THE UNIVERSITY OF READING

DEPARTMENT OF MATHEMATICS

A cell by cell adaptive mesh Lagrangian scheme for the numerical solution of the Euler Equations.

J. Morrell

Numerical Analysis Report 1/05

May 2005

Abstract

A cell by cell adaptive mesh technique is described and implemented to solve the Euler equations. The adaptive mesh scheme builds upon a staggered grid Lagrangian code similar to the AWE code CORVUS. The mesh is automatically refined in a cell by cell manner using a solution gradient refinement criteria. The method has the automatic response of adaptive mesh refinement but without storing and solving each level separately. Disjoint nodes are used in the transition from fine to coarse elements. Time refinement is not included at present. The adaptive mesh technique produces results comparable to the uniformly fine mesh in a fraction of the computational time.

Acknowledgements

I would like to acknowledge the help of my supervisors Dr. Sweby of Reading University 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 Reading University for his advice. This research was funded by AWE.

Contents

List of Figures

Chapter 1

Introduction

Hyperbolic differential equations, such as the Euler equations, exhibit shocks and shock formation. Computational fluid dynamics has been used for decades to model the propagation and formation of shocks. There is much interest in increasing the resolution of such features of interest. Lagrangian schemes are often used to allow the mesh to follow the movement of the material and therefore cluster elements in the areas of interest. However this technique is limited by the initial number of elements as no new elements can be created. As the mesh becomes more dense around the feature of interest surrounding areas of the mesh can become underresolved. Furthermore, the Lagrangian mesh may become tangled because of solution vorticity.

To reduce mesh tangling Lagrangian schemes are often combined with a remap step where the grid is relaxed and the state variables are remapped or advected. The grid may be completely mapped back to a fixed Eulerian grid [13]. A group of arbitrary Lagrange-Eulerian or ALE methods [15], [2] use Lagrangian plus remap schemes to map to a more optimal grid, in-between the possible Lagrangian and Eulerian grids. ALE methods provide another method of moving the grid to reflect the solution. However once again the total number of elements is fixed. Finer resolution in one area will reduce the resolution in the other areas so that not all features of interest can be resolved, therefore ALE is described as a penalty based adaption method.

An alternative approach to the problem of resolution is adaptive mesh refinement, AMR. This method allows the total number of elements in a problem to be increased by introducing local fine regions, thereby avoiding global refinement that would prove too expensive. These techniques have traditionally been applied on Eulerian meshes, where

the mesh remains fixed throughout time and the material moves relative to the mesh. A hierarchical set of grids representing different refinement levels are automatically created as further resolution is required [6], [10], [19]. This technique has also been applied to elliptic equations often in the form of the very successful multigrid method, which uses quadtree data structures. However in time dependent problems the Eulerian formulation makes it more expensive to track or introduce physics on material interfaces. Clearly a Lagrangian adaptive mesh technique would provide a solution to these problems.

An ALE adaptive mesh refinement method has been successfully developed by Anderson, Elliott and Pember [1]. In their AMR approach refinement occurs in rectangular blocks, which simplifies the data structures needed. However a greater number of fine elements are required and the rectangular blocks may increase the chances of the mesh imprinting on the solution. Furthermore solutions must be obtained for all levels even though many will not be used. Therefore in this work a cell by cell refinement strategy is presented and the levels are not considered separately, rather the whole grid containing coarse and fine elements is used.

Instead of creating separate levels, new elements are inserted creating a mesh with both fine and coarse elements. Adaptive mesh insertion or AMI was developed for element insertion in CORVUS [2]. The refinement is limited as elements can only be inserted in one direction, in contrast to the isotropic technique developed in this work. The AMI developed in CORVUS is triggered when element aspect ratios become large, rather than using a refinement trigger based on the solution. In this work the solution gradient is used to sense when refinement is required in a similar way to the AMR methods.

The staggered grid Lagrangian method used here is based on the Lagrangian step in the code CORVUS, which was developed at AWE [2]. CORVUS uses quadrilateral bilinear finite elements to calculate spatial derivatives. A predictor-corrector scheme is used for time advancement. Two methods of artificial viscosity are employed in this code; bulk artificial viscosity and a two dimensional formulation of Christensen's artificial viscosity. Excellent reviews on artificial viscosity can be found in Wilkins [26] and Caramana, Shashkov et al [11].

In the next chapter the Lagrangian step is outlined in detail. The predictor-corrector time discretisation and spatial finite element schemes are considered. Bulk artificial viscosity and a two dimensional form of Christensen's artificial viscosity are outlined

and discussed. Finally results for a Sod's shock tube problem, radial Sod problem and two dimensional Riemann problem are presented by way of validating the Lagrangian code.

In Chapter 3 we consider the introduction of disjoint nodes, nodes with three neighbours rather than four, on an interface between different mesh densities. The alteration of the Lagrangian step to include disjoint nodes is detailed. A piston problem is run with a fine patch to highlight the oscillations that can be caused by a shock crossing an interface between two different mesh densities. This motivates the need for an adaptive technique for refining the mesh.

The adaptive mesh technique is described fully in Chapter 4. The notation, data structures and refinement criteria are all introduced. The features of interest must remain totally encapsulated within the fine regions throughout the Lagrangian step. Buffer cells are introduced to ensure this. The preliminary first order solution transfer method for refinement and derefinement is considered. Results are presented for Sod's shock tube and a two dimensional Riemann problem. These verify that the adaptive mesh technique, with one refinement level, produces results comparable to the uniformly fine mesh in less computational time. Although the existing scheme only contains one refinement level it should generalise fairly simply to include an arbitrary number of refinement levels.

Finally some conclusions are drawn in Chapter 5 and suggestions for further work are made. These include; an arbitrary Lagrange Eulerian capability, second order solution transfer for refinement or derefinement and the extension of the existing scheme to a general number of refinement levels.

Chapter 2

The Lagrangian Scheme

The staggered grid Lagrangian method that forms the basis for this work is based on the Lagrangian step in the code CORVUS, which was developed at AWE [2]. In a Lagrangian method the mesh moves with the material. This is in contrast to a Eulerian mesh, which remains fixed as the material moves through it. It is easier to track material properties and interfaces using a Lagrangian mesh. However, a Lagrangian mesh may distort severely as the material deforms.

CORVUS uses an unstructured grid formulation. This means that the structure of the grid is addressed indirectly, rather than being implied by the node numbering. The grid connectivity is still fixed but it is defined using connectivity matrices that record the neighbouring nodes or elements. Grid points no longer need to have the same number of neighbours. An unstructured grid is much more flexible for handling complex geometries. Furthermore it is more suitable when mesh movement or element insertion is required. However unstructured grids do increase program complexity, run time and storage.

During the Lagrangian step quadrilateral bilinear isoparametric finite elements are used to calculate spatial derivatives. Finite elements are used rather than finite volume or finite difference methods partly because the method is uniquely defined once the element shape and finite element functions have been chosen. Finite elements are often used with unstructured grids. Many finite difference algorithms cannot be used for unstructured grids and require mesh spacing to stay the same. The isoparametric formulation is designed to deal with non-orthogonal meshing. Finite elements are very flexible for use with complex geometries and can vary in shape and size.

Due to the implicit pressure dependence in the Euler equations time advancement is

performed using a predictor-corrector scheme. This avoids the iteration that would be required for an implicit time scheme.

2.1 The Euler equations

The Euler equations in a Lagrangian reference frame are

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{u} \tag{2.1}$$

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p \tag{2.2}$$

$$\rho\frac{D\epsilon}{Dt} = -p\,\nabla\cdot\mathbf{u}, \tag{2.3}$$

where

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

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,

$$p = (\gamma - 1)\rho\,\epsilon. \tag{2.5}$$

Throughout this report $\gamma = 1.4$ is used for air, other values of γ could be used to run problems with different fluids.

The conservation of mass equation (2.2) can be derived from

$$\frac{D}{Dt}\int \rho dV = 0, \tag{2.6}$$

where V is any control volume. This tells us that in the Lagrangian formulation any element will retain the same mass throughout the calculation.

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 is defined by the mass transfer through the boundary. The perpendicular velocity is kept constant throughout the calculation in all the problems presented here.

The Euler equations are derived for smooth inviscid compressible flow, not shock problems. The physics across shock waves is represented by the Rankine-Hugoniot shock

conditions. Shock physics can be built into the numerical method by using artificial viscosity, solving a Riemann problem or flux limiting. 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 idea of adding an artificial viscosity term q to the pressure terms in the Euler equations was suggested by von Neumann [25] in the 1950's. This spreads the shock over 3-4 elements, mimicking irreversible shock heating. The form of q will be discussed in a later section.

2.2 The Lagrangian mesh

A series of logically rectangular regions are defined initially. The regions are divided into grids of quadrilateral elements. Connectivity arrays for the element-node, element-element and node-node connections are then created so that an unstructured grid approach can be used.

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. 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 make the tracking of interfaces and the inclusion of mixed cells easier.

Each element's mass is assumed to be constant since the Lagrangian mesh moves with the material.

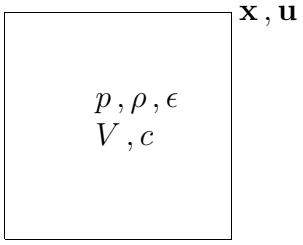

Figure 2.1: Diagram showing the positions of the nodal and element centred variables.

2.3 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 CFL denominator approximates the element shock speed and must be adjusted to take the artificial viscosity into account. 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.
- Calculate a stable time step using the CFL condition,

$$\Delta t < Cl/\sqrt{c_s^2 + 2q/\rho}. \tag{2.7}$$

- Move the nodes to their half time step positions using

$$\mathbf{x}^{n+1/2} = \mathbf{x}^n + \frac{\Delta t}{2}\,\mathbf{u}^n. \tag{2.8}$$

- 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),

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

  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,

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \frac{\Delta t}{M^{nodal}}\nabla(p^{n+1/2} + q^n), \tag{2.10}$$

  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

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\, \bar{\mathbf{u}}. \tag{2.11}$$

- Calculate the final volumes and densities.

- Evaluate the energies at the end of the time step using

$$\epsilon^{n+1} = \epsilon^n - \frac{\Delta t}{M^e}\,(p^{n+1/2} + q^n)\nabla \cdot \bar{\mathbf{u}}. \tag{2.12}$$

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

2.4 Spatial discretisation

The domain must be discretised in order to evaluate the $\nabla(p+q)$ term in the momentum equation and the $\nabla \cdot \mathbf{u}$ term in the energy equation. This can be done using finite volumes, finite elements or finite differences. This work uses bilinear isoparametric finite elements. 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

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta) \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta) \\ N_3 &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_4 &= \frac{1}{4}(1-\xi)(1+\eta). \end{aligned} \tag{2.13}$$

2.4.1 The momentum equation

The acceleration in element e can be expressed as the summation over the element's nodes of the acceleration multiplied by the finite element function at each node,

$$\dot{\mathbf{u}} = \sum_k \dot{u}_k N_k\,. \tag{2.14}$$

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

$$\rho_e \int_\Omega \sum_k \dot{u}_k N_k N_j d\,\Omega = -\int_\Omega \frac{\partial(p_e + q_e)}{\partial r_i} N_j d\,\Omega\,, \tag{2.15}$$

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'

$$\int_\Omega \sum_k \dot{u}_k N_k N_j d\,\Omega = \dot{u}_j \int_{-1}^{1}\int_{-1}^{1} N_j detJd\xi d\eta\,. \tag{2.16}$$

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

$$\rho_e \dot{u}_j \int_{-1}^{1}\int_{-1}^{1} N_j detJd\xi d\eta = (p_e + q_e)\int_{-1}^{1}\int_{-1}^{1} \frac{\partial N_j}{\partial r_i} detJd\xi d\eta\,. \tag{2.17}$$

The term on the right is the mass contribution from element e to its local node j. The term on the left is the force in the direction r_i acting on node j due to element e's pressure. In practice the elements are looped and the force and pressure that the element exerts on the node is evaluated. The four element values are then gathered for each node. A nodal acceleration can then be calculated and discretised to give the updated nodal velocity. The process must be done for each Cartesian direction.

2.4.2 The energy equation

The Euler energy equation (2.3) can be rewritten as

$$\rho_e \dot{\epsilon} = -(p+q)\,\nabla \cdot \mathbf{u}. \tag{2.18}$$

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

$$u = \sum_k u_k N_k; \qquad v = \sum_k v_k N_k. \tag{2.19}$$

On integrating (2.18), realising that p_e, q_e and ρ_e are constant within each element and substituting (2.19) into (2.18) we obtain

$$\rho_e \dot{\epsilon}\int_\Omega d\,\Omega = -(p_e+q_e)\int_\Omega \sum_k \left(u_k \frac{\partial N_k}{\partial x} + v_k \frac{\partial N_k}{\partial y}\right) d\,\Omega\,. \tag{2.20}$$

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

$$\dot{\epsilon} = -\frac{(p_e+q_e)}{M_e}\sum_{j=1}^{4}\left[u_j\left(\int_{-1}^{1}\int_{-1}^{1}\frac{\partial N_j}{\partial x} detJd\xi d\eta\right) + v_j\left(\int_{-1}^{1}\int_{-1}^{1}\frac{\partial N_j}{\partial y} detJd\xi d\eta\right)\right]. \tag{2.21}$$

For each element the nodal summation is performed first using the record of the element's nodes. Then the constant element quantities are included and the energy value updated via the time discretisation.

2.5 Axisymmetric changes

The Cartesian coordinates (x, y) can be mapped to the Axisymmetric coordinates (z, r) by including a radial dependence in the finite element and volume equations. This effectively rotates the Cartesian domain 360° around the x-axis. Cylindrical problems can then be run very efficiently with only a two dimensional number of elements.
In the element volume equation $dxdy$ becomes $rdrdz$ and we express the radius in terms of the finite element functions. The element volume then becomes

$$V_e = \sum_{j=1}^{4} (r_j \int_{-1}^{1} \int_{-1}^{1} N_j \mathrm{detJ} d\xi d\eta) \,. \tag{2.22}$$

The energy equation in Axisymmetric coordinates is

$$\rho \dot{\epsilon} = -(p+q) \left[\frac{\partial u}{\partial z} + \frac{\partial v}{\partial r} + \frac{v}{r} \right] . \tag{2.23}$$

Integrating the equation, substituting in the finite element functions, simplifying the equation and applying mass lumping in a similar way to the Cartesian case gives the following expression

$$\dot{\epsilon} = -\frac{(p_e + q_e)}{M_e} \sum_{j=1}^{4} \left[u_j r_j \left(\int \frac{\partial N_j}{\partial z} \right) + v_j r_j \left(\int \frac{\partial N_j}{\partial r} \right) + v_j \left(\int N_j \right) \right] , \tag{2.24}$$

$$\text{where} \quad \left(\int \frac{\partial N_j}{\partial z} \right) = \left(\int_{-1}^{1} \int_{-1}^{1} \frac{\partial N_j}{\partial z} \mathrm{detJ} d\xi d\eta \right) \quad \text{etc.} \,.$$

When this equation is discretised in time as part of the predictor corrector scheme the r_j's are taken from the latest nodal positions.
A reflecting boundary conditions is used along the x-axis. Care must still be taken if nodes are near the x-axis because the $\frac{1}{r}$ dependence in the energy could cause the energy to become unbounded. Therefore an option to calculate the energy using the work done, pdv, is provided. The integral of the divergence is then calculated using the change in volume over the time step divided by the time step.

2.6 Artificial Viscosity

As a first approach a scalar bulk artificial viscosity was used. This is a crude extension to two dimensions of von Neumann's originally one dimensional artificial viscosity [25], where Δu has been replaced by $l\nabla \cdot \mathbf{u}$.

$$q = c_q \rho (l\nabla \cdot \mathbf{u})^2 - c_l \rho c_s l \left|\nabla \cdot \mathbf{u}\right| , \tag{2.25}$$

where l is the element characteristic length, c_s is the element sound speed, c_q and c_l are constants whose values are problem dependent. In the program the characteristic length is approximated by the square root of the element area. The artificial viscosity uses the characteristic length times the divergence merely as a measure of the change in u across an element. Therefore we do not require the Axisymmetric form of the divergence in the artificial viscosity, the Cartesian version is sufficient.

The bulk artificial viscosity contains a linear combination of quadratic and linear viscosities. The quadratic viscosity acts over a fixed number of zones. A small amount of the linear viscosity, which can act over many zones, is added to try to smooth any oscillations. However the bulk artificial viscosity is not monotonic and ignores the directional nature of viscosity, which should only act in the direction of compression [26]. A more sophisticated approach was required but without the complications of a full tensor artificial viscosity [11].

A two dimensional generalisation, devised by Tipton [2], of Christensen's scalar monotonic artificial viscosity [12] was included in the program. Christensen's artificial viscosity uses a Taylor expansion to improve the approximation of the velocity jump by taking the velocity slopes into account, see Figure 2.2. Christensen's viscosity uses a limiter ϕ, similar to that devised by van Leer [23], to insure the second order artificial viscosity is monotonic.

In Tipton's 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 2.3. 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

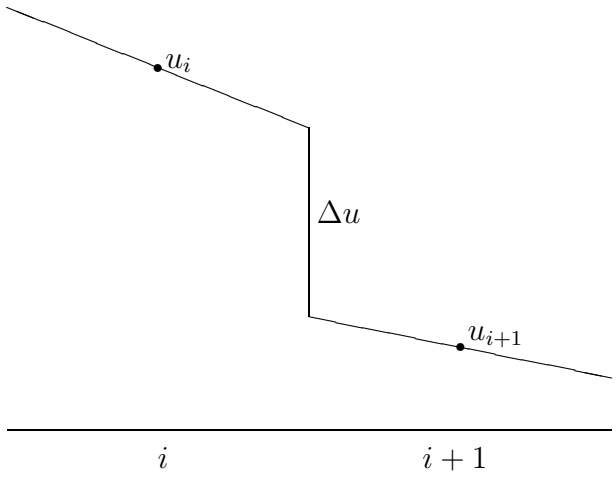

Figure 2.2: Diagram showing the Δu used in Christensen's artificial viscosity.

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 are evaluated by adding the two side vectors that are perpendicular to the required edge and then taking the normal. Therefore they have the required direction but their length is proportional to the perpendicular sides,

$$\begin{aligned} Lhor_r &= -(z_4 + z_3 - z_2 - z_1) \\ Lhor_z &= (r_4 + r_3 - r_2 - r_1) \\ Lver_r &= (z_3 + z_2 - z_4 - z_1) \\ Lver_z &= -(r_3 + r_2 - r_4 - r_1). \end{aligned} \tag{2.26}$$

The distance measures for each of the directions then become

$$\begin{aligned} \Delta x_{tb} &= \frac{area}{\left|\bar{Lhor}\right|} \\ \Delta x_{lr} &= \frac{area}{\left|\bar{Lver}\right|}, \end{aligned} \tag{2.27}$$

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

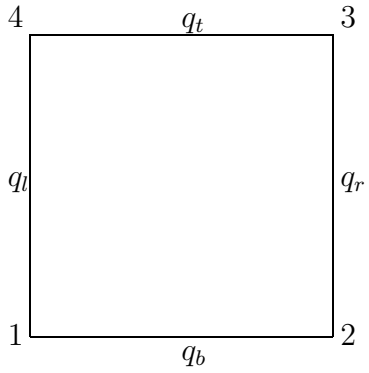

Figure 2.3: Diagram edge viscosities and node numbering. Each node has position (z,r).

onto the mesh leg,

$$\begin{aligned}
\frac{\Delta u_b}{\Delta x_b} &= \frac{L\bar{h}or \cdot (\bar{u_2} - \bar{u_1})}{area} \\
\frac{\Delta u_l}{\Delta x_l} &= \frac{L\bar{v}er \cdot (\bar{u_4} - \bar{u_1})}{area} \\
\frac{\Delta u_t}{\Delta x_t} &= \frac{L\bar{h}or \cdot (\bar{u_3} - \bar{u_4})}{area} \\
\frac{\Delta u_r}{\Delta x_r} &= \frac{L\bar{v}er \cdot (\bar{u_3} - \bar{u_2})}{area}.
\end{aligned} \tag{2.28}$$

The variables used in this process are shown in Figure 2.4. Any velocity gradient that indicates expansion is set to zero at this stage,

$$\text{If} \quad \frac{\Delta u}{\Delta x} > 0 \quad \text{then} \quad \frac{\Delta u}{\Delta x} = 0. \tag{2.29}$$

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.

For each edge the element whose value is being calculated is considered to be the central element and the neighbouring elements along the same edge are defined to be the "left" and "right" elements, see Figure 2.5. Left and right ratios of the velocity gradient are then defined as

$$\begin{aligned}
R_L &= \frac{(\frac{\Delta u}{\Delta x})_L}{(\frac{\Delta u}{\Delta x})_C} \\
R_R &= \frac{(\frac{\Delta u}{\Delta x})_R}{(\frac{\Delta u}{\Delta x})_C}.
\end{aligned} \tag{2.30}$$

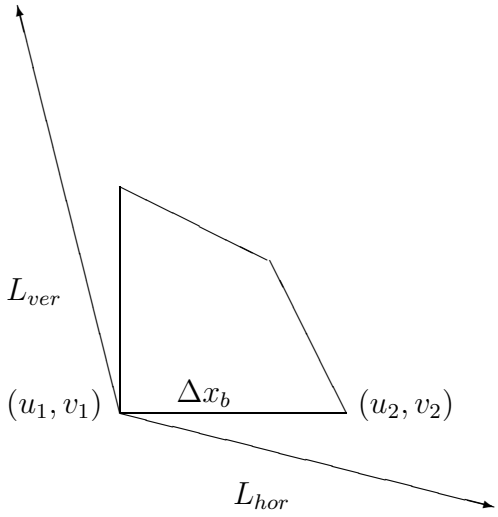

Figure 2.4: Diagram showing the variables for the calculation of the bottom edge velocity gradient.

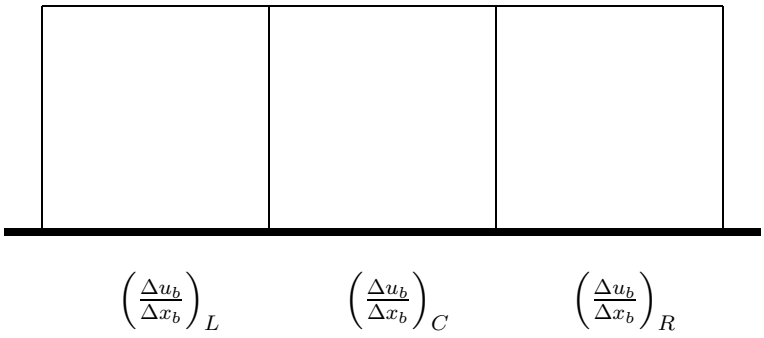

Figure 2.5: Diagram of limiting procedure for bottom edge.

The velocity slope ratio is set to one on a reflecting boundary.

The limiter ϕ is defined as

$$\phi = max(0, min(\frac{1}{2}(R_L + R_R), 2R_L, 2R_R, 1)). \tag{2.31}$$

The monotonic q for the central edge is finally given as

$$q_{edge} = c_q \rho \left|\Delta u\right|^2 (1 - \phi^2) + c_l \rho c_s \left|\Delta u\right| (1 - \phi), \tag{2.32}$$

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 $|\Delta u|$ 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

$$q_{scalar} = \frac{1}{2}(q_b + q_l + q_t + q_r). \tag{2.33}$$

If the final q indicates expansion the artificial viscosity is set to zero in that element.

2.7 Program validation

2.7.1 Sod's shock tube problem

This test problem consists of a 1 dimensional shock tube with 2 constant states divided by a diaphragm (not modelled) that is burst at t=0. If $p_l > p_r$ and $\rho_l > \rho_r$ then 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 initial conditions are $p_l = 1.0$, $\rho_l = 1.0$, $\mathbf{u}_l = 0.0$, $p_r = 0.1$, $\rho_r = 0.125$ and $\mathbf{u}_r = 0.0$.

Results for Sod's shock tube problem using bulk artificial viscosity and Christensen's artificial viscosity are shown in Figure 2.8 and Figure 2.9 respectively. Both results use 100 elements in the x direction and are in good agreement, considering the use of artificial viscosity, with the exact results from Toro's iterative Riemann solver [22].

The results using bulk artificial viscosity are very oscillatory. However the position of the contact is in excellent agreement with the exact results. The shock is spread over

4-5 elements and is positioned slightly ahead of the exact solution. This causes most of the 3.5% error in the density.

An extensive investigation in to the influence of the bulk artificial viscosity coefficients was undertaken. The number of cells that the shock is spread over was reduced by decreasing the quadratic artificial viscosity coefficient c_q. This reduced the percentage error but at the expense of introducing small oscillations around the shock. The energy overshoot was also reduced slightly. For small c_q values, such as $c_q = 0.3$, c_l must be large enough to damp oscillations behind the shock but must not increase the shock width, for example $c_l = 0.1$. In conclusion, a careful balance between c_l and c_q is needed and experimentation with several values may be required for each problem.

The bulk artificial viscosity shows a large amount of oscillation between the rarefaction and the contact discontinuity. This is because the scheme uses no limiting to guarantee monotonicity.

The results for the Christensen artificial viscosity are less oscillatory because of the limiting. The oscillations behind the contact have been completely removed. The usual values for the coefficients are $c_l = 0.5$ and $c_q = 0.75$. It should be noted that the shock is now spread over more elements and this causes a higher average error for the density of 4.7%. This problem is only one dimensional so the benefits of the edge approach to the artificial viscosity are not fully evident.

There is a large overshoot in the energy profile at the contact discontinuity that appears regardless of the type of artificial viscosity used. The overshoot is probably 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 [2].

2.7.2 Radial Sod test problem

A radial Sod problem was modelled using the Axisymmetric option. A line of 200 elements along the r-axis was initialised to represent an inner circle of radius 0.4 with $\rho_i = 1.0$ and $p_i = 1.0$ and an outer region with $\rho_o = 0.125$ and $p_o = 0.1$.

The results shown in Figure 2.10 and Figure 2.11 were compared to those in Toro [22] and found to be in good agreement. The rarefaction, contact discontinuity and shock positions seem correct. The density curve between the rarefaction fan and the contact

was successfully reproduced. Constant density states were observed in front of the contact as required. The velocity profile peaked and then curved down to the shock. This was well resolved by the program.

The bulk artificial viscosity caused oscillations behind the contact in both this problem and the one dimensional Sod's shock tube problem. The Christensen artificial viscosity retains a smooth profile in this area due to the benefits of the limiting procedure. Both artificial viscosities spread the shock over the same number of elements.

The energy profile overerestimated the contact jump and then sloped down to the correct height. This is again due to too much artificial viscosity being given to the elements around the contact at the start of the calculation until the shock is spread out.

2.7.3 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 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 $y < 0.5$. There are nineteen possible configurations of constant states divided by shocks, contact discontinuities, rarefaction fans or slip lines.

Figure 2.12 shows results for the following configuration involving 4 shocks,

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

The general form of the results shown in Figure 2.12 and Figure 2.14, for the different artificial viscosities compare well with those given in Kurganov and Tadmor [21]. Since Kurganov and Tadmor give no values for the density contours a detailed analysis is not possible. The positions of the shocks and the shapes of density contours in between them are well matched.

There are some density contours located in the constant states, these occur because too much artificial viscosity is given to the compressed elements during the first few time steps. The side they appear on depends on the velocity values of the nodes where

the quarters meet. Since Kurganov and Tadmor did not have to define velocities at the nodal values they did not experience the problem of how to define the velocities where the quarters meet. As every quarter has a different velocity there are two or more possible values for these nodal velocities. Changing these values moves where the velocity discontinuity actually occurs, this then changes whether the artificial viscosity error during the first few steps occurs behind or in front of the contact and this decides which quarter the contours will appear in.

The importance of limiting the artificial viscosity is shown very clearly in this problem. For the unlimited bulk viscosity density contours in the areas in between the shocks or rarefactions are seen to oscillate rapidly and appear noisy. This 'noise' was found to reduce as the linear artificial viscosity coefficient c_l was increased. However, this increased the density variation in the constant states and the test failed earlier due to mesh tangling. The results shown use $c_l = 0.08$ and $c_q = 1.0$. It should be noted that Kurganov and Tadmor witnessed similar 'noise' with the 4 shocks example when using an overly compressive second order scheme.

The Christensen artificial viscosity results are free from the"noise" experienced with the bulk artificial viscosity and all contours appear smooth. Christensen's artificial viscosity gives superior results because the method is monotone and provides a more two dimensional approach in the form of edge viscosities.

These calculations experience a lot of mesh distortion because the mesh moves in a Lagrangian manner. The usual Christensen artificial viscosity coefficients resulted in slightly earlier mesh tangling and calculation breakdown. This mesh tangling is not unexpected in a Lagrangian calculation and can be solved only by remapping the tangled grid, which will be implemented in a future version of the code. 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 2.14. They are much better than those obtained with the bulk viscosity but are starting to show the effect of the mesh tangling along the oval axis seen in Figure 2.13.

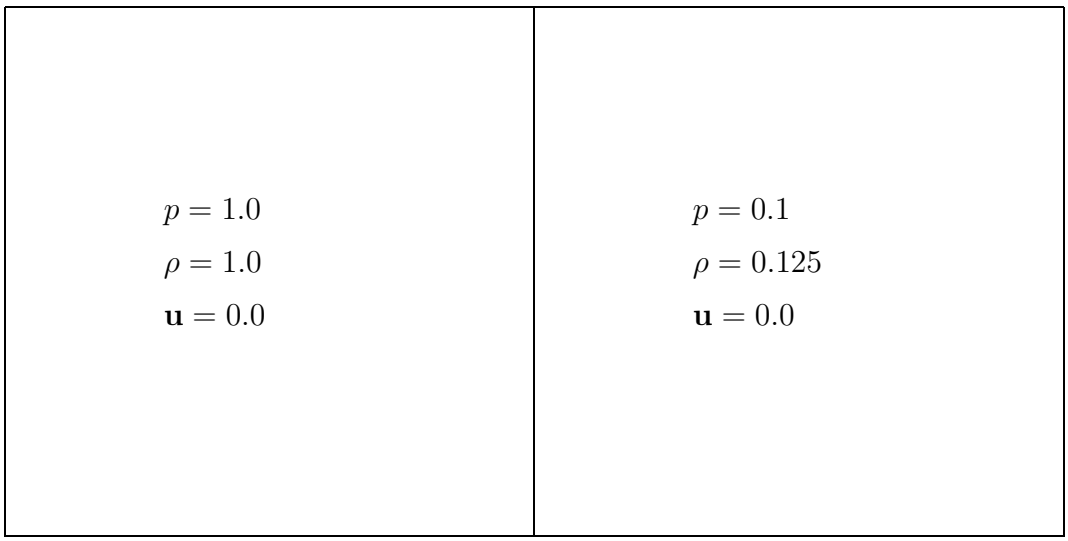

Figure 2.6: Diagram of Sod's shock tube.

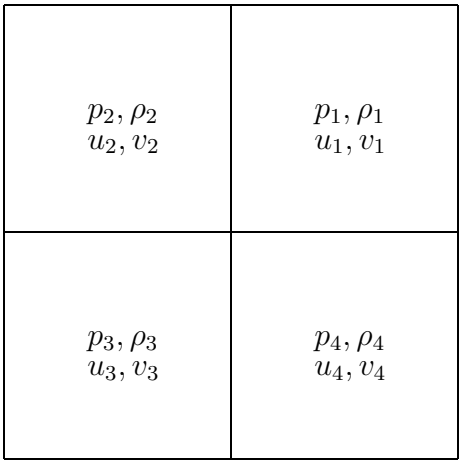

Figure 2.7: Diagram of two dimensional Riemann problem.

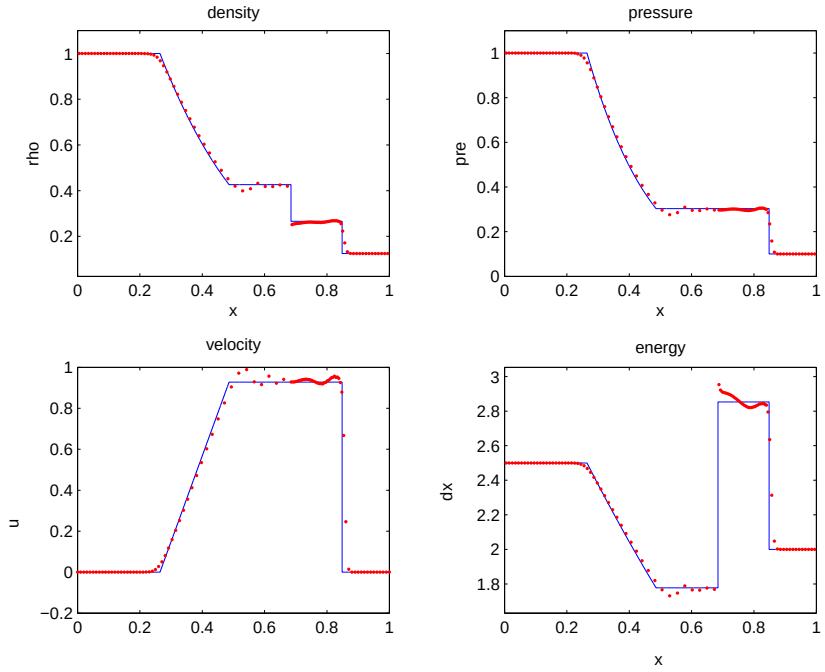

Figure 2.8: Sod's shock tube problem at t=0.2 with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$.

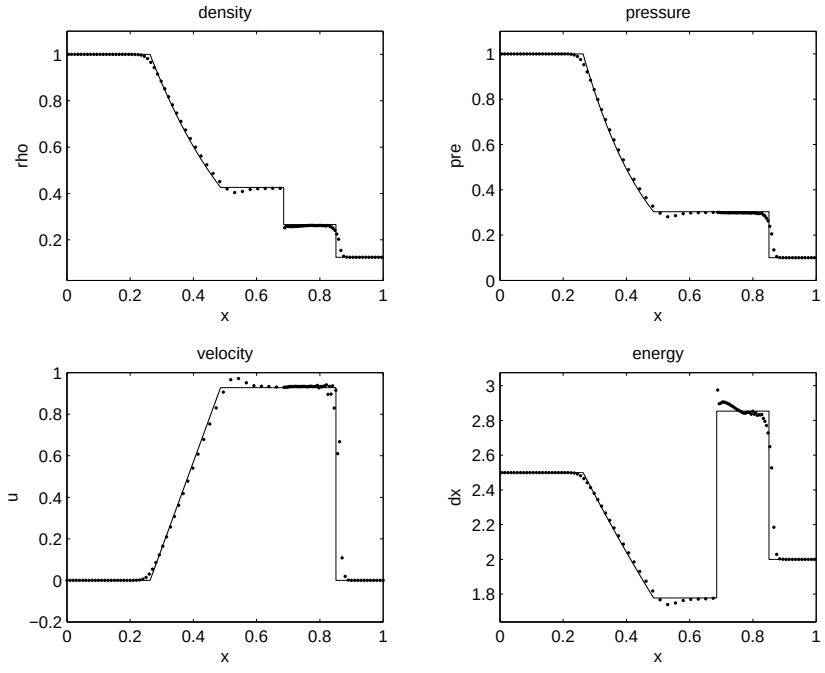

Figure 2.9: Sod's shock tube problem at t=0.2 with Christensen artificial viscosity coefficients $c_l = 0.5$ and $c_q = 0.75$.

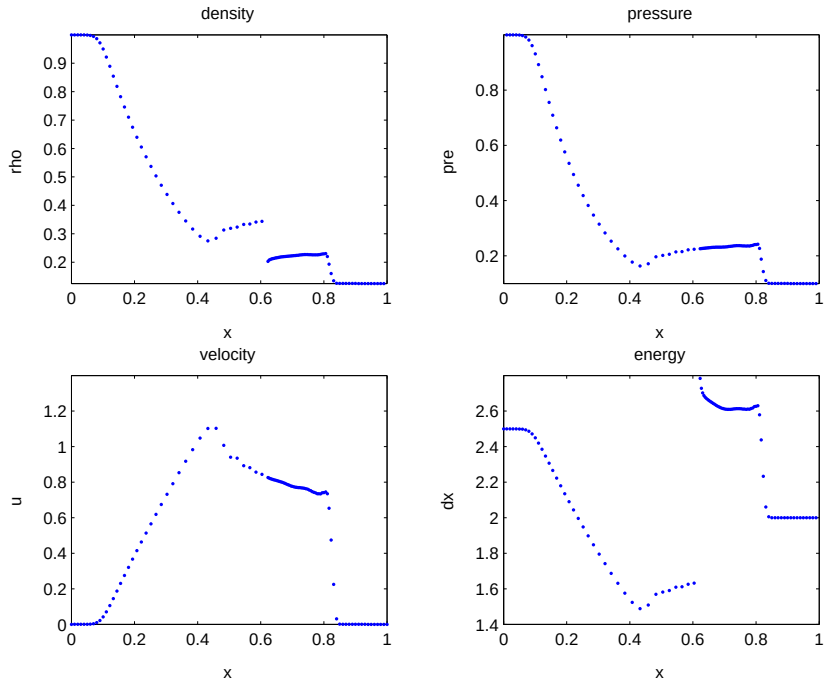

Figure 2.10: Radial Sod problem at t=0.25 with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 2.0$, 200 mesh.

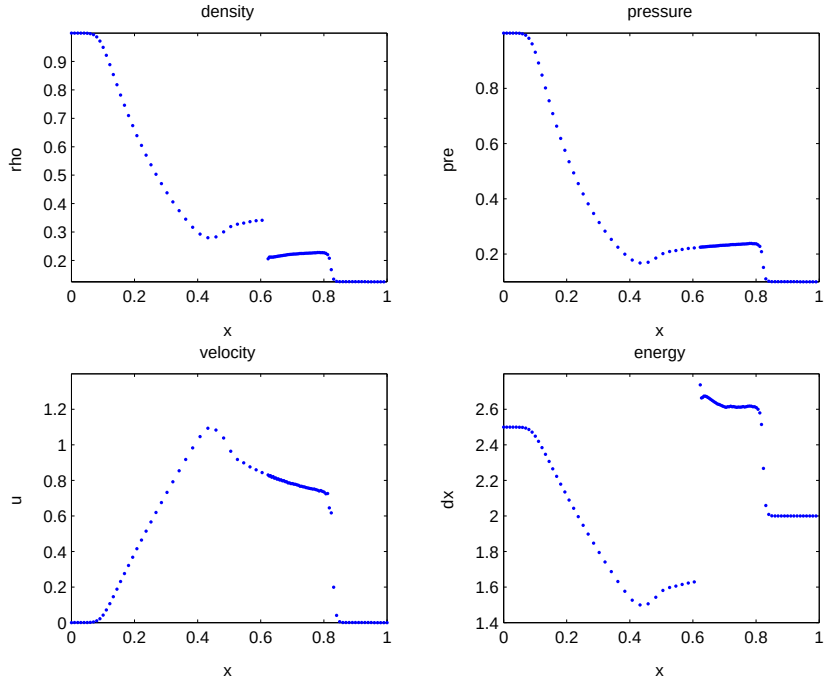

Figure 2.11: Radial Sod problem at t=0.25 with Christensen artificial viscosity coefficients $c_l = 0.5$ and $c_q = 0.75$, 200 mesh.

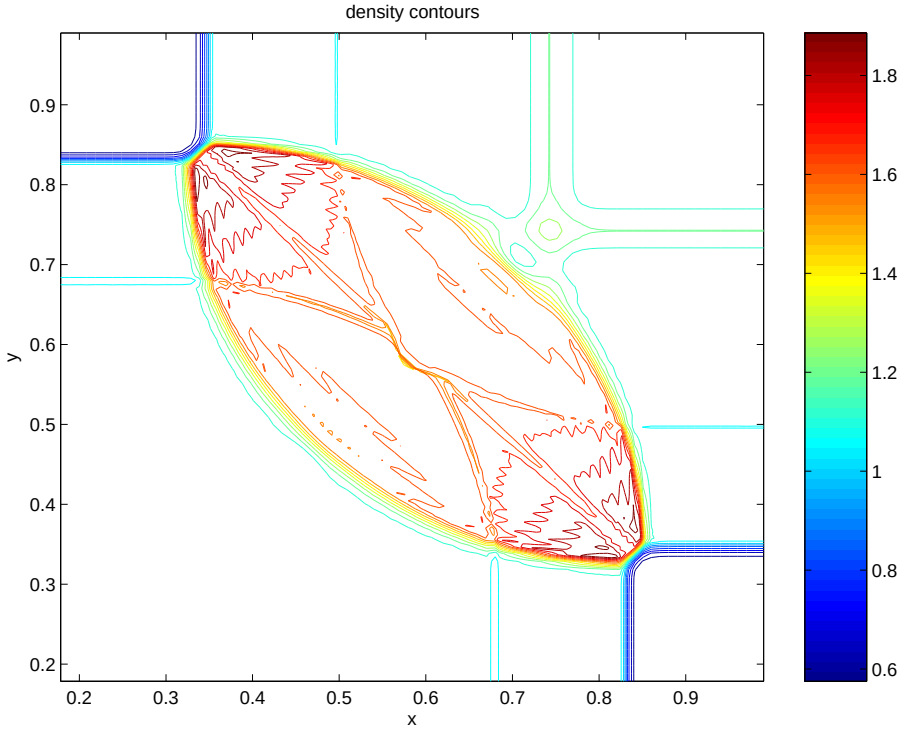

Figure 2.12: Two dimensional Riemann problem 4 shocks, at t=0.2 with bulk artificial viscosity coefficients $c_l = 0.08$ and $c_q = 1.0$.

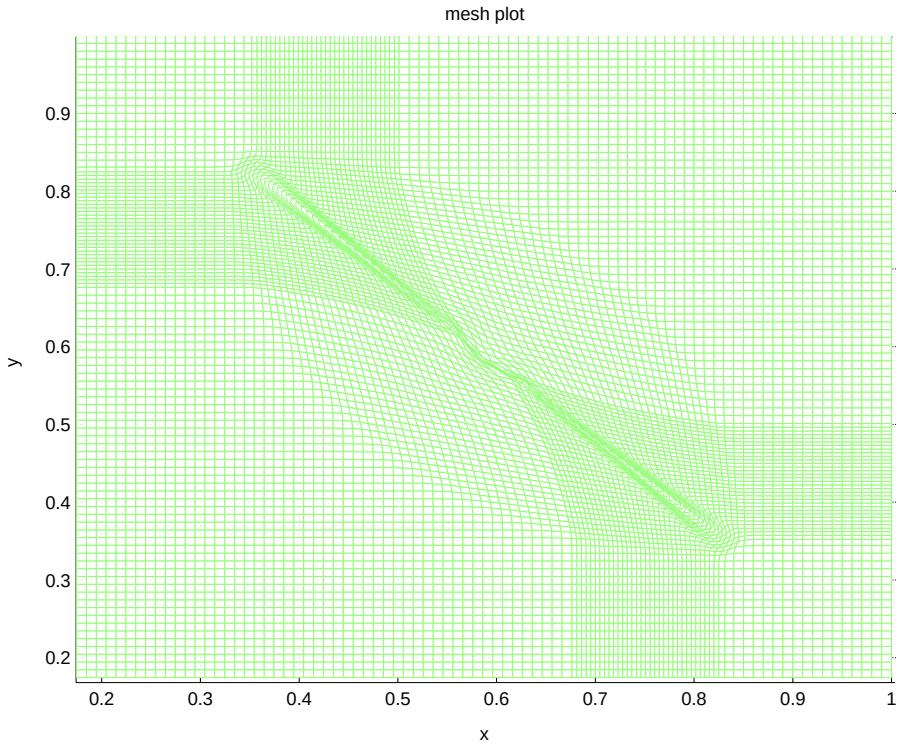

Figure 2.13: Two dimensional Riemann problem mesh 4 shocks, at t=0.2 with Christensen artificial viscosity coefficients $c_l = 0.3$ and $c_q = 0.65$.

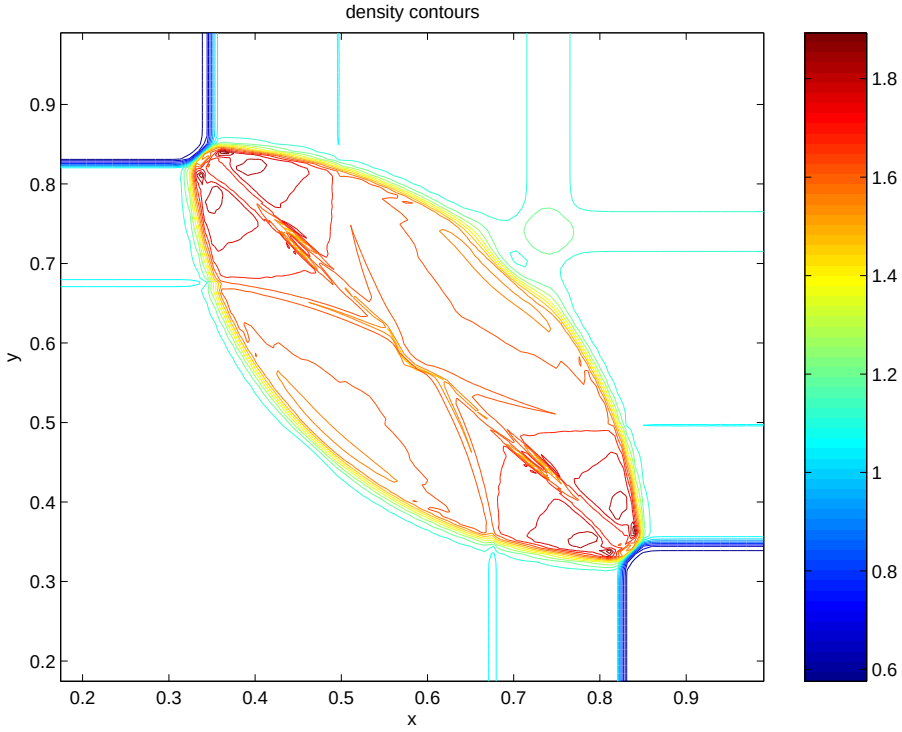

Figure 2.14: Two dimensional Riemann problem density contours 4 shocks, at t=0.2 with Christensen artificial viscosity coefficients $c_l = 0.3$ and $c_q = 0.65$.

Chapter 3

Multiple regions and disjoint nodes

3.1 Regions with different mesh densities and disjoint nodes

Before considering adaptive mesh refinement the implications of having neighbouring fine and coarse regions were considered. This was done with an unchanging mesh connectivity, in contrast to the AMR connectivity which evolves through time. On an interface between two regions with different mesh densities, in the ratio 1:2 or 1:3, there are disjoint nodes. These disjoint nodes have 3 rather than 4 nodal neighbours. The influence of disjoint nodes on the Lagrangian step is considered. Finally results are presented that highlight the oscillations that can be generated when a shock crosses an interface between areas of different mesh density.

To be able to run these problems the program was altered to include multiple regions. The connectivity arrays were generalised to contain entries from all regions. During the evaluation of nodal masses and forces the contributions from each region were taken into account. Boundary conditions were then applied only on the external boundaries. Sod's shock tube problem was rerun with separate regions and the results were found to be the same to within rounding errors.

Disjoint nodes were introduced on each interface (not an external boundary) where the change in mesh density required nodes with only three neighbours. Each disjoint node lies between two non-disjoint nodes on the interface. The distance ratio from the neighbouring non-disjoint nodes and the global node numbers of the non-disjoint and disjoint nodes are recorded.

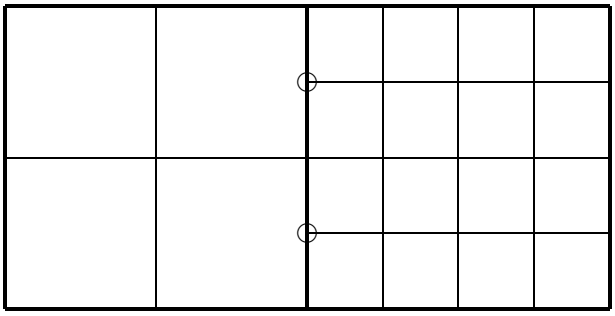

Figure 3.1: Diagram of a course-to-fine interface. Empty circles indicate disjoint nodes.

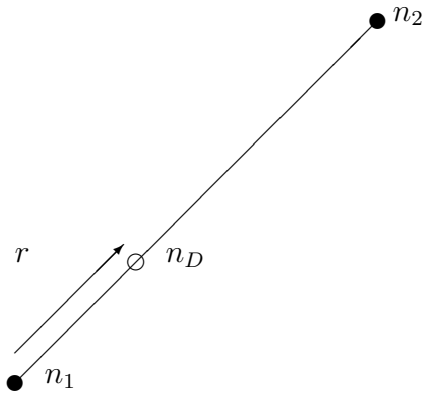

Figure 3.2: Diagram showing the distance ratio. Filled circles represent non-disjoint nodes and the empty circle represents the disjoint node.

During the Lagrangian step, 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 the above diagram the disjoint node's position becomes

$$\mathbf{x}(n_D) = r\mathbf{x}(n_2) + (1-r)\mathbf{x}(n_1). \tag{3.1}$$

This alters the disjoint node's Lagrangian motion. To account for this the nodal mass of the disjoint node is split, in the required ratio, between the two neighbouring non-disjoint nodes

$$\begin{aligned} M_n(n_1) &= M_n(n_1) + (1-r)M_n(n_D) \\ M_n(n_2) &= M_n(n_2) + rM_n(n_D). \end{aligned} \tag{3.2}$$

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

$$\begin{aligned} f_x(n_1) &= f_x(n_1) + (1-r)M_n(n_D) \\ f_x(n_2) &= f_x(n_2) + rM_n(n_D). \end{aligned} \tag{3.3}$$

The velocity of the disjoint node, although only used for the energy and artificial viscosity, must be made consistent

$$\mathbf{u}(n_D) = r\mathbf{u}(n_2) + (1-r)\mathbf{u}(n_1). \tag{3.4}$$

3.2 Piston tests with a fine patch

A piston problem was run with $\rho = 1.0$, $p = 1.0$ and a left boundary velocity of 0.1. The first experiment consisted of six regions, three bottom regions and three top regions. The middle bottom region had double the mesh density in both the x and y directions. Results were plotted for variables along constant y values for the bottom and top regions. As the shock passed through the coarse-to-fine grid interface very small oscillations were produced that moved to the left. When the shock passed out of the fine patch crossing the fine-to-coarse interface larger oscillations were produced that also moved to the left. The spurious oscillations occurred only in plots of the bottom regions. Further oscillations located directly behind the shock were observed in both the top and

bottom plots. These were eliminated if only the bottom regions in the calculation were considered.

To assess the effect of the fine-to-coarse and coarse-to-fine interfaces separately a series of two region test problems were run at a higher boundary velocity of 1.0 and a mesh density ratio of 1:2, see Figure 3.3 and Figure 3.4. No oscillations were seen before the shock crossed the interface. After crossing the coarse-to-fine interface a spurious dip and peak appeared. The dip moves to the left, moving against the Lagrangian flow. The peak remained roughly at the interface, moving to the right with the Lagrangian flow but with the distance between it and the shock increasing. For the fine-to-coarse interface a peak is observed where the dip was and a dip is observed where the peak was in the coarse-to-fine calculation. The spurious reflections had a larger amplitude in the fine-to-coarse calculation. The shock width was thinner in the fine regions. Further calculations were run with a mesh density ratio of 1:3, see Figure 3.5 and Figure 3.6. The oscillations were found to have larger amplitudes than for the mesh density ratio of 1:2. The largest amplitudes were again observed for the fine-to-coarse interface. The difference between the coarse and fine shock widths was also more pronounced.

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. It would be useful to investigate how to reduce or eliminate these oscillations since a complicated test problem may have many small reflected shocks that can not all be refined. A comparison of the oscillations shown here with those obtained using ghost cells, rather than disjoint nodes, may also be beneficial. These remain as possible areas for further research.

These experiments have highlighted the need for a sophisticated procedure for local refinement. Allowing shocks to cross interfaces between fine and coarse meshing can lead to spurious oscillations. The positioning of fine regions will need to be adapted throughout the calculation time, requiring the solution to be monitored automatically. Any fine regions created should completely surround the feature of interest and be large enough to prevent the feature escaping the fine region during the next Lagrangian step.

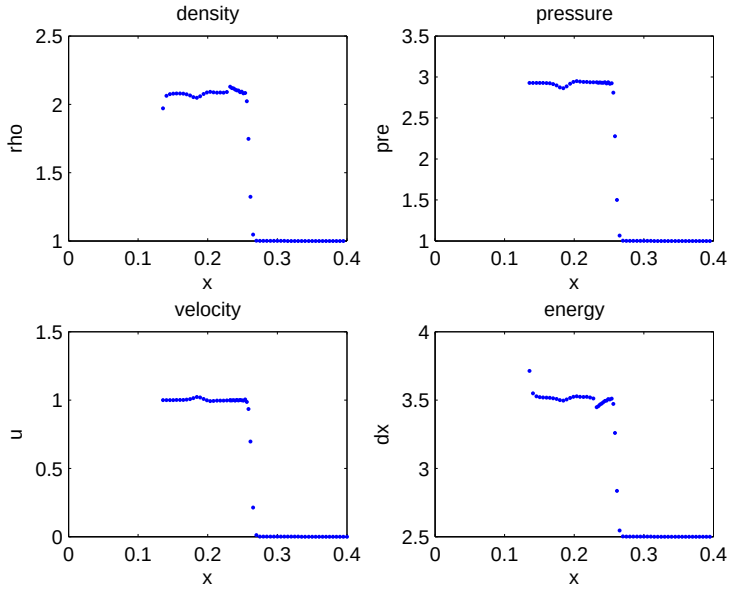

Figure 3.3: Coarse-to-fine 2 region test problem, mesh density ratio 1:2, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

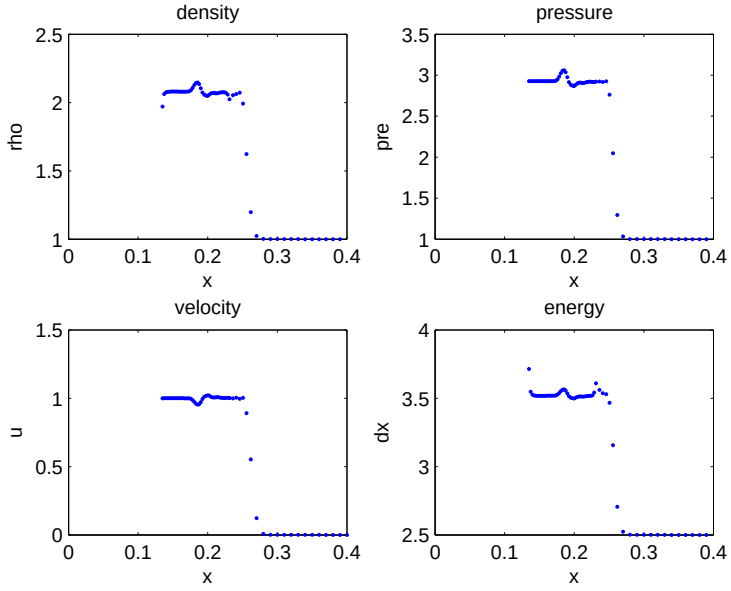

Figure 3.4: Fine-to-coarse 2 region test problem, mesh density ratio 2:1, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

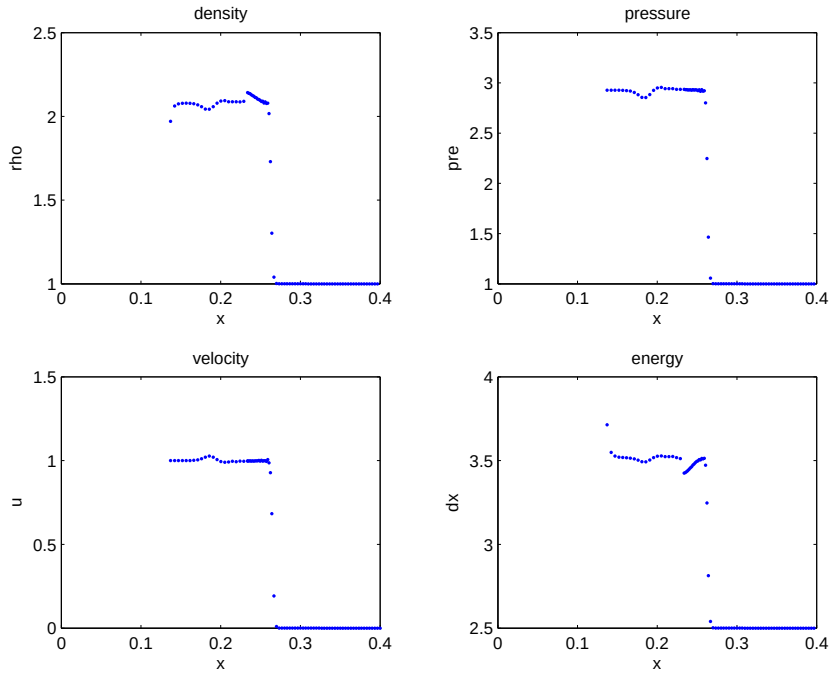

Figure 3.5: Coarse-to-fine 2 region test problem, mesh density ratio 1:3, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

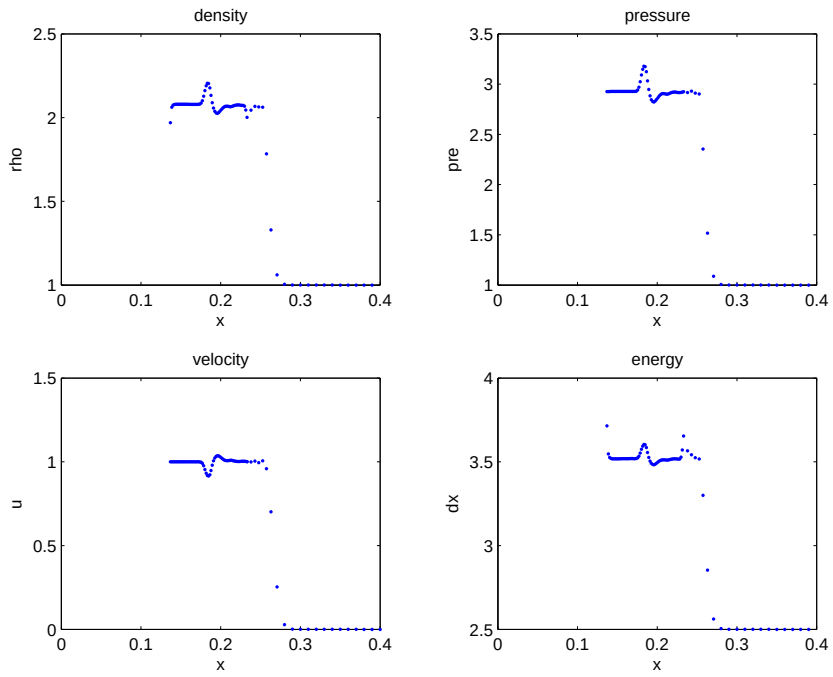

Figure 3.6: Fine-to-coarse 2 region test problem, mesh density ratio 3:1, t=0.13 with $c_l = 0.1$ and $c_q = 1.0$.

Chapter 4

Adaptive mesh technique

4.1 Adaptive mesh technique introduction

This chapter details our adaptive mesh technique, which can be seen as a combination of adaptive mesh insertion and adaptive mesh refinement. The significant features of the method are:

- cell by cell refinement
- AMI data structure, elements are inserted forming a combined mesh
- only solve on combined mesh
- disjoint nodes are used on interfaces between coarse and fine meshing
- isotropic refinement
- automatic refinement based on solution gradient
- one refinement level at present, technique should generalise to an arbitrary number of refinement levels
- no time refinement at present.

Adaptive mesh refinement, AMR, and adaptive mesh insertion, AMI, increase the mesh resolution locally around a feature of interest without coarsening the surrounding mesh. This is in contrast to penalty based methods such as ALE that draw elements into the area of interest leaving the surrounding mesh coarser and possibly under resolved. This

local approach avoids the refinement of the whole computational domain, which would prove too expensive.

A cell by cell approach to refinement is used instead of the clustering of elements into rectangular refinement blocks. Cell by cell refinement is often used in unstructured grids with triangular elements [16], [24] and sometimes used for rectangular element refinement [18], [16]. However the AMR ALE code developed by Anderson, Elliott and Pember [1] uses rectangular refinement blocks similar to those used by Quirk and Berger [19], [6], [9], [17]. 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 cell by cell approach is possible because the existing Lagrangian code is formulated using element by element rather than matrix techniques. An element to be refined is divided into four fine elements. A 1:2 refinement ratio was used instead of a 1:3 ratio because the spurious oscillations would be smaller if a shock did escape from the fine region, see Figure 3.3 and Figure 3.4. The element refinement in both directions helps to maintain an isotropic approach. This is in contrast to the adaptive mesh insertion scheme in CORVUS which only refines in one direction [2].

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. The new combined mesh, see Figure 4.1, contains both coarse and fine cells and is represented by the dynamic connectivity arrays, which change every time the mesh changes. This mesh contains all the present elements regardless of their refinement level and could be considered as a mesh where new elements have been inserted. This is very different to the usual AMR data structures [7], [9], [10], [19] that make every refinement block a new grid array. Our method does not require the solution to be stored at each level it is only stored for the present dynamic mesh containing both coarse and fine elements. As the program does not use a matrix representation we do not need to worry about requiring different arrays for different levels to avoid large sparse matrices. Furthermore the cell by cell approach would require so many small arrays that the usual AMR data structure would be impractical. The method is best described as a cross between a mesh insertion technique and AMR.

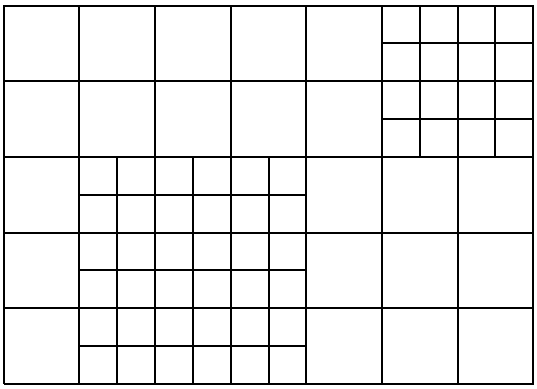

Figure 4.1: Diagram of the dynamic mesh with fine elements inserted.

Usually different levels are stored separately allowing them to be integrated in time individually. 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. Furthermore the extra complexity introduced by time refinement may outweigh the benefits. Therefore many argue that the increase in efficiency from the spatial refinement may be sufficient [1]. This is an area for further research and time refinement may be introduced at a later date. The elemental rather than matrix formulation of the code should allow a selected group of elements to be integrated in time without having to define them as a separate level with its own boundary data.

Disjoint 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 exactly like the Lagrangian disjoint node case. No ghost cells are required to transfer or interpolate data, since no separate grid levels exist the disjoint nodes are sufficient to transfer and constrain data between coarse and fine regions. This is important because requiring ghost cells in a cell by cell approach would prove very expensive. Future work could include comparing the efficiency of the two approaches and investigating which is better at dealing with small shocks passing out of the fine regions.

It is unusual to know in advance at what time and whereabouts an increase in resolution will be required. Despite this the CORVUS AMI technique is either manually triggered

or based on the elements aspect ratio [2] rather than being solution dependent. In this work we wish to sense when a solution feature is under resolved and add the required elements automatically. This also allows a feature to be tracked by a finer region as it moves, which would be impossible to do manually.

Although the present method contains only one level of refinement, i.e) the coarse mesh and one fine level, the ultimate aim will be to have an arbitrary number of refinement levels. The technique detailed has been developed with this in mind and should generalise fairly easily when the n-level coding is added.

4.2 Adaptive mesh notation

At this point we introduce some notation for the grids, groups of nodes and groups of elements involved in the adaptive mesh scheme.

M	The set of the dynamic grid nodes
E	The set of the dynamic elements
M_0	The set of the original coarse grid nodes, where $M_0 \subset M$
E_0	The set of the original coarse elements
M_f	The set of the dynamic fine nodes, where $M_f \subset M$
E_f	The set of the dynamic fine elements, where $E_f \subset E$
E_c	The set of the dynamic coarse elements, where $E_c \subset E$
E_r	The set of the elements to be refined, where $E_r \subset E_0$
E_d	The set of the elements to be derefined, where $E_d \subset E_0$

Using this notation the dynamic grid node set and element set are such that

$$M = M_0 \cup M_f \tag{4.1}$$

$$E = E_f \cup E_c. \tag{4.2}$$

Furthermore it can be noticed that

$$M_0 \cap M_f = \emptyset. \tag{4.3}$$

4.3 Adaptive mesh data structures

The coarse number of nodes, `nnodcoarse`, and coarse number of elements, `nelcoarse`, are retained for all time. The number of regions in the problem remains the same. The following arrays representing the original coarse grid, with nodes belonging to M_0 and elements belonging to E_0, are retained unchanged for all time:

- coarse element-node connectivity array (1:4,1:`nelcoarse`)
- coarse element-element connectivity array (1:4,1:`nelcoarse`)
- coarse node-node connectivity array (1:4,1:`nnodcoarse`)
- coarse maximum element in region vector (1:`nreg`).

The Lagrangian step only uses the dynamic variables as they represent the present status of the mesh including both fine and coarse cells. The dynamic number of nodes, `nnod`, and the dynamic number of elements, `nel`, are changed as new elements are added. The connectivity array bounds are at the moment set to the maximum number of elements and nodes possible for one level of refinement in the problem, where `mel`=4×`nelcoarse` and `mnod`=5×`nnodcoarse`. These are overestimates at the moment and hence waste storage space. It should be possible to use dynamic pointers to solve this in a later version of the code for n levels of refinement. The dynamic versions of the arrays are:

- dynamic element-node connectivity array (1:4,1:`mel`)
- dynamic element-element connectivity array (1:4,1:`mel`)
- dynamic node-node connectivity array (1:4,1:`mnod`)
- dynamic maximum element in region vector (1:`nreg`).

The array `cellstatus`(1:`nelcoarse`) records the status of each element, it has entry 1 if the element is fine and 0 if the element is coarse.
An element that is to be refined is divided into four quarters to create new elements. The three new elements are renumbered dynamically as they are created. Hence a coarse element with general element number 6, see Figure 4.2, becomes four fine elements with general element numbers 6, 7, 8, 9 in an anticlockwise direction. All the elements with

higher numbers have their general element numbers shifted to fit these in. Therefore even though an element has not been refined its element number may have changed.
The new element numbers are tracked using a new connectivity array `coarsemem`(1:`nelcoarse`,1:4). For each original coarse element this represents the new element number if the element is coarse and the four new element numbers if it is fine. Therefore if coarse element 6 is refined and coarse element 7 is not, the entry for row 6 would read 6, 7, 8, 9, where as row 7 would have the single entry 10 (the new general element number). This allows the program to reference between the original element numbers and the new element numbers. The element centred solution variables are dynamically given in terms of the new element numbers.
The general node numbers for the new nodes are allocated once all the new elements have been defined, since a knowledge of whether the surrounding elements are coarse or fine is required. The new node numbers are placed at the end of the list of nodes and the original coarse node numbers are never changed. The new centre node is numbered first and then the surrounding four edge nodes are numbered in an anticlockwise manner starting at the bottom, see Figure 4.3. A new node number is given only if the node has not been numbered already. The state variables that are defined at the nodes are once more dynamically given in terms of the new node numbers.
Nodes which have only three nodal neighbours and are not on external boundaries are recorded as disjoint. The disjoint node connectivity arrays are then updated. Finally the element-node, element-element and node-node dynamic connectivity arrays are changed. The maximum element number for the region is also recorded.

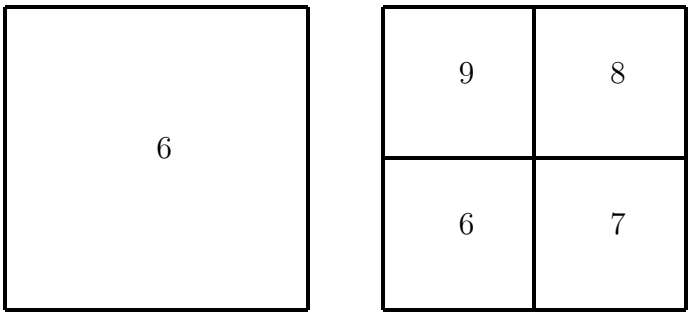

Figure 4.2: Element numbering changes when an element is inserted, the existing element numbers are all shifted to fit the new elements in.

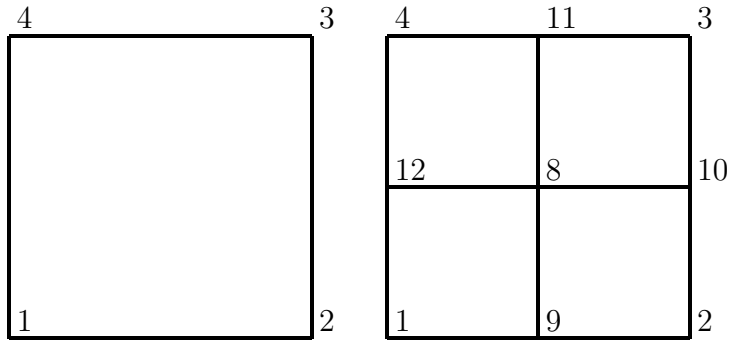

Figure 4.3: Node numbering for new nodes, the original coarse node numbers never change. Only fine node numbers are ever shifted to fit new nodes in.

4.4 Refinement criteria

Many properties can be used to determine if an element should be refined or derefined, for example the element's aspect ratio or the solution gradient. In this work the solution gradient is used since it is the solution we are ultimately interested in. In particular the element density was found to be most reliable because it will identify contact discontinuities. Firstly values for the element densities on the coarse grid are obtained. If the element is coarse this is straight forward. For fine elements, the density values of the four fine elements are averaged to provide a coarse density,

$$\rho_{coarse}(e) = \begin{cases} \rho(e) & \text{if cellstatus}(e) = 0 \\ \frac{\sum_{i=1}^{4} \rho(i)V(i)}{\sum_{i=1}^{4} V(i)} & \text{if cellstatus}(e) = 1, \quad \forall e \in E_0, \end{cases} \tag{4.4}$$

where the i's are the four elements dividing the coarse element e and V is the element volume.

The density change between a coarse grid element and its four nearest neighbours is calculated and the maximum change is recorded in the array `coarsecrit`(1:`nelcoarse`). If the value of coarsecrit is above the refinement parameter the element is flagged for refinement. If the value of `coarsecrit` is below the derefinement parameter the fine elements that make up that coarse element are flagged for derefinement,

$$\begin{aligned} &\text{if coarsecrit}(e) > \text{amrreftol and cellstatus}(e) = 0 \ \text{ then } e \in E_r, \forall e \in E_0, \\ &\text{if coarsecrit}(e) < \text{amrdereftol and cellstatus}(e) = 1 \ \text{ then } e \in E_d, \forall e \in E_0. \end{aligned} \tag{4.5}$$

Note that the refinement and derefinement parameters must differ otherwise elements will derefine too quickly.
The number of elements to be refined/derefined is recorded as `nelref`/`nelderef`. The arrays `elref` and `elderef`, with lengths `nelref` and `nelderef`, record the elements to be refined and derefined. The entries of `cellstatus` are then updated.

4.5 Buffer cells

The previous chapter highlighted the spurious oscillations that can be generated if a shock crosses an interface between fine and coarse meshing. It is important therefore that the features of interest can not move out of the region of fine elements within the 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.
To insure this buffer zone is not derefined immediately a second version of coarsecrit is checked in which buffer elements take the coarsecrit value of the element they surround so that they have values above the derefinement parameter.
Further work is required on buffer elements to ensure that any derefinement does not derefine too far and allow, for example a rarefaction wave, to escape the fine patch during the Lagrangian step. 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 the whole domain in a problem with many shocks.

4.6 Solution transfer

When elements are refined or derefined the solution vectors for both nodal and cell centred variables must be altered. At the moment only a first order method of solution transfer is implemented. This allows the adaption procedure to be tested before a more complicated approach is taken. Solution transfer involves;

- calculating fine variables from coarse values during refinement
- calculating coarse variables from fine values during derefinement

- shifting the solution vector entries for cell centred variables when elements are added
- shifting the nodal solution vector for nodal variables when nodes are removed during derefinement or added during derefinement.

The coarse cell centred variables for elements being derefined are computed first. This is because during derefinement only coarse cell centred variables need to be calculated. For nodal variables the fine values are simply removed. The solution ϕ for the new coarse element is given by the weighted average of the four fine elements' solutions. Using X for the weighting variable the formula that is applied for each element $e \in E_d$ is

$$\phi(p) = \frac{\sum_{i=1}^{4} \phi(i)X(i)}{\sum_{i=1}^{4} X(i)}, \tag{4.6}$$

where the i's denote the four elements that are being derefined and p is the unshifted element number. Whether a volume or mass weighted average is used depends on the state variable. Volume weighting is used for density while energy is mass weighted. The procedure uses the previous step's `coarsemem`, stored as `coarsememold`.

For nodal variables, only the fine node solution values require shifting when refinement or derefinement occurs. The shifting utilises the previous and present `coarsemem` and node-element connectivity arrays. The resulting nodal variables now correspond to the present mesh's nodes.

The nodal variables for those nodes created during refinement are then considered. If a node is introduced in the centre, $n_c \in M_f$, the solution ϕ is given by an average of the

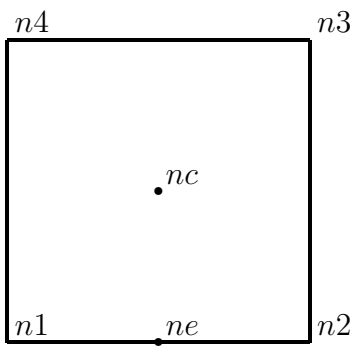

Figure 4.4: Diagram to illustrate the calculation of new fine nodal variables.

four corner nodes $n_{1,2,3,4} \in M_0$

$$\phi(n_c) = \frac{1}{4}(\phi(n_1) + \phi(n_2) + \phi(n_3) + \phi(n_4)). \tag{4.7}$$

If an edge node, $n_e \in M_f$, is introduced between two coarse nodes n_1 and $n_2 \in M_0$ the solution ϕ is an average of the coarse nodes' solutions

$$\phi(n_e) = \frac{1}{2}(\phi(n_1) + \phi(n_2)). \tag{4.8}$$

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

The disjoint node positions are clamped so that they lie on the interface half way between the non-disjoint nodes. Their velocity is clamped as an average of the non-disjoint node velocities as detailed in the disjoint node chapter. Element volumes are then calculated. The element variables for those elements created during refinement are calculated and shifted simultaneously. The coarse element is assumed to have constant solution and the four fine elements that are created take this solution value. Therefore for an element $e_c \in E_r$ that is to be refined

$$\phi(i) = \phi(e), \quad \text{for i=1,2,3,4,} \tag{4.9}$$

where e represents the dynamic element number of e_c that is found using the previous `coarsemem` and the $i's \in E_f$ represent the new fine elements. At the same time the remaining cell centred solution values are shifted to correspond to the new mesh's general element numbers. This uses the previous and present `coarsemem`.

The above technique is only first order accurate. Further work is required to develop a method of second order solution transfer. One possibility would be a second order conservative rezoning approach such as outlined by Ramshaw [20] and Dukowicz [14]. An alternative approach would be to adapt a continuous advection rezone method [23], [2], [4], [5]. Without investigation it is unclear which method would be the most successful.

4.7 Adaptive mesh refinement with Christensen's artificial viscosity

In order to use Christensen's artificial viscosity with the adaptive mesh scheme some alterations to the Christensen routine 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.

Firstly the element-element stencil must be global with connections across interfaces and including both the coarse and fine elements. The element-element connections across coarse-to-fine or fine-to-coarse interfaces are set to -1. This acts as a pointer to a new disjoint element-element connectivity array. For each disjoint node this specifies the coarse element number, the side that is disjoint with respect to the coarse element and the two fine elements. For the arrangement shown in Figure 4.5 row nd would have entries $ec, 2, ef1, ef2$.

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

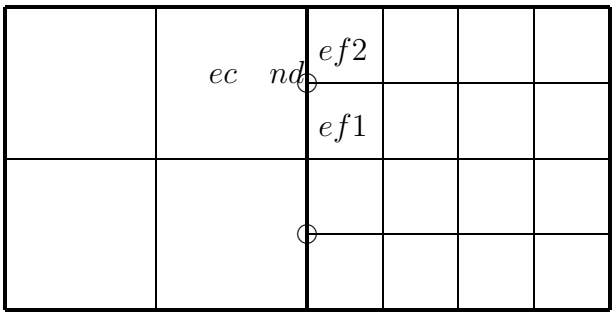

Figure 4.5: Diagram of a course-to-fine interface to show entries of disjoint element-element array. The empty circle indicates the disjoint node.

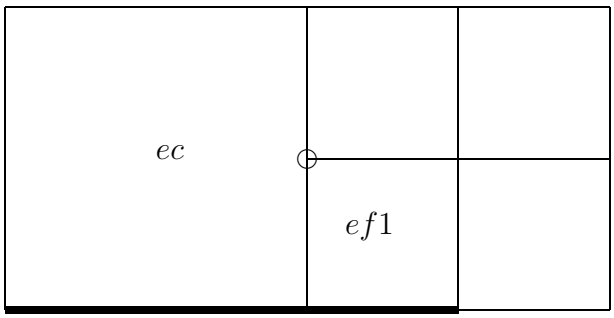

Figure 4.6: Diagram of a course-to-fine interface showing the correct aligning element for limiting along the bottom edge.

element for the edge must be identified. Referring to Figure 4.6, when limiting the edge artificial viscosity for element ec's bottom edge the correct aligning element is ef. When the routine tries to find the element neighbour along that edge the element-element value will be -1 informing us that the disjoint element-element array must be used to find the correct fine element. Once this is done the rest of the Christensen method remains unchanged.

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 which 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 4.7 for the 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.

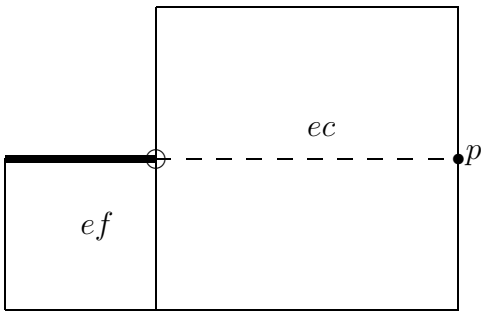

Figure 4.7: Diagram of a fine-to-coarse interface. Limiting along top edge of the fine element requires the continuation of this edge through the neighbouring coarse element to point p.

4.8 Adaptive mesh results

4.8.1 One dimensional test problems

A piston problem with left boundary velocity 1.0, pressure 1.0 and density 1.0 was run with a base grid of 50×5. The refinement parameter was 0.1 and the derefinement parameter was 0.075. This problem verified that both refinement and derefinement worked well. The fine cells formed into a region which followed and encased the shock. Buffer cells were used in this calculation to stop the shock escaping from the fine region. The problem was run with both the bulk artificial viscosity and the Christensen artificial viscosity. The shock width was comparable to those obtained with a 100×10 uniform grid. The AMR calculation required only 325-340 elements in comparison to the 1000 elements needed on a uniform grid. Therefore with one level of refinement the number of elements is fairly constant in this problem as we would expect. The only noticeable difference between the two artificial viscosities is that the Christensen density profile is smoother than the bulk profile although the shock is slightly wider to enable monotonicity to be enforced. The adaptive mesh and density profiles in the bulk viscosity case are shown in Figure 4.8 and Figure 4.9

A two region Sod Shock tube problem was used to see if the adaptive mesh could track a rarefaction fan, contact and shock wave. The refinement parameter in this case was reduced to 0.05 and the derefinement parameter was 0.01. When no buffer cells were included three distinct areas of refinement following these features were formed by the fine cells. However spurious oscillations appeared when the features escaped

from the fine regions. To remove this problem buffer cells were introduced. The results then showed one region of fine cells encasing both the rarefaction fan and the contact discontinuity and another region surrounding the shock. The density was used for the refinement criteria because it would vary around the contact discontinuity unlike a pressure refinement criteria.

The shock width and relative density error were comparable to a 100×10 uniform grid calculation. The adaptive mesh calculation increased the number of elements over time as the rarefaction and shock regions grew, see Figure 4.11. The maximum number of elements reached 550 because the fine region was quite extensive. However this is still less than half the elements required for a 100×10 uniform grid. The total number of elements is clearly dependent on the problem tested.

The calculation was run with both artificial viscosities the mesh adaption was almost identical, with the $t = 0.2$ meshes being the same as shown in Figure 4.10. The density profile was much smoother for the Christensen viscosity shown in Figure 4.13 although the shock was slightly thinner with the bulk viscosity Figure 4.12. Both density profiles looked almost identical to those obtained with a 100×10 uniform grid calculation. This problem also verified that the adaptive mesh code could be run with more than one region.

4.8.2 Two dimensional Riemann problem

This problem was rerun with adaptive meshing to test the performance of two dimensional refinement. The initial grid was 50×50. The refinement parameter was reduced to 0.04 and the derefinement parameter was 0.01. The Christensen artificial viscosity was used with $c_l = 0.3$ and $c_q = 0.65$. The buffer cells were included to guarantee that the features of interest remained in the fine areas.

The adaptive mesh at t=0.2 is shown in Figure 4.14. The areas of finer meshing encased the shocks and the oval region of high density as they evolved. This problem highlighted the cell by cell nature of the adaptivity as single derefined elements are present in the oval density region. The density contours in Figure 4.16 are extremely similar to those obtained with a uniform 100×100 grid. This is better than expected considering that only a first order method of data transfer is employed. No spurious oscillations or reflections are seen in the grid.

The adaptive mesh calculation took 6 minutes of cpu time as opposed to the uniform 100×100 grid calculation that took 17 minutes. Clearly the adaptive mesh reduces the calculation time by around 1/3. Throughout the run the number of elements increases at a fairly constant rate as the high density oval region grows, see Figure 4.15. Therefore although 70% of the domain is refined by t=0.2 the smaller number of elements at earlier times substantially reduces the run time. The total number of elements for the whole adaptive calculation is just below 60% of the value for the uniform grid calculation.

A remaining area to be considered is whether derefinements involving only a couple of elements are more expensive than the saving they provide from reducing the number of elements. In conclusion this example has shown how successfully the two dimensional refinement can reduce the calculation time without degrading the solution.

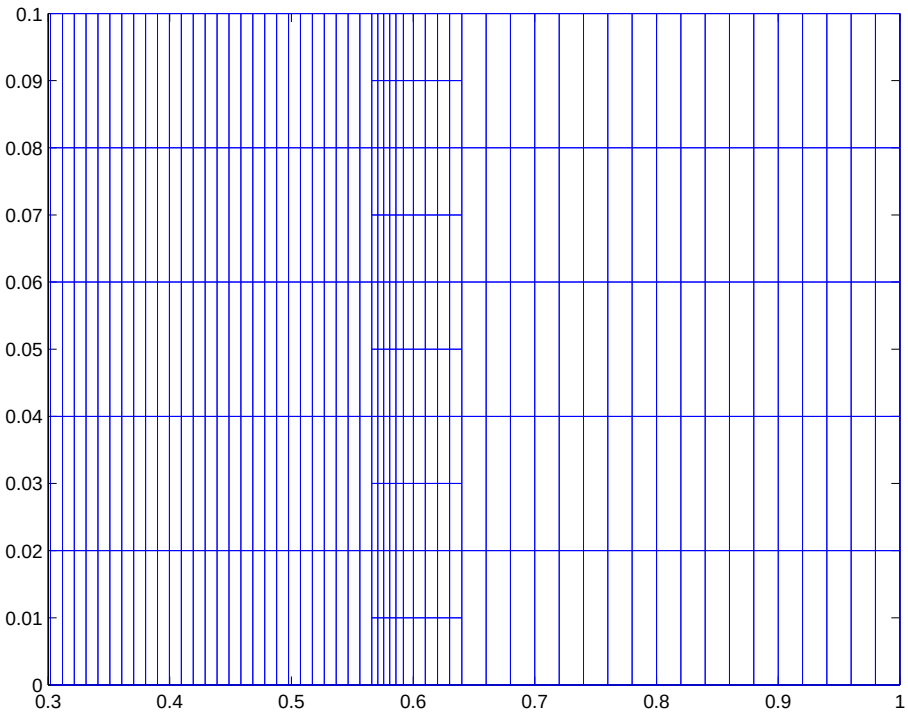

Figure 4.8: Two level adaptive mesh for piston problem with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$ at t=0.3.

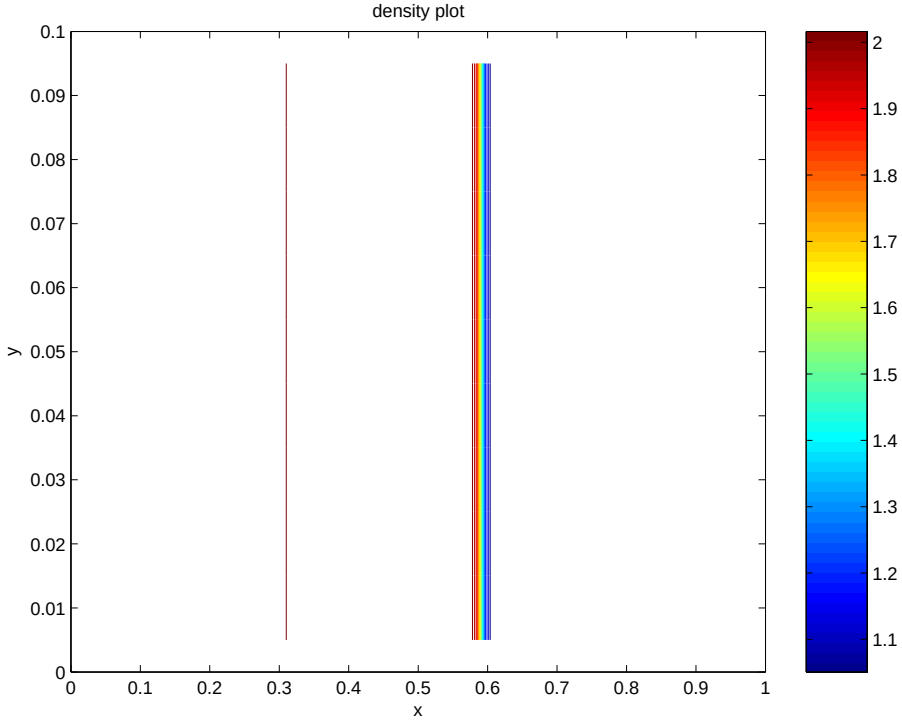

Figure 4.9: Two level adaptive mesh density contours for piston problem with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$ at t=0.3.

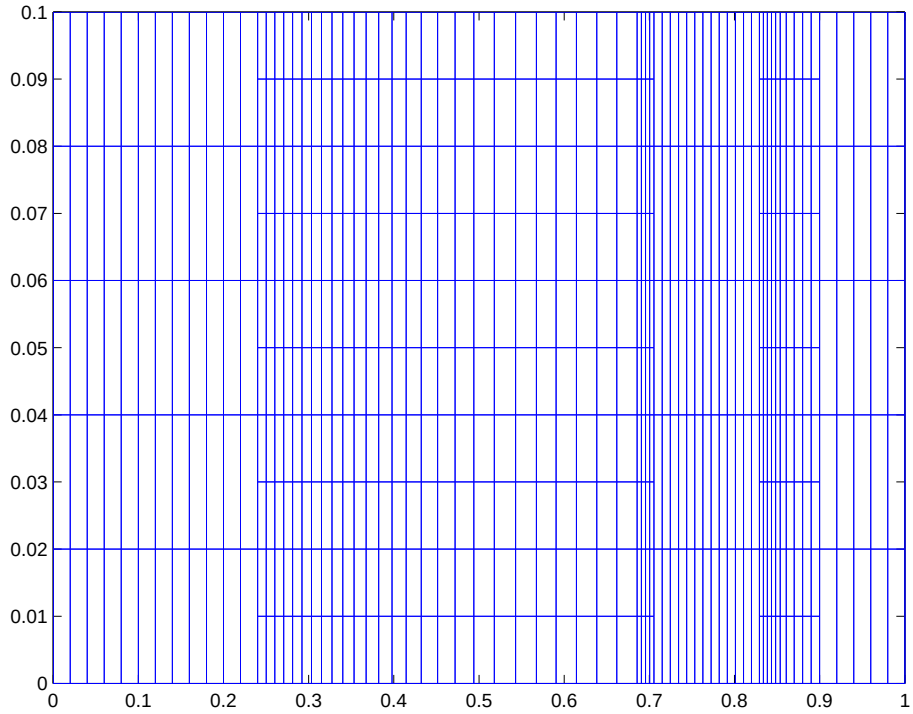

Figure 4.10: Two level adaptive mesh for Sod problem with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$ at t=0.2.

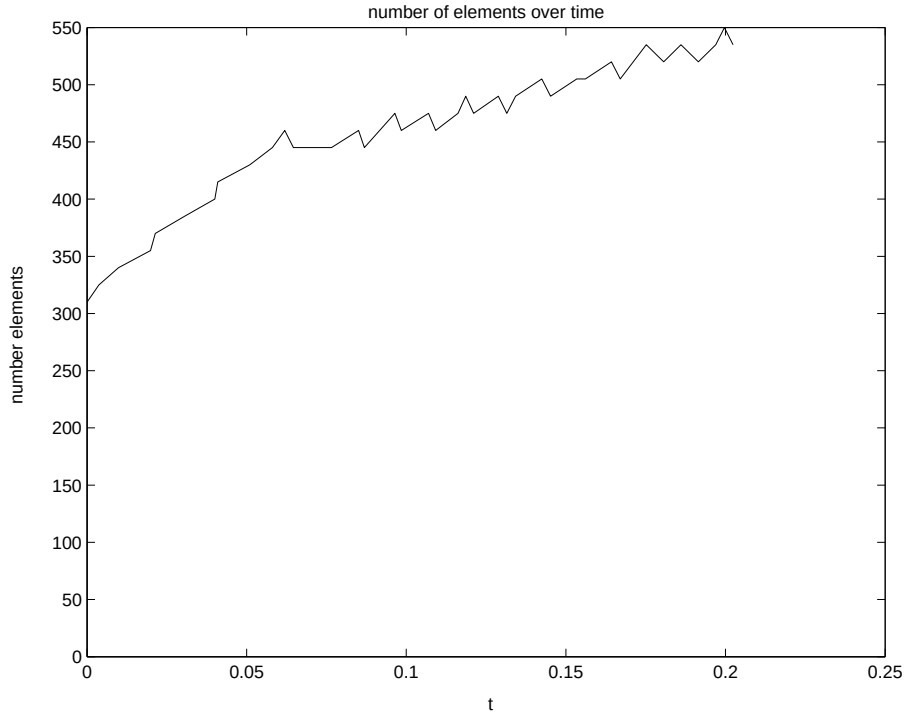

Figure 4.11: Variation of number of elements over time for Sod problem with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$.

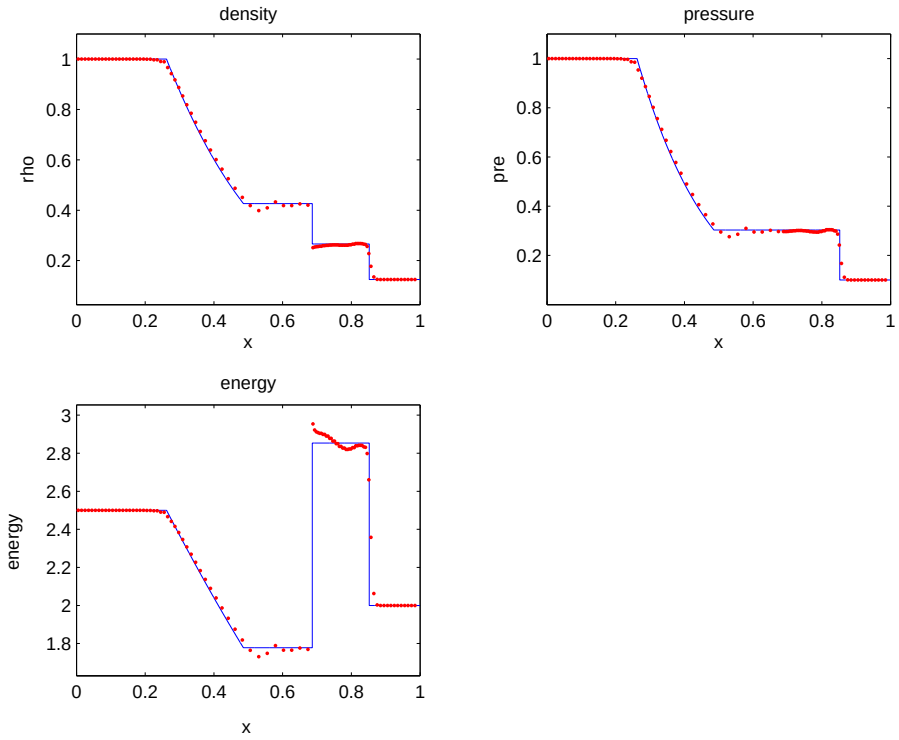

Figure 4.12: Two level adaptive mesh results for Sod problem with bulk artificial viscosity coefficients $c_l = 0.1$ and $c_q = 1.0$ at t=0.2.

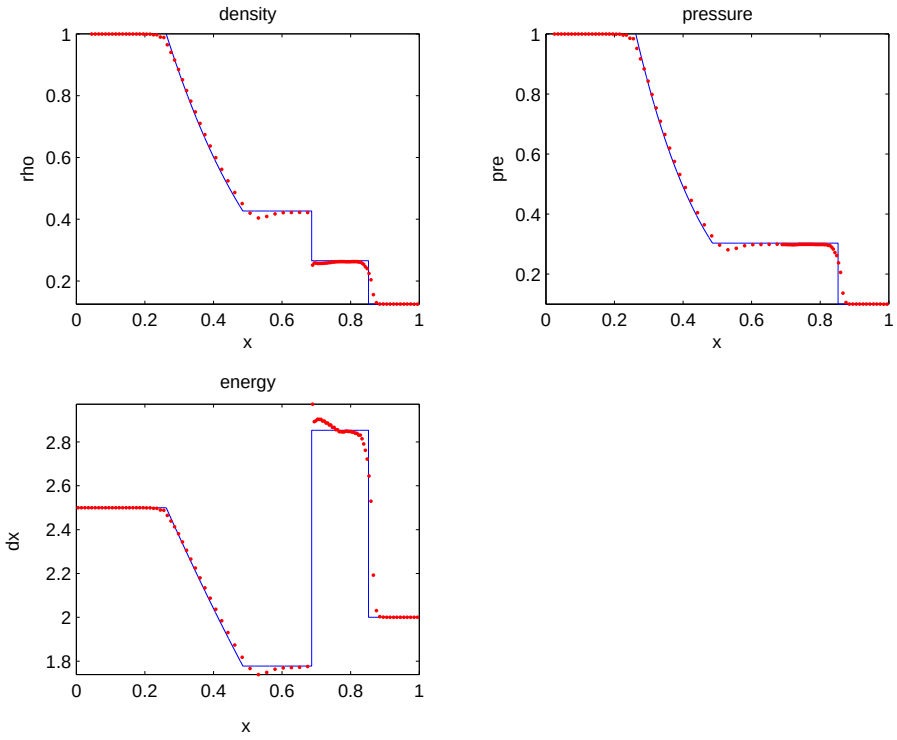

Figure 4.13: Two level adaptive mesh results for Sod problem with Christensen artificial viscosity coefficients $c_l = 0.5$ and $c_q = 0.75$ at t=0.2.

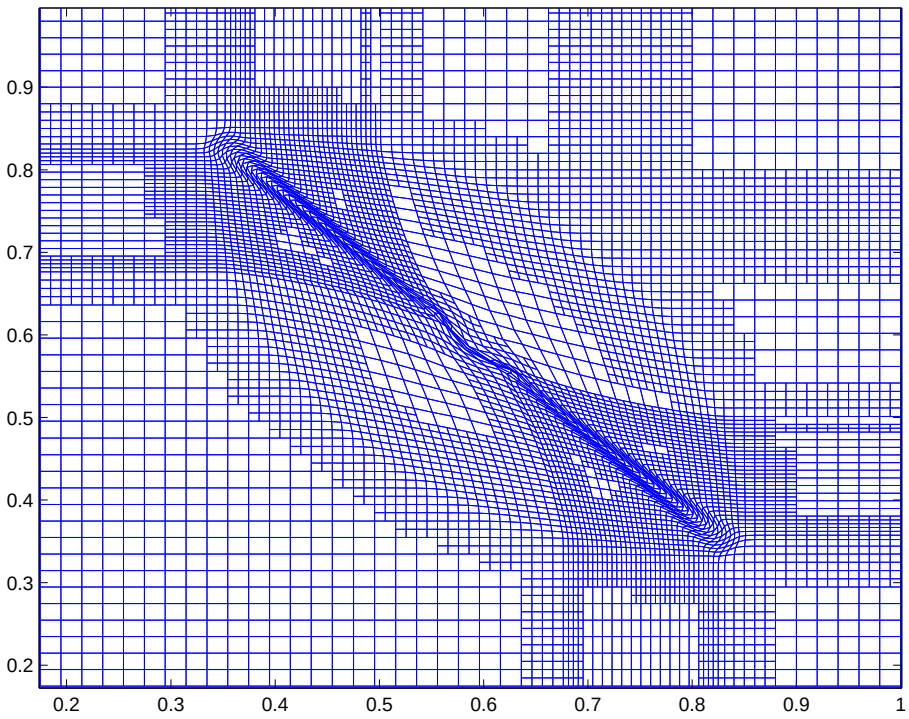

Figure 4.14: Two level adaptive mesh for 2D Riemann problem with Christensen artificial viscosity at t=0.2.

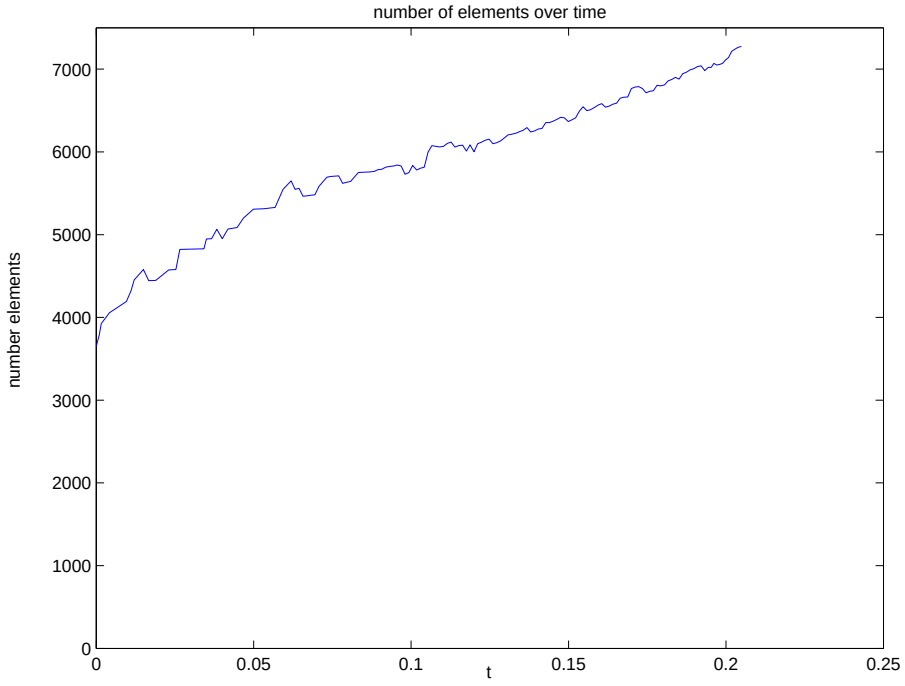

Figure 4.15: Variation of number of elements over time for 2D Riemann problem with Christensen artificial viscosity at t=0.2.

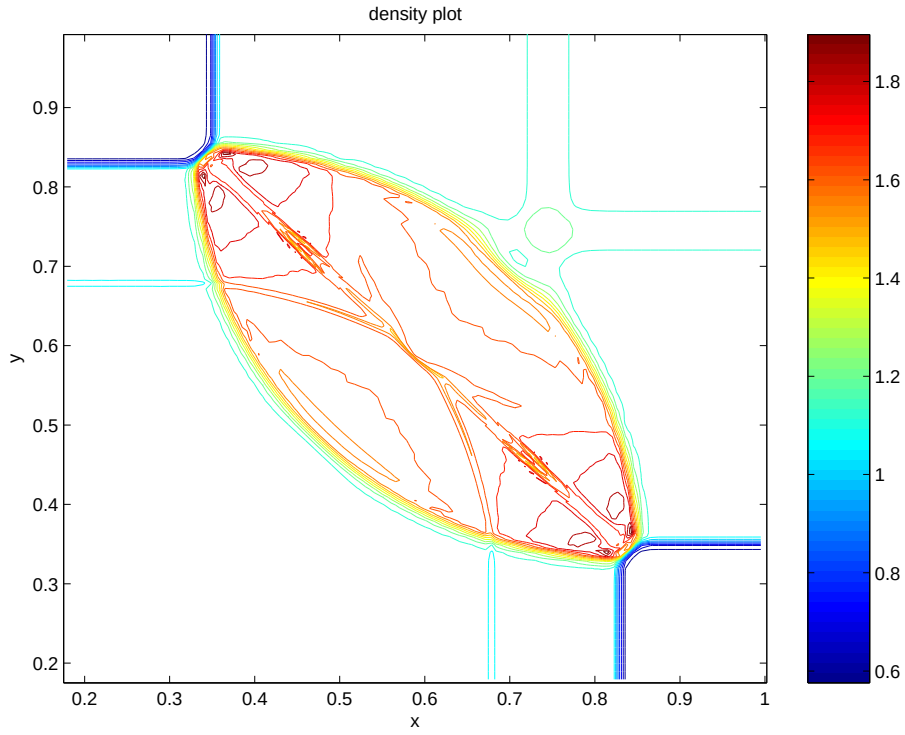

Figure 4.16: Two level adaptive mesh calculation density contours for 2D Riemann problem with Christensen artificial viscosity at t=0.2.

Chapter 5

Conclusions and Further Work

An adaptive mesh technique has been developed that builds upon a staggered grid Lagrangian code. The Lagrangian code uses spatial finite elements and predictor-corrector time discretisation. The Lagrangian code has been validated on a Sod shock tube problem. An Axisymmetric form of the code has been tested on a radial Sod problem. Good results were obtained considering the use of artificial viscosity. A two dimensional version of Christensen's artificial viscosity gave less oscillatory results than the bulk artificial viscosity, although the shock was spread over a larger number of elements. Further investigations into the artificial viscosity coefficients may be required.

A two dimensional Riemann problem was also run to validate the two dimensional performance of the code. Excellent results were obtained although the artificial viscosity did cause some variation in the constant density regions. This was caused by the production of too much artificial viscosity in the compressed elements during the first few time steps. The Lagrangian grids became fairly distorted by the end of the run time. This problem highlights the necessity of performing a mesh relaxation and advection remap step to reduce the mesh tangling.

Disjoint nodes have been introduced to allow changes in mesh density. An investigation of disjoint nodes with static mesh connectivity showed that spurious oscillations can be generated if shocks are allowed to cross interfaces between different mesh densities. This motivated the need for an adaptive method of mesh refinement.

A cell by cell adaptive mesh technique has been developed that locally refines elements without coarsening the surrounding grid. This uses a data structure based on the original mesh and the partially refined mesh at that time. Different level grids are not

stored separately therefore the technique is a cross between adaptive mesh refinement and adaptive mesh insertion. The method automatically refines elements where the density gradient is high. Disjoint nodes are used at the interfaces between coarse and fine elements, no ghost cells are required. The present method only considers one level of refinement and is not adaptive in time.

Excellent results were achieved for a piston problem and Sod's shock tube. The mesh refined or derefined to follow the features of interest. The inclusion of buffer cells has made sure that interesting features are always fully encased in a fine area, therefore no spurious reflections are generated. The adaptive calculations show the same shock width and relative error as calculations performed with a uniform grid at the finer resolution. The adaptive run times are significantly faster. However, the number of coarse and fine elements is clearly dependent on the problem.

Two dimensional refinement was used very successfully with the two dimensional Riemann problem reducing the run time by 1/3. The results obtained were almost identical to those of a uniform calculation at the finer resolution. The refined region encased the areas of high density and shocks and echoed the oval shape of the high density region very well. Individual element derefinement was observed highlighting the cell by cell nature of the technique. Further investigation is required into the cost of refining or derefining individual cells compared to the saving gained from using less elements.

An adaptive mesh technique has been successfully developed for the Lagrangian code. However a Lagrangian mesh may distort severely as the material deforms. This can lead to very small time steps and errors because of the poor mesh quality. The next main area of research will be to combine the adaptive methodology with an arbitrary Lagrange Eulerian scheme to enable the mesh to be relaxed. This will require investigation into the stencil used for equipotential relaxation. Advection methods in areas of the grid with fine and coarse cells will also have to be considered.

Further work is required to extend the method for a general number of refinement levels, implement second order data transfer and investigate the effects of the refinement parameters and buffer cells. The conservation of the scheme must also be considered.

Bibliography

[1] 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.

[2] A. Barlow. An adaptive multi-material arbitrary Lagrange Eulerian algorithm for computational shock hydrodynamics, Swansea, 2002.

[3] D. Benson. Computational methods in Lagrangian and Eulerian Hydrocodes, University of California, 1990.

[4] D. Benson. "An efficient, accurate simple ALE method for nonlinear finite element programs", Computer methods in applied mechanics and engineering, 72, 305-350, 1989.

[5] D. Benson. "Momentum advection on a staggered mesh", Journal of Computational Physics, 100, 001-019, 1992.

[6] M. J. Berger, P. Colella. "Local adaptive mesh refinement for shock hydrodynamics", Journal of Computational Physics, 82, 64-84, 1989.

[7] M. J. Berger. "Data structures for adaptive grid generation", SIAM Journal scientific and statistical computing, 7 (3), 904-916, 1986.

[8] M. J. Berger, A. Jameson. "Automatic adaptive grid refinement for the Euler equations", AIAA Journal, 23 (4), 561-568, 1985.

[9] M. J. Berger, J. Oliger. "Adaptive mesh refinement for hyperbolic partial differential equations", Journal of Computational Physics, 53, 484-512, 1984.

[10] M. J. Berger. Adaptive mesh refinement for hyperbolic partial differential equations, Stanford University, 1982.

[11] E. Caramana, M. Shashkov, et al. "Formulations for artificial viscosity for multi-dimensional shock wave computations", Journal of Computational Physics, 144, 70-97, 1998.

[12] R. Christensen. Godunov methods on a staggered mesh - an improved artificial viscosity, private communication cited in [2], 1991.

[13] P. Colella and P. Woodward. "The piecewise parabolic method for gas-dynamical simulations", Journal of Computational Physics, 54, 174-201, 1984.

[14] 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.

[15] C. Hirt, A. Amsden, et al. "An arbitrary Lagrange-Eulerian computing method for all flow speeds", Journal of Computational Physics, 14 (3), 227-253, 1974.

[16] M.J. Marchant and N.P. Weatherill. "Adaptive techniques for compressible inviscid flows", Computer Methods in Applied Mechanics and Engineering, 106, 83-106, 1993.

[17] 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.

[18] S. Popinet. "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries", Journal of Computational Physics, 190, 572-600, 2003.

[19] J. J. Quirk. An adaptive algorithm for computational shock hydrodynamics, Cranfield, 1991.

[20] J. Ramshaw. "Simplified second-order rezoning algorithm for generalized two dimensional meshes", Journal of Computational Physics, 67, 214-222, 1986.

[21] A. Kurganov and E. Tadmor. Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers, pages 1-20, 2000.

[22] Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag.

[23] 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.

[24] D. Vollmer. Adaptive mesh refinement using subdivision of Unstructured elements for conservation laws, Msc thesis University of Reading, 2003.

[25] 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.

[26] M. Wilkins. "Use of artificial viscosity in multidimensional fluid dynamic calculations", Journal of Computational Physics, 36, 281-303, 1980.