THE UNIVERSITY OF READING

DEPARTMENT OF MATHEMATICS

A cell by cell anisotropic adaptive mesh Arbitrary Lagrangian Eulerian method for the numerical solution of the Euler equations.

J. M. Morrell

Thesis submitted for the degree of

Doctor of Philosophy

January 2007

Abstract

A cell by cell anisotropic adaptive mesh technique is combined with a staggered mesh Lagrange plus remap finite element ALE code for the solution of the Euler equations. The quadrilateral finite elements may be subdivided isotropically or anisotropically in a cell by cell manner. An efficient computational method is proposed, which only solves on the finest level of resolution that exists for each part of the domain, referred to as the Dynamic Mesh, with disjoint or hanging nodes being used at resolution transitions. The Lagrangian, equipotential mesh relaxation and advection (solution remapping) steps are generalised so that they may be applied on the unstructured Dynamic Mesh. The method is validated on a series of test problems and it is shown that the anisotropic adaptive mesh method can give a dramatic decrease in run time (run times are up to nine times faster on the test problems used).

Declaration

I confirm that this is my own work and the use of all materials from other sources has been properly and fully acknowledged.

Jennifer M. Morrell

Acknowledgements

I would like to acknowledge the help of my supervisors Dr. Sweby of The University of Reading and Dr. Barlow of AWE. I would also like to acknowledge Dr. Powell of AWE for his help with the adaptive mesh technique and Professor Baines of The University of Reading for his advice. I acknowledge the financial support of AWE.

Contents

List of Figures

Chapter 1

Introduction

This thesis considers the development and validation of an anisotropic cell by cell mesh refinement technique for an Arbitrary Lagrangian Eulerian (ALE) method. These two areas represent different approaches to obtaining high mesh resolution around local features of interest, without requiring a prohibitively large number of elements and nodes. In this chapter we provide an insight into these different areas and outline the approach to combine them.

1.1 Background

The Euler equations describe compressible inviscid hydrodynamics and can be used to model gas flows, shock reflections and collisions, blast waves, wind tunnel experiments, flow around airfoils, turbulence and many other physical phenomena. Analytical solutions only exist in the simplest cases of geometry, boundary conditions and initial conditions; while physical experiments are often costly, difficult to interpret, affected by additional physics, and time consuming to set up. Computational modelling is often cheaper, faster to set up, allows numerous parameter and geometry studies, and can be used to model situations where a real experiment would be impossible or dangerous.
When the Euler equations are discretised on a mesh the solution accuracy will depend on the mesh resolution. Increasing the resolution in areas of the mesh where the solution varies rapidly or over small length scales will increase the solution accuracy. The resolution throughout the domain could be increased, but in realistic models the physical variations or phenomena may be occurring over length scales that are several magni-

tudes smaller than the domain geometry. Increasing the resolution throughout would require a prohibitively large number of elements and nodes, making the calculation time immense. However, many problems are characterised by a few features of interest, such as shocks, shear layers or boundary layers, while the rest of the solution varies slowly or remains constant. The practical and efficient approach is to increase the resolution locally around only the features of interest, rather than throughout the entire domain. Obtaining the resolution required, focusing it in the areas of interest, and keeping the number of nodes and elements to a minimum can be considered to be an optimisation problem. There are two main approaches to this optimisation problem: r-refinement where the number of nodes is fixed but they may move to provide higher resolution in specific areas of the domain, or h-refinement where the number of nodes and elements is increased by the subdivision of elements. In this work we do not consider p-refinement, where the order of the numerical discretisation method is increased.

1.2 Refinement by node movement: r-refinement

There are many types of method that could be classified as r-refinement techniques and Eiseman [37] gives an excellent review of these. Many of the solution adaptive methods are linked to the principle of equidistribution of errors, or to monitor functions related to the physical variables if the error is difficult to determine accurately. Some of the solution adaptive techniques have evolved from mesh generation techniques like Winslow's equipotential relaxation method [86], [51], based on the discretisation of the inverse Laplace equation. Addessio [1] explains how a weight function introduced into Winslow's equations can be used to introduce finer meshing around features of interest, while Lapenta links the weight function to truncation errors in [46] and [47]. A solution adaptive method based on a modified Laplace operator has been used for node movement on unstructured meshes by Marchant and Weatherill [50].
A wide variety of solution adaptive schemes, including Winslow's weight function method, may be derived using variational principles [27], [41], [38]. Moving mesh methods such as those developed by Huang [40], [42] and Baines [15], [16] use monitor functions and solve coupled equations for the mesh velocity and the solution. If the monitor function is density a Lagrangian formulation can be achieved.

Methods based on the Lagrangian form of the Euler equations have meshes that move with the material, aligning with the flow directions and increasing resolution throughout the width of a shock. However, as the material deforms so does the mesh and this may result in mesh tangling. The alternative is an Eulerian approach, where the mesh remains fixed and the material moves through it. The Eulerian formulation is robust, the mesh remains fixed and so mesh tangling does not occur. However, interface positions are not given and must be reconstructed or tracked and it is difficult to include interface physics such as sliding or friction.

1.3 The ALE method and r-refinement

In order to combine the advantages of both the Lagrangian and Eulerian formulations Hirt *et al.* [39] developed the Arbitrary Lagrangian Eulerian (ALE) method where the mesh moves at an arbitrary speed. Although the hydrodynamic equations can be derived and solved for an arbitrary reference frame, see for example [48], [49] and [84], the more usual approach is the Lagrangian plus remap ALE method as used by Hirt [39], Benson [19], [20], Peery and Carroll [51], Addessio [1], and recently by Barlow [13], Souli [62] and Lapenta [46]. The ALE method has been used to model a wide variety of phenomena, such as turbulence, sloshing in water tanks, bending beams, high velocity impacts, material and interface distortion, blast waves, explosions, fracture, friction, soil-structure interactions and blood flow.

In the Lagrangian plus remap ALE method the mesh follows the movement of the material during the Lagrangian step; the mesh quality is then improved by using a mesh relaxation method (such as Winslow's method); finally the solution is remapped to the new mesh. This approach automatically behaves as an r-refinement method focusing the nodes in the areas of interest. However, the number of elements remains fixed and this can lead to under resolution in other parts of the domain. The method developed in this work combines the advantages of ALE with increasing the number of elements using cell by cell refinement.

1.4 Refinement by subdivision: h-refinement

The h-refinement approach attempts to increase local resolution by subdividing elements thereby increasing the total number of elements and nodes in the mesh. As we are interested in the time dependent Euler equations we only consider dynamic refinement, where the refined regions change throughout the calculation time. Refinement sensors based on the variation of one or more of the physical variables are required to select elements for refinement or derefinement. There are two alternative strategies to h-refinement: Adaptive Mesh Refinement, AMR, and cell by cell refinement.

Adaptive Mesh Refinement or structured AMR was developed by Berger and Oliger [25] as a finite difference method, and it has subsequently been extended and generalised in [24], [23], [22], [58], [21], [57]. The elements requiring refinement are clustered into rectangular groups, referred to as grids, and then they are refined together. The method creates a hierarchy of refinement levels, every level (containing the grids at that resolution) is integrated using a standard solution procedure with any missing values being provided by interpolation and/or ghost cells, fine values update coarser underlying values through an injection procedure. Time refinement is included so that fine levels take more time steps than coarse levels. AMR has become a very popular method and has been applied to hyperbolic conservation laws, steady state problems, astrophysics, turbulence, magneto-hydrodynamics and other areas; as considered in [54].

Anderson *et al.* have combined structured AMR with an ALE code in [6], [7], [8], [9]. The Lagrangian step, mesh relaxation and remapping are applied on each refinement level (with interpolation used at level interfaces) and the advancement in time must be interwoven similarly to Berger's approach if time refinement is used.

The alternative strategy is cell by cell refinement, where individual or small groups of elements can be refined. As elements are subdivided the new elements are inserted into the mesh creating a combined mesh that represents the finest resolution existing in each part of the mesh, this mesh is unstructured and dynamic as it changes as the refinement changes. The Dynamic Mesh contains nodes with three nodal neighbours, rather than four, at resolution transitions. These nodes are referred to as disjoint or hanging nodes and require special treatment if any variables are discretised at the nodes. Traditionally cell by cell refinement has used tree based data structures like quadtree or octree, but some methods use connectivity arrays to record the elements and their neighbours. The solution is obtained on the Dynamic Mesh, rather than expensively solving on every level in the hierarchy.

Much of the research into cell by cell refinement has used triangular elements as these already form unstructured meshes [53], [88], [77]. One of the earliest cell by cell refinement methods for quadrilateral elements was created by Khokhlov [45] and extended to incompressible flows by Popinet [55]. Marchant and Weatherill [50] created a cell by cell refinement method for general polygons which is combined with their solution adaptive r-refinement method. Cell by cell refinement often includes an anisotropic refinement capability, where elements may be subdivided in one direction only.

1.5 Anisotropic refinement

Many features of interest, such as shocks, involve large variable changes in one dominant direction, anisotropic subdivision of elements can achieve the required resolution around these features without wasting refinement in the other directions. Apel, Jimack *et al.* [11], [12], [80], [81], [82] have investigated the more theoretical aspects of anisotropic finite elements and anisotropic error indicators; while Walkley, Jimack and Berzins [80], [81], [82], [12] have combined isotropic h-refinement with anisotropic node movement and edge swapping using triangular elements. Again the research into anisotropic refinement is more developed for triangular or tetrahedral elements.

Anisotropic h-refinement for quadrilateral and hexahedral elements has been investigated by Aftosmis [2], this work first introduced the idea of the refinement quadrant map used to decide in which direction an element should refine. Kallinderis and Baron [43] have developed an anisotropic refinement technique that uses a connectivity array approach, rather than a tree based data structure.

In the anisotropic method developed by van der Vegt and van der Ven [71] a complicated refinement sensor based on all flow variables is used. Keats and Lien [44] discuss the performance of various refinement sensors and the calculation of slopes on unstructured meshes. All of the anisotropic research reviewed above uses Eulerian meshes that do not move with the fluid flow, rather than Lagrangian or ALE meshes. The ALE mesh already aligns with the flow directions, therefore anisotropic refinement in the element's local directions should be a very beneficial and efficient way of including anisotropic adaptivity.

1.6 Method summary

In this thesis the staggered grid finite element Lagrange plus remap ALE method is combined with anisotropic cell by cell refinement by subdividing the quadrilateral elements. The new elements are inserted into the mesh to form the Dynamic Mesh which represents the finest resolution in each part of the domain, with disjoint or hanging nodes at resolution transitions. The equipotential relaxation and advection methods are generalised so that they can be applied on the unstructured Dynamic Mesh. The solution is therefore obtained efficiently on this Dynamic Mesh, rather than solving on every level of the hierarchy separately.

1.7 Thesis outline

In Chapter 2 the staggered grid finite element ALE method [13] that forms the starting point for this work is described. The Lagrangian form of the Euler equations is solved using a predictor-corrector time advancement scheme. Winslow's equipotential mesh relaxation method [86], [51] is then applied to obtain new relaxed positions for the nodes. The solution remapping or advection procedure is described for a structured mesh and this uses the volume coordinate approach developed by Benson [19]. Results are then presented for the Sod problem, two-dimensional Riemann problem, square Sod problem, radial Sod problem, and a Taylor-Sedov blast wave.

In Chapter 3 h-refinement is introduced into the Lagrangian method and the new isotropic adaptive mesh technique is developed. The cell by cell isotropic refinement technique subdivides a quadrilateral element in both of the element's local directions so that the refinement aligns with the Lagrangian mesh. An efficient solution procedure is developed that solves on the Dynamic Mesh, representing the finest resolution existing in each part of the domain, rather than Anderson's structured AMR approach [6] that solves separately on every refinement level. Results are given for the Sod problem and the square Sod problem; the calculations are shown to take less time than a uniformly fine calculation.

In Chapter 4 the equipotential relaxation and advection (remapping) methods are generalised so that they can be applied on the unstructured Dynamic Mesh, producing a cell by cell isotropic adaptive mesh ALE method. If the original Winslow method was applied on the Dynamic Mesh this would cause the finer meshing to spread into the coarse regions. To prevent this, nodal length spacings are taken into account and the resulting equations contain weights that depend on the length values. The adaptive ALE method is tested on the radial Sod problem, Taylor-Sedov problem and the two-dimensional Riemann problem verifying that comparable accuracy can be achieved in a seventh of the fine calculation's computational time.

In Chapter 5 a second order interpolation method is developed that takes the variable slope into account when calculating the new fine values during refinement. Although second order refinement does increase the accuracy of the adaptive technique a little, there is not a substantial improvement. The use of buffering elements causes refinement to occur in areas where the variable slopes are almost negligible and hence first order solution transfer is almost as successful as second order.

The anisotropic adaptive mesh ALE method is developed in Chapter 6, allowing the quadrilateral elements to be subdivided in only one of their local directions. The ratio of the element's vertical and horizontal change in density is used to decide whether anisotropic or isotropic refinement occurs. The nodal length weighted equipotential relaxation method is generalised to apply to Dynamic Meshes containing anisotropic elements. The anisotropic method is validated on the test problems from the previous chapters and also on a Mach 3.0 step problem (a particularly challenging problem containing many sensitive features requiring refinement). The spectacular results achieved with the anisotropic methods are discussed, highlighting that the anisotropic method is as accurate as the isotropic method, while requiring significantly fewer elements and reducing the calculation time dramatically (by as much as a ninth on the test problems used).

Finally in Chapter 7 the major achievements are summarised, some conclusions are drawn and areas for further work are discussed.

Chapter 2

The Arbitrary Lagrangian Eulerian method

The Arbitrary Lagrangian Eulerian method can be considered as an r-refinement method, since the total number of elements and nodes of the mesh remain fixed but the mesh can move so that higher mesh resolution is obtained around features of interest without reducing the quality of the mesh.

The mesh may remain fixed, as in an Eulerian formulation, move with the material, as in a Lagrangian formulation, or move in a manner in between these two formulations. The Lagrangian formulation more accurately determines the positions of material interfaces and facilitates the inclusion of slip, slide or friction at interfaces. Boundary and initial conditions are easier to implement and it is easy to follow the history of any material particle. Furthermore, no advection is required making the Lagrangian formulation cheaper without introducing diffusion. While the mesh quality is good the alignment of the mesh with the material can reduce mesh imprinting because the numerical stencil is aligned with the flow. However, deformation of the material can result in mesh tangling that reduces mesh quality, can cause mesh stiffness, reduces the time step of the calculation and may imprint on the solution.

The Eulerian formulation is more robust, since the mesh remains fixed no tangling occurs. It is often easier to implement additional physics on an orthogonal Eulerian mesh and multiple materials can be defined within one element using a Volume of Fluid approach, this is beneficial at highly distorted interfaces. However, interface positions are not given and must be reconstructed or tracked, it is difficult to include interface physics and it is difficult to implement boundary and initial conditions in complicated geometries. On an Eulerian Cartesian mesh higher resolution can only be achieved by uniformly increasing the resolution or by including AMR or cell by cell refinement. A further disadvantage is that diffusion can be caused by the advection step. A detailed discussions of the advantages and disadvantages of the two methods is given in [4]. Benson gives an excellent review of the two methods and the different approaches within them [18].

The ALE method combines the advantages of the two approaches while limiting the disadvantages such as diffusion. For example, Lagrangian interfaces can be used with mesh relaxation within the domain and finer resolution can be achieved locally around features of interest. There are two distinct approaches: the Lagrangian plus remap approach [39], [3], [46], [62], [13] that is used in this work and solving the ALE equations, the hydrodynamic equations in an arbitrary reference frame. Fully solving the coupled ALE equations is a recent approach tried by amongst others Wells [84], Luo [49] and Liu [48] but it is still subject to many difficulties including robustness, good mesh quality around all features and how to include multi-material cells [62].

The Lagrangian plus remap approach consists of a Lagrangian step where the mesh moves with the material giving higher resolution around shocks, a mesh relaxation step to remove mesh tangling and improve the mesh quality, and an advection or remapping step to transfer the solution variables onto the relaxed mesh. This approach can be used to map back to a fixed Eulerian mesh.

The Lagrangian plus remap approach was pioneered by Hirt, Amsden and Cook with a staggered grid quadrilateral finite volume method [39]. A finite difference approach is used for time advancement of velocities and specific total energy. This method included an iteration procedure for obtaining time advanced pressures for flow at all speeds without the restriction of the Courant condition.

Benson provides one of the most comprehensive articles on ALE, discussing the relevant literature and approaches [19]. Nodes are selected for remapping and then the equipotential mesh relaxation equations determine the new positions, the remapping step is discussed using the donor cell method and Van Leer limiting. The volume coordinate approach, used in this work, is developed by Benson. Momentum advection on a staggered mesh is discussed and is also considered in more detail in [20].

Peery and Carroll have developed a multi-material finite element ALE code for unstructured grids [51]. Equipotential mesh relaxation is used only on nodes that have failed an angle or volume test. A directional split approach is used for the advection step.
Souli gives a good recent review of a finite element ALE method in three-dimensions for fluid-structure interactions and surveys recent applications ranging from medical physics to soil science [62]. Three mesh relaxation algorithms are discussed: the equipotential algorithm, a simple average and a volume weighting algorithm. A staggered grid is used but the flux limiters for the remapping or advection step are only discussed in one dimension and no indication is given of how the advection scheme is applied to the staggered grid velocities.
In contrast to the staggered mesh approach, CAVEAT (Addessio *et al.* [1]) has cell centred velocities but retains the close coupling between velocity and pressure by using a Godunov Riemann solver, because of this no artificial viscosity is required. Total energy rather than internal energy is used. However, the approach sacrifices second order accuracy in time. This method also uses equipotential mesh relaxation and the advection step uses a directional split approach. Pember and Anderson compare staggered mesh artificial viscosity methods with cell centred Godunov methods in [10].
Lapenta's recent papers consider modifications to the ALE method where variational grid adaptation based on error indicators is included [46], [47]. This work also introduces a modified equation approach for the remapping or advection step, a donor step is applied, a correction velocity computed and then used in a correction donor step.
Barlow's PhD thesis [13] includes a great deal of detail about the development of the ALE method that is used as the basis for this work. In particular, the artificial viscosity and the advection procedure are explained in detail.

2.1 The ALE mesh

Before the ALE method can be described the type of mesh must be considered. Meshes may be structured or unstructured; this refers to the way of referencing the mesh. An unstructured mesh is referenced by connectivity arrays explicitly defining the element and node connections and neighbours, while a structured mesh is arranged regularly enough for the neighbours to be obvious. The data structures and data management are more complicated in the unstructured case but this provides greater flexibility in meshing. An unstructured mesh is used in this work because it is more flexible for handling complex geometries, allows for initial variations in the resolution of the mesh, and provides the required information for the connectivity of the mesh to change when cell by cell adaptivity is included.

Meshes containing triangular elements are typically unstructured, while those containing quadrilateral elements may be structured or unstructured. Some hybrid meshes contain both triangular and quadrilateral elements. Although triangular elements can be more robust the mesh may move too stiffly. The meshes used in this work consist purely of quadrilateral elements, which are traditionally used in Hydrocodes because they are computationally efficient and the initial mesh is easy to form.

The positions and velocities are stored at the nodes of the elements, while density, pressure, specific internal energy, element mass and element sound speed are discretised in the centre of the element, see Figure 2.1. This is known as a staggered grid, rather than a cell centred grid where all variables are defined in the centre of the cell. Staggered grids make the accurate calculation of the strain rate tensor easier, making material strength simpler to include. Nodal positions and velocities also simplify the tracking of interfaces and the inclusion of mixed cells.

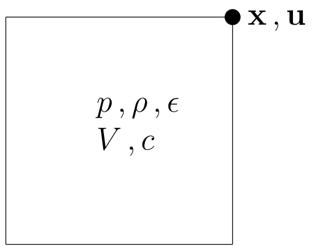

Figure 2.1: Diagram showing the positions of the nodal and element centred variables.

2.2 The Lagrangian step

During the Lagrangian step each element's mass remains unchanged. The Euler equations in a Lagrangian reference frame are

$$\begin{aligned}
\frac{D\rho}{Dt} &= -\rho\nabla\cdot\mathbf{u} && (2.1)\\
\rho\frac{D\mathbf{u}}{Dt} &= -\nabla p && (2.2)\\
\rho\frac{D\epsilon}{Dt} &= -p\,\nabla\cdot\mathbf{u}, && (2.3)
\end{aligned}$$

where

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla \qquad (2.4)$$

is the Lagrangian derivative, ρ is the density, $\mathbf{u}$ is the velocity vector, p is the pressure and ϵ is the specific internal energy. These equations represent respectively conservation of mass, momentum and energy. The system of equations is closed by including the ideal gas equation of state,

$$p = (\gamma - 1)\rho\,\epsilon. \qquad (2.5)$$

For air $\gamma = 1.4$, but other values of γ can be used to run problems with different fluids. For each problem initial conditions are provided for p, ρ, $\mathbf{u}$ and ϵ. Reflective or free surface boundary conditions are used in most problems. The perpendicular velocity at a boundary is defined by the mass transfer through that boundary, and is kept constant throughout the calculation for most of the problems presented here.

The Euler equations are derived for smooth inviscid compressible flow, the physics across shock waves is represented by the Rankine-Hugoniot shock conditions. Including artificial viscosity in the numerical method or solving the Riemann problem are two alternative approaches to this problem. Artificial viscosity increases the stability of the numerical method by spreading the shock over a few elements, replacing a discontinuity with a rapid continuous change. Solving the Riemann problem requires a more accurate sound speed and cell centred variables. It can also be exceedingly expensive. Artificial viscosity is favoured in this code as it is cheaper to apply and can be used with a staggered grid. The form of the artificial viscosity term q will be discussed in a later section.

2.2.1 Time discretisation

A predictor-corrector time discretisation is used for the implicit pressure dependence in the Euler equations. The energy equation and the equation of state are used to obtain

a half step pressure prediction. This predicted pressure is then used in the momentum equation to derive full step nodal velocities. All state variables are then recalculated during a full time step correction.
The time step is limited by the CFL condition, ensuring that signals cannot cross a whole element in one time step. The Courant number C is taken as $\frac{1}{2}$ or $\frac{1}{3}$ in this work. The procedure for one time step is detailed below.

- Calculate the artificial viscosity, q.

- Calculate a stable time step using the CFL condition,

$$\Delta t < Cl/\sqrt{c_s^2 + 2q/\rho}, \tag{2.6}$$

  where c_s is the element sound speed and l is a measure of the minimum distance across the element.

- Move the nodes to their half time step positions using

$$\mathbf{x}^{n+1/2} = \mathbf{x}^n + \frac{\Delta t}{2}\,\mathbf{u}^n. \tag{2.7}$$

- Calculate the half time step element volumes and update the element densities.

- Evaluate the half time step element energies using the time discretisation of (2.3),

$$\epsilon^{n+1/2} = \epsilon^n - \frac{\Delta t}{2\,M^e}\,(p^n + q^n)\nabla\cdot\mathbf{u}^n, \tag{2.8}$$

  where M^e is the element mass and the spatial derivatives are evaluated using finite elements.

- Update the half time step element pressures using the equation of state.

- Use the discretisation of the momentum equation to find the second order accurate velocities at the end of the full time step,

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \frac{\Delta t}{M^{nodal}}\nabla(p^{n+1/2} + q^n), \tag{2.9}$$

  where M^{nodal} is the nodal mass and the spatial derivatives are evaluated using finite elements.

- Calculate the final nodal positions using the average velocity $\bar{\mathbf{u}} = \frac{1}{2}(u^n + u^{n+1})$ over the full time step

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \, \bar{\mathbf{u}}. \tag{2.10}$$

- Calculate the final volumes and densities.
- Evaluate the energies at the end of the time step using

$$\epsilon^{n+1} = \epsilon^n - \frac{\Delta t}{M^e} (p^{n+1/2} + q^n) \nabla \cdot \bar{\mathbf{u}}. \tag{2.11}$$

- Calculate the full time step pressures from the equation of state.

2.2.2 Spatial discretisation

Quadrilateral bilinear isoparametric finite elements are used to calculate the spatial derivatives: the $\nabla(p + q)$ term in the momentum equation and the $\nabla \cdot \mathbf{u}$ term in the energy equation. Finite elements, rather than finite difference or finite volume methods, are employed because they are well suited to unstructured meshes and staggered grids. Many finite difference algorithms cannot be used for unstructured grids and require mesh spacing to stay the same. Finite elements are very flexible for use with complex geometries and can vary in shape and size. The isoparametric formulation is designed to deal with non-orthogonal meshing.

Each element is mapped, using an isoparametric mapping, on to the square with sides $\xi = \pm 1$ and $\eta = \pm 1$. The bilinear finite element functions are

$$\begin{aligned} N_1 &= \frac{1}{4}(1 - \xi)(1 - \eta) \\ N_2 &= \frac{1}{4}(1 + \xi)(1 - \eta) \\ N_3 &= \frac{1}{4}(1 + \xi)(1 + \eta) \\ N_4 &= \frac{1}{4}(1 - \xi)(1 + \eta). \end{aligned} \tag{2.12}$$

For the momentum equation the acceleration in element e is expressed as the summation over the element's nodes of the acceleration multiplied by the finite element function at each node,

$$\dot{\mathbf{u}} = \sum_k \dot{u}_k N_k \, . \tag{2.13}$$

The momentum equation in planar geometry (2.2) is multiplied by the finite element function and integrated to obtain the finite element weak form

$$\rho_e \int_{\Omega} \sum_k \dot{u}_k N_k N_j d\,\Omega = - \int_{\Omega} \frac{\partial (p_e + q_e)}{\partial r_i} N_j d\,\Omega\,, \tag{2.14}$$

where $d\,\Omega = dxdy = detJd\xi d\eta$, J is the Jacobian and r_i is x or y in Cartesian coordinates.

The left hand side can be diagonalised by 'mass lumping'

$$\int_{\Omega} \sum_k \dot{u}_k N_k N_j d\,\Omega = \dot{u}_j \int_{-1}^{1} \int_{-1}^{1} N_j detJd\xi d\eta\,. \tag{2.15}$$

Applying Green's theorem in the plane to the right hand side and noting that p_e, q_e and ρ_e are constant within each element we finally achieve

$$\rho_e \dot{u}_j \int_{-1}^{1} \int_{-1}^{1} N_j detJd\xi d\eta = (p_e + q_e) \int_{-1}^{1} \int_{-1}^{1} \frac{\partial N_j}{\partial r_i} detJd\xi d\eta\,. \tag{2.16}$$

The term on the left is the mass contribution from element e to its local node j. The term on the right is the force in the direction r_i acting on node j due to element e's pressure. From this a nodal acceleration is calculated and discretised to give the updated nodal velocity.

The Euler energy equation (2.3) can be rewritten as

$$\rho_e \dot{\epsilon} = -(p_e + q_e)\, \nabla \cdot \mathbf{u}. \tag{2.17}$$

This can also be expressed in finite element weak form using the linear finite element representations

$$u = \sum_k u_k N_k; \qquad v = \sum_k v_k N_k. \tag{2.18}$$

On integrating (2.17), realising that p_e, q_e and ρ_e are constant within each element and substituting (2.18) into (2.17) we obtain

$$\rho_e \dot{\epsilon} \int_{\Omega} d\,\Omega = -(p_e + q_e) \int_{\Omega} \sum_k \left(u_k \frac{\partial N_k}{\partial x} + v_k \frac{\partial N_k}{\partial y} \right) d\,\Omega\,. \tag{2.19}$$

The left hand side is just the element mass M_e multiplied by $\dot{\epsilon}$. The right hand side can be simplified by interchanging the integral with the summation, changing the summation variable, and taking the constant nodal velocities out of the integral. Finally we obtain

$$\dot{\epsilon} = -\frac{(p_e + q_e)}{M_e} \sum_{j=1}^{4} \left[u_j \int_{-1}^{1} \int_{-1}^{1} \frac{\partial N_j}{\partial x} detJd\xi d\eta + v_j \int_{-1}^{1} \int_{-1}^{1} \frac{\partial N_j}{\partial y} detJd\xi d\eta \right]\,. \tag{2.20}$$

2.2.3 Artificial viscosity

The idea of adding an artificial viscosity term q to the pressure terms in the Euler equations, spreading the shock over 3-4 zones, was suggested by von Neumann in the 1950's [79]. Serious oscillations can be produced as the artificial viscosity is not monotone. This method also ignores the directional nature of the artificial viscosity, which should only act in the direction of compression [85].

Christensen's scalar monotonic artificial viscosity uses a Taylor expansion to take the slopes in the velocity jump into account [18], [29]. This viscosity uses a limiter Φ, similar to that devised by Van Leer [75], [76], to ensure the second order artificial viscosity is monotonic. However, this approach is still one dimensional.

A two dimensional generalisation of Christensen's viscosity, without the complications of a full tensor artificial viscosity [28], is detailed in Barlow [13], based on private communications with Tipton. In this generalisation an element centred value for the artificial viscosity is obtained by combining artificial viscosity values for the four edges of the element. Therefore the four one dimensional edge viscosities are q_b and q_t, associated with compression along the horizontal logical coordinate and q_l and q_r, associated with compression along the vertical logical coordinate. The edge viscosities are shown in Figure 2.2. The edge artificial viscosities are calculated from the change in the velocity along that edge divided by the length of that edge. The edge viscosities are limited using the edge values for the neighbouring elements along that edge. Direction vectors for the horizontal and vertical logical mesh directions must be found, these are known as mesh legs.

The mesh legs have the required direction but their length is proportional to the perpendicular sides,

$$\begin{aligned}
Lhor_y &= -(x_4 + x_3 - x_2 - x_1) \\
Lhor_x &= (y_4 + y_3 - y_2 - y_1) \\
Lver_y &= (x_3 + x_2 - x_4 - x_1) \\
Lver_x &= -(y_3 + y_2 - y_4 - y_1).
\end{aligned} \tag{2.21}$$

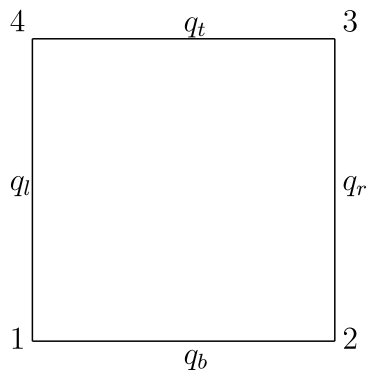

Figure 2.2: Diagram of edge viscosities and node numbering. Each node has position (x,y).

The distance measures for each of the directions then become

$$\begin{aligned} \Delta x_{tb} &= \frac{area}{|\bar{Lhor}|} \\ \Delta x_{lr} &= \frac{area}{|\bar{Lver}|}, \end{aligned} \tag{2.22}$$

where the area is that of the element in question. The velocity gradient parallel to the edge is given by projecting the change in velocity between the two nodes of that edge onto the mesh leg,

$$\begin{aligned} \frac{\Delta u_b}{\Delta x_b} &= \frac{\bar{Lhor} \cdot (\bar{u_2} - \bar{u_1})}{area} \\ \frac{\Delta u_l}{\Delta x_l} &= \frac{\bar{Lver} \cdot (\bar{u_4} - \bar{u_1})}{area} \\ \frac{\Delta u_t}{\Delta x_t} &= \frac{\bar{Lhor} \cdot (\bar{u_3} - \bar{u_4})}{area} \\ \frac{\Delta u_r}{\Delta x_r} &= \frac{\bar{Lver} \cdot (\bar{u_3} - \bar{u_2})}{area}. \end{aligned} \tag{2.23}$$

The variables used in this process are shown in Figure 2.3. The velocity gradients and distance measures must be calculated for all elements before Christensen's monotonic limit is applied to each of the four edges in a one dimensional fashion.

Left and right ratios of the velocity gradient, see Figure 2.4, are then defined as

$$\begin{aligned} R_L &= \frac{(\frac{\Delta u}{\Delta x})_L}{(\frac{\Delta u}{\Delta x})_C} \\ R_R &= \frac{(\frac{\Delta u}{\Delta x})_R}{(\frac{\Delta u}{\Delta x})_C}. \end{aligned} \tag{2.24}$$

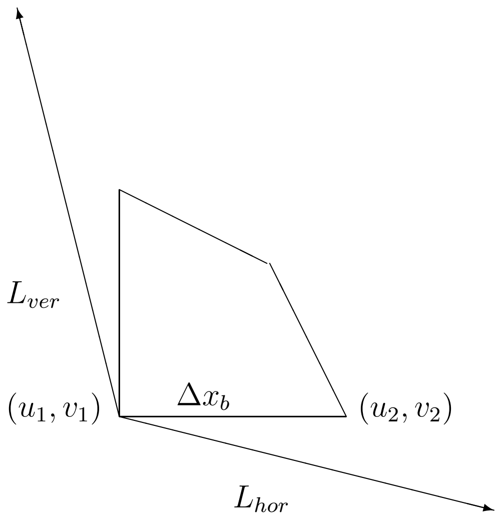

Figure 2.3: Diagram showing the variables for the calculation of the bottom edge velocity gradient.

The velocity slope ratio is set to one on a reflecting boundary.
Any velocity gradient that indicates expansion is then set to zero as artificial viscosity should only act in the direction of compression [85],

$$\text{if} \qquad \frac{\Delta u}{\Delta x} > 0 \qquad \text{then} \qquad \frac{\Delta u}{\Delta x} = 0. \tag{2.25}$$

The limiter Φ is defined as

$$\Phi = max(0, min(\frac{1}{2}(R_L + R_R), 2R_L, 2R_R, 1)), \tag{2.26}$$

and the monotonic q for the central edge is finally given as

$$q_{edge} = c_q \rho \left|\Delta u\right|^2 (1 - \Phi^2) + c_l \rho c_s \left|\Delta u\right| (1 - \Phi), \tag{2.27}$$

where c_s is the element sound speed, c_q and c_l are the quadratic and linear Christensen artificial viscosity coefficients usually taken to be 0.75 and 0.5 respectively. The $|\Delta u|$ term should be evaluated by multiplying the required velocity gradient with the edge distance.
Finally the edge q's in the same logical directions are averaged and the two resulting values are summed to give a value for the artificial viscosity of the element

$$q_{scalar} = \frac{1}{2}(q_b + q_l + q_t + q_r). \tag{2.28}$$

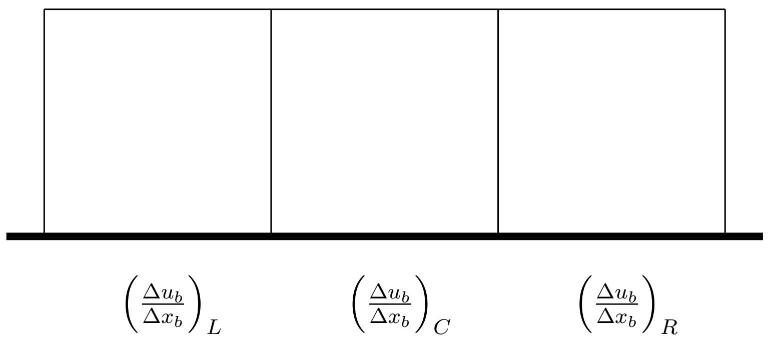
$\left(\frac{\Delta u_b}{\Delta x_b}\right)_L$
$\left(\frac{\Delta u_b}{\Delta x_b}\right)_C$
$\left(\frac{\Delta u_b}{\Delta x_b}\right)_R$

Figure 2.4: Diagram of limiting procedure for bottom edge.

If the element's divergence indicates expansion the artificial viscosity is set to zero in that element.

This completes the description of the Lagrangian step. The mesh having moved with the material will have increased the resolution around shocks. However, in highly two dimensional flows, the mesh may have become very distorted and may imprint on the solution. A tangled Lagrangian mesh and the solution obtained are shown in Figure 2.15 and Figure 2.16 for comparison with the ALE results.

2.3 Equipotential mesh relaxation

In this section we look at techniques that relax the mesh to reduce mesh tangling whilst still retaining the majority of the Lagrangian behaviour. The simplest method of obtaining new relaxed node positions is moving the node to the average position of its neighbours or a volume weighted average as described in [62]. However, these methods do not always produce smooth meshes and the most reliable method remains the Winslow-Crowley equipotential mesh relaxation formula [86], [70], [13].

The appendix of Winslow's original paper [86] describes the construction of a mesh of triangular elements, and discusses the Laplace equations, inverse Laplace equation and its finite difference discretisation.

Work by Thompson, Thames and Mastin [68] and Thomas and Middlecoff [69] generalised Winslow's approach by adding inhomogenous terms to vary the spacing of the coordinate lines within the domain interior or change the angle at which they meet the boundary. A review of elliptic grid generation is provided within [67].

Brackbill and Saltzman developed a variational approach to mesh relaxation focussing on smoothness, orthogonality and volume variation [27]. Winslow's equations result from interchanging the variables in the smoothness integral and differentiating the resulting Euler equations. Brackbill also develops a dynamic mesh adaptation technique using these ideas and includes the idea of a weight function. The variational approach has received much attention and the work of Giannakopoulos [38] and Huang [41] is particularly interesting.

Peery and Carroll give details of Tipton's unpublished work in [51], outlining the equations discussed here and also the variational derivation that is applied using finite element formalisms to an unstructured mesh. This work also includes the weight function. Addessio *et al.* discuss in [1] the use of the weight function with Winslow's equations, which can produce areas of finer resolution, stop polar meshing from collapsing onto the singular point, focus the mesh around a point or line, or be related to the material density.

Lapenta in [46] and [47] develops a variational approach using the equidistribution principle which results in the weighted Winslow equations but links the monitor or weight function to the truncation errors.

This discussion has shown how the subject of mesh relaxation has developed to include solution adaptive r-refinement. However, since the original mesh was Lagrangian the mesh is already aligned with the fluid flow with higher resolution around shocks and so it is doubtful how much more can be gained by introducing a solution dependent relaxation scheme. In fact the most beneficial approach to further solution refinement is the inclusion of h-refinement and this is considered in the next chapter. Therefore, the ALE scheme uses the simplest Winslow-Crowley scheme without the weight function.

In the Winslow-Crowley method potentials $\phi(x, y)$ and $\psi(x, y)$ are assigned to the non-orthogonal continuous mesh lines of the logical mesh coordinates, i.e) $\phi - 1$, ϕ, $\phi + 1$ correspond to mesh lines as shown in Figure 2.5. A smooth mesh will be obtained if the potentials satisfy Laplace's equation in x and y coordinates:

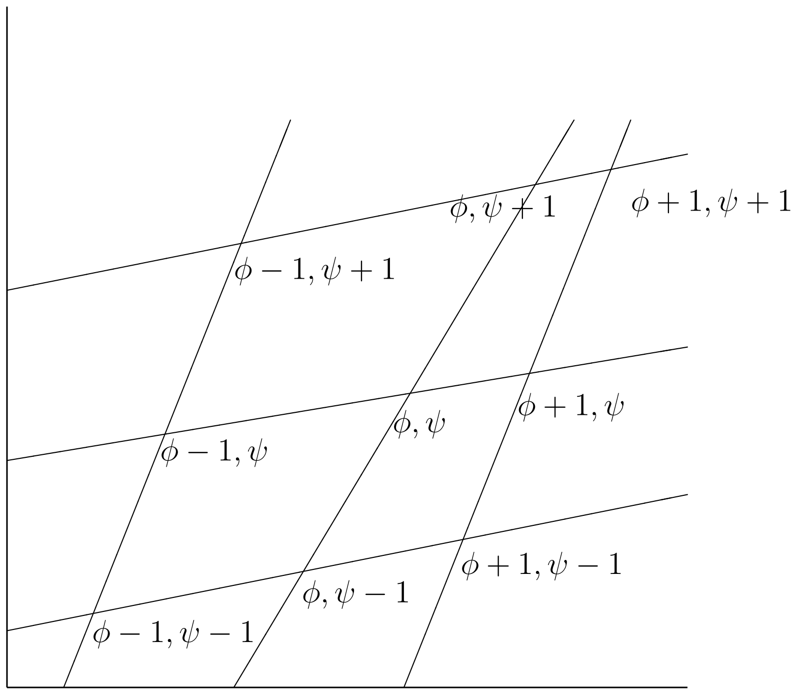

Figure 2.5: Diagram showing the nodal stencil.

$$\begin{aligned} \nabla^2 \phi &= 0 \\ \nabla^2 \psi &= 0. \end{aligned} \tag{2.29}$$

Since we wish to solve for x and y we need differential equations for x and y in terms of ϕ and ψ. The inverse equations are

$$\frac{\alpha}{4}\frac{\partial^2 x}{\partial \phi^2} - \beta \frac{\partial^2 x}{\partial \phi \psi} + \frac{\gamma}{4}\frac{\partial^2 x}{\partial \psi^2} = 0 \tag{2.30}$$

$$\frac{\alpha}{4}\frac{\partial^2 y}{\partial \phi^2} - \beta \frac{\partial^2 y}{\partial \phi \psi} + \frac{\gamma}{4}\frac{\partial^2 y}{\partial \psi^2} = 0 \tag{2.31}$$

where

$$\alpha = 4\left((\frac{\partial x}{\partial \psi})^2 + (\frac{\partial y}{\partial \psi})^2 \right) \tag{2.32}$$

$$\beta = 2\left(\frac{\partial x}{\partial \phi}\frac{\partial x}{\partial \psi} + \frac{\partial y}{\partial \phi}\frac{\partial y}{\partial \psi} \right) \tag{2.33}$$

$$\gamma = 4\left((\frac{\partial x}{\partial \phi})^2 + (\frac{\partial y}{\partial \phi})^2 \right). \tag{2.34}$$

and the Jacobian is $J = \frac{\partial x}{\partial \psi}\frac{\partial y}{\partial \phi} - \frac{\partial x}{\partial \phi}\frac{\partial y}{\partial \psi} \neq 0$.

The inverse equations are discretised using a nine point stencil and central differences. This moves the nodes toward equal spacing and provides second order accuracy. The central difference formulae for the x second derivatives are

$$\frac{\partial^2 x}{\partial \phi^2} = x_{\phi-1,\psi} - 2x_{\phi,\psi} + x_{\phi+1,\psi} \tag{2.35}$$

$$\frac{\partial^2 x}{\partial \psi^2} = x_{\phi,\psi-1} - 2x_{\phi,\psi} + x_{\phi,\psi+1} \tag{2.36}$$

$$\frac{\partial^2 x}{\partial \phi \psi} = \frac{1}{4}(-x_{\phi+1,\psi-1} + x_{\phi+1,\psi+1} - x_{\phi-1,\psi+1} + x_{\phi-1,\psi-1}). \tag{2.37}$$

Similar expressions are used for the y derivatives. In order to linearise the equations, the coefficients α, β and γ are considered to vary slowly and set to be constant throughout each iteration step. They are evaluated at the node using central differences inserted into equations (2.32), (2.33) and (2.34)

$$\alpha_{\phi,\psi} = (x_{\phi,\psi+1} - x_{\phi,\psi-1})^2 + (y_{\phi,\psi+1} - y_{\phi,\psi-1})^2 \tag{2.38}$$

$$\begin{aligned} \beta_{\phi,\psi} &= \frac{1}{2}((x_{\phi+1,\psi} - x_{\phi-1,\psi})(x_{\phi,\psi+1} - x_{\phi,\psi-1}) \\ &+ (y_{\phi+1,\psi} - y_{\phi-1,\psi})(y_{\phi,\psi+1} - y_{\phi,\psi-1})) \end{aligned} \tag{2.39}$$

$$\gamma_{\phi,\psi} = (x_{\phi+1,\psi} - x_{\phi-1,\psi})^2 + (y_{\phi+1,\psi} - y_{\phi-1,\psi})^2. \tag{2.40}$$

Substituting these central differences into the inverse equations and rearranging to give $x_{\phi,\psi}$ and $y_{\phi,\psi}$ gives the following formulae

$$\begin{aligned} x_{\phi,\psi} &= \frac{1}{2(\alpha_{\phi,\psi} + \gamma_{\phi,\psi})}[\alpha_{\phi,\psi}(x_{\phi+1,\psi} + x_{\phi-1,\psi}) \\ &+ \gamma_{\phi,\psi}(x_{\phi,\psi+1} + x_{\phi,\psi-1}) \\ &+ \beta_{\phi,\psi}(x_{\phi+1,\psi-1} - x_{\phi+1,\psi+1} + x_{\phi-1,\psi+1} - x_{\phi-1,\psi-1})] \end{aligned} \tag{2.41}$$

$$\begin{aligned} y_{\phi,\psi} &= \frac{1}{2(\alpha_{\phi,\psi} + \gamma_{\phi,\psi})}[\alpha_{\phi,\psi}(y_{\phi+1,\psi} + y_{\phi-1,\psi}) \\ &+ \gamma_{\phi,\psi}(y_{\phi,\psi+1} + y_{\phi,\psi-1}) \\ &+ \beta_{\phi,\psi}(y_{\phi+1,\psi-1} - y_{\phi+1,\psi+1} + y_{\phi-1,\psi+1} - y_{\phi-1,\psi-1})]. \end{aligned} \tag{2.42}$$

These formulae can be iteratively applied to give relaxed positions for the nodes [13] [70]. The greater the number of iterations the greater the relaxation of the mesh. One iteration of the Gauss-Jacobi method per time step is used in this work. The Gauss-Seidel method, although providing faster convergence, was found to relax the mesh so much that all of the Lagrangian nature was destroyed. Once new relaxed positions for the nodes have been determined the solution needs to be transferred to the new mesh.

2.4 The advection step

The solution is transferred to the new mesh during the remapping step. Benson [18] gives an excellent review of the many methods that may be used. The two main types of method that are conservative are integral rezoning and advection (continuous rezoning). Integral rezoning integrates the values from the old mesh within the new element and then divides by the new volume. Methods based on the divergence theorem have been developed by Dukowitz [36], [34], Ramshaw [60], [59] and Zalesak [90]. However, these require searching through the old and new mesh lines for intersections, the first order methods are too diffusive to apply every time step and the second order methods require the expensive calculation of a two dimensional gradient. Dukowitz and Baumgardner [35] have recently developed the incremental remapping approach, which attempts to close the gap between integral rezoning and advection, although the approach requires the complicated calculation of all the geometrical cases of overlap volume. Therefore, in this work we apply the alternative method based on advection.
The advection method can be used when small mesh changes are involved as in this case. Van Leer developed second and third order schemes based on the linear advection equation [72], [73], [74], [75], [76]. Benson [19] and Barlow [13] extended the method to two dimensional non-orthogonal grids by using the isoparametric directions of the element and volume coordinates to calculate slopes and distances. The solution value is obtained from the conserved quantity perturbed by the fluxes through the element edges that result from the edge's change in position.
The new value in an element α is

$$\varphi_\alpha^+ = \frac{1}{V_\alpha^+}\left(\varphi_\alpha^- V_\alpha^- + \Delta\varphi_{b3} + \Delta\varphi_{t1} + \Delta\varphi_{l2} + \Delta\varphi_{r4} - \sum_{i=1}^{4}\Delta\varphi_{\alpha i}\right), \tag{2.43}$$

where V_α^- is the old element volume, V_α^+ is the new element volume, b, t, l, r represent the bottom, top, left and right neighbouring elements of α and the $\Delta\varphi_{\alpha i}$ are the element's fluxes. The influxes are set to 0 and the outfluxes are set as positive to reduce the amount of calculation, ensure conservation and provide upwinding. This explains why both the neighbours' fluxes and the element's fluxes appear.

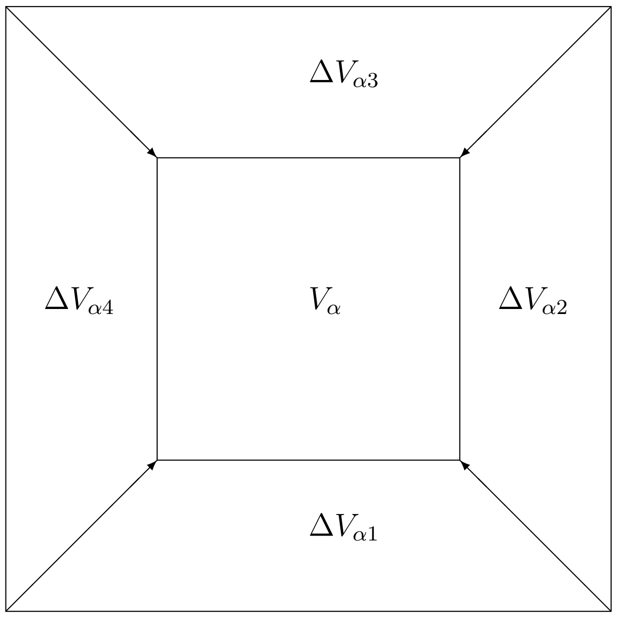

Figure 2.6: Diagram of the element overlap volumes, which are the volumes swept out as each edge changes its position (uniform motion is shown for clarity).

The flux $\Delta\varphi_{\alpha i}$ through each of the four sides is calculated using

$$\begin{aligned}
\Delta\varphi_{\alpha 1} &= \Delta V_{\alpha 1}(\varphi_\alpha^- - \varphi'_{\alpha,\xi}(V_{\alpha 1} + V_{\alpha 2} + \frac{1}{2}\Delta V_{\alpha 1})) \\
\Delta\varphi_{\alpha 2} &= \Delta V_{\alpha 2}(\varphi_\alpha^- + \varphi'_{\alpha,\eta}(V_{\alpha 2} + V_{\alpha 3} + \frac{1}{2}\Delta V_{\alpha 2})) \\
\Delta\varphi_{\alpha 3} &= \Delta V_{\alpha 3}(\varphi_\alpha^- + \varphi'_{\alpha,\xi}(V_{\alpha 3} + V_{\alpha 4} + \frac{1}{2}\Delta V_{\alpha 3})) \\
\Delta\varphi_{\alpha 4} &= \Delta V_{\alpha 4}(\varphi_\alpha^- - \varphi'_{\alpha,\eta}(V_{\alpha 4} + V_{\alpha 1} + \frac{1}{2}\Delta V_{\alpha 4})), \qquad (2.44)
\end{aligned}$$

where the $\Delta V_{\alpha i}$ are overlap volumes as shown in Figure 2.6 (for clarity the diagrams show orthogonal elements, in reality the elements will be distorted quadrilaterals), $V_{\alpha 1}$, $V_{\alpha 2}$, $V_{\alpha 3}$, $V_{\alpha 4}$ are partial volumes as illustrated in Figure 2.7, approximated from volume integrals of each shape function. The average value is found by interpolating the cell centred value φ_α^- to the middle of the overlap volume using the monotonic slopes $\varphi'_{\alpha,\xi}$ and $\varphi'_{\alpha,\eta}$ in the η and ξ directions respectively, derived in volume coordinates.
Benson [19] calculates a parabolic slope by fitting a parabola through elements $\alpha - 1$, α, $\alpha + 1$ with centres $x_{\alpha-1}$, x_α, $x_{\alpha+1}$

$$\frac{\partial\varphi_\alpha}{\partial x} = \frac{(\varphi_{\alpha+1} - \varphi_\alpha)\Delta x_\alpha^2 + (\varphi_\alpha - \varphi_{\alpha-1})\Delta x_{\alpha+1}^2}{\Delta x_\alpha \Delta x_{\alpha+1}(\Delta x_\alpha + \Delta x_{\alpha+1})}, \qquad (2.45)$$

where the volume between $x_{\alpha-1}$ and x_α is Δx_α and between x_α and $x_{\alpha+1}$ is $\Delta x_{\alpha+1}$.

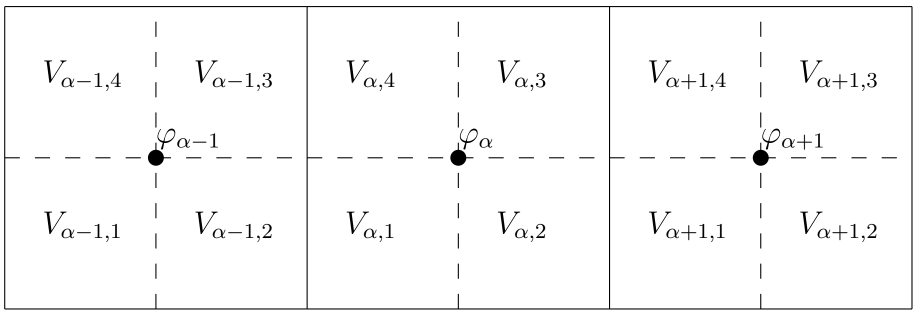

Figure 2.7: Diagram of the three elements used to calculate the slopes and their partial volumes.

These volumes are given by

$$\Delta x_{\alpha} = V^-_{\alpha-1\,2} + V^-_{\alpha-1\,3} + V^-_{\alpha\,1} + V^-_{\alpha\,4}$$

$$\Delta x_{\alpha+1} = V^-_{\alpha\,2} + V^-_{\alpha\,3} + V^-_{\alpha+1\,1} + V^-_{\alpha+1\,4}, \tag{2.46}$$

where the minus denotes old mesh quantities.

A second order limited slope is obtained by requiring that the interpolated value at the edge of the element is not above or below that of the neighbour. The limited slope is

$$\varphi'_{\alpha,\eta} = \frac{1}{2}(sgn(\Delta\varphi_{\alpha}) + sgn(\Delta\varphi_{\alpha+1}))min\left(\left|\frac{\partial\varphi_{\alpha}}{\partial x}\right|, |\Delta\varphi_{\alpha}|, |\Delta\varphi_{\alpha+1}|\right), \tag{2.47}$$

where the left and right limiting slopes are

$$\begin{aligned} \Delta\varphi_{\alpha} &= \frac{\varphi_{\alpha} - \varphi_{\alpha-1}}{V^-_{\alpha\,1} + V^-_{\alpha\,4}} \\ \Delta\varphi_{\alpha+1} &= \frac{\varphi_{\alpha+1} - \varphi_{\alpha}}{V^-_{\alpha\,2} + V^-_{\alpha\,3}}. \end{aligned} \tag{2.48}$$

Density is calculated using volume weighting as outlined above. Both specific internal energy and velocity are advected using mass weighting. For mass weighting the mass fluxes are used instead of the overlap volumes and the slopes are derived with respect to change in mass not volume.

2.4.1 Momentum advection

Advecting nodal velocities is complicated because there are no equivalent quantities to the element volumes and overlap volumes making it difficult to calculate the fluxes. Cell centred advection schemes [20] use the nodal variable to calculate new cell centred variables, these are then advected and then the cell centred values are converted back.

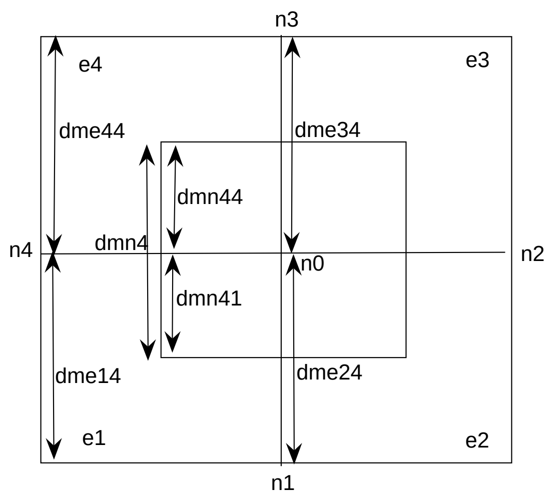

Figure 2.8: Variables for nodal mass flux calculation.

Although this approach naturally lends itself to unstructured grids, it is too expensive as it requires four cell centred variables to be advected for each velocity component to avoid adding dispersion errors. The alternative approach employed here is to create a dual mesh and use the element mass fluxes to define nodal mass fluxes. The original advection scheme is then applied on the dual mesh and therefore monotonicity is ensured.

A dual mesh is made from joining the element centroids and points half way along the element sides. The nodal fluxes are calculated from the average of the surrounding element mass fluxes [13]. From Figure 2.8 the nodal flux dmn_4 between the node and its neighbour n_4 is made up of the fluxes across the two polygon sides, $dmn_4 = dmn_{44} + dmn_{41}$. If dme_{11} to dme_{14} are the four element mass fluxes of e_1 and so on for the other elements then

$$\begin{aligned} dmn_{44} &= \frac{1}{2}\frac{(dme_{34} + dme_{44})}{2} \\ dmn_{41} &= \frac{1}{2}\frac{(dme_{24} + dme_{14})}{2} \end{aligned} \tag{2.49}$$

$$dmn_4 = dmn_{44} + dmn_{41} = \frac{1}{4}(dme_{34} + dme_{44} + dme_{24} + dme_{14}). \tag{2.50}$$

Nodal masses are calculated for each node as the sum of each surrounding element's quarter mass

$$M_n = \sum_{i=1}^{4} \frac{1}{4} M_{e_i}. \tag{2.51}$$

The four nodal weights centred on the node are defined as the element quarter masses $W_{ni} = \frac{1}{4}M_{e_i}$. These replace the partial volumes in the slope and momentum flux calculations. The overlap volumes are replaced by the nodal mass fluxes. The nodal momentum fluxes can now be evaluated similarly to equation (2.44). Finally the velocities are given from (2.43), with V_α^- replaced by M_n. The post advection nodal mass M_n^+ is calculated by allowing for the nodal mass influxes and outfluxes.

2.4.2 Split advection

In Barlow [13] all four edge fluxes are used at once, so that flux is only passed to the four nearest neighbours and not the corner elements. It will take twice the time for material to move diagonally as it does for flow aligned with the mesh. This can lead to a corner coupling error that can destroy symmetry. This could be detrimental to the performance of an adaptive mesh technique. The corner transport upwind method developed by Colella [31] does take corner transport into account, however it would be useful to solve this problem by adjusting the present method rather than implementing another.

The alternative to the unsplit approach is an operator or directional split method. Strang proved that applying a half volume x direction sweep, followed by a full volume y sweep and then another half volume x sweep is second order accurate [63]. A more efficient and stable version of this is an x sweep followed by a y sweep at one time step and then a y sweep followed by an x sweep on the next time step. The directional splitting is in terms of the spatial directions and so is unsuitable for non-orthogonal meshing.

In this work the directional split is in terms of the isoparametric local coordinates based on Peery and Carroll's work [51]. All four fluxes are calculated initially, but they are selected for use as edge pairs, "left" and "right" forming one sweep and "up" and "down" the other sweep. The element volume and advected variables are updated with the first flux pair, these new variables are then used in the second sweep and updated using the last two fluxes. Both sweeps are performed each time step but their order is alternated each time step. This scheme was a substantial improvement upon the old method, shown in Figure 2.9, the new results show near perfect symmetry with only a little more diffusion, see Figure 2.10. The split scheme is therefore used throughout this thesis.

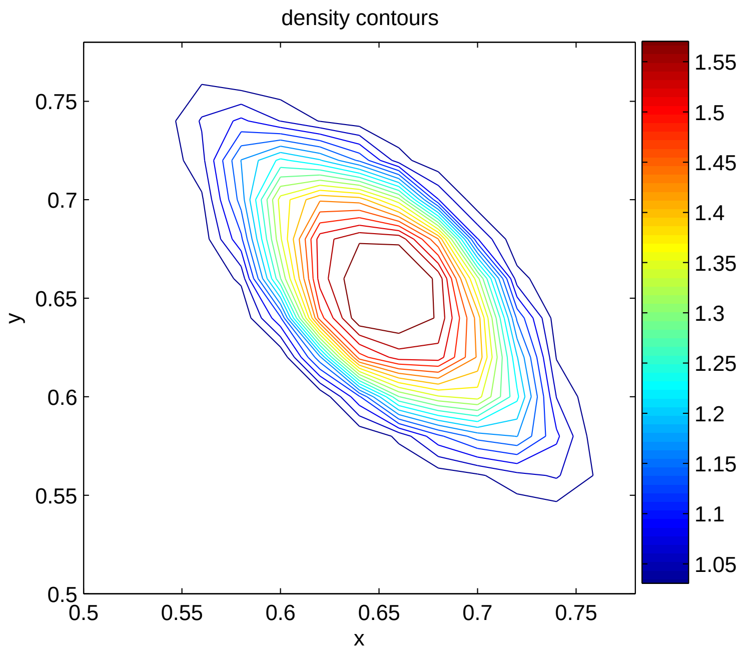

Figure 2.9: Advection of a square bubble using unsplit scheme. Density contours t=0.15.

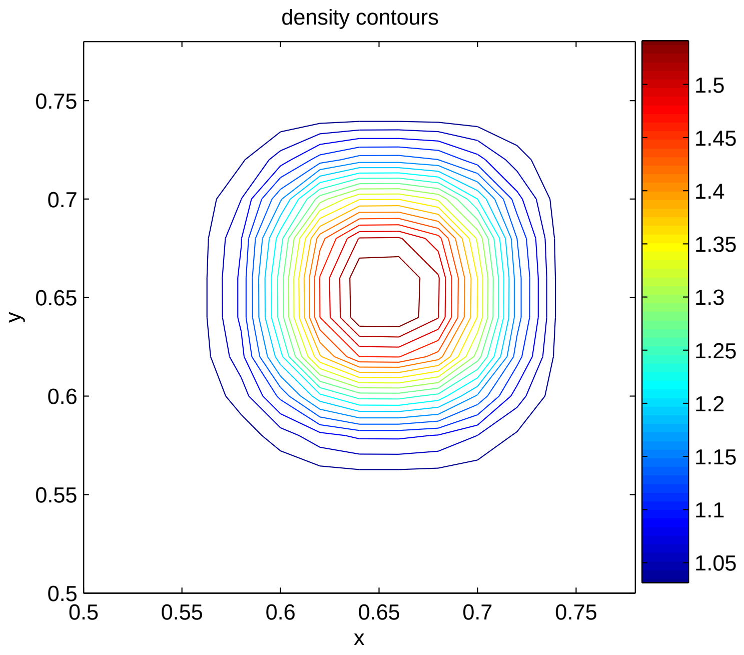

Figure 2.10: Advection of a square bubble using alternating two sweep split scheme. Density contours t=0.15.

2.5 Numerical results for the ALE method

2.5.1 Sod's shock tube problem

This test problem, illustrated in Figure 2.11, consists of a one-dimensional shock tube with initial conditions $p_l = 1.0$, $\rho_l = 1.0$, $\mathbf{u}_l = 0.0$, $p_r = 0.1$, $\rho_r = 0.125$ and $\mathbf{u}_r = 0.0$ [61]. The solution consists of 4 constant states divided by a rarefaction fan, a contact discontinuity and a shock. The density changes across the contact discontinuity but the velocity and pressure remain constant.

The numerical results with 100 elements in the x direction are shown in Figure 2.12 and Figure 2.13 and are in good agreement, considering the use of artificial viscosity, with the exact results from Toro's iterative Riemann solver [66]. The Christensen artificial viscosity, with $c_l = 0.5$ and $c_q = 0.75$, ensures that the results are less oscillatory than for a simple bulk artificial viscosity. This problem is only one dimensional so the benefits of the two-dimensional generalisation of Christensen's artificial viscosity are not fully evident.

There is a large overshoot in the internal energy profile at the contact discontinuity due to too much artificial viscosity being given to the cells located at the discontinuity during the first few time steps until the shock is spread over 4-5 cells. It should be noted that these results are at a later time than those shown in [13].

2.5.2 Two-dimensional Riemann problem

In this example different constant initial conditions are given in each quarter of a square domain. The quarters 1,2,3 and 4 shown in Figure 2.14 are defined respectively as $x > 0.5$ and $y > 0.5$, $x < 0.5$ and $y > 0.5$, $x < 0.5$ and $y < 0.5$ and finally $x > 0.5$ and $y < 0.5$. The following configuration involving 4 shocks given in Kurganov and Tadmor [64] is tested,

$$\begin{array}{llll}
p_2 = 0.3500 & \rho_2 = 0.5065 & p_1 = 1.1000 & \rho_1 = 1.1000 \\
u_2 = 0.8939 & v_2 = 0.0000 & u_1 = 0.0000 & v_1 = 0.0000 \\
\\
p_3 = 1.1000 & \rho_3 = 1.1000 & p_4 = 0.3500 & \rho_4 = 0.5065 \\
u_3 = 0.8939 & v_3 = 0.8939 & u_4 = 0.0000 & v_4 = 0.8939\,.
\end{array}$$

The solution should evolve as 4 shocks, which interact forming a region of high density in the shape of an oval. This calculation experiences a lot of mesh distortion and tangling along the oval axis if the mesh moves in a Lagrangian manner, sometimes leading to calculation failure. Results can be obtained for t=0.2 by reducing the Christensen coefficients to $c_l = 0.3$ and $c_q = 0.65$ these are shown in Figure 2.15 and Figure 2.16. This test problem highlights the benefits of applying the mesh relaxation and advection step, with 1 relaxation iteration per time step the mesh tangling has been alleviated and the mesh quality is significantly improved, see Figure 2.17. The results for the ALE calculation are better as the mesh no longer imprints on the solution, see Figure 2.18, and the shocks and high density oval are excellently captured.

There are density contours appearing in the constant states, where tiny rarefactions and contacts have appeared. This is due to the staggered mesh Lagrangian step not retaining the purely shock solution exactly. A cell centred Roe Riemann solver will retain the purely shock form of the solution, with no contacts or rarefactions. However, the staggered mesh causes a velocity jump not at the same position as the density and pressure jumps, the Lagrangian step then gives a solution that has small contacts and rarefactions together with the expected shocks.

2.5.3 Square Sod problem

In a (0,1)×(0,1) domain a (0.3,0.7)×(0.3,0.7) square with ρ=1, p=1 and zero velocity was initialised. Outside the square the initial conditions were ρ=0.125, p=0.1 and zero velocity, as illustrated in Figure 2.19. The problem was run up to t=0.1 with a 100 × 100 mesh, with Christensen artificial viscosity coefficients $c_l = 0.3$ and $c_q = 0.65$, and evolved as a one dimensional Sod shock tube along the middle of each side with radial Sod behaviour at the corners.

The square Sod test problem was run in a purely Lagrangian manner as the mesh remained smooth and untangled, see Figure 2.20. The results are shown as a surface plot, see Figure 2.21, and have captured the rarefactions, contacts and shocks well, both where the shocks are straight and where they curve like in a radial Sod problem.

2.5.4 Radial Sod problem

A quarter radial Sod problem [66] was run using the ALE method with 1 relaxation iteration per time step to t=0.25. Within a 0.4 radius of (0,0) the initial conditions were ρ=1, p=1 and zero velocity, elsewhere the initial conditions were ρ=0.125, p=0.1 and zero velocity. A 100 $\times$ 100 mesh was used in the calculation with Christensen artificial viscosity coefficients $c_l = 0.5$ and $c_q = 0.75$.

The ALE mesh, see Figure 2.22, is untangled and has increased resolution around the shocks as desired. The density contours shown in Figure 2.23 are not quite symmetrical due to the initial conditions being implemented on a quadrilateral mesh and therefore forming a stepped circle. However, the results when plotted against radius, see Figure 2.24, compare exceedingly well with a very fine one dimensional approximate solution, with 5000 points, obtained using a Roe approximate Riemann solver, with the radial terms included as a source term as explained in [84].

2.5.5 Taylor-Sedov blast wave

After an intense explosion at a point a blast wave propagates out. This problem has a self-similar solution that can be obtained using dimensional analysis [17], [87]. The initial conditions for this problem were no flow, zero pressure, the density was 1 throughout the domain and gamma was $\frac{5}{3}$. A quarter of the domain was represented by (0,1) $\times$ (0,1). All cell energies were 0 except for the cell closest to the origin that was given a total energy of 8. The total energy had to be converted into an internal energy by dividing it by four times the cell area, remembering that we were only representing a quarter of the domain. The end time of the calculation was t=0.1.

The self similar solution gives a density just before the shock front of

$$\rho = \frac{\gamma + 1}{\gamma - 1}\rho_0 = 4. \tag{2.52}$$

The shock radius, for the planar problem, is

$$R = k\left(\frac{Et^2}{\rho_0}\right)^{\frac{1}{4}} \approx 0.61, \tag{2.53}$$

where k is a constant dependent on gamma, $k = 1.15$ for $\gamma = \frac{5}{3}$.

The ALE calculation with a mesh of 64 $\times$ 64 and $c_l = 0.5$ and $c_q = 0.75$ gives an untangled mesh as required, see Figure 2.25. The calculation reaches a lower density, 3.73, than the self similar solution peak and the shock radius ranges from 0.606 to 0.615. These results are excellent when it is considered that this problem has not been run on a radial mesh and that the initial condition was a square of energy rather than a point source or circle of energy. Furthermore the peak height is unlikely to be reached because of diffusion caused by the artificial viscosity. The density contours, shown in Figure 2.26, are not perfectly radially symmetric again because of the implementation of the energy square on the quadrilateral mesh.

2.5.6 Adaptivity and the ALE method

The test problems presented here contain local features of interest, such as the Riemann problem shocks and the density oval circumference; the Sedov peak; or the contacts, rarefactions and shocks in the Sod, radial Sod and square Sod test problems. However, the rest of the problem domain often consists of uniform, constant or slowly varying states. In the ALE method the number of elements remains fixed therefore achieving high resolution around the features of interest may leave other areas of the domain under resolved. To resolve the features of interest and prevent under resolution may require the wasteful use of higher resolution throughout the domain. The Lagrangian motion of the mesh does help to keep resolution around features of interest but mesh relaxation is required in some problems to prevent tangling. The next section attempts to resolve the features of interest, by the inclusion of local refinement through subdivision, without having to waste elements in the parts of the domain where the solution varies slowly. The existing ALE method is generalised to include a cell by cell adaptive mesh technique.

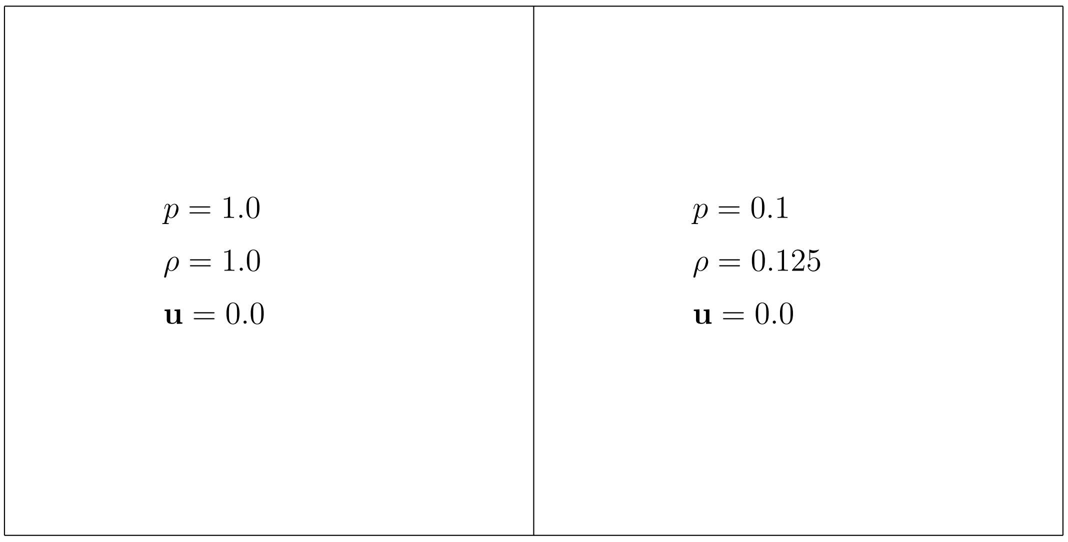

Figure 2.11: Diagram of Sod's shock tube.

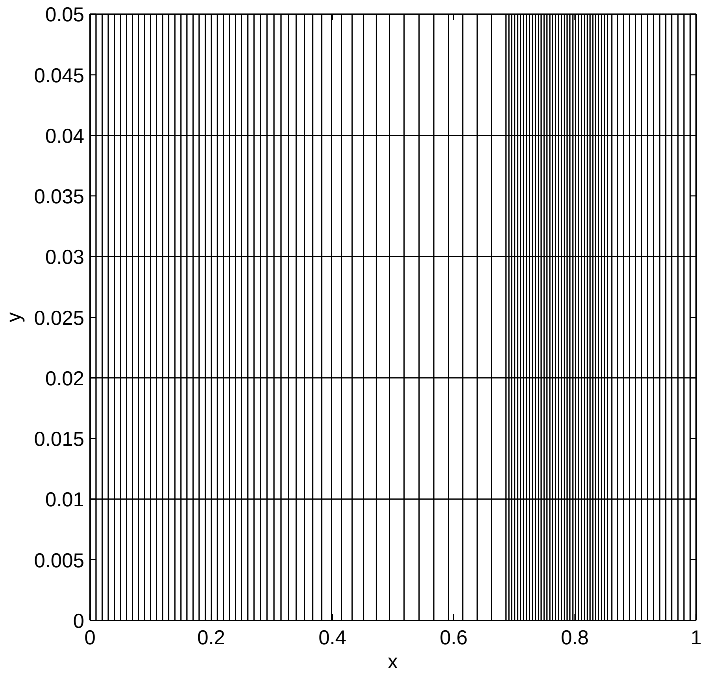

Figure 2.12: Sod's shock tube problem Lagrangian mesh at t=0.2.

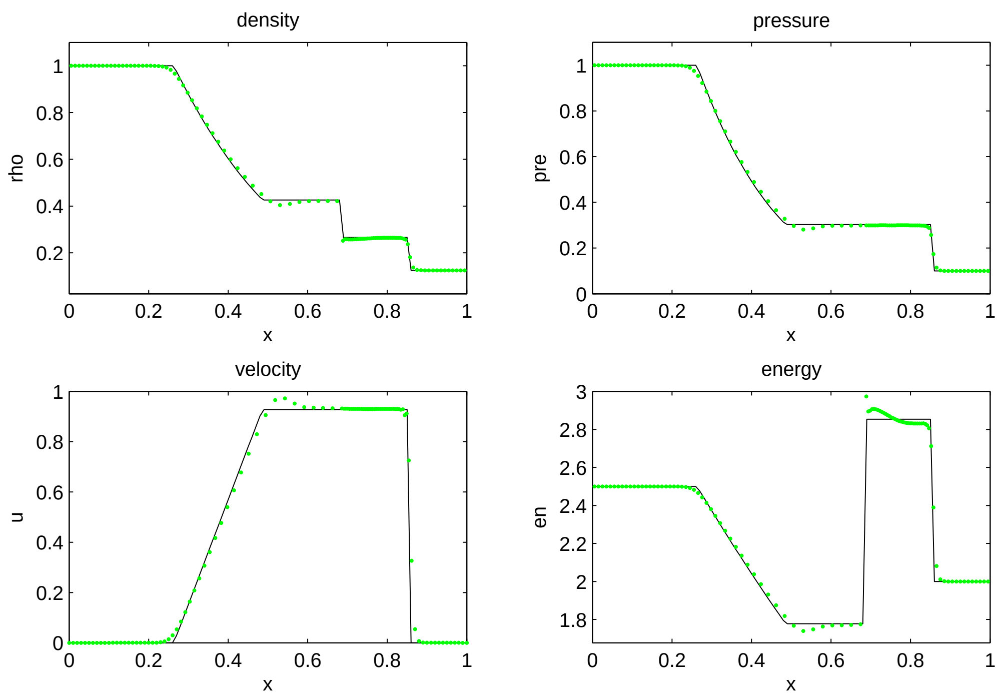

Figure 2.13: Sod's shock tube problem density, pressure, velocity and specific internal energy results at t=0.2, Lagrangian calculation in green compared to 1D Riemann solver solution in black.

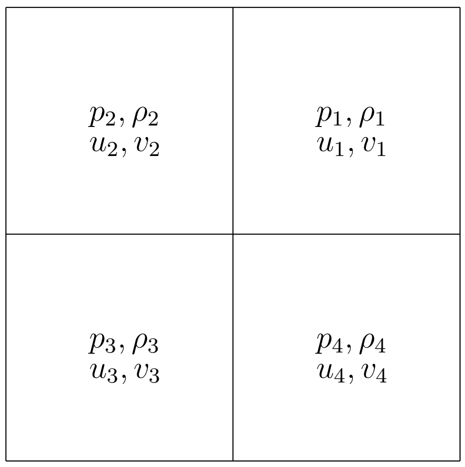

Figure 2.14: Diagram of two-dimensional Riemann problem.

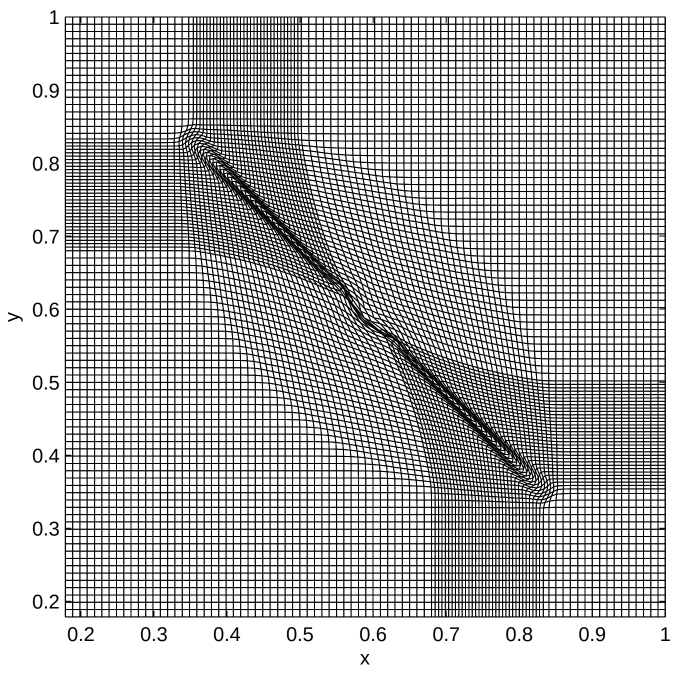

Figure 2.15: Two-dimensional Riemann problem Lagrangian mesh at t=0.2 with Christensen artificial viscosity coefficients $c_l = 0.3$ and $c_q = 0.65$.

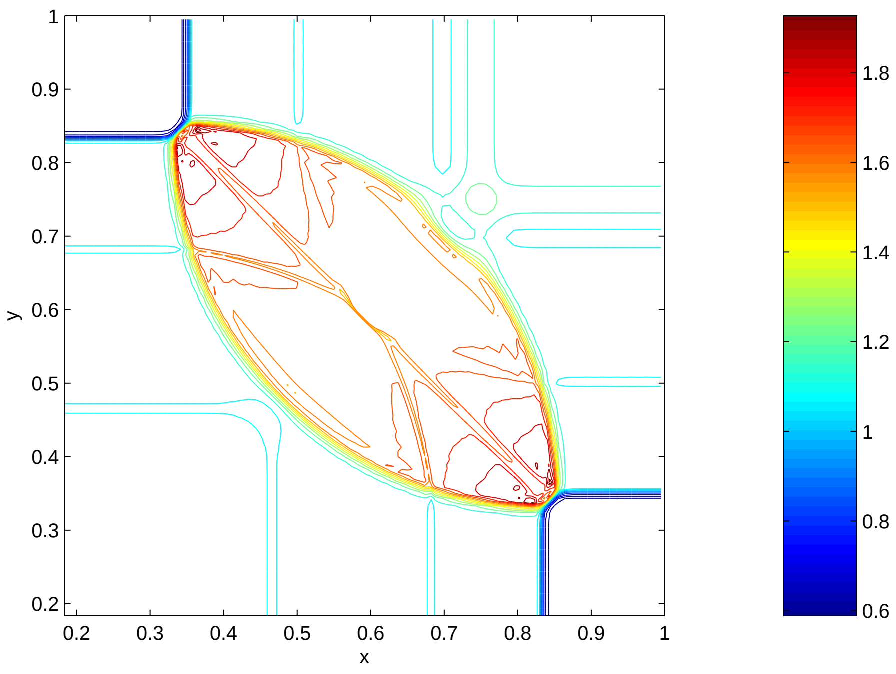

Figure 2.16: Two-dimensional Riemann problem Lagrangian calculation density contours at t=0.2 with Christensen artificial viscosity coefficients $c_l = 0.3$ and $c_q = 0.65$.

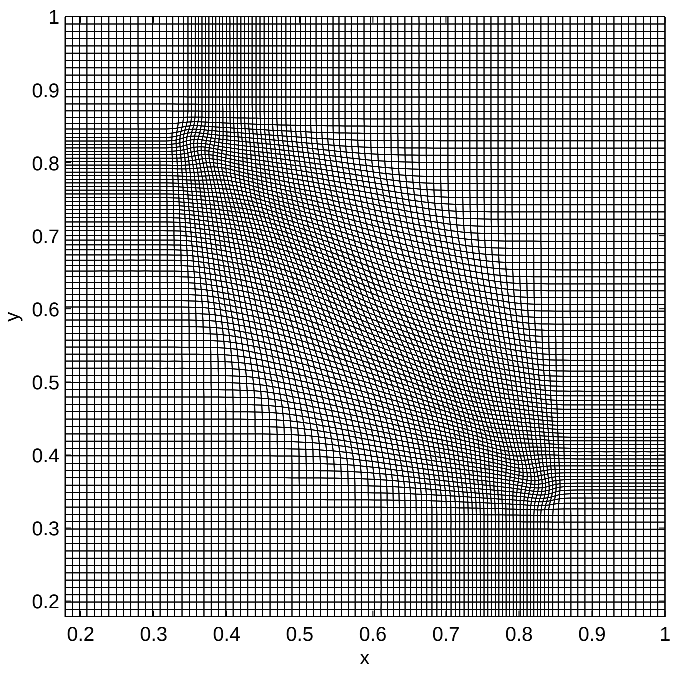

Figure 2.17: Two-dimensional Riemann problem ALE mesh at t=0.2.

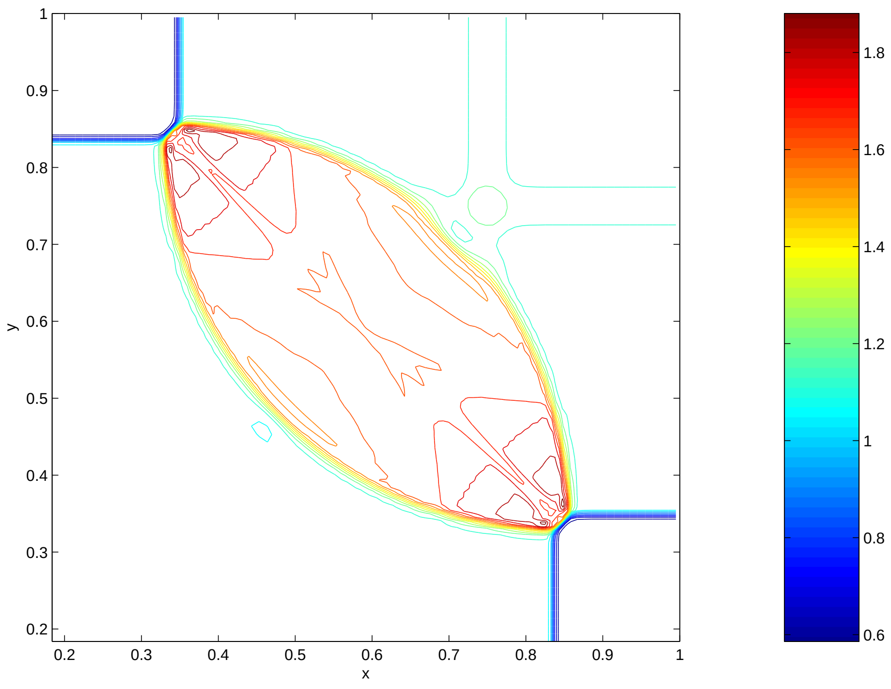

Figure 2.18: Two-dimensional Riemann problem ALE calculation density contours at t=0.2.

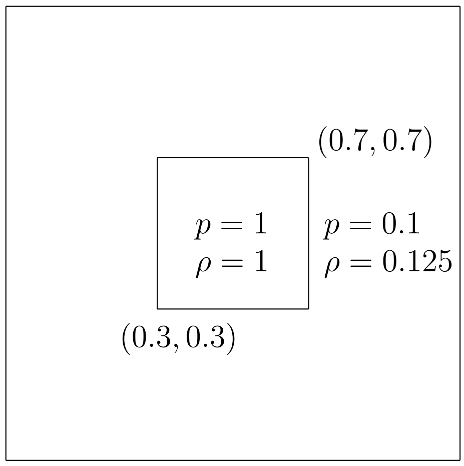

Figure 2.19: Diagram showing the initial conditions for the square Sod problem.

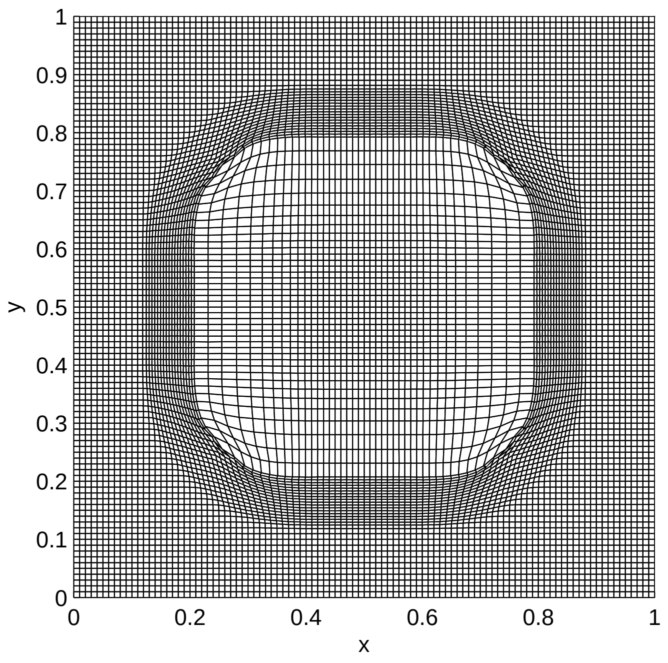

Figure 2.20: Square Sod problem Lagrangian mesh at t=0.1.

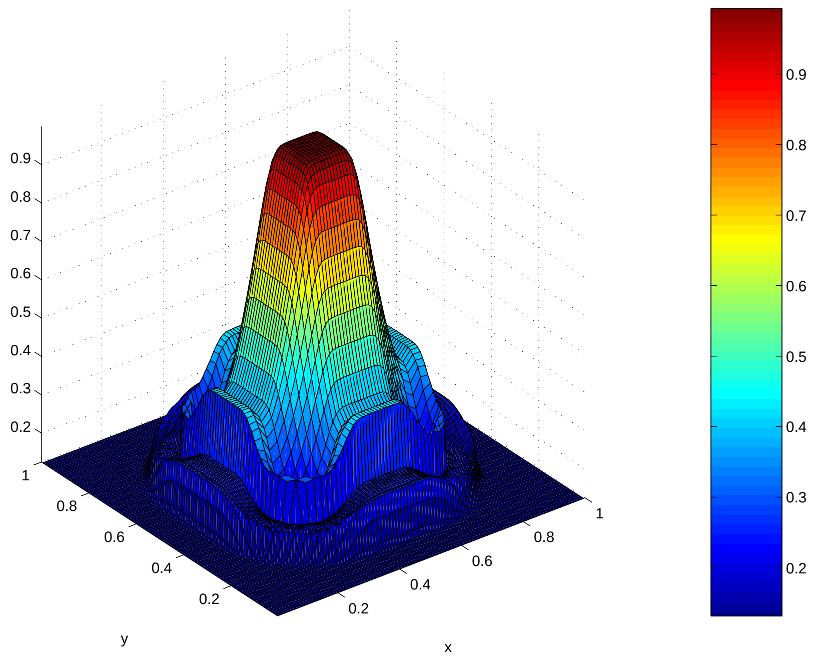

Figure 2.21: Square Sod problem Lagrangian calculation density surface plot at t=0.1.

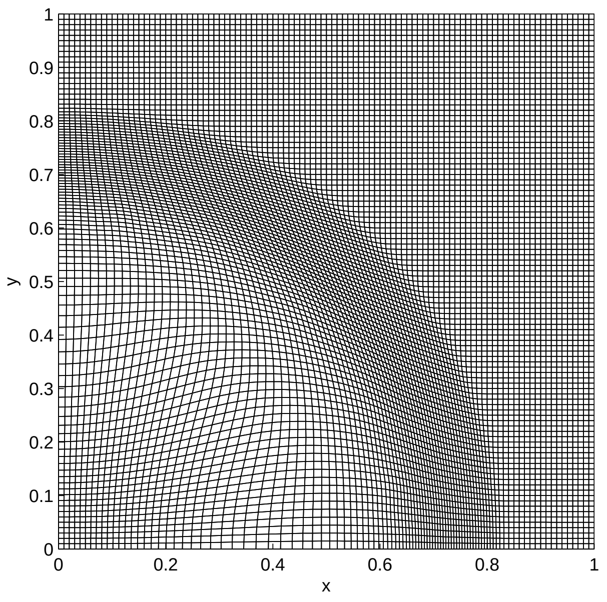

Figure 2.22: Radial Sod problem ALE mesh at t=0.25.

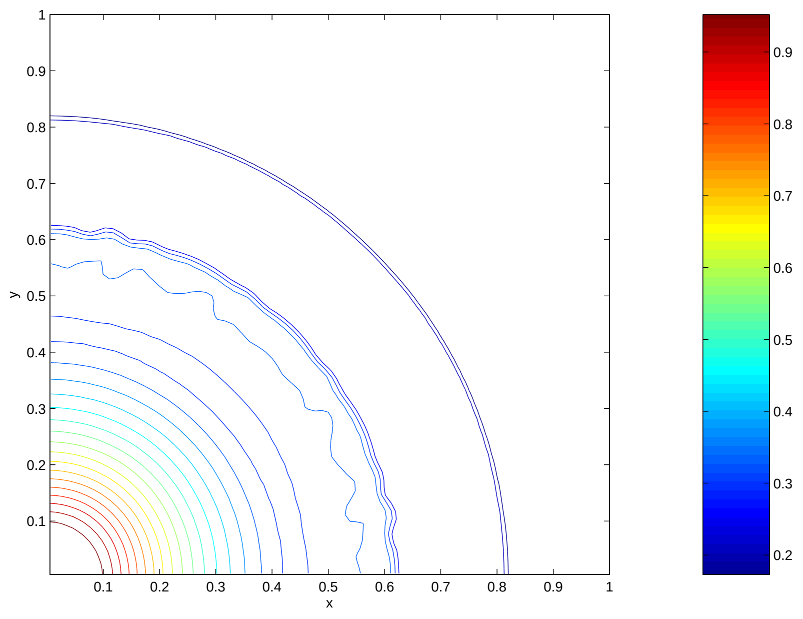

Figure 2.23: Radial Sod problem ALE calculation density contours at t=0.25.

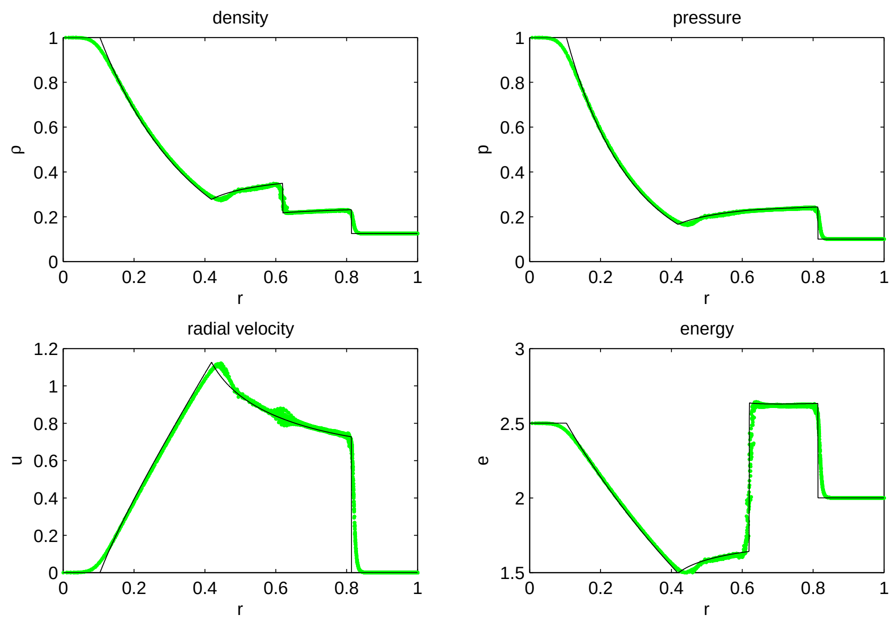

Figure 2.24: Radial Sod problem density, pressure, radial velocity and specific internal energy results at t=0.25, ALE calculation in green compared to 1D Riemann solver solution in black.

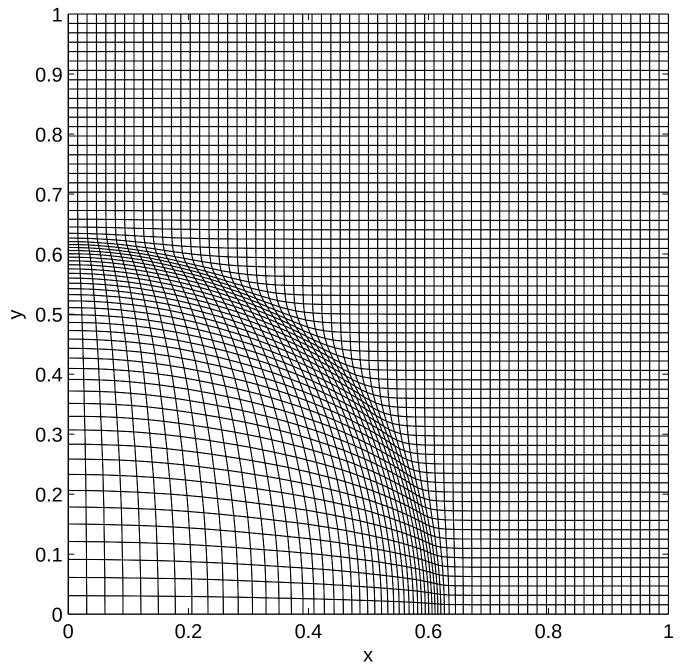

Figure 2.25: Sedov problem ALE mesh at t=0.1.

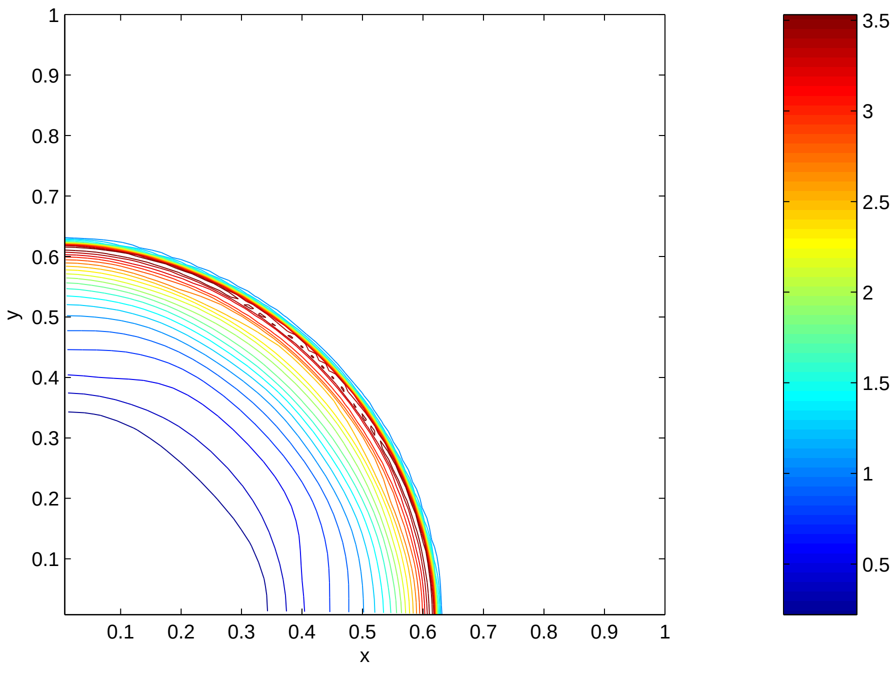

Figure 2.26: Sedov problem ALE calculation density contours at t=0.1.

Chapter 3

The isotropic adaptive mesh technique

In this chapter we develop an isotropic adaptive mesh technique, to increase the adaptivity of the code by including h-refinement. Instead of increasing the resolution by drawing more elements into the areas of interest, possibly leaving other areas of meshing under resolved, this method increases the total number of elements by subdividing elements. There are two competing approaches to subdividing elements, structured Adaptive Mesh Refinement and cell by cell refinement.

Adaptive Mesh Refinement or structured AMR was developed by Berger and Oliger in the early 1980's as an adaptive finite difference method [25], [26]. The method creates a hierarchy of refinement levels, each level containing one or more grids. Every level is integrated using a standard method with any missing data at coarse-fine interfaces being provided by interpolation. Time refinement is included, the fine levels take more time steps than the coarser levels and the integration of the hierarchical levels is interwoven. The data for coarse elements that have been refined is replaced by injecting from the fine values to give a new coarse value. Refinement or derefinement is triggered using a Richardson extrapolation to estimate the solution error, and flagged elements are clustered into rectangular grids to be refined. In the original paper rotated grids were also employed using transforms.

The original method was converted into a finite volume method for the steady state Euler equations in [24], this paper paid particular attention to conservation at the coarse-fine interfaces. The data structure of AMR is described in [23], where levels are maintained in a tree data structure and the variables are held in linked lists. Berger and Colella applied these ideas to shock hydrodynamics, again considering in detail the conservation at coarse-fine interfaces [22]. Quirk gives a great deal of detail about AMR when combining it with a Roe approximate Riemann solver in his PhD thesis [58]. AMR was extended to hyperbolic systems of conservation laws in three dimensions in [21], to irregular regions in [52] and to multimaterial gas dynamics in [57].

Anderson, Elliott and Pember have combined structured Adaptive Mesh Refinement with a staggered mesh ALE method [6], [7], [8], [9]. The interpolation formulae for transferring the solution during refinement and derefinement are second order, monotone, preserve constant fields, and conserve mass and momentum. The mesh relaxation strategy requires the mesh levels to be carefully linked: the finest mesh is relaxed (using the coarse-fine interface as a fixed boundary), its new positions replace the repeated nodes on the mesh one level coarser, any nodes on this coarser level not already relaxed are now relaxed. The Lagrangian step, mesh relaxation and remapping are applied on each level (with interpolation used at coarse-fine interfaces) and the advancement in time must be interwoven similarly to Berger's approach if time refinement is used.

An excellent review of the current areas of interest and applications within AMR research is given in [54]. Present application areas include: astrophysics, turbulent molecular clouds, black holes, weather simulations, magneto-hydrodynamics, radiation transport, reacting flows, hyperbolic conservation laws, shock waves and vortex simulations. Details of the main AMR codes are given and parallelisation is also discussed. The predominant approach within the book is based on the structured AMR approach of Berger and all the articles consider only Eulerian meshes that do not move, with the exception of a paper by Anderson [5] where AMR is applied to the ALE formulation.

The implementation, coding and parallelisation of structured Adaptive Mesh Refinement methods are discussed in [14]. Some useful details of solving elliptic equations and dealing with slaved or disjoint nodes are discussed, but the main thrust of the text is toward computer science and implementation issues.

The Adaptive Mesh Refinement discussed above uses a hierarchy of structured meshes, this is in contrast to refining individual or small groups of elements in a cell by cell manner. Cell by cell refinement is often used in unstructured grids with triangular elements: Piggott, Pain *et al.* have applied h-refinement and r-refinement in ocean modelling [53]; Wood has investigated edge swapping, point insertion and deletion, and r-refinement for advection and advection-diffusion equations [88]; Walter evaluates the benefits of coarsening compared to just refinement on mesh quality and the number of elements required [83]; Vijayan and Kallinderis developed a 3D finite volume scheme including refinement and derefinement to solve the Euler equations [77]; Vollmer uses quadtree data structures for adaptive mesh refinement applied to scalar conservation laws [78]. Details of other research that are highly related to anisotropic refinement are given in Chapter 6.

A cell by cell isotropic refinement method for quadrilateral elements that uses quad/octree data structures and includes time refinement was developed for the Euler equations on an Eulerian mesh by Khokhlov in [45]. Popinet uses cell by cell refinement of rectangular elements to solve the incompressible Euler equations using a multilevel Poisson solver [55].

Marchant and Weatherill [50] have developed a refinement strategy that uses quadtree data structures and includes point enrichment, deletion and movement for the Euler equations applied on meshes containing triangles and quadrilaterals. The cell centred finite volume scheme can be applied to arbitrary polygons. The mesh relaxation uses a modified Laplace operator that takes all neighbouring nodes into account.

In a paper by van der Vegt and van der Ven a discontinuous Galerkin finite element method using unstructured anisotropic refinement of hexahedrons is considered [71]. The advantage of using the discontinuous Galerkin method is that the discretisation is local within each element, meaning that disjoint nodes pose no problem. The disadvantage is that both the variables and their moments must be carried for each element. This paper uses tree like data structures and the mesh never moves with the flow or r-refines, the coarse mesh is fixed. Aftosmis uses anisotropic refinement of quadrilateral elements to simulate viscous flow [2].

Kallinderis and Baron also use anisotropic refinement of quadrilaterals in their work with the Navier-Stokes equation [43]. Their viscous refinement includes equation refinement to reduce calculation run times. Their work compares feature detection using various variables. Cell based connectivity arrays are used similar to the approach in this thesis. Interfaces are treated by using the parent cell, rather than the fine cells, and then interpolating. The adaptive mesh strategy is coupled with a multigrid method. Once again the mesh does not move with the flow; the coarse mesh is fixed.

Keats and Lien have developed an anisotropic refinement technique for quadrilateral elements on a Cartesian mesh [44]. New elements are inserted into the mesh and given refinement level indices. The adaptation criteria are discussed, the first order gradient was found to be the most reliable indicator, and optimal cell sizes are determined from it. The derivation of the fluxes on the unstructured mesh, the determination of gradients and Van Leer limiters are well developed. Results are presented for flow over a backward and forward step and attempts are made to quantify processor and memory savings. The mesh is an Eulerian mesh, and the fact that it does not move with the flow simplifies matters somewhat, also all variables are cell centred so disjoint nodes need not be discussed.

In [13] a technique referred to as adaptive mesh insertion is described. Elements are inserted into the mesh, this is either manually triggered or based on the element's aspect ratio [13] rather than being solution dependent. Furthermore, elements may only be inserted in one direction.

Hyperbolic conservation laws involve features that evolve with time, any refinement technique that attempts to resolve these features must be dynamic. The refinement must track and surround the feature of interest. Therefore, a solution adaptive or error adaptive trigger for refinement is required, the mesh and data structures must also be easy to update.

There are two competing approaches to data structures: the hierarchical level approach and the quad/octree approach. Although storing the levels in a tree data structure is efficient, storing the refined elements in a tree (each element having 2-4 'leaves') can become very expensive to traverse. Extra pointers often have to be introduced to include information about neighbours or allow backward movement through the tree. In this work we use a hierarchical level data structure, but without defining separate arrays for each cell by cell piece of refinement, as this would create many arrays. The data structure used builds upon the existing programs element-node, element-element and node-node connectivity arrays. The original coarse connectivity arrays are retained and never altered. New dynamic levels of connectivity arrays are created that contain the coarse and fine elements and describe the combined Dynamic Mesh.

To summarise the different methods, structured AMR must refine a rectangular group

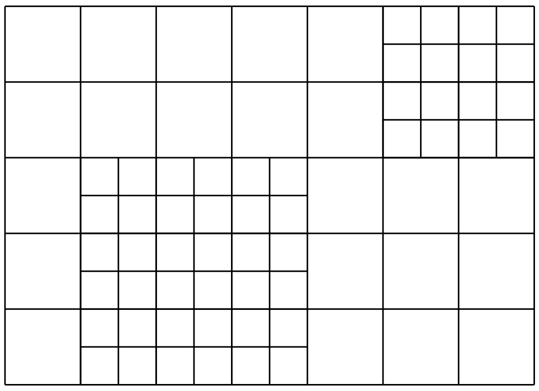

Figure 3.1: Diagram of the Dynamic Mesh with fine elements inserted.

or cluster of elements as opposed to refinement or derefinement of individual elements. The cell by cell approach reduces the number of fine elements required and avoids any need for a clustering algorithm, although the required data structures are less efficient. This approach may reduce the possibility of the mesh imprinting on the solution by not forcing the refinement blocks to align with the spatial directions. The ALE code is already a finite element unstructured code and so it is flexible enough to adopt the cell by cell refinement approach.

In structured AMR, solutions are obtained for every level of the hierarchy and solution vectors for all of them are required. In contrast elements may be inserted into the mesh forming a Dynamic Mesh, illustrated in Figure 3.1, that changes in time. The solution can then be obtained on this combined or Dynamic Mesh, rather than wastefully solving on every level, which requires more time, complicated interweaving of levels and increased storage.

In structured AMR the original structured mesh methods are unchanged they are just applied on every level. However interpolation is required for missing data, flux fixes are required for conservation, and the coarse solution must be replaced by an average of the finer solutions. If we solve once on the Dynamic Mesh the flux fixes and replacement of the coarse values are not required as only one solution exists. However, this approach requires changes to the solution methods so that they may be applied on an unstructured mesh.

In the staggered mesh disjoint or hanging nodes are used in the transition from a fine to a coarse region. The disjoint nodes' positions and velocities are constrained by the surrounding coarse nodes. No ghost cells are required to transfer or interpolate data, since no separate grid levels exist. This is important because requiring ghost cells in a cell by cell approach would prove very expensive.

In structured AMR different levels are stored separately allowing them to be individually integrated in time easily. However, there is much debate as to how necessary time refinement actually is. Although a small time step throughout the domain will lead to some coarse cells that have too small a time step, these should be fewer in number than the fine cells. Many argue that the increase in efficiency from the spatial refinement is sufficient and any advantages of time refinement are outweighed by the increased code complexity [6], [33]. Therefore, in this work only spatial refinement is considered.

This chapter develops a refinement technique with the following features:

- cell by cell refinement,
- element insertion,
- solution obtained on the combined Dynamic Mesh,
- disjoint nodes on interfaces between coarse and fine meshing,
- automatic refinement based on the solution gradient.

In this chapter we consider isotropic refinement, where both directions are subdivided, in a later chapter the method is generalised to include anisotropic refinement, where elements may be subdivided in only one direction.

3.1 Element insertion

The isotropic cell by cell refinement divides coarse elements into four fine elements. The mesh is considered as a combined Dynamic Mesh, containing both coarse and fine elements, and the new fine elements are inserted into the Dynamic Mesh. Hence a coarse element with general element number 6, see Figure 3.2, becomes four fine elements with general element numbers 6, 7, 8, 9 in an anticlockwise direction and the later element

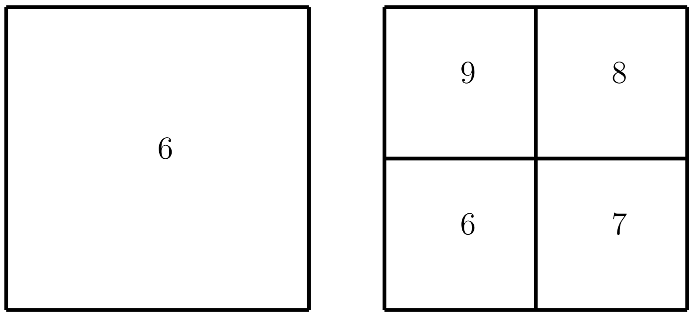

Figure 3.2: Element numbering changes when an element is inserted, the existing element numbers are all shifted to insert the new elements.

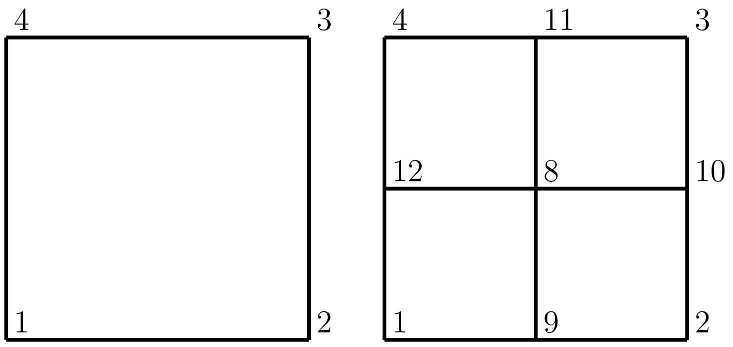

Figure 3.3: Node numbering for new nodes, the original coarse node numbers never change.

numbers are shifted to insert these. The original coarse node numbers are never changed, new fine nodes are added to the end of the list of nodes, see Figure 3.3.

The data structure from the original coarse mesh is retained, including all of the original connectivity arrays. This allows us to loop through the parent elements and refer to their neighbours for refinement and coarsening operations. As elements are inserted, the connectivity of the Dynamic Mesh changes and new connectivity arrays are created. These connectivity arrays change as the mesh evolves and are used in the ALE method as they represent the Dynamic Mesh for which the solution is obtained. The original and the dynamic arrays are related by an array that lists, for each original coarse element, its dynamic element number or the four fine dynamic element numbers (if it has been refined).

3.2 Solution transfer

We wish to solve on the finest resolution in each part of the mesh, thus the solution is obtained on the Dynamic Mesh, containing both coarse and fine elements. The solution vectors for the cell centred quantities and nodal variables are dynamic, they must be altered to correspond to the new element and node numbering as the Dynamic Mesh changes.

An element that is refined is subdivided through the edge midpoints forming 4 fine elements. Note that this subdivision is with respect to the element's local coordinates, not global spatial coordinates. This is beneficial as the refinement will remain aligned with the original element orientation and the ALE method will have aligned the original element with the flow or features of interest.

During refinement the old coarse values are interpolated to provide new fine values. In this chapter we describe the approach for first order interpolation, Chapter 5 extends the solution transfer to second order accuracy. For cell centred quantities the coarse element is assumed to have constant solution and the four new fine elements take this solution value. Therefore for an element e that is to be refined

$$\varphi_{e_i} = \varphi_e, \quad \text{for i=1,2,3,4,} \tag{3.1}$$

where the e_i's represent the new fine elements and φ is the variable.

The new nodal variable introduced at the centre and shown in Figure 3.4, φ_{n_c}, is given by an average of the values at the four corner nodes

$$\varphi_{n_c} = \frac{1}{4}(\varphi_{n_1} + \varphi_{n_2} + \varphi_{n_3} + \varphi_{n_4}). \tag{3.2}$$

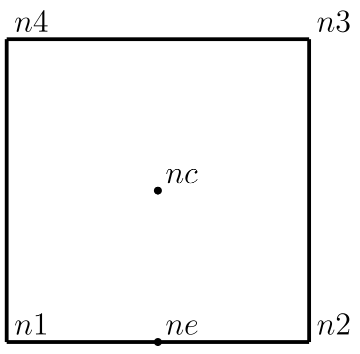

Figure 3.4: Diagram to illustrate the calculation of new fine nodal variables.

For an edge node, n_e, introduced between two coarse nodes n_1 and n_2 the solution ϕ is an average of the coarse nodes' solutions

$$\varphi_{n_e} = \frac{1}{2}(\varphi_{n_1} + \varphi_{n_2}). \tag{3.3}$$

Nodal boundary conditions must be set when a node is introduced on an external boundary.

During derefinement the unwanted nodal values are simply removed. Derefinement of the cell centred quantities is achieved using a conservative averaging procedure

$$\varphi_e = \frac{\sum_{i=1}^{4} \varphi_{e_i} X_{e_i}}{\sum_{i=1}^{4} X_{e_i}}, \tag{3.4}$$

where the e_i's denote the four elements that are being derefined, X is the weight (or conservative basis) and e is the new coarse element. Volume weighting is used for density while specific internal energy is mass weighted.

3.3 The adaptive mesh Lagrangian step

In the Lagrangian step we want to solve on the combined Dynamic Mesh, rather than expensively solving on a hierarchy of refinement levels. Almost all the cell centred quantities (the exception is artificial viscosity discussed at the end of this section) are evaluated in an element by element manner and require no changes when calculated on the Dynamic Mesh.

On an interface between different refinement regions, as shown in Figure 3.5, there are nodes with 3 nodal neighbours, rather than 4. These nodes are termed disjoint or hanging nodes and require special treatment to ensure that they remain constrained. Each disjoint node lies a set distance between two non-disjoint nodes on the interface, this distance ratio from the neighbouring non-disjoint nodes is recorded and the disjoint node retains this spacing throughout the calculation.

Each disjoint node is slaved to lie the same ratio along the line joining the neighbouring non-disjoint nodes on the interface. Using the notation in Figure 3.6 the disjoint node's position becomes

$$\mathbf{x}_{n_D} = r\mathbf{x}_{n_2} + (1 - r)\mathbf{x}_{n_1}. \tag{3.5}$$

The velocity of the disjoint node must also be constrained consistently as

$$\mathbf{u}_{n_D} = r\mathbf{u}_{n_2} + (1 - r)\mathbf{u}_{n_1}. \tag{3.6}$$

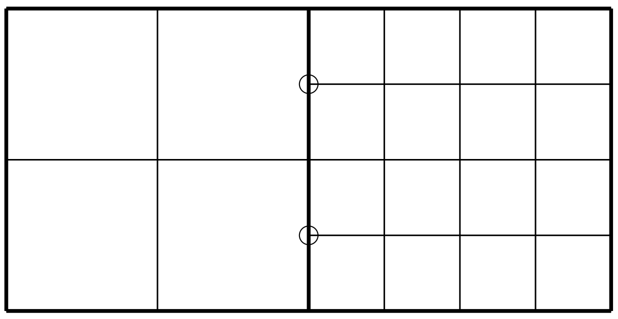

Figure 3.5: Diagram of a course-to-fine interface. Empty circles indicate disjoint nodes.

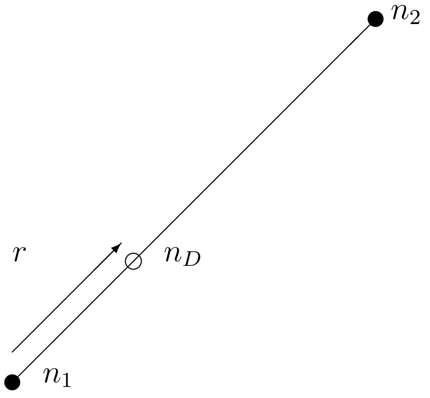

Figure 3.6: Diagram showing the distance ratio. Filled circles represent non-disjoint nodes and the empty circle represents the disjoint node.

Disjoint nodes are treated as non-dynamic points that have no mass or momentum, so the mass gathered at these points must be redistributed to the non-disjoint nodes

$$\begin{aligned} M_{n1} &= M_{n1} + (1-r)M_{nD} \\ M_{n2} &= M_{n2} + rM_{nD}. \end{aligned} \tag{3.7}$$

The nodal forces are similarly altered, for example the nodal force in the x direction becomes

$$\begin{aligned} f_{x_{n1}} &= f_{x_{n1}} + (1-r)f_{x_{nD}} \\ f_{x_{n2}} &= f_{x_{n2}} + rf_{x_{nD}}. \end{aligned} \tag{3.8}$$

3.3.1 Christensen's artificial viscosity

In order to use Christensen's artificial viscosity with the adaptive mesh scheme some alterations to the method are required. To retain the accuracy of the fine solutions and not spread the shock over the coarse elements we wish to use the finest possible element stencil for the Christensen's artificial viscosity.

If an element is surrounded by four elements of the same type then the Christensen method is unchanged regardless of whether the elements are coarse or fine. Therefore we need only consider the situations on coarse-to-fine interfaces and fine-to-coarse interfaces.

The Christensen limiter compares the velocity gradient over three elements joined on the same edge. These edges must be identified for a coarse element on a coarse-to-fine interface and created for a fine element on a fine-to-coarse interface. The length of the edge is taken into account in the velocity gradient, therefore the change in size of the elements should not in itself cause a problem.

In the case of a coarse element on a coarse-to-fine interface the correct aligning fine element for the edge must be identified. Referring to Figure 3.7, when limiting the edge artificial viscosity for element ec's bottom edge the correct aligning element is ef.

The situation is more complicated for a fine element on a fine-to-coarse interface. The edge joining the disjoint node perpendicular to the interface terminates at the disjoint node. To provide a velocity gradient in the coarse element that the edge would have divided (if it had continued from the disjoint node) a velocity value is required in the middle of the opposite side at point p. This situation is illustrated in Figure 3.8 for the

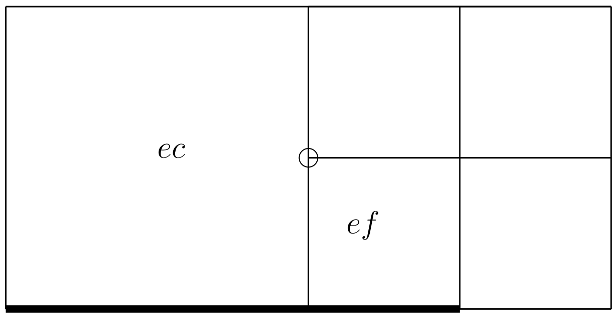

Figure 3.7: Diagram of a coarse-to-fine interface showing the correct aligning element for limiting along the bottom edge.

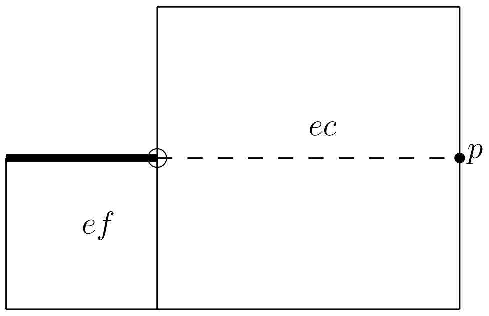

Figure 3.8: Diagram of a fine-to-coarse interface. Limiting along top edge of the fine element requires the continuation of this edge through the neighbouring coarse element to point p.

top edge of the fine element. The velocity at p is interpolated from the coarse nodes either side of it. The mesh legs L_{hor} and L_{ver} have the same directions for half of the element as the coarse element had. Since both the area and L_{hor} would have been halved no correction is required. Therefore the velocity gradient can be calculated using the velocity at p and the rest of the equation remains unchanged. The remaining parts of the method can continue as before.

3.4 Refinement criteria

It remains to describe how the elements to be refined or derefined are selected. There are many different refinement sensors. Berger and Colella [22] use a method based on Richardson extrapolation, this requires both a fine solution and a coarsened solution to be advanced in time and then compared. This is clearly expensive and may predict overly large errors around discontinuities. Other approaches based on equidistribution of errors or minimising error functions are possible but are expensive. Barlow [13] uses an aspect ratio sensor for adaptive mesh insertion, but here we adapt based on the solution.

Anderson *et al.* use a normalised second difference of pressure and/or density to trigger their refinement [6]. Aftosmis argues that both smooth features and shocks must be refined to converge to the correct solution, and that shock detection should differ from smooth feature detection as shock width is related to the element size [2]. Aftosmis uses an undivided second difference in pressure normalised by a weighted average of the local pressures as a shock sensor that will locate both strong and weak shocks. An undivided second difference of density or velocity was then used on the rest of the flow to locate smooth features [2].

Kallinderis suggests undivided differences are used for shocks and suggests pressure or velocity be used, as density tends to lead to more refined elements [43]. While Khokhlov [45] and van der Vegt [71] use adaptation sensors based on a combination of many flow variables.

Keats [44] describes three approaches: an absolute value difference between the element and its neighbours, a ratio of the second and first order terms in a Taylor expansion and an estimation of the optimal cell widths using the second derivative. However, Keats found that the second derivative was too noisy for anisotropic refinement [44]. Furthermore, around shocks the code is reduced to first order, making a first order difference more appropriate.

Dannenhoffer found that using second differences lead to ragged refinement and holes in the refinement around an airfoil using the steady state Euler equation [32]. Dannenhoffer describes density as the best refinement variable based on accuracy, computation time and the refinement completely encasing the features of interest [32].

It was decided that an undivided first order difference would be used in this work as it is simpler than a second difference and does not suffer from the excessive noise noted by Keats and Dannenhoffer. The detection of shocks is most important, but smooth features and contacts may also need to be refined to converge to the correct solution. In particular the density was found to be the most reliable variable for sensing because it identifies contact discontinuities and is a cell centred variable (unlike velocity).

There seems to be much debate over whether normalisation is required, and if so whether a local or global value should be used. Normalisation is most useful where the range of solution values changes greatly over the calculation time. Local normalisation may be dangerous as it could cause division by zero. Normalisation by a global value is possible, either by the average value of the variable or by the average value of the change in variable, but it is not implemented at present.

The density change between a coarse grid element and its four nearest neighbours is used as the refinement sensor in this work. Values for the element densities on the coarse grid are obtained using

$$\rho_{coarse_e} = \begin{cases} \rho_e & \text{if coarse} \\ \frac{\sum_{i=1}^{4} \rho_i V_i}{\sum_{i=1}^{4} V_i} & \text{if fine.} \end{cases} \tag{3.9}$$

If the change in density is above the refinement parameter the element will refine, if the change in density drops below the derefinement parameter the four fine elements will derefine. Note that the derefinement parameter must be slightly lower than the refinement parameter otherwise elements will derefine too quickly. The refinement parameter must be specified for each problem and this is clearly a drawback in using solution differences as a sensor. The ideal refinement parameter will keep the number of elements small but provide enough refinement to encase the features of interest. Plotting the cumulative frequency of the change in density at different times throughout the problem can help to determine an efficient refinement parameter, this method was inspired by Dannenhoffer's automatic method for steady state problems [32]. This approach proved very successful when used for the two-dimensional Riemann problem.

3.5 Buffering

To assess the effect of a shock crossing a fine-to-coarse or coarse-to-fine interface a piston problem was run with a 1:2 refinement ratio and static refinement regions. Spurious oscillations were generated in both cases, see Figure 3.9 and Figure 3.10. The fine-to-coarse interface caused the biggest oscillations, in contrast to Quirk's results where the worst oscillations were caused by the coarse-to-fine interface [58]. The spurious oscillations are reflections or transmissions (depending on which way they move) generated by the change in mesh density. The shock data is perturbed slightly at the interface and the oscillations move according to the characteristics as if they were real waves. Further calculations were run with a mesh density ratio of 1:3, the fine-to-coarse results are shown for comparison in Figure 3.11. The oscillations were found to have even larger amplitudes, confirming that the 1:2 refinement ratio is indeed the better choice for the adaptive scheme, rather than using higher refinement ratios and causing too much of a transition in the meshing.

These experiments have confirmed the need for a sophisticated dynamic adaptive mesh technique. Any fine regions created must completely surround the feature of interest and be large enough to prevent the feature escaping the fine region during the next Lagrangian step. To stop features escaping, when a cell is flagged for refinement its eight immediate neighbours are also flagged to provide a buffer zone.

Further research into reducing the spurious oscillations generated when a shock crosses an interface between coarse and fine meshing would be beneficial to avoid having to refine large areas of the domain in a problem with many shocks.

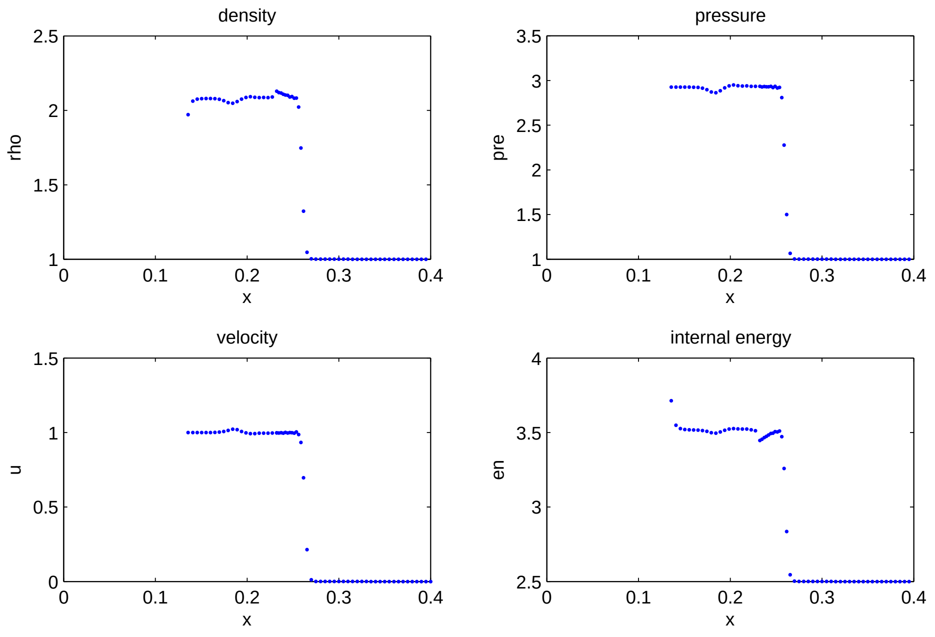

Figure 3.9: Coarse-to-fine 2 region test problem, mesh density ratio 1:2, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

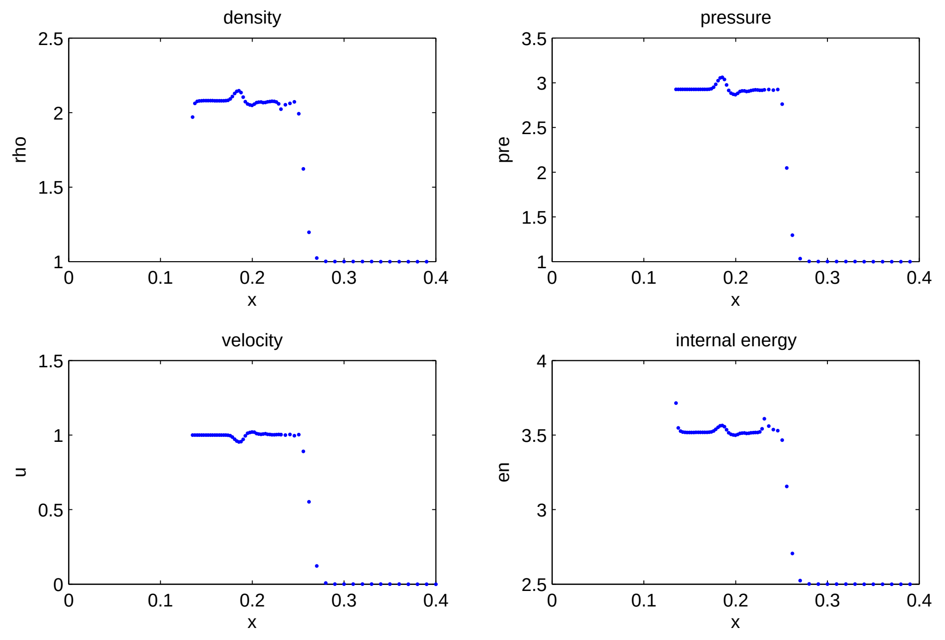

Figure 3.10: Fine-to-coarse 2 region test problem, mesh density ratio 2:1, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

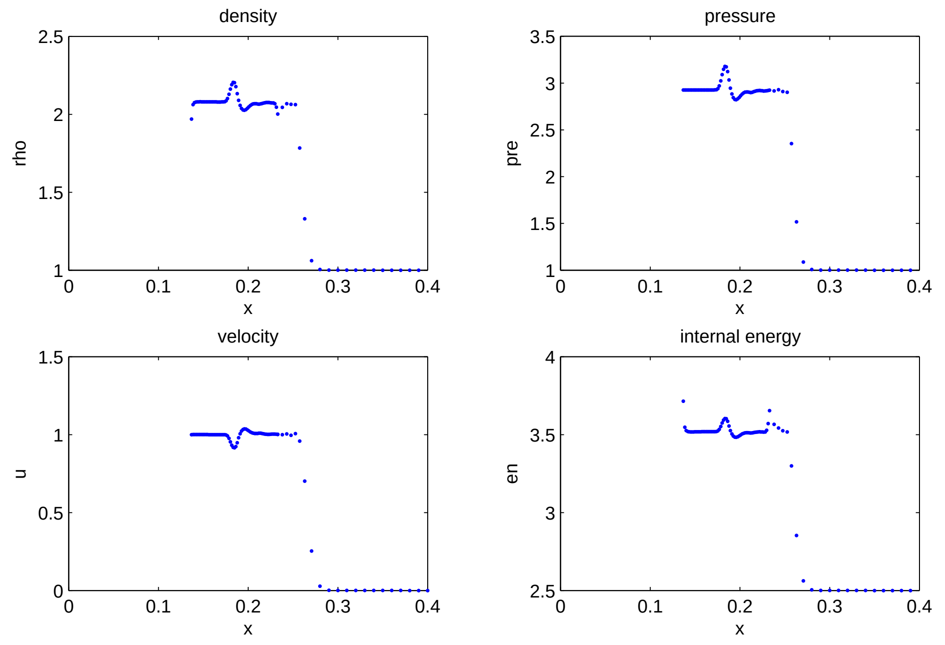

Figure 3.11: Fine-to-coarse 2 region test problem, mesh density ratio 3:1, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

3.6 Lagrangian results for the adaptive method

3.6.1 Sod's shock tube problem

Sod's shock tube problem was rerun using the isotropic adaptive mesh method with an initially 50×5 mesh, the refinement parameter was 0.05 and the derefinement parameter was 0.01. The adaptive mesh is shown in Figure 3.12 and shows that one area of refinement forms around the rarefaction fan and contact and a separate refinement region occurs around the shock, with coarse meshing in the constant states as required. Note that due to buffering the constant state between the rarefaction and contact is not derefined.

Figure 3.13 compares the solutions for a 100×10 uniformly fine calculation and the adaptive calculation, they are seen to agree very favourably. The adaptive mesh results were compared against the exact results from Toro's iterative Riemann solver [66] to obtain a measure of the error. The 1-norm error for a calculation with N elements of volume V_i is calculated as

$$\|e\|_1 = \sum_{i=1}^{N} \left[\left| \rho_{amr} - \rho_s \right| V_i \right], \tag{3.10}$$

whilst the 2-norm error is

$$\|e\|_2 = \sqrt{\sum_{i=1}^{N} \left[\left(\rho_{amr} - \rho_s \right)^2 V_i \right]}. \tag{3.11}$$

Table 3.1 shows that the adaptive calculation errors are almost identical to the uniformly fine calculation, verifying that comparable errors can be achieved with significantly fewer elements. Figure 3.14 shows the reduction in the number of elements achieved when using the adaptive method. Both the uniform calculation and the adaptive calculation require 207 time steps, summing the elements each step over the entire calculation leads to 207000 elements for the uniform calculation, while the adaptive calculation only requires 40% of that number. Having verified the performance of the method for a one-dimensional problem a two-dimensional problem is used to ascertain a more realistic idea of the savings that the adaptive method could provide.

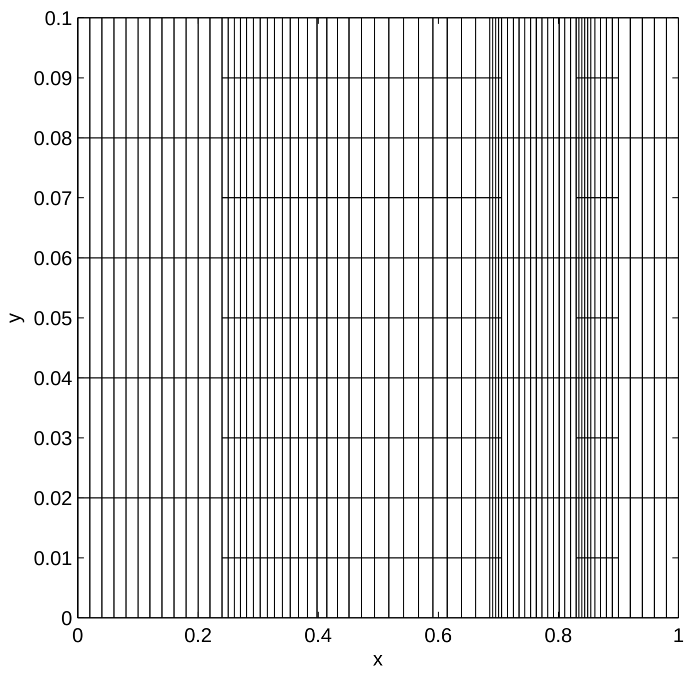

Figure 3.12: Sod's shock tube problem Lagrangian isotropic adaptive mesh at t=0.2.

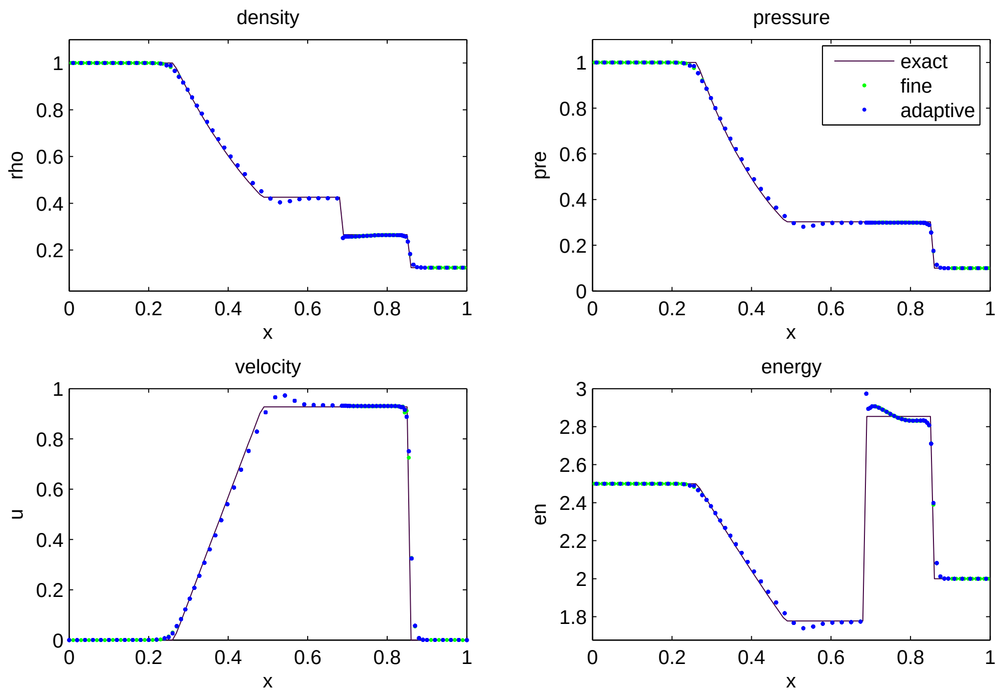

Figure 3.13: Sod's shock tube problem comparing density, pressure, velocity and specific internal energy for isotropic adaptive calculation solution and fine solution at t=0.2.

Calculation	$\|e\|_1$	$\|e\|_2$	max error
fine	0.000593	0.002828	5.523×10^{-6}
adaptive	0.000597	0.002833	5.533×10^{-6}

Table 3.1: Sod's shock tube problem solution errors for uniformly fine and adaptive calculations at t=0.1.

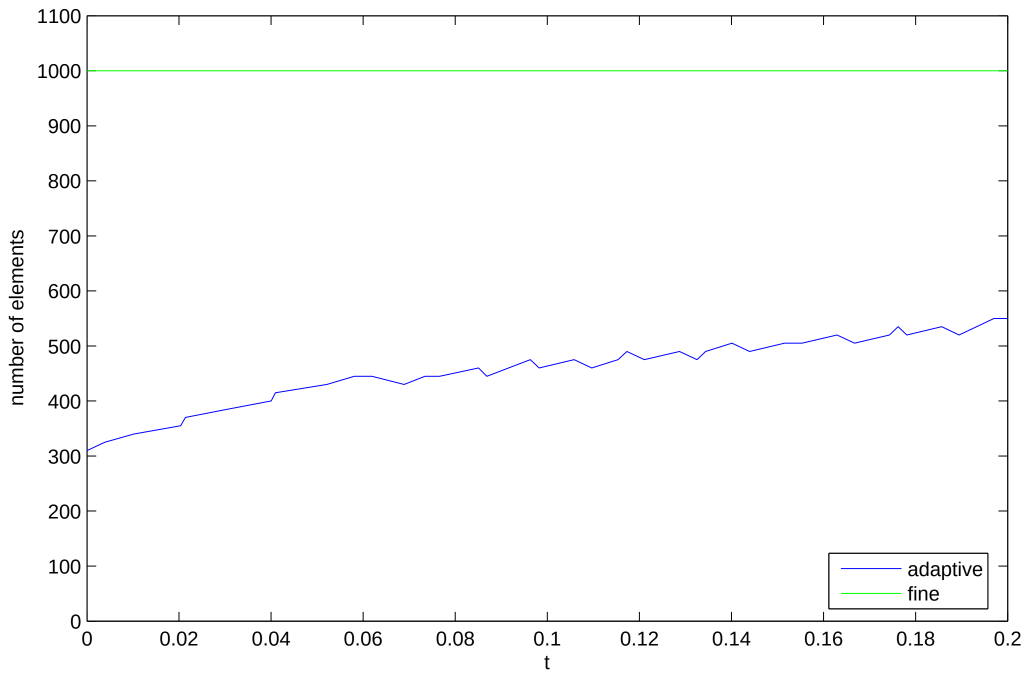

Figure 3.14: Sod's shock tube problem comparing number of elements required over calculation time.

3.6.2 Square Sod problem

The square Sod problem was run to t=0.1 using the isotropic adaptive mesh method and an initially 50×50 mesh, with a refinement parameter of 0.04 and a derefinement parameter of 0.025. The solution evolved as a one dimensional Sod shock tube along the middle of each side with radial Sod behaviour at the corners. Refinement occurs from the start of the rarefaction fan, shown in Figure 2.21, until after the contact. There is a derefined region in-between the contact and the shock, and then refined areas around the shock, see Figure 3.15. Referring to Figure 3.15 we see that the mesh is smooth and untangled, confirming that this problem does not require mesh relaxation. The adaptive mesh calculation and the uniformly fine 100×100 calculation gave very comparable density contours, see Figures 3.16 and 3.17. The adaptive calculation required a few more time steps, 545, compared to the uniformly fine calculation's 423. However, the number of elements at each time step is significantly lower, see Figure 3.18, and the total number of elements over the whole calculation is reduced to 60% of that for the uniform fine calculation. The adaptive calculation reduces the run time by 30%, a very significant saving.

We have verified that the Lagrangian isotropic adaptive mesh method provides excellent savings in calculation run time without loss of accuracy. In the next chapter we extend the method to include mesh relaxation and advection, forming an isotropic adaptive mesh ALE method that can then be applied to a wider range of test problems.

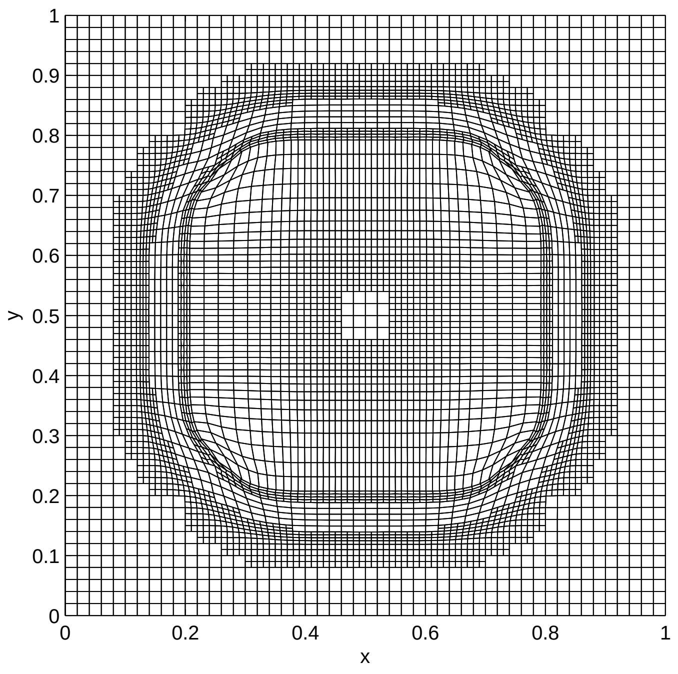

Figure 3.15: Square Sod problem Lagrangian adaptive mesh at t=0.1.

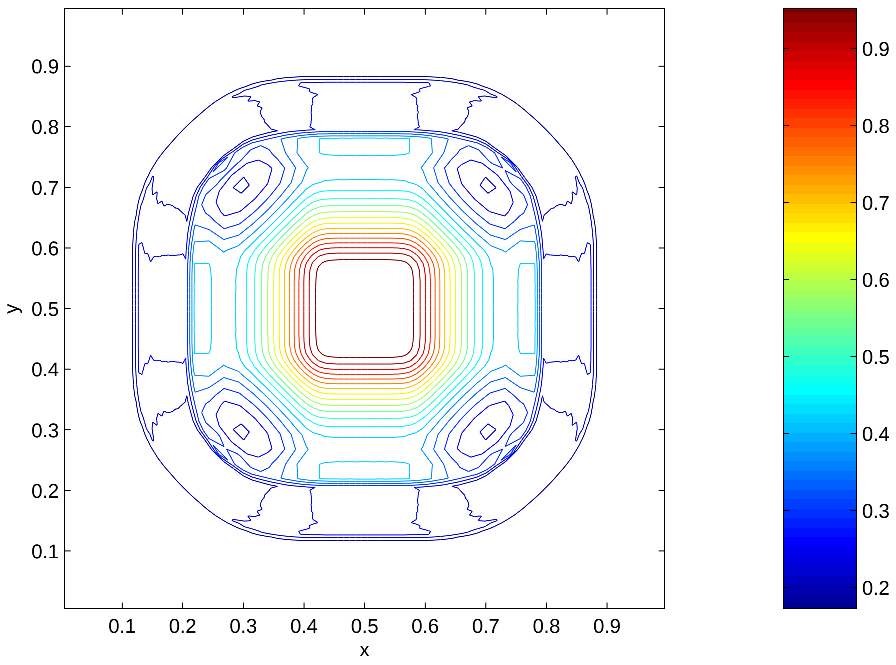

Figure 3.16: Square Sod problem uniformly fine calculation density contours at t=0.1.

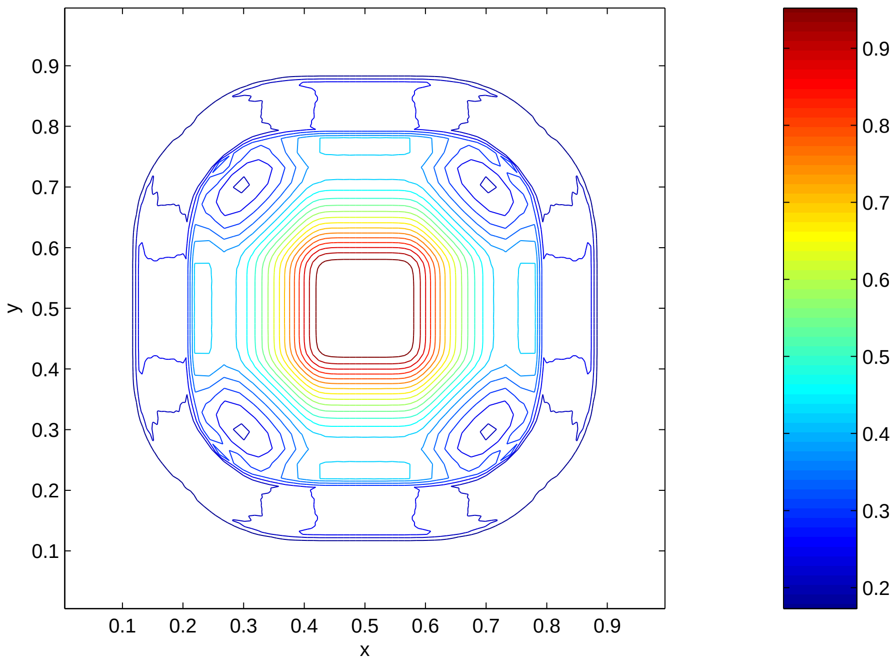

Figure 3.17: Square Sod problem adaptive mesh calculation density contours at t=0.1.

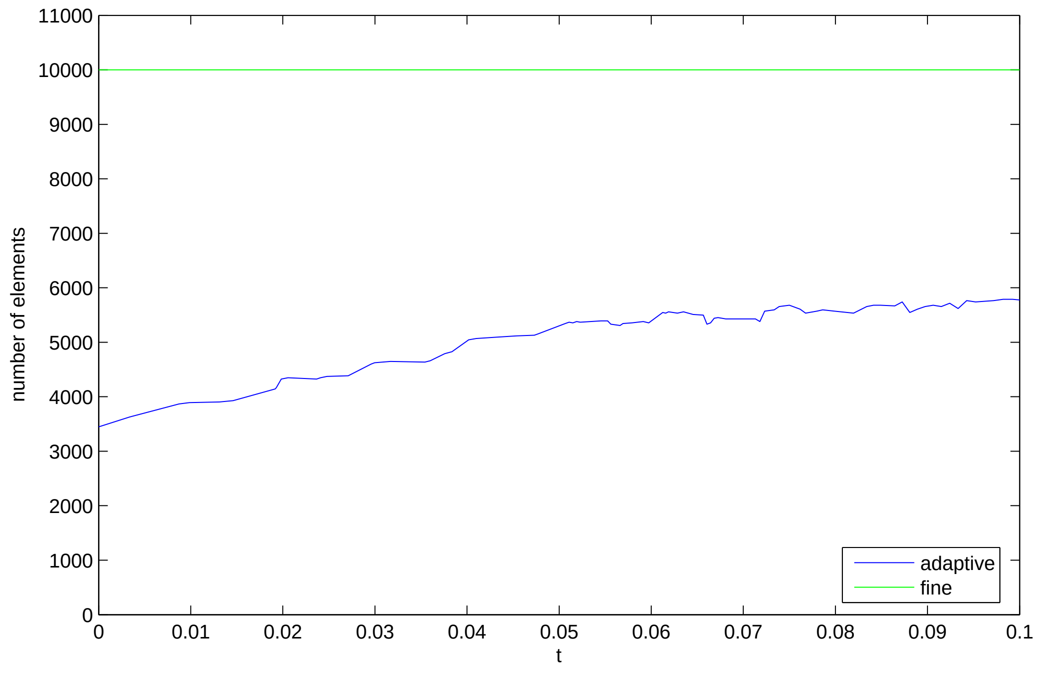

Figure 3.18: Square Sod problem comparing the number of elements required over calculation time.

Chapter 4

The isotropic adaptive mesh ALE method

In the original ALE method the total number of elements was fixed and so increased resolution in one area could lead to other parts of the mesh becoming under resolved. By introducing cell by cell refinement the total number of elements can vary, and the resolution around features of interest can be increased without other areas of the mesh becoming under resolved.

Expanding the adaptive mesh technique into an isotropic adaptive mesh ALE method will improve the quality of the mesh by reducing any mesh tangling formed during the Lagrangian step, and hence expand the range of test problems that the method may be applied to.

We wish to solve on the finest resolution level existing in each part of the domain, represented by the combined or Dynamic Mesh, which uses disjoint nodes at resolution transitions. This is a more efficient strategy than solving on every refinement level and is more applicable for cell by cell refinement. Powell [56] at AWE has implemented an alternative strategy that does decompose the Dynamic Mesh into levels.

4.1 Equipotential mesh relaxation on an isotropic adaptive mesh

The Winslow-Crowley equipotential relaxation method [86], [51] discretises the inverse Laplace equation using finite differences to obtain relaxed node positions, this moves nodes toward an equally spaced orthogonal distribution. On the Dynamic Mesh this would cause the fine meshing to spread into the coarse regions. The Dynamic Mesh also contains disjoint nodes that are non-dynamic points and do not have enough nearest neighbours to form the finite difference stencil.

Anderson *et al.* comment on the need for care when applying equipotential relaxation to their structured AMR scheme [6]. The elliptic equations behave in a global manner and require the mesh levels to be carefully linked: the finest mesh is relaxed (using the coarse-fine interface as a fixed boundary), its new positions replace the repeated nodes on the mesh one level coarser, any nodes on this coarser level not already relaxed are now relaxed. However we wish to apply the elliptic equations to the Dynamic Mesh as an unstructured mesh, rather than iterating levels separately.

It is fairly easy to derive one dimensional unequally spaced finite difference schemes, but it becomes more complicated in two dimensions. We do not want to have to revert back to the least squares or length weighted finite difference schemes that have been formulated for a cloud of unstructured points, as we would like to retain the basics of the quadrilateral derivation.

Marchant and Weatherill have developed solution adaptive r-refinement combined with h-refinement on an unstructured mesh [50]. This scheme can deal with arbitrary polygons and because it is solution adaptive does not suffer from the finer regions spreading into the coarse regions. As the Lagrangian method behaves similarly to solution adaptive r-refinement we do not use this approach.

Winslow's method has been applied on unstructured meshes using a variational principal and finite elements [51]. However, this method will not solve the problem of finer regions spreading into the coarse regions.

Winslow's method can be applied without modification to nodes surrounded by elements of the same resolution, only the nodes at resolution transitions require extra consideration. In this work two approaches are detailed for applying Winslow's equipotential relaxation on the Dynamic Mesh (or unstructured quadrilateral meshes that include h-refinement). In both approaches disjoint nodes are treated by slaving them to lie on the interface between the non-disjoint nodes. Not only does this eliminate the need for a stencil for the disjoint nodes, but it also treats the disjoint nodes as non-dynamic points as in the Lagrangian step.

The simplest method investigated was to revert to a coarse nodal stencil at a resolution transition. This retains second order accuracy with respect to the coarse mesh spacing and stops the fine nodes moving into the coarse regions. However nodes on coarse-fine boundaries are moved with respect to the coarse nodes only, reducing the effect of the meshes fine detail and possibly moving nodes out of their surrounding fine stencil. This method reverts between two different stencils rather than using a consistent approach for all nodes.

The second more general method uses the same stencil for all non-disjoint nodes. The method identifies the different fine and coarse nodal spacings and takes these into account when discretising the derivatives. Each node of the stencil is given a length value, for example $C_{\phi,\psi+1}$, which denotes if it is coarse or finely spaced from the centre node (the node that is being moved). Due to the isotropic nature of the refinement one value is enough to indicate the spacing in both directions. Each derivative is discretised as a weighting of the positions of the stencil nodes, the weights being chosen to minimise the truncation error. The truncation error is determined by expanding each nodal position in a Taylor Series expansion about the centre node. When the length values are included in these expansions the nodal weights become dependent on the length values.

Consider the discretisation of $\frac{\partial^2 x}{\partial \phi^2}$ as a weighting of the $(\phi-1,\psi)$ node, the (ϕ,ψ) node and the $(\phi+1,\psi)$ node (recalling that $\phi-1$, ϕ, and $\phi+1$ correspond to mesh lines). The spacing between the (ϕ,ψ) node and the $(\phi+1,\psi)$ node is expressed as $lC_{\phi+1,\psi}$, where $C_{\phi+1,\psi}=1$ represents a fine spacing and $C_{\phi+1,\psi}=2$ represents a coarse spacing. The Taylor expansion of $x_{\phi+1,\psi}$ is

$$\begin{aligned} x_{\phi+1,\psi} &= x_{\phi,\psi} + C_{\phi+1,\psi} l (\frac{\partial x}{\partial \phi}) + \frac{C^2_{\phi+1,\psi} l^2}{2} (\frac{\partial^2 x}{\partial \phi^2}) \\ &+ \frac{C^3_{\phi+1,\psi} l^3}{6} (\frac{\partial^3 x}{\partial \phi^3}) ... \end{aligned} \tag{4.1}$$

A similar equation will be obtained for $x_{\phi-1,\psi}$. Minimising the $\frac{\partial^2 x}{\partial \phi^2}$ discretisation truncation error using these expansions gives an equation whose weights depend on the C's instead of equation (2.35)

$$\begin{aligned} \frac{\partial^2 x}{\partial \phi^2} &= \frac{2}{l^2 (C^2_{\phi-1,\psi} + C_{\phi-1,\psi} C_{\phi+1,\psi})} x_{\phi-1,\psi} \\ &- \frac{2}{l^2 (C_{\phi-1,\psi} C_{\phi+1,\psi})} x_{\phi,\psi} \\ &+ \frac{2}{l^2 (C^2_{\phi+1,\psi} + C_{\phi-1,\psi} C_{\phi+1,\psi})} x_{\phi+1,\psi}. \end{aligned} \tag{4.2}$$

This can be shown to induce no node movement from the difference between the fine and coarse spacings, thereby solving the problem of fine nodes diffusing into coarse areas. The equation for the mixed derivative is

$$\begin{aligned}\frac{\partial^2 x}{\partial\phi\psi} &= \frac{C_{\phi+1,\psi+1}C_{\phi-1,\psi-1} - C_{\phi+1,\psi-1}C_{\phi-1,\psi+1}}{2l^2 C_{\phi+1,\psi+1}C_{\phi+1,\psi-1}C_{\phi-1,\psi+1}C_{\phi-1,\psi-1}} x_{\phi,\psi} \\ &+ \frac{x_{\phi+1,\psi+1}}{2l^2(C_{\phi-1,\psi-1}C_{\phi+1,\psi+1} + C^2_{\phi+1,\psi+1})} \\ &- \frac{x_{\phi-1,\psi+1}}{2l^2(C_{\phi-1,\psi+1}C_{\phi+1,\psi-1} + C^2_{\phi-1,\psi+1})} \\ &- \frac{x_{\phi+1,\psi-1}}{2l^2(C_{\phi-1,\psi+1}C_{\phi+1,\psi-1} + C^2_{\phi+1,\psi-1})} \\ &+ \frac{x_{\phi-1,\psi-1}}{2l^2(C_{\phi-1,\psi-1}C_{\phi+1,\psi+1} + C^2_{\phi-1,\psi-1})}. \end{aligned} \tag{4.3}$$

The new α's, β's and γ's are related to the previous central difference values $\alpha_{\phi,\psi}$, $\beta_{\phi,\psi}$ and $\gamma_{\phi,\psi}$ (equations (2.38), (2.39), (2.40)) by

$$\alpha = \frac{4\alpha_{\phi,\psi}}{(C_{\phi,\psi+1} + C_{\phi,\psi-1})^2} \tag{4.4}$$

$$\beta = \frac{4\beta_{\phi,\psi}}{(C_{\phi+1,\psi} + C_{\phi-1,\psi})(C_{\phi,\psi+1} + C_{\phi,\psi-1})} \tag{4.5}$$

$$\gamma = \frac{4\gamma_{\phi,\psi}}{(C_{\phi+1,\psi} + C_{\phi-1,\psi})^2}. \tag{4.6}$$

Substituting these into the inverse equation results in the following equation for the new relaxed x nodal value

$$\begin{aligned} x_{\phi,\psi} &= \frac{\alpha_{\phi,\psi}(\frac{x_{\phi-1,\psi}}{C_{\phi-1,\psi}} + \frac{x_{\phi+1,\psi}}{C_{\phi+1,\psi}})}{D(C_{\phi,\psi+1} + C_{\phi,\psi-1})^2(C_{\phi-1,\psi} + C_{\phi+1,\psi})} \\ &+ \frac{\gamma_{\phi,\psi}(\frac{x_{\phi,\psi-1}}{C_{\phi,\psi-1}} + \frac{x_{\phi,\psi+1}}{C_{\phi,\psi+1}})}{D(C_{\phi+1,\psi} + C_{\phi-1,\psi})^2(C_{\phi,\psi-1} + C_{\phi,\psi+1})} \\ &+ \frac{\beta_{\phi,\psi}\left(\frac{(\frac{x_{\phi-1,\psi+1}}{C_{\phi-1,\psi+1}} + \frac{x_{\phi+1,\psi-1}}{C_{\phi+1,\psi-1}})}{(C_{\phi-1,\psi+1} + C_{\phi+1,\psi-1})} - \frac{(\frac{x_{\phi+1,\psi+1}}{C_{\phi+1,\psi+1}} + \frac{x_{\phi-1,\psi-1}}{C_{\phi-1,\psi-1}})}{(C_{\phi+1,\psi+1} + C_{\phi-1,\psi-1})}\right)}{D(C_{\phi+1,\psi} + C_{\phi-1,\psi})(C_{\phi,\psi+1} + C_{\phi,\psi-1})}, \end{aligned} \tag{4.7}$$

where D is given by

$$\begin{aligned} D &= \frac{\alpha_{\phi,\psi}}{(C_{\phi,\psi+1} + C_{\phi,\psi-1})^2(C_{\phi-1,\psi}C_{\phi+1,\psi})} \\ &+ \frac{\gamma_{\phi,\psi}}{(C_{\phi+1,\psi} + C_{\phi-1,\psi})^2(C_{\phi,\psi-1}C_{\phi,\psi+1})} \\ &+ \frac{\beta_{\phi,\psi}(C_{\phi+1,\psi+1}C_{\phi-1,\psi-1} - C_{\phi+1,\psi-1}C_{\phi-1,\psi+1})}{\Pi(C_{\phi+1,\psi} + C_{\phi-1,\psi})(C_{\phi,\psi+1} + C_{\phi,\psi-1})}, \end{aligned} \tag{4.8}$$

with

$$\Pi = C_{\phi+1,\psi+1}C_{\phi+1,\psi-1}C_{\phi-1,\psi+1}C_{\phi-1,\psi-1}. \tag{4.9}$$

Notice that the l's have canceled as with the original Winslow formulae. The formulae above uses $\alpha_{\phi,\psi}$, $\beta_{\phi,\psi}$ and $\gamma_{\phi,\psi}$ as given by the original central difference formulae (2.38), (2.39) and (2.40) making it easier to convert existing Winslow-Crowley subroutines. The expression for the relaxed y value is similar and hence omitted.

This method reduces to the original formulae and is second order accurate if the nine point stencil is entirely in a fine or coarse region. However, when the nine point stencil contains both coarse and fine nodes the central difference is lost and the accuracy is formally first order. An advantage of this method is that the stencil used is always the finest that exists. The finer detail of the original mesh is fully taken into account. This also removes the concern that using the coarse stencil might move a fine node too far.

4.1.1 Adaptive mesh equipotential relaxation results

The equipotential mesh relaxation methods were tested on a randomly perturbed adaptive grid. The initial set up consisted of a square of high density with no energy or pressure. A square region $x \in (0.125, 0.375)$, $y \in (0.15, 0.4)$ of high density 2.0 is contained in a $(0, 0.5) \times (0, 0.5)$ domain with density 1.0. The calculation was run with an initial mesh of 20×20. The adaptive mesh was then generated producing finer regions around the perimeter of the square where the density changed. A random perturbation was applied to all the nodes, see Figure 4.1. The equipotential mesh relaxation methods were then tested on the perturbed grid over a varying number of iterations.

The original grid is a solution of the equipotential equations therefore the difference between the relaxed grid and the non-perturbed grid can be viewed as a measure of the error of the method. The maximum value over the nodes and the average value, calculated as the sum of the squared distances divided by the number of nodes, give an indication of maximum error squared and average squared error per node.

Two further measures of grid quality were considered for problems where the exact solution is not known [67], [51]. These consisted of a measure of orthogonality σ_{orthog} and a measure of regularity per node σ_{vol}.

Orthogonality

The orthogonality of the grid is measured by

$$\begin{aligned}
\sigma_{orthog} &= \sum_{allnodes \neq disjoint} \sigma_{i,j} \\
\sigma_{i,j} &= [\frac{(\mathbf{x_{i,j}} - \mathbf{x_{i+1,j}}).(\mathbf{x_{i,j}} - \mathbf{x_{i,j+1}})}{|\mathbf{x_{i,j}} - \mathbf{x_{i+1,j}}||\mathbf{x_{i,j}} - \mathbf{x_{i,j+1}}|}]^2 + [\frac{(\mathbf{x_{i,j}} - \mathbf{x_{i-1,j}}).(\mathbf{x_{i,j}} - \mathbf{x_{i,j+1}})}{|\mathbf{x_{i,j}} - \mathbf{x_{i-1,j}}||\mathbf{x_{i,j}} - \mathbf{x_{i,j+1}}|}]^2 \\
&+ [\frac{(\mathbf{x_{i,j}} - \mathbf{x_{i+1,j}}).(\mathbf{x_{i,j}} - \mathbf{x_{i,j-1}})}{|\mathbf{x_{i,j}} - \mathbf{x_{i+1,j}}||\mathbf{x_{i,j}} - \mathbf{x_{i,j-1}}|}]^2 + [\frac{(\mathbf{x_{i,j}} - \mathbf{x_{i-1,j}}).(\mathbf{x_{i,j}} - \mathbf{x_{i,j-1}})}{|\mathbf{x_{i,j}} - \mathbf{x_{i-1,j}}||\mathbf{x_{i,j}} - \mathbf{x_{i,j-1}}|}]^2, \quad (4.10)
\end{aligned}$$

where $\mathbf{x_{i,j}}$ represents the position vector of the node being calculated and $\mathbf{x_{i\pm1,j}}$ and $\mathbf{x_{i,j\pm1}}$ represent the position vectors of its four surrounding nodes. This quantity is then divided by the number of nodes summed over to give a measure of the grid's orthogonality per node.

Regularity

The grid's regularity is measured for each non-disjoint node as

$$\sigma_{vol} = \sum_{allnodes \neq disjoint} \frac{max(V_1, V_2, V_3, V_4) - min(V_1, V_2, V_3, V_4)}{max(V_1, V_2, V_3, V_4)}, \quad (4.11)$$

where V_1, V_2, V_3, V_4 are the four element volumes surrounding each node. This quantity is then divided by the number of nodes summed over to give a measure of the grid's regularity per node.

Convergence

The convergence of the two different methods was investigated using Gauss-Jacobi iteration. Preliminary investigations indicated that Gauss-Seidel iteration resulted in such fast relaxation that all of the Lagrangian character of the mesh was removed and the mesh was almost entirely smooth, Gauss-Seidel iteration was therefore discounted.

Both relaxation methods worked well, the resulting meshes were almost identical and both were comparable to the non-perturbed adaptive mesh. Both methods retained the fine and coarse areas and no diffusion of fine nodes into coarser meshing was observed, see Figure 4.2. The largest errors, when comparing the non-perturbed and relaxed grids, were observed around the corners of the fine meshing.

The convergence of the methods are compared in Figure 4.3 and Figure 4.4 for the angle orthogonality measure and average squared distance error respectively. Very little

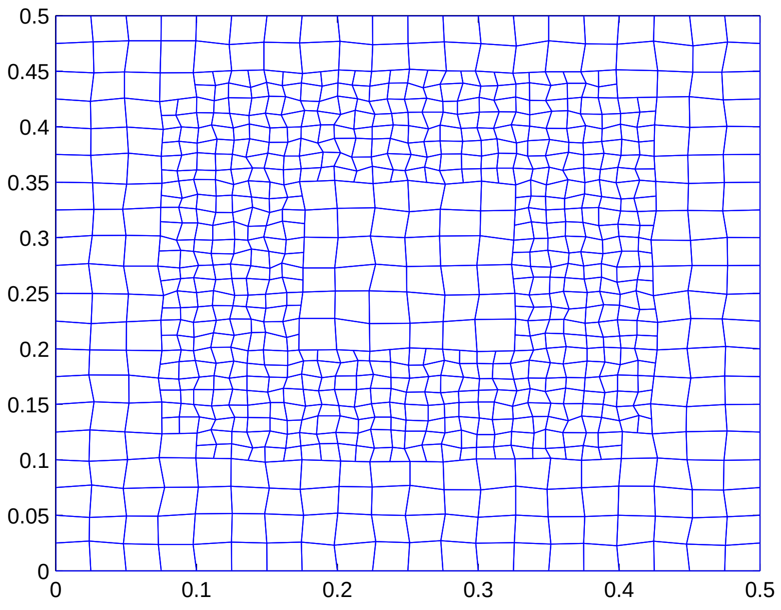

Figure 4.1: Randomly perturbed adaptive mesh for density square.

difference is observed between Method 1, the coarse stencil, and Method 2, the length weighted stencil. The gradient of a ln-normal graph approaches the asymptotic convergence rate as the number of iterations increases. This is shown for the the orthogonality measure in Figure 4.5 and the average squared distance error in Figure 4.6. The length weighted stencil, Method 2, converges slightly slower than the coarse weighted stencil.

The majority of relaxation that occurs in the test problems considered is usually around shocks. An investigation into the performance of the relaxation method when resolution transitions occur around shocks was undertaken. Using the refined square the nodes in $x < 0.175$ had their distance from 0.175 halved, forming a shock like feature shown in Figure 4.7. The different relaxation methods were then applied as before.

After 20 iterations there are areas of curvature around the resolution transitions, see Figure 4.8 and Figure 4.9, these are more pronounced for Method 1 (reverting to the coarse stencil). This should have been a one-dimensional relaxation, but the coarser meshing relaxes more due to its larger spacings than the finer meshing, resulting in curving mesh lines as the fine mesh drags them back. Method 1 uses the coarse stencil at resolution transitions and so its relaxation appears closer to that which would have been expected from a uniformly coarse mesh. Method 2 uses a length weighted stencil

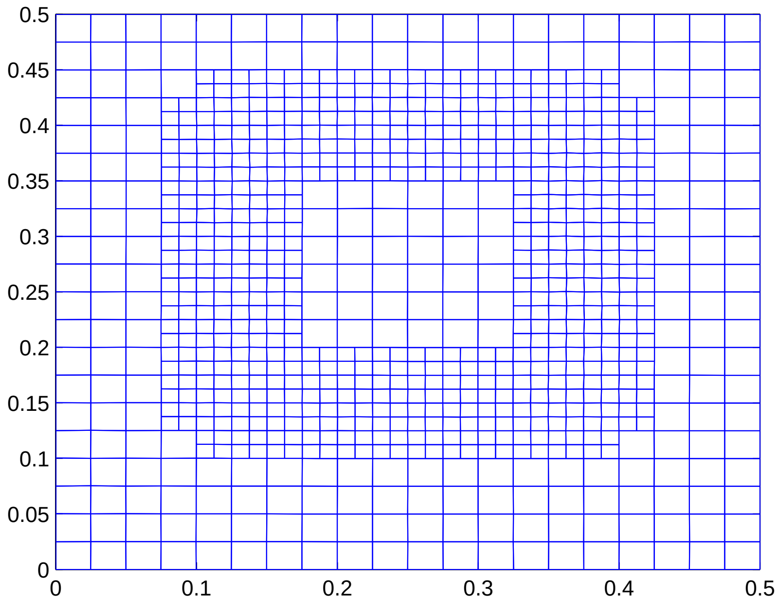

Figure 4.2: Equipotential mesh relaxation Method 2, 20 Gauss-Jacobi iterations.

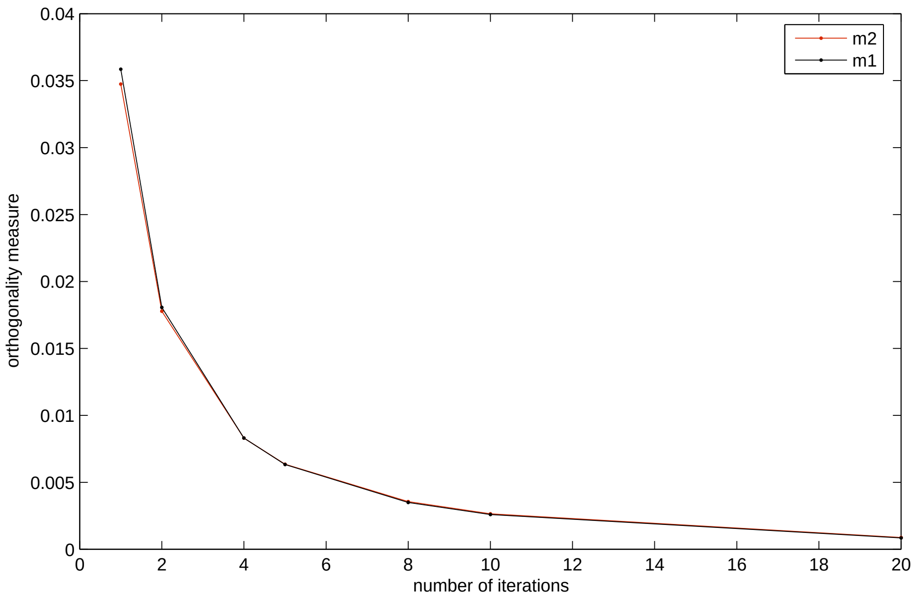

Figure 4.3: Convergence of orthogonality quality measure for different stencil methods.

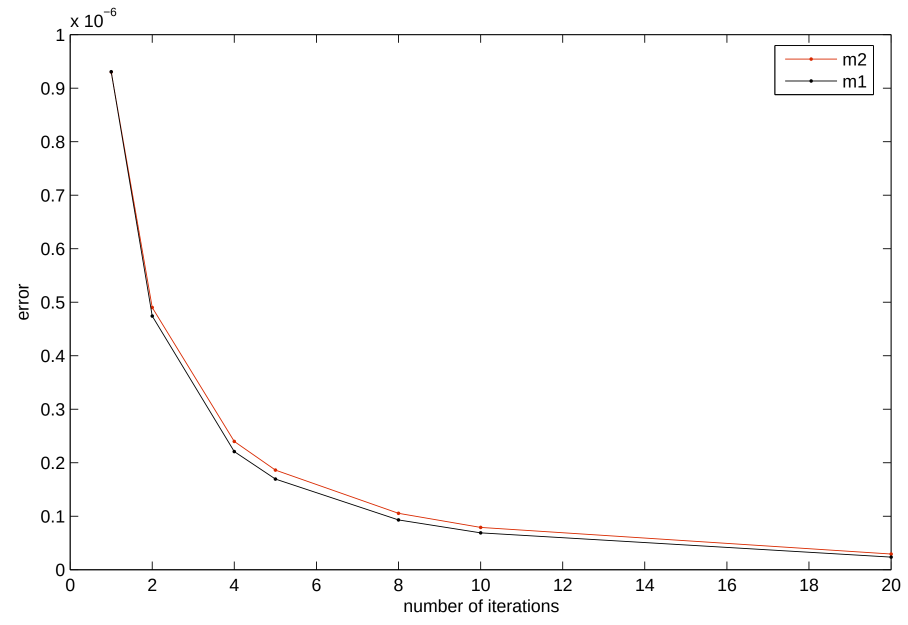

Figure 4.4: Convergence of average squared distance error for different stencil methods.

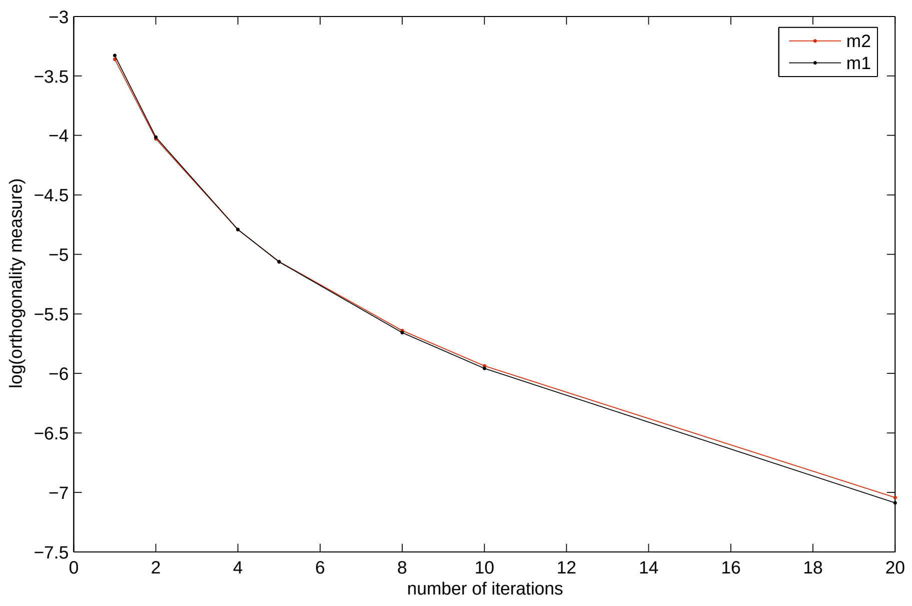

Figure 4.5: Log-normal graph of orthogonality quality measure for different stencil methods.

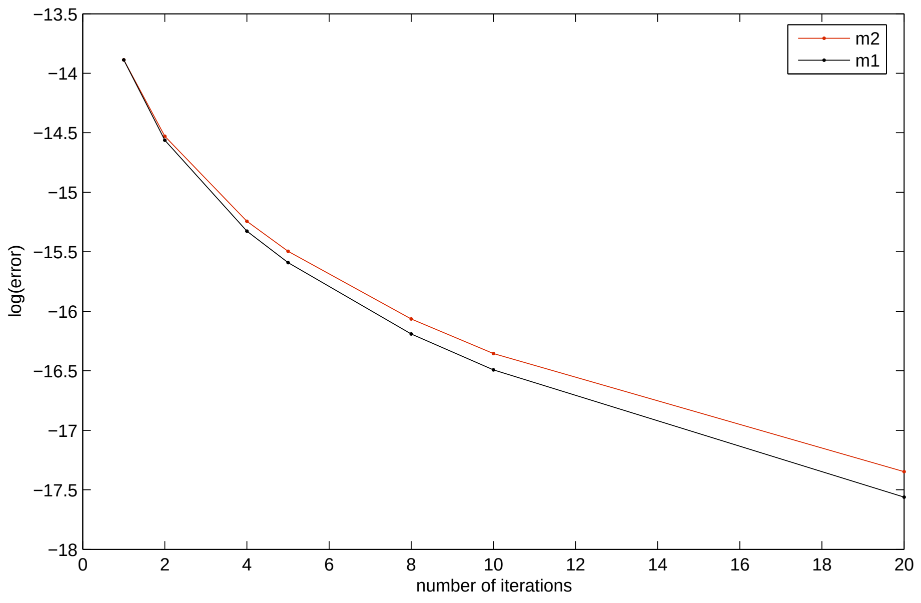

Figure 4.6: Log-normal graph of average squared distance error for different stencil methods.

and the closest nodes and therefore its relaxation is closer to that of a uniformly fine mesh relaxing.

This prompts the question "how do we want the Dynamic Mesh to relax?" The faster relaxation and hence convergence would come from emulating the relaxation of the coarser meshing, this helps to explain why Method 1 gave the better convergence for the perturbed mesh. However, the aims throughout have been to remove tangling while retaining the Lagrangian character of the mesh and to produce behaviour comparable with a uniformly fine calculation. These criteria would imply that the fine regions are most important as these encase the features of interest, we wish to relax in the manner most similar to the fine mesh, especially as this will avoid over relaxing the mesh and losing all the Lagrangian nature. Furthermore, if we consider the Dynamic Mesh as an unstructured mesh then the dynamic meshing would relax by different amounts.

It is worth noting here that by relaxing the finer meshing each refined time step Anderson *et al.* [6] bring the relaxation of the fine mesh up to the amount of the coarse mesh. This is likely to result in less curvature but as we do not implement time-refinement would lead to a greater possibility of over-relaxing the fine mesh in our approach.

The refinement in our adaptive method occurs along the whole of a shock's extent

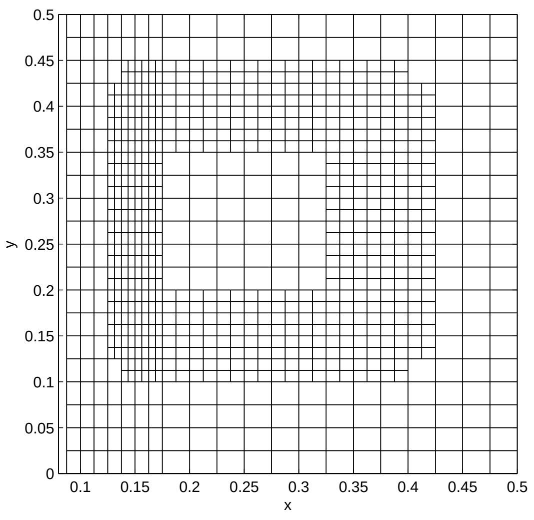

Figure 4.7: Quasi-shock adaptive mesh for density square.

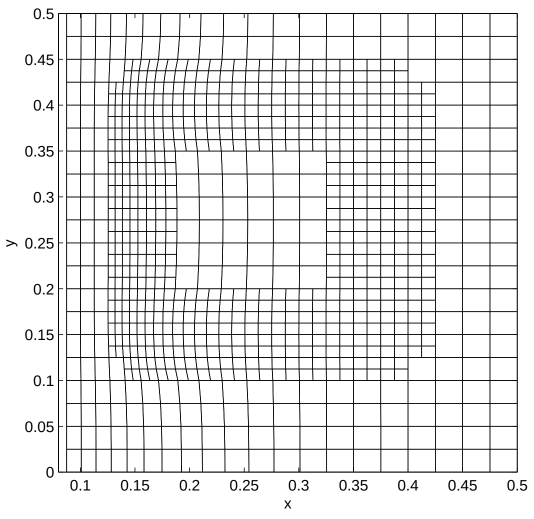

Figure 4.8: Equipotential mesh relaxation Method 1, 20 Gauss-Jacobi iterations.

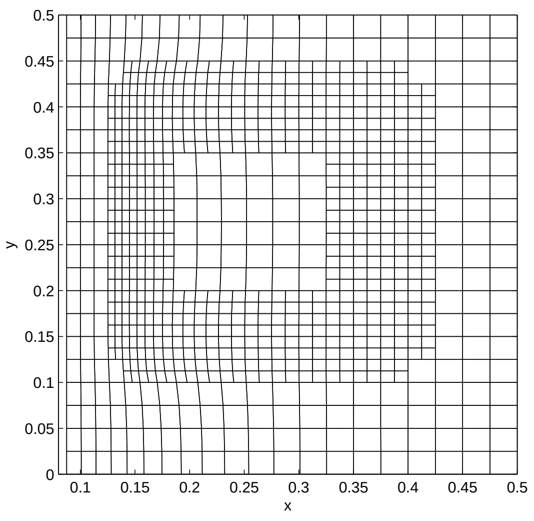

Figure 4.9: Equipotential mesh relaxation Method 2, 20 Gauss-Jacobi iterations.

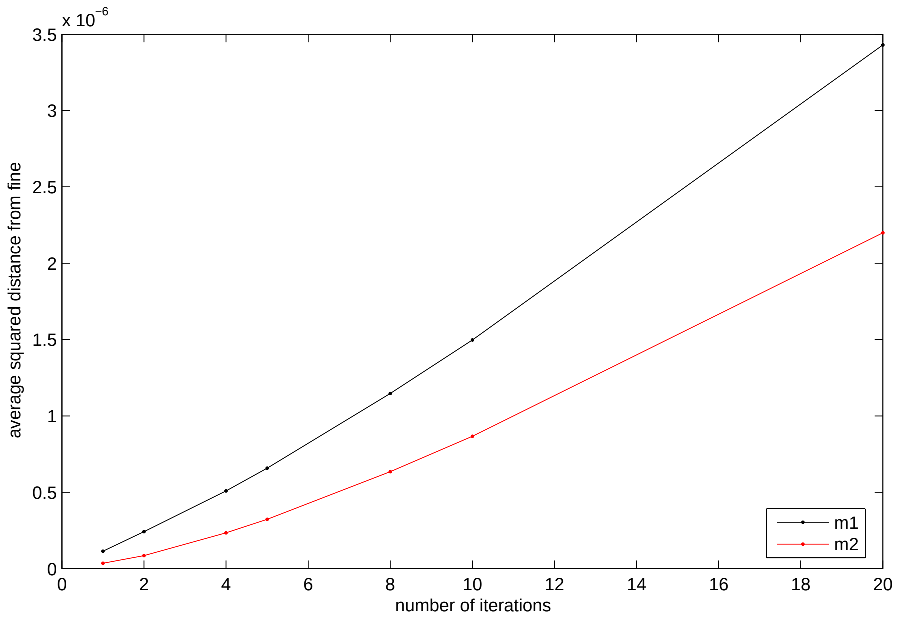

Figure 4.10: Plot of average squared distance from relaxed fine mesh for different stencil methods.

therefore this extreme case of curvature would not occur, and any marginal amount of curvature will be kept to a minimum as usually we only relax once per time step. By emulating the relaxation of the fine mesh at resolution transitions we should be relaxing the Dynamic Mesh in a way that retains as much of the Lagrangian nature of the fine mesh, does not over relax, and is consistent for the Dynamic Mesh being considered as an unstructured mesh.

Having come to these conclusions, the type of convergence test applied above (comparing to the completely relaxed mesh) is not really applicable as we do not iterate to convergence each time step and coarser meshing will relax faster. Using the quasi-shock relaxation test problem, Method 1 and Method 2 were compared to a uniformly fine mesh that had been relaxed by the same number of iterations to analyse how closely the adaptive mesh behaviour corresponds to that of the fine. Figure 4.10 shows that Method 2 is consistently closer to the relaxation of the uniformly fine mesh, the difference is reduced by a third when using Method 2. Therefore we conclude that using the nodal length weighted stencil provides a better form of adaptive mesh relaxation that is more comparable to the relaxation of a fine mesh.

The equipotential mesh relaxation methods were then tested on the two dimensional Riemann problem. Since this problem involves severe Lagrangian mesh tangling along the axis of the high density oval it represents a challenging and realistic test case. The problem was run in a Lagrangian manner until t=0.15 and then 10 iterations of adaptive mesh relaxation were applied. Both the relaxation methods worked well on this problem. The tangling along the axis was greatly reduced, while the areas of fine and coarse meshing remained distinct. Method 2 does appear to allow the grid to follow the finer mesh lines more naturally, see Figure 4.11, whereas Method 1 by reverting back to the coarse stencil restricts the curve of the line around nodes with a mixed stencil.

4.2 Advection on an isotropic adaptive mesh

Having determined new relaxed positions for the Dynamic Mesh, we need to transfer the solution to this new mesh by generalising the advection step so that it can be applied on an unstructured mesh. Advection is implemented on the Dynamic Mesh, rather than splitting the refinement levels up and treating them individually. The changes required

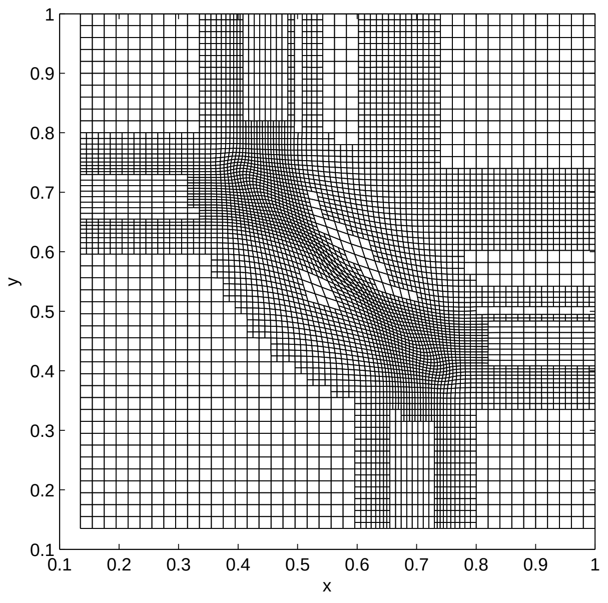

Figure 4.11: Two-dimensional Riemann problem adaptive relaxed mesh, 10 iterations Method 2.

for the advection of cell centred variables are considered first and then the momentum advection on an unstructured staggered mesh is considered.

4.2.1 Cell centred variables

The advection of cell centred variables, for example density, consists of: the calculation of overlap volumes, evaluating slopes, determining the mass fluxes through each side, evaluating the neighbouring contributions, using conservation of mass to calculate the new density value for that element. These stages must be generalised for the unstructured Dynamic Mesh.

The Dynamic Mesh is unstructured in the sense that an element may have more than four neighbouring elements. At a resolution transition a coarse element has two fine neighbours adjoining that side, see Figure 4.12 (for clarity the diagrams show orthogonal elements, in reality the elements will be distorted quadrilaterals). The evaluation of neighbouring flux contributions will need to be considered. Calculating variable slopes using volume coordinates will require three cell centred values in a line with aligning volumes so that the width perpendicular to the direction of slope does not change dramatically.

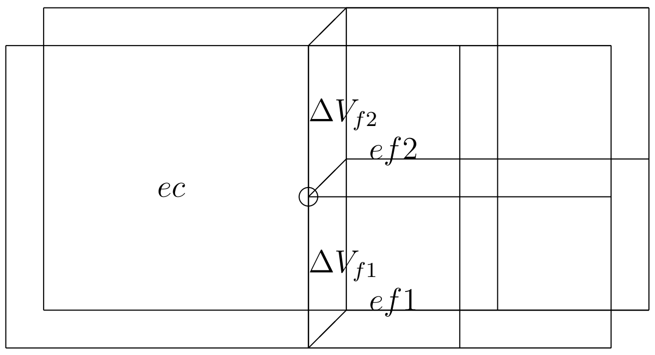

Figure 4.12: Diagram of a course-to-fine interface to show overlap volumes. The empty circle indicates the disjoint node. Mesh is shown both before and after the mesh relaxation.

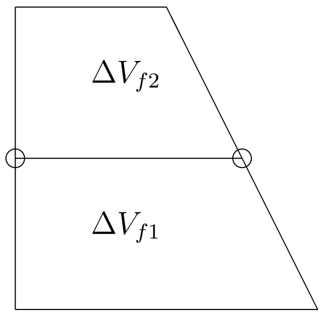

Figure 4.13: Diagram showing that the fine overlap volumes are not just half of the coarse overlap volume. The empty circle indicates the disjoint node.

When each element side has moved from its old position to a new relaxed position, it has moved through an overlap volume. The calculation of overlap volumes is unaffected by the unstructured mesh. At a resolution transition the two fine overlap volumes ΔV_{f1} and ΔV_{f2} added up will equal the coarse overlap volume ΔV_c, see Figure 4.12. Note that the fine overlap volumes are not just half the coarse overlap volume, this is illustrated in Figure 4.13.

Cell centred advection slopes

When calculating the slope of an element at a resolution transition there are two situations to consider; a coarse element next to a fine neighbour, Figure 4.14, and a fine element next to a coarse neighbour, Figure 4.15.

The only variation we wish taken into account is in the direction of the slope. The variable positions are not in the same place in coarse and fine elements, while the width in the direction orthogonal to the slope changes dramatically because the three cells do not line up, this will affect the calculation of the slope in terms of volume coordinates. Therefore the volume that aligns with the element whose slope is being calculated is used in the volume coordinates.

For the case in Figure 4.14, the slope of a coarse element next to a fine, the value of the required variable is given by conservative averaging

$$\varphi_{\alpha+1} = \frac{(\varphi_A V_A + \varphi_B V_B)}{(V_A + V_B)}. \tag{4.12}$$

Noting that the left and right slopes used for limiting only require α volumes, we need only consider neighbouring volumes for the parabolic slope. The partial volumes used when calculating the parabolic slope are then

$$\begin{aligned} V_{\alpha+1,1} &= V_{A,1} + V_{A,4} \\ V_{\alpha+1,4} &= V_{B,1} + V_{B,4}. \end{aligned} \tag{4.13}$$

For the case in Figure 4.15, the slope of a fine element next to a coarse neighbour, the value of the required variable is given by

$$\varphi_{\alpha-1} = \varphi_C. \tag{4.14}$$

This approach uses first order interpolation, this is consistent with the values being constant within each element. The total volume used when calculating the parabolic slope is then

$$V_{C,2} = V_{\alpha-1,2} + V_{\alpha-1,3}. \tag{4.15}$$

Keats and Lien [44] calculate slopes to both fine neighbours and then use a limiter to decide which to use. However, they have to prevent negative pressures and densities using a quasi-first-order approach to calculating interface fluxes. As their work uses an Eulerian mesh they do not have to use volume coordinates and the element's local coordinates for slope directions. This would be an interesting area for further work.

Cell centred advection fluxes

The four variable fluxes can now be calculated using equation (2.44) for each coarse or fine element. To calculate the new cell centred value using equation (2.43) the neighbouring fluxes are required. This is because all element influxes have been set to zero

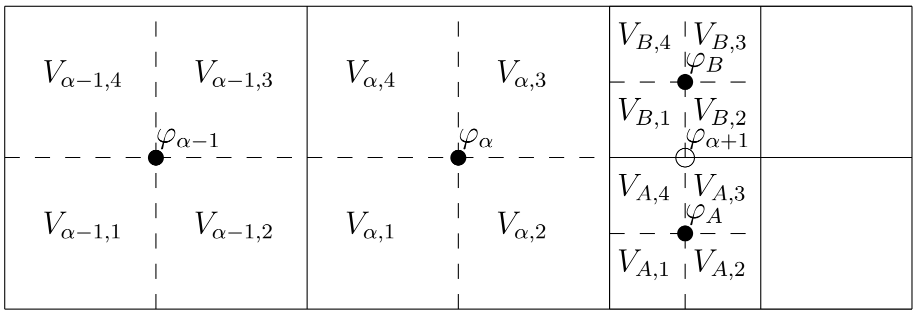

Figure 4.14: Diagram of a coarse cell with coarse and fine neighbours to show slope variables. The empty circle indicates the averaged value.

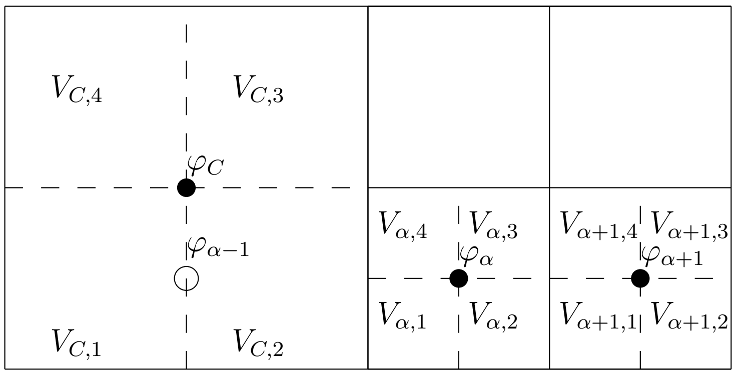

Figure 4.15: Diagram of a fine cell with coarse and fine neighbours to show slope variables. The empty circle indicates the interpolated value.

so that upwinding is guaranteed for stability. Therefore information about the neighbouring fluxes across a coarse fine interface is required.

Consider an interface between a coarse element and two fine neighbours. If the fine neighbours are outfluxing then the required neighbouring flux to the coarse element must be for conservation,

$$\Delta\varphi_c = \Delta\varphi_{f1} + \Delta\varphi_{f2}. \tag{4.16}$$

The more complicated case is if the coarse element outfluxes into the two fine elements. The neighbouring flux for each fine element must be calculated. If the variable is considered approximately constant over the coarse side (an approximation made in the original advection flux interpolation) then the required fine fluxes are given by

$$\begin{aligned} \Delta\varphi_{f1} &\approx \frac{\Delta V_{f1}\Delta\varphi_c}{\Delta V_{f1} + \Delta V_{f2}} \\ \Delta\varphi_{f2} &\approx \frac{\Delta V_{f2}\Delta\varphi_c}{\Delta V_{f1} + \Delta V_{f2}}. \end{aligned} \tag{4.17}$$

At this point it is worth investigating whether the sum of the individual fine fluxes would be the same as the whole flux. We assume there is no difference between the two fine values, $\varphi_{f1} = \varphi_{f2} = \varphi$, or the variable change over half the element, $d\varphi$, where

$$d\varphi = \varphi'(V_{\alpha,1} + V_{\alpha,2}). \tag{4.18}$$

The variable flux from the whole (the two fine elements together) is

$$\Delta\varphi_w = \Delta V_w(\varphi - \frac{d\varphi}{V_{w,1} + V_{w,2}}(V_{w,1} + V_{w,2} - \frac{1}{2}\Delta V_w)), \tag{4.19}$$

which simplifies to

$$\Delta\varphi_w = \Delta V_w(\varphi - d\varphi + \frac{d\varphi}{2(V_{w,1} + V_{w,2})}\Delta V_w). \tag{4.20}$$

The fine variable fluxes when simplified are

$$\begin{aligned} \Delta\varphi_{f1} &= \Delta V_{f1}(\varphi - d\varphi + \frac{d\varphi}{2(V_{f1,1} + V_{f1,2})}\Delta V_{f1}) \\ \Delta\varphi_{f2} &= \Delta V_{f2}(\varphi - d\varphi + \frac{d\varphi}{2(V_{f2,1} + V_{f2,2})}\Delta V_{f2}). \end{aligned} \tag{4.21}$$

Then their total is

$$\Delta\varphi_{f1} + \Delta\varphi_{f2} = \Delta V_w\left[\varphi - d\varphi + \frac{d\varphi}{2\Delta V_w}\left[\frac{\Delta V_{f1}^2}{V_{f1,1} + V_{f1,2}} + \frac{\Delta V_{f2}^2}{V_{f2,1} + V_{f2,2}}\right]\right], \tag{4.22}$$

which is not generally equal to $\Delta\varphi_w$. Simply splitting the domain into more elements may give different answers because the equation is not linear in the overlap volumes.

An extensive perturbation study was undertaken and the difference between the two formulae was found to be of lower order than the error due to assuming the variable is constant over the side (an approximation in the original scheme) and the error involved in equation (2.44). In reality the fine velocity and slope values will be different so that the sum of the fine fluxes will not equal the coarse flux anyway. In this case we want the fine elements to behave individually and not use the "whole" formula. Therefore the individual flux formulae (4.16), (4.17) are used as the non-linear difference is insignificant.

A later version of the method may wish to retain the greater accuracy of the fine data by always keeping fine fluxes and setting the coarse to 0. Conservation could then still be ensured by calculating the coarse flux as the sum of the two fine. However, this would remove the upwinding obtained by setting influxes to zero and so a study into the stability implications would be required.

The programming is greatly simplified if the slopes and fluxes affected by coarse fine interfaces are evaluated during a loop through the disjoint nodes, stored, then called upon when required in the main loops.

Similar changes are required to make the specific internal energy advection compatible with an adaptive mesh. The only difference is the use of mass rather than volume. The ratio calculation for fine flux neighbours must now be with respect to mass not volume.

4.2.2 Momentum advection

As discussed earlier, unstructured meshes prove particularly problematic for momentum advection. Although it may be possible to advect disjoint node momentum using triple slopes as outlined in [13], it is more consistent with the Lagrangian step to view disjoint nodes as non-dynamic points and ensure that their velocities are slaved. This implies that only the momentum advection of non-disjoint nodes needs to be considered.

Following the previous approach to momentum advection we must define a dual mesh, which must be consistent with the usual dual mesh in regions that are completely coarse or completely fine. It remains to define the dual mesh around resolution transitions and this is discussed in detail. The nodal masses, nodal mass weights and nodal mass

fluxes will be highly dependent on the dual mesh. Neighbour fluxes will also be needed and since the dual mesh sides may not line up all nodes fluxed to must be found and considered carefully.
Further questions arise when calculating slopes, which are required only for non-disjoint nodes. Calculating dual mesh slopes at resolution transitions is very difficult. This prompts the question "do we need to use the same dual mesh for slopes?" A measure of the slope for that node is all that is required, therefore a new slope dual mesh could be defined. Note that due to the use of nodal mass coordinates the difference between the widths perpendicular to the slope and the width of the node's original dual cell must be taken into account. Finally momentum fluxes on the dual mesh must be calculated and neighbouring momentum fluxes must also be considered. These areas are discussed further in the following sections.

The dual mesh

Following the original approach a dual mesh is defined so that the velocities can be considered as cell centred values of the dual cells. In areas of completely fine or completely coarse meshing the dual mesh is defined through the cell centres and edge midpoints as usual.
Around resolution transitions on the Dynamic Mesh we wish to ensure that the dual mesh lines follow the same directions as the slopes. Since disjoint nodes are non-dynamic points any nodal mass that would have been gathered to the disjoint node must be redistributed to the non-disjoint nodes. It is easiest if this is done before the dual mesh is defined. Therefore the disjoint nodes are not included in the dual mesh, no disjoint node dual cells exist and the dual mesh lines run straight through the disjoint nodes, see Figure 4.16.
For the usual dual mesh each surrounding element contributes a quarter of its mass to the node, see equation (2.51). Recall that each element mass contribution corresponds to one of the node's four weights, used in the slope calculations instead of partial volumes. Coarse elements always give a quarter of their mass to each node. Consider the fine element in Figure 4.17 and the node n. Nodes n_1 and n_2 also have quarters of that element's mass associated with them. However if n_1 or n_2 are disjoint nodes then those masses must be redistributed to the non-disjoint node n. Therefore the node n's second

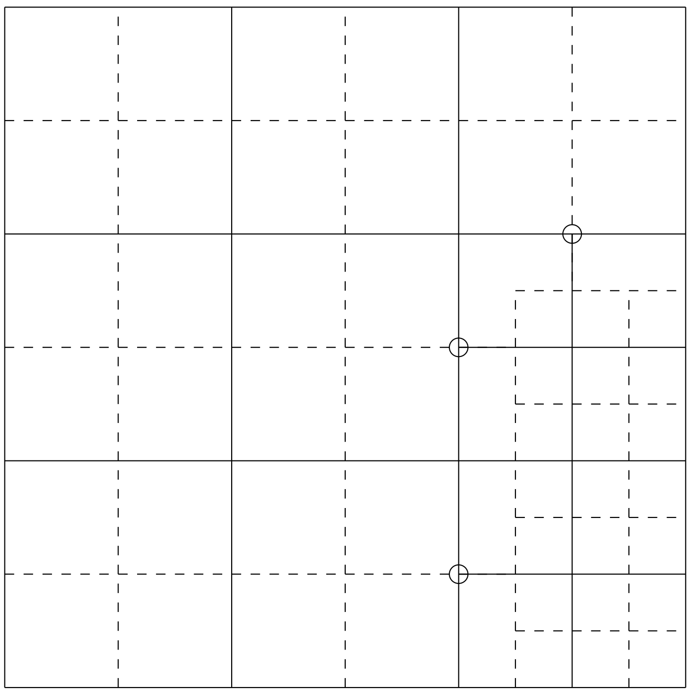

Figure 4.16: Diagram of the chosen dual mesh. Dotted lines are the dual mesh, empty circles indicate disjoint nodes.

nodal weight (second quarter) is

$$W_{n,2} = \begin{cases} \frac{1}{4}M_e & \text{if } n_1 \neq \text{djn and } n_2 \neq \text{djn} \\ \frac{2}{4}M_e & \text{if } n_1 \neq \text{djn and } n_2 = \text{djn} \\ \frac{2}{4}M_e & \text{if } n_1 = \text{djn and } n_2 \neq \text{djn} \\ \frac{3}{4}M_e & \text{if } n_1 = \text{djn and } n_2 = \text{djn} \end{cases}, \tag{4.23}$$

where M_e is the element mass. The total mass is then the sum of the weights for each node. This corresponds exactly to the dual mesh shown in Figure 4.16.

Nodal mass fluxes

Recall that in the original advection scheme the total nodal mass flux passed between a node and its neighbouring node across a dual cell side, that dual cell side was constructed as the sum of two dual mesh lines each line lying along the side of one of the nodal mass weights (or element mass contributions). Thus the total nodal mass flux was calculated as the sum of two nodal mass fluxes $dmn_4 = dmn_{44} + dmn_{41}$.

With the Dynamic Mesh it is easiest if we consider the nodal mass flux associated with each of these dual mesh lines separately, i.e. dmn_{44}. The nodal mass flux associated

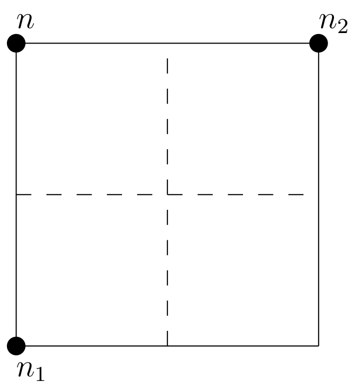

Figure 4.17: Diagram of the element showing quarter divisions.

with the original type of dual mesh line was calculated as the average of the element mass fluxes divided by two because the dual mesh line only ran half way through the element, see equation (2.49). The dual mesh of the Dynamic Mesh now has two other types of dual mesh line. The dual mesh line that passes straight through a disjoint node lies along one of the element mass fluxes but is only half as long. Referring to Figure 4.18 showing a refinement interface, f_a the nodal mass flux associated with this line is given by

$$f_a = dmn_{1,2} = \frac{1}{2}dme_{2,1}. \tag{4.24}$$

The dual mesh line not passing through the disjoint node is similar to the usual dual mesh line except that it cuts the whole of the element not just half, therefore the associated nodal mass flux is

$$f_b = dmn_{2,2} = \frac{1}{2}(dme_{2,2} + dme_{2,4}). \tag{4.25}$$

For the case of a refinement corner, shown in Figure 4.19, d_1 and d_2 represent diagonal fluxes to a diagonal neighbour dual cell and are calculated as

$$\begin{aligned} d_1 &= diag1_{n,2} = \frac{1}{4}(dme_{2,1} + dme_{2,3}) \\ d_2 &= diag2_{n,2} = \frac{1}{4}(dme_{2,2} + dme_{2,4}). \end{aligned} \tag{4.26}$$

For clarity the above formulae have been expressed in terms of the mass fluxes of the particular element e_2. In reality and in equation (2.49) the fluxes of the neighbouring elements e_1, e_4 and e_3 are used, this is so that influxes and outfluxes are not mixed in the same formulae. The dual mesh line nodal mass fluxes can be combined at the end exactly, for example the nodal mass flux for node n across side 2 is $dmn_2 = dmn_{22} + dmn_{23}$.

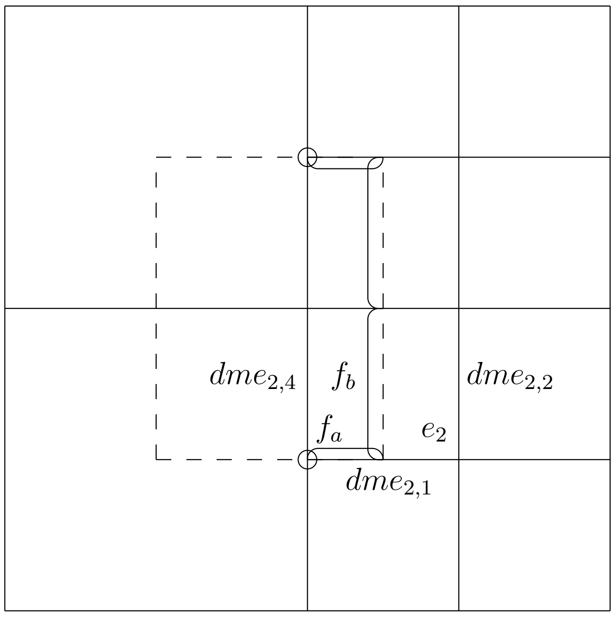

Figure 4.18: Diagram of the chosen dual mesh around an interface showing the mass fluxes.

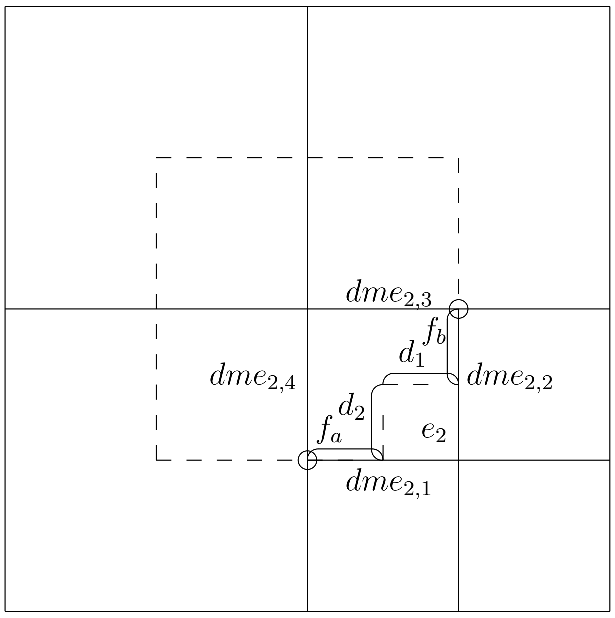

Figure 4.19: Diagram of the chosen dual mesh around a corner showing the mass fluxes.

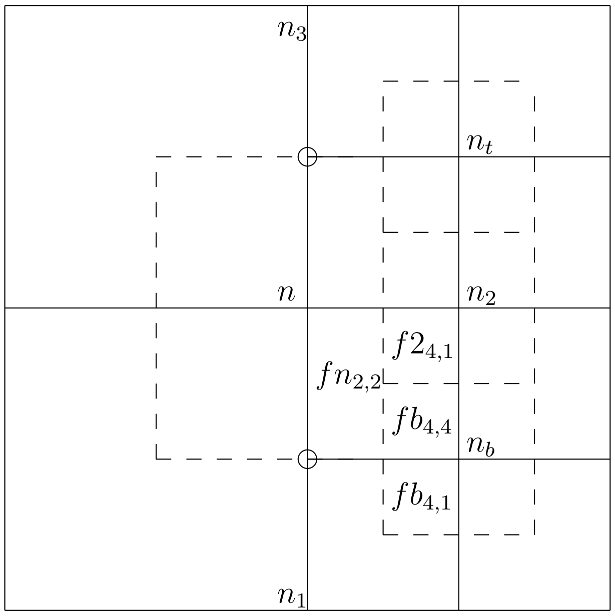

Figure 4.20: Diagram of the neighbour nodes, n's, and the neighbour mass fluxes, f's, for an interface.

Neighbouring nodal mass fluxes

To be able to calculate the post advection nodal mass we require the contributions from the neighbours. Therefore we must consider between which nodes fluxes are passed and the value of the neighbouring nodal mass fluxes.

First consider the case at a refinement interface (or two interfaces meeting), shown in Figure 4.20. The f's represent the nodal mass fluxes, let $fnghb(node, i)$ represent the required neighbour fluxes of the node with i going from 1 to 4. The neighbours of n are n_1 to n_4, n_b and n_t, note that disjoint nodes do not count as they have no dual cells. The fluxes from n_2, n_b and n_t all make up the neighbouring flux corresponding to dmn_2 therefore we wish to combine them into one neighbour flux $fnghb(n, 2)$ such that

$$fnghb(n, 2) = dmn2_{4,1} + dmn2_{4,4} + dmnb_{4,4} + dmnt_{4,1}. \tag{4.27}$$

Note for brevity that $dmnb_{4,4} = fb_{4,4}$ in the diagram and so on for the other dual mesh line nodal mass fluxes. The first two terms in this equation are the usual neighbour flux. Calculating the other neighbour fluxes requires us to know the ratio that the coarse fluxes are split by when passed to the fine dual cells, according to the derivation of the

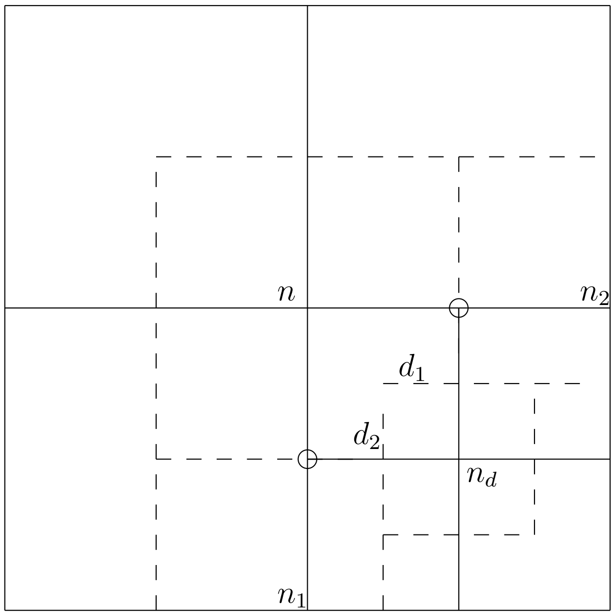

Figure 4.21: Diagram of the neighbour nodes, n's, and the neighbour mass fluxes for a corner.

nodal mass fluxes they are split equally, half is passed to each. Therefore

$$\begin{aligned} fnghb(n_2,4) &= \frac{1}{2}dmn_{2,2} + \frac{1}{2}dmn_{2,3} \\ fnghb(n_2,4) &= dmn_{2,2} + dmn_{2,3} - \frac{1}{2}dmn_{2,2} - \frac{1}{2}dmn_{2,3}, \end{aligned} \tag{4.28}$$

$$fnghb(n_b,4) = \frac{1}{2}dmn_{2,2} + \text{contribution from other node,} \tag{4.29}$$

$$fnghb(n_t,4) = \frac{1}{2}dmn_{2,3} + \text{contribution from other node.} \tag{4.30}$$

Consider the neighbour fluxes as two terms that are always used for all nodes and then perturbations caused by coarse fine interfaces. A loop through the disjoint nodes is used to calculate and store the perturbations, which are for example

$$\begin{aligned} fnghb(n_2,4) &= dmn_{2,2} + dmn_{2,3} + pert, \\ pert &= -\frac{1}{2}dmn_{2,2} - \frac{1}{2}dmn_{2,3}. \end{aligned} \tag{4.31}$$

At a refinement corner, Figure 4.21, there are also diagonal flux neighbours $dnghb1(node,i)$ and $dnghb2(node,i)$ representing the 2 possible diagonal neighbour fluxes of each i

quarter

$$\begin{aligned}
dnghb1(n,2) &= dmnd_{3,4} \\
dnghb2(n,2) &= dmnd_{4,4} \\
fnghb(nd,3) &= diag1_{n,2} + \text{contribution from other node} \\
fnghb(nd,4) &= diag2_{n,2} + \text{contribution from other node.} \qquad (4.32)
\end{aligned}$$

The $fnghb$ expressions above can be considered as the perturbations due to the disjoint node.

Using the perturbation approach means that the changes resulting from minus the nodes outflux plus the neighbouring fluxes are for side 2 with neighbouring node nghb,

$$\begin{aligned}
\Delta M_{n,2} &= -dmn_{2,2} - dmn_{2,3} + dmnghb_{4,1} + dmnghb_{4,4} + pert \qquad \text{nghb} \neq \text{djn} \\
\Delta M_{n,2} &= -dmn_{2,2} - dmn_{2,3} + pert \qquad \text{nghb=djn.} \qquad (4.33)
\end{aligned}$$

Similarly for the diagonal fluxes in quarter 2,

$$\begin{aligned}
\Delta diag1_{n,2} &= -diag1_{n,2} + dnghb1(n,2) \\
\Delta diag2_{n,2} &= -diag2_{n,2} + dnghb2(n,2). \qquad (4.34)
\end{aligned}$$

Finally the post advection nodal mass is calculated using

$$M_n^+ = M_n^- + \sum_{i=1}^{4} \Delta M_{n,i} + \sum_{i=1}^{4} \Delta diag1_{n,i} + \sum_{i=1}^{4} \Delta diag2_{n,i}. \qquad (4.35)$$

4.2.3 Slopes

When calculating velocity slopes on the dual mesh, fine nodes that are joined to disjoint nodes in the direction orthogonal to the interface have no suitable neighbouring value because there are no disjoint node dual cells. Furthermore, calculating slopes for coarse nodes at resolution transitions is also very difficult because the dual cell width perpendicular to the direction of slope can change not only between dual cells but also within the dual cell itself.

All we need is a measure of slope for the velocity variables: the dual mesh does not have to be the same as long as any dual cell width differences perpendicular to the direction of the slope are taken into account. Therefore new slope dual meshes are defined.

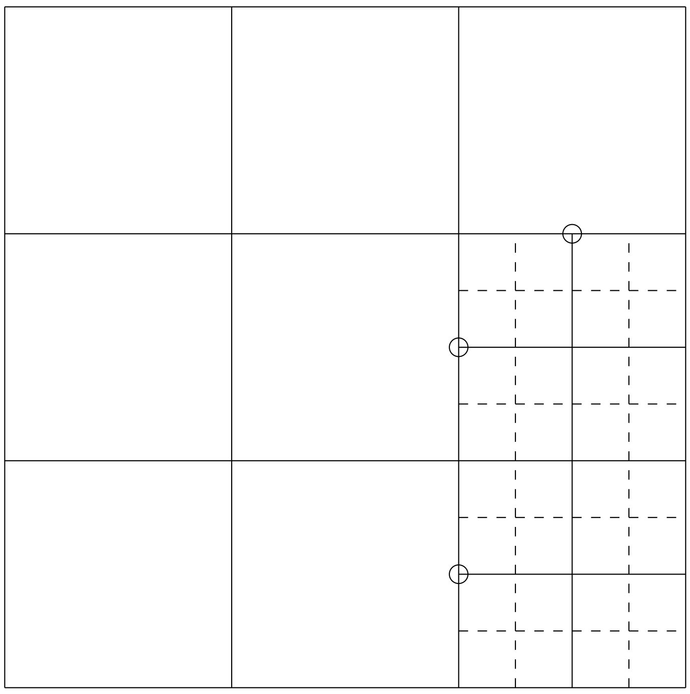

Figure 4.22: Diagram of the dual mesh used for slopes in fine regions. Dotted lines are the dual mesh, empty circles indicate disjoint nodes.

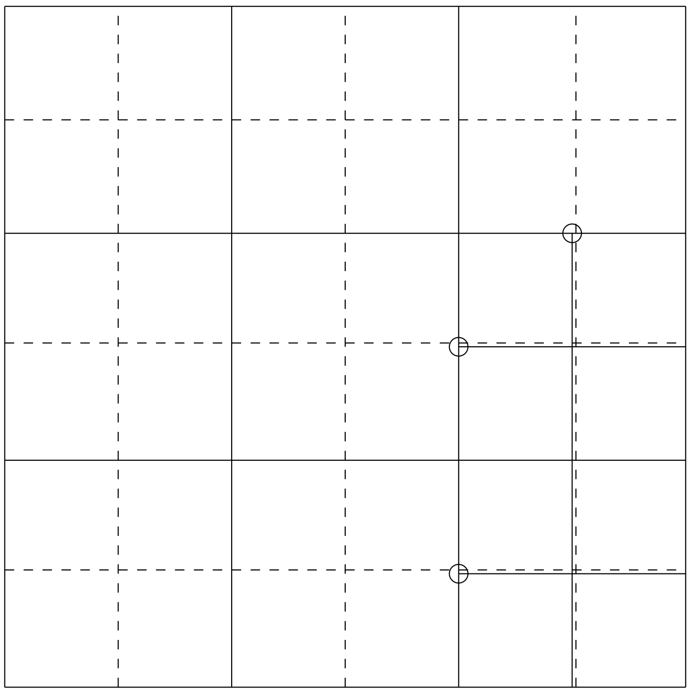

Figure 4.23: Diagram of the coarse dual mesh used for slopes around corners and interfaces. Dotted lines are the dual mesh, empty circles indicate disjoint nodes.

Slopes for nodes in the fine areas of meshing are calculated using a different dual mesh that includes dual cells around disjoint nodes, see Figure 4.22. While for coarse nodes at resolution transitions a slope dual mesh is defined using the coarse nodal stencil, see Figure 4.23. For nodes inside coarse areas using the coarse stencil is the same as using the normal dual mesh since all nodes are coarse.

Note that the nodal weights used may not be those calculated previously for the dual mesh. For the coarse stencil dual mesh they will either be a quarter of the coarse element's mass or the whole fine element's mass. For the fine dual mesh they will be a quarter of the fine element's mass.

Further work into slope limiting may be required in the future. Limiting on the coarse stencil is between the coarse nodes, which are not necessarily the nodes fluxed to, thus it may be possible for monotonicity to be violated. However monotonic limiting may always be a problem because of the unstructured nature of the mesh.

Momentum fluxes

It is easier to calculate the momentum fluxes associated with each dual mesh line separately. This is equivalent to considering the nodal weight quadrants and the two momentum fluxes associated with the two dual mesh lines. The velocity slopes are calculated using nodal mass coordinates so we require the width of the dual cell perpendicular to the direction of slope to be the same as the slope width.

For an ordinary quadrant within the same refinement area the slope was calculated over the whole dual cell, we require it over half the dual cell (i.e. the quadrant) so we take a ratio of the slope. The two momentum fluxes from quadrant 2, for the u component of velocity, that are passed to nodes n_2 and n_1 respectively are

$$\Delta un_{2,2} = -dmn_{2,2}\left(u + u_\eta\left(\frac{W_{n,3}+W_{n,2}}{W_{n,2}}\right)\left(W_{n,2} - \frac{1}{2}dmn_{2,2})\right)\right), \tag{4.36}$$

$$\Delta un_{1,2} = -dmn_{1,2}\left(u - u_\xi\left(\frac{W_{n,1}+W_{n,2}}{W_{n,2}}\right)\left(W_{n,2} - \frac{1}{2}dmn_{1,2})\right)\right), \tag{4.37}$$

where the minus signs appear because these are outfluxes.

In the case of an interface shown in Figures 4.24 and 4.25 the slope weights are coarse, Wc, and these are not equal to the flux weights $W_{n,2}$ and $W_{n,3}$. In Figure 4.24, $k1 = \Delta un_{1,2}$ lies along the dual mesh line cutting the disjoint node, the slope width was calculated over $Wc_{n,1} + Wc_{n,2}$ but the required width is only $W_{n,2}$ and this gives the

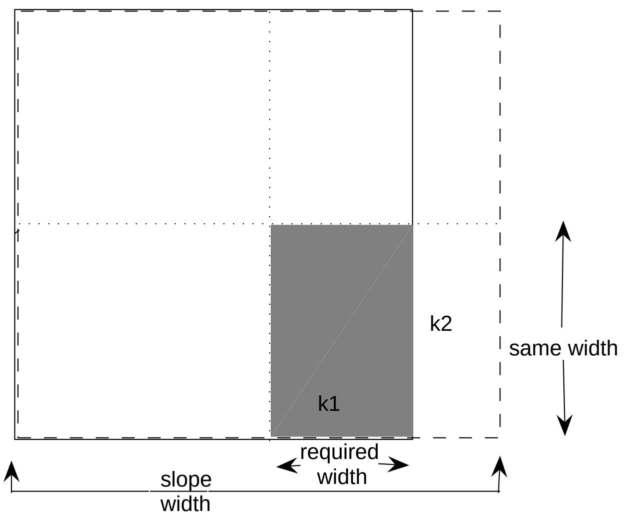

Figure 4.24: Calculation of momentum flux k_1 for an interface, dotted boundary is the coarse dual cell, line boundary is the flux dual cell.

required ratios. The momentum flux from this quadrant associated with the dual mesh line cutting the disjoint node is thus

$$\Delta un_{1,2} = -dmn_{1,2}\left(u - u_\xi\left(\frac{Wc_{n,1}+Wc_{n,2}}{W_{n,2}}\right)\left(W_{n,2}-\frac{1}{2}dmn_{1,2})\right)\right). \tag{4.38}$$

In Figure 4.25, $k2 = \Delta un_{2,2}$ lies along the dual mesh line not cutting the disjoint node, the slope width was calculated over $Wc_{n,3}+Wc_{n,2}$ but the required width is only $Wc_{n,2}$ and this gives the required ratios. The interpolation to the flux dual mesh boundary does not move as far as the coarse dual mesh boundary it only moves through $W_{n,2}$. The momentum flux for this quadrant associated with the other dual mesh line is

$$\Delta un_{2,2} = -dmn_{2,2}\left(u + u_\eta\left(\frac{Wc_{n,3}+Wc_{n,2}}{Wc_{n,2}}\right)\left(W_{n,2}-\frac{1}{2}dmn_{2,2})\right)\right). \tag{4.39}$$

In the case of a corner, Figures 4.26 and 4.27, a similar process can be done for the n_1 and n_2 fluxes except that the required width is now $\frac{2}{3}W_{n,2}$ in both cases and the interpolation moves through $\frac{2}{3}W_{n,2}$. Therefore the momentum fluxes are

$$\Delta un_{2,2} = -dmn_{2,2}\left(u + u_\eta\left(\frac{Wc_{n,3}+Wc_{n,2}}{\frac{2}{3}W_{n,2}}\right)\left(\frac{2}{3}W_{n,2}-\frac{1}{2}dmn_{2,2})\right)\right), \tag{4.40}$$

$$\Delta un_{1,2} = -dmn_{1,2}\left(u - u_\xi\left(\frac{Wc_{n,1}+Wc_{n,2}}{\frac{2}{3}W_{n,2}}\right)\left(\frac{2}{3}W_{n,2}-\frac{1}{2}dmn_{1,2})\right)\right). \tag{4.41}$$

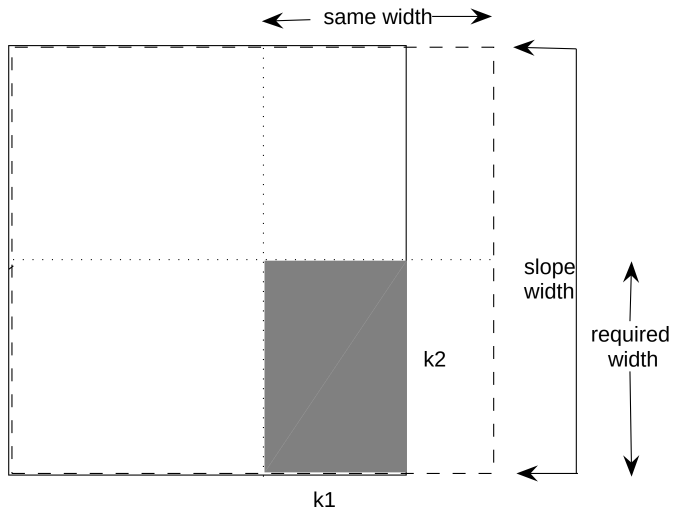

Figure 4.25: Calculation of momentum flux k_2 for an interface, dotted boundary is the coarse dual cell, line boundary is the flux dual cell.

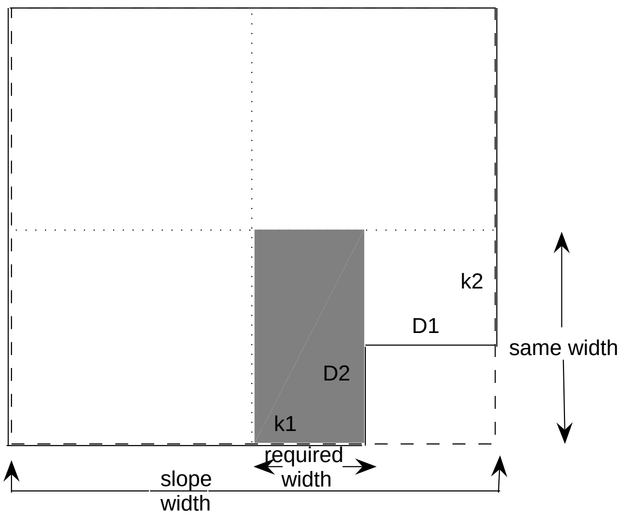

Figure 4.26: Calculation of momentum flux k_1 for a corner, dotted boundary is the coarse dual cell, line boundary is the flux dual cell.

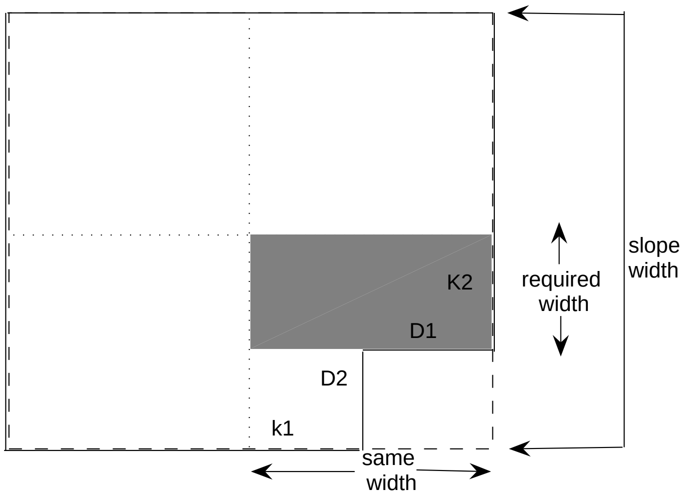

Figure 4.27: Calculation of momentum flux k_2 for a corner, dotted boundary is the coarse dual cell, line boundary is the flux dual cell.

Consider diagonal momentum flux D1 in Figure 4.28. In the original advection scheme the value is assumed constant along the flux side, u is assumed to not vary in that direction. The only interpolation required is in the direction perpendicular to the dual mesh line, following the arrow in the diagram. The required width is now $\frac{2}{3}W_{n,2}$ and the interpolation moves through $\frac{1}{3}W_{n,2}$. Therefore the total diagonal momentum flux D1 is

$$\Delta udiag1_{n,2} = -diag1_{n,2}\left(u - u_\xi\left(\frac{Wc_{n,1}+Wc_{n,2}}{\frac{2}{3}W_{n,2}}\right)\left(\frac{W_{n,2}}{3} - \frac{diag1_{n,2}}{2}\right)\right). \tag{4.42}$$

The other diagonal momentum flux is calculated as

$$\Delta udiag2_{n,2} = -diag2_{n,2}\left(u + u_\eta\left(\frac{Wc_{n,3}+Wc_{n,2}}{\frac{2}{3}W_{n,2}}\right)\left(\frac{W_{n,2}}{3} - \frac{diag2_{n,2}}{2}\right)\right). \tag{4.43}$$

This is consistent with the approach taken for interfaces.

The momentum fluxes for the dual mesh lines added together do not equal the side momentum calculated using the original scheme because of the non-linear dependence on the nodal mass fluxes. However this situation has been investigated and the difference in error was found to be negligible. It is far easier to evaluate the fluxes using the individual quadrants as this does not require information about the refinement configurations of the other quadrants.

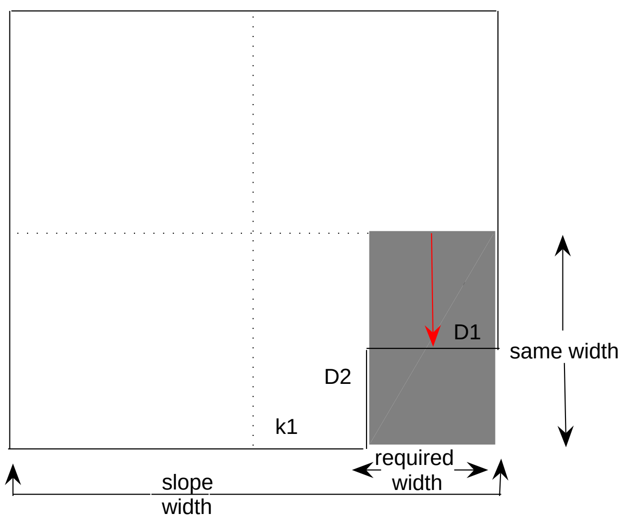

Figure 4.28: Calculation of diagonal momentum flux D_1 for a corner, dotted boundary is the coarse dual cell, line boundary is the flux dual cell.

Once the momentum fluxes have been calculated the neighbour fluxes need to be obtained. Using the ratios described for cell centred advection implies that as the nodal mass fluxes split exactly in half this should be the case for the momentum fluxes. Again it must be noted that since the momentum fluxes are non-linear in the nodal mass fluxes it is only an approximation that half the momentum coarse neighbour flux is given to each fine node. However the difference is negligible compared to the assumption that the variable value is constant over the fluxing side.

The rest of the calculation for neighbour momentum fluxes follows exactly the same procedure as for nodal mass fluxes. Then the neighbour fluxes can be added and the outfluxes subtracted to calculate the change in variable for each side $\Delta var_{n,i}$ and the diagonals $\Delta vardg1_{n,i}$ and $\Delta vardg2_{n,i}$.

Finally the new post advection velocity is found, using the pre and post advection nodal masses

$$u_n^+ = \frac{u_n^- M_n^- + \sum_{i=1}^4 \Delta var_{n,i} + \sum_{i=1}^4 \Delta vardg1_{n,i} + \sum_{i=1}^4 \Delta vardg2_{n,i}}{M_n^+}. \tag{4.44}$$

4.3 ALE results for the isotropic adaptive method

We now present results to verify the performance of the ALE adaptive method. The adaptive calculations were run with only one level of mesh refinement. The method should generalise to many refinement levels as long as the refined elements of each level are properly nested. This is a coding issue and generalisation of the code to many levels is an area that is discussed in the future work section.

4.3.1 Radial Sod problem

The radial Sod problem considered in Section 2.5.4 was run to t=0.25 using the isotropic adaptive mesh method and an initially 50×50 mesh, with a refinement parameter of 0.04 and a derefinement parameter of 0.025. One relaxation iteration of the length weighted equipotential method was used per time step.

Refined regions occurred around the rarefaction fan, contact and shock, with derefinement in between these features. The mesh in Figure 4.29 is smooth and untangled. This problem benefits greatly from the mesh relaxation as the mesh becomes highly distorted if the problem is run purely Lagrangian.

The density contours are very similar for the isotropic calculation, see Figure 4.31, and the fine calculation, see Figure 4.30, with only small differences occurring at the edges of the rarefaction fan and contact.

A very fine one dimensional approximate solution, with 5000 points, can be obtained using a Roe approximate Riemann solver, with the radial terms included as a source term as explained in [84]. This fine approximation is compared against the adaptive mesh ALE results in Figure 4.32 and the results appear as good as for the uniformly fine calculation. The 1-norm error for a calculation with N elements of volume V_i is

$$\|e\|_1 = \sum_{i=1}^{N} \left[|\rho_{amr} - \rho_s| \, V_i \right], \tag{4.45}$$

whilst the 2-norm error is

$$\|e\|_2 = \sqrt{\sum_{i=1}^{N} \left[(\rho_{amr} - \rho_s)^2 \, V_i \right]}. \tag{4.46}$$

Table 4.1 shows that the isotropic calculation has comparable errors to the uniformly fine calculation and the results are much better than for a uniformly coarse calculation.

The isotropic calculation runs 6.6 times faster than the uniformly fine calculation, which is an excellent speed up for only one level of refinement. The reduction in the number of elements is illustrated in Figure 4.33. The adaptive calculation takes slightly fewer time steps than the uniformly fine calculation, and the adaptive calculation only requires 36% of the fine calculation's number of elements over the entire calculation time.

The alternative equipotential relaxation strategy, using the coarse nodal stencil at resolution transitions, was also investigated. The slight differences in the mesh movement do cause some differences in refinement. However, the two approaches were found to produce comparable errors and required a very similar total number of elements. This might be expected because the solution would not be expected to vary very much just by relaxing the mesh differently. Although no significant difference is shown in the solution, the recommended approach is still the nodal length weighted method because the mesh relaxation is closer to that expected from the uniformly fine mesh.

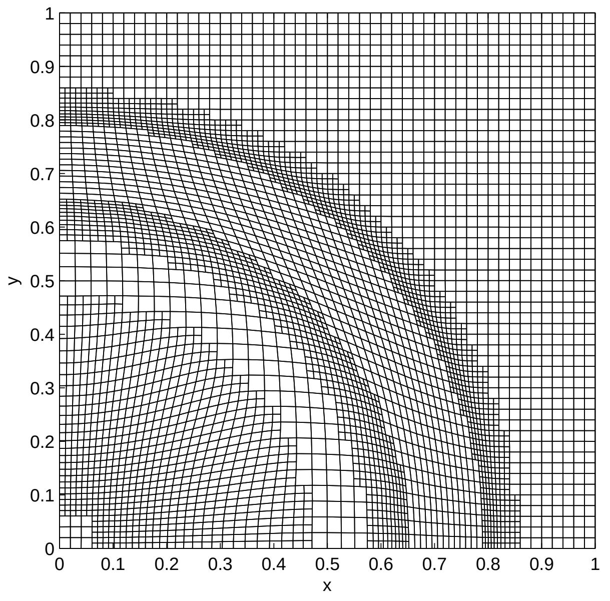

Figure 4.29: Radial Sod problem isotropic adaptive ALE mesh at t=0.25.

4.3.2 Taylor-Sedov blast wave

The Taylor-Sedov blast wave, discussed in Section 2.5.5, was run to t=0.1 using the isotropic adaptive mesh method with an initial mesh of 32×32, the refinement parameter was 0.25 and the derefinement parameter was 0.20. One iteration of the length weighted

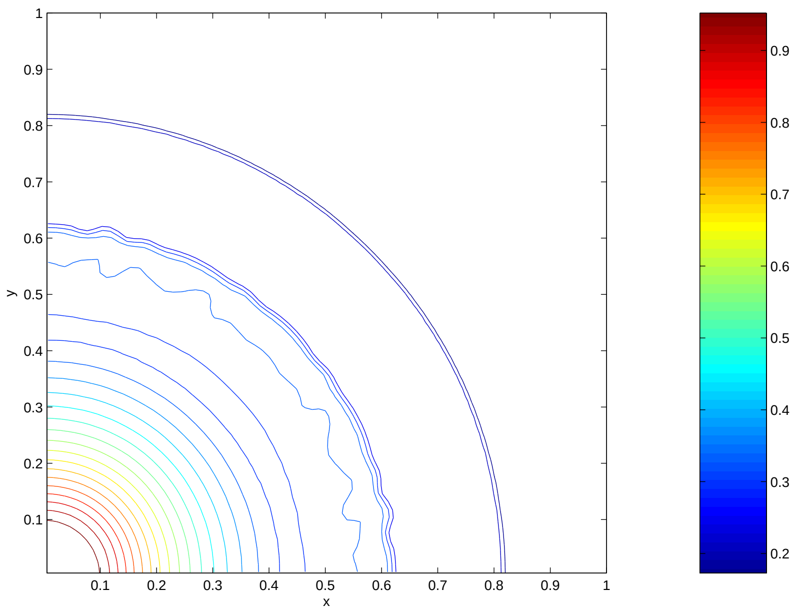

Figure 4.30: Radial Sod problem uniformly fine calculation density contours at t=0.25.

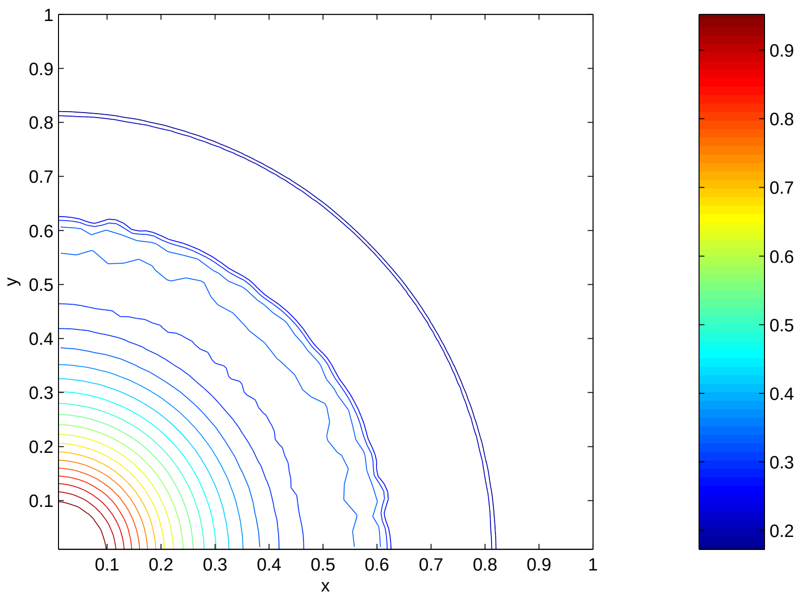

Figure 4.31: Radial Sod problem isotropic calculation density contours at t=0.25.

equipotential method was used per time step. The adaptive mesh is shown in Figure 4.34 and shows the refinement that occurs around the shock radius. There is also some refinement further in near to the axes and this is probably due to faster density variations occurring close to the axis because the energy was initialised as a square of energy on

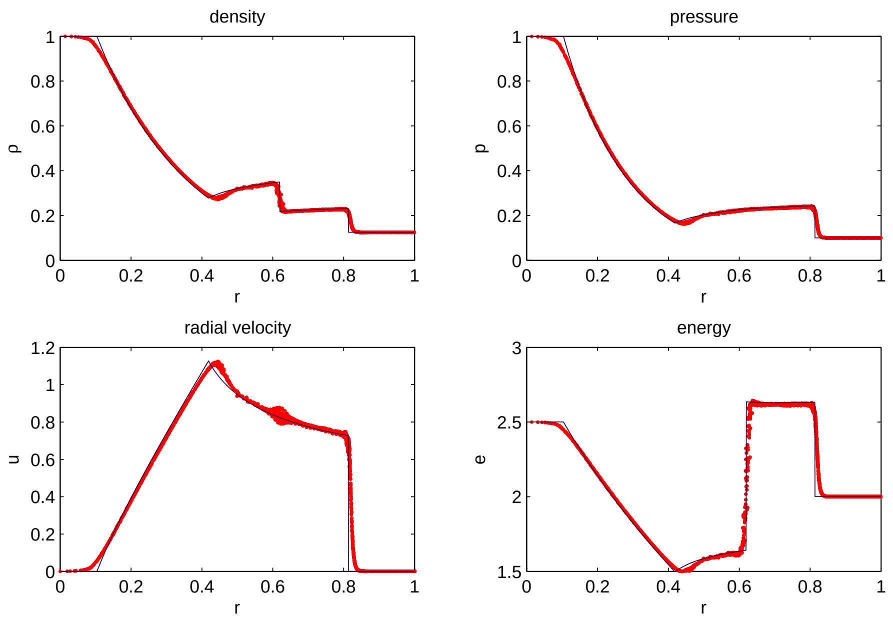

Figure 4.32: Radial Sod problem density, pressure, radial velocity and specific internal energy for isotropic solution at t=0.25.

Calculation	$\|e\|_1$	$\|e\|_2$
fine	0.0044	0.0105
isotropic	0.0046	0.0109
coarse	0.0072	0.0139

Table 4.1: Errors for fine, isotropic, and coarse radial Sod calculations t=0.25.

the quadrilateral mesh rather than a quarter circle and this has introduced some radial asymmetry.

The density contours for the adaptive calculation and the uniformly fine calculation are shown in Figure 4.36 and Figure 4.35 and compare very favourably. It is easier to compare the results when the density is plotted with respect to radius, see Figure 4.37. Again the results for the uniformly fine calculation and the adaptive are very similar and both agree well with the similarity solution used in [9]. The isotropic calculation reaches a peak value of 3.731 compared to the fine value of 3.735, the shock radius ranges from 0.604 to 0.612, which is further to the left than the fine calculation and a little less than the self similar solution radius. However, this is not considered a substantial difference considering the use of first order solution transfer. These adaptive results compare well

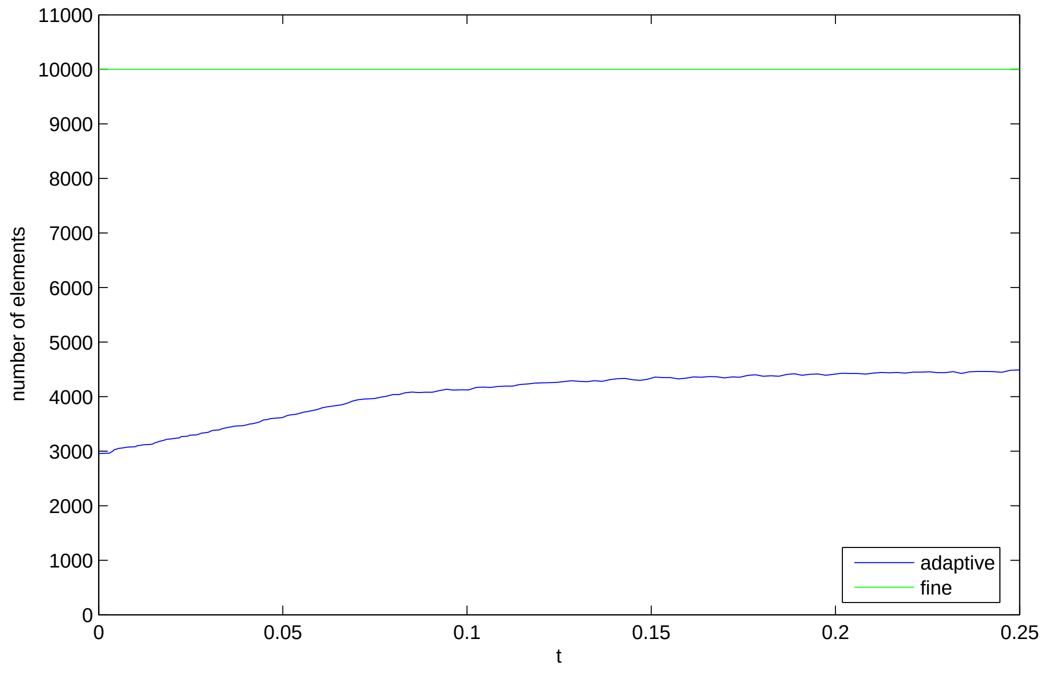

Figure 4.33: Radial Sod problem comparing the number of elements required over calculation time.

with those shown in [9].

The reduction in the number of elements achieved using the adaptive method is shown in Figure 4.38. The adaptive calculation only requires 32% of the uniformly fine calculation's total number of elements. The adaptive calculation runs more than seven times faster than the uniformly fine calculation.

4.3.3 Two-dimensional Riemann problem

The problem, see Section 2.5.2, was run with isotropic refinement on a 50×50 initial mesh. One iteration of the nodal length weighted equipotential method was employed per time step. Initially a refinement parameter of 0.04 and a derefinement parameter of 0.01 were used and these resulted in a mesh with many refined areas. Cumulative frequency graphs of the number of elements with change in density above each change in density value were plotted at various times. This suggested that a better refinement threshold was 0.15 and the derefinement value was 0.13. The difference between the two meshes, shown in Figure 4.39 and Figure 4.40, illustrates that the 0.15 parameter leads

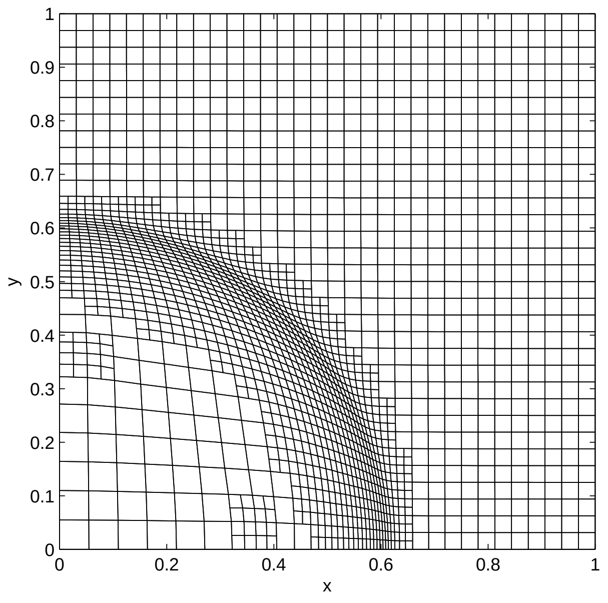

Figure 4.34: Sedov problem isotropic mesh at t=0.1.

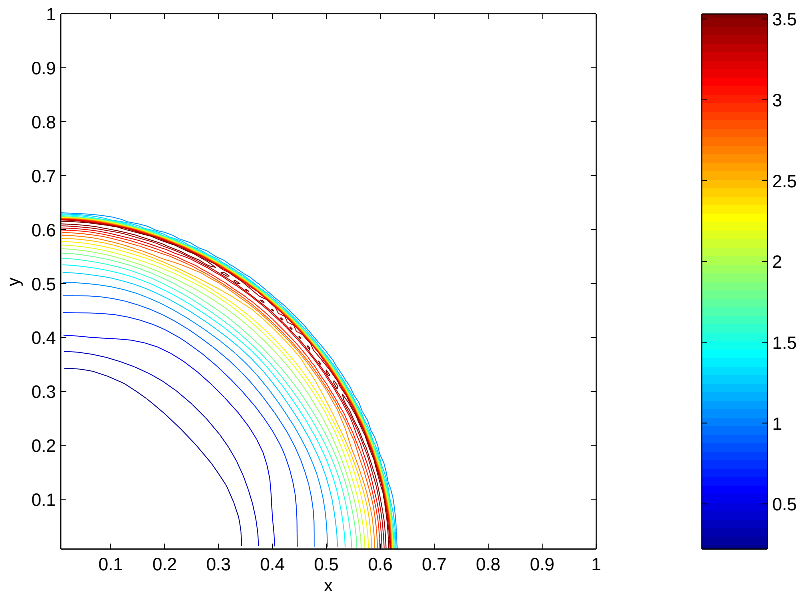

Figure 4.35: Sedov problem uniformly fine calculation density contours at t=0.1.

to a mesh that is refined around the oval circumference and the shocks but without wasteful refinement in the almost constant regions. It was decided therefore that the refinement threshold of 0.15 and the derefinement value of 0.13 should be used. With these values within the high density oval cells have derefined completely.

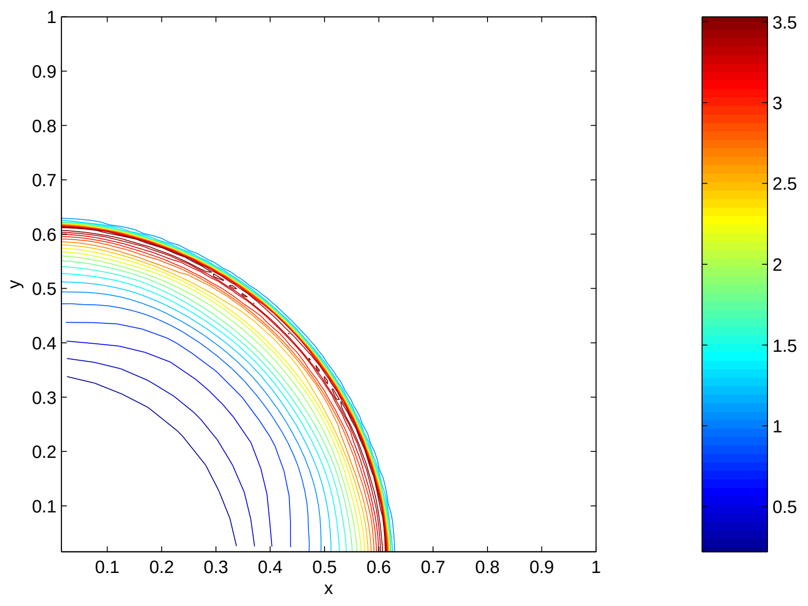

Figure 4.36: Sedov problem isotropic calculation density contours at t=0.1.

The density contours, see Figure 4.42, are comparable with those achieved from a uniformly fine calculation, see Figure 4.41. There is some loss of detail within the high density oval because this is only coarsely resolved. However, the features with significant changes in density, the shocks and oval circumference, have the resolution of the fine mesh, they are as thin and sharp as the results achieved using the uniformly fine mesh.

Figure 4.43 shows the significant reduction in elements obtained by using the adaptive method, the adaptive calculation only requires 42% of the fine calculation's number of elements over the entire calculation time. The isotropic calculation runs 5.5 times faster than the uniformly fine calculation.

These results have been achieved using first order solution transfer for the adaptive method and although they are very promising the next chapter investigates whether the results might be improved using a more accurate solution transfer method.

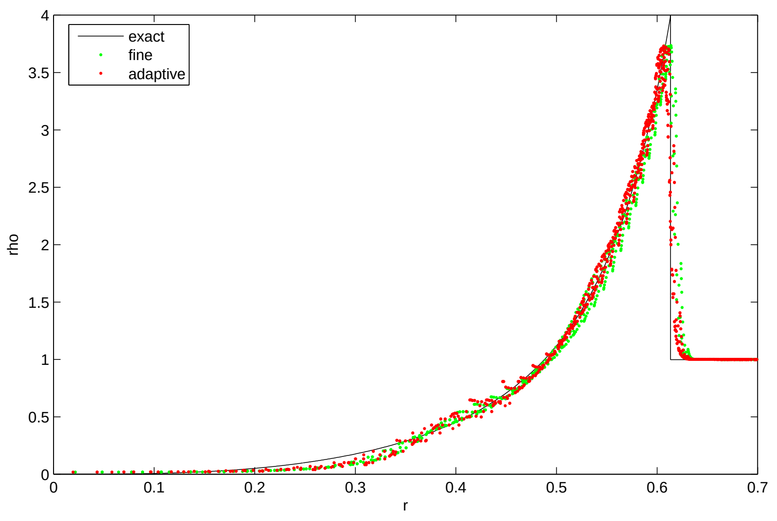

Figure 4.37: Sedov problem radial density at t=0.1.

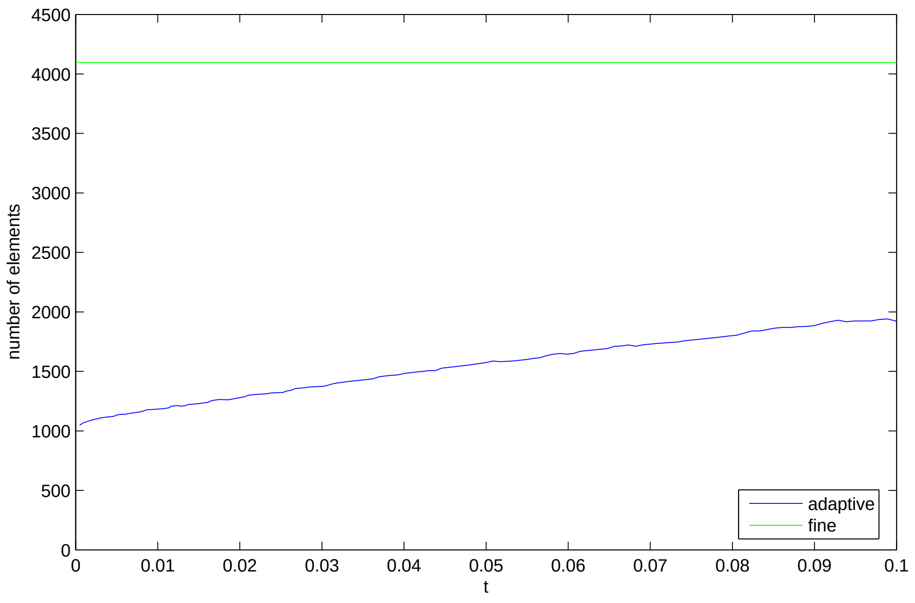

Figure 4.38: Sedov problem comparing the number of elements required over calculation time.

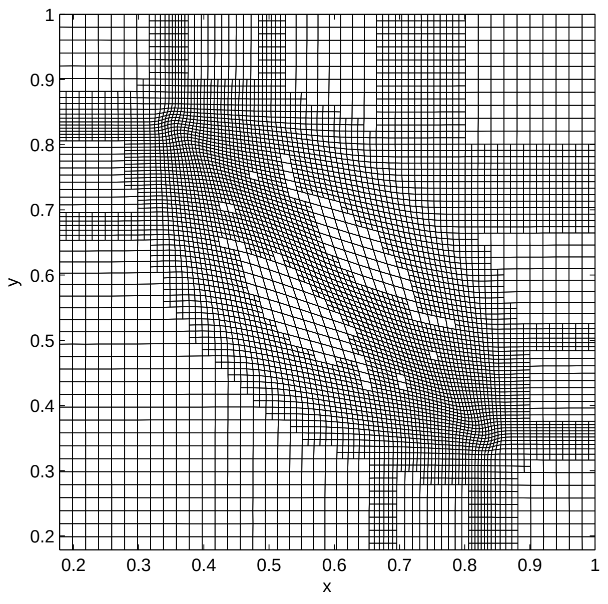

Figure 4.39: Two-dimensional Riemann problem isotropic mesh at t=0.2, refinement threshold 0.04.

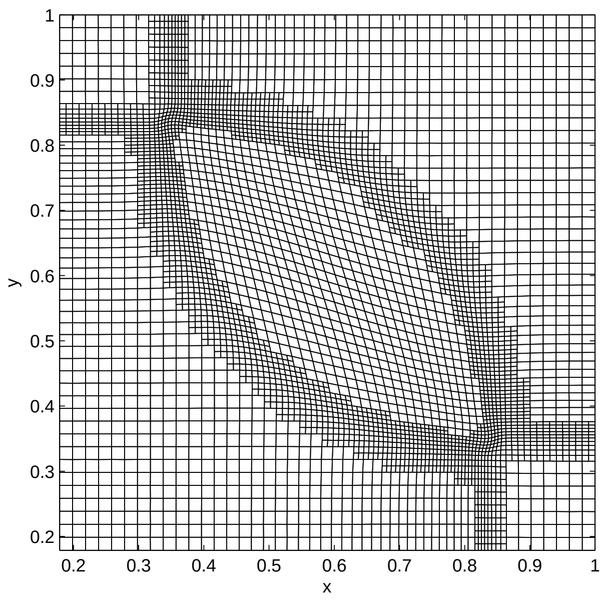

Figure 4.40: Two-dimensional Riemann problem isotropic mesh at t=0.2, refinement threshold 0.15.

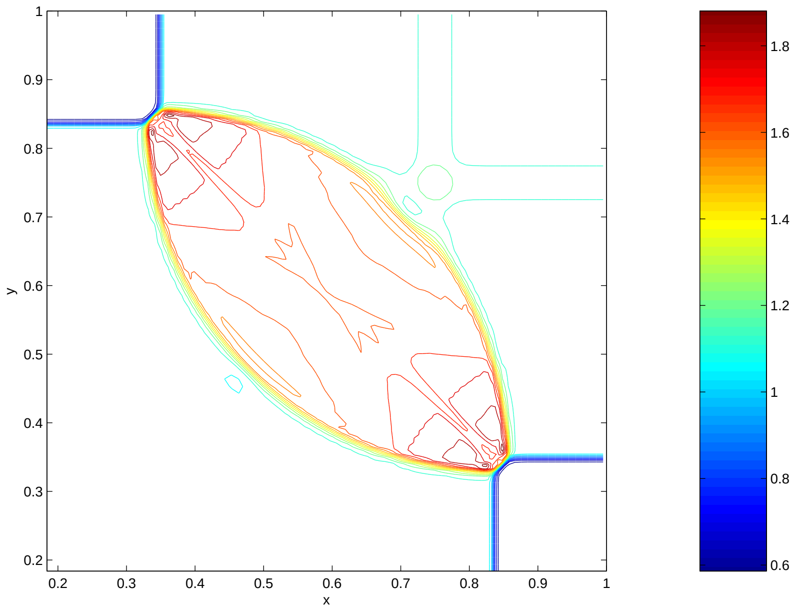

Figure 4.41: Two-dimensional Riemann problem uniformly fine calculation density contours at t=0.2.

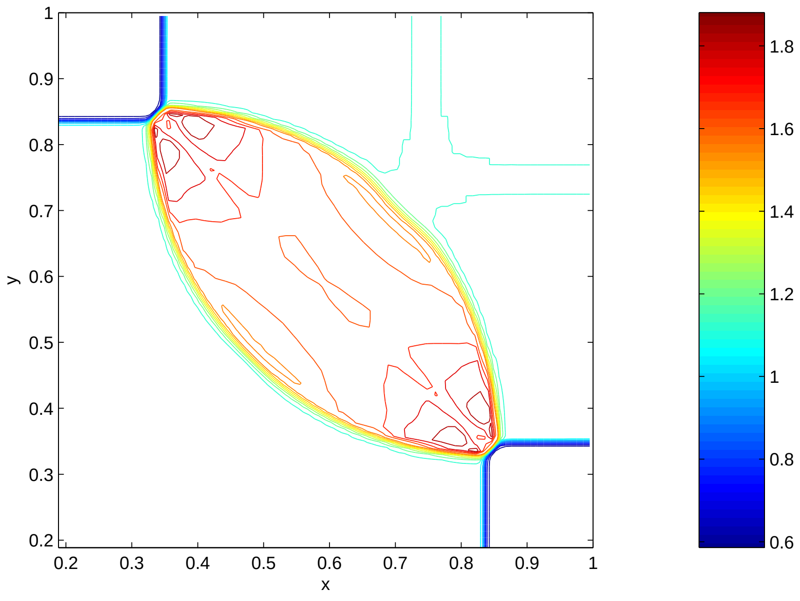

Figure 4.42: Two-dimensional Riemann problem isotropic calculation density contours at t=0.2.

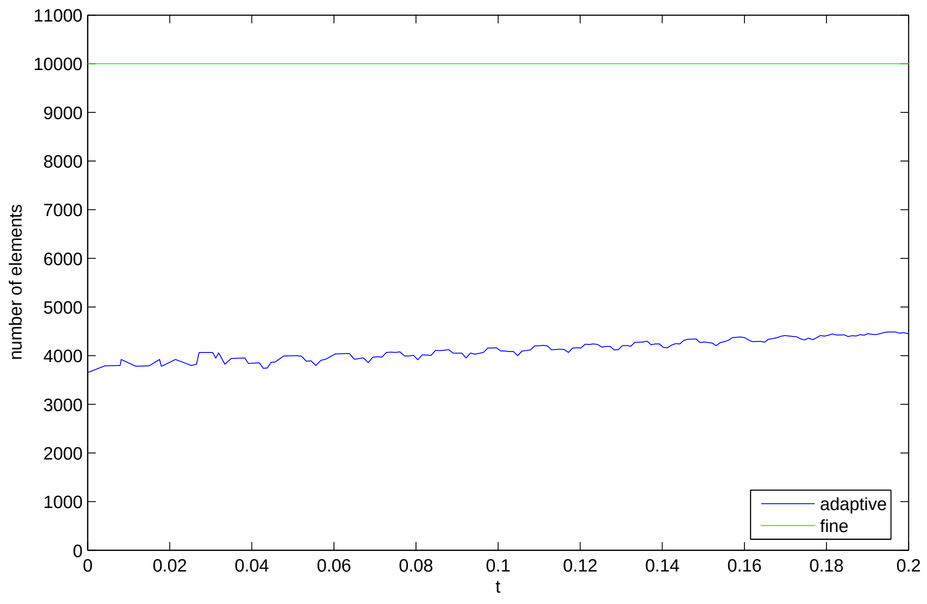

Figure 4.43: Two-dimensional Riemann problem comparing the number of elements required over calculation time.

Chapter 5

Second order solution transfer

In the adaptive mesh technique detailed in the previous chapters the values for the new fine elements and nodes have been obtained using first order methods. The cell centred values are set equal to the coarse value whilst the nodal values are formed from an average of the coarse nodal values. This may result in a loss of accuracy and so in this chapter we derive a second order solution transfer method.

There are three possible approaches to second order solution transfer that are also conservative: integral rezoning such as outlined by Dukowitz [36], [34], Ramshaw [60], [59] and Zalesak [90], an adaptation of the advection procedure detailed in Chapter 4 or a technique based on a method Anderson, Pember and Elliott use for AMR [6], [9].

Integral rezoning is often used when the mesh connectivity is altered and has been used in the Adaptive Mesh Insertion technique developed by Barlow [13]. The integral rezoning method involves integrating the values from the old mesh contained in the new element and then dividing by the new volume. Dukowitz [36], [34], Ramshaw [60], [59] and Zalesak [90] have developed efficient integral rezone methods by applying the divergence theorem to the conservation law and reducing the problem to a series of line integrals. This is very costly since it requires searching through the mesh lines for new and old mesh intersections. Furthermore, a second order rezoning method requires a two dimensional slope therefore making it expensive. A first order rezoning method was applied for Adaptive Mesh Insertion [13] but this would be too diffusive to apply often.

The advection procedure from the previous chapter is cheaper and second order. However, when the mesh connectivity changes it is difficult to ascertain the correct neighbours to interchange flux with. It is conceivable that the advection method could be applied four times to calculate the four fine values. However, since conservation is preserved by using neighbouring fluxes and setting influxes to zero, it would be vital to know the neighbours and their fluxes and these would change as the connectivity changes during the refinement.

Anderson, Pember and Elliott in their work on AMR [6], [9] developed a method where the interpolation itself is conservative removing the neighbouring flux problem. The method is easy to apply in a cell by cell sense. The method is second order, constant field preserving, uses monotonic limiters and can be applied easily to cell centred variables. The method is more difficult to apply to nodal variables, there are significant problems when trying to apply the method for a 2:1 refinement ratio such as used in this work. Solutions to these problems were developed and are detailed in this chapter. Due to the conservative nature and relative ease of the method it was considered the most suitable method for second order solution transfer.

5.1 Cell centred variables

Anderson, Pember and Elliott's formula [6], [9], [7], [8] simplifies for the case of subdividing into four fine elements to

$$\begin{aligned}
\varphi_{f,1} &= \varphi_c - \varphi_\xi \frac{1}{2}(X_{f,3} + X_{f,4}) - \varphi_\eta \frac{1}{2}(X_{f,2} + X_{f,3}) \\
\varphi_{f,2} &= \varphi_c - \varphi_\xi \frac{1}{2}(X_{f,3} + X_{f,4}) + \varphi_\eta \frac{1}{2}(X_{f,1} + X_{f,4}) \\
\varphi_{f,3} &= \varphi_c + \varphi_\xi \frac{1}{2}(X_{f,1} + X_{f,2}) + \varphi_\eta \frac{1}{2}(X_{f,1} + X_{f,4}) \\
\varphi_{f,4} &= \varphi_c + \varphi_\xi \frac{1}{2}(X_{f,1} + X_{f,2}) - \varphi_\eta \frac{1}{2}(X_{f,2} + X_{f,3}),
\end{aligned} \tag{5.1}$$

where f denotes a fine value, c denotes a coarse value, φ_ξ and φ_η are the monotonic isoparametric slopes in the η and ξ directions respectively. The $X_{f,i}$ denote the basis values of the new fine elements, see Figure 5.1, the basis X is volume for density and mass for specific internal energy.

The above expression depends on the basis consistency condition

$$X_c = X_{f,1} + X_{f,2} + X_{f,3} + X_{f,4}. \tag{5.2}$$

Only if the basis consistency condition is true is equation (5.1) conservative such that

$$\varphi_c X_c = \varphi_{f,1} X_{f,1} + \varphi_{f,2} X_{f,2} + \varphi_{f,3} X_{f,3} + \varphi_{f,4} X_{f,4}. \tag{5.3}$$

Coarse slopes are calculated in the same way as the advection slopes, any neighbouring fine data is conservatively coarsened. The partial volumes are calculated exactly, rather than from the integrals of the finite element functions.

The method is conservative, second order (in smooth regions) and constant field preserving. Note also that refining a coarse element and then coarsening it will obtain the original value therefore an exact inversion principle holds. Finally, monotonicity can be ensured in one dimension by the use of a Van Leer limiter [74], [75] on the slopes as was used for advection.

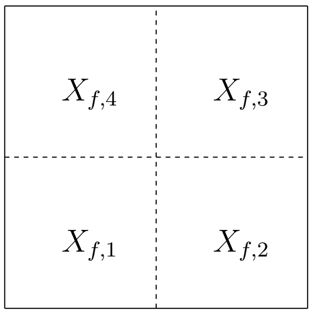

Figure 5.1: Diagram showing the basis weights for the coarse element being refined.

5.1.1 Monotonicity and interpolation in two dimensions

When interpolation occurs in two dimensions at once there is no guarantee that the results will be completely monotonic. This is because the limiting is only along the one dimensional slopes, it does not imply monotonicity when both slopes are used together. The biggest problem is that coarse values adjoin two fine values. If the minimum fine value could be guaranteed to be above its neighbouring coarse values in both directions and the maximum fine value could be below its neighbouring coarse values, the results would be monotone.

This is a procedure similar to that used in the fully multidimensional flux corrected transport of Zalesak [90]. Zalesak used a lower order flux to provide predicted values.

The fluxes were then limited according to the predicted neighbouring values in both directions. Each flux could be limited by a different amount.

In this work, the two neighbouring coarse values (or the conservative average of neighbouring fine values) provide the predicted values. Unfortunately because there are no fine predicted values it is not feasible to limit the fluxes individually. However, the whole slope, including the contribution from both directions, can be reduced by a factor. This factor is determined so that the minimum value is above its neighbouring coarse values and the maximum value is below its neighbouring coarse values.

This is a much harder limiter and can reduce the accuracy of the solution transfer. It will also reduce both slopes to zero if one slope was zero. Therefore, it is only used in strongly two-dimensional problems or where the interpolated densities or energies may drop below zero.

5.2 Nodal variables

When the cell centred interpolation is applied to nodal variables only an odd refinement ratio will maintain an r:1 correspondence between fine and coarse data, as shown in Figure 5.2, making the refinement exactly invertible by coarsening. Anderson *et al.* [6] use a 3:1 refinement ratio because of this, but by enforcing the base consistency condition they later loose exact invertibility anyway. We consider exact invertibility as less important, since we never refine then derefine immediately. We continue using a 2:1 refinement ratio because should a shock escape the fine areas, referring to Section 3.5, the resulting oscillations would be smaller than for a 3:1 ratio.

An associated problem is how to define the coarse nodal mass region, see Figure 5.3. The new fine nodes lie exactly on the coarse dual mesh lines; no fine nodal mass will ever lie totally inside just one coarse nodal mass. Conservative interpolation can not be applied to a whole nodal mass and so a new method of applying the cell centred interpolation was developed. The nodal velocities are interpolated out to the partial volumes to become nodal velocity weights. Each partial volume velocity weight can then be interpolated to form four new velocity weights. The new fine velocity weights and the unchanged existing velocity weights are then gathered back to the nodes to give the new nodal velocities.

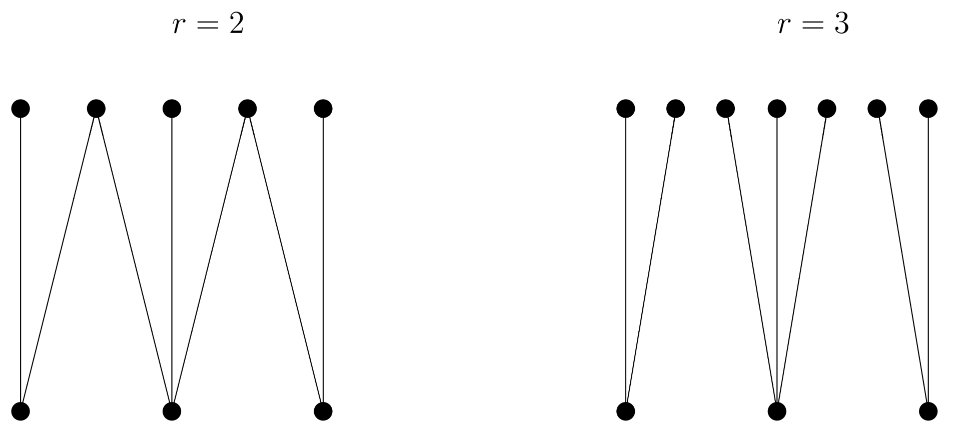

Figure 5.2: Even and Odd refinement ratio fine and coarse data correspondences.

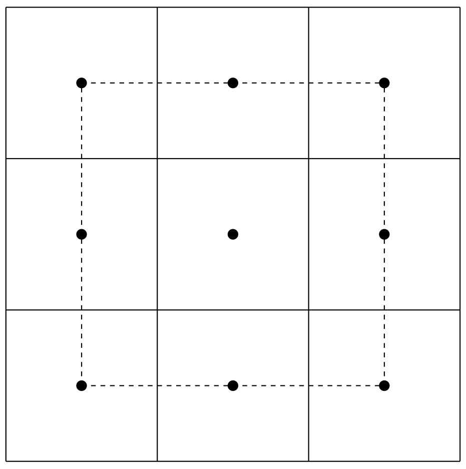

Figure 5.3: Diagram showing the coarse nodal mass region, dashed line indicates coarse nodal mass region boundary, solid lines indicate fine nodal mass region boundaries.

5.2.1 Nodal mass weights

Nodal mass weights are required for the old mesh and the newly refined mesh. The old mesh provides the coarse weights for the refinement procedure, see Figure 5.4, and these are given by

$$M_{c,i}^n = \begin{cases} \rho_c V_{c,i}^n & \text{if cell coarse} \\ \rho_f V_{f,i}^n & \text{if cell fine already,} \qquad \text{for } i = 1 \text{ to } 4, \end{cases} \tag{5.4}$$

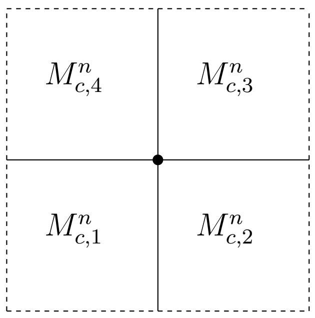

Figure 5.4: Diagram showing the mass weights associated with the partial volumes surrounding a node.

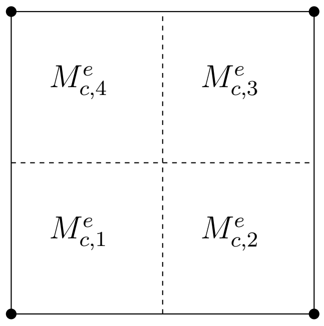

Figure 5.5: Diagram showing the mass weights associated with the partial volumes of an element.

where the $V^n_{c,i}$ or $V^n_{f,i}$ are the coarse or fine cell partial volumes that surround the node. The densities and volumes are the before refinement values, the fine values signifying that the surrounding element has already been refined in a previous step.

Note here that the weights could also be thought of as mass weights associated with the partial volumes of an element as illustrated in Figure 5.5, for example

$$M^e_{c,j} = \begin{cases} \rho_c V^e_{c,j} & \text{if cell coarse} \\ \rho_f V^e_{f,j} & \text{if cell fine already,} \qquad \text{for } j = 1 \text{ to } 4. \end{cases} \tag{5.5}$$

where e is the element that the partial volume is in.

The weights of the newly refined mesh, such as in Figure 5.6, use the coarse and fine element partial volumes and the new refined density values

$$M^e_{f,j} = \begin{cases} \rho_c V^e_{c,j} & \text{if cell coarse} \\ \rho_f V^e_{f,j} & \text{if cell fine,} \qquad \text{for } j = 1 \text{ to } 4. \end{cases} \tag{5.6}$$

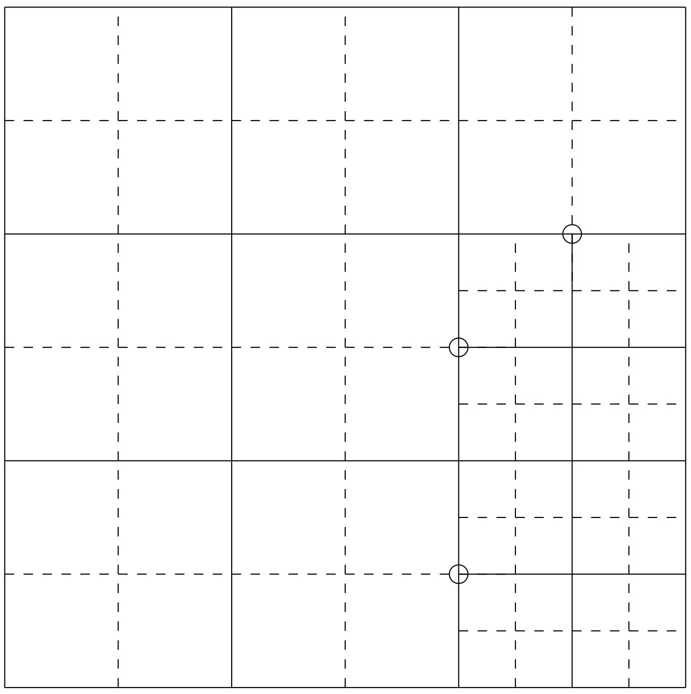
Figure 5.6: Diagram showing volumes associated with the weights of the newly refined mesh, indicated by the dotted lines, empty circles indicate disjoint nodes.

5.2.2 Velocity weights

Nodal velocity weights, see Figure 5.7, are formed by interpolating the nodal values out to the four partial volumes

$$\begin{aligned}
\mathbf{uw}_{c,1} &= \mathbf{u} - \mathbf{u}_\xi \frac{1}{2}(M^n_{c,3} + M^n_{c,4}) - \mathbf{u}_\eta \frac{1}{2}(M^n_{c,2} + M^n_{c,3}) \\
\mathbf{uw}_{c,2} &= \mathbf{u} - \mathbf{u}_\xi \frac{1}{2}(M^n_{c,3} + M^n_{c,4}) + \mathbf{u}_\eta \frac{1}{2}(M^n_{c,1} + M^n_{c,4}) \\
\mathbf{uw}_{c,3} &= \mathbf{u} + \mathbf{u}_\xi \frac{1}{2}(M^n_{c,1} + M^n_{c,2}) + \mathbf{u}_\eta \frac{1}{2}(M^n_{c,1} + M^n_{c,4}) \\
\mathbf{uw}_{c,4} &= \mathbf{u} + \mathbf{u}_\xi \frac{1}{2}(M^n_{c,1} + M^n_{c,2}) - \mathbf{u}_\eta \frac{1}{2}(M^n_{c,2} + M^n_{c,3}),
\end{aligned} \tag{5.7}$$

where $\mathbf{u}_\xi$ and $\mathbf{u}_\eta$ are the nodal velocity slopes and $\mathbf{u}$ is the nodal velocity. The $M^n_{c,i}$ are the coarse nodal mass weights, the numbering is in order from the bottom left anticlockwise around the node. The calculation of the nodal velocity slopes is discussed in the next section.

Since the disjoint nodes are non-dynamic points their nodal mass must be redistributed to the non-disjoint nodes. The nodal mass of a coarse node can be the sum of its mass and any disjoint node mass. The nodal velocity weights are associated with this form of nodal mass.

The partial volumes can then be considered like cells and the velocity weights like cell centred values. Considering this in an element framework, the coarse element to be refined has a velocity weight for each partial volume such that

$$\mathbf{uw}^e_{c,j} = \mathbf{uw}^n_{c,i}, \tag{5.8}$$

when

$$V^e_{c,j} = V^n_{c,i}. \tag{5.9}$$

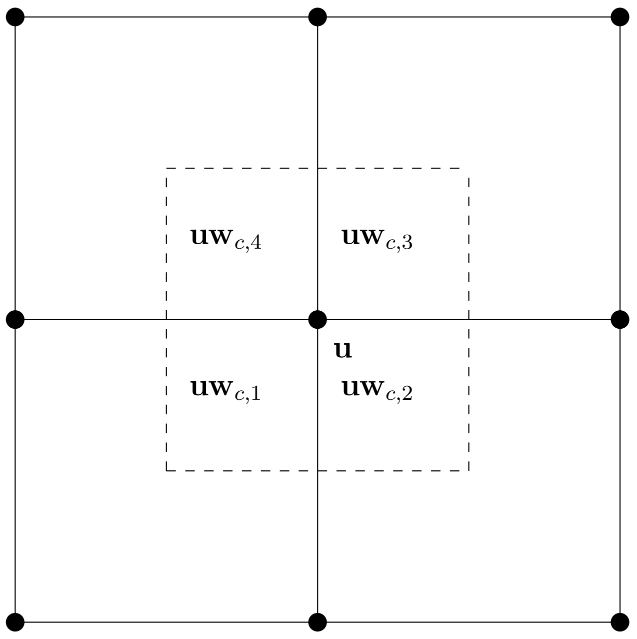

Figure 5.7: Diagram showing velocity weights associated with a node.

5.2.3 Nodal slopes used to calculate the velocity weights

The slopes $\mathbf{u}_\xi$ and $\mathbf{u}_\eta$ are calculated on the old mesh containing both coarse and fine elements. Each slope is calculated using three nodal velocity values and the nodal masses. The coarse nodes have their own mass plus any disjoint nodal mass associated with them. There are three unstructured mesh configurations that require changes to the slope calculation.

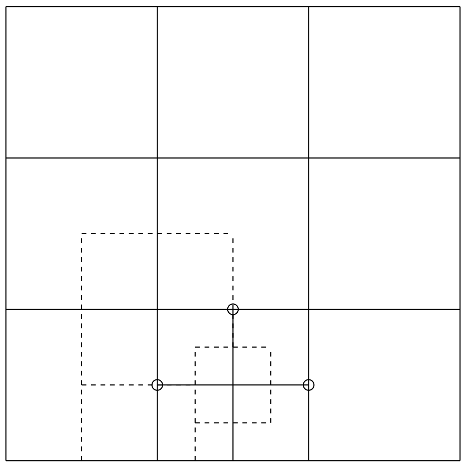

Figure 5.8: Diagram showing a node surrounded by one fine quarter, vertical slope being considered. Circles represent disjoint nodes.

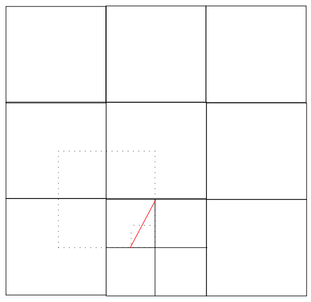

Figure 5.9: Nodal slopes for second order velocity weights. The nodal mass step caused by two disjoint nodes is approximated by a varying width, shown in red.

The configuration shown in Figure 5.8 involves a nodal mass step due to two neighbouring disjoint nodes, this is approximately equivalent to an element whose width is varying as shown in Figure 5.9. The slope calculation takes width variations within an element into account, thus no change to the parabolic slope is required. More care must be taken when calculating the Van Leer limiting slopes, see equation (2.48). There are two dual cells that adjoin the central dual cell so both of their velocities are used to calculate limiting slopes on that side. The lowest absolute value of the limiting slope is

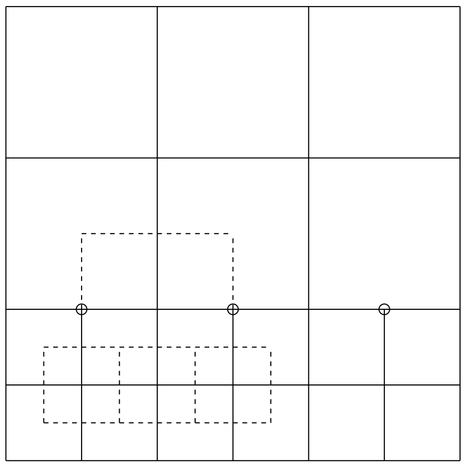

Figure 5.10: Diagram showing a coarse fine interface perpendicular to the slope direction. Circles represent disjoint nodes.

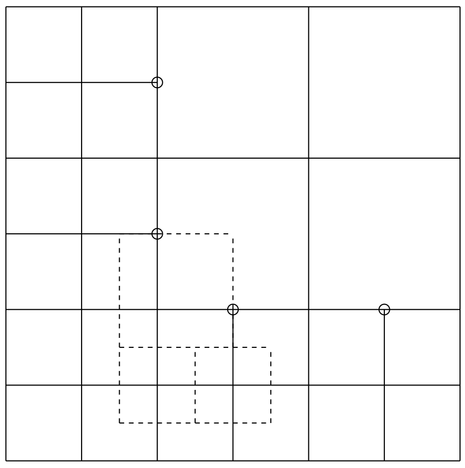

Figure 5.11: Diagram showing a node surrounded by three fine quarters as two coarse fine interfaces meet forming a corner, vertical slope being considered. Circles represent disjoint nodes.

then used.

For the configuration shown in Figure 5.10, the central cell adjoins three dual cells on the side of interest. All three values are used to calculate limiting slopes and the minimum slope is used. The calculation of the parabolic slope must take into account the abrupt change in dual cell width between the central cell and its neighbour. The width of the neighbour is fine on both sides, to make it comparable to the coarse dual cell the width is doubled. For the case shown in Figure 5.11, two values are used for the limiting on that side and the neighbouring width is multiplied by 3/2.

5.2.4 Basis consistency and the coarse velocity field

A further problem that needs to be considered when using the cell centred interpolation procedure is that the basis consistency condition may not hold for nodal values. For equation (5.1) to be conservative the coarse velocity weights must be associated with a coarse mass weight equal to the sum of the fine mass weights

$$M_c = M_{f,1} + M_{f,2} + M_{f,3} + M_{f,4}. \tag{5.10}$$

The sum of the fine partial volumes equals the coarse partial volume but the coarse density ρ_c does not usually equal the fine density ρ_f because the second order cell centred interpolation was used to calculate the fine element densities. Therefore, the coarse nodal mass does not equal the sum of the fine nodal masses

$$\rho_c V_c \neq \rho_f V_{f,1} + \rho_f V_{f,2} + \rho_f V_{f,3} + \rho_f V_{f,4}, \tag{5.11}$$

$$M_{c,j} \neq M_{f,1} + M_{f,2} + M_{f,3} + M_{f,4}, \tag{5.12}$$

where $M_{f,1}$ etc. equal the fine mass weights inside the fine element with volume equal to $V_{c,j}$.

The basis consistency condition can be satisfied by altering the coarse nodal mass weight and coarse velocity weight so that they are associated with the sum of the fine mass weights. Consider the difference between the coarse mass weight and the sum of the fine mass weights as the result of mass fluxes caused by the density gradients used in calculating the new fine densities. These mass fluxes pass between the new fine elements or coarse partial volumes perturbing the coarse mass weights. The mass fluxes cause momentum fluxes between the new fine elements. These momentum fluxes perturb the coarse velocity weights.

Multiplying the mass fluxes by the velocity at the boundary where the mass is interchanged gives the momentum fluxes that perturb the coarse velocity weights. This is illustrated in Figure 5.12. Since the mass change should be a small perturbation it was considered accurate enough to use an average of the velocity weights for the point velocities

$$\begin{aligned}
\mathbf{u}_1^* &= \frac{1}{2}(\mathbf{uw}_{c,1} + \mathbf{uw}_{c,2}) \\
\mathbf{u}_2^* &= \frac{1}{2}(\mathbf{uw}_{c,2} + \mathbf{uw}_{c,3}) \\
\mathbf{u}_3^* &= \frac{1}{2}(\mathbf{uw}_{c,3} + \mathbf{uw}_{c,4}) \\
\mathbf{u}_4^* &= \frac{1}{2}(\mathbf{uw}_{c,4} + \mathbf{uw}_{c,1}) \\
\mathbf{u}_m^* &= \frac{1}{4}(\mathbf{uw}_{c,1} + \mathbf{uw}_{c,2} + \mathbf{uw}_{c,3} + \mathbf{uw}_{c,4}). \qquad (5.13)
\end{aligned}$$

The momentum changes to the four velocity weights are then

$$\begin{aligned}
\Delta\mathbf{u}_{c,1} &= \rho_\eta \frac{1}{2}(-V_{c,1}V_{c,2}\mathbf{u}_1^* - V_{c,1}V_{c,3}\mathbf{u}_m^*) + \rho_\xi \frac{1}{2}(-V_{c,1}V_{c,4}\mathbf{u}_4^* - V_{c,1}V_{c,3}\mathbf{u}_m^*) \\
\Delta\mathbf{u}_{c,2} &= \rho_\eta \frac{1}{2}(V_{c,1}V_{c,2}\mathbf{u}_1^* + V_{c,2}V_{c,4}\mathbf{u}_m^*) + \rho_\xi \frac{1}{2}(-V_{c,2}V_{c,3}\mathbf{u}_2^* - V_{c,2}V_{c,4}\mathbf{u}_m^*) \\
\Delta\mathbf{u}_{c,3} &= \rho_\eta \frac{1}{2}(V_{c,3}V_{c,4}\mathbf{u}_3^* + V_{c,1}V_{c,3}\mathbf{u}_m^*) + \rho_\xi \frac{1}{2}(V_{c,2}V_{c,3}\mathbf{u}_2^* + V_{c,1}V_{c,3}\mathbf{u}_m^*) \\
\Delta\mathbf{u}_{c,4} &= \rho_\eta \frac{1}{2}(-V_{c,3}V_{c,4}\mathbf{u}_3^* - V_{c,2}V_{c,4}\mathbf{u}_m^*) + \rho_\xi \frac{1}{2}(V_{c,1}V_{c,4}\mathbf{u}_4^* + V_{c,2}V_{c,4}\mathbf{u}_m^*), \qquad (5.14)
\end{aligned}$$

where ρ_ξ and ρ_η are the monotonic isoparametric slopes in the η and ξ directions respectively used in equation (5.1).

When the $\Delta\mathbf{u}_{c,j}$ are constructed in this manner we have

$$\sum_{j=1}^{4} \Delta\mathbf{u}_{c,j} = 0, \qquad (5.15)$$

which ensures conservation of momentum is preserved as detailed in Section 5.2.7.

The new value for the weight is given by

$$\mathbf{uw}_{c,j}^* = \frac{\mathbf{uw}_{c,j}M_{c,j} + \Delta\mathbf{u}_{c,j}}{M_c}, \qquad (5.16)$$

where M_c equals the sum of the fine mass weights.

Redistributing the velocity weights will lead to a loss of precise invertibility for any refinement ratio (Anderson *et al*'s work [6] looses precise invertibility at this point also). However, since invertibility has already been abandoned by using an even refinement ratio it will cause no additional problems.

5.2.5 Disjoint nodes

In previous chapters disjoint nodes have been considered non-dynamic points, their velocities have been slaved and their nodal mass redistributed. Slaving the disjoint

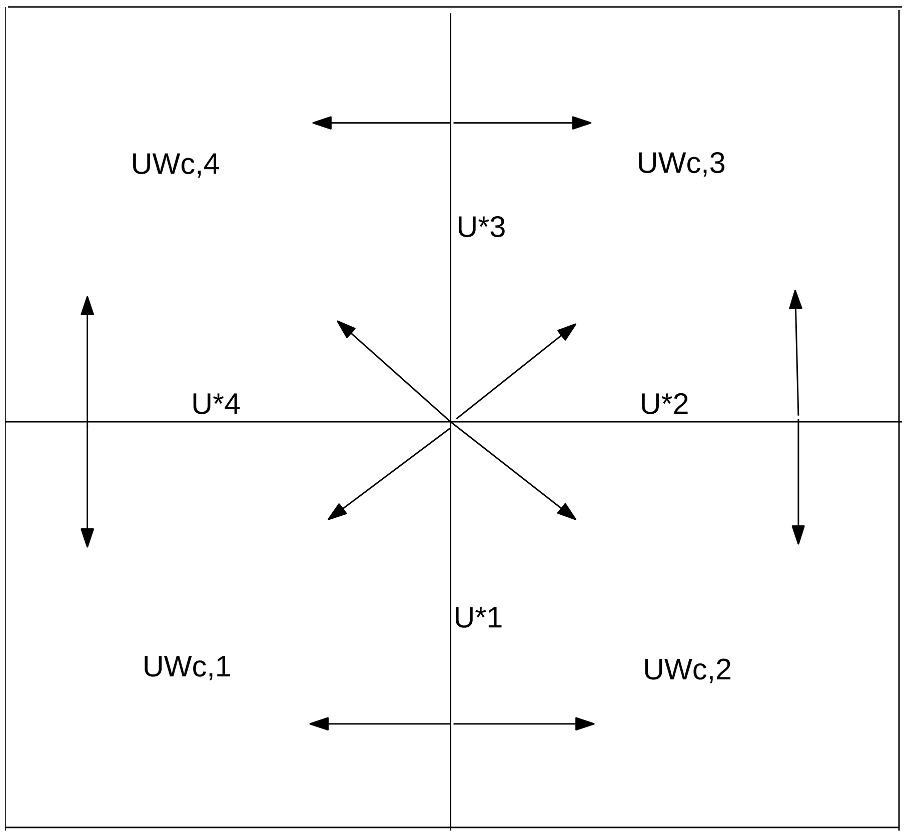

Figure 5.12: Momentum fluxes caused by the density slopes and the point velocities used to calculate them.

nodes when we interpolate from the coarse velocity weights to form the new fine velocity weights would require interpolation to two or three strangely shaped cells. This would be highly impractical. The approach therefore adopted is to retain the usual interpolation to four values. The disjoint nodes are considered to have nodal mass associated with them. To be able to slave the disjoint node velocities at the very end of the solution transfer we require for conservation a procedure equivalent to the disjoint nodes having no associated nodal mass. The disjoint nodal masses are set so that the conservation of explicitly calculating the disjoint node values is as if only the non-disjoint node velocities with the additional mass had been calculated.
When the disjoint node velocity is slaved

$$\mathbf{u}_d = \frac{\mathbf{u}_1 + \mathbf{u}_2}{2}, \tag{5.17}$$

this implies for conservation that

$$M_1\mathbf{u}_1 + M_2\mathbf{u}_2 + M_d\left(\frac{\mathbf{u}_1 + \mathbf{u}_2}{2}\right) = \left(M_1 + \frac{M_d}{2}\right)\mathbf{u}_1 + \left(M_2 + \frac{M_d}{2}\right)\mathbf{u}_2. \tag{5.18}$$

Therefore, each non-disjoint node must be given $M_d/2$. To do this the disjoint node nodal masses are altered to both equal half the total disjoint nodal mass, $M_d/2$. This represents a change in the coarse nodal mass weights therefore the coarse velocities must be altered using the principles from Section 5.2.4.

Once all the new fine velocity weights have been calculated as described in the next section, the disjoint node fine velocity weights and nodal masses can be redistributed to the non-disjoint nodes. Considering Figure 5.13 and non-disjoint node 1 its fine velocity weight becomes

$$\mathbf{uw}_1 = \frac{\mathbf{uw}_{f,3} M_{f,3} + \mathbf{uw}_d \frac{M_d}{2}}{M_{f,3} + \frac{M_d}{2}}. \tag{5.19}$$

At the end of the solution transfer once the new nodal velocities have been calculated the disjoint node velocities are set to the average of the non-disjoint node velocities.

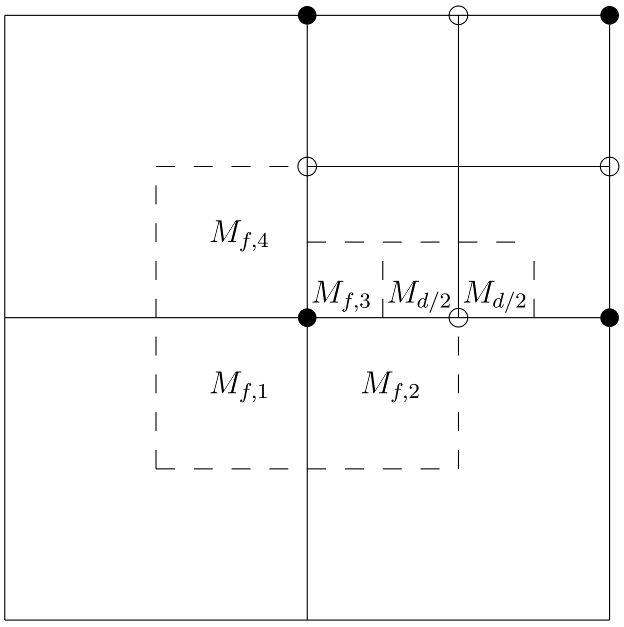

Figure 5.13: Diagram showing the disjoint node procedure.

5.2.6 Interpolation and calculating fine nodal velocities

Four fine velocity weights are calculated from each coarse velocity weight using the cell centred interpolation. For the velocity weight $\mathbf{uw}_{c,1}$ in Figure 5.14 the four new weights are

$$\begin{aligned}
\mathbf{uw}_{f,1} &= \mathbf{uw}_{c,1} - \mathbf{uw}_{\xi}\frac{1}{2}(M^e_{f,3} + M^e_{f,4}) - \mathbf{uw}_{\eta}\frac{1}{2}(M^e_{f,2} + M^e_{f,3}) \\
\mathbf{uw}_{f,2} &= \mathbf{uw}_{c,1} - \mathbf{uw}_{\xi}\frac{1}{2}(M^e_{f,3} + M^e_{f,4}) + \mathbf{uw}_{\eta}\frac{1}{2}(M^e_{f,1} + M^e_{f,4}) \\
\mathbf{uw}_{f,3} &= \mathbf{uw}_{c,1} + \mathbf{uw}_{\xi}\frac{1}{2}(M^e_{f,1} + M^e_{f,2}) + \mathbf{uw}_{\eta}\frac{1}{2}(M^e_{f,1} + M^e_{f,4}) \\
\mathbf{uw}_{f,4} &= \mathbf{uw}_{c,1} + \mathbf{uw}_{\xi}\frac{1}{2}(M^e_{f,1} + M^e_{f,2}) - \mathbf{uw}_{\eta}\frac{1}{2}(M^e_{f,2} + M^e_{f,3}),
\end{aligned} \tag{5.20}$$

where the $M^e_{f,i}$ are the mass weights of the new fine element e. Each new velocity weight is centred in a smaller partial volume $V_{f,j}$ dividing the new fine element e.

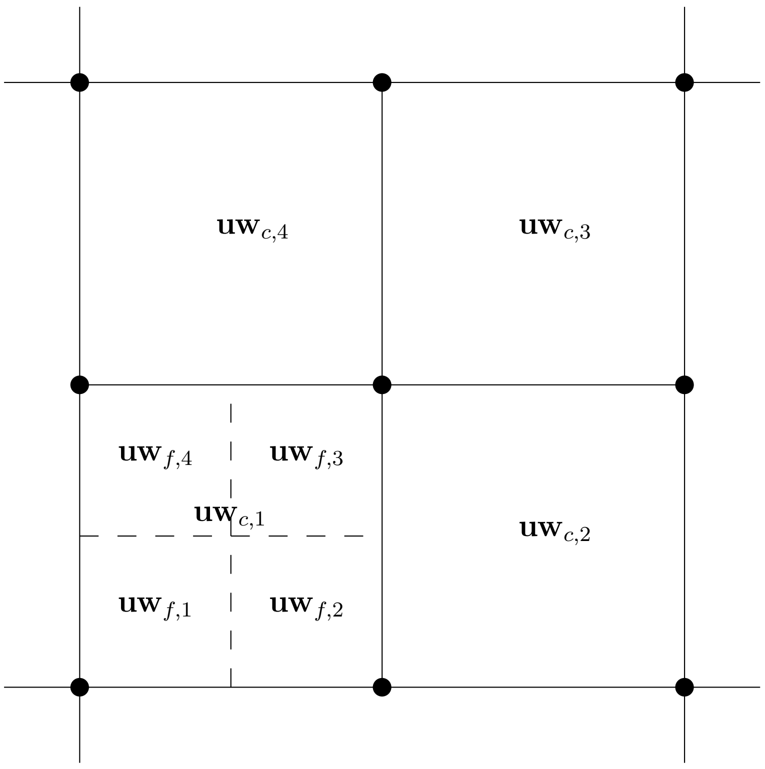

Figure 5.14: Diagram showing nodal interpolation of $\mathbf{uw}_{c,1}$.

The slopes $\mathbf{uw}_{\xi}$ and $\mathbf{uw}_{\eta}$ are taken through three adjacent velocity weights. Since the coarse velocity weights were calculated with disjoint nodes as non-dynamic points whose nodal mass was redistributed to the non-disjoint nodes, the neighbouring velocity weight cell may have three possible configurations, see Figure 5.15. In the slope calculation we require fine mass weights for each velocity weight cell. Where the neighbouring cell is not being refined, the fine mass weights are simply set as a quarter of the coarse nodal mass weight associated with the coarse velocity weight. If the neighbour is to be refined, we already have the fine mass weights and there is no problem.

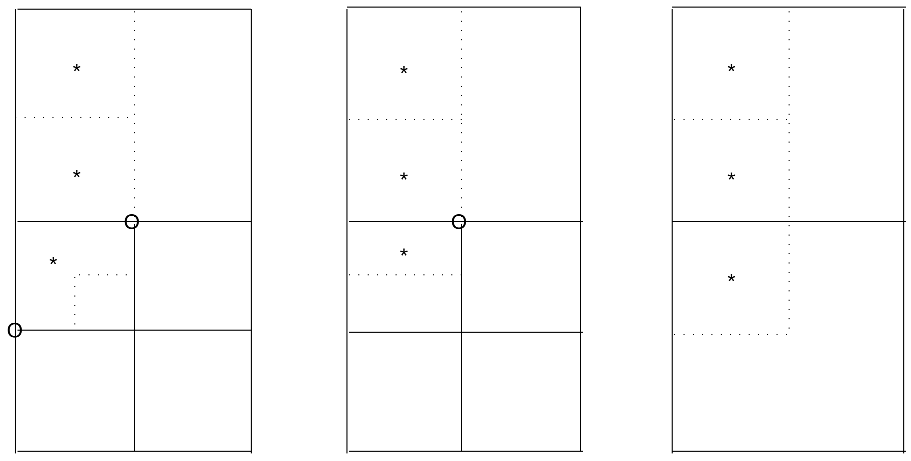
Figure 5.15: Slope configurations showing coarse velocity weights as stars. The diagrams show a fine neighbour with two disjoint nodes, a fine neighbour with one disjoint node and a coarse neighbour.

Once the fine velocity weights have been calculated those surrounding each node, as shown in Figure 5.16, can then be conservatively gathered back to that node. The nodal value then becomes

$$\mathbf{u} = \frac{\sum_{i=1}^{4} \mathbf{uw}_{f,i} M^n_{f,i}}{\sum_{i=1}^{4} M^n_{f,i}}, \tag{5.21}$$

where the $\mathbf{uw}_{f,i}$ are equal to the newly calculated values if the element has been refined and the unchanged velocity weights if the element has not.

5.2.7 Conservation and uniform field preservation

The gather procedure is conservative

$$\sum_{nodes} \mathbf{u}_{new} \sum_{i=1}^{4} M_{f,i} = \sum_{elements} \sum_{j=1}^{4} \sum_{k=1}^{4} \mathbf{uw}_{f_j,k} M_{f_j,k}, \tag{5.22}$$

here *elements* denotes the coarse elements.

The conservation properties of Equation (5.1) and the basis consistency condition combined give

$$\sum_{elements} \sum_{j=1}^{4} \sum_{k=1}^{4} \mathbf{uw}_{f_j,k} M_{f_j,k} = \sum_{elements} \sum_{j=1}^{4} \mathbf{uw}^*_{c,j} \sum_{k=1}^{4} M_{f_j,k} \tag{5.23}$$

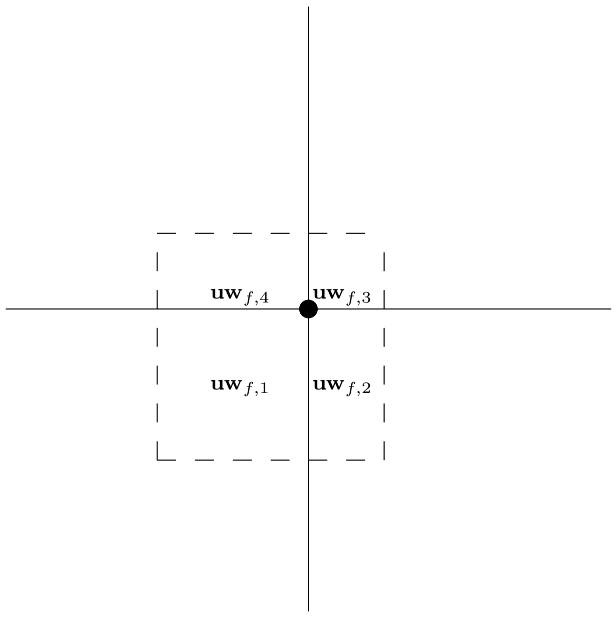

Figure 5.16: Diagram showing the velocity weights being gathered to the node.

and using the equation for $\mathbf{uw}^*_{c,j}$ we obtain

$$\sum_{elements}\sum_{j=1}^{4}\mathbf{uw}^*_{c,j}\sum_{k=1}^{4}M_{f_j,k} = \sum_{elements}\sum_{j=1}^{4}\left(\left[\frac{\mathbf{uw}_{c,j}M_{c,j}+\Delta\mathbf{u}_{c,j}}{\sum_{k=1}^{4}M_{f_j,k}}\right]\sum_{k=1}^{4}M_{f_j,k}\right). \tag{5.24}$$

Simplifying and using equation (5.15) then gives

$$\sum_{elements}\sum_{j=1}^{4}(\mathbf{uw}_{c,j}M_{c,j}+\Delta\mathbf{u}_{c,j}) = \sum_{elements}\sum_{j=1}^{4}\mathbf{uw}_{c,j}M_{c,j} \tag{5.25}$$

and therefore we obtain

$$\sum_{elements}\sum_{j=1}^{4}\mathbf{uw}^*_{c,j}\sum_{k=1}^{4}M_{f_j,k} = \sum_{elements}\sum_{j=1}^{4}\mathbf{uw}_{c,j}M_{c,j}. \tag{5.26}$$

The element sum is equivalent to a sum of the four weights around the node

$$\sum_{elements}\sum_{j=1}^{4}\mathbf{uw}_{c,j}M_{c,j} = \sum_{nodes}\sum_{l=1}^{4}\mathbf{uw}_{c,l}M_{c,l} \tag{5.27}$$

and so using the conservation properties of Equation (5.1) again gives

$$\sum_{nodes}\sum_{l=1}^{4}\mathbf{uw}_{c,l}M_{c,l} = \sum_{nodes}\mathbf{u}\sum_{l=1}^{4}M_{c,l}. \tag{5.28}$$

Therefore, conservation over the whole domain still holds

$$\sum_{nodes}\mathbf{u}_{new}\sum_{i=1}^{4}M_{f,i} = \sum_{nodes}\mathbf{u}\sum_{l=1}^{4}M_{c,l}. \tag{5.29}$$

The nodal interpolation conserves momentum around the nodes when the velocity weights are formed whilst the later procedures conserve momentum within the coarse element. Therefore, momentum is conserved locally within the nodal mass perimeter of the group of cells being refined, see Figure 5.17.

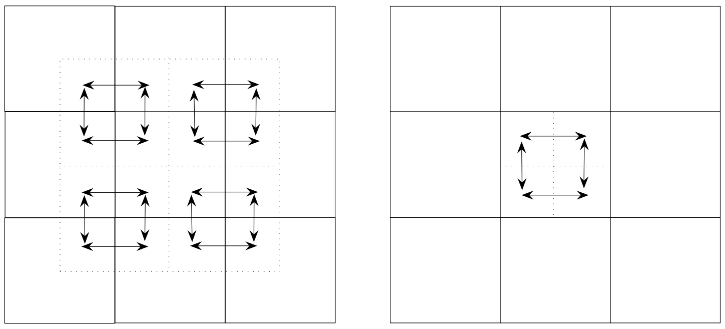

Figure 5.17: Momentum conservation when one element is refined. The left diagram shows interpolation to form coarse velocity weights. The right diagram shows the velocity fluxes caused by the density slope. The large dotted rectangle in the left diagram shows the outermost perimeter within which fluxes are passed, local conservation holds within this rectangle.

The procedure will still preserve a constant velocity field. If there is no slope then the second order interpolation will just give constant velocity weights. If the velocity is constant there will be no change in the velocity weight due to the flux transfer resulting from the density slope. The interpolation to fine weights will give constant velocity values. The gathering process must then produce constant nodal velocities.

5.2.8 Monotonicity

The interpolation itself is monotonic, when the Van Leer limiting in one dimension and the hard limiter discussed in Section 5.1.1 are employed. However, the whole method for calculating new nodal values can not be guaranteed to be monotonic. The parts of the method highlighted below are not necessarily monotonicity preserving.

- The momentum flux transfer caused by the density slopes can destroy monotonicity because it alters the coarse velocity weights. This procedure is required for conservation and the flux is defined by the density slope. Since this procedure is the result of the already set fine densities, it can not be limited and it must be accepted that this may not produce monotonic values.
- Slaving the disjoint node values, although promoting monotonicity along the coarse fine interface, takes no account of monotonicity perpendicular to it. However, this must also be the case when the disjoint nodes are slaved in the Lagrangian step. The possible loss of monotonicity is accepted as a result of the slaving procedure.
- Gathering the velocity weights to the nodes may not always result in monotonic nodal velocities. Gathering values that can vary with both their x and y positions, will not always produce monotonic results. However, in the usual case, where the values for the four nodal mass weights are close to each other, the procedure will normally be monotonic. As gathering can be thought of as a coarsening procedure, it seems correct to use it and there is little that can be done about the possible loss of monotonicity.

The gathering procedure failed to produce monotonic results when calculations were performed without the disjoint node mass being redistributed to the non-disjoint nodes after the final interpolation. Slaving the disjoint nodes leads to a more equal collection of nodal mass weights and the velocity values are exceedingly close to being monotone. Redistributing the disjoint node mass and slaving the velocities of the disjoint nodes appears preferable.

In conclusion, the new nodal refinement procedure is uniform field preserving, second order in smooth regions, conservative and nearly monotonic. Any scheme that uses one-dimensional slope calculations will never guarantee monotonicity during interpolation in two dimensions. However, with the use of the hard limiter, steps have been taken to try and improve this. The use of a fully two dimensional slope calculation or limited two dimensional parabolic interpolation would be extremely expensive. This method fulfills most of the requirements and is considered a suitable compromise.

5.2.9 Nodal variable derefinement

The second order solution transfer requires a nodal variable derefinement procedure. The velocity weights for the refined area must be conservatively gathered to form coarse nodal values. Considering the refined element in Figure 5.18, n_1, n_3, n_7 and n_9 will remain as coarse nodes after derefinement.

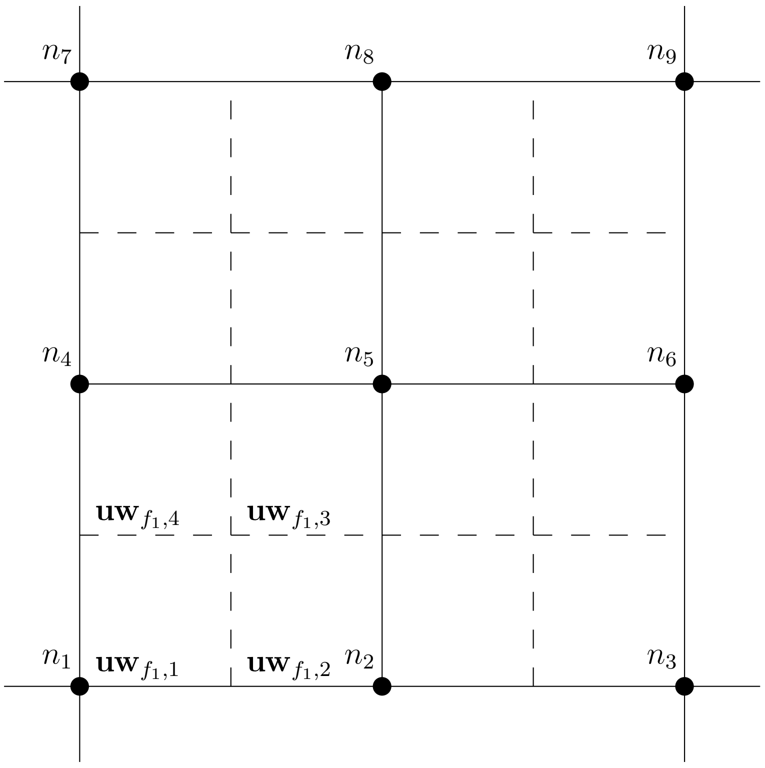

Figure 5.18: Diagram illustrating the derefinement of an element and the gathering of velocity weights.

The four velocity weights around each node are given the nodal value $\mathbf{u}$. All four velocity weights within the fine element are then gathered to the neighbouring coarse node, forming a coarse velocity weight for the retained coarse node

$$\mathbf{uw}_{c,j} = \frac{\sum_{k=1}^{4} \mathbf{uw}_{f_j,k} M_{f_j,k}}{\sum_{k=1}^{4} M_{f_j,k}}. \tag{5.30}$$

Once again the basis consistency condition must hold. The coarse velocity weights are associated with the coarse mass weights $M_{c,j}$ that depend on ρ_c, while the gathered fine weights are associated with $\sum_{k=1}^{4} M_{f_j,k}$ that depend on ρ_f. These are not equivalent because of the second order density transfer. To satisfy the basis consistency condition, momentum fluxes are passed between the four coarse velocity weights in the coarse element so that the velocity weights are associated with the $M_{c,j}$.

This requires density gradients that are approximated as

$$\begin{aligned} \rho_\eta &= \frac{2}{\sum_{j=1}^{4} V_{f_j}} \left(\frac{\rho_{f2}V_{f2} + \rho_{f3}V_{f3}}{V_{f2} + V_{f3}} - \frac{\rho_{f1}V_{f1} + \rho_{f4}V_{f4}}{V_{f1} + V_{f4}} \right) \\ \rho_\xi &= \frac{2}{\sum_{j=1}^{4} V_{f_j}} \left(\frac{\rho_{f4}V_{f4} + \rho_{f3}V_{f3}}{V_{f4} + V_{f3}} - \frac{\rho_{f1}V_{f1} + \rho_{f2}V_{f2}}{V_{f1} + V_{f2}} \right). \end{aligned} \tag{5.31}$$

The adjusted velocity coarse weights are

$$\mathbf{uw}^*_{c,j} = \frac{\mathbf{uw}_{c,j} \sum_{k=1}^{4} M_{f_j,k} + \Delta\mathbf{u}_{c,j}}{M_{c,j}}, \tag{5.32}$$

where the $\Delta\mathbf{u}_{c,j}$ are calculated as in Section 5.2.4.

Once velocity weights have been constructed for the new mesh they are gathered conservatively to form the coarse nodal velocities.

5.3 Results for second order refinement

The radial Sod problem was rerun with second order refinement. The adaptive calculation error was calculated as before by comparing to the one-dimensional fine Riemann solver solution [84]. The second order refinement reduced the error for both the length weighted equipotential relaxation method and the coarse stencil relaxation method. The error reduced by the greatest amount for the length weighted equipotential relaxation (Method 2) and this is illustrated in Table 5.1. The number of elements required for the adaptive calculation reduced slightly as well.

Calculation	$\|e\|_1$	$\|e\|_2$
fine	0.0044	0.0105
first order	0.0046	0.0109
second order	0.0045	0.0108

Table 5.1: Errors for fine, isotropic first order and isotropic second order.

The Sedov problem was rerun with second order transfer producing results that were closer to the fine shock radius and reached a higher peak value than the first order results did, see Figure 5.19. However, the improvement seen when using second order transfer is fairly small.

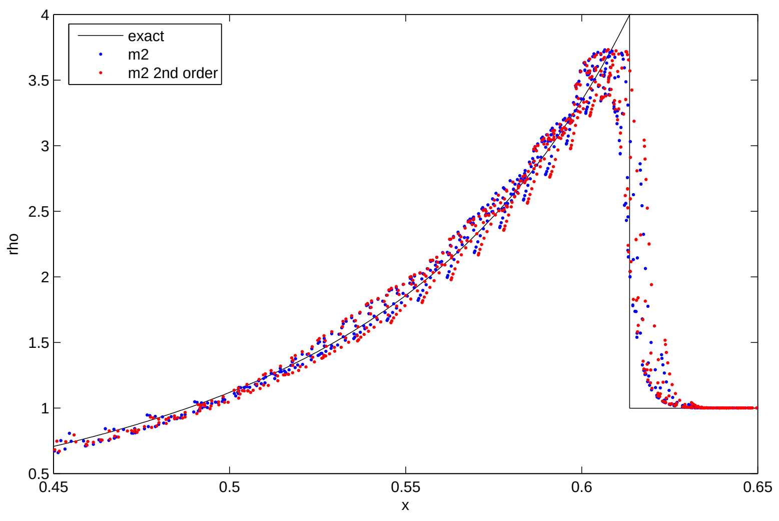

Figure 5.19: Sedov problem radial density comparing first order and second order transfer results, Method 2 stencil at t=0.1.

The two-dimensional Riemann problem was rerun with second order transfer. This resulted in sharper shocks and less elements being required during the adaptive calculation. However, more noise was evident inside the density oval in the derefined region, see Figure 5.20. This could be due to a lack of monotonicity as the hard limiter has not been applied.

The hard limiter when applied to all variables reduced the effect of the second order transfer because many slopes were set to zero, the results were almost identical to the original first order method. It is recommended therefore that the hard limiter is only employed to stop variables such as internal energy and density dropping below zero. This illustrates the difficult balance between higher order accuracy and monotonicity, and further research into higher order transfer and monotonicity would be beneficial.

It can be seen that the second order refinement although reducing the error a little has not had a large effect. This is probably because the buffer elements ensure that elements refine before significant variable changes occur around those elements. The second order accuracy takes slopes into account, but if these are small because the variables are not varying, the result will be the same or only a little better than the

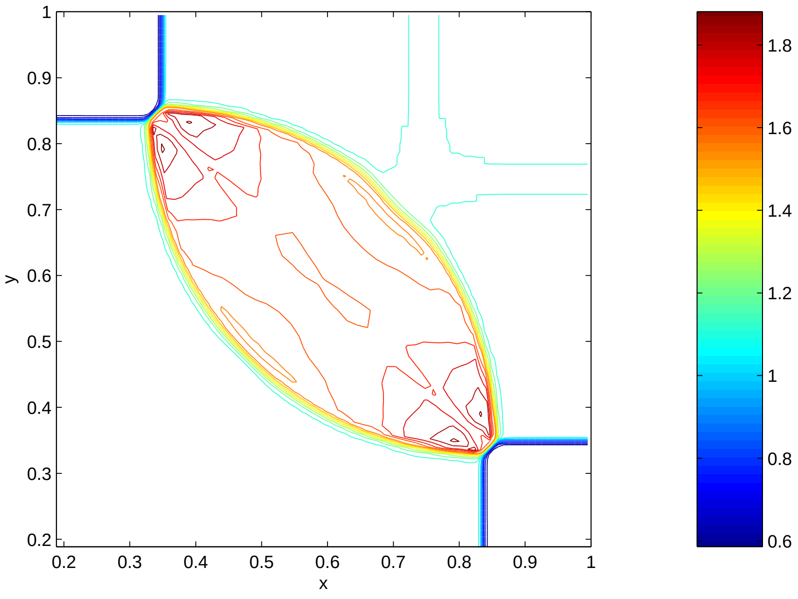

Figure 5.20: Two dimensional Riemann problem second order transfer, Method 2 stencil at t=0.2.

original first order transfer. Furthermore, two of these problems consisted of constant states where the slope will have no affect, and shocks where limiting may reduce the second order transfer to first order anyway. This helps to explain why the initial results with the first order transfer were so successful.

In conclusion, although second order transfer has resulted in improved results the difference is not substantial. Hard limiting can be introduced to ensure monotonicity but this can further reduce the effectiveness of the second order transfer. It should be noted here that the second order transfer has only been tried with one level of refinement, second order transfer is likely to provide more significant improvements when many levels of refinement are used as refinement may be occurring where variable slopes are more important and it will be vital to retain the accuracy of the data at each level. Having developed and investigated an isotropic adaptive mesh method, in the next chapter we extend the adaptive technique to include anisotropic h-refinement.

Chapter 6

Anisotropic mesh refinement

In this chapter the refinement method is extended to include anisotropically refined elements as well as the existing isotropic refinement. Anisotropic refinement enables the refinement to be defined by the dominant direction of the problem. Since the Lagrangian mesh already approximately aligns with the directions of flow, subdividing in the dominant direction using anisotropic refinement should be particularly beneficial and efficient.

Many physical features of interest, such as shocks and boundary layers, and many physical processes, such as shear and friction, represent large physical variations or changes over small length scales in one dominant direction. Anisotropic refinement enables the rapid variations or fine scale features to be resolved in the direction of interest, without requiring unnecessary refinement in the other direction. This allows calculations to be performed with fewer elements, taking advantage of the physics and allowing resolutions to be reached that would otherwise be unobtainable.

Anisotropic adaptivity may come in different forms - subdivision of elements in one direction, edge swapping in the case of tetrahedra and movement of nodes to align with features of the solution are examples of commonly used methods. These methods can be coupled to an anisotropic error indicator to reduce the error in a calculation.

Apel, Jimack *et al.* [11], [12], [80], [81], [82] have investigated the theoretical aspects of anisotropic finite element methods in elliptic and hyperbolic problems. This includes research into: the validity of high aspect ratio finite elements; estimating local interpolation error; error estimator theory; residual error estimators; local problems; equidistribution according to edge second derivatives; and using error estimators to drive the refinement. The key features that must be utilized for a successful anisotropic algorithm were identified as information on the stretching direction, aspect ratio and element size. However, it is explained that none of these areas are fully understood yet. Some of the work is specific to the equation and the work considers a time independent equation and not the Euler equations.

Anisotropic adaptivity with triangular finite elements for convection dominated hyperbolic problems has been considered in a series of papers by Walkley, Jimack and Berzins [80], [81], [82], [12]. These combine the usual isotropic h-refinement subdivision, with nodal movement and edge swapping to align the mesh according to the anisotropy of the problem. Different ways of describing mesh quality and driving adaptivity using error estimators are considered including equidistribution according to edge second derivatives and minimisation of a functional with respect to the nodal solution and nodal coordinates. However, this work is for time independent problems with one equation, not a system of equations, and the functional approach is highly dependent on the equation used. Therefore, this area of work is not directly applicable to the Lagrangian plus remap solution of the Euler equations. Furthermore, by solving first with a Lagrangian step our mesh does already attempt to follow the anisotropic nature of the flow, for example collecting more nodes throughout the width of a shock. Directional subdivision would be a more productive approach to anisotropic problems solved using a Lagrangian or ALE method.

Triangular elements and anisotropic refinement are often combined because these meshes are naturally highly unstructured and disjoint nodes are easier to deal with. There are fewer examples of quadrilateral anisotropic finite elements even though they may provide the following benefits. The accuracy of calculations involving the subdivision of tetrahedral elements may be limited. When a tetrahedral's side is continually subdivided very sharp angles may be obtained, reducing the accuracy of the calculation. No such problem exists with quadrilateral elements as the element angles will not degenerate with side subdivision. Furthermore, quadrilateral or hexahedral elements are more suited to highly stretched grids and can be used with a standard grid generator.

Apel and Jimack [11] do use subdivision of rectangles with a reaction-diffusion test problem to investigate error indicators. However, again this work is dependent on the equation, does not consider a system of equations and is not applied to time dependent equations. Therefore, it is not directly applicable for the Lagrangian plus remap solution of the Euler equations.

In the paper by van der Vegt and van der Ven [71] unstructured anisotropic refinement of hexahedrons is considered by subdividing separately in each of the three directions. The adaptation criteria is complicated and based on all flow variables in order to capture all relevant flow phenomena without preference. Although this paper includes the key principle of subdividing hexahedrons it still uses tree like data structures, which we wish to avoid because of the expense of traversing up and down trees when the refinement is cell by cell. The mesh in van der Vegt and van der Ven's work never moves with the flow or r-refines, the coarse mesh is fixed. This is an Eulerian approach rather than the Lagrangian or ALE approach that is required in this work.

Anisotropic refinement of quadrilateral elements is also used to simulate viscous flow in a paper by Aftosmis [2]. This paper gives a good overview of different refinement triggers for feature detection and considers shock and smooth feature detection separately. It is in this paper that the sketch of the anisotropic refinement quadrant map appears which is generalised in this chapter to include derefinement tolerances. Similarly to van der Vegt and van der Ven, Aftosmis adopts an Eulerian mesh approach; the mesh never moves with the flow or r-refines, the coarse mesh is fixed.

Kallinderis and Baron [43] also use anisotropic refinement of quadrilaterals coupled with a multigrid method. Anisotropic and isotropic subdivision of cells is included by using a cell based connectivity array approach similar to that used in this thesis. Once again the mesh does not move with the flow; the coarse mesh is fixed.

Keats and Lien [44] in their paper on anisotropic refinement of quadrilateral elements on a Cartesian mesh discuss the problems of using a second order difference as a refinement sensor for anisotropic refinement.

This chapter extends the isotropic adaptive mesh ALE code to include anisotropic refinement, in the form of subdividing an element in one of its local directions.

6.1 Anisotropic refinement and derefinement

In contrast to subdivision on fixed Cartesian meshes, where the refinement follows the spatial coordinates, our refinement follows the element's local coordinates ξ and η. The

ALE formulation approximately aligns elements with the flow, by bisecting the element sides, the refinement will also align with the flow. The subdivision occurs in one direction for anisotropic refinement, or in both directions for isotropic refinement. As the elements already approximately align with the dominant directions anisotropic refinement is particularly efficient and effective, resolution can be focused in the direction it is required without unnecessarily refining in the other direction.

The change in density is used as the refinement criteria in this work. The refinement criteria is modified for anisotropic refinement by considering the change in density as having ξ and η components that can be considered to plot out a refinement quadrant map [2] as shown in Figure 6.1.

The change in density in the element's ξ direction is given by

$$|\Delta\rho_\xi| = \mathrm{MAX}\left[|\rho_r - \rho_i|, |\rho_i - \rho_l|\right], \tag{6.1}$$

where ρ_i is the coarse cell density, ρ_l is the coarse density in the left neighbour cell and ρ_r is the coarse density in the right cell. A similar equation applies for the η direction. If $|\Delta\rho|$ is greater than the refinement radius then the element is refined. The ratio of the density differences $|\Delta\rho_\eta|$ and $|\Delta\rho_\xi|$ is used to decide whether anisotropic or isotropic refinement is required [2]

$$\begin{aligned} \frac{|\Delta\rho_\eta|}{|\Delta\rho_\xi|} &< \tan(30^\circ) \Rightarrow \xi\text{-refinement} \\ \tan(30^\circ) < \frac{|\Delta\rho_\eta|}{|\Delta\rho_\xi|} &< \tan(60^\circ) \Rightarrow \text{isotropic refinement} \\ \frac{|\Delta\rho_\eta|}{|\Delta\rho_\xi|} &> \tan(60^\circ) \Rightarrow \eta\text{-refinement.} \end{aligned} \tag{6.2}$$

If the magnitude becomes lower than the derefinement radius the fine elements are derefined. The refinement and derefinement radii must differ otherwise elements may alternate between refinement and derefinement over the time steps, this is referred to as 'blinking'. Isotropic to anisotropic derefinement is triggered when the $\Delta\rho$ angle is below the derefinement angle (which must be below 30° to prevent 'blinking'). The anisotropic refinement quadrant map including the derefinement radius and angle is shown in Figure 6.1. An element is never allowed to refine from one anisotropic direction straight to the other anisotropic direction, since $\Delta\rho$ should always pass from one anisotropic direction region to the other anisotropic direction region through isotropic or coarse states.

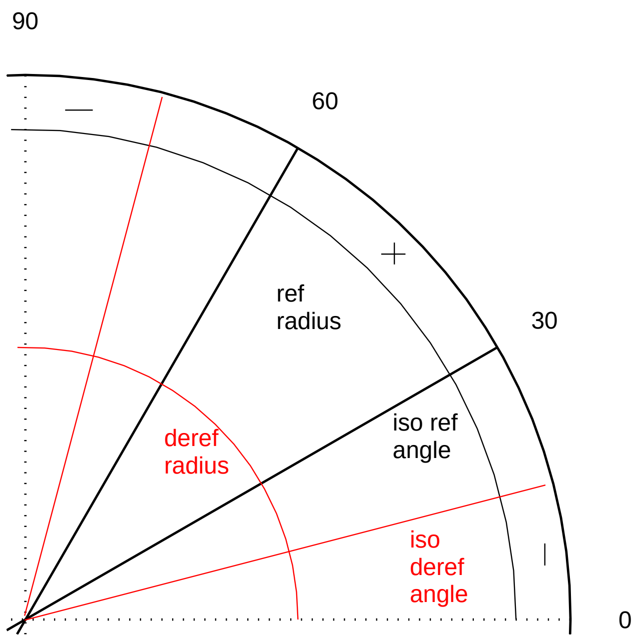

Figure 6.1: Change in density quadrant and the refinement regions.

The types of refinement we allow are:

- coarse to anisotropic
- coarse to isotropic
- anisotropic to isotropic.

The types of derefinement that we allow are:

- anisotropic to coarse
- isotropic to coarse
- isotropic to anisotropic.

Following the principles of the isotropic cell by cell refinement, the anisotropic and isotropic elements are inserted into the mesh. Again this forms the Dynamic Mesh, which consists of the finest resolution existing in each part of the domain. The Dynamic Mesh now contains anisotropic, isotropic and coarse elements.

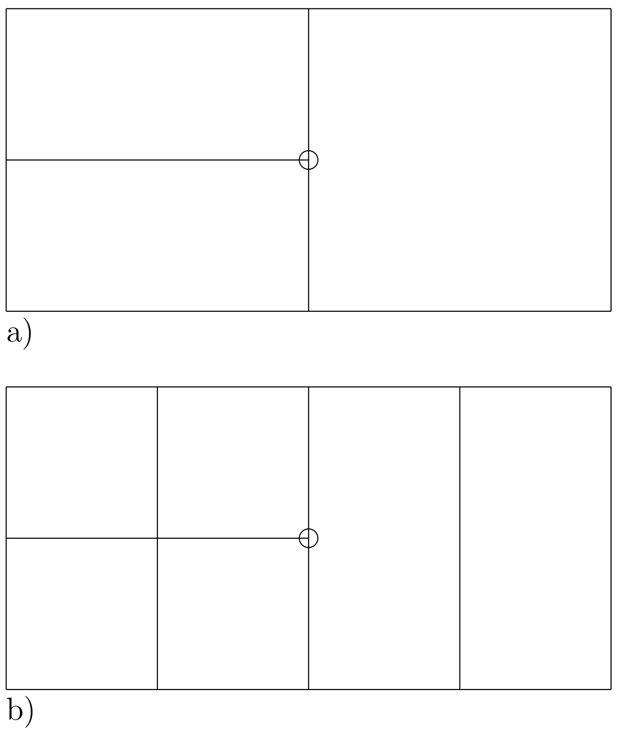

Figure 6.2: Diagram showing in a) disjoint node on anisotropic to coarse transition and in b) disjoint node at isotropic to anisotropic transition.

Disjoint or hanging nodes occur at resolution transitions, the transitions between anisotropic and coarse elements, and isotropic and anisotropic elements are shown in Figure 6.2. Only a 2:1 refinement ratio is allowed at each transition, this ensures that the resolution does not change too drastically and is a restriction implemented in many anisotropic papers such as Tan and Varghese [65] and Keats and Lien [44]. Therefore every disjoint node, considered as a non-dynamic point, has non-disjoint nodes (dynamic points) either side of it on the interface.

Due to the anisotropic refinement subdividing elements in one direction only, it is now possible for the resolution to change between an element and its neighbour without a disjoint node being created at that edge. Consider a coarse element with a horizontal neighbour that has been refined to increase the resolution in the ξ direction, as shown in Figure 6.3. The ξ-refined element has been subdivided through the top and bottom edges, introducing no disjoint node on the left edge shared by the coarse element. In Keats and Lien [44] restrictions are also placed on this type of transition ensuring that no change in resolution of more than 2:1 occurs from element to element.

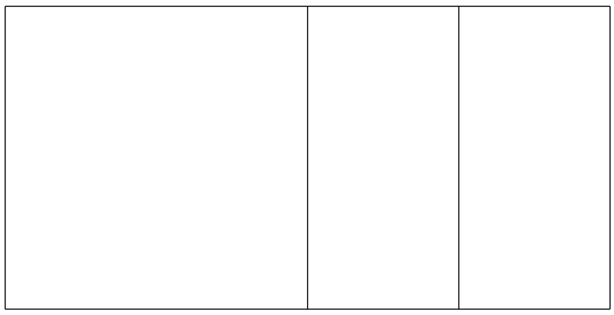

Figure 6.3: Diagram showing a coarse element with an anisotropically refined neighbour whose refinement is in the ξ direction.

With only one level of refinement as in this work only 2:1 changes can occur. However, generalisations to many refinement levels will require special considerations and careful level nesting criteria, this is discussed in Chapter 7.

As before the original connectivity arrays are retained and new dynamic connectivity arrays are created. We record the type of refinement for each original element: anisotropic ξ-refined, anisotropic η-refined, isotropic or coarse. The original and dynamic arrays are linked using an array that records for each original coarse element its dynamic element number if it is coarse, or the 2 anisotropic elements, or the 4 isotropic elements.

6.2 Solution transfer during refinement

Only the first order solution transfer has been generalised for anisotropic refinement, combining second order transfer with anisotropic refinement is a possible area for future work. Considering that only small improvements were seen with second order transfer and one refinement level, until multiple levels of refinement are investigated it is unclear whether the increased complexity involved in second order transfer can be justified.

Whenever an anisotropic element is isotropically refined the anisotropic dynamic element values are interpolated rather than derefining them to coarse values first. For cell centred variables, the new values are the piecewise constant values of the dynamic element being subdivided, shown in Figure 6.4 (for clarity we illustrate the steps of the method on

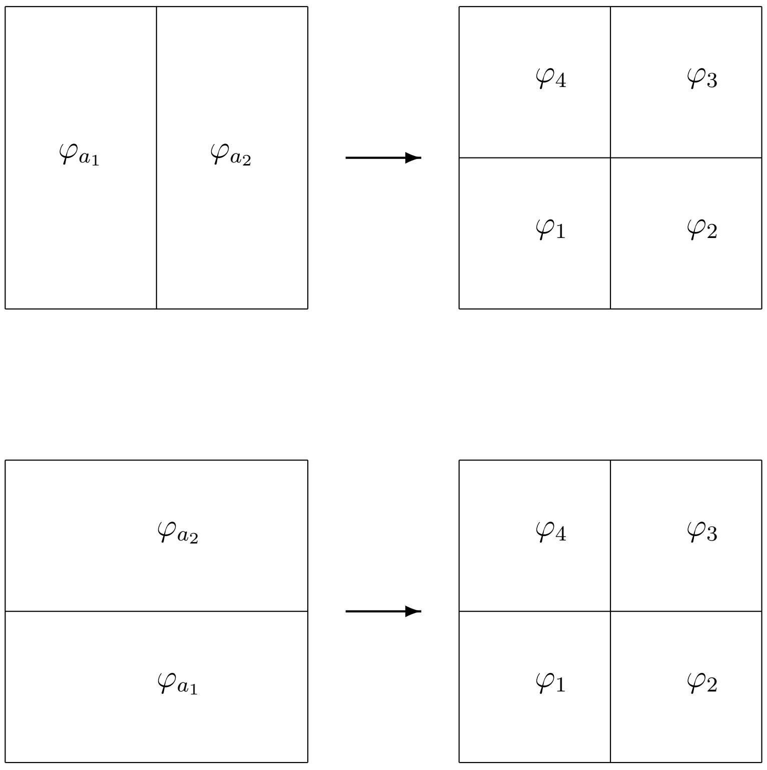

Figure 6.4: Diagram showing the isotropic refinement of anisotropic elements.

orthogonal grids, in reality the grids will consist of distorted quadrilaterals). In the case of refinement of ξ-refined elements to isotropic elements

$$\begin{aligned}\varphi_1 &= \varphi_{a_1}\\ \varphi_4 &= \varphi_{a_1}\\ \varphi_2 &= \varphi_{a_2}\\ \varphi_3 &= \varphi_{a_2}.\end{aligned} \tag{6.3}$$

In the case of refinement of η-refined elements to isotropic elements

$$\begin{aligned}\varphi_1 &= \varphi_{a_1}\\ \varphi_2 &= \varphi_{a_1}\\ \varphi_3 &= \varphi_{a_2}\\ \varphi_4 &= \varphi_{a_2}.\end{aligned} \tag{6.4}$$

For derefinement, the fine cell centred values are conservatively averaged to give the new coarse values. In particular for the isotropic to anisotropic coarsening shown in

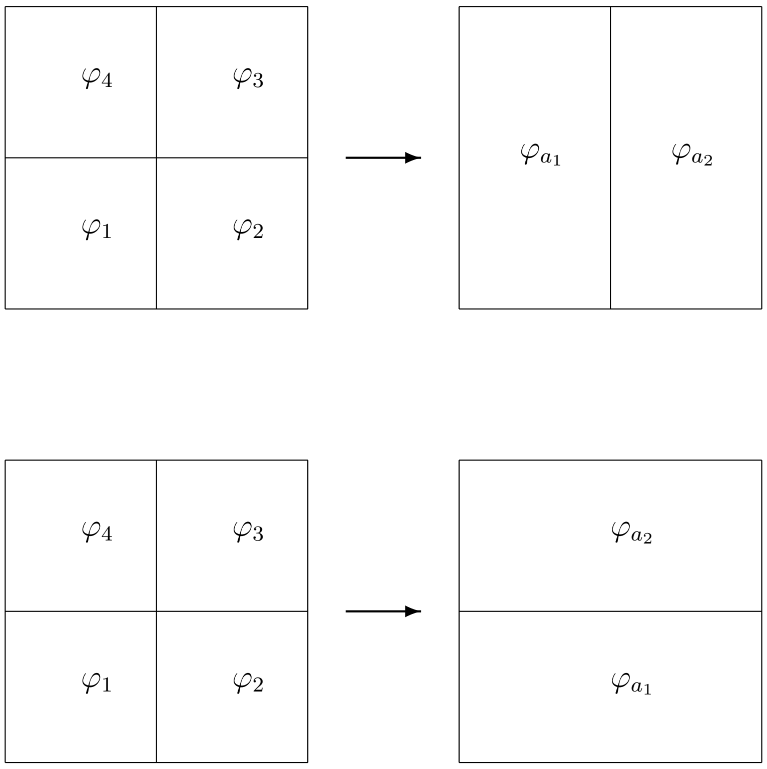

Figure 6.5: Diagram showing the derefinement of isotropic elements to anisotropic elements.

Figure 6.5, derefinement to ξ-refined elements gives

$$\begin{aligned} \varphi_{a_1} &= \frac{\varphi_1 X_1 + \varphi_4 X_4}{X_1 + X_4} \\ \varphi_{a_2} &= \frac{\varphi_2 X_2 + \varphi_3 X_3}{X_2 + X_3}, \end{aligned} \tag{6.5}$$

derefinement to η-refined elements gives

$$\begin{aligned} \varphi_{a_1} &= \frac{\varphi_1 X_1 + \varphi_2 X_2}{X_1 + X_2} \\ \varphi_{a_2} &= \frac{\varphi_3 X_3 + \varphi_4 X_4}{X_3 + X_4}, \end{aligned} \tag{6.6}$$

where X represents either volume or mass.

New nodal values are linearly interpolated from the coarse nodal values. When an anisotropic element is isotropically refined the existing positions and velocities are retained; they may not correspond to the bisection of the coarse sides. The new node in the middle, shown in Figure 6.6, takes the position of the intercept between the retained anisotropic line and the new perpendicular bisector,

$$\begin{aligned}
x_m &= \frac{(x_{f2}-x_{f1})(x_{a2}y_{a1}-x_{a1}y_{a2})-(x_{a2}-x_{a1})(x_{f2}y_{f1}-x_{f1}y_{f2})}{(x_{a2}-x_{a1})(y_{f2}-y_{f1})-(x_{f2}-x_{f1})(y_{a2}-y_{a1})}\\
y_m &= \frac{(y_{a2}-y_{a1})(x_{f1}y_{f2}-x_{f2}y_{f1})-(y_{f2}-y_{f1})(x_{a1}y_{a2}-x_{a2}y_{a1})}{(x_{a2}-x_{a1})(y_{f2}-y_{f1})-(x_{f2}-x_{f1})(y_{a2}-y_{a1})}.
\end{aligned} \tag{6.7}$$

The velocity for the middle node is given by interpolating between the existing anisotropic velocities by an amount given by the ratio of the new middle node distance, l_m, to the side's full length L

$$\mathbf{u}_m = \mathbf{u}_{a1} + \frac{l_m}{L}(\mathbf{u}_{a2} - \mathbf{u}_{a1}), \tag{6.8}$$

where

$$\begin{aligned}
l_m &= \sqrt{(x_m - x_{a1})^2 + (y_m - y_{a1})^2}\\
L &= \sqrt{(x_{a2} - x_{a1})^2 + (y_{a2} - y_{a1})^2}.
\end{aligned} \tag{6.9}$$

The nodes $a1$ and $a2$, shown in Figure 6.6, are those that already existed on the anisotropic line.

When derefinements take place the redundant nodal values are simply removed. The positions and velocities of the nodes still required for the anisotropic elements are unaltered, this ensures that these nodes will still align with neighbouring elements or remain slaved as disjoint nodes.

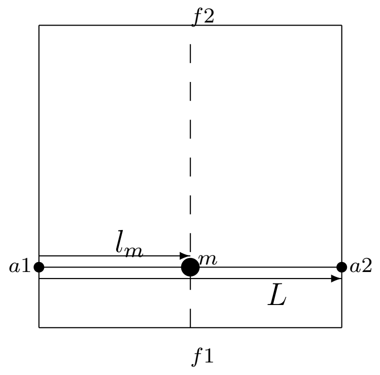

Figure 6.6: Diagram showing the interpolation procedure for the middle velocity.

6.3 Buffering

Buffering is required to ensure that during the Lagrangian step the features of interest do not move outside of the fine regions. If features were to cross out of the fine region this could cause oscillations as discussed in Section 3.5. For buffering, every primary element has the eight surrounding coarse elements also refined. An element is referred to as primary if its directional density changes are above the refinement thresholds for that type of element. With anisotropic refinement there are ξ-refined, η-refined and isotropic elements that make what type of element the buffer cell should be much more complicated. The method outlined below stops the anisotropic regions penetrating into the isotropic regions and tries to ensure that the solution, and not the mesh dictates the buffering. It also removes isolated anisotropic elements that could lead to oscillations and noise.

- If a buffer element is only surrounded by one type of refinement it becomes an element of that type.
- Any primary isotropic element has 8 isotropic cells around it. This may mean altering a primary anisotropic element to isotropic or setting a buffer cell to isotropic even though it may also have anisotropic neighbours. Figure 6.7 illustrates buffering for isotropic elements.
- If a buffer cell has both types of anisotropic neighbour its type depends on where the change in density lies on the anisotropic refinement quadrant map.
- A ξ-refined element butting up against an η-refined element can not change the η-refined element to a ξ-refined element and vice versa.
- If one type of primary anisotropic refinement is sandwiched between two elements of the other type of primary anisotropic refinement then all three elements become isotropic. This is shown in Figure 6.8.

6.4 The anisotropic Lagrangian method

The Lagrangian method requires no additional changes to include anisotropic refinement. When the Christensen artificial viscosity was generalised for isotropic refinement, the changes were implemented using the element structure around each disjoint node. Exactly the same structure exists around all disjoint nodes, hence when anisotropic refinement is included no further changes are required.

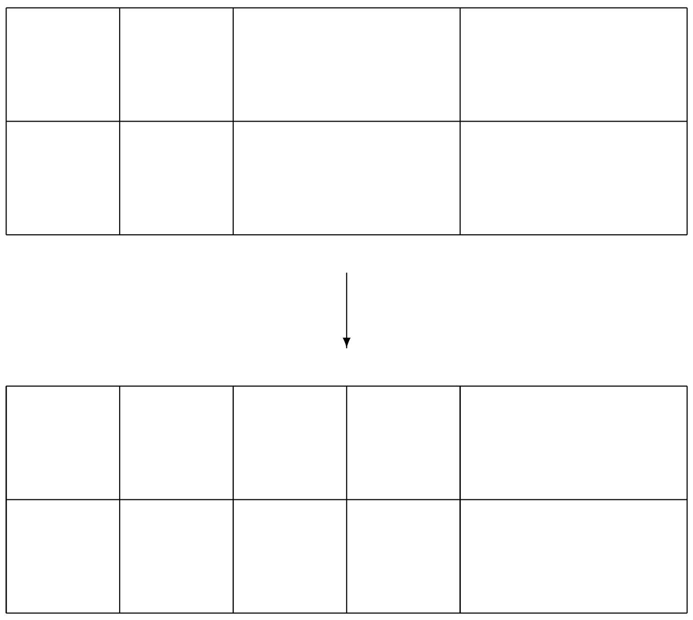
Figure 6.7: Buffering for a primary isotropic element can change an already anisotropic element.

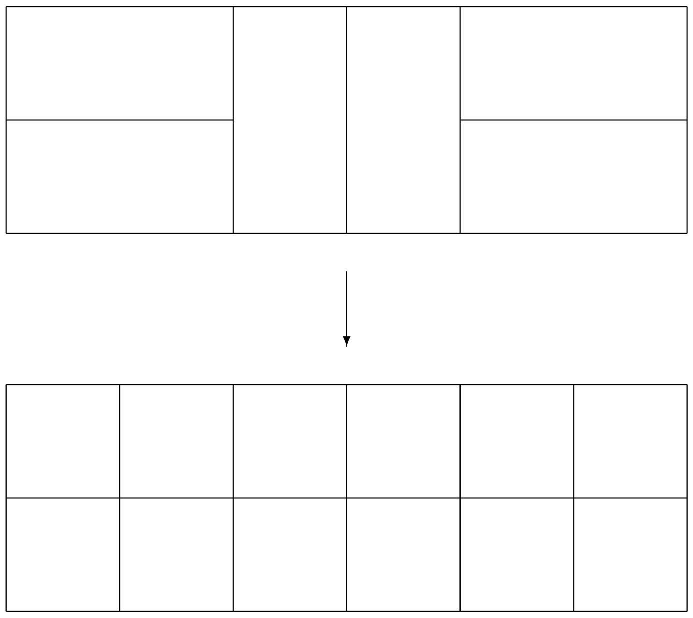
Figure 6.8: Buffering to stop anisotropic elements being sandwiched.

6.4.1 Square Sod problem

The anisotropic adaptive mesh Lagrangian method with one level of refinement was tested on the square Sod problem first discussed in Subsection 2.5.3. The problem was run up to t=0.1 with the anisotropic refinement applied on an initially 50×50 mesh. The adaptive mesh refinement value was 0.04 and the derefinement value was 0.025, the isotropic refinement angle was 30° and the derefinement angle was 25° from each axis. The same features as in the isotropic calculation were picked out for refinement, but the refinement is isotropic at the corners and anisotropic along the edges of the square just as expected, see Figure 6.9.

Referring to Figure 6.9 we see that the mesh is smooth and untangled. The isotropic, anisotropic and fine 100×100 calculations give very similar density contours, see Figures 3.16, 3.17 and 6.10.

The anisotropic calculation required substantially less elements, as shown in Figure 6.11. Plotting whether each of the originally coarse base elements remained coarse, became anisotropically refined or became isotropically refined allowed the types of refinement occurring over the calculation to be analysed. Figure 6.12 shows that the majority of elements remained coarse and the main proportion of the refinement occurring was anisotropic, as would be hoped since large parts of the problem involved one dimensional Sod shock tube behaviour. The number of isotropic refinements increased as the radius of the shock grew because the circumference of the corners was increasing.

The anisotropic calculation only required 383 steps, while the isotropic took 545 steps. The total number of elements, given by elements multiplied by the number of steps taken with that element number, reduced by just under a half for the anisotropic calculation. The cpu time has been substantially reduced to half that of the isotropic calculation and 15% of that of the uniformly fine calculation. In other words, the anisotropic adaptive calculation runs nearly seven times faster than a uniformly fine calculation.

This test problem shows that the anisotropic refinement can make a very significant difference to the number of elements and run time without any loss in accuracy. A substantial reduction in run time and elements is also achieved compared to the isotropic refinement, verifying that it was definitely beneficial to include an anisotropic refinement capability into the adaptive code. Excellent anisotropic results have been obtained that illustrate the power of combining anisotropic subdivision of elements with the Lagrangian mesh's alignment to the flow directions.

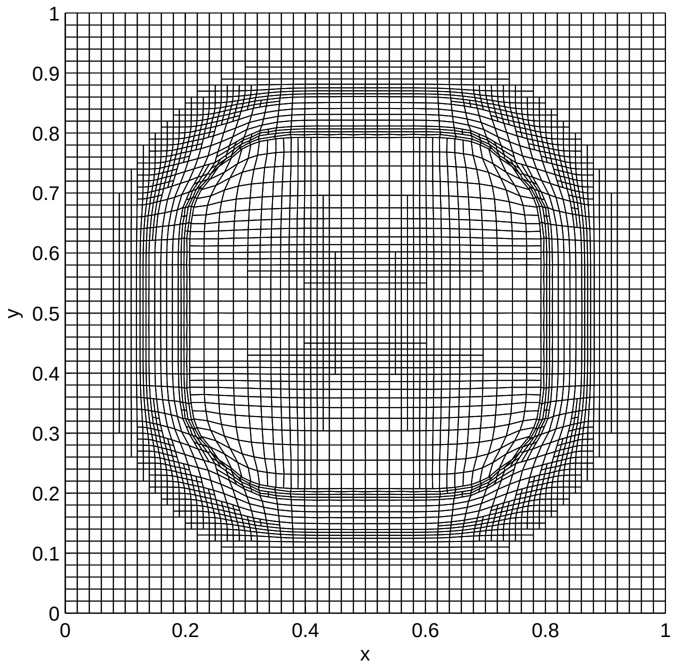

Figure 6.9: Square Sod problem Lagrangian anisotropic adaptive mesh at t=0.1.

6.5 Equipotential relaxation and anisotropic refinement

In this section the adaptive equipotential mesh relaxation strategy is generalised to make it compatible with anisotropic refinement. We derive a mesh relaxation strategy that does not cause areas of finer meshing to diffuse into the areas of coarser meshing as the node spacing moves toward equality, not only do coarse-fine interfaces behave correctly but also isotropic-anisotropic and anisotropic-coarse interfaces as well. Following the isotropic case all disjoint nodes are slaved, nodal stencils are not required for disjoint nodes and the equipotential relaxation formulae are not applied to these nodes.
In the isotropic case: Method 1 reverted to a coarse nodal stencil if the nodal stencil contained both coarse and fine nodal spacings; Method 2 used the same stencil for all nodes by identifying the different coarse and fine spacings and taking them into

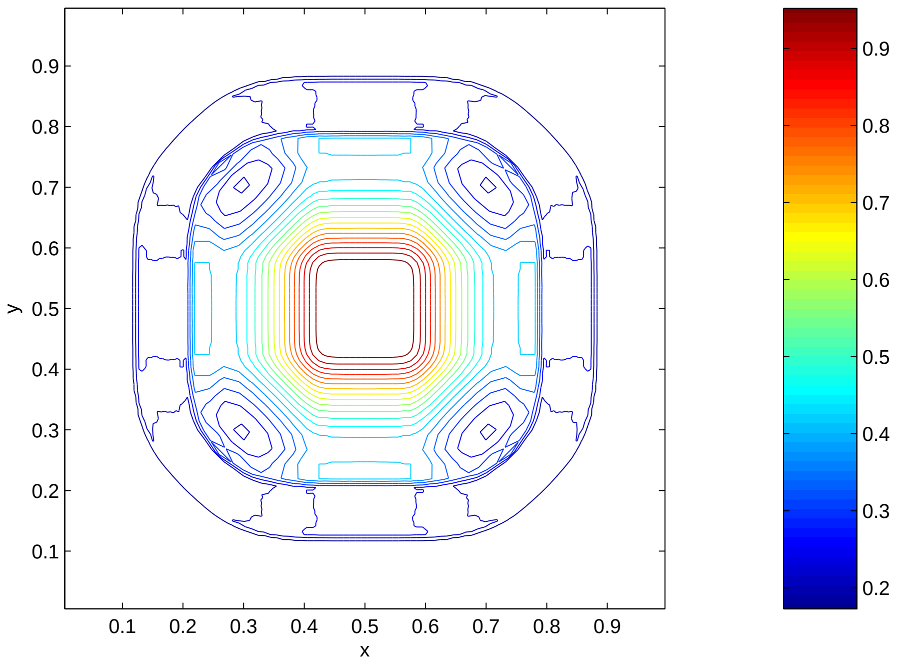

Figure 6.10: Square Sod problem anisotropic adaptive mesh calculation density contours at t=0.1.

account in the discretisation of the derivatives. Although Method 1 was generalised for anisotropic refinement the results obtained when relaxing the Riemann problem mesh showed significant mesh curvature where the isotropic refinement transferred to anisotropic along the shock, see Figure 6.13. Switching to the coarse stencil where the refinement changes from isotropic to anisotropic causes those nodes to move more than their neighbouring nodes in accordance with the greater relaxation of the coarse mesh, as discussed in Section 4.1.1. This introduces a curve in the meshing where we should only be seeing one-dimensional behaviour along the shock. This is an undesirable effect that the anisotropic Method 2 (outlined below) does not show, see Figure 6.14, and Section 4.1.1 has already illustrated that Method 2 relaxes in a manner more comparable with that of a uniformly fine mesh.

Method 2 was generalised by setting separate C's for the horizontal and vertical spacings for the nodes at the corners of the 9 point stencil and calculating the weights for the x-coordinate relaxation using the horizontal Cx's and the weights for the y-coordinate relaxation using the vertical Cy's.

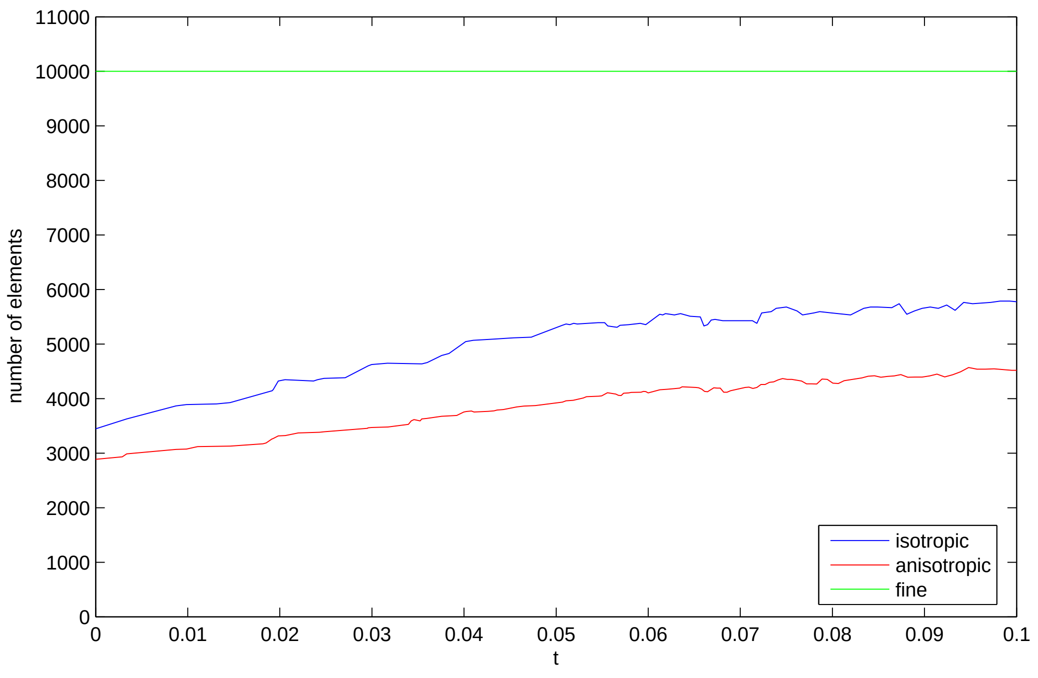

Figure 6.11: Square Sod problem comparing the number of elements required over calculation time for anisotropic method.

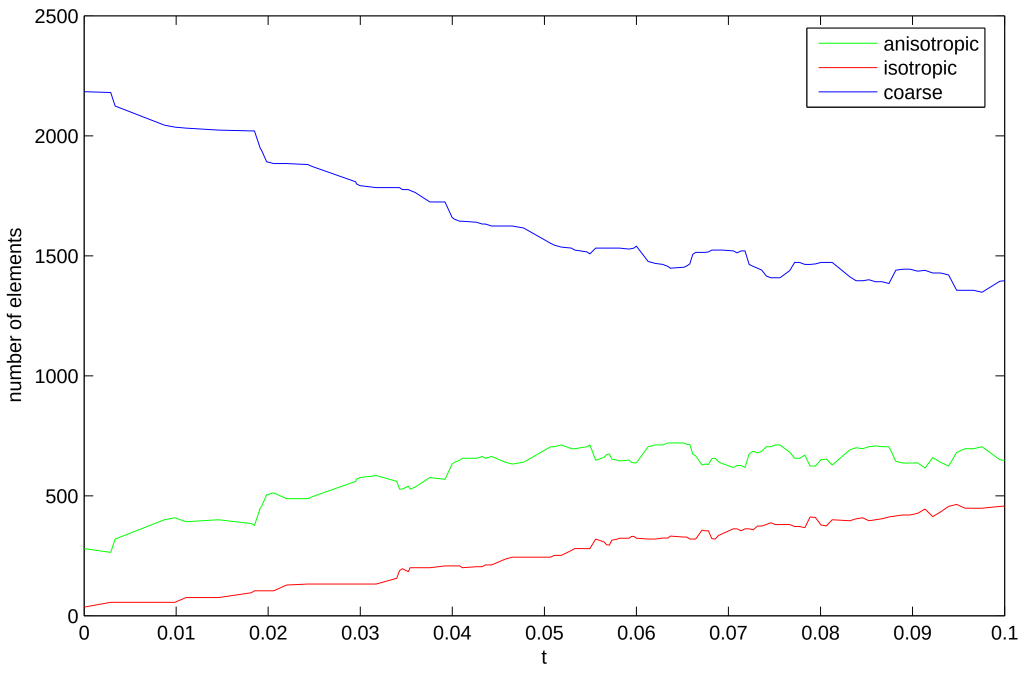

Figure 6.12: Number of each type of element over time for the square Sod problem.

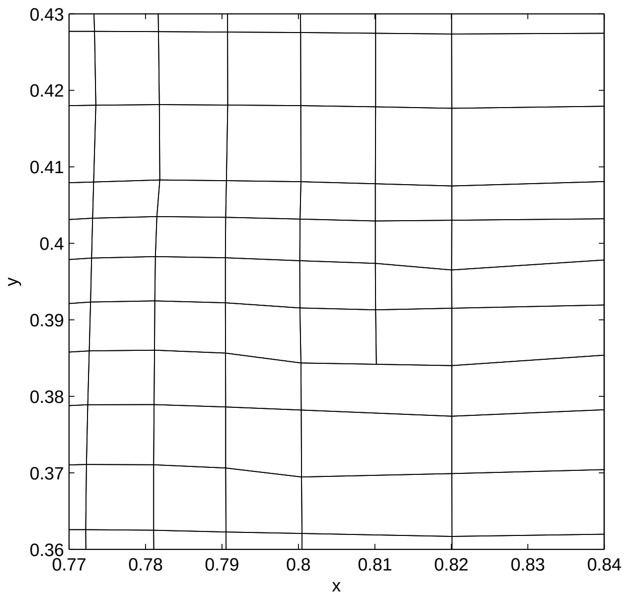

Figure 6.13: Method 1 equipotential relaxation of Riemann problem t=0.15, showing curvature problem.

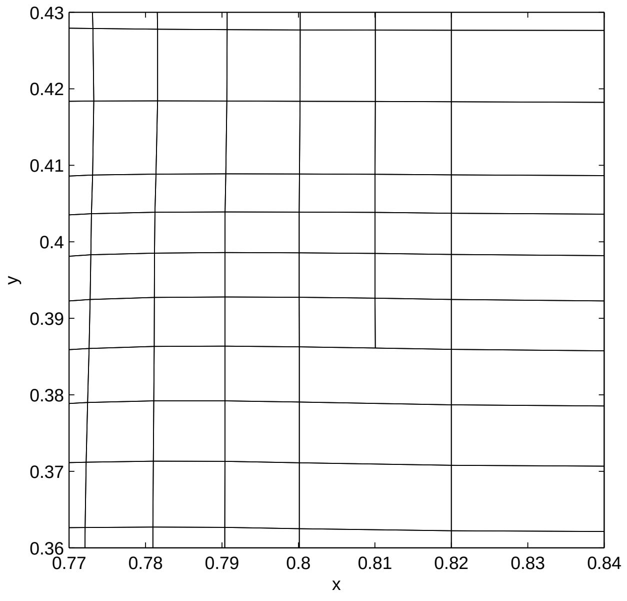

Figure 6.14: Method 2 equipotential relaxation of Riemann problem t=0.15, no curvature problem.

6.5.1 Anisotropic generalisation of Method 2

The approach for the isotropic mesh was to take the nodal length spacings into account by giving each node a length value, $C_{\phi,\psi+1}$ etc., denoting if it was coarse or finely spaced

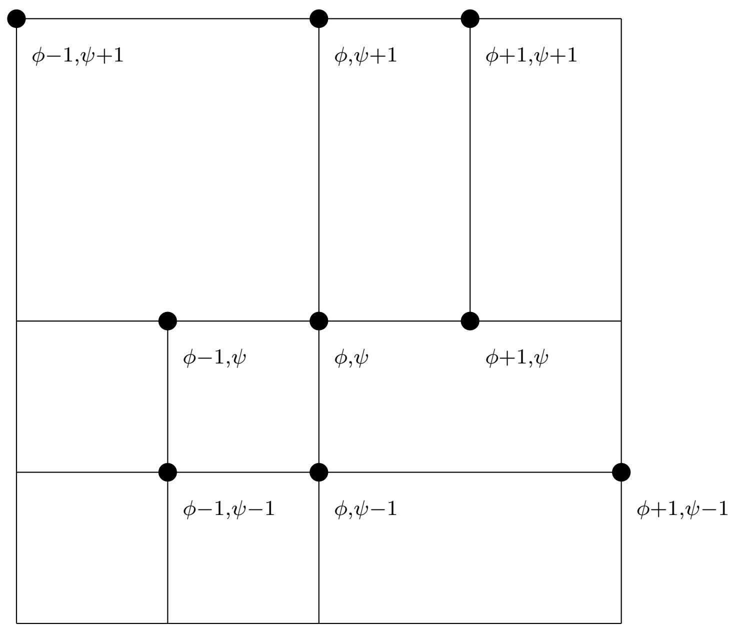

Figure 6.15: Diagram showing the stencil for a node with coarse, isotropic and anisotropic neighbouring elements.

from the centre node. Taylor expansions involving these length values were then used to discretise the derivatives. The resulting relaxation equation weights then depended on the length values.

In the isotropic case only one length value was required to express the nodal spacing in both directions of the node from the centre. This will not be the case with anisotropic refinement as shown in Figure 6.15. The top, bottom, left and right nodes do have nodal spacing in one direction only. However, the $(\phi+1,\psi+1)$ node, the $(\phi+1,\psi-1)$ node, the $(\phi-1,\psi+1)$ node and the $(\phi-1,\psi-1)$ node can have coarse or fine nodal spacings from the centre node that are different in each direction. We want to prevent the finer regions from spreading into the coarser regions, and we also want to retain the careful balance between the diagonal weights.

A first attempt at this problem which used Taylor expansions of the form

$$x_{\phi+1,\psi+1} = x_{\phi,\psi} + Cx_{\phi+1,\psi+1} l(\frac{\partial x}{\partial \phi}) + Cy_{\phi+1,\psi+1} h(\frac{\partial x}{\partial \psi})..., \tag{6.10}$$

failed to keep the delicate balance between the diagonal weights and lead to a large amount of diagonal movement for nodes at corner configurations of anisotropic and isotropic elements. These nodes were moved substantially outside of their neighbours,

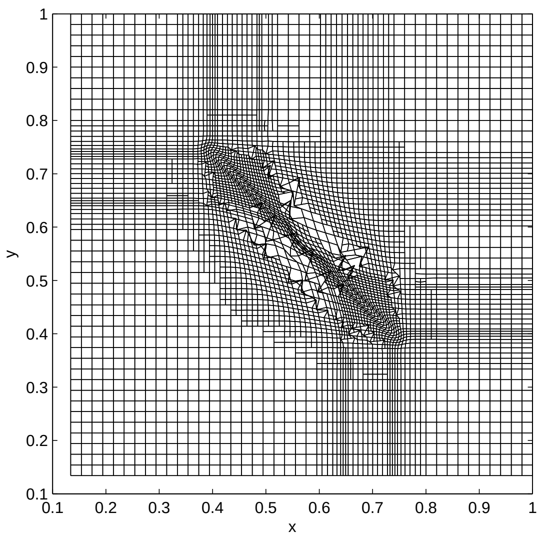

Figure 6.16: Two-dimensional Riemann problem anisotropic relaxed mesh, first attempt to generalise Method 2 resulted in extreme tangling, 1 iteration at t=0.15.

see Figure 6.16, when implemented on the Riemann problem where the skewed meshes lead to a significant mixed derivative.

The successful approach was to derive different relaxation weights for the x-coordinate and the y-coordinate separately. The non mixed second order derivatives $\frac{\partial^2 x}{\partial \phi^2}$, $\frac{\partial^2 y}{\partial \phi^2}$, $\frac{\partial^2 x}{\partial \psi^2}$ and $\frac{\partial^2 y}{\partial \psi^2}$ and the first order derivatives all involve three nodes that lie on the same equipotential line, for example Figure 6.17 showing the ϕ =constant line for the $\frac{\partial^2 x}{\partial \psi^2}$ derivative. We know that the difference formulae weights will only depend on the spacing along that equipotential line, so that for the four nearest neighbour nodes only one C is required. The equations for these derivatives are exactly the same as the isotropic case, regardless of whether we are relaxing the x-coordinate or the y-coordinate. As the first order derivatives are unchanged, α, β and γ are the same as in the isotropic case.

Considering the mixed derivative again, the diagonal nodes do not necessarily lie on the same equipotential lines, see Figure 6.18, and so defining the spacing is more complicated. The first anisotropic relaxation formulae failed because the diagonal terms were out of balance. When the new x position is calculated it depends only on the nodal x positions and not on the y positions. We can consider the nodal positions on the x-line as illustrated in Figure 6.19.

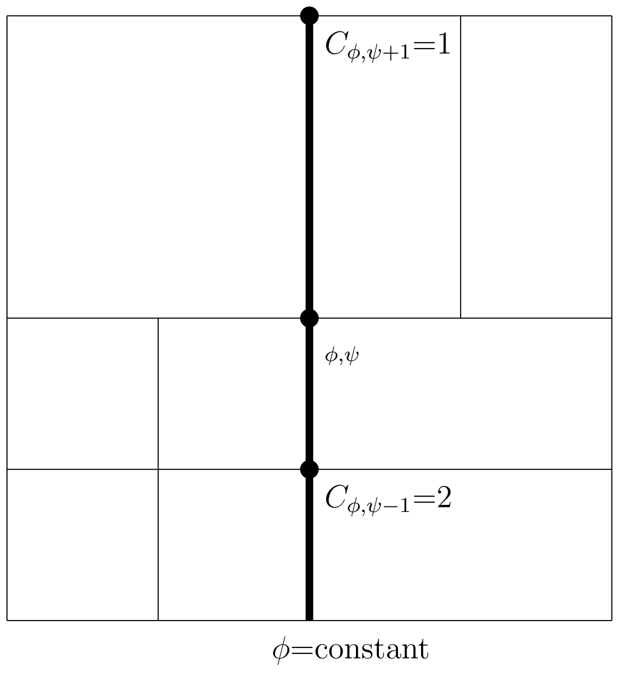

Figure 6.17: Diagram showing the stencil for the non mixed second derivative.

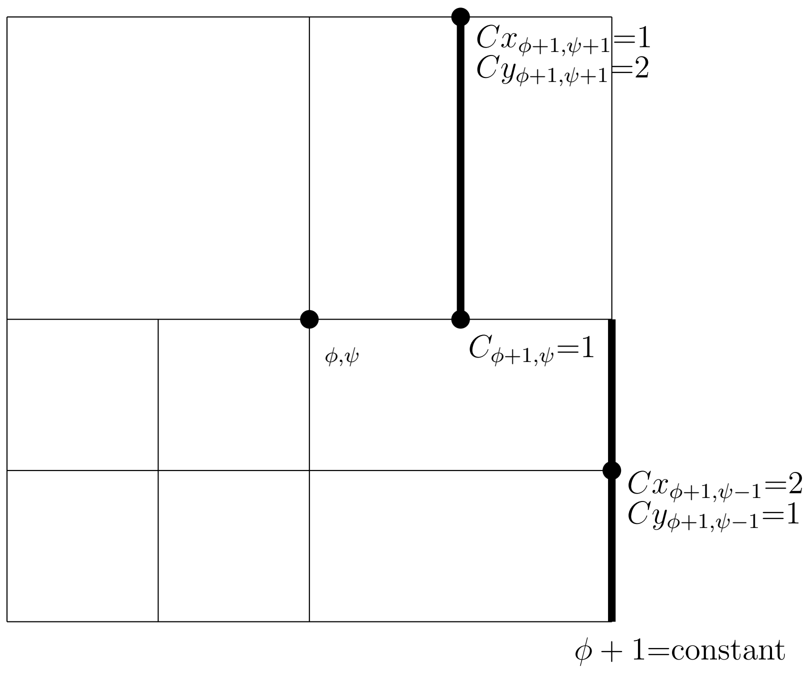

Figure 6.18: Diagram showing the broken equipotential line involving the diagonal nodes on the right for the mixed second derivative.

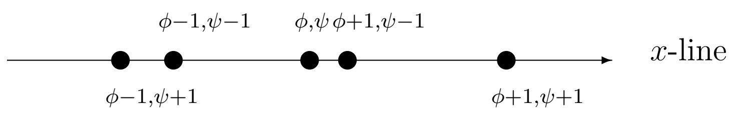

Figure 6.19: Diagram showing the diagonal nodes projected on to the x-line.

The change in position specified by the relaxation formulae is

$$x_{new} - x_{old} = \sum_{i=1,8} r_i x_i - x_{old}. \tag{6.11}$$

Let

$$x_i = x_{old} + \Delta x_i + \delta x_i, \tag{6.12}$$

where the Δx_i represents the correct desired mesh spacing away from the central node, this is referred to as the "perfect" mesh and can be anisotropic and coarse or fine, and δx_i is the perturbation from the "perfect" mesh that we wish to reduce by applying the equipotential equations. This gives

$$x_{new} - x_{old} = \sum_{i=1,8} r_i \Delta x_i + \sum_{i=1,8} r_i \delta x_i + (\sum_{i=1,8} r_i - 1) x_{old}, \tag{6.13}$$

and using $\sum_{i=1,8} r_i = 1$ we get

$$x_{new} - x_{old} = \sum_{i=1,8} r_i \Delta x_i + \sum_{i=1,8} r_i \delta x_i. \tag{6.14}$$

We wish this quantity to be small otherwise the node will have moved too far. In fact since for the "perfect" mesh there should be no movement we require

$$\sum_{i=1,8} r_i \Delta x_i = 0, \tag{6.15}$$

which will give a small amount of node movement as long as the δx_i are small

$$x_{new} - x_{old} = \sum_{i=1,8} r_i \delta x_i. \tag{6.16}$$

From the isotropic case and the failure to balance the diagonals in the previous attempt we know that the node $(\phi+1, \psi+1)$ term balances the $(\phi-1, \psi-1)$ term and the node $(\phi+1, \psi-1)$ term balances the $(\phi-1, \psi+1)$ term.

In the "perfect" mesh part we require

$$r_{\phi+1,\psi+1} \Delta x_{\phi+1,\psi+1} + r_{\phi-1,\psi-1} \Delta x_{\phi-1,\psi-1} = 0, \tag{6.17}$$

similarly for the other diagonal terms. Now in the "perfect" mesh

$$\Delta x_{\phi+1,\psi+1} = -\kappa \Delta x_{\phi-1,\psi-1}, \tag{6.18}$$

where because our "perfect" mesh can be coarse, fine, anisotropic or isotropic κ is a scaling factor that takes this into account, for example it will equal $\frac{1}{2}$ when the $(\phi-1,\psi-1)$ is coarser than the $(\phi+1,\psi+1)$ spacing. Substituting this in the above gives

$$r_{\phi+1,\psi+1}\kappa\Delta x_{\phi-1,\psi-1} = r_{\phi-1,\psi-1}\Delta x_{\phi-1,\psi-1}, \tag{6.19}$$

so that

$$\kappa r_{\phi+1,\psi+1} = r_{\phi-1,\psi-1}. \tag{6.20}$$

In a "perfect" mesh with rectangular boundaries and x and y perpendicular to the boundaries then x will be parallel with ϕ and y will be parallel with ψ. Therefore for the x-coordinate, κ represents the ϕ spacing

$$\kappa = \frac{C\phi_{\phi+1,\psi+1}}{C\phi_{\phi-1,\psi-1}} = \frac{Cx_{\phi+1,\psi+1}}{Cx_{\phi-1,\psi-1}}. \tag{6.21}$$

In conclusion we have

$$r_{\phi-1,\psi-1} = \frac{Cx_{\phi+1,\psi+1}}{Cx_{\phi-1,\psi-1}} r_{\phi+1,\psi+1}. \tag{6.22}$$

This is exactly what we have from the isotropic equations if $C_{\phi+1,\psi+1} = Cx_{\phi+1,\psi+1} = C\phi_{\phi+1,\psi+1}$ a measure of the ϕ spacing. The isotropic mixed derivative only required diagonal terms and the centre node not the nearest neighbours, unlike the failed attempt equation. This analysis shows that it is very important that the diagonal terms balance by satisfying the above equation. In the method that failed we had

$$r_{\phi-1,\psi-1} = \frac{\sqrt{Cx_{\phi-1,\psi-1}Cy_{\phi+1,\psi+1}}}{\sqrt{Cx_{\phi+1,\psi+1}Cy_{\phi-1,\psi-1}}} \frac{Cx_{\phi+1,\psi+1}}{Cx_{\phi-1,\psi-1}} r_{\phi+1,\psi+1}, \tag{6.23}$$

so these terms did not balance and the nodes were moved out of their nodal neighbour region.

If we consider the isotropic equations they were derived using

$$\begin{aligned} x_{\phi+1,\psi+1} &= x_{\phi,\psi} + C_{\phi+1,\psi+1} l (\frac{\partial x}{\partial \phi}) + C_{\phi+1,\psi+1} h (\frac{\partial x}{\partial \psi}) \\ &+ \frac{C^2_{\phi+1,\psi+1} l^2}{2} (\frac{\partial^2 x}{\partial \phi^2}) + C^2_{\phi+1,\psi+1} lh (\frac{\partial^2 x}{\partial \psi^2}) + \frac{C^2_{\phi+1,\psi+1} h^2}{2} (\frac{\partial^2 x}{\partial \psi^2}) ..., \end{aligned} \tag{6.24}$$

and so the weights for the x-coordinate must be derived using

$$\begin{aligned} x_{\phi+1,\psi+1} &= x_{\phi,\psi} + Cx_{\phi+1,\psi+1} l(\frac{\partial x}{\partial \phi}) + Cx_{\phi+1,\psi+1} h(\frac{\partial x}{\partial \psi}) \\ &+ \frac{Cx^2_{\phi+1,\psi+1} l^2}{2}(\frac{\partial^2 x}{\partial \phi^2}) + Cx^2_{\phi+1,\psi+1} lh(\frac{\partial^2 x}{\partial \psi^2}) + \frac{Cx^2_{\phi+1,\psi+1} h^2}{2}(\frac{\partial^2 x}{\partial \psi^2}) .... \end{aligned} \tag{6.25}$$

Note the subtle changes from the unsuccessful Taylor expansion. Although only the Cx's are required for the diagonal nodes, these must appear with both the ϕ and ψ terms in the Taylor expansion. Otherwise the balance of the diagonal nodes will be lost and the relaxation equation of a node with uniformly coarse nodal spacings in both directions will not reduce to the equipotential relaxation equations.

Using the Taylor expansion to discretise the x-coordinate mixed derivative results in

$$\begin{aligned} \frac{\partial^2 x}{\partial \phi \psi} &= -\frac{Cx_{\phi+1,\psi-1} Cx_{\phi-1,\psi+1} - Cx_{\phi+1,\psi+1} Cx_{\phi-1,\psi-1}}{2lh Cx_{\phi+1,\psi+1} Cx_{\phi+1,\psi-1} Cx_{\phi-1,\psi+1} Cx_{\phi-1,\psi-1}} x_{\phi,\psi} \\ &+ \frac{x_{\phi+1,\psi+1}}{2lh(Cx_{\phi-1,\psi-1} Cx_{\phi+1,\psi+1} + Cx^2_{\phi+1,\psi+1})} \\ &- \frac{x_{\phi-1,\psi+1}}{2lh(Cx_{\phi-1,\psi+1} Cx_{\phi+1,\psi-1} + Cx^2_{\phi-1,\psi+1})} \\ &- \frac{x_{\phi+1,\psi-1}}{2lh(Cx_{\phi-1,\psi+1} Cx_{\phi+1,\psi-1} + Cx^2_{\phi+1,\psi-1})} \\ &+ \frac{x_{\phi-1,\psi-1}}{2lh(Cx_{\phi-1,\psi-1} Cx_{\phi+1,\psi+1} + Cx^2_{\phi-1,\psi-1})}. \end{aligned} \tag{6.26}$$

Substituting into the inverse equation results in the following formulae for the new x nodal coordinate:

$$\begin{aligned} x_{\phi,\psi} &= \frac{\alpha_{\phi,\psi}(\frac{x_{\phi-1,\psi}}{C_{\phi-1,\psi}} + \frac{x_{\phi+1,\psi}}{C_{\phi+1,\psi}})}{D(C_{\phi,\psi+1} + C_{\phi,\psi-1})^2 (C_{\phi-1,\psi} + C_{\phi+1,\psi})} \\ &+ \frac{\gamma_{\phi,\psi}(\frac{x_{\phi,\psi-1}}{C_{\phi,\psi-1}} + \frac{x_{\phi,\psi+1}}{C_{\phi,\psi+1}})}{D(C_{\phi+1,\psi} + C_{\phi-1,\psi})^2 (C_{\phi,\psi-1} + C_{\phi,\psi+1})} \\ &+ \frac{\beta_{\phi,\psi}\left(\frac{(\frac{x_{\phi-1,\psi+1}}{Cx_{\phi-1,\psi+1}} + \frac{x_{\phi+1,\psi-1}}{Cx_{\phi+1,\psi-1}})}{(Cx_{\phi-1,\psi+1} + Cx_{\phi+1,\psi-1})} - \frac{(\frac{x_{\phi+1,\psi+1}}{Cx_{\phi+1,\psi+1}} + \frac{x_{\phi-1,\psi-1}}{Cx_{\phi-1,\psi-1}})}{(Cx_{\phi+1,\psi+1} + Cx_{\phi-1,\psi-1})}\right)}{D(C_{\phi+1,\psi} + C_{\phi-1,\psi})(C_{\phi,\psi+1} + C_{\phi,\psi-1})}, \end{aligned} \tag{6.27}$$

where D is given by

$$\begin{aligned} D &= \frac{\alpha_{\phi,\psi}}{(C_{\phi,\psi+1} + C_{\phi,\psi-1})^2 (C_{\phi-1,\psi} C_{\phi+1,\psi})} \\ &+ \frac{\gamma_{\phi,\psi}}{(C_{\phi+1,\psi} + C_{\phi-1,\psi})^2 (C_{\phi,\psi-1} C_{\phi,\psi+1})} \\ &+ \frac{\beta_{\phi,\psi}(Cx_{\phi+1,\psi+1} Cx_{\phi-1,\psi-1} - Cx_{\phi+1,\psi-1} Cx_{\phi-1,\psi+1})}{\Pi(C_{\phi+1,\psi} + C_{\phi-1,\psi})(C_{\phi,\psi+1} + C_{\phi,\psi-1})}, \end{aligned} \tag{6.28}$$

with

$$\Pi = Cx_{\phi+1,\psi+1}Cx_{\phi+1,\psi-1}Cx_{\phi-1,\psi+1}Cx_{\phi-1,\psi-1}. \tag{6.29}$$

The expression for the y-coordinate has the same form but all of the Cx's are replaced by Cy's in both the $y_{\phi,\psi}$ equation and the expression for D. Notice that the l's and h's cancel out because the derivative denominator multiplied by the α, β, or γ denominator always gives an l^2h^2 denominator, which is canceled throughout. Again the $\alpha_{\phi,\psi}$ etc. are the original Winslow-Crowley terms.

The method automatically reduces to the isotropic case if the Cx's are equal to the Cy's and reduces to the original Winslow-Crowley relaxation equations if the nodal spacings are also all the same. Again the method is only first order when the stencil has mixed spacings. However, Method 2 has the advantage that the same approach is applied to all nodes (except disjoint nodes) and the node's nearest neighbours are always used reducing the possibility that a node will be moved too far outside its neighbouring nodes.

6.5.2 Convergence

The anisotropic equipotential relaxation was tested on the refined square quasi-shock problem that was introduced in Section 4.1.1. Anisotropic Method 2 prevented the fine meshing from spreading into the coarser areas as shown in Figure 6.20. The results for a number of Gauss-Jacobi iterations were compared to those obtained using the same number of iterations on a uniformly fine mesh and the difference between them was plotted in Figure 6.21. For completeness Method 1 was also tested and the difference from the uniformly fine mesh was found to be two times worse than for Method 2. Anisotropic Method 2 has higher errors than for the isotropic Method 2 although this might be expected as the anisotropic refinement introduces more resolution transitions. Anisotropic equipotential relaxation Method 2 was also tested on the Riemann problem as in Section 4.1.1. The Lagrangian mesh at t=0.15 was used as the initial mesh and 10 Gauss-Jacobi iterations of Method 2 were applied. The results were exceedingly promising, see Figure 6.22, and showed that the finer regions were prevented from spreading into any coarser areas. In conclusion anisotropic Method 2 successfully relaxes the anisotropic Dynamic Mesh and behaves in a manner close to the relaxation of a uniformly fine mesh. Method 1 has been shown to be far less successful than Method

2 and was abandoned at this stage.

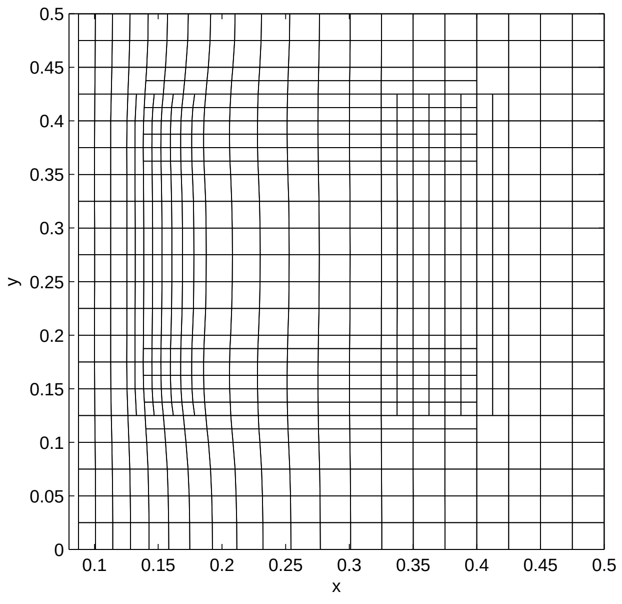

Figure 6.20: Equipotential anisotropic mesh relaxation Method 2, 20 Gauss-Jacobi iterations.

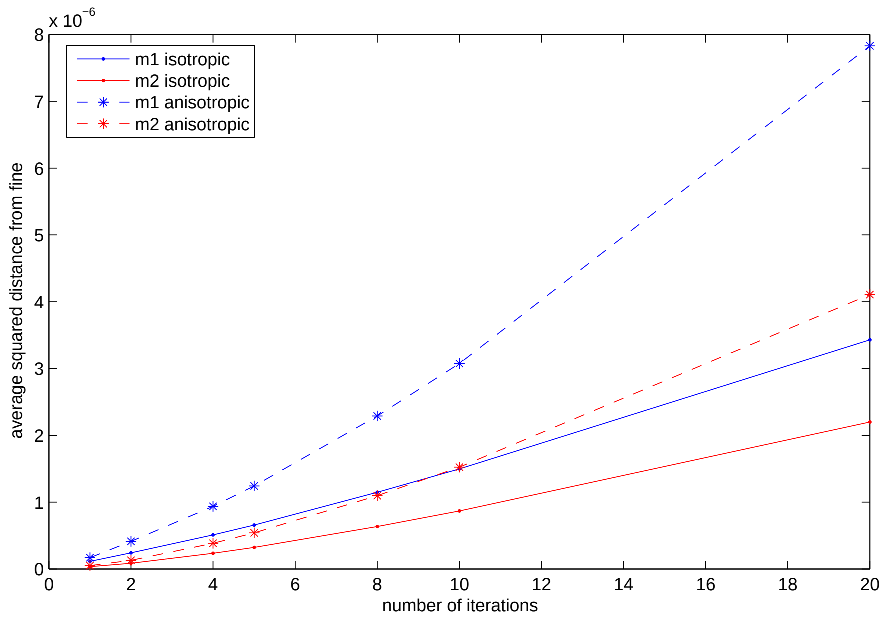

Figure 6.21: Plot of average squared distance from relaxed fine mesh for different stencil methods.

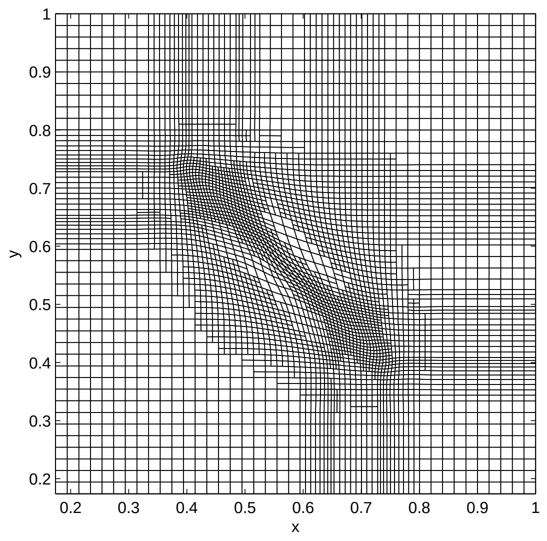

Figure 6.22: Two-dimensional Riemann problem anisotropic relaxed mesh, 10 iterations of Method 2 at t=0.15.

6.6 Anisotropic advection

The advection of cell centred variables requires no additional changes for anisotropic meshes; those made for the isotropic unstructured Dynamic Mesh are sufficient. This is because the previous changes at resolution transitions depended on the element arrangement around disjoint nodes. Even with anisotropic refinement a disjoint node still has one larger element on one side and two smaller elements on the other side, and so the slope calculation and neighbouring flux alterations for the isotropic refinement developed in Section 4.2.1 carry through unchanged.

Only a few changes are required for momentum advection on an anisotropically refined mesh. The dual mesh is defined similarly to Section 4.2.2, the nodal weight associated with an anisotropic element will be a quarter of the element's mass plus at most only one redistributed disjoint node contribution as the element has only been refined in one direction. The dual mesh within an anisotropic element never has two redistributed contributions from disjoint nodes, therefore the refinement corner case, shown in Figure 4.19 does not occur and no diagonal flux passing occurs. Only the dual mesh around isotropic elements can be associated with diagonal flux passing.

The nodal fluxes for the dual mesh lines can be calculated as before by considering if

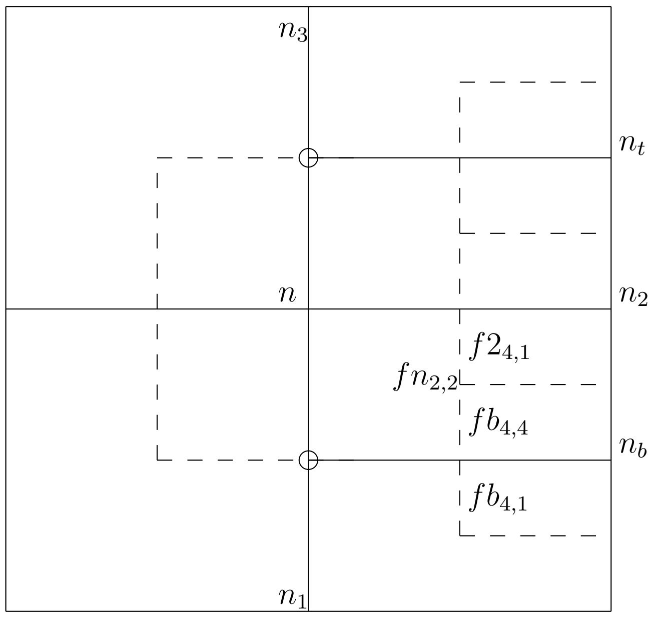

Figure 6.23: Diagram of the neighbour nodes, n's, and the neighbour mass fluxes, f's, for anisotropic refinement.

the dual line runs through a disjoint node and along an element side, does not run along an element side and cuts the whole of an element, or does not run along an element side and cuts half way across an element.

When calculating neighbouring nodal mass fluxes we proceed as before but with two new additions. Consider the dual mesh around the anisotropic elements shown in Figure 6.23 for the case when n_b and n_t are not disjoint nodes. We proceed as for the isotropic interface case (Figure 4.20), except that the neighbouring nodes are not fine nodes introduced in the middle of the top and bottom coarse sides and in the centre (as in the isotropic case) they are now coarse nodes, i.e. n_2, and new nodes on the right sides, i.e. n_b. If n_b and n_t are disjoint nodes the dual mesh aligns perfectly as shown in Figure 6.24 and nothing need be done.

Calculating slopes on the dual mesh is again done by using different slope dual meshes: within refined regions the fine slope dual mesh is used, for coarse nodes at resolution transitions the coarse slope dual mesh is used. The momentum fluxes can then be calculated following the principles in Section 4.2.2. Any difference between the width of the flux dual cell and the slope dual cell is taken into account by taking a ratio of the slope. The nodal velocity is interpolated out to the edge of the flux dual cell. As mentioned previously the dual mesh around the anisotropic elements can not contain a

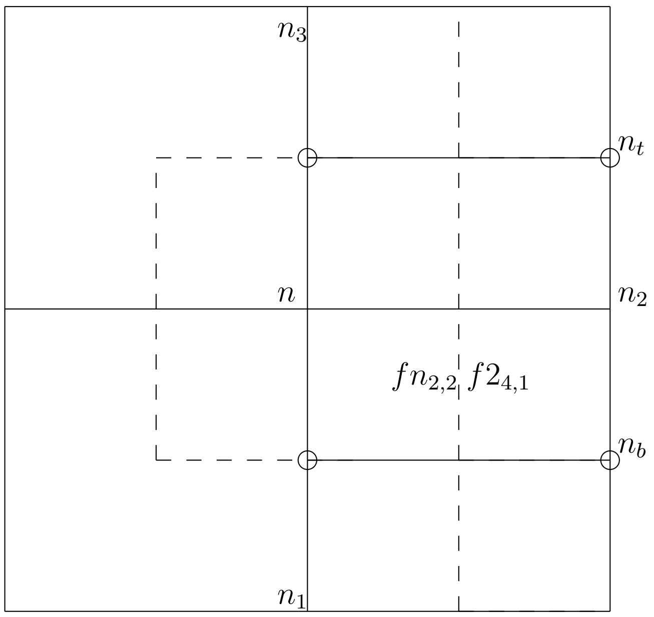

Figure 6.24: Diagram of the neighbour nodes, n's, and the neighbour mass fluxes, f's, for anisotropic refinement when n_b and n_t are disjoint nodes.

resolution corner, therefore we only need to worry about diagonal flux passing for the dual mesh around isotropic elements. In conclusion, very few changes are required to include anisotropic advection.

6.7 ALE results for the anisotropic adaptive method

We now present results to verify the performance of the ALE anisotropic adaptive method. The anisotropic adaptive calculations were run with only one level of anisotropic and isotropic mesh refinement, generalisation to many refinement levels is discussed in Chapter 7.

6.7.1 Radial Sod problem

The radial Sod problem considered in Section 2.5.4 was run to t=0.25 using the anisotropic adaptive mesh method and an initially 50×50 mesh, with a refinement parameter of 0.04 and a derefinement parameter of 0.025. The isotropic refinement angle was 30° and the derefinement angle was 25° from each axis. One relaxation iteration of the length weighted anisotropic equipotential method was used per time step.

Refined regions occurred around the rarefaction fan, contact and shock, with derefinement in between these features. The anisotropic mesh is shown in Figure 6.25, isotropic refinement only occurred within 15-20 degrees of the diagonal where the gradient would be expected to vary in both directions, near the axes the refinement was anisotropic. The mesh shown in Figure 6.25 is smooth, untangled and the finer regions have not spread into any coarser areas illustrating that the anisotropic mesh relaxation performs excellently.

The anisotropic calculation density contours, see Figure 6.26, are very similar to those of the isotropic calculation shown in Figure 4.31. Both results compare very favourably with the fine calculation see Figure 2.23, with only small differences occurring at the edges of the rarefaction fan and contact. The loss in radial symmetry and noise experienced in all the calculations between the rarefaction fan and contact is due to the implementation of the initial conditions on the quadrilateral mesh forming a stepped radius rather than a perfect quarter circle.

A very fine one-dimensional approximate solution [84] is compared against the anisotropic adaptive mesh ALE results in Figure 6.27 and the results are very similar to the uniformly fine calculation results (Figure 2.24).

The error norms for the anisotropic results were calculated as in Section 4.3.1 and are shown in Table 6.1. This verifies that the anisotropic calculation has comparable errors to the isotropic calculation. Both the isotropic and anisotropic calculation errors are nearly as good as the fine calculation error, and are much better than the coarse error. The reduction in the number of elements is illustrated in Figure 6.28. The anisotropic adaptive calculation takes slightly fewer time steps than the uniformly fine calculation, and the anisotropic adaptive calculation only requires 32% of the fine calculation's number of elements over the entire calculation time. The types of refinement occurring over the calculation time are plotted in Figure 6.29 and this shows that the majority of the elements remain coarse. Even in this problem, which contains a shock that curves rather than aligns with the mesh, the amount of anisotropic refinement outweighs the number of isotropic refinements and has resulted in a substantial reduction in the number of elements.

The anisotropic calculation runs 7.8 times faster than the uniformly fine calculation, where the isotropic calculation only runs 6.6 times faster. The anisotropic method has resulted in an excellent speed up even with only one level of refinement.

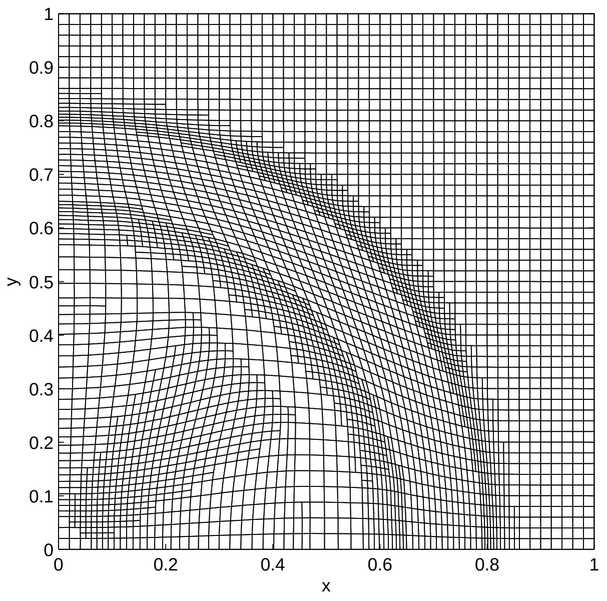

Figure 6.25: Radial Sod problem anisotropic adaptive ALE mesh at t=0.25.

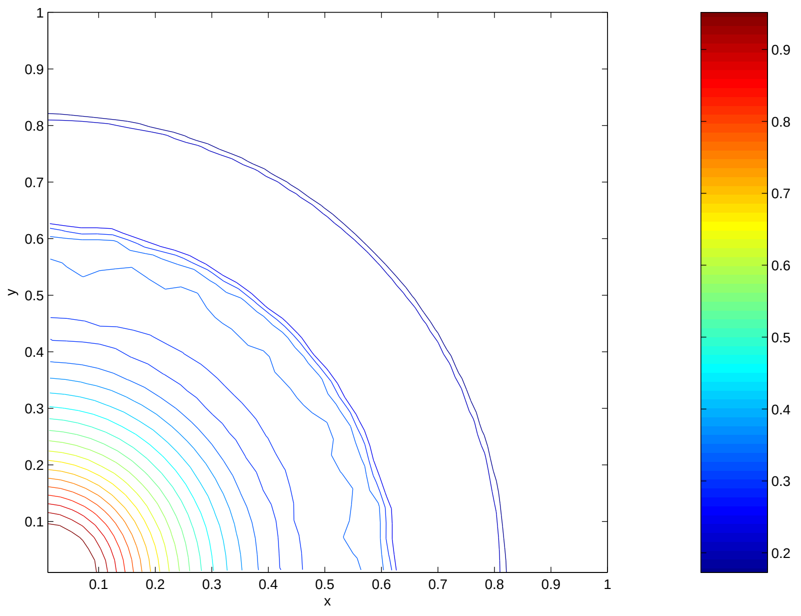

Figure 6.26: Radial Sod problem anisotropic calculation density contours at t=0.25.

6.7.2 Taylor-Sedov blast wave

The Taylor-Sedov blast wave, discussed in Section 2.5.5, was run to t=0.1 using the anisotropic adaptive mesh method with an initial mesh of 32×32, the refinement pa-

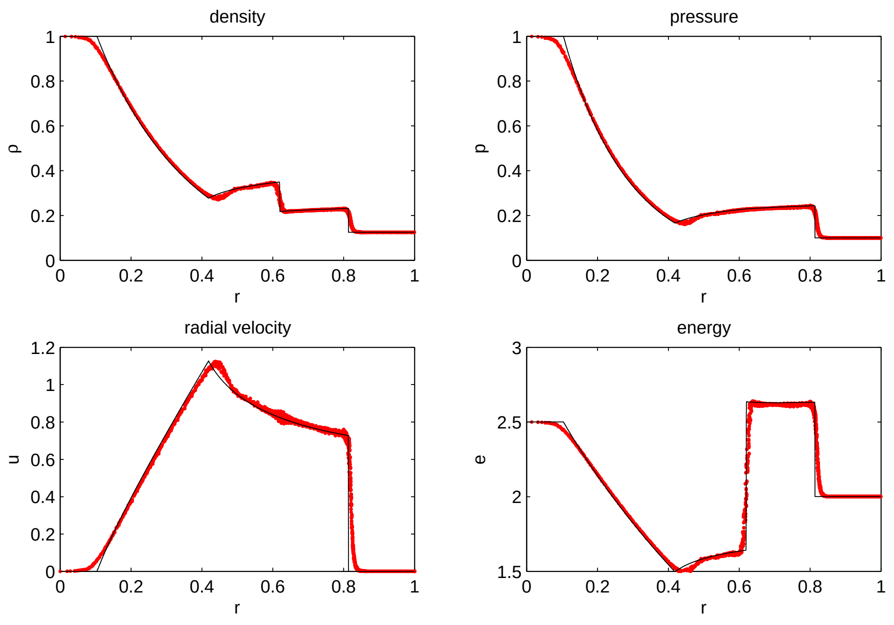

Figure 6.27: Radial Sod problem density, pressure, radial velocity and specific internal energy for anisotropic solution at t=0.25.

Calculation	$\|e\|_1$	$\|e\|_2$
fine	0.0044	0.0105
isotropic	0.0046	0.0109
anisotropic	0.0046	0.0107
coarse	0.0072	0.0139

Table 6.1: Errors for fine, isotropic, anisotropic and coarse radial Sod calculations t=0.25.

rameter was 0.25 and the derefinement parameter was 0.20. The isotropic refinement angle was 30° and the derefinement angle was 25° from each axis. One iteration of the length weighted anisotropic equipotential method was used per time step.

The adaptive mesh is shown in Figure 6.30 and shows the refinement that occurs around the shock radius. The refinement between 25° and 65° is isotropic, closer to the axes we have anisotropic refinement as desired. There is some radial asymmetry in the refinement but this is probably the result of initialising the blast wave with a square of energy.

The density contours for the anisotropic adaptive calculation, shown in Figure 6.31,

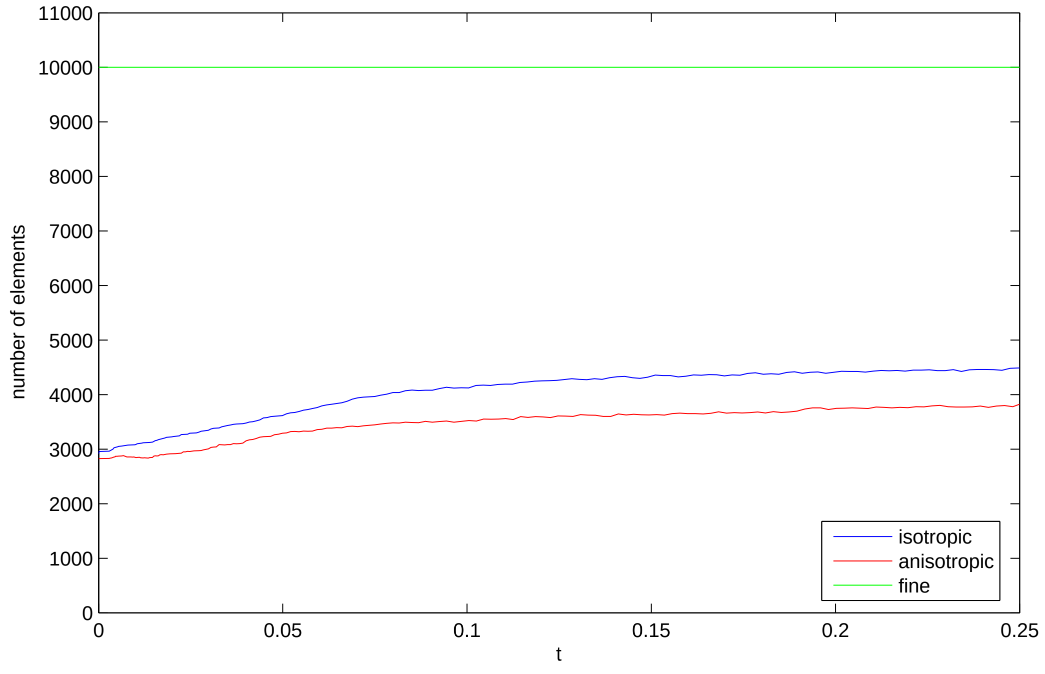

Figure 6.28: Radial Sod problem comparing the number of elements required over calculation time.

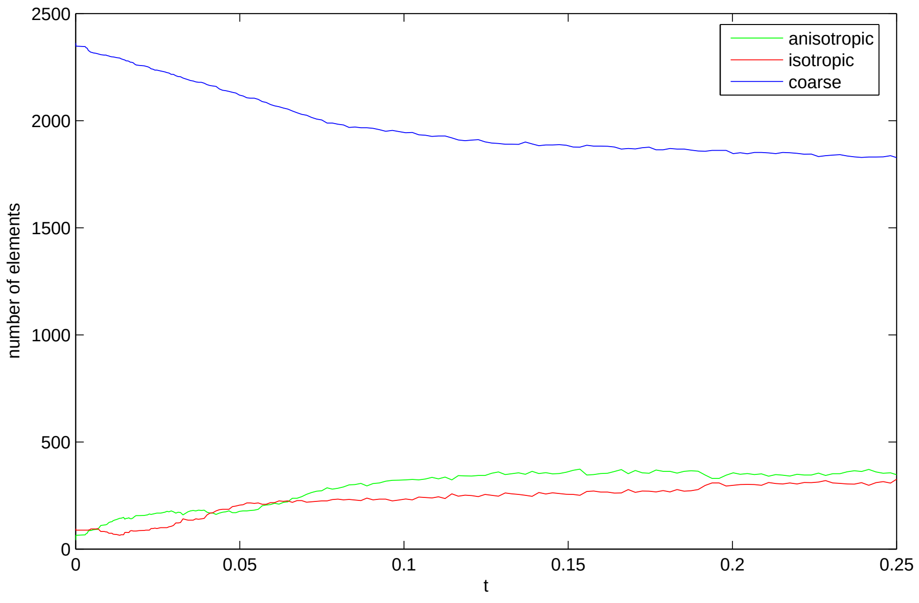

Figure 6.29: Number of each type of element over time for the radial Sod problem.

and the uniformly fine calculation compare very favourably. It is easier to compare the anisotropic, isotropic and uniformly fine results when the density is plotted with respect to radius, see Figure 6.32. Again the results for the uniformly fine calculation and the anisotropic adaptive calculation are very similar and both agree well with the similarity solution. The anisotropic calculation reached a peak value of 3.793 that is closer to the predicted value of 4, in comparison the isotropic calculation reached a peak value of 3.731 and the fine calculation reached 3.735. The shock radius ranges from 0.600 to 0.608, which is further to the left than the isotropic and fine results and a little less than the self similar solution radius. It is probably to be expected that as the problem is not particularly anisotropic the use of anisotropic refinement may have reduced the resolution in more isotropic regions affecting the positioning of the shock radius slightly. However, the difference between the anisotropic shock radius and the uniformly fine radius is not a substantial difference considering the reduction in number of elements and the use of first order solution transfer.

The reduction in the number of elements achieved using the adaptive method is shown in Figure 6.33. The anisotropic adaptive calculation only requires 29% of the uniformly fine calculation's total number of elements. The majority of the refinement was isotropic at early times and anisotropic at later times, see Figure 6.34. The anisotropic adaptive calculation runs nine times faster than the uniformly fine calculation.

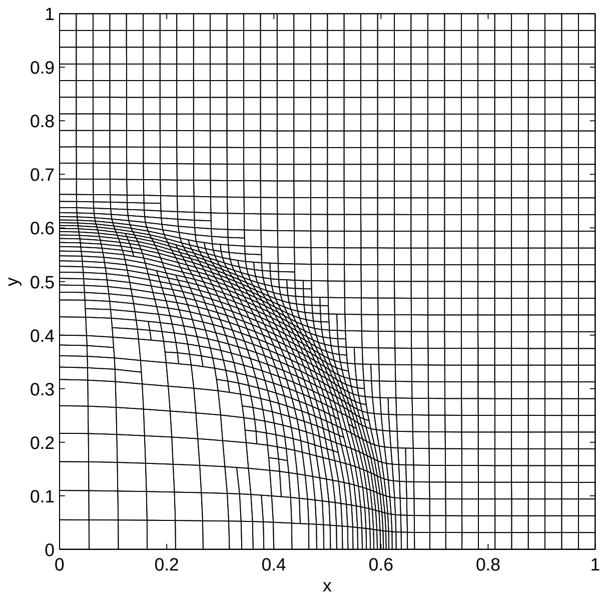

Figure 6.30: Sedov problem anisotropic mesh at t=0.1.

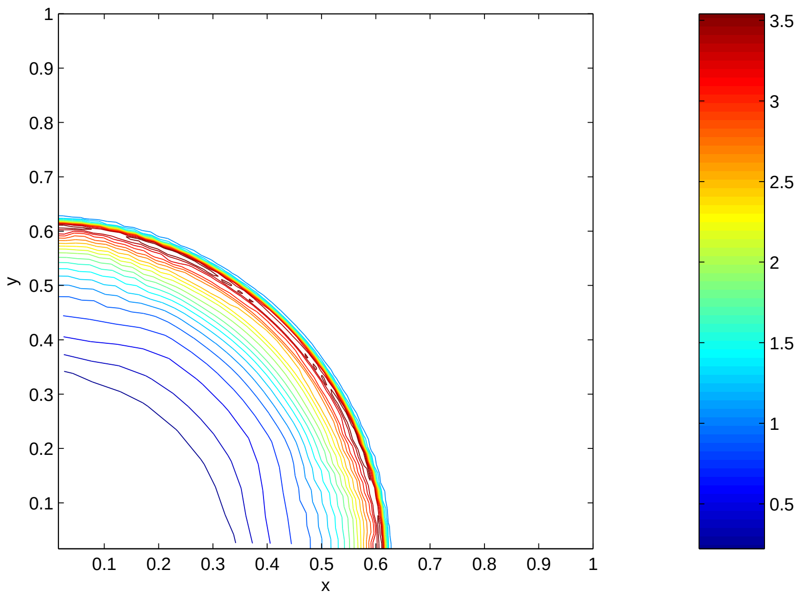

Figure 6.31: Sedov problem anisotropic calculation density contours at t=0.1.

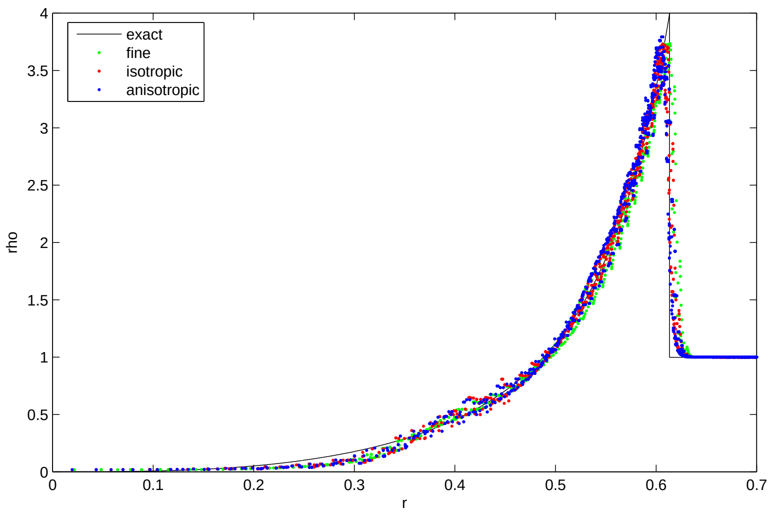

Figure 6.32: Sedov problem radial density at t=0.1.

6.7.3 Two-dimensional Riemann problem

The problem, see Section 2.5.2, was run with anisotropic refinement on a 50 × 50 initial mesh. One iteration of the nodal length weighted equipotential method was employed per time step. The refinement threshold was 0.15 and the derefinement value was 0.13. The isotropic refinement angle was 30° and the derefinement angle was 25° from each

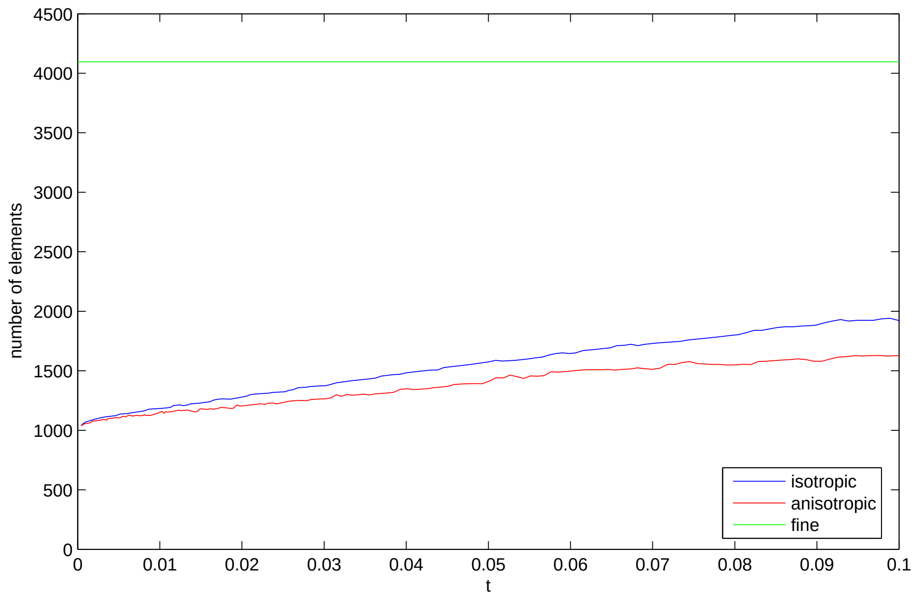

Figure 6.33: Sedov problem comparing the number of elements required over calculation time.

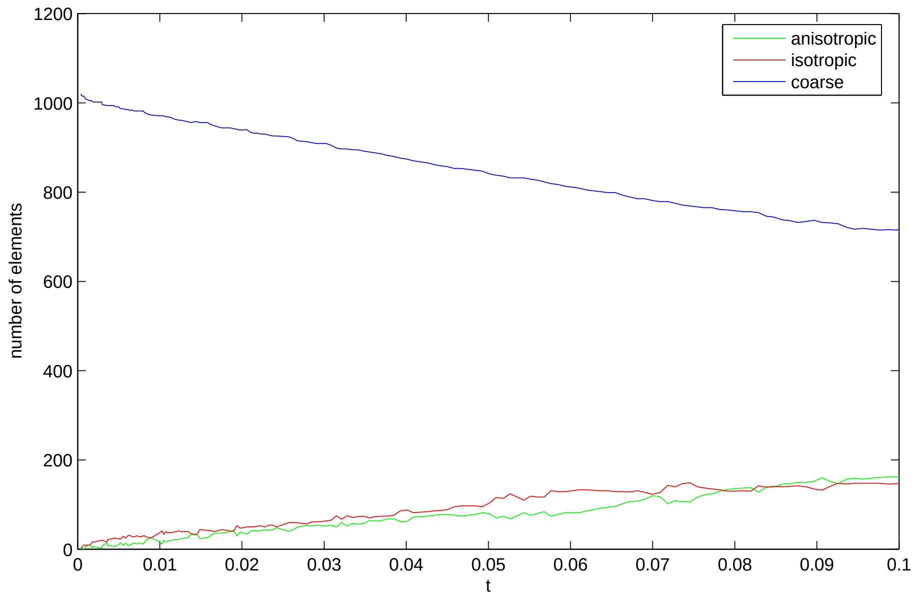

Figure 6.34: Number of each type of element over time for the Sedov problem.

axis.

The anisotropic method performs very well on this problem, as illustrated by comparing Figure 2.17 with Figure 6.35. The majority of the refinement is anisotropic. Isotropic refinement only occurs where the shocks intersect and where the circumference of the oval does not align with the direction of the mesh. Within the high density oval cells have derefined completely. The density contours, see Figure 6.36, are comparable with those achieved with the isotropic refinement, see Figure 4.42, and compare favourably with those of the uniformly fine calculation, see Figure 4.41. Again there is some loss of detail within the high density oval but the shocks and oval circumference have the resolution of the fine mesh.

Figure 6.37 shows that the anisotropic calculation requires significantly less elements per time step. The majority of the refinement was anisotropic, see Figure 6.38. The number of isotropic refinements increased slightly as the oval became larger. The anisotropic adaptive calculation only requires 34% of the fine calculation's number of elements over the entire calculation time. This problem highlights the significant reduction in calculation time that can be achieved with anisotropic refinement, the isotropic calculation only runs 5.5 times faster than the uniformly fine calculation, while the anisotropic calculation runs 8.6 times faster.

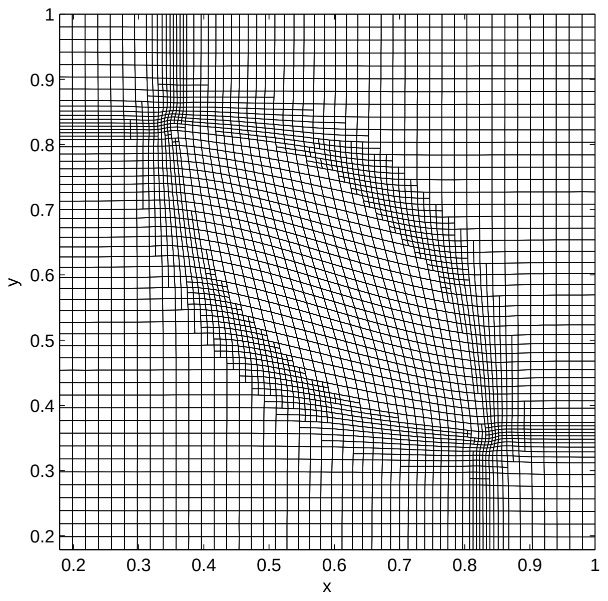

Figure 6.35: Two-dimensional Riemann problem anisotropic mesh at t=0.2.

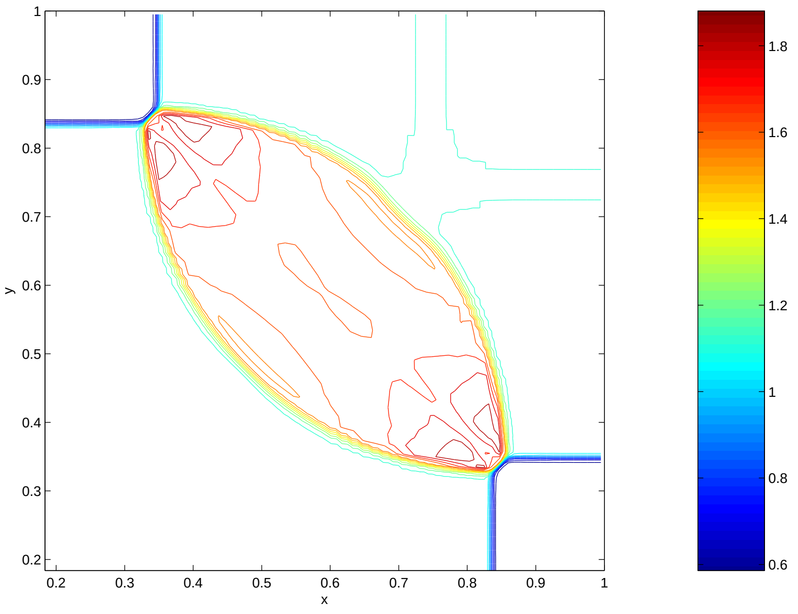

Figure 6.36: Two-dimensional Riemann problem anisotropic calculation density contours at t=0.2.

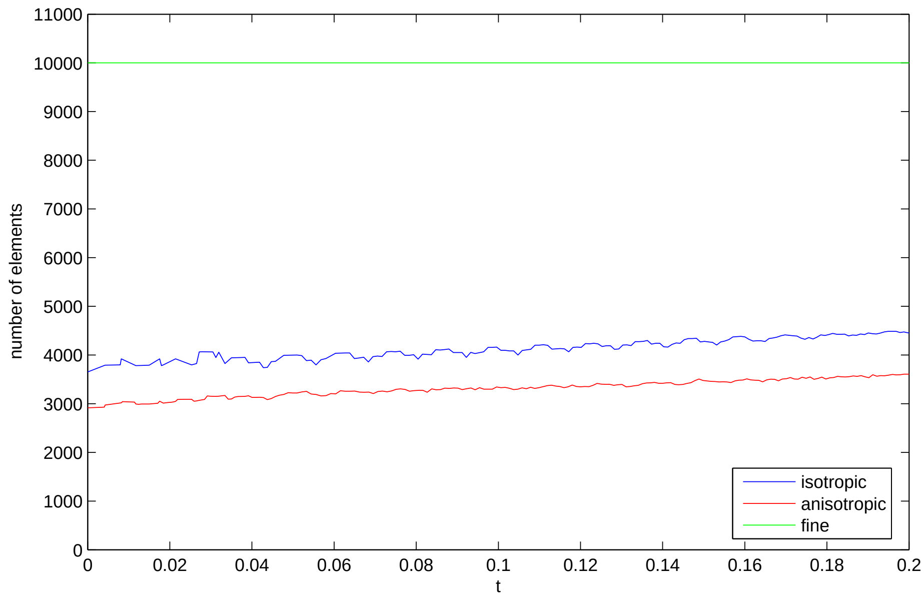

Figure 6.37: Two-dimensional Riemann problem comparing the number of elements required over calculation time.

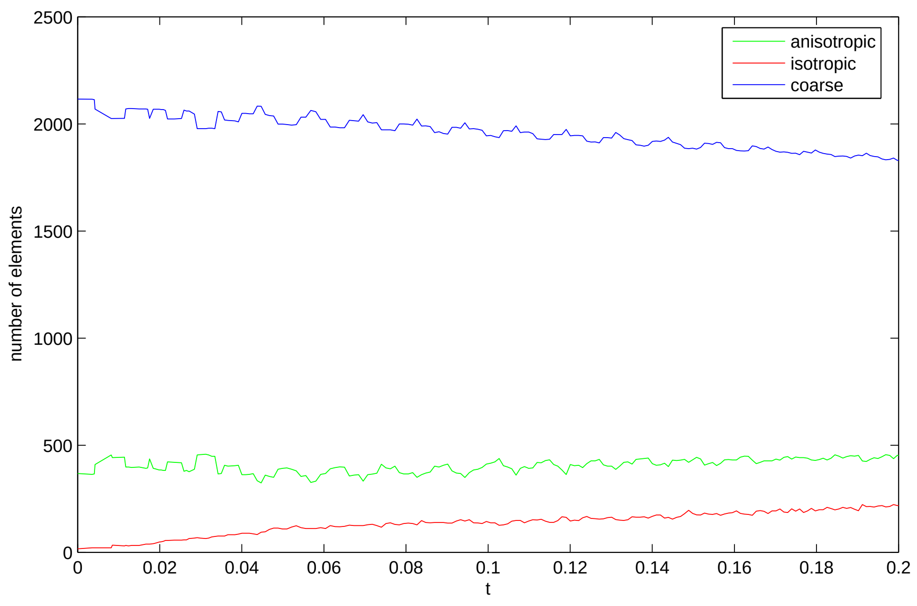

Figure 6.38: Number of each type of element over time for the two-dimensional Riemann problem.

6.7.4 Mach 3.0 flow over a forward step

The anisotropic adaptive method was tested on the Woodward and Colella Mach 3.0 step problem [89]. Woodward and Colella used this problem to compare and evaluate many numerical methods including PPM [30] and a staggered mesh Lagrangian plus remap scheme (BBC) [89], more recently it has been used to test an anisotropic code in Keats and Lien [44]. The initial conditions are a uniform flow with horizontal velocity 3.0, density 1.4, pressure 1.0 and gamma 1.4 in a wind tunnel with a forward facing step as illustrated in Figure 6.39. Reflecting boundary conditions are applied on the top and bottom walls of the wind tunnel and also along the edges of the step. An inflow condition with velocity 3.0 is applied at the left boundary of the wind tunnel and an outflow boundary condition is applied at the right boundary. At t=4.0 the flow features are: a rarefaction fan around the singular point at the corner of the step, a Mach stem above the step, a contact discontinuity emerging from the three-shock intersection, and reflected shocks. To obtain the correct positions of the Mach stem and shocks is a significant challenge for a numerical method.

The singular point at the centre of the rarefaction fan can lead to a numerical boundary layer along the top of the step. This is treated using the approach of Woodward and Colella [89], resetting the density and magnitude of the velocity in 6 elements to obtain the same entropy and enthalpy plus kinetic energy per unit mass as the zone to the left of the singular point. It should be noted that because of the difficulty of applying this approach on a staggered mesh the boundary layer and its influences have been reduced but not removed completely.

The problem was run in an Eulerian manner using the Lagrangian plus full remap approach. The refinement was performed on an initial mesh with a resolution of $\frac{1}{40}$. The adaptive meshes at t=0.5, t=1.0, t=2.0, and t=4.0 are shown in Figures 6.40, 6.41, 6.42, and 6.43 respectively. At t=0.5 the refinement follows the curvature of the shock as expected, with anisotropic refinement where the shock aligns with the mesh and isotropic refinement where it crosses diagonally. At t=4.0 anisotropic refinement occurs around the shock in front of the step, the Mach stem and wherever the shocks align with the mesh. At later times there is a significant amount of isotropic refinement and less anisotropic refinement because many of the shocks diagonally cross the mesh, and the mesh can not align with the flow because it is mapped back to a fixed mesh each time step.

The anisotropic density contours are exceedingly similar to those obtained from the uniformly fine calculation and therefore only the anisotropic results are presented in Figure 6.44. The Mach stem is situated above the step and its length is comparable to that shown in Woodward and Colella [89], the shock positions are also agreeable with those in Woodward and Colella. The contact discontinuity emerging from the three-shock intersection has been captured as has the weak reflection from the top of the step. Although there is a boundary layer occurring along the top of the step, the singular point treatment has reduced it and stopped spurious Mach reflections from occurring.

Figure 6.45 shows that the anisotropic calculation reduces the number of elements per time step, the difference between the isotropic calculation and anisotropic calculation is not as striking as in the other test problems because much of the domain contains shocks that cross the elements diagonally. Figure 6.46 shows the type of refinement occurring during the anisotropic calculation, at times earlier than 0.5 the refinement is predominantly anisotropic, at later times the majority of the refinement is isotropic as expected. The anisotropic calculation reduces the total number of elements required to

45% of the fine calculation's number of elements, while the isotropic calculation required 51%. The anisotropic calculation runs 4 times faster than the uniformly fine calculation, while the isotropic calculation only runs 3.5 times faster. These speed ups are obtained even though a large proportion of the domain contains diagonal shocks and hence is isotropically refined.

This test problem has shown that even for an extremely challenging test problem with many sensitive features of interest and a mesh that does not align with the flow the anisotropic adaptive method produces results that are comparable to the uniformly fine calculation, while reducing the number of elements required and the calculation run time.

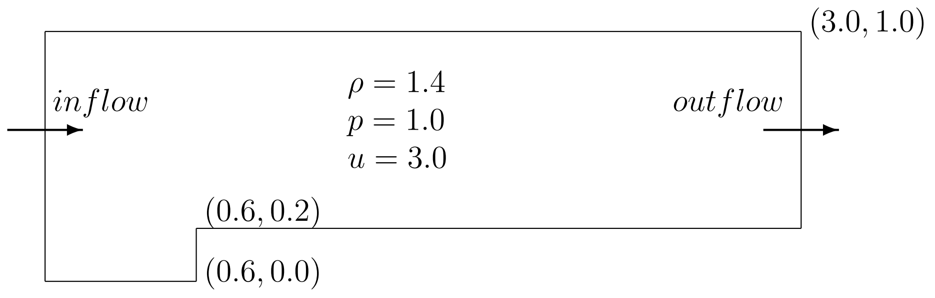

Figure 6.39: Diagram showing initial conditions for the Mach 3.0 step problem.

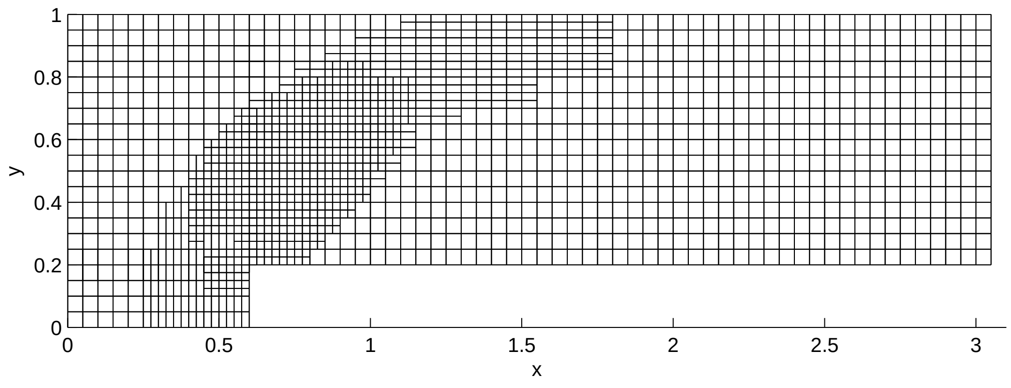

Figure 6.40: Mach 3.0 step problem anisotropic mesh at t=0.5.

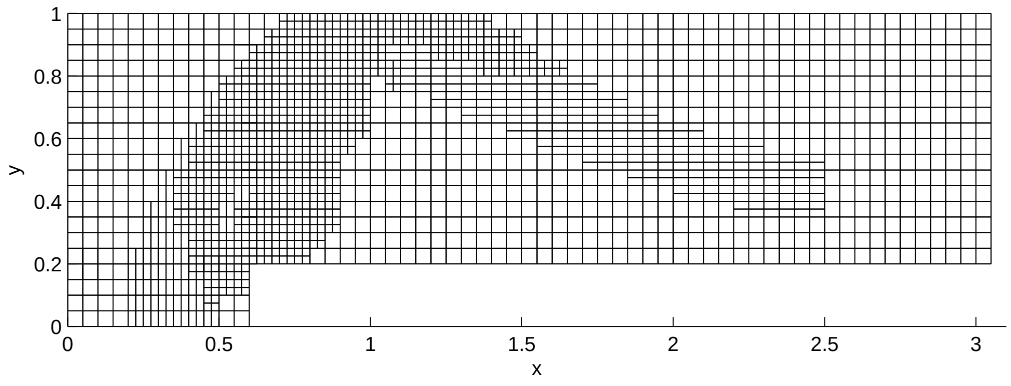

Figure 6.41: Mach 3.0 step problem anisotropic mesh at t=1.0.

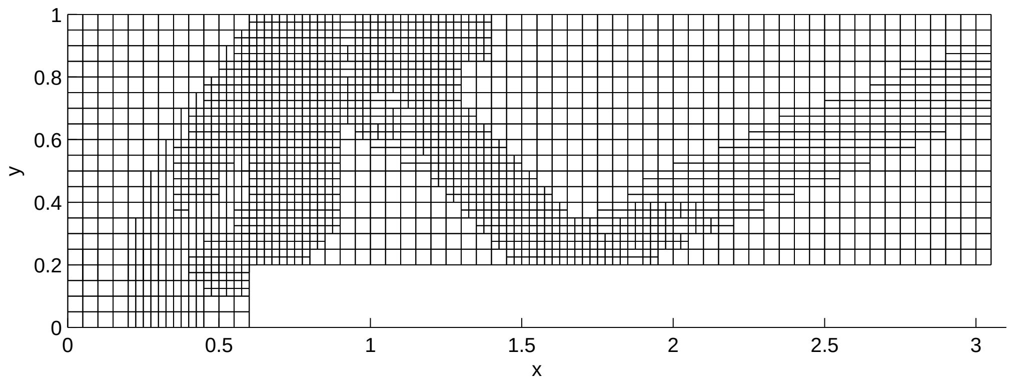

Figure 6.42: Mach 3.0 step problem anisotropic mesh at t=2.0.

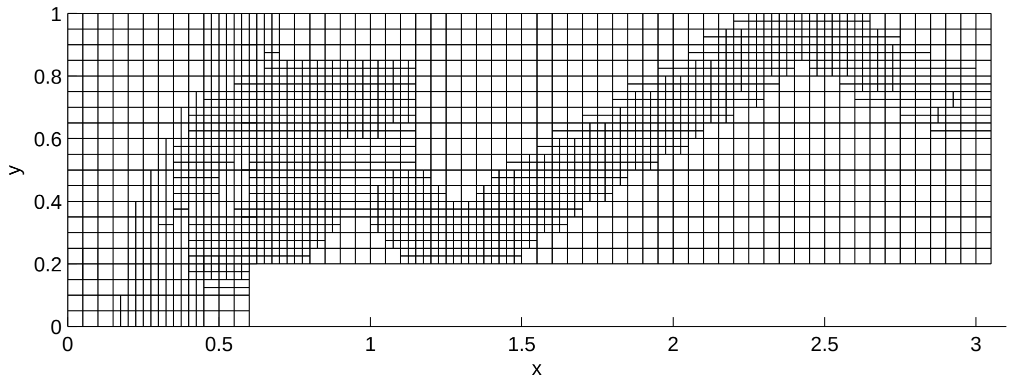

Figure 6.43: Mach 3.0 step problem anisotropic mesh at t=4.0.

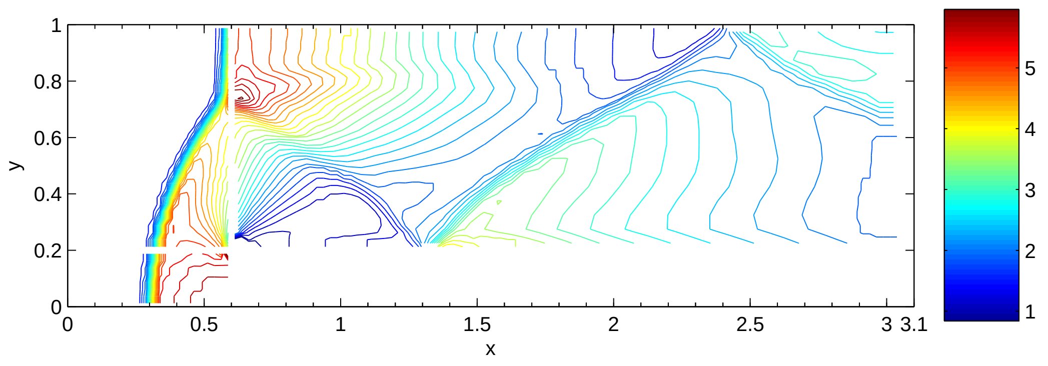

Figure 6.44: Mach 3.0 step problem anisotropic calculation density contours at t=4.0.

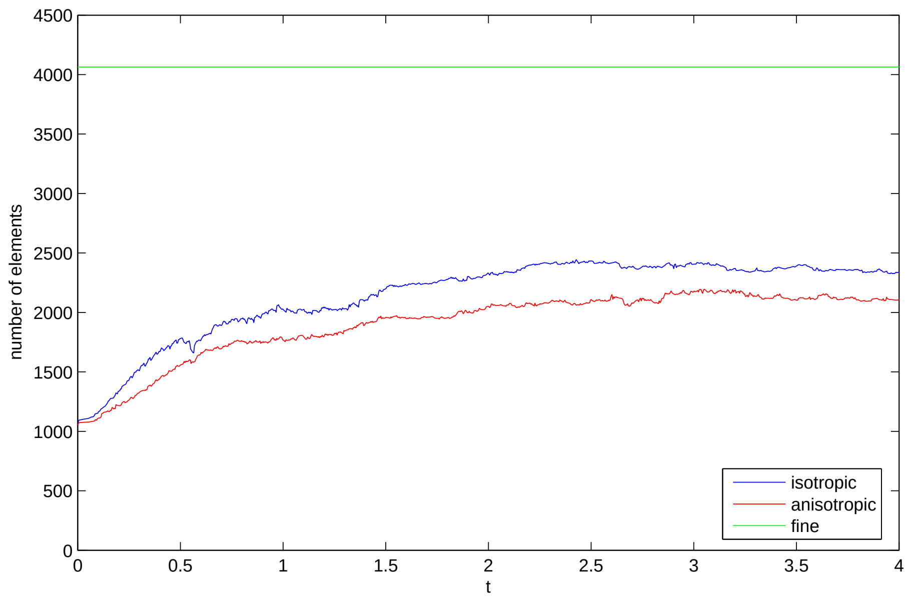

Figure 6.45: Mach 3.0 step problem comparing the number of elements required over calculation time.

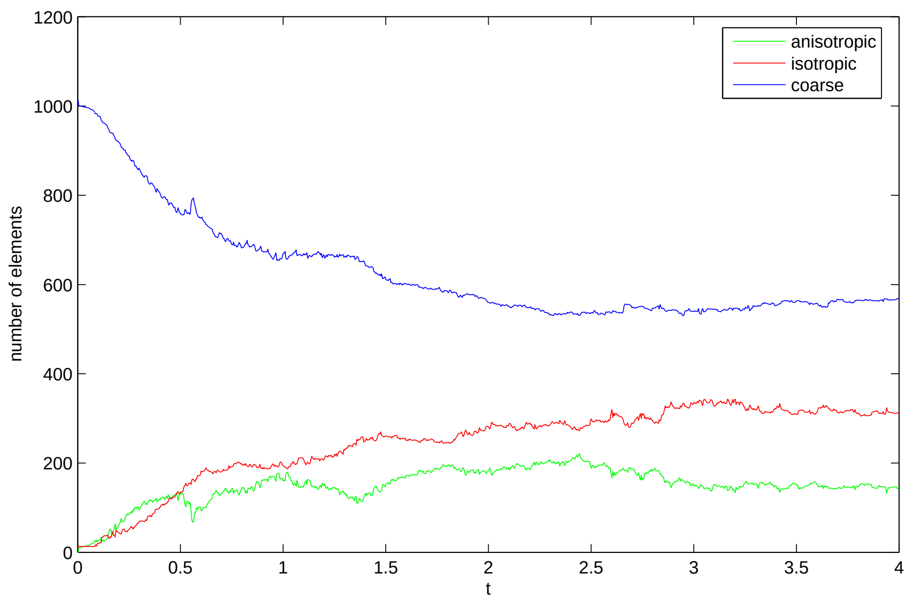

Figure 6.46: Number of each type of element over time for the Mach 3.0 step problem.

6.7.5 Summary of results

The anisotropic adaptive mesh ALE method has been validated on all of the test problems originally considered in Chapter 2. The anisotropic adaptive results have been shown to be comparable to the uniformly fine results, but the anisotropic method only requires a ninth of the calculation time. The anisotropic method has also been successfully tested on the Mach 3.0 step problem, a challenging test problem containing many features of interest.

In conclusion, the method achieves a considerable speed up with only one level of refinement and the anisotropic method has been shown to be a worth while addition to the mesh refinement technique. In the following chapter we summarise the methods developed and the results achieved, and suggest future areas for further work.

Chapter 7

Conclusions and further work

The primary achievement of this work has been the development and validation of an anisotropic cell by cell adaptive mesh Arbitrary Lagrangian Eulerian gas code for the solution of the Euler equations. An efficient approach to equipotential mesh relaxation on anisotropically refined meshes has also been developed in the process.

When the Euler equations are solved on a particular mesh the solution accuracy will depend on the mesh resolution. Focusing resolution in areas of the mesh where the solution varies rapidly or over small length scales will increase the accuracy of the solution. There are two main approaches to this optimisation problem: r-refinement where the nodes move providing higher resolution in specific areas of the domain, but the number of nodes and elements remain fixed, or h-refinement where the number of elements is increased by subdivision of elements.

In the Lagrangian plus remap ALE method the mesh follows the movement of the material during the Lagrangian step, increasing the resolution around features of interest; the mesh quality is then improved by using an equipotential mesh relaxation method to reduce any mesh tangling resulting from the Lagrangian step; finally the solution is remapped to the new relaxed mesh. This approach automatically behaves as an r-refinement method focusing the nodes in the areas of interest. However, the number of elements remains fixed and this can lead to under resolution in other areas of the domain. The method developed here combines the advantages of ALE whilst varying the number of elements through cell by cell anisotropic refinement.

Many features of interest, such as shocks, involve large variations in one dominant direction, anisotropic subdivision of elements can achieve the required resolution around

these features without wasting refinement in the other directions. As the ALE mesh attempts to align with the flow directions, anisotropic refinement is particularly efficient and beneficial. Furthermore, the refined elements will also align with the flow directions.
The starting point for this work was the staggered grid finite element ALE method developed by Barlow [13], which is outlined in Chapter 2. The Lagrangian form of the Euler equations was solved using quadrilateral isoparametric bilinear finite elements and a predictor-corrector time advancement scheme. A two-dimensional generalisation of Christensen's artificial viscosity [13] was used for shocks. Winslow's equipotential mesh relaxation method [86], [51] was applied to obtain new relaxed positions for the nodes. The solution remapping or advection procedure was then described for a structured mesh. Results were presented for the Sod problem, two-dimensional Riemann problem, square Sod problem, radial Sod problem and a Taylor-Sedov blast wave; providing a set of results that the adaptive mesh results could be compared to in later chapters.
The new isotropic adaptive mesh Lagrangian method was developed in Chapter 3. The cell by cell refinement technique subdivides the quadrilateral elements in both of the element's local directions. This is beneficial as the refinement aligns with the Lagrangian mesh, which aligns with the flow directions, and is novel since cell by cell refinement is normally applied on Eulerian rather than Lagrangian or ALE meshes. The new elements were inserted into the Dynamic Mesh, which consisted of the finest resolution in each part of the domain, with disjoint or hanging nodes at resolution transitions. An efficient solution procedure was developed that solved only on the Dynamic Mesh, rather than solving separately on every refinement level as in Anderson's structured AMR work [6]. Promising results were obtained for the Sod problem and the square Sod problem in less time and with accuracy comparable to the uniformly fine calculation.
In Chapter 4 the equipotential relaxation and advection (remapping) methods were generalised for the unstructured Dynamic Mesh, producing a cell by cell isotropic adaptive mesh ALE method. This enabled the solution to be obtained efficiently on the combined Dynamic ALE Mesh, rather than solving on every level of a refinement hierarchy. The normal Winslow-Crowley equipotential relaxation method applied on the Dynamic Mesh would cause the finer meshing to spread into the coarse regions. To prevent this, nodal length spacings were taken into account by giving each node a length value that denoted coarse or fine spacing from the centre node. Taylor Series expansions involving

these length values were then used to discretise the equation, and the resulting relaxation equations involved weights that depended on the length values. This method and simply reverting to a coarse stencil at resolution transitions were tested and the nodal length weighted stencil was found to behave in the manner most similar to the relaxation of the uniformly fine mesh, ensuring that the nodes did not move too far and avoiding over relaxation that could remove all the Lagrangian nature of the mesh.

The advection method was generalised for the unstructured Dynamic Mesh in Chapter 4 by: interpolating or conservatively averaging to obtain any values required for the slope calculation, and taking all neighbouring fluxes into account. The momentum advection was generalised by carefully defining the dual mesh and considering all neighbouring nodes, which introduced the possibility of fluxes that passed between diagonal nodes at resolution corners.

The adaptive ALE method was tested on the radial Sod problem, Taylor-Sedov problem and the two-dimensional Riemann problem. Excellent results were achieved verifying that comparable accuracy could be achieved in between a fifth and a seventh of the computational time of the uniformly fine calculations.

A second order solution transfer method for the adaptive technique was developed in Chapter 5, which used the conservative interpolation approach of Anderson [6] for cell centred variables. For nodal variables, this method was extended and developed so that it could be applied to even refinement ratios. Although second order refinement did increase the accuracy of the adaptive technique a little, there was not a substantial improvement. The use of buffering elements results in refining elements before substantial changes in the variables occur, particularly in test problems where shocks separate constant states this often leads to almost negligible slopes and hence first order solution transfer is almost as successful as second order. Furthermore, due to monotonicity considerations, refinement occurring where variables change very rapidly may interpolate with smaller limited slopes. It is feasible that if second order solution transfer were applied on many levels of refinement a more significant improvement would be seen and this would be an interesting future investigation.

In Chapter 6 the anisotropic adaptive mesh ALE method was developed as anisotropic refinement can be particularly beneficial with Lagrangian or ALE meshes. The quadrilateral elements were subdivided in one of their local directions for anisotropic refinement, or in both directions for isotropic refinement, according to where the element's change in density occurred on the refinement quadrant map. The Lagrangian method and the advection method generalised to include anisotropic refinement fairly easily, more work was required to extend the nodal length weighted equipotential relaxation method to the Dynamic Mesh containing anisotropic elements. Two nodal length values were required for each diagonal node (one for each direction of spacing from the centre node) and the relaxation weights for each coordinate were calculated using the nodal length values appropriate for the spacing in that direction.

Excellent results were achieved with the anisotropic method on all the test problems, much of the refinement was anisotropic with isotropic refinement occurring only where shocks crossed or where the feature of interest diagonally crossed an element. The anisotropic method was as accurate as the isotropic method in the radial Sod problem and comparable to the uniformly fine results. The density contour plots looked exceedingly similar to the uniformly fine and isotropic results for the square Sod problem, Taylor-Sedov problem and two-dimensional Riemann problem.

The anisotropic method was also tested on the Mach 3 step problem to verify the performance of the method on an exceedingly challenging problem containing many sensitive features of interest. The anisotropic results were comparable to the uniformly fine calculation results. In this problem a large amount of the domain contains features of interest that cross the elements diagonally, despite this the anisotropic method still reduced the number of elements required and the calculation run time.

For all of the test problems considered the anisotropic method significantly reduced the number of elements required and the calculation time. For the Mach 3 step problem the calculation time reduced to a fourth of the uniformly fine calculation time. In the other test problems the total number of elements decreased to as little as 29% of that required for the uniformly fine calculation, and the calculation time reduced to between a seventh and a ninth of the uniformly fine calculation time. This verifies that including an anisotropic refinement capability has been an exceedingly successful and worth while area of work.

In conclusion both the isotropic and anisotropic refinement methods allow us to focus resolution locally around features of interest, reducing the number of elements required in the calculation and the run time significantly. The accuracy of the calculations has been shown to be comparable to that of a uniformly fine calculation. The anisotropic refinement method reduces the number of elements required by a significant amount, since much of the refinement is anisotropic. The anisotropic method reduces the run time substantially compared to both the isotropic method and the original uniformly fine calculations. In the next section we consider remaining areas for further work.

7.1 Further work

The main area of further work is considered to be generalising to many refinement levels, rather than one refinement level. The isotropic method will generalise to many refinement levels as long as the levels are properly nested so that only a 2:1 refinement ratio occurs at each resolution transition (one disjoint node between two dynamic nodes). Buffering must be implemented from the finest level down to achieve proper nesting. Proper nesting will ensure that the isotropic length weighted mesh relaxation and cell centred advection can still be applied as usual on the Dynamic Mesh. The dual mesh can be defined as before and this will enable the momentum fluxes and their neighbours to be calculated. The problems remaining are coding issues such as data structures, and these can be solved using derived data types and implementing a hierarchy of connectivity arrays, which should be efficient as neighbours are recorded. As the solutions are obtained on the Dynamic Mesh the data structures will not influence the actual method. The anisotropic refinement will be harder to implement with many refinement levels. The easiest approach would be to apply anisotropic refinement only on the finest level and this is the approach of Aftosmis [2]. In this case the buffering, anisotropic equipotential relaxation and advection methods can be applied without additional work. If anisotropic refinement is to be utilised on all levels the situation is more complicated. Proper nesting is still required, but the 2:1 refinement ratio restriction may be required across and along a refinement interface. It is unclear whether, inside an anisotropic region, only further similarly directed anisotropic refinement should be allowed, or whether isotropic refinement should be included as well. Anisotropic buffering will need further consideration. The anisotropic equipotential relaxation could be applied with higher nodal length values and the advection should generalise simply. It would be interesting to test how the method performs with high aspect ratio anisotropic elements.

Including multiple level refinement would also allow an investigation into the reduction in elements and calculation speed up that the anisotropic method provides for a set number of refinement levels and the best number of refinement levels could hence be determined.

While investigating multiple refinement levels the second order transfer could also be assessed to see if a more noticeable improvement is obtained. It is expected that this would be the case as more refinement will be occurring in areas where the variables change steadily and the variable slope should hence be more important. Second order transfer should also help to ensure that second order accuracy is maintained at each level up to the finest. If the second order refinement was highly beneficial it could then be generalised for the anisotropic refinement.

Other interesting areas for future work include: reducing the oscillations that may occur if a shock crosses a resolution transition; investigating other refinement criteria such as second differences and combined variable sensors; optimisation studies into the ideal refinement and derefinement tolerances; and a more detailed investigation into slope limiting on unstructured meshes.

Bibliography

[1] F. Addessio, J. Baumgardner, J. Dukowicz *et al.* "CAVEAT: A computer code for fluid dynamics problems with large distortion and internal split", Los Alamos technical report UC-905, 1992.

[2] M. Aftosmis. "Upwind method for simulation of viscous flow on adaptively refined meshes", AIAA Journal, Vol. 32, 2, 268-277, 1994.

[3] A. Amsden, H. Ruppel and C. Hirt. "SALE: A simplified ALE computer program for fluid flow at all speeds", Los Alamos technical report 8095, 1980.

[4] C. Anderson. "An overview of the theory of hydrocodes", International Journal of Impact Engineering, Vol. 5, 33-59, 1987.

[5] R. W. Anderson, R. B. Pember and N. S. Elliott. "A dynamically adaptive Arbitrary Lagrangian-Eulerian method for hydrodynamics", Adaptive Mesh Refinement - Theory and Applications, Springer-Verlag, 73-81, 2005.

[6] R. W. Anderson, N. S. Elliott *et al.* "An Arbitrary Lagrangian-Eulerian method with adaptive mesh refinement for the solution of the Euler equations", Journal of Computational Physics, 199, 598-617, 2004.

[7] R. W. Anderson, N. S. Elliott and R. B. Pember. "A dynamically adaptive Arbitrary Lagrangian-Eulerian method for solution of the Euler equations", Lawrence Livermore National Laboratory, Preprint UCRL-JC-151904, 2003.

[8] R. W. Anderson, N. S. Elliott and R. B. Pember. "A dynamically adaptive Arbitrary Lagrangian-Eulerian method for Hydrodynamics", Lawrence Livermore National Laboratory, Preprint UCRL-JC-150660, 2002.

[9] R. W. Anderson, R. B. Pember and N. S. Elliott. "An Arbitrary Lagrangian-Eulerian method with local structured adaptive mesh refinement for modeling shock hydrodynamics", 40th AIAA Aerospace sciences meeting and exhibit, 2002.

[10] R. B. Pember and R. W. Anderson. "A comparison of staggered-mesh Lagrange plus remap and cell-centred direct Eulerian Godunov schemes for Eulerian shock hydrodynamics", Lawrence Livermore National Laboratory, Preprint UCRL-JC-139820, 2000.

[11] T. Apel, S. Grosman, P. Jimack and A. Meyer. "A new methodology for anisotropic mesh refinement based upon error gradients", Applied Numerical Mathematics, Vol. 50, 329-341, 2004.

[12] T. Apel, M. Berzins, P. Jimack *et al.* "Mesh shape and anisotropic elements: theory and practice", The mathematics of finite elements and applications X, Ed. J. R. Whiteman, 367-376, 2000.

[13] A. Barlow. An adaptive multi-material Arbitrary Lagrange Eulerian algorithm for computational shock hydrodynamics, PhD Thesis, Swansea, 2002.

[14] S. Baden and N. Christochoides. Structured Adaptive Mesh Refinement (SAMR) Grid Methods, Springer-Verlag, 2000.

[15] M. Baines. "Grid adaptation via node movement", Applied Numerical Mathematics, 26, 77-96, 1998.

[16] M. Baines, M. Hubbard and P. Jimack. "A moving finite element method using monitor functions", School of Computing Leeds, Research report series 2003.04, 1-22, 2003.

[17] G. I. Barenblatt. Scaling, Self-Similarity and intermediate asymptotics, Cambridge texts for applied maths, 1996.

[18] D. Benson. "Computational methods in Lagrangian and Eulerian Hydrocodes", Computer Methods in Applied Mechanics and Engineering, 99, 235, 1992.

[19] D. Benson. “An efficient, accurate simple ALE method for nonlinear finite element programs”, Computer Methods in Applied Mechanics and Engineering, 72, 305-350, 1989.

[20] D. Benson. “Momentum advection on a staggered mesh”, Journal of Computational Physics, 100, 143-162, 1992.

[21] J. Bell, M. J. Berger *et al.* “ Three-dimensional adaptive mesh refinement for hyperbolic conservation laws”, SIAM Journal on Scientific Computing, Vol. 15, 1, 127-138, 1994.

[22] M. J. Berger and P. Colella. “Local adaptive mesh refinement for shock hydrodynamics”, Journal of Computational Physics, 82, 64-84, 1989.

[23] M. J. Berger. “Data structures for adaptive grid generation”, SIAM Journal scientific and statistical computing, Vol. 7, 3, 904-916, 1986.

[24] M. J. Berger and A. Jameson. “Automatic adaptive grid refinement for the Euler equations”, AIAA Journal, Vol. 23, 4, 561-568, 1985.

[25] M. J. Berger and J. Oliger. “Adaptive mesh refinement for hyperbolic partial differential equations”, Journal of Computational Physics, 53, 484-512, 1984.

[26] M. J. Berger. Adaptive mesh refinement for hyperbolic partial differential equations, PhD Thesis, Stanford University, 1982.

[27] J. Brackbill and J. Saltzman. “Adaptive zoning for singular problems in two dimensions”’, Journal of Computational Physics, 46, 342-368, 1982.

[28] E. Caramana, M. Shashkov *et al.* “Formulations for artificial viscosity for multidimensional shock wave computations”, Journal of Computational Physics, 144, 70-97, 1998.

[29] R. Christensen. Godunov methods on a staggered mesh - an improved artificial viscosity, private communication cited in [13], 1991.

[30] P. Colella and P. Woodward. “The piecewise parabolic method for gas-dynamical simulations”, Journal of Computational Physics, 54, 174-201, 1984.

[31] P. Colella. "Multidimensional upwind methods for hyperbolic conservation laws", Journal of Computational Physics, 87, 171-200, 1990.

[32] J. Dannenhoffer and J. Baron. "Grid adaptation for the 2D Euler equations", AIAA Paper 85-0484, 1-11, 1985.

[33] L. Dursi and M. Zingale. "Efficiency gains from time refinement on AMR meshes and explicit time stepping", Adaptive Mesh Refinement - Theory and Applications, Springer-Verlag, 103-113, 2005.

[34] J. Dukowicz. "Conservative rezoning (Remapping) for general quadrilateral meshes", Journal of Computational Physics, 54, 411-424, 1984.

[35] J. Dukowicz and J. Baumgardner. "Incremental remapping as a transport/advection algorithm ", Journal of Computational Physics, 160, 318-335, 2000.

[36] J. Dukowicz and J. Kodis. "Accurate conservative remapping (rezoning) for arbitrary Lagrangian-Eulerian computing", Journal of Scientific and Statistical Computing, Vol. 8, 3, 305-321, 1987.

[37] P. Eiseman and G. Erlebacher. "Grid generation for the solution of partial differential equations", NASA Langley Research Center, ICASE report 87-57, 1-85, 1987.

[38] A. Giannakopoulos and A. Engel. "Directional control in grid generation", Journal of Computational Physics, 74, 422-439, 1988.

[39] C. Hirt, A. Amsden *et al.* "An Arbitrary Lagrange-Eulerian computing method for all flow speeds", Journal of Computational Physics, Vol. 14, 3, 227-253, 1974.

[40] W. Huang, Y. Ren, and R. Russell. "Moving mesh partial differential equations (MMPDES) based on the equidistribution principle", SIAM Journal Numerical Analysis, Vol. 31, 3, 709-730, 1994.

[41] W. Huang. "Variational mesh adaptation: isotropy and equidistribution", Journal of Computational Physics, 174, 903-924, 2001.

[42] W. Huang and R. Russell. "Adaptive mesh movement - the MMPDE approach and its applications", Journal of Computational and Applied Mathematics, 128, 383-398, 2001.

[43] Y. Kallinderis and J. Baron. "Adaptation methods for a new Navier-Stokes algorithm", AIAA Journal, Vol. 27, 1, 37-43, 1989.

[44] W. Keats and F. Lien. "Two-dimensional anisotropic Cartesian mesh adaptation for the compressible Euler equations", International Journal for Numerical Methods in Fluids, 46, 1099-1125, 2004.

[45] A. Khokhlov. "Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics", Journal of Computational Physics, 143, 519-543, 1998.

[46] G. Lapenta. "Grid adaptation and remapping for Arbitrary Lagrangian Eulerian (ALE) methods", Numerical grid generation in computational simulations conference 8, 849-858, 2002.

[47] G. Lapenta. "Variational grid adaptation based on the minimization of local truncation error: time-independent problems", Journal of Computational Physics, 193, 159-179, 2004.

[48] W. Liu, H. Chang *et al.* "Arbitrary Lagrangian-Eulerian Petrov-Galerkin finite elements for nonlinear continua", Computer Methods in Applied Mechanics and Engineering, 68, 259-310, 1988.

[49] H. Luo and J. D. Baum. "On the computation of multi-material flows using ALE formulation", Journal of Computational Physics, 194, 304-328, 2004.

[50] M.J. Marchant and N.P. Weatherill. "Adaptive techniques for compressible inviscid flows", Computer Methods in Applied Mechanics and Engineering, 106, 83-106, 1993.

[51] J. Peery and D. Carroll. "Multi-Material ALE methods in unstructured grids", Computer Methods in Applied Mechanics and Engineering, 187, 591-619, 2000.

[52] R. B. Pember, J. B. Bell *et al.* "An adaptive cartesian grid method for unsteady compressible flow in irregular regions", Journal of Computational Physics, 120, 278-304, 1995.

[53] M. Piggott, C. Pain *et al.* "h, r, and hr adaptivity with applications in numerical ocean modelling", Ocean Modelling, 10, 95-113, 2005.

[54] T. Plewa, T. Linde and V. G. Weirs. Adaptive Mesh Refinement - Theory and Applications, Springer-Verlag, 2005.

[55] S. Popinet. "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries", Journal of Computational Physics, 190, 572-600, 2003.

[56] C. Powell. Private communication.

[57] E. Puckett and J. Saltzman. "A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics ", Physica D, 60, 84-93, 1992.

[58] J. J. Quirk. An adaptive algorithm for computational shock hydrodynamics, PhD Thesis, Cranfield, 1991.

[59] J. Ramshaw. "Conservative rezoning algorithm for generalized two dimensional meshes", Journal of Computational Physics, 59, 193-199, 1985.

[60] J. Ramshaw. "Simplified second-order rezoning algorithm for generalized two dimensional meshes", Journal of Computational Physics, 67, 214-222, 1986.

[61] G. Sod. "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws", Journal of Computational Physics, 27, 1-31, 1978.

[62] M. Souli, A. Ouahsine and L. Lewin. "ALE formulation for fluid-structure interaction problems", Computer Methods in Applied Mechanics and Engineering, 190, 659-675, 2000.

[63] G. Strang. "On the construction and comparison of difference schemes", SIAM Journal of Numerical Analysis, Vol. 5, No. 3, 1968.

[64] A. Kurganov and E. Tadmor. “Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers”, Numerical Methods Partial Differential Equations, Vol. 18, 5, 584-604, 2002.

[65] Z. Tan and P. Varghese. “Directionally adaptive finite element method for multidimensional Euler and Navier-Stokes equations”, AIAA Paper 93-3320, 248-259, 1993.

[66] E. F. Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag, 1999.

[67] J. Thompson. Handbook of grid generation, CRC press, page 33, 1999.

[68] J. Thompson, F. Thames and C. Wayne Mastin. “TOMCAT - A code for numerical generation of boundary-fitted curvilinear coordinate systems on fields containing any number of arbitrary two-dimensional bodies”, Journal of Computational Physics, 24, 274-302, 1977.

[69] P. Thomas and J. Middlecoff. “Direct control of the grid point distribution in meshes generated by elliptic equations”, AIAA Journal, Vol. 18, 6, 652-656, 1980.

[70] R. Tipton. Grid optimization by equipotential relaxation, Lawrence Livermore National Laboratory, 1992.

[71] J. van der Vegt and H. van der Ven. “Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows”, Journal of Computational Physics, 141, 46-77 , 1998.

[72] B. Van Leer. “Towards the ultimate conservative difference scheme. I. The quest for monotonicity”, Lecture Notes in Physics, 18, 163-168, 1973.

[73] B. Van Leer. “Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second order scheme”, Journal of Computational Physics, 14, 263-275, 1974.

[74] B. Van Leer. “Towards the ultimate conservative difference scheme. III. Upstream-centred finite difference scheme for ideal compressible flow”, Journal of Computational Physics, 23, 263-275, 1977.

[75] B. Van Leer. "Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection", Journal of Computational Physics, 23, 276-299, 1977.

[76] B. Van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method", Journal of Computational Physics, 32, 101-136, 1979.

[77] P. Vijayan and Y. Kallinderis. "A 3D Finite-Volume scheme for the Euler equations on adaptive tetrahedral grids", Journal of Computational Physics, 113, 249-267, 1994.

[78] D. Vollmer. Adaptive mesh refinement using subdivision of Unstructured elements for conservation laws, Msc Thesis, The University of Reading, 2003.

[79] J. Von Neumann and R. D. Richtmyer. "A method for the numerical calculation of hydrodynamic shocks", Journal of Applied Physics, 21, pages 232-237, 1950.

[80] M. Walkley, P. K. Jimack and M. Berzins. "Mesh quality and anisotropic adaptivity for finite element solutions of 3-D convection-dominated problems", ECCOMAS computational fluid dynamics conference, Swansea, 2001.

[81] M. Walkley, P. K. Jimack and M. Berzins. "Anisotropic adaptivity for finite element solutions of 3-D convection-dominated problems", Numerical methods for fluid dynamics VII, ICFD conference, Ed. M. Baines, 525-531, 2001.

[82] M. Walkley, P. K. Jimack and M. Berzins. "Mesh quality for three-dimensional finite element solutions on anisotropic meshes", In proceedings of FEM3D, Gakuto international series, mathematical sciences and applications, Vol. 15, 310-321, 2001.

[83] M. Walter, A. Abdu *et al.* "Evaluation of adaptive mesh refinement and coarsening for the computation of compressible flows on unstructured meshes", International Journal for Numerical Methods in Fluids, 49, 999-1014, 2005.

[84] B. Wells. A moving mesh finite element method for the numerical solution of partial differential equations and systems, PhD Thesis, The University of Reading, October 2004.

[85] M. Wilkins. “Use of artificial viscosity in multidimensional fluid dynamic calculations”, Journal of Computational Physics, 36, 281-303, 1980.

[86] A. M. Winslow. “Numerical solution of the quasilinear poisson equation in a non-uniform triangular mesh”, Journal of Computational Physics, 1, 149-172, 1966.

[87] G. Whitham. Linear and Non-linear waves, Wiley, 1974.

[88] W. Wood and W. Kleb. “On multi-dimensional unstructured mesh adaption”, AIAA Paper 99-3254, 1-15, 1999.

[89] P. Woodward and P. Colella. “The numerical simulation of two-dimensional fluid flow with strong shocks”, Journal of Computational Physics, 54, 115-173, 1984.

[90] S. Zalesak. “Fully multidimensional flux-corrected transport algorithms for fluids”, Journal of Computational Physics, 31, 335-362, 1979.