, B-field is calculated in terms of the flux through the primary mesh faces and the half time step nodal positions using the modified Cauchy method.

2. The half time step flux through the median mesh face is calculated. The total force due to the half time step B-field is calculated.

3. These forces are used to update the velocity fields, in conjunction with the hydrodynamical forces.
   Optionally:

4. A remap is carried out which modifies the flux through the primary mesh.

7.10 Boundary Conditions

All that remains is to define how the boundary conditions should be applied to the B-field components. The third component of the B-field is treated in the same manner as the scalar quantities within subsection 2.3.1. For periodic boundary conditions, the in-plane B-field components are calculated in the same manner as the velocity components. For reflective boundary conditions the in-plane components are calculated in the same manner as the tangential velocity components in subsection 2.3.1. This yields the useful result that the B-field is consistent with the both the reflective nature of the boundary conditions, and the solenoidal constraint.

7.11 Results

7.11.1 Brio and Wu MHD Shock Tube

Like its counterpart in hydrodynamics the Brio and Wu shock tube [43] represents a basic test of a MHD codes ability to capture shocks. The initial conditions are very similar to the Sod shock tube, but with an imposed constant B-field x-component, and a y-component which changes sign at the interface between the states.

$$\begin{aligned}(\rho, P, B_y) &= \begin{cases}(1, 1, 1) & \text{if } x < 0.5\\ (0.125, -1, 0.1) & \text{if } x > 0.5\end{cases}\\ B_x &= 0.75\\ \vec{v} &= 0.0\end{aligned} \tag{7.60}$$

The initial tests with the Brio and Wu shock tube show good agreement with other codes (e.g. [10]), but the simplicity of the problem allows an easy investigation into the question of the compression switch in MHD. The results shown in figure 7.14 represent a calculation where the compression switch is relaxed, i.e. the viscosity is applied everywhere for MHD problems. This follows the method presented in [10]. However it is also possible to run the test with the compression switch active, as shown in figure 7.15. The results in figure 7.14 show less oscillations and undershoot than in figure 7.15 justifying the decision to relax the compression switch.

7.11.2 Magnetised Noh

Like the previous test case the magnetised Noh problem represents an adaption of a hydrodynamical shock test to MHD with the addition of a B-field. The Noh problem in Cartesian coordinates (i.e. a collapsing sphere) is modified with the addition of a

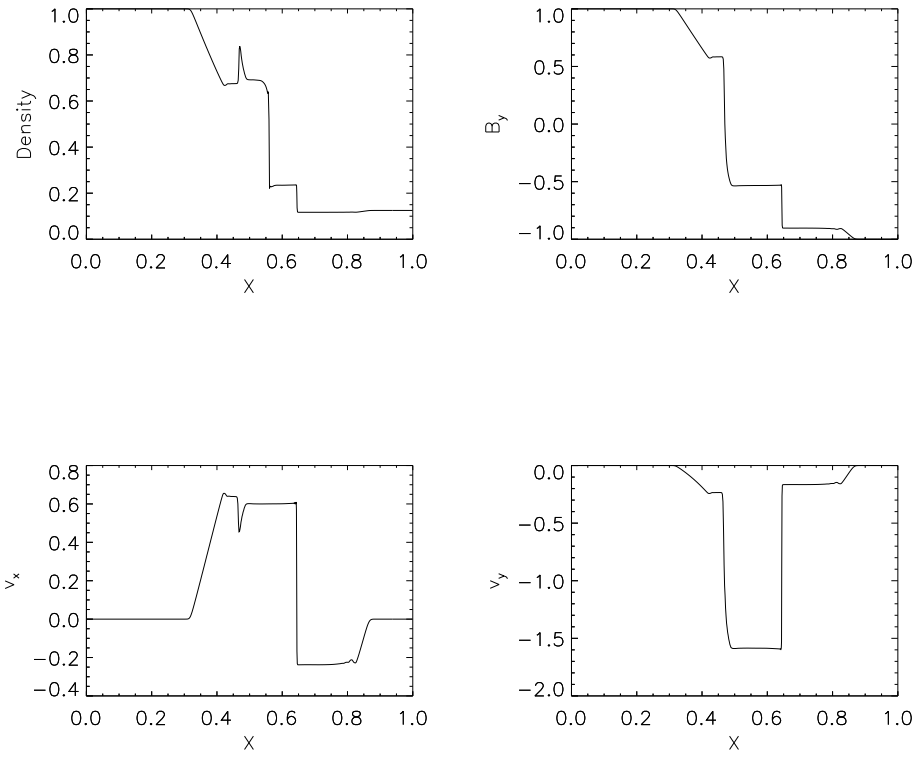


Figure 7.14: Brio and Wu magnetised shock tube problem, fully Lagrangian Results. 800 cells.

radially varying B_θ. The problem was first devised to provide z-pinch codes with a verification test [46], and has been used also as a test of expanding existing codes to cylindrical MHD [47]. However, this problem can easily be transferred to a Cartesian scheme, and represents a good test of the codes ability to model B-fields and flows not aligned with the grid.

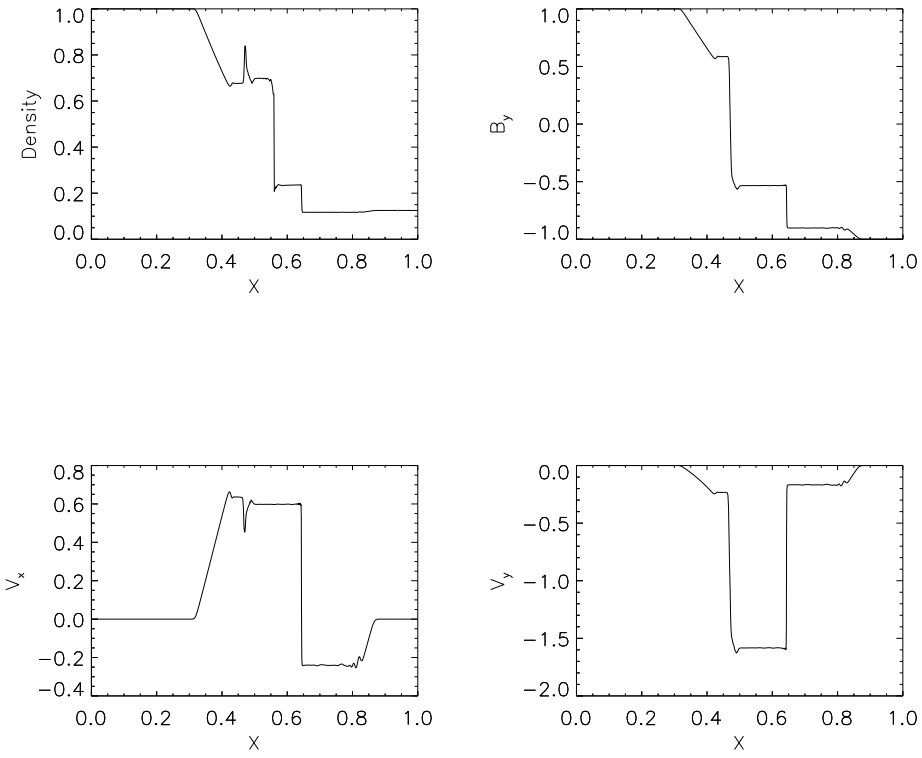


Figure 7.15: Brio and Wu magnetised shock tube problem, fully Lagrangian Results, with compression switch active. 800 cells.

The initial conditions (given in cylindrical coordinates) are,

$$\begin{aligned} \rho &= 3.1831 \times 10^{-5} r^2 g/cm^3 \\ v_r &= -3.24101 \times 10^7 cm/s \\ v_z &= 0 \\ v_\theta &= 0 \\ B_r &= 0 \\ B_z &= 0 \\ B_\theta &= 6.35584 \times 10^5 r Gauss \\ P &= 0. \end{aligned} \tag{7.61}$$

Here r is the radial coordinate in cm, and the problem is run to 30 ns.

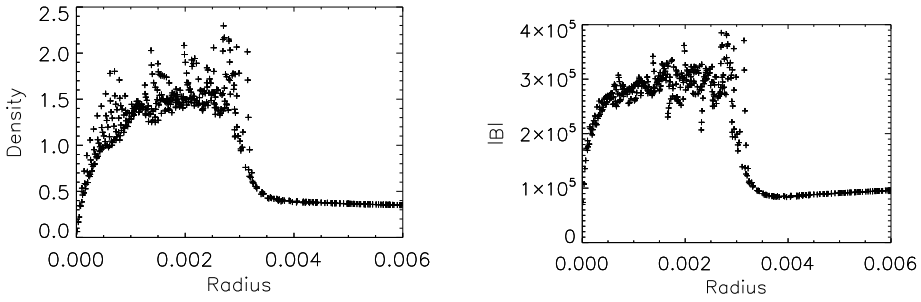


Figure 7.16: Magnetised Noh problem, run with viscosity coefficients $c_1 = 0.1$, $c_2, = 0.5$ Fully Lagrangian, 50x50.

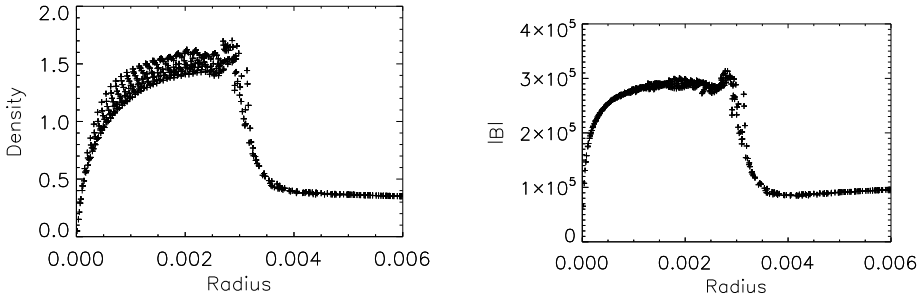


Figure 7.17: Magnetised Noh problem, run with viscosity coefficients $c_1 = 1.0$, $c_2, = 1.0$. Fully Lagrangian, 50x50.

This problem was run twice, in fully Lagrangian mode. As discussed previously, when running with both viscosity coefficients set to unity excessive dissipation appears in some MHD problems. However for this problem, which has a one dimensional solution, significantly better results were obtained with coefficients at unity, as shown by comparison of figure 7.16 and figure 7.17.

7.11.3 MHD Rotor

The MHD rotor test [48] represents a more demanding test of Lagrangian MHD solver. Unlike in previous tests where the flow was strictly one dimensional the flows in this case are more complex, and grid twisting is a natural part of the solution. The problem consists of a rotating dense disc at the origin surrounded by a less dense stationary background fluid. There exists a matching region between the two materials, that is not strictly needed for ALE codes, however it was included

to aid comparison with previously published results.
The initial conditions are given by,

$$(\rho, v_x, v_y) = \begin{cases} (10, -v_0 (y - 0.5)/r_0, -v_0 (x - 0.5)/r_0) & \text{if } r < r_0 \\ (1 + 9f, -f v_0 (y - 0.5)/r, -f v_0 (x - 0.5)/r) & \text{if } r_0 \leq r < r_1 \\ (1, 0, 0) & \text{if } r \geq r_1 \end{cases} \tag{7.62}$$

Here r is the radial distance, $r_0 = 0.1$ and $r_1 = 0.115$. f is the matching function given by,

$$f = \frac{r_1 - r}{r_1 - r_0}. \tag{7.63}$$

The pressure is given by $p = 1.0$ everywhere and finally, the B-field is given by,

$$\vec{B} = (B_x, B_y, B_z) = \left(5/\sqrt{4\pi}, 0, 0\right). \tag{7.64}$$

The first set of results shown in figure 7.18 show good resolution of features seen in previous calculations of this problem. The two dimensional nature of this problem also allows a good test of the basic remapping strategies presented in this chapter. The remapping here is first order, however given the problem is shock dominated this is not expected to damage the solution by a great amount. For the second run of this test the problem was run until t=0.39, at which point the grid was locked and a complete remap was carried out every time step. Due to the presence of a remap this problem was able to be run at a higher resolution, previously such high resolutions in conjunction with fully Lagrangian grid motion resulted in unacceptably small time steps. The results in figure 7.19 show good agreement with figure 7.18. Some additional (minor) features are present, most likely resulting from the increased resolution.

7.11.4 Orszag Tang Vortex

The final test case presented here is the Orszag Tang vortex [49],[50]. Like the previous rotor test the flows are complex and two dimensional, but it is also a shock dominated problem, so allows a test of both the Lagrangian and remapping techniques previously discussed.

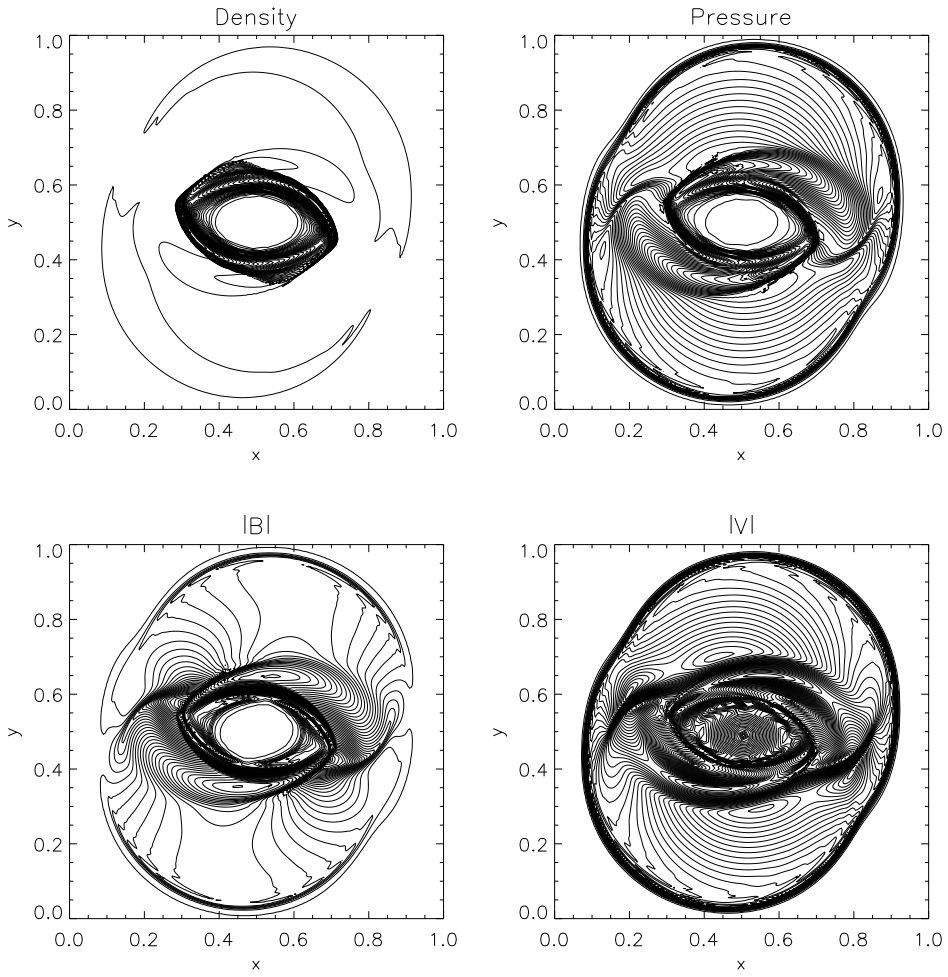


Figure 7.18: MHD Rotor problem, fully Lagrangian Results. 200x200.

The initial conditions are given by,

$$\begin{aligned} \rho &= 25/9 \\ v_x &= -\sin y \\ v_y &= \sin x \\ v_z &= 0 \\ B_x &= -\sin y \\ B_y &= \sin 2x \\ B_z &= 0 \\ P &= 5/3. \end{aligned} \tag{7.65}$$

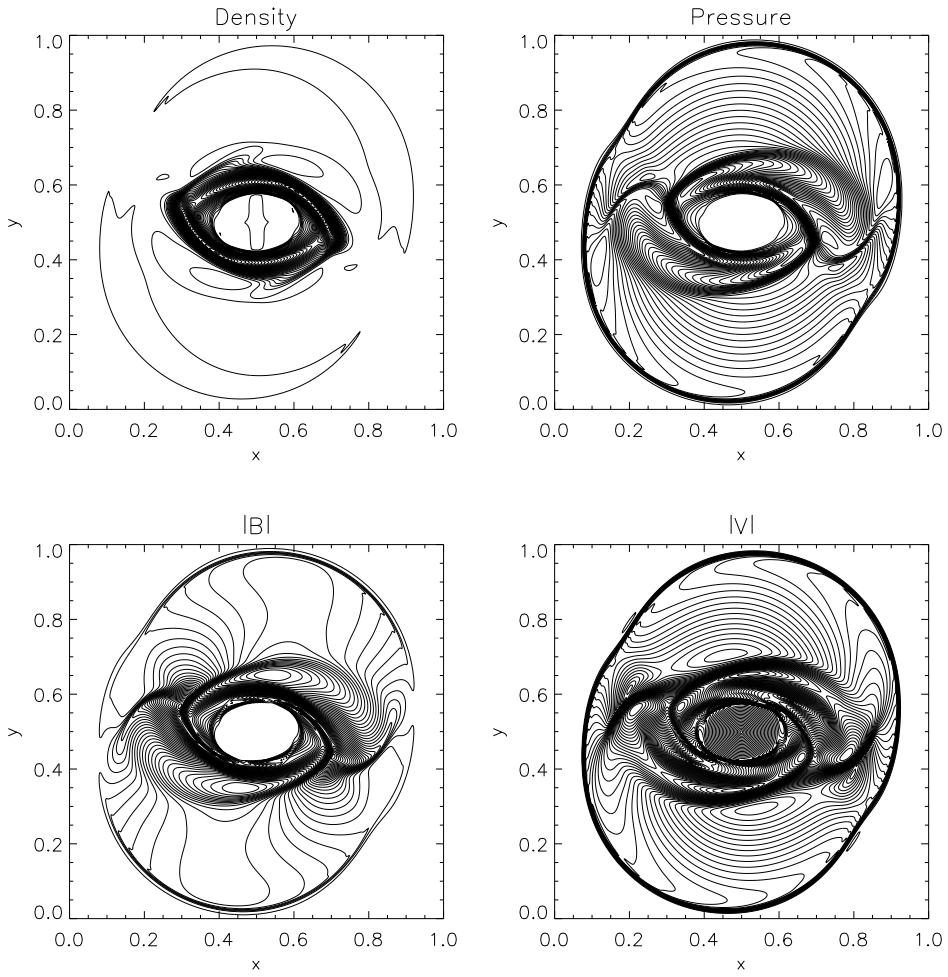


Figure 7.19: MHD Rotor problem, fully Lagrangian until t = 0.39, then fully Eulerian. 400x400.

The first test run is carried out with fully Eulerian grid motion for all time, as an initial base example. When comparing the results in figure 7.20 with other published results some short comings are apparent, particularly in the resolution of the central bar which is not as distinct as in previous results. Also the central quadrilateral feature is not particularly well defined, most likely due to the lack of upwinding in the remap scheme. The second run of the Orszag Tang vortex figure 7.21 was run in a fully Lagrangian mode until t=1.0, and then the grid was locked, and a full remap was carried out at the end of all time steps. This allows a good test of how the grid imprints on the solution. The results in figure 7.20 and figure 7.21 show good qualitative agreement, which supports the idea that the Cauchy solution for the

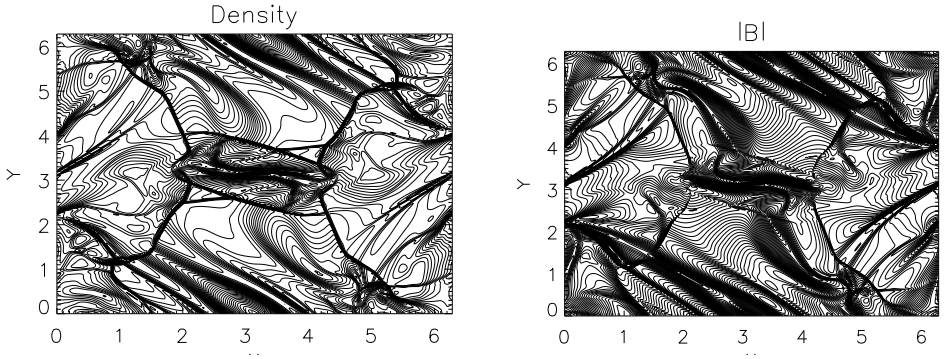


Figure 7.20: Orszag Tang Vortex, run in fully Eulerian mode. 400x400

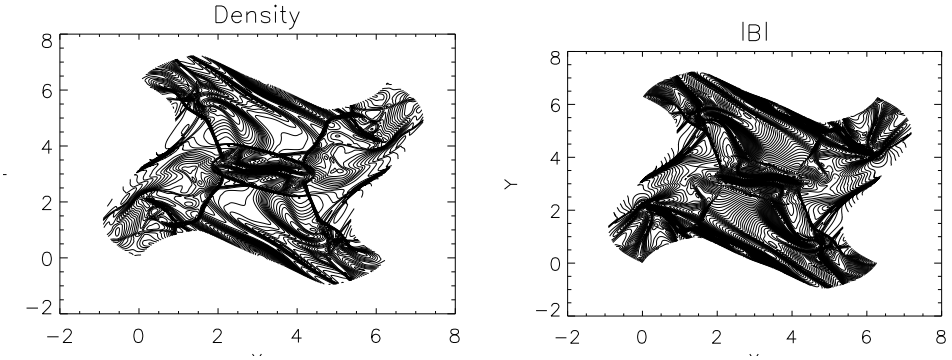


Figure 7.21: Orszag Tang Vortex, run in fully Lagrangian mode until t=1.0, fully Eulerian thereafter. 400x400

B-field is grid independent. There is no clear winner as far as resolution of central features is concerned. The final test with the Orszag Tang vortex was run with a

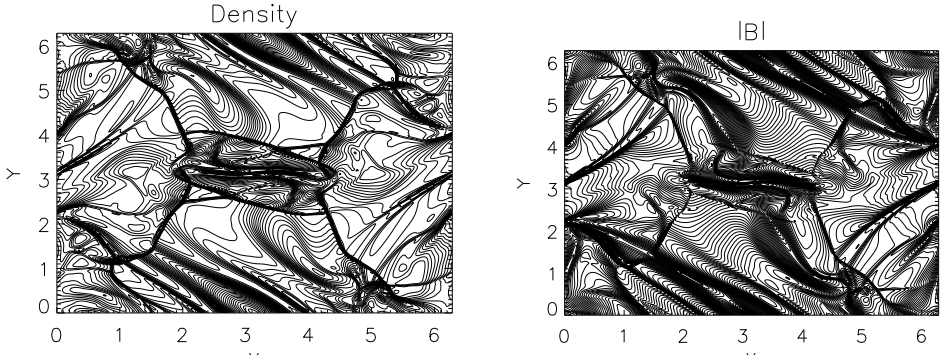


Figure 7.22: Orszag Tang Vortex, run with Gaussian remapping function. 400x400

Gaussian remapping function. This method remaps the grid every time step with a

weight of the fluid velocity. This weight is determined by a Gaussian function of the distance from the radius, such that the point at the centre is allowed to move in an entirely Lagrangian mode, whereas the points at the boundary are almost entirely Eulerian. Considering the resolution of the central features this method is perhaps the most successful.

The results presented here, whilst promising suggest improvement is required for the remapping method, particularly the need to properly upwind solutions at shocks, and to increase the order of the remap for other regions. However the Lagrangian phase seems successful at modelling flows for arbitrary grids.

Chapter 8

Ideal MHD in Cylindrical Coordinates

8.1 Review of Cylindrical Hydrodynamics

Odin carries out cylindrical hydrodynamic simulations by utilising an area weighting scheme. In order to make the magnetohydrodynamics compatible with the cylindrical calculation it is necessary to formulate an equivalent area-weighted acceleration of nodes due to magnetic forces.
The cylindrical scheme is well explained in a previous chapter, however recalling the key points is useful for the preceding discussion. The force in an area weighted scheme is derived by integrating the continuous equations around an area,

$$\vec{f}_{AW} = \int_A \vec{f} dA, \tag{8.1}$$

and the inertia of the node is defined by the product of density and cross sectional area. This area was calculated by decomposing the subzonal masses so that the mass of the node had the form,

$$M_{node} = \sum_z \sum_{i=1}^{4} w_i a_i^z \rho^z, \tag{8.2}$$

where the first summation is over all cells connected to the node, and the second over all subzonal cross sections of the cell. w_i are weightings derived from the decomposition of zonal mass. The hydrodynamic case for area weighted schemes was particularly simple in that the continuous equations had the same form in both coordinate systems, however this is not the case for the magnetic stress tensor, and

as shall be shown additional terms are needed.

8.2 Area weighted Cylindrical MHD

If the thermal pressure is neglected, the momentum equation for cylindrical coordinates is,

$$\rho \frac{D\vec{v}}{Dt} = -\nabla \cdot \boldsymbol{T}, \tag{8.3}$$

where,

$$\boldsymbol{T} = \frac{B^2}{2}\boldsymbol{I} + \vec{B}\vec{B}. \tag{8.4}$$

It's beneficial to split this tensor into two parts,

$$\begin{aligned} \boldsymbol{T} &= P_B \boldsymbol{I} + \vec{B}\vec{B} \\ &= \boldsymbol{T_1} + \boldsymbol{T_2}, \end{aligned} \tag{8.5}$$

where P_B is the magnetic pressure. Consider first $\boldsymbol{T_1}$. Taking the divergence of a rank-2 tensor,

$$\begin{aligned} \left(\nabla \cdot \boldsymbol{T_1}\right)_r &= \frac{1}{r}\frac{\partial}{\partial r}\left(rT_{rr}\right) - \frac{T_{\phi\phi}}{r} \\ &= \frac{\partial}{\partial r} P_B, \end{aligned} \tag{8.6}$$

$$\begin{aligned} \left(\nabla \cdot \boldsymbol{T_1}\right)_\phi &= \frac{1}{r}\frac{\partial}{\partial \phi} P_B \\ &= 0, \end{aligned} \tag{8.7}$$

and,

$$\left(\nabla \cdot \boldsymbol{T_1}\right)_z = \frac{\partial}{\partial z} P_B. \tag{8.8}$$

Here the case is identical to the hydrodynamic case, the continuous equations are identical in both coordinate systems, thus the magnetic pressure is implemented in the same way as the hydrodynamic pressure in an area weighted scheme.
Considering now $\boldsymbol{T_2}$,

$$\begin{aligned} \left(\nabla \cdot \boldsymbol{T_2}\right)_r &= \frac{1}{r}\frac{\partial}{\partial r}\left(B_r^2 r\right) + \frac{1}{r}\frac{\partial}{\partial \phi}\left(B_r B_\phi\right) + \frac{\partial}{\partial z}\left(B_r B_z\right) - \frac{B_{\phi^2}}{r} \\ &= \frac{\partial}{\partial r}\left(B_r^2\right) + \frac{\partial}{\partial z}\left(B_r B_z\right) + \frac{B_r^2}{r} - \frac{B_\phi^2}{r}. \end{aligned} \tag{8.9}$$

This equation must now be integrated about an area, to derive the area weighted force,

$$\begin{aligned} f_r^{AWD} &= \int_{dA} \left[\frac{\partial}{\partial r}\left(B_r^2\right) + \frac{\partial}{\partial z}\left(B_r B_z\right) + \frac{B_r^2}{r} - \frac{B_\phi^2}{r} \right] dA \\ &= \oint_{\partial\Omega} \vec{B}\left(\vec{B}\cdot\vec{n}\right) dS + \int_A \left(\frac{B_r^2}{r} - \frac{B_\phi^2}{r} \right) dA. \end{aligned} \tag{8.10}$$

The first term is identical to the Cartesian case, and can be inherited from the previous scheme, although it should be stressed that the force would now be calculated in terms of a Cartesian flux. The second two terms are however additional forces due to the geometry, and must be implemented separately. The forces derived by integrating the equations about an area must be multiplied by a radial weighting so as to be used in an equation of the form,

$$M\frac{D\vec{u}}{Dt} = \sum \vec{F}. \tag{8.11}$$

The implication of this is that rather than using the continuous source term multiplied by the volume (as would be the case if a volume weighted scheme were being used), the source term is in fact multiplied by the area, so neglecting the first two terms in (8.9),

$$f_r^{AWD} = \left(\overline{\frac{B_r^2}{r}} - \overline{\frac{B_\phi^2}{r}} \right) A r_{node}. \tag{8.12}$$

Here barred quantities represent area weighted averages around the node. As a final note, it must be remembered that the area associated with the nodal inertia in area-weight cylindrical schemes, does not correspond exactly with the area associated with a node in Cartesian coordinates. For example the area associated with the node $i = 1$ is given by,

$$a_1^{rz} = \frac{1}{16}\left(9a_1^{xy} + 3\left(a_2^{xy} + a_4^{xy}\right) + a_3^{xy}\right). \tag{8.13}$$

The other components follow a similar pattern,

$$\left(\nabla\cdot\boldsymbol{T_2}\right)_z = \frac{\partial}{\partial r}\left(B_r B_z\right) + \frac{\partial}{\partial z}\left(B_z^2\right) + \frac{B_r B_z}{r}, \tag{8.14}$$

and,

$$\left(\nabla\cdot\boldsymbol{T_2}\right)_\phi = \frac{\partial}{\partial r}\left(B_r B_\phi\right) + \frac{\partial}{\partial z}\left(B_z B_\phi\right) + \frac{2B_r B_\phi}{r}, \tag{8.15}$$

where the partial derivatives are taken from the Cartesian numerical scheme, and the other terms are added as geometric correction terms in the same manner as for the radial coordinate. The final area weighted forces are,

$$f_r^{AWD} = rA\left[f_x^{Cart.} + \overline{\frac{B_r^2 - B_\phi^2}{r}}\right], \tag{8.16}$$

$$f_\phi^{AWD} = rA\left[f_z^{Cart.} + \overline{\frac{2B_r B_\phi}{r}}\right], \tag{8.17}$$

and,

$$f_z^{AWD} = rA\left[f_y^{Cart.} + \overline{\frac{B_r B_z}{r}}\right]. \tag{8.18}$$

These equations must be used in an equation of the form,

$$rA\rho\frac{D\vec{v}}{Dt} = \vec{f}^{AWD}, \tag{8.19}$$

where A results from decomposition of the subzonal masses.

8.3 Results

The magnetised Noh's problem as defined in the Cartesian MHD result section can now be calculated in cylindrical coordinates. The results compare well to the self

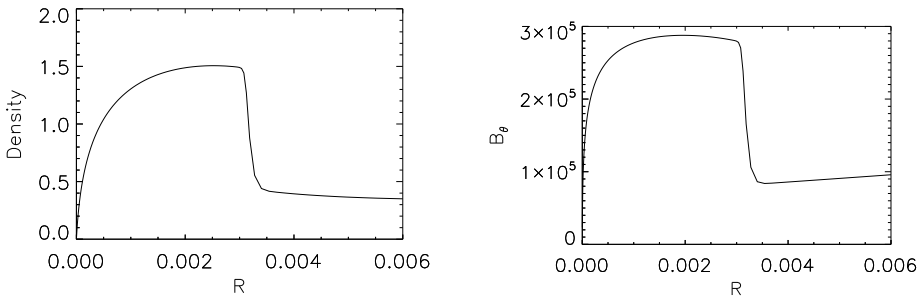


Figure 8.1: Magnetised Noh problem, run with viscosity coefficients $c_1 = 0.1$, $c_2, = 0.5$ in cylindrical coordinates. Fully Lagrangian, 250x1.

similar solutions presented in [46], as well as the previously published numerical results, [47].

8.4 Summary

An area weighted scheme for modelling magnetohydrodynamic problems in cylindrical coordinates has been developed, and tested on a basic test problem. This was done in order to try and preserve the benefits of area weighted schemes as discussed previously; specifically simplicity in adaptation of an existing Cartesian scheme, and symmetry preservation. However further testing and a detailed code comparison would be required to assess the true validity of this scheme.

Chapter 9

Second Order Remaps

In the previous section on first order remaps a swept region based remap was chosen over an intersection based remap, a decision based mostly on computational and algorithmic simplicity, however a key required improvement to the swept region based remap was identified; the need to introduce information transfer between diagonal neighbours in a single remap. There are essentially two approaches to this problem, a split remap, or direct corner transport. It is the second of these methods which will be first explained and then assessed in the context of an ALE code.

9.1 Corner Transport

Upwinded corner transport was first introduced by Colella for use with fixed grid codes, as an alternative to the operator split methods described in the next section. However their order of discussion isn't particularly important so Colella's corner transport upwinded (CTU) methods shall be described first.
Although derived for a fixed grid method, CTU has been previously suggested, e.g. [51], as a solution to the corner coupling problem of ALE code remaps. However it is most simply explained in the context of orthogonal fixed grids, before assessing within the context of ALE codes. It should be pointed out however that Colella's original formulation was not limited to orthogonal grids, and in fact he explicitly presented a formulation for non-orthogonal grids, however again for the sake of simplicity it shall be presented in an orthogonal frame.
Colella first derived the CTU scheme for the two dimensional advection problem described by,

$$\frac{\partial \rho}{\partial t} + \vec{u} \cdot \nabla \rho = 0. \tag{9.1}$$

The velocity components are denoted as,

$$\vec{u} = (u, v)\,, \tag{9.2}$$

and for simplicity shall be assumed to be constant. Consider the case shown in 9.1, given the values at $t = t_n$ the value at t_{n+1} could be estimated by tracing back the area occupied by the cell at t_{n+1} to t_n so that the updated density is given by,

$$\rho_{i,j}^{n+1} = \frac{1}{\sigma_{i,j}} \left(A_1 \rho_{i,j}^{n} + A_2 \rho_{i,j-1}^{n} + A_3 \rho_{i-1,j}^{n} + A_4 \rho_{i-1,j-1}^{n} \right), \tag{9.3}$$

where $\sigma_{i,j}$ is the total cross sectional area of the cell at $n+1$ and the cell's extent in the third direction, Δz has been cancelled. It is possible to formulate (9.1) in conservative (predictor corrector) finite volume form as,

$$\rho_{i,j}^{n+1} = \rho_{i,j}^{n} + \frac{u\Delta t}{\Delta x} \left(\rho_{i-1/2,j}^{n+1/2} - \rho_{i+1/2,j}^{n+1/2} \right) + \frac{v\Delta t}{\Delta y} \left(\rho_{i,j-1/2}^{n+1/2} - \rho_{i,j+1/2}^{n+1/2} \right), \tag{9.4}$$

where the predictor level densities are given by,

$$\rho_{i,j+1/2}^{n+1/2} = \rho_{i,j}^{n} + \frac{u\Delta t}{2\Delta x} \left(\rho_{i,j-1}^{n} - \rho_{i,j}^{n} \right), \tag{9.5}$$

and,

$$\rho_{i+1/2,j}^{n+1/2} = \rho_{i,j}^{n} + \frac{v\Delta t}{2\Delta y} \left(\rho_{i-1,j}^{n} - \rho_{i,j}^{n} \right). \tag{9.6}$$

Colella justified the derivation of (9.5) and (9.6) by noting they are equal to the average of the density of the region swept out by the cell edge during the half time step. It should be noted that (9.5) and (9.6) are upwinded based on the direction of the advective flow. Combining (9.4), (9.5) and (9.6) the following expression for $\rho_{i,j}^{n+1}$ is obtained,

$$\begin{aligned} \rho_{i,j}^{n+1} = \rho_{i,j}^{n} &+ \frac{u\Delta t}{\Delta x} \left[\rho_{i-1,j}^{n} + \frac{v\Delta t}{2\Delta y} \left(\rho_{i-1,j-1}^{n} - \rho_{i-1,j}^{n} \right) - \rho_{i,j}^{n} - \frac{v\Delta t}{2\Delta y} \left(\rho_{i,j-1}^{n} - \rho_{i,j}^{n} \right) \right] \\ &+ \frac{v\Delta t}{\Delta y} \left[\rho_{i,j-1}^{n} + \frac{u\Delta t}{2\Delta x} \left(\rho_{i-1,j-1}^{n} - \rho_{i,j-1}^{n} \right) - \rho_{i,j}^{n} - \frac{u\Delta t}{2\Delta x} \left(\rho_{i-1,j}^{n} - \rho_{i,j}^{n} \right) \right]. \end{aligned} \tag{9.7}$$

Collecting terms,

$$\begin{aligned}\rho^n_{i,j} = \rho^n_{i,j}&\left(-\frac{u\Delta t}{\Delta x}+\frac{u\Delta t}{\Delta x}\frac{v}{2}\frac{\Delta t}{\Delta y}-\frac{v\Delta t\Delta y}{+}\frac{v\Delta t}{\Delta y}\frac{u}{2}\frac{\Delta t}{\Delta x}\right)\\ &+\rho^n_{i-1,j}\left(\frac{u\Delta t}{\Delta x}-\frac{u\Delta t}{\Delta x}\frac{v}{2}\frac{\Delta t}{\Delta y}-\frac{v\Delta t}{\Delta y}\frac{u}{2}\frac{\Delta t}{\Delta x}\right)\\ &+\rho^n_{i,j-1}\left(-\frac{u\Delta t}{\Delta x}\frac{v}{2}\frac{\Delta t}{\Delta y}+\frac{v\Delta t}{\Delta y}-\frac{v\Delta t}{\Delta y}\frac{u}{2}\frac{\Delta t}{\Delta x}\right)\\ &+\rho^n_{i-1,j-1}\left(\frac{u\Delta t}{\Delta x}\frac{v}{2}\frac{\Delta t}{\Delta y}+\frac{v\Delta t}{\Delta y}\frac{u}{2}\frac{\Delta t}{\Delta x}\right).\end{aligned} \tag{9.8}$$

Rewriting once more,

$$\begin{aligned}\rho^{n+1}i,j = \frac{\rho^n_{i,j}}{A_{i,j}}&\left(A_{i,j}-u\Delta t\Delta y-v\Delta t\Delta x+uv\left(\Delta t\right)^2\right)\\ &+\frac{\rho^n_{i-1,j}}{A_{i,j}}\left(u\Delta t\Delta y-uv\left(\Delta t\right)^2\right)\\ &+\frac{\rho^n_{i,j-1}}{A_{i,j}}\left(v\Delta t\Delta x-uv\left(\Delta t\right)^2\right)\\ &+\frac{\rho^n_{i-1,j-1}}{A_{i,j}}\left(uv\left(\Delta t\right)^2\right),\end{aligned} \tag{9.9}$$

where $A_{i,j}$ is the area (volume in 3D) associated with cell (i,j). Considering figure 9.1 and rewriting in the following form,

$$\rho^{n+1}_{i,j} = \frac{1}{A_{i,j}}\left(\rho^n_{i,j}a_1+\rho^n_{i-1,j}a_3+\rho^n_{i,j-1}a_2+\rho^n_{i-1,j-1}a_4\right), \tag{9.10}$$

it is apparent this scheme is equivalent to a fist order intersection based remap. However given its formulation in (9.1) it can be implemented as a swept region based remap. The scheme demonstrated above shows a method of introducing communication between diagonal neighbours for swept region based remaps, however it needs extending to second order. This is achieved by modifying the definition of the half time step densities, however if this scheme were to be adapted to a swept region based remap it would essentially involve a modification of the definition of

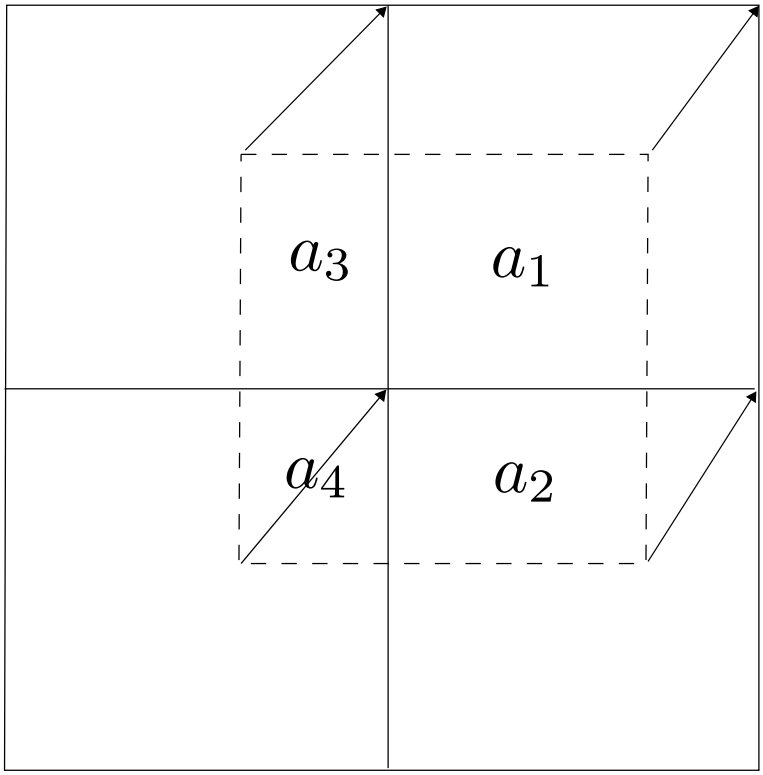


Figure 9.1: Overlap areas arising from a corner transport upwind method. The fluid is assumed to be moving with a constant velocity, shown by the arrows. The fluid parcel moves from the area shown by the dashed line to the top right solid cell in a single time step.

overlap densities. A second order estimate of the density takes the form,

$$\begin{aligned}\rho_{i+1/2,j}^{n+1/2} &= \rho_{i,j}^{n} + \frac{\Delta t}{2}\frac{\partial \rho}{\partial t} + \frac{\Delta x}{2}\frac{\partial \rho}{\partial x} \\ &= \rho_{i,j}^{n} - \frac{\Delta t}{2}\left(u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y}\right) + \frac{\Delta x}{2}\frac{\partial \rho}{\partial x} \\ &= \rho_{i,j}^{n} + \left(\frac{\Delta x}{2} - u\frac{\Delta t}{2}\right)\frac{\partial \rho}{\partial x} - \frac{v\Delta t}{2}\frac{\partial \rho}{\partial y}. \end{aligned} \tag{9.11}$$

The first and third terms in (9.11) are already included in the first order scheme previously demonstrated. From this it may be inferred that the scheme is only first order in the perpendicular direction. Thus the 2nd order correction looks like,

$$\Delta\rho_{2nd} = \left(\frac{\Delta x}{2} - u\frac{\Delta t}{2}\right)\frac{\partial \rho}{\partial x}. \tag{9.12}$$

Colella [52] used a simple central difference with van Leer's flux limiter [53] of the form,

$$\Delta x\left(\frac{\bar{\partial}\rho}{\partial x}\right) = \begin{cases} min\left(\frac{1}{2}\left|\rho_{i+1,j} - \rho_{i-1,j}\right|, 2\left|\rho_{i+1,j} - \rho_{i,j}\right|, 2\left|\rho_{i,j} - \rho_{i-1,j}\right|\right)\cdot \\ \text{SIGN}\left(\rho_{i+1,j} - \rho_{i-1,j}\right) \text{if } \left(\rho_{i+1,j} - \rho_{i,j}\right)\left(\rho_{i,j} - \rho_{i-1,j}\right) > 0 \\ 0 \text{ otherwise.} \end{cases} \tag{9.13}$$

A description of gradient/flux limiting is given in a later section. Colella's CTU scheme described above is the most basic presented in the original work [52]. For strongly non-linear problems and for gas dynamics Colella suggested a more complex algorithm. Given the apparent complexity of the CTU scheme (especially for complex grids), complications of extension to multi-material, and (as shall be shown) the simplicity of split remaps, corner transport upwinding was not pursued further in Odin.

9.2 Split Remaps

Dimensionally splitting equations is common practice in fixed grid codes. Strang [54] showed that it is second order accurate to perform half a time step in the x-direction, a full time step in the y-direction, and a final half time step in the x-direction. A more stable to version of this method [55] is to carry out a full step in the x-direction, then one in the y-direction, and accuracy could be maintained by varying the order in which the split steps were taken.

For a fixed grid, be it Eulerian or Lagrangian remap, code the directions along

which the splitting should take place are obvious; the coordinate directions. It was often argued (e.g. [51]) that because the grid in an ALE code is arbitrary, and thus not guaranteed to lie along any set of directions that splitting the remap section of an ALE code in a similar way to that of Lagrangian remap codes (e.g. [10]) would not be appropriate. However a large number of ALE codes are structured quadrilateral/hexahedral codes, and thus the grid connectivity provides a natural set of directions along which to split the remap. This idea has been used in a number of codes, for example [55], [56]. In fact this ideology can be equally applied to unstructured quadrilateral grids, the grid connectivity provides the direction along which to split the remap.

Direction here is perhaps a poor choice of nomenclature. In (directionally) splitting the remap phase of a Lagrangian remap code first the advection is carried out in the x-direction, then the y-direction. What this really means in terms of information transfer, is that the remap transfers information along the x-direction then the y-direction, or in other words through the x-interfaces and then the y-interfaces. To split a remap in an ALE code all that is required is to split the information transfer, and this is where the grid connectivity is used. Information is transferred in the *logical* x-direction and then the *logical* y-direction, thus it is clear how grid connectivity provides *directions* along which to split a remap.

The method of splitting the remap phase of an ALE code as used by [56],[55], is to first define overlap volumes in all directions. Having done this the remap masses and remap quantities of other variables are calculated in the logical x-direction, in exactly the same manner as an un-split remap. The variables are then updated according to these overlap values, and the intermediate volume calculated as the pre-remap volume updated by just the overlap volumes in the logical x-direction. These intermediate values are then used to calculate values in the logical y-direction overlap regions, before completing the remap. This has the added benefit that one-dimensional limiters can be applied without concerns over multidimensional problems (see e.g. [57], [58]), although it has the conceptual oddity that there is no intermediate grid like there is in a Lagrangian remap code. This, as will be shown in the next section limits the choice of method for extension to second order. Essentially this method uses swept volume in each logical direction as the independent coordinate rather than spatial coordinates.

A simpler method of splitting the remap is to carry out a two dimensional remap twice, by splitting the grid update first in the x-direction, then in the y-direction. Essentially this method directionally splits the grid update, and carries out two remaps. This has the advantage that having already developed a two dimensional

remap very little extra development is needed. It can also be easily applied to a non quadrilateral/hexahedral grid, as it is not dependent on each cell having neighbours in a set number of logical directions. However it may be necessary to apply true multidimensional limiters [57] when extending to higher orders, and it is not clear if Benson's claims that a multidimensional remap can never be truly more than first order have been addressed without the need for such a method as explicit corner transport.

9.2.1 Isoparametric Remaps

An isoparametric remap was initially suggested as a generalisation for ALE codes of the method used in Lagrangian remap codes. In a Lagrangian remap code the remap is split along directions defined by the mesh, thus it was suggested that having defined a remap vector for each node, this would be split in such a way that the first part of the remap is carried out parallel to one of the two mesh vectors in the logical x-direction, and the remaining displacement would be parallel to the resulting vector in the logical y-direction. This method is obviously beneficial for the remap of the B-field as it is effectively a remap in the a-space used in defining the Cauchy method. This would enable the remap methods of Lagrangian remap codes to be directly applied to this portion of the remap with only a change of coordinates necessary to adapt the method to ALE codes.

However the method has a number of complications. Firstly the choice of which of the two edge vectors in each logical direction to use isn't straight forward. The choice would (presumably) be made based on upwinding arguments, however both vectors may have positive components in the direction of the remap vector. Also the second vector is defined in terms of the first remap vector. The resulting system of equations is not trivial and may involve a global solve.

A final complication when compared to the method of Lagrangian remap codes is that although the remap may be carried out parallel to displacement vectors in one logical direction there is no guarantee that the overlap volume in the second logical direction would be zero. For example see figure 9.2. These remap volumes in the second logical direction are smaller than the overlap volumes in the primary direction, but may not always be negligible. As such isoparametric remaps are not immune to the complications and possible short falls of two dimensional remaps.

For these reasons despite the obvious advantages for the B-field remap an isoparametric remap was not pursued for use within Odin.

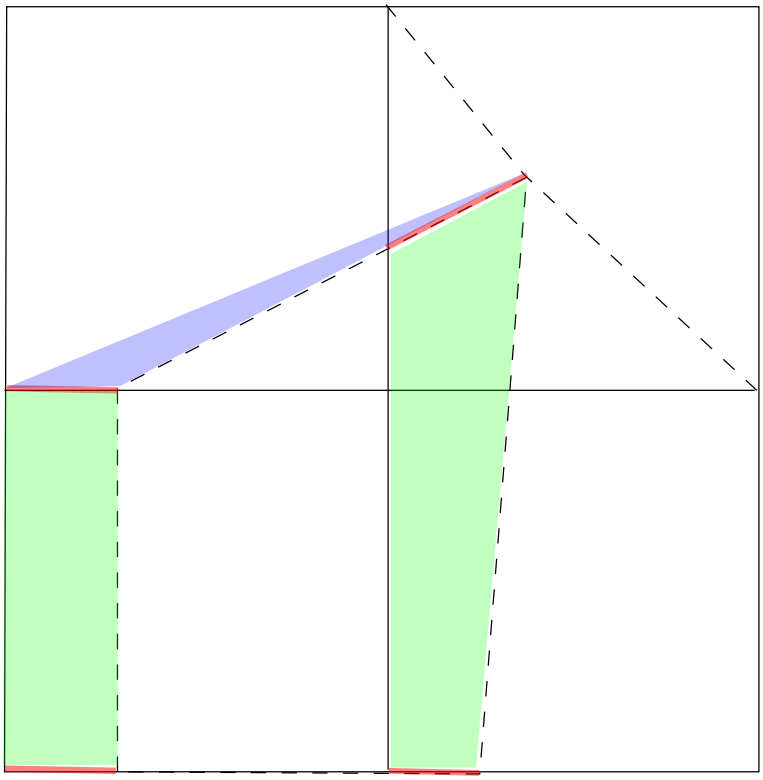


Figure 9.2: Overlap areas resulting from an isoparametric remap. Two non zero overlap areas in the (logical) parallel direction are shaded green. The remap vectors are highlighted red, and the non-zero perpendicular overlap area is shaded blue.

9.3 Extension to Second Order

The extension to a second order remap is a simple concept, in the form of a correction to the first order remap described in a previous chapter. This correction is applied to the value of the variable being remapped in the centre of the remap volume, and takes the form,

$$\rho = \rho_0 + \frac{\partial \rho}{\partial x} \Delta x, \tag{9.14}$$

where ρ_0 is the first order (donor cell) value. The variable ρ need not be density, it is only used as an example here. The variable x again need not be the x-coordinate, it is not even required to be a length, for example it could be a volume or a mass. Δx represents the change in this variable between the position at the centre of the donor cell and the centre of the overlap volume.

This method requires a limiter to be applied to the gradient to prevent the development of oscillations around shocks or rapidly variations in the variable field. Numerous limiters exist, but the theory behind them is well explained in [59], [60]. Viewed simplistically it is desirable that a limiter switches the gradient to zero if the donor cell is a local maxima or minima, and in other cases limits the gradient such that the interpolated value does not go beyond local bounds, that is it does not create a new maxima or minima in the solution. Such a scheme is known as monotonicity preserving, and a scheme which preserves monotonicity is a total variation diminishing (TVD) scheme. Total variation regions for limiter functions may be visualised using a Sweby [61]diagram.

For use within Odin Dr. C. S. Brady used a minmod limiter. A limiter (in general, but not always, see e.g. [60]) requires as an input two gradients and limit the gradient accordingly. The minmod limiter may be summarised as,

$$\frac{\bar{\partial \rho}}{\partial x} = \begin{cases} SIGN(\frac{\partial \rho}{\partial x}_l) \times min\left(\left|\frac{\partial \rho}{\partial x}_l\right|, \left|\frac{\partial \rho}{\partial x}_r\right|\right) \text{ if } SIGN(\frac{\partial \rho}{\partial x}_l) = SIGN(\frac{\partial \rho}{\partial x}_r) \\ 0 \text{ otherwise.} \end{cases} \tag{9.15}$$

Here r, l represent right and left states, although to be clear both of these gradients are upwinded. The right state is always the gradient taken across the interface in question, the left state is the gradient across the next interface in the upwind direction, which may not be left of the first state, but is denoted as such for the sake of simplicity. Clearly the sign matching fulfils the requirement of switching to first order in the case of the donor cell being a local maxima or minima. Should this not be the cases there are two possibilities. Firstly if the gradient across the interface being considered is the smaller in magnitude this gradient will be used, and the

resulting interpolated value of the variable will be a weighted average of the (true) left and right states, with non negative weights, thus the scheme does not introduce a new minima or maxima. Should the gradient across the interface be the larger then this situation does not change, except the weights are limited such that the resulting weighted average is still within local bounds.
As alluded to earlier such limiters are one dimensional in nature, they do not take into account flux transfer or variable values in other logical directions, and if used in a multidimensional remap no longer guarantee the characteristics described. This may be addressed by either multidimensional limiters [57] or some form of repair scheme (e.g. [62],[63]). The major problem in departing from local bounds is the potential loss of positivity in variables such as density, energy, or in the case of a multi-material scheme volume fraction. In the following sections two options for second order remaps in ALE codes are presented. Loss of positivity only occurred in a small number of cases, in the multi-material case where very small volume fractions existed before the remap phase. Such problems are beyond the scope of this work, but were easily mitigated using a simple repair method based around a threshold of minimum volume fractions to consider.

9.3.1 Geometric Based Remap

Essentially the only choice to be made now is how to construct the unlimited gradients. As alluded to in the introduction to this section differences will be taken across the interface around which the overlap volume is formed, and the interface to either the left or right, chosen in an upwinded manner based on the sign of the overlap volume. Thus the remaining question is what to use as the independent variable. The first method implemented by Dr C. S. Brady was to use the distance between points. So the unlimited gradient across interface i is given by,

$$\frac{\partial \rho}{\partial x} = \frac{\rho_{i+1} - \rho_i}{|\vec{r}_{i+1} - \vec{r}_i|}, \tag{9.16}$$

where $\vec{r}_i$ is the position vector of the i-th cell. Having limited the gradient the interpolated value in the overlap volume is given by,

$$\rho_{remap} = \rho_i + \frac{\bar{\partial \rho}}{\partial x} \left| \vec{r}_{overlap} - \vec{r}_i \right|, \tag{9.17}$$

assuming cell i is the donor cell, and where $\vec{r}_{overlap}$ is the position vector of the centre of the overlap volume.

In splitting this remap it is required to actually carry out two two-dimensional remaps, remapping once in the x-coordinate direction, and then another in the y-direction. Such an approach is required as the geometric remap requires an intermediate grid.
This scheme was relatively simple to implement and proved successful but had some limitations when extending Odin to multi-material. When remapping a multi-material cell materials are either remapped at the same time (in parallel) or sequentially depending on if the interface is inferred to be in parallel or perpendicular to the logical direction currently being remapped. When remapping sequentially it is necessary to know at which position a material has been exhausted, in order to re-interpolate the value of a variable to the centre of what is effectively a new remap volume. This requires solving a non trivial geometric problem. As will be shown in the following subsection a volume based remap does not require such a problem to be solved.

9.3.2 Volume Based Remaps

As suggested by the name, volume based remaps use volume as the independent coordinate to be used to construct derivatives. Volume has been used in such a manner in both un-split [11] and split [56] remaps in ALE codes. The unlimited gradient is given by,

$$\frac{\partial \rho}{\partial x} = 2\frac{\rho_{i+1} - \rho_i}{V_{i+1} + V_i}. \tag{9.18}$$

Having limited this gradient the interpolated value is given by,

$$\rho_{remap} = \rho_i + \frac{\bar{\partial}\rho}{\partial x}\frac{1}{2}\left(V_i + V_{remap}\right). \tag{9.19}$$

Having updated the density using (9.19) mass is then used as the independent coordinate rather than volume so that for calculating the second order estimate for internal energy (9.18) becomes,

$$\frac{\partial \epsilon}{\partial x} = \frac{\epsilon_{i+1} - \epsilon_i}{1/2\left(M_{i+1} + M_i\right)}, \tag{9.20}$$

and the interpolated value is given by,

$$\epsilon_{remap} = \epsilon_i + \frac{\bar{\partial}\rho}{\partial x}\frac{1}{2}\left(M_i + M_{remap}\right). \tag{9.21}$$

This method has the interesting quality that it has no intermediate grid, unlike the geometric remap method. However it is well suited to multi-material cases in that

the new position in volume space of the overlap volume is trivially calculated having exhausted a material (of known volume).

9.3.3 Extension to Magnetohydrodynamics

The remap for the B-field is carried out in a-space, the unit space used in the formulation of the Cauchy method. Firstly displacement vectors are defined for each node,

$$\vec{\Delta}s = \begin{pmatrix} x - x' \\ y - y' \end{pmatrix}. \tag{9.22}$$

A local transformation matrix at each node is then calculated,

$$A = \frac{1}{2}\begin{pmatrix} x_{i+1,j} - x_{i-1,j} & x_{i,j+1} - x_{i,j-1} \\ y_{i+1,j} - y_{i-1,j} & y_{i,j+1} - y_{i,j-1} \end{pmatrix}. \tag{9.23}$$

which enables the displacement vector to be transformed into a-space as,

$$\bar{\vec{\Delta}}s = A\vec{\Delta}s. \tag{9.24}$$

The next step is to derive a first order, upwinded scheme. In a-space the flux through a surface is equal to it's area, so the first order scheme is given as

$$\phi_{remap,x} = \begin{cases} \phi_{i+1,j}\bar{\vec{\Delta}}s_x \text{if } \bar{\vec{\Delta}}s_x > 0 \\ \phi_{i,j}\bar{\vec{\Delta}}s_x \text{ otherwise.} \end{cases} \tag{9.25}$$

What can be inferred from this is that the displacement vector represents the fraction of each flux surface being transferred. This is a natural result of the fact that the a-space is a unit space.

To extend this to second order a gradient is constructed in perpendicular B-field. The perpendicular B-field is calculated from the flux as,

$$B_\perp = \frac{\phi}{A}, \tag{9.26}$$

where A is the area of the face. The gradient is calculated with respect to the perpendicular distance, so that the unlimited gradient taken at a node (ir, iz) would be,

$$\frac{\partial B_\perp}{\partial x_1} = \frac{B_\perp^{i+1,j} - B_\perp^{i,j}}{1/2\left(l_{i+1,j} + l_{i,j}\right)}, \tag{9.27}$$

where $l_{i,j}$ is the length of a cell edge, and equal to it the face area in xy. In cylindrical coordinates an additional radial weighting is necessary.
Having calculated unlimited gradients a limited gradient is then calculated using the same method as for other variables. The final second order remapped flux looks like:

$$\phi_2 = \phi_1 + \frac{\partial \bar{B}_\perp}{\partial x_1} \cdot \left(\frac{1}{2} - \frac{\bar{\Delta} s_x}{2} \right) l_{i,j}, \tag{9.28}$$

where the fact that the remap vector in a-space represents the fraction of the cell edge (face) to be remapped has been used.

9.4 Split Volume Based Remap Method for Odin

The previous sections describe a number of remapping options. This section explains how a directionally split volume based remap has been implemented in Odin. For the sake of brevity, only the logical x-direction sweep is explained, but the necessary changes for the y-direction sweep are trivial. Again for the sake of brevity only the calculation of the unlimited gradients and calculation of overlap quantities is explained. The limiting process uses a minmod limiter as described above, and the actual update of the variables is not changed from the one directional update formulae in chapter 5, only the calculation of overlap quantities is changed.

9.4.1 Density Remap

The density remap is calculated with volume as the independent coordinate. This shown by figure 9.3. The unlimited right hand side gradient for this case is given by,

$$\frac{\partial \rho}{\partial x} = \frac{\rho_{i+1,j} - \rho_{i,j}}{v_b + v_c + v_e + v_h}, \tag{9.29}$$

and the left hand side gradient is calculated in an analogous manner. Having appropriately limited the gradient the interpolated value of the density is given by,

$$\rho_{ov} = \rho_0 + \frac{\bar{\partial \rho}}{\partial x} \frac{1}{2} \left(v_{ov} - (v_b + v_c) \right), \tag{9.30}$$

where ρ_0 is the first order (donor cell) density value. The barred gradient is the limited gradient, and v_{ov} is the overlap volume. The energy remap, shown by figure 9.4 is carried out in a similar manner. In this case, the mass is the independent variable rather than volume, and the remap is carried out in an analogous manner.

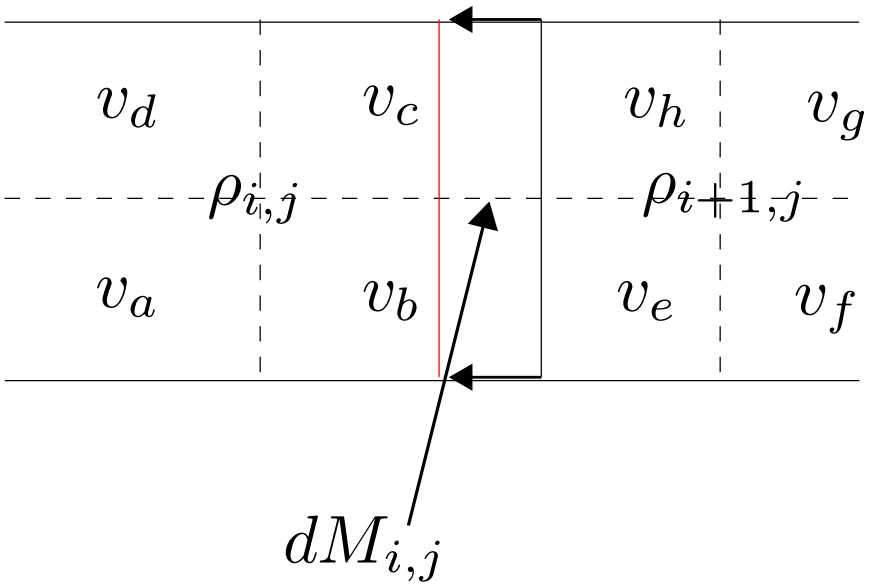


Figure 9.3: Volume based remap for density. The primary mesh is shown by solid lines, and the median mesh by dashed lines. The remapped mesh is shown in red, and the remap displacement by arrows.

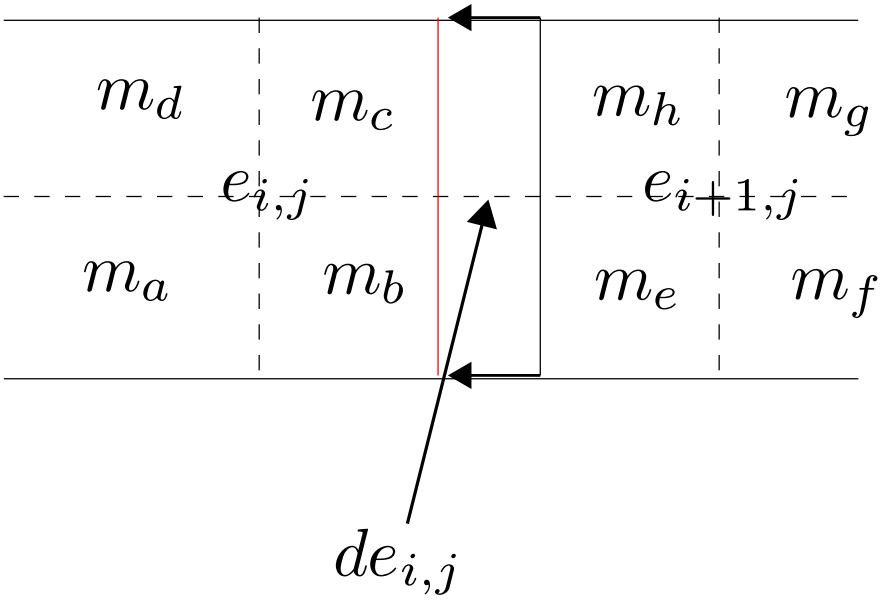


Figure 9.4: Mass based remap for energy.

The unlimited gradient is given by,

$$\frac{\partial e}{\partial x} = \frac{e_{i+1,j} - e_{i,j}}{m_b + m_c + m_e + m_h}, \tag{9.31}$$

and the interpolated value is given by,

$$e_{ov} = e_0 + \frac{\bar{\partial e}}{\partial x} \frac{1}{2} \left(m_{ov} - (m_b + m_c) \right). \tag{9.32}$$

The remap masses are then distributed to the faces of the nodal cells, but otherwise the momentum components are updated in the same manner as the energy.

Chapter 10

Implosion Tests

In order to test the final capabilites of Odin various implosion problems were run in Cartesian coordinates. The general problem consisted of an imploding dense cylindrical shell, with an imposed implosion velocity. The inner surface of the shell was perturbed with a 5% perturbation, and upon stagnation of the implosion this becomes Rayleigh Taylor unstable. At this stage bubbles and spikes form and mixing occurs.

10.1 Viscosity Testing

In order to assess the differences between edge and tensor shock viscosity an implosion problem was set up on a 256×256 grid with the following initial conditions.

$$\begin{aligned} v_x &= \frac{-15x}{r} \\ v_y &= \frac{-15y}{r} \\ P &= \begin{cases} 2.0 \text{ if } r \leq 0.3 \\ 1.0 \text{ otherwise.} \end{cases} \\ \rho &= \begin{cases} 1.0 \text{ if } 0.3 \leq r \leq 0.4 \\ 0.01 \text{ otherwise.} \end{cases} \end{aligned} \tag{10.1}$$

Here $r = \sqrt{x^2 y^2}$. The initial conditions correspond to a dense cylindrical shell imploding with a Mach number of 15. The pressure is doubled in the inner region to cause the implosion to stagnate earlier, and the inner radius was perturbed by 5%. An ideal gas equation of state was used, with $\gamma = 5/3$. These runs both used a Winslow type remapping algorithm [38]. The resultant density is illustrated by

figure 10.1 and figure 10.2. The problem was also run with tensor based viscosity on

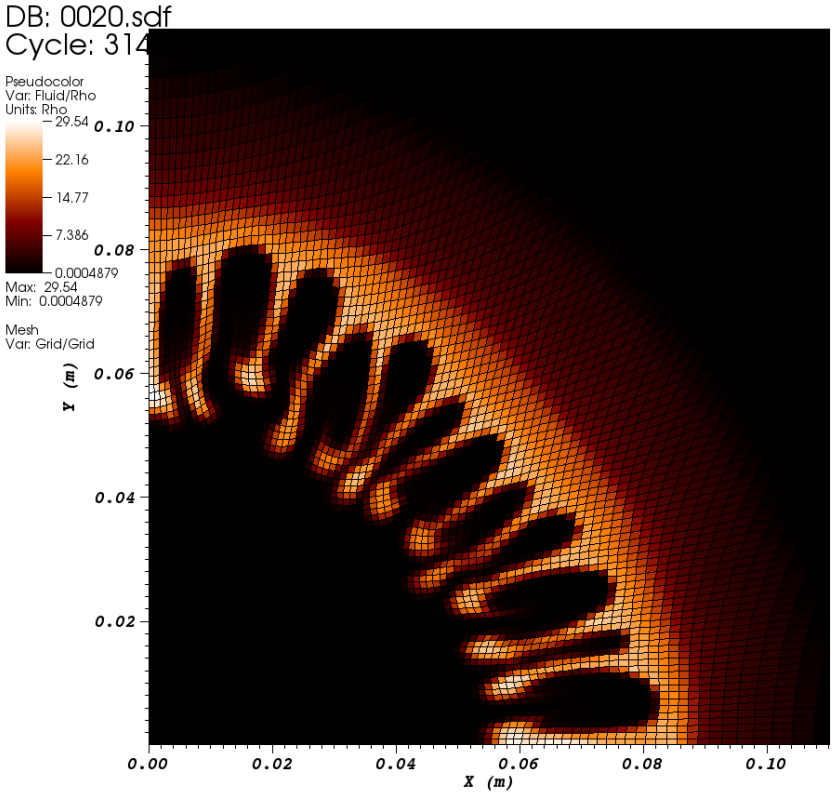


Figure 10.1: Results for implosion test with edge based shock viscosity.

a 1024×1024 (uniform) grid with Eulerian grid motion. As discussed in subsection 3.6.1 for this case the two viscosities should be very similar (the edge viscosity involves some additional averaging), so this test represented a good benchmark for the two lower resolution tests. Clearly the tensor viscosity better recovers the high resolution results. The edge viscosity results in excessive mixing, and an incorrect shock speed.

10.1.1 Implosion Test Problem with B-field

In order to investigate the effects of the addition of a B-field a similar problem was run with and without B-fields in varying directions. The problem was modified so that pressure was now unite everywhere, and the implosion was reduced to having a Mach number of 10. Otherwise the basic (without B-field) problem remained unchanged. All problems were run with viscosity coefficients of $c_1 = 0.1$ and $c_2 = 0.5$,

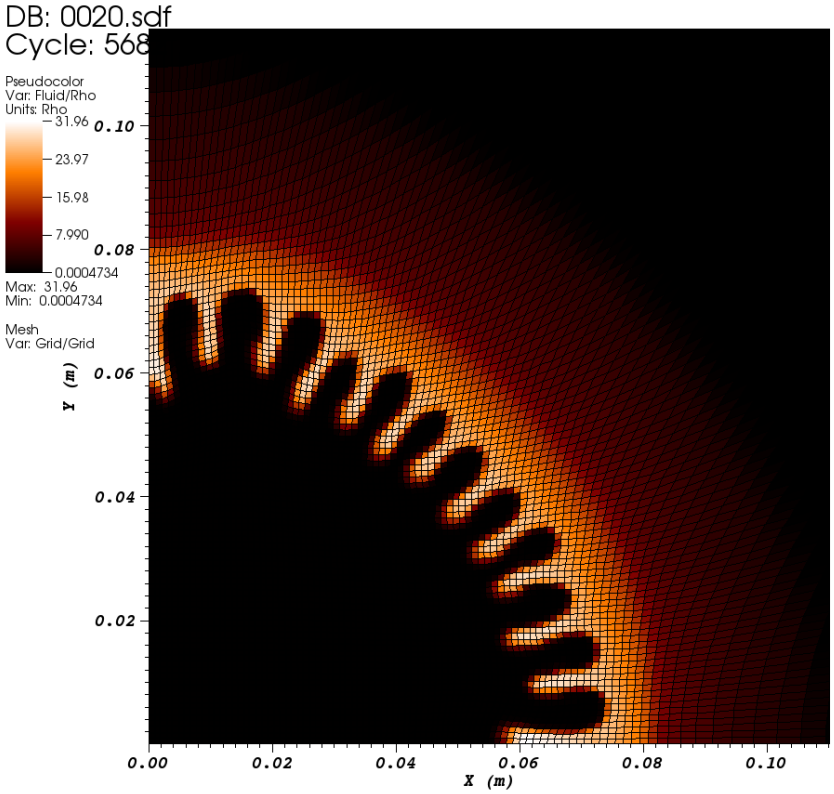


Figure 10.2: Results for implosion test with tensor shock viscosity.

and a Winslow remapping function throughout. Initially the problem was run with no B-field until t=0.1 to provide a reference solution. The resulting density plot is shown in figure 10.4. The inner surface has begun to undergo mixing caused by the Rayleigh Taylor instability.

The same problem was then run with an imposed B-field in the z-direction, of magnitude 1.0. As the B-field is in the third direction it should act in the same manner as the pressure, thus slowing the implosion.

The resulting density plot is shown in figure 10.5, again at $t = 0.1$. Clearly the implosion has been slowed significantly, with the outer edge now reaching approximately 0.26 whereas previously the corresponding position was approximately 0.18. The inner most point of the shell material has also changed from 0.075 to approximately 0.11. The corresponding plot of the magnitude of the B-field is shown in

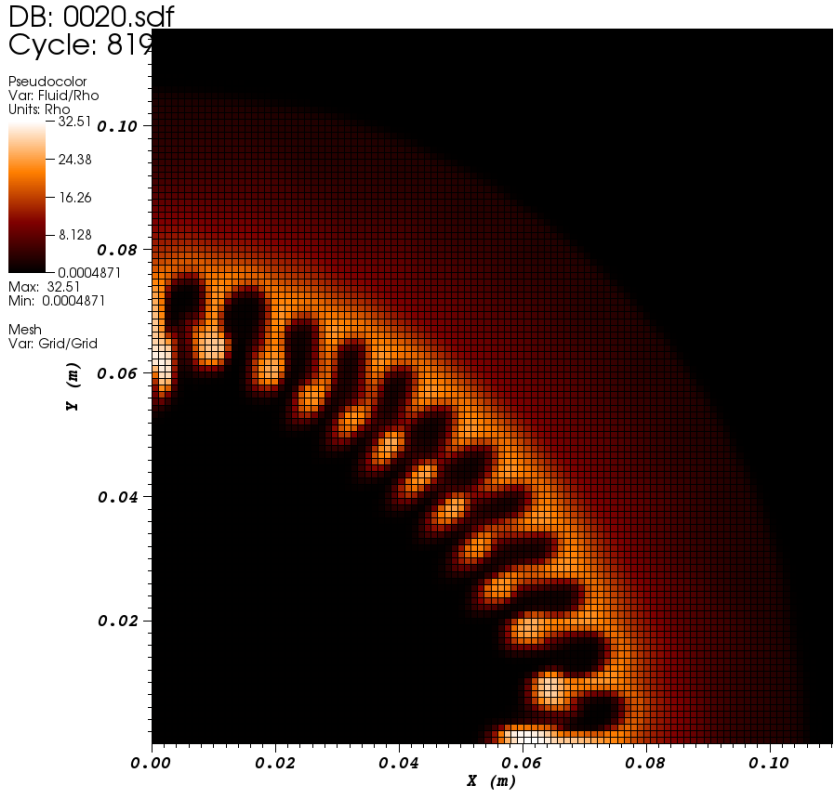


Figure 10.3: Reference solution for implosion test run with Eulerian grid motion and tensor shock viscosity.

10.6.

As discussed previously, the B-field in the third direction should act in the same manner as the thermodynamic pressure. As such by halving the initial thermodynamic pressure whilst adding a B-field of size $B_z = 1.0$ the total initial pressure would be the same as the reference problem, so it would be expected a more similar result would be obtained. This problem is illustrated by figure 10.7 and figure 10.8.

Whilst the results shown in figure 10.7 are closer to the original pure hydrodynamic test case there are still significant differences. It is worthwhile at this point to examine the evolution of thermodynamic pressure, and total pressure. In it's continuous form (neglecting viscosity), for an ideal gas the thermal pressure evolves

as,

$$\begin{aligned}\frac{D}{Dt}P &= (\gamma - 1.0)\left[\rho\frac{De}{Dt} + e\frac{D\rho}{Dt}\right] \\ &= (\gamma - 1.0)\left[-P\nabla\cdot\vec{v} - e\rho\nabla\cdot\vec{v}\right] \\ &= -\gamma P\nabla\cdot\vec{v}. \end{aligned} \tag{10.2}$$

Now including magnetic pressure the equation for total pressure becomes,

$$\begin{aligned}\frac{D}{Dt}P_{tot} &= \frac{D}{Dt}P_{thermal} + \frac{D}{Dt}P_B \\ &= -\gamma P_{thermal}\nabla\cdot\vec{v} + \frac{D}{Dt}\frac{B^2}{2}. \end{aligned} \tag{10.3}$$

For the case of the B-field being purely in the third direction, this can be simplified as,

$$\begin{aligned}\frac{D}{Dt}P_{tot} &= -\gamma P_{thermal}\nabla\cdot\vec{v} - B_z^2\nabla\cdot\vec{v} \\ &= -\gamma P_{thermal}\nabla\cdot\vec{v} - 2P_B\nabla\cdot\vec{v}. \end{aligned} \tag{10.4}$$

This has the interesting result, that if $\gamma = 2$ the magnetic and thermal pressure evolve in the same way, thus presenting an interesting test, running an implosion problem with a thermal pressure of unity should produce the same result as running with a thermal pressure of 0.5 and an initial B-field of $B_z = 1.0$, if $\gamma = 2.0$. The previous argument omits viscous heating, however if the viscous forces are equal, which they will be if the fast speed in the modified test is the same as the sound speed in the purely hydrodynamic case, then the same behaviour should be expected. It should also be noted that this argument neglects discrete considerations. Neither the thermal pressure nor the magnetic pressure are evolved directly, so some differences may occur.

To investigate if the expected behaviour can be recovered a new reference solution was calculated. This is the same problem as at the start of this section, but run with $\gamma = 2.0$. This is shown in figure 10.9. A second computation was then made, with halved thermal pressure, applied $B_z = 1.0$, and $\gamma = 1.0$. The density plot is shown for this in figure 10.11. Whilst there are still slight differences between the two results, there is a strong similarity between the two sets of results, much more so than the previous comparison when using $\gamma = 5/3$. These small differences occur due to the different methods of evolving the thermal and magnetic pressure, both in the Lagrangian step, and the remap step. The resulting B-field for the

run in figure 10.11 is shown in figure **??**. One of the major differences between hydrodynamic and magnetohydrodynamic calculations in Odin is the relaxation of the compression switch on the shock viscosity, so that it is applied everywhere rather than just cells undergoing compression. This was previously introduced in line with previous methods [10], and was shown to produce improved solutions. However it is not clear why this should be applied in calculations with a B-field only in the third direction. It is also possible that the increased viscous heating could cause the implosion speed to change, specifically make them slower. As such the same problem as shown in figure 10.7 and figure 10.8 but with the compression switch imposed.

The results from running with the compression switch imposed are shown in figure 10.12 and figure 10.13. There are some differences in the runs, most notably in the peak density, however the speeds do not to have been significantly altered. As a final test case the original problem was run, but with a uniform B-field in the x-direction. This case is a little more difficult to predict. Along the x-axis (i.e. parallel to the B-field) similar results to the case with reduced thermal pressure and imposed $B_z = 1.0$ field would be expected; as the motion is parallel to the B-field here it will not evolve, so the pressure gradient should remain unchanged. Along the y-axis the B-field will be compressed and a similar result to the first example with an imposed B-field would be expected.
Clearly these expectations are simplistic, and a break in symmetry will occur, and grow at all times. This break in symmetry will cause the results to become more difficult to predict as time increases. The results from this simulation are shown in figure 10.14 and figure 10.15.

The predictions discussed have some quantitative agreement with the results shown, particularly in the case of the inner extent of the shell material. Along the x-axis the problem with reduced thermal pressure and imposed B_z field the inner edge reached approximately 0.1, which is very similar to the position shown for the case for imposed B_x. However the position of the outer edge is approximately 0.22 when running with a purely B_z whereas with an in-plane B-field along the x-axis the outer edge is at approximately 0.16, and the surface is about to break. This is not the case along the y-axis, although the position is only slightly different, at approximately 0.17. These approximations could be better quantified by running a multi-material calculation where material position could be better known. Whilst the peak density is approximately double in the case with B_x, the peak magnitude

of the B-field is much closer to the purely out-of-plane B-field calculations. It is also apparent there is a reduced amount of Rayleigh Taylor induced mixing in the presence of imposed B_x. This may in part be due to the break in symmetry, indeed a saw tooth motion is beginning to occur due to this break in symmetry and it is likely that this will cause mixing in this region. Along the x-axis significant structure is present in the B-field caused by the beginning of Rayleigh Taylor related motion. However this localised build up of B-field appears to have stiffened the material, reducing mixing.

10.2 Summary

The first section of this chapter tested the two types of shock viscosities described earlier, but for a more realistic test problem. The tensor shock viscosity was shown to be significantly more accurate than the edge viscosity, with more accurate speeds and mixing.

In the second section implosion problems were run, with and without imposed B-fields. Whilst there were some deviations from expected behaviour, which was postulated to be caused by increased viscous heating, the results largely agreed with expectations.

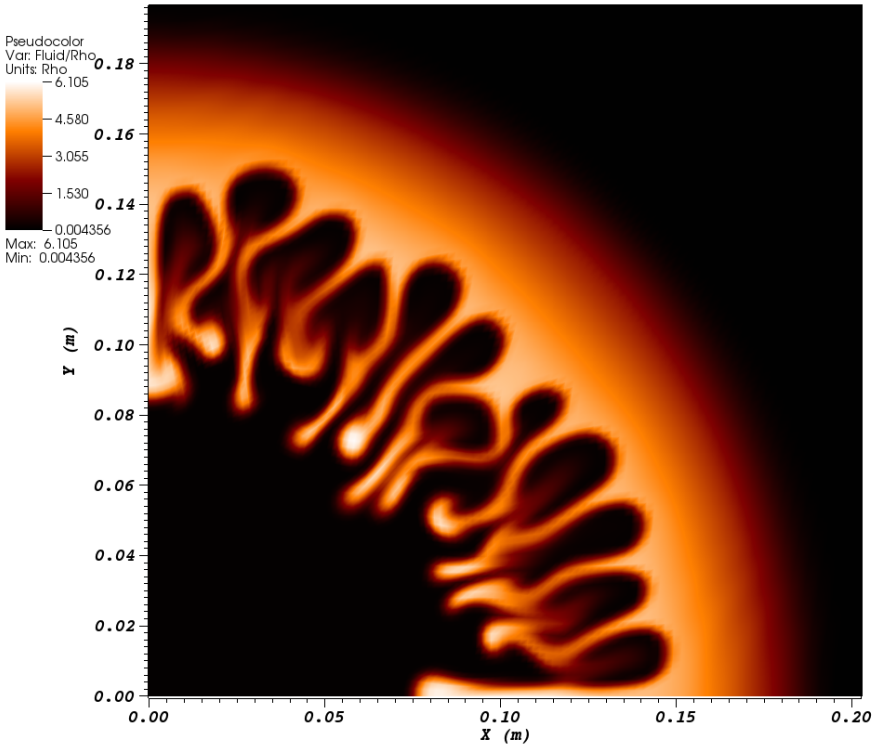


Figure 10.4: Density plot for implosion test case, without imposed B-field.

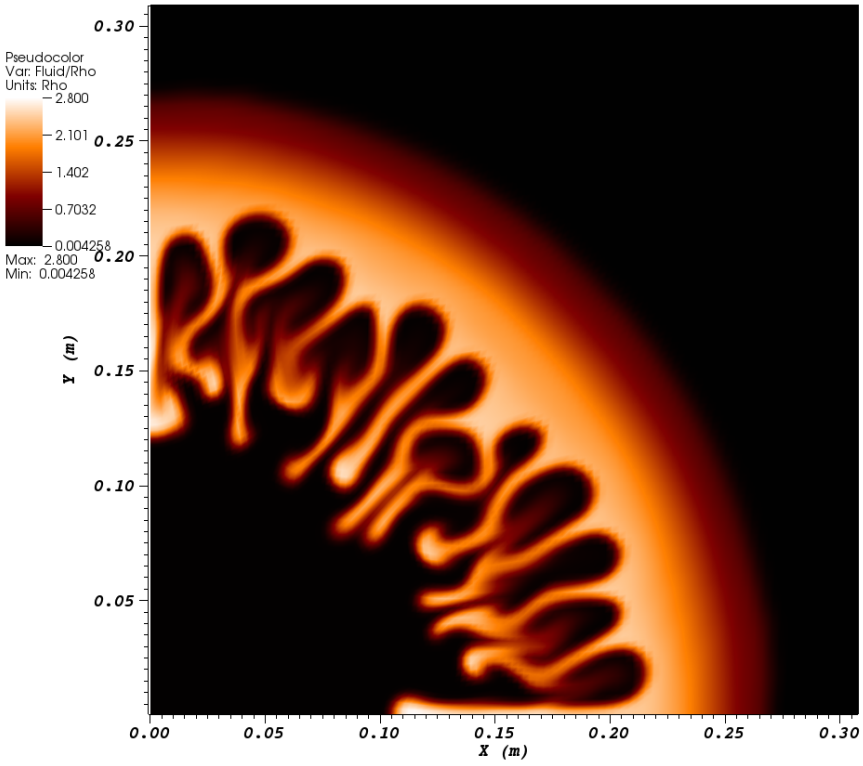


Figure 10.5: Density plot for implosion test case, with imposed B-field, in the z-direction.

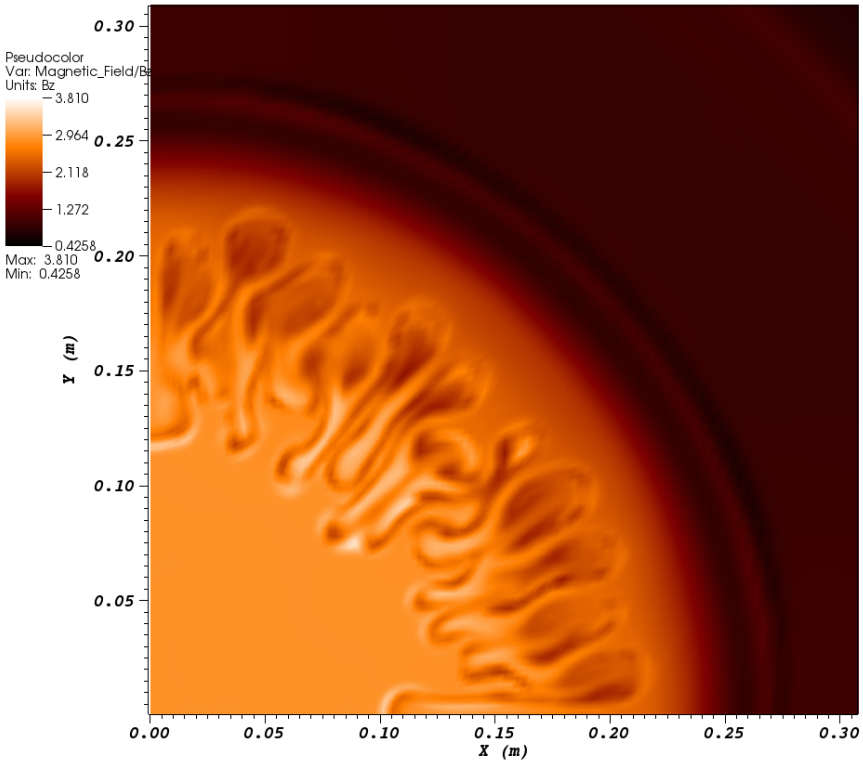


Figure 10.6: B-field plot for implosion test case, with imposed B-field, in the z-direction.

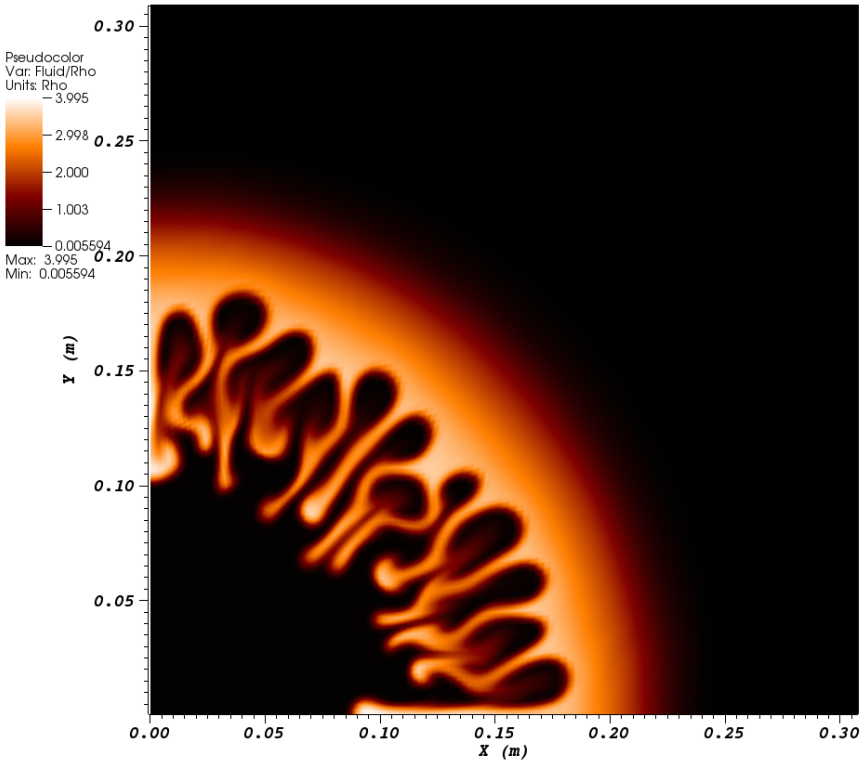


Figure 10.7: Density plot for implosion test case, with imposed B-field, in the z-direction, and reduced thermal pressure.

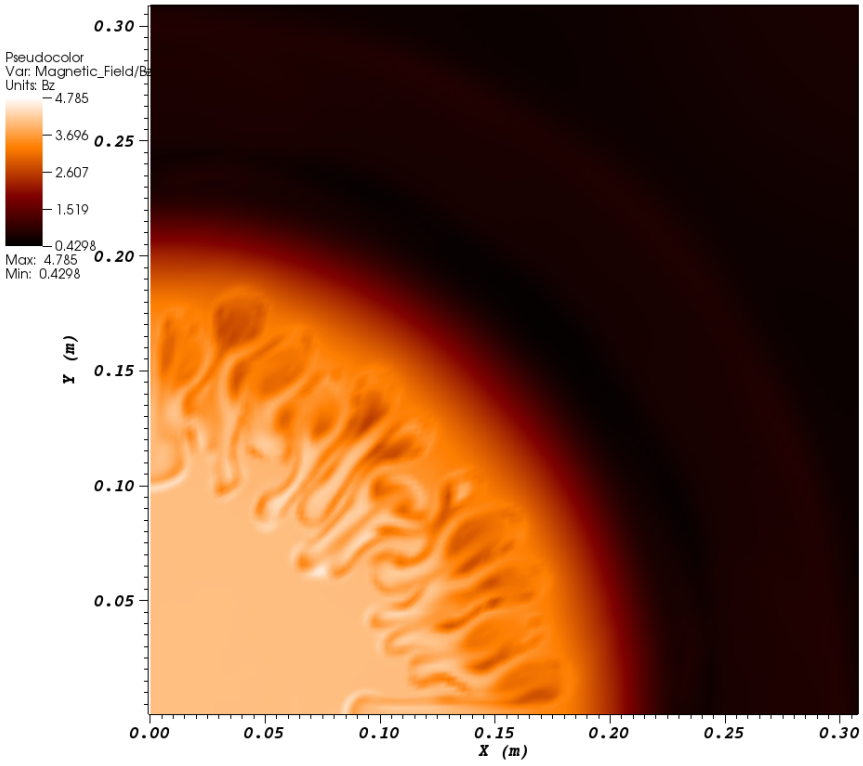


Figure 10.8: B-field plot for implosion test case, with imposed B-field, in the z-direction, and reduced thermal pressure.

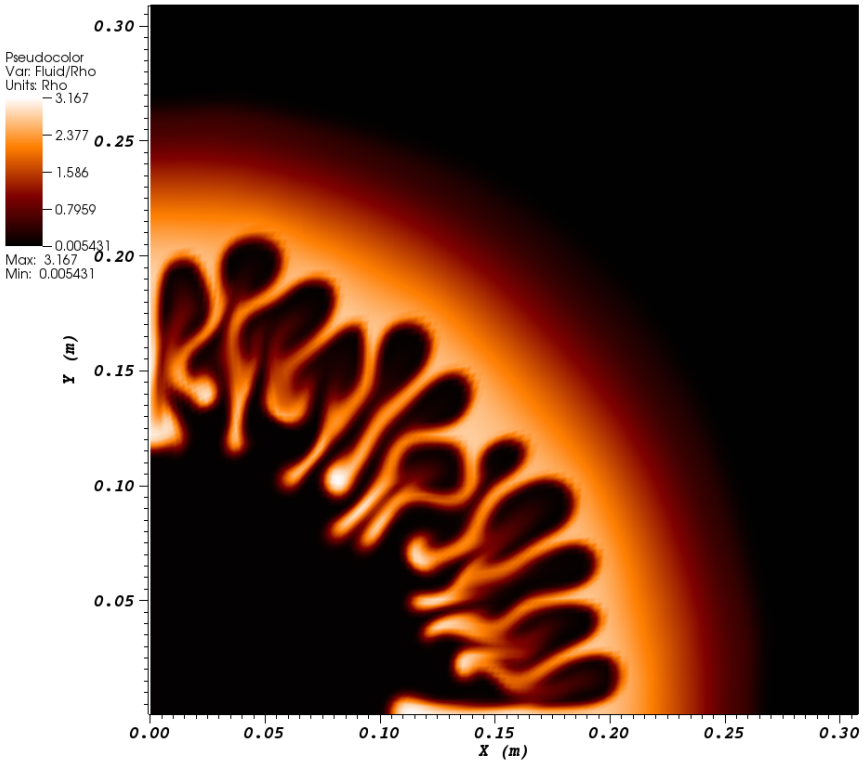


Figure 10.9: Density plot for implosion test case, without imposed B-field, and $\gamma = 2.0$.

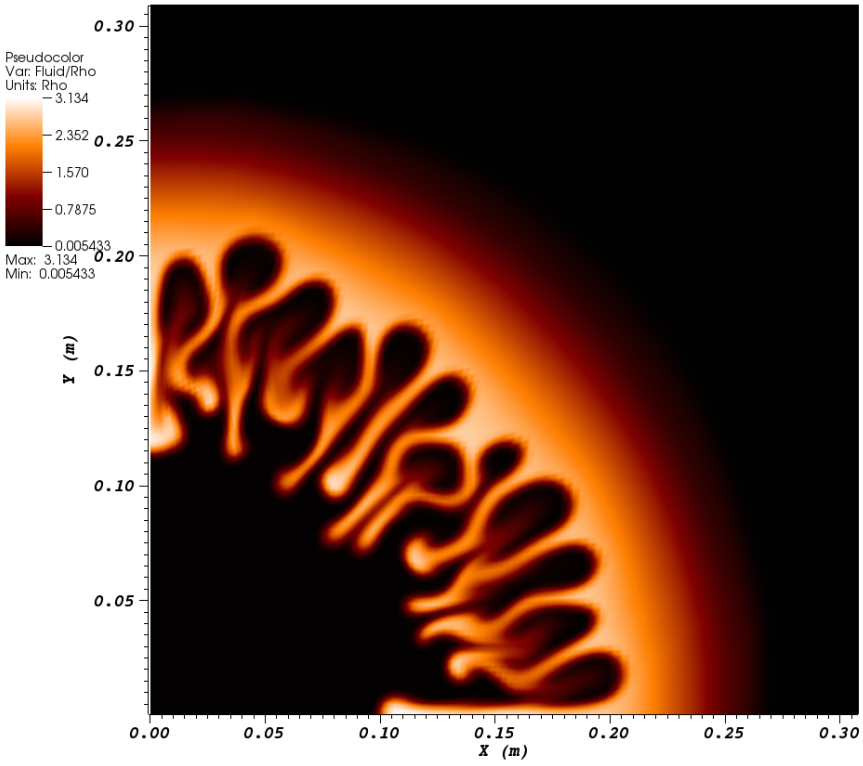


Figure 10.10: Density plot for implosion test case, with imposed B-field, reduced thermal pressure, and $\gamma = 2.0$.

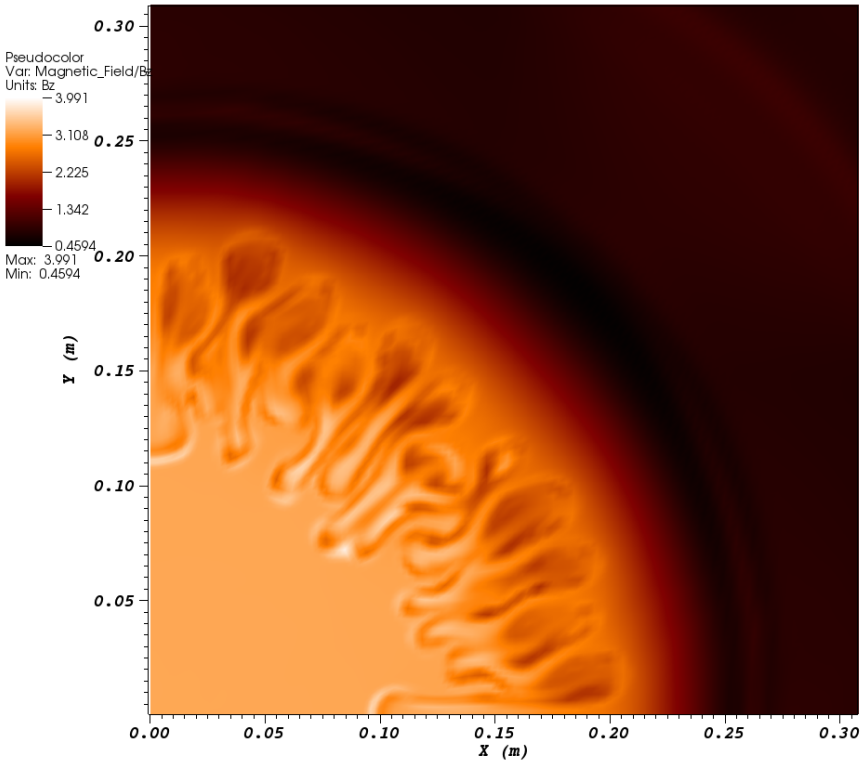


Figure 10.11: B-field plot for implosion test case, with imposed B-field, reduced thermal pressure, and $\gamma = 2.0$.

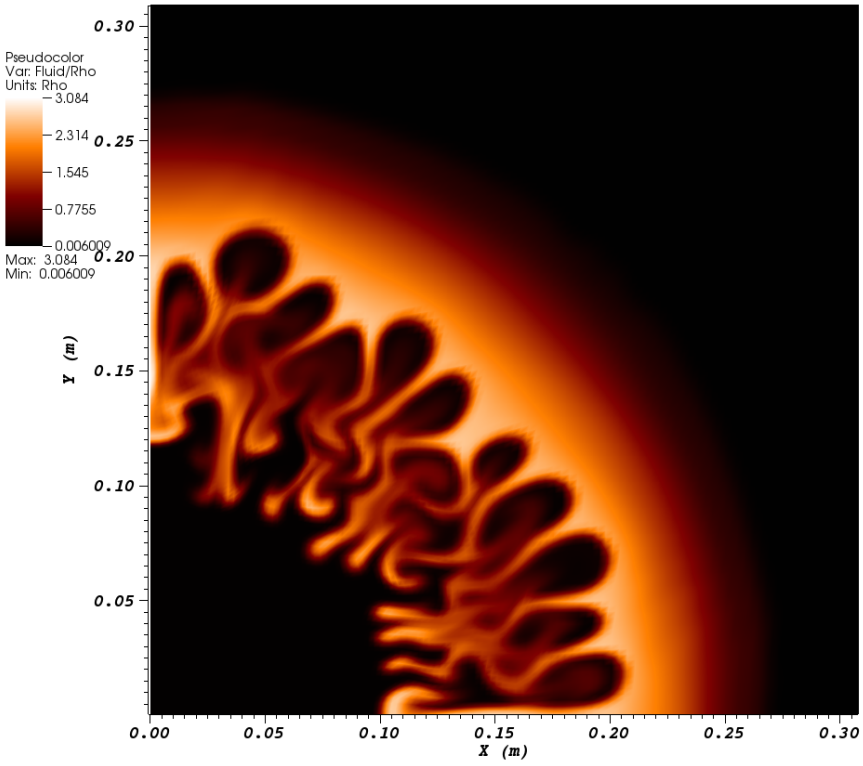


Figure 10.12: Density plot for implosion test case, with imposed B-field, in the z-direction, reduced thermal pressure, and compression switch active. $\gamma = 2.0$.

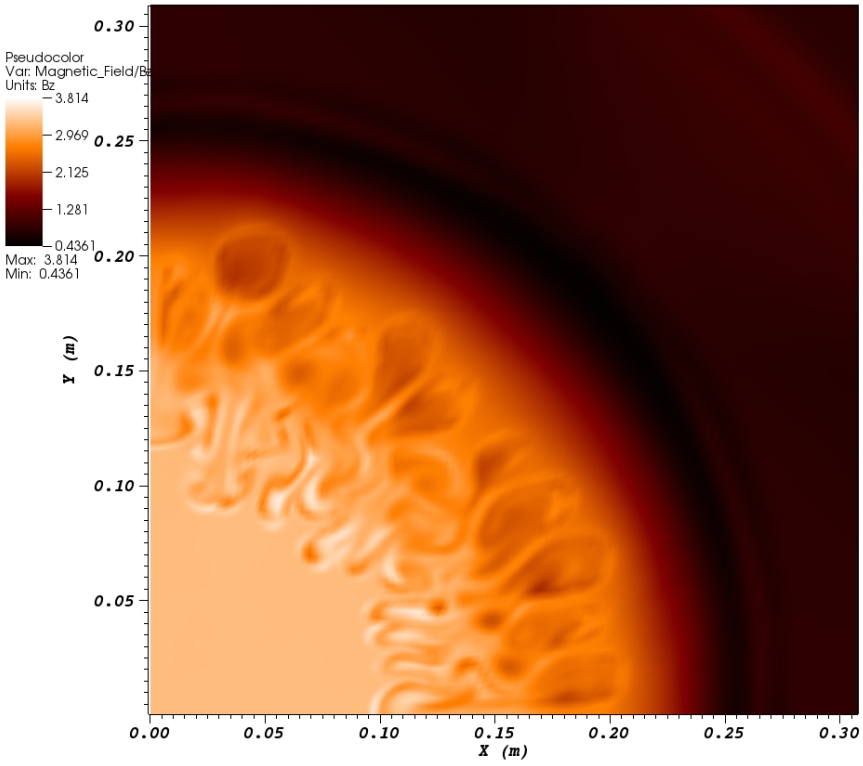


Figure 10.13: B-field plot for implosion test case, with imposed B-field, in the z-direction, reduced thermal pressure, and compression switch active. $\gamma = 2.0$.

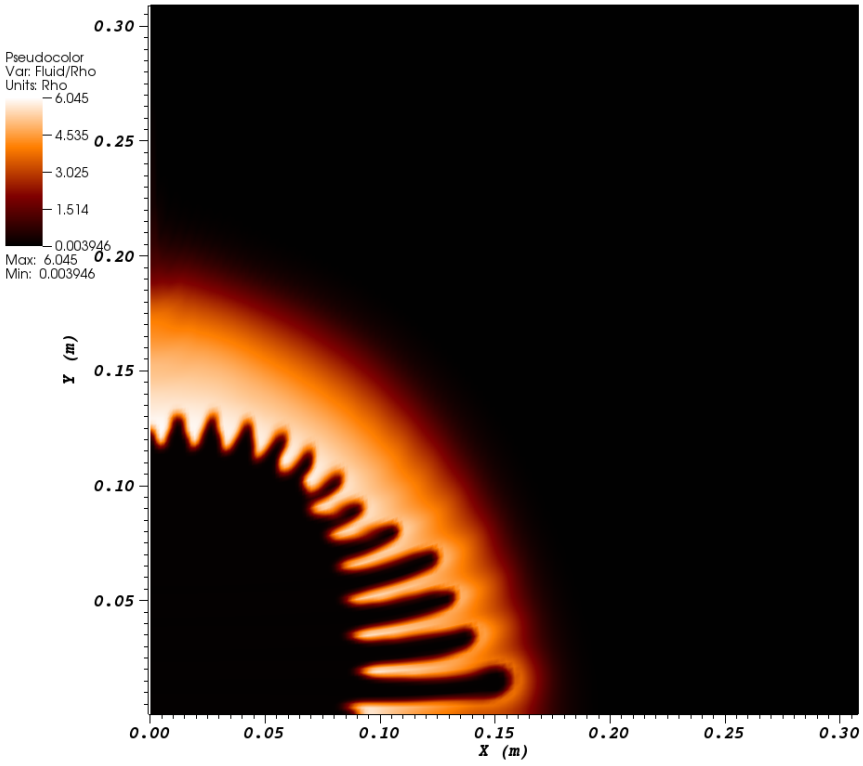


Figure 10.14: Density plot for implosion test case, with imposed B-field in the x-direction. $\gamma = 2.0$.

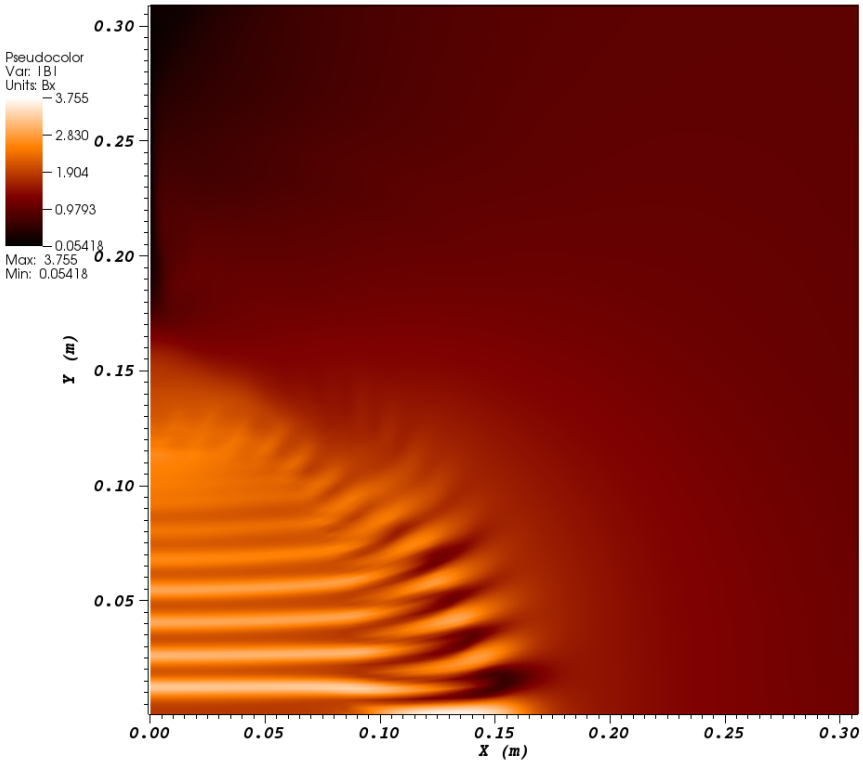


Figure 10.15: B-field plot for implosion test case, with imposed B-field in the x-direction. $\gamma = 2.0$.

Chapter 11

Further Work

11.1 Cylindrical Magnetohydrodynamics

Of the work presented in this thesis the section requiring the most additional work is the expansion to cylindrical magnetohydrodynamics. The test cases presented shows promising results, but has an orthogonal grid for all time. To be completely sure of the accuracy and successful implementation of the cylindrical MHD scheme an in-depth comparison with existing codes would have to be made. This is due to the lack of test problems with known solutions for such a scheme, in particular test problems with a non-zero $r - z$ B-field.

11.2 Multi-material

In order to fully take advantage of the strengths of an ALE scheme a multi-material code is desirable. Much of this work has already been completed by Dr C. S. Brady in conjunction with the author. This includes extension to arbitrary equation of state, and material interface reconstruction within the remap, as well as the generalisation of the Lagrangian scheme to more than one material; this involves the calculation of an effective pressure for the combination of all materials.

11.3 Additional Physics

Beyond this the next step to take would be to add additional physics such as radiative transport and laser ray tracing. Both of these, as with other extra physics are complicated by the arbitrary nature of the grid, and as mentioned in the introduction to this work this is one of the disadvantages of the ALE method. However this

work, in particular the final section on implosions has demonstrated the significant strengths of an ALE scheme.

Appendix A

Tensor Preliminaries

A.1 Tensor Preliminaries

As dyadics are rank 2 (contravariant) tensors, it is necessary to introduce the divergence theorem for tensors, and a few tensor definitions. Firstly, the dyadic is defined as,

$$\boldsymbol{A} = \vec{a} \otimes \vec{b}, \tag{A.1}$$

where,

$$A^{ij} = a^i b^j. \tag{A.2}$$

The divergence theorem for a general tensor is [64],

$$\int_V \frac{\partial T^{ij\ldots k\ldots m}}{\partial x_k} = \oint_s T^{ij\ldots k\ldots m} n_k ds, \tag{A.3}$$

where tensor contraction occurs over the index k. A surface (n_k in (A.3) is in fact a rank one covariant tensor (a one form, see e.g. [65]), and the definition of (A.3) has been limited to a general rank m contravariant tensor for simplicity.
Tensor contraction is a mathematical operation which reduces the rank of a tensor by summing over a pair of repeated indices. Some texts (e.g. [66] will state a stronger definition of tensor contraction, that it requires one of the repeated indices to be contravariant and the second covariant. Tensor contraction is in fact the second half of the calculation of the dot product of two vectors, so it is worth examining these concepts within a familiar context,

$$\vec{a} \cdot \vec{b} = a^i b^j \eta_{ij} = a^i b_i, \tag{A.4}$$

where $\boldsymbol{\eta}$ is the metric tensor. In the form of (A.4) the metric tensor is a rank 2 covariant tensor. The second equality in (A.4) is the result of a process known as index lowering, which has transformed the vector (a rank one contravariant tensor) into a one-form (a rank one covariant tensor). The final result for the dot product is calculated by contracting these two tensors over their repeated index. There is a corresponding index raising operation, which can give a mixed or purely contravariant form of the metric tensor. A tensor may also be contracted on itself.
It is clear that for a rank n (contravariant tensor) there are n possible contractions with the surface normal. Examining the two cases for the dyadic,

$$a^i b^j n_j = \vec{a}\left(\vec{b}\cdot\vec{n}\right), \tag{A.5}$$

and,

$$a^i b^j n_i = \vec{b}\left(\vec{a}\cdot\vec{n}\right), \tag{A.6}$$

where in both (A.5) and (A.6) the vector $\vec{n}$ is the vector corresponding to having raised the index on the surface normal, to aid comparison with vector calculus. It is common to see (e.g. [67]) equations relating the dot product between a dyadic and a vector, which gives results equivalent to (A.5) and (A.6). Presumably some generalised form of the dot product based on (A.4) has been carried out so that,

$$\begin{aligned}\vec{n}\cdot\left(\vec{a}\otimes\vec{b}\right) &= n^i a^j b^k \eta_{ij} \\ &= n^i a^j \eta_{ij} b^k \\ &= \left(\vec{n}\cdot\vec{a}\right)\vec{b},\end{aligned} \tag{A.7}$$

and similarly,

$$\begin{aligned}\left(\vec{a}\otimes\vec{b}\right)\cdot\vec{n} &= a^j b^k n^i \eta_{ki} \\ &= \left(\vec{n}\cdot\vec{b}\right)\vec{a}.\end{aligned} \tag{A.8}$$

Clearly whilst (A.7) and (A.8) aren't particularly well defined, they do represent similar relationships to (A.5) and (A.6) and are sometimes used in conjunction with the tensor divergence theorem. As such they are included to aid comparison with other texts.
Like the surface normal, the divergence operator is a one form. Taking the divergence of a tensor is simply another contraction operation, and as such there are n possible divergences for a rank n tensor. However for the purpose of these discussions, the

tensor divergence shall be defined as,

$$\nabla \cdot \boldsymbol{T} = \frac{\partial T_{ji}}{\partial x_j}, \tag{A.9}$$

for a rank 2 (contravariant) tensor, $\boldsymbol{T}$. This builds upon the previous concept of a generalised dot product. Using this definition of the divergence of a rank two tensor the divergence of a dyadic can be shown to be,

$$\begin{aligned}
\nabla \cdot \left(\vec{a} \otimes \vec{b}\right) &= \frac{\partial a^i b^j}{\partial x^i} \\
&= \left(\frac{a^1 b^1}{\partial x^1} + \frac{\partial a^2 b^1}{\partial x^2} + \frac{\partial a^3 b^1}{\partial x^3}\right) \vec{e_1} \\
&+ \left(\frac{a^1 b^2}{\partial x^1} + \frac{\partial a^2 b^2}{\partial x^2} + \frac{\partial a^3 b^2}{\partial x^3}\right) \vec{e_2} \\
&+ \left(\frac{a^1 b^3}{\partial x^1} + \frac{\partial a^2 b^3}{\partial x^2} + \frac{\partial a^3 b^3}{\partial x^3}\right) \vec{e_3} \\
&= \left(\nabla \cdot \vec{a}\right) \vec{b} + \left(\vec{a} \cdot \nabla\right) \vec{b}.
\end{aligned} \tag{A.10}$$

Similarly, the definition of tensor divergence, (A.9) and be combined with the divergence theorem for tensors, (A.3) to give,

$$\begin{aligned}
\int_V \nabla \cdot \left(\vec{a} \otimes \vec{b}\right) &= \int_V \frac{\partial \left(a^i b^j\right)}{\partial x_i} \\
&= \oint_s a^i b^j n_i ds \\
&= \oint_s \left(\vec{a} \cdot \vec{n}\right) \vec{b}.
\end{aligned} \tag{A.11}$$

Appendix B

Summary of Odin

B.1 Summary of Odin program flow

This appendix contains a brief summary of the program flow within Odin. It deliberately omits a number of details such as file input and output, and various parallel considerations, instead concentrating on the physical applications.

1. Calculate initial grid positions, cell volumes and subzone volumes.
2. Calculate initial values of density, energy and velocity components according to initial conditions. Also calculate conserved quantities, subzonal masses and pre-initial B-field.
3. Call boundary conditions.
4. Begin main evolution loop:
   (a) Calculate forces due to shock viscosity.
   (b) Calculate time step limited by the CFL condition, and the requirement that grid cells do not cross over.
   (c) Carry out a predictor-corrector Lagrangian time step, evolving density, specific internal energy, and velocity components. Also update grid positions, and volumes.
   (d) Call boundary conditions.
   (e) Decide if a remap should be applied. If not, return to (4.a)4, otherwise:
      i. Calculate new grid positions, according to remap strategy.
      ii. Carry out a B-field remap, calculating new values for pre-initial B-field.

iii. Remap third component of B-field.

iv. Calculate remap volumes.

v. Calculate remap masses, and remap density.

vi. Remap specific internal energy.

vii. Calculate nodal remap masses.

viii. Remap velocity components.

ix. Calculate new corner masses.

x. Call boundary conditions, and return to (4.*a*).

Bibliography

[1] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[2] Culbert B Laney. *Computational gasdynamics*. Cambridge University Press, 1998.

[3] Joe D Hoffman and Steven Frankel. *Numerical methods for engineers and scientists*. CRC press, 2001.

[4] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer, 2009.

[5] Sergei Konstantinovich Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959.

[6] Rosa Donat and Antonio Marquina. Capturing shock reflections: an improved flux formula. *Journal of Computational Physics*, 125(1):42–58, 1996.

[7] AJ Barlow and PL Roe. A cell centred lagrangian godunov scheme for shock hydrodynamics. *Computers & Fluids*, 46(1):133–136, 2011.

[8] WD Schulz. Two-dimensional lagrangian hydrodynamic difference equations, in" methods in computational physics", vol. 3, alder, fernbach and rotenberg, eds, 1964.

[9] William F Noh and Paul Woodward. Slic (simple line interface calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28–July 2, 1976 Twente University, Enschede*, pages 330–340. Springer, 1976.

[10] T.D. Arber, A.W. Longbottom, C.L. Gerrard, and A.M. Milne. A staggered grid, lagrangianeulerian remap code for 3-d mhd simulations. *Journal of Computational Physics*, 171(1):151 – 181, 2001.

[11] Andrew Barlow. *An adaptive multi-material arbitrary Lagrangian Eulerian algorithm for computational shock hydrodynamics*. PhD thesis, University of Wales Swansea, 2002.

[12] Jerrold E Marsden and Anthony Tromba. *Vector calculus*. Macmillan, 2003.

[13] E.J. Caramana, D.E. Burton, M.J. Shashkov, and P.P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics*, 146(1):227 – 262, 1998.

[14] John VonNeumann and RD Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–237, 1950.

[15] R Landschoff. A numerical method for treating fluid flow in the presence of shocks. Technical report, Los Alamos Scientific Laborarory Report LA-1930, 1955.

[16] V. Kuropatenko. *Dierence Methods for Solutions of Problems of Mathematical Physics*. 1967.

[17] Mark L. Wilkins. Use of artificial viscosity in multidimensional fluid dynamic calculations. *Journal of Computational Physics*, 36(3):281 – 303, 1980.

[18] EJ Caramana, MJ Shashkov, and PP Whalen. Formulations of artificial viscosity for multi-dimensional shock wave computations. *Journal of Computational Physics*, 144(1):70–97, 1998.

[19] EJ Caramana and R Loub re. Curl-q: A vorticity damping artificial viscosity for essentially irrotational lagrangian hydrodynamics calculations. *Journal of Computational Physics*, 215(2):385–391, 2006.

[20] DL Hicks. Stability analysis of wondy (a hydrocode based on the artificial viscosity method of von neumann and richtmyer) for a special case of maxwells law. *Mathematics of computation*, 32(144):1123–1130, 1978.

[21] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1): 1–31, 1978.

[22] John K Dukowicz and Bertrand JA Meltz. Vorticity errors in multidimensional lagrangian codes. *Journal of Computational Physics*, 99(1):115–134, 1992.

[23] William F Noh. Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux. *Journal of Computational Physics*, 72(1): 78–120, 1987.

[24] JC Campbell and MJ Shashkov. A tensor artificial viscosity using a mimetic finite difference algorithm. *Journal of Computational Physics*, 172(2):739–765, 2001.

[25] James M. Hyman and Mikhail Shashkov. Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids. *Applied Numerical Mathematics*, 25(4):413 – 442, 1997.

[26] J.M. Hyman and M. Shashkov. Natural discretizations for the divergence, gradient, and curl on logically rectangular grids. *Computers & Mathematics with Applications*, 33(4):81 – 104, 1997.

[27] LI Sedov. Similarity and dimensional methods in mechanics (similarity and dimensional methods in mechanics, new york, 1959.

[28] E. J. Caramana and P. P. Whalen. Numerical preservation of symmetry properties of continuum problems. *J. Comput. Phys.*, 141(2):174–198, 1998.

[29] ML Wilkins. Calculation of elastic-plastic flow, in" methods in computational physics", vol. 3, alder, fernbach and rotenberg, eds, 1964.

[30] Jeremiah W Murphy and Adam Burrows. Bethe-hydro: an arbitrary lagrangian-eulerian multidimensional hydrodynamics code for astrophysical simulations. *The Astrophysical Journal Supplement Series*, 179(1):209, 2008.

[31] EJ Caramana and MJ Shashkov. Elimination of artificial grid distortion and hourglass-type motions by means of lagrangian subzonal masses and pressures. *Journal of Computational Physics*, 142(2):521–561, 1998.

[32] PL Browne and KB Wallick. Reduction of mesh tangling in two-dimensional lagrangian hydrodynamics codes by the use of viscosity, artificial viscosity, and tts (temporary triangular subzoning for long, thin zones). Technical report, Los Alamos Scientific Lab., N. Mex., 1971.

[33] Robert E. Peterkin Jr., Michael H. Frese, and Carl R. Sovinec. Transport of magnetic flux in an arbitrary coordinate ale code. *Journal of Computational Physics*, 140(1):148 – 171, 1998.

[34] M Kucharik and M Shashkov. Extension of efficient, swept-integration-based conservative remapping method for meshes with changing connectivity. *International journal for numerical methods in fluids*, 56(8):1359–1365, 2008.

[35] L.G. Margolin and Mikhail Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics*, 184(1):266 – 298, 2003.

[36] Markus Berndt, Jérôme Breil, Stéphane Galera, Milan Kucharik, Pierre-Henri Maire, and Mikhail Shashkov. Two-step hybrid conservative remapping for multimaterial arbitrary lagrangian–eulerian methods. *Journal of Computational Physics*, 230(17):6664–6687, 2011.

[37] M Kucharik, J Breil, S Galera, P-H Maire, M Berndt, and M Shashkov. Hybrid remap for multi-material ale. *Computers & Fluids*, 46(1):293–297, 2011.

[38] Alan M Winslow. Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh. *Journal of Computational Physics*, 1(2):149 – 172, 1966.

[39] C. R. Evans and J. F. Hawley. Simulation of magnetohydrodynamic flows - A constrained transport method. *Astrophysical Journal*, 332:659–677, September 1988.

[40] J.U Brackbill and D.C Barnes. The effect of nonzero $\nabla \cdot \mathrm{B}$ on the numerical solution of the magnetohydrodynamic equations. *Journal of Computational Physics*, 35(3):426 – 430, 1980.

[41] Andrew L. Zachary, Andrea Malagoli, and Phillip Colella. A higher-order godunov method for multidimensional ideal magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 15(2):263–22, 1994.

[42] Dinshaw S Balsara and Jongsoo Kim. A comparison between divergence-cleaning and staggered-mesh formulations for numerical magnetohydrodynamics. *The Astrophysical Journal*, 602(2):1079, 2008.

[43] M Brio and C.C Wu. An upwind differencing scheme for the equations of ideal magnetohydrodynamics. *Journal of Computational Physics*, 75(2):400 – 422, 1988.

[44] H. K. Moffatt. *Magnetic field generation in electrically conducting fluids*. 1978.

[45] I. J. D. Craig and A. D. Sneyd. A dynamic relaxation technique for determining the structure and stability of coronal magnetic fields. *Astrophysical Journal*, 311:451–459, December 1986.

[46] AL Velikovich, JL Giuliani, ST Zalesak, JW Thornhill, and TA Gardiner. Exact self-similar solutions for the magnetized noh z pinch problem. *Physics of Plasmas*, 19:012707, 2012.

[47] P Tzeferacos, M Fatenejad, N Flocke, G Gregori, DQ Lamb, D Lee, J Meinecke, A Scopatz, and K Weide. Flash magnetohydrodynamic simulations of shock-generated magnetic field experiments. *High Energy Density Physics*, 2012.

[48] Gábor Tóth. The $\nabla \cdot \mathrm{B} = 0$ constraint in shock-capturing magnetohydrodynamics codes. *Journal of Computational Physics*, 161(2):605–652, 2000.

[49] Steven A Orszag and Cha-Mei Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, 90(01):129–143, 1979.

[50] Wenlong Dai and Paul R Woodward. A simple finite difference scheme for multidimensional magnetohydrodynamical equations. *Journal of Computational Physics*, 142(2):331–369, 1998.

[51] A Barlow, D Burton, and M Shashkov. Compatible, energy and symmetry preserving 2d lagrangian hydrodynamics in rzcylindrical coordinates. *Procedia Computer Science*, 1(1):1893–1901, 2010.

[52] Phillip Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87(1):171–200, 1990.

[53] Bram van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of Computational Physics*, 32(1):101 – 136, 1979.

[54] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[55] J. M. Morrell. *A cell by cell anisotropic adaptive mesh Arbitrary Lagrangian Eulerian method for the numerical solution of the Euler equations.* PhD thesis, The University of Reading, 2007.

[56] James S Peery and Daniel E Carroll. Multi-material ale methods in unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 187(3): 591–619, 2000.

[57] Steven T Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of computational physics*, 31(3):335–362, 1979.

[58] David J Benson. Computational methods in lagrangian and eulerian hydrocodes. *Computer methods in Applied mechanics and Engineering*, 99(2): 235–394, 1992.

[59] Bram van Leer. Towards the ultimate conservative difference scheme. ii. monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14(4):361 – 370, 1974.

[60] Bram Van Leer. Towards the ultimate conservative difference scheme iii. upstream-centered finite-difference schemes for ideal compressible flow. *Journal of Computational Physics*, 23(3):263 – 275, 1977.

[61] Peter K Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM journal on numerical analysis*, 21(5):995–1011, 1984.

[62] Mikhail Shashkov and Burton Wendroff. The repair paradigm and application to conservation laws. *Journal of Computational Physics*, 198(1):265–277, 2004.

[63] Milan Kucharik, Mikhail Shashkov, and Burton Wendroff. An efficient linearity-and-bound-preserving remapping method. *Journal of Computational Physics*, 188(2):462–471, 2003.

[64] KF Riley, MP Hobson, SJ Bence, and Donald Spector. Mathematical methods for physics and engineering. *American Journal of Physics*, 67:165, 1999.

[65] Bernard Schutz. *A first course in general relativity*. Cambridge university press, 2009.

[66] David C Kay. *Schaum's outline of theory and problems of tensor calculus*. McGraw-Hill, 1988.

[67] Jon Mathews and Robert Lee Walker. *Mathematical methods of physics*, volume 271. WA Benjamin New York, 1970.