E N ER Y OF READ N

DEPAR MEN OF MA EMA

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

cvoo295o295e-0.14787(e)-39-334.145(i)0.218504(c)27

the mesh remains fixed throughout time and the material moves relative to the mesh. A hierarchical set of grids representing di□erent 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

4.103(d)-0.312673(e) 0.0901136(v) 27.7408(e) 0.0901136(l) 0.218509(o) 1.24507(t)]TJ 3.819 18.9091 Td(y) 0.211426(i) 0.0901136(e) 27.7408(e) 0.0901136(l) 0.218509(o) 1.24507(t)]TJ 3.819 18.9091 Td(y) 0.211426(i)

Inbec

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 di□erent 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 di□erent 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. Bu□er

Chapter 2

The Lagrangian Scheme

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

$$\frac{D}{D} = - \nabla \cdot \mathbf{u} \quad (2.1)$$

$$\frac{D\mathbf{u}}{D} = -\nabla p \quad (2.2)$$

$$\frac{D\epsilon}{D} = -p \nabla \cdot \mathbf{u}, \quad (2.3)$$

where

$$\frac{D}{D} = — + \mathbf{u} \cdot \nabla \quad (2.4)$$

is the Lagrangian derivative, is the density, u is the velocity vector, p is the pressure and ϵ

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.

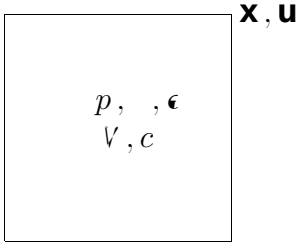


Figure 2.1: Diagram showing the positions of the nodal and element centred variables.

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,

$$\Delta \quad C / \sqrt{c^2 + 2q/} \, . \tag{2.7}$$

- Move the nodes to their half time step positions using

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

- 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),

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

  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,

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

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

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

where $d\Omega = d\ dy = de\ \ d\xi d$, J is the Jacobian and is or y in Cartesian coordinates.

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

$$\int_{\Omega} \sum \dot{} \ N\ N\, d\Omega = \dot{} \int_{-1}^{1}\int_{-1}^{1} N\, de\ \ d\xi d\ . \tag{2.16}$$

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

$$_e \dot{} \int_{-1}^{1}\int_{-1}^{1} N\, de\ \ d\xi d = (p_e + q_e)\int_{-1}^{1}\int_{-1}^{1} \frac{N}{}de\ \ d\xi d\ . \tag{2.17}$$

The term on the right is the mass contribution from element e to its local node . The term on the left is the force in the direction acting on node 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

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

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

$$\ = \sum \ N\ ; \qquad = \sum\ \ N\ . \tag{2.19}$$

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

$$_e \dot{} \int_{\Omega} d\Omega = -(p_e + q_e)\int_{\Omega} \sum \left(\ \frac{N}{} + \ \ \frac{N}{y} \right) d\Omega\,. \tag{2.20}$$

The left hand side is just the element mass M_e multiplied by . 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

$$\dot{} = -\frac{(p_e + q_e)}{M_e} \sum_{=1}^{4} \left[\ \left(\int_{-1}^{1}\int_{-1}^{1} \frac{N}{}de\ \ d\xi d \right) + \ \ \left(\int_{-1}^{1}\int_{-1}^{1} \frac{N}{y}de\ \ d\xi d \right) \right]\ . \tag{2.21}$$

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 ($, y$) can be mapped to the Axisymmetric coordinates ($z,$) by including a radial dependence in the finite element and volume equations. This e□ectively rotates the Cartesian domain 360° around the -axis. Cylindrical problems can then be run very e□ciently with only a two dimensional number of elements.
In the element volume equation $d\ dy$ becomes $d\ dz$ and we express the radius in terms of the finite element functions. The element volume then becomes

$$V_e = \sum_{=1}^{4} \left(\int_{-1}^{1}\int_{-1}^{1} N \ \mathrm{detJ} d\xi d \right). \qquad (2.22)$$

The energy equation in Axisymmetric

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 Δx has been replaced by $\nabla \cdot \mathbf{u}$.

$$q = c_q \, (\nabla \cdot \mathbf{u})^2 - c \, c \, |\nabla \cdot \mathbf{u}| \, ,$$

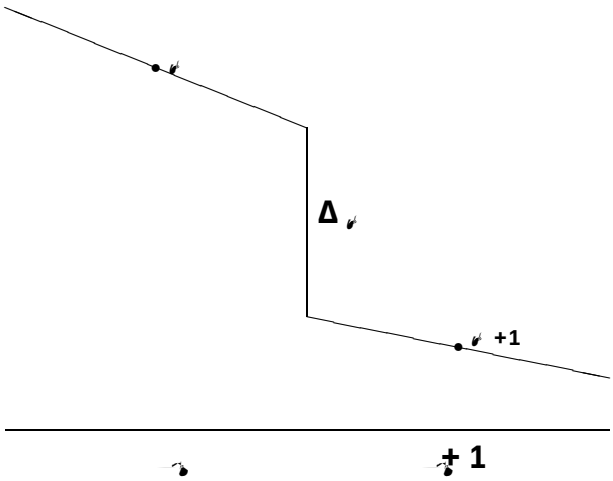
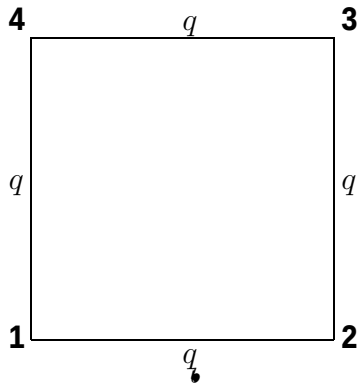
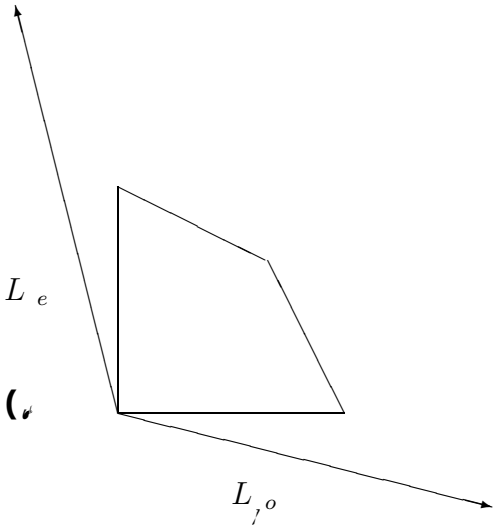


Figure 2.2: Diagram sho


4
q
3
q
q
1
2
q


L_{e}
$L_{,o}$

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

The limiter ψ is defined as

$$\psi = \max(0, \min(\frac{1}{2}(1+r), 2, 2r))$$

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.

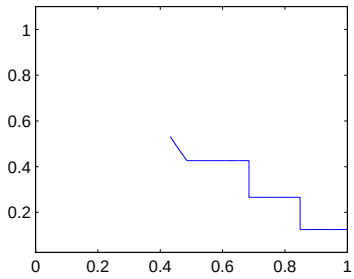
2.7.3 Two dimensional Riemann problem

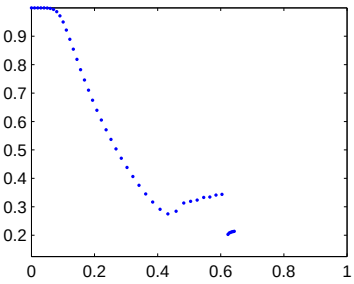
In this example di□erent constant initial conditions are given in each quarter of a square domain. The quarters 1,2,3 and 4 are defined respectively as 0.5 and y 0.5, 0.5 and y 0.5, 0.5 and y 0.5 and finally 0.5 and y 0.5. There are nineteen possible configurations of constant states divided by shocks, contact discontinuities, rarefaction fans or slip lines.

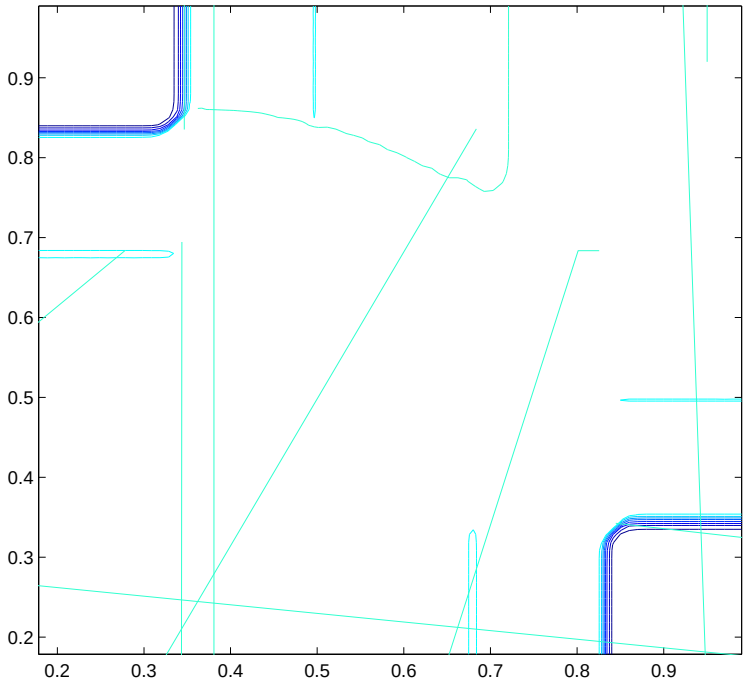
Figure 2.12 shows results for the following configuration involving 4 shocks,

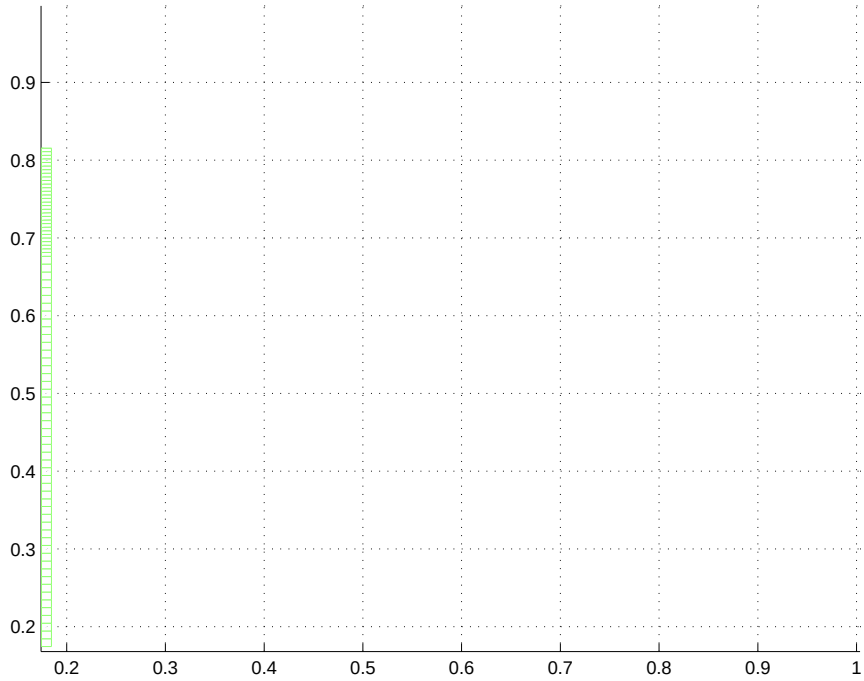
$$
= \begin{array}{cccc}
p_2 = 0.3500 & {}_2 = 0.5065 & p_1 = 1.1000 & {}_1 = 1.1000 \\
{}_{x2} = 0.8939 & {}_2 = 0.0000 & {}_{x1} = 0.0000 & {}_1 = 0.0000 \\
 & & & \\
p_3 = 1.1000 & {}_3 = 1.1000 & p_4 = 0.3500 & {}_4 = 0.5065 \\
{}_{x3} = 0.8939 & {}_3 = 0.8939 & {}_{x4} = 0.0000 & {}_4 = 0.8939 .
\end{array}
$$

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 di□erent 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


1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
0.2
0.4
0.6
0.8
1


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Chapter 3

Multiple regions and disjoint nodes

3.1 Regions with di□erent mesh densities and disjoint nodes

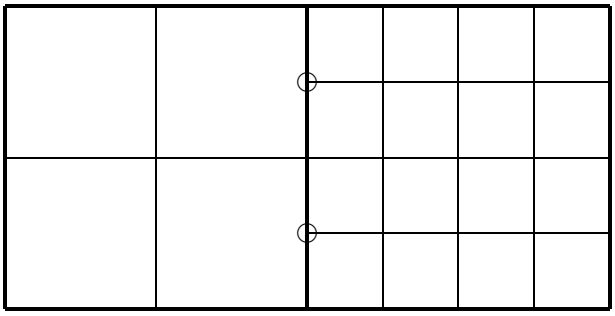
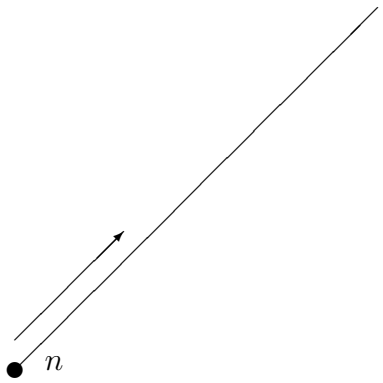


Figure 3.1: Diagram of a course-to-fine interface. Empty circles indicate disjoint nodes.

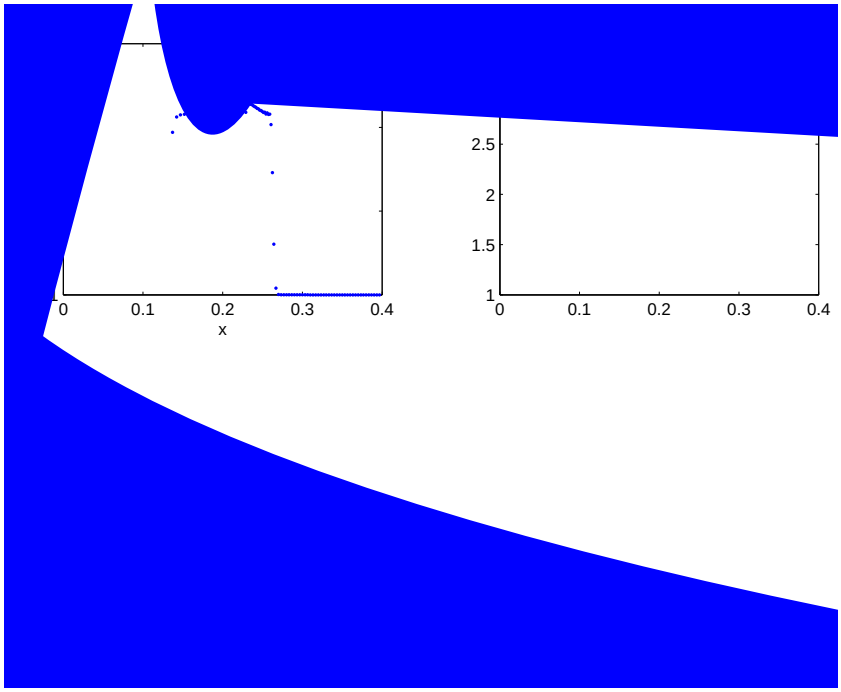


During the Lagrangian step, each disjoint node is slaved to lie the same ratio along the line joining the neighbouring non-disjoint nodes on the interface. Using the notation in

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.


2.5
2
1.5
1
0
0.1
0.2
0.3
0.4
x

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 f

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

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** $_0$

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

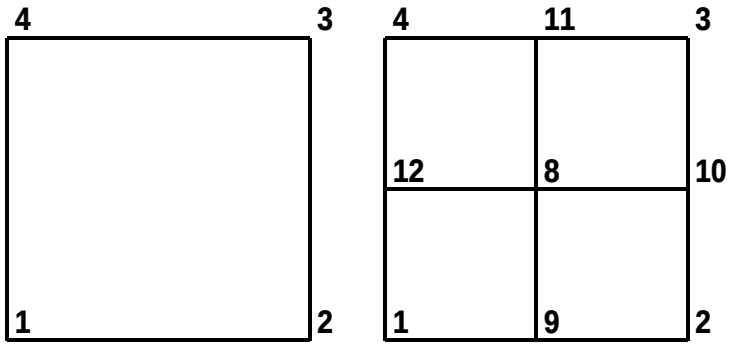


Figure 4.3: Node numbering for new nodes, the original coarse node numbers never change. Only fine node numbers are ever shifted to fit new nodes in.

4.4 Refinement criteria

- 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 ρ 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 $e \in \ _d$ is

$$\rho(p) = \frac{\sum_{=1}^{4} \rho(\ ), X(\ ),}{\sum_{=1}^{4} X(\ ),}; \qquad (4.6)$$

where the '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 varia

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

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

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 ifaces a

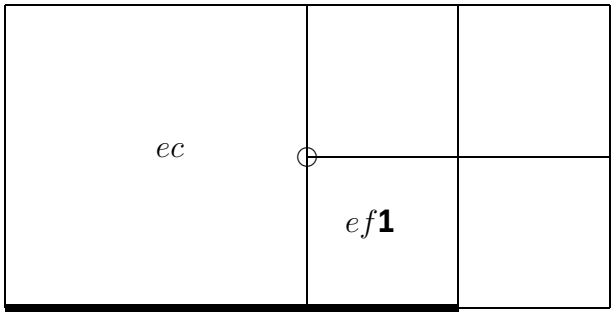


Figure 4.6: Diagram of a course-to-fine interface showing the correct aligning element for limiting along the bottom edge.

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 $L_{/o}$ and L_{e} have the same directions for half of the element as the coarse element had. Since both the area and $L_{/o}$ would have been halved no correction is required. Therefore the velocity gradient can be calculated using the velocity at p

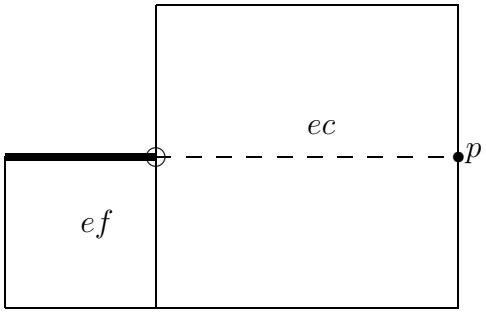


Figure 4.7: Diagram of a fine-to-coarse interface. Limiting along top edge of the fine element requires the continuation of this edge through the neighbouring coarse element to point p.

4.8 Adaptive mesh results

4.8.1 One dimensional test problems

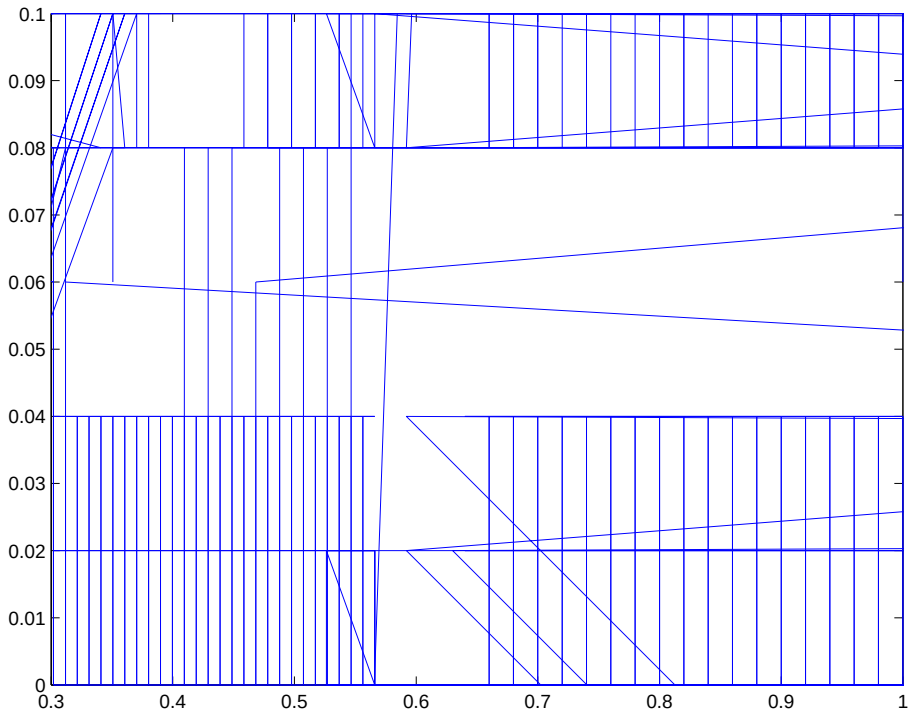
A piston problem with left boundary velocity 1.0, pressure 1.0 and density 1.0 was run with a base grid of 50 $\times$ 5. The refinement parameter was 0.1 and the derefinement parameter was 0.075. This problem verified that both refinement and derefinement worked well. The fine cells formed into a region which followed and encased the shock. Bu□er 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

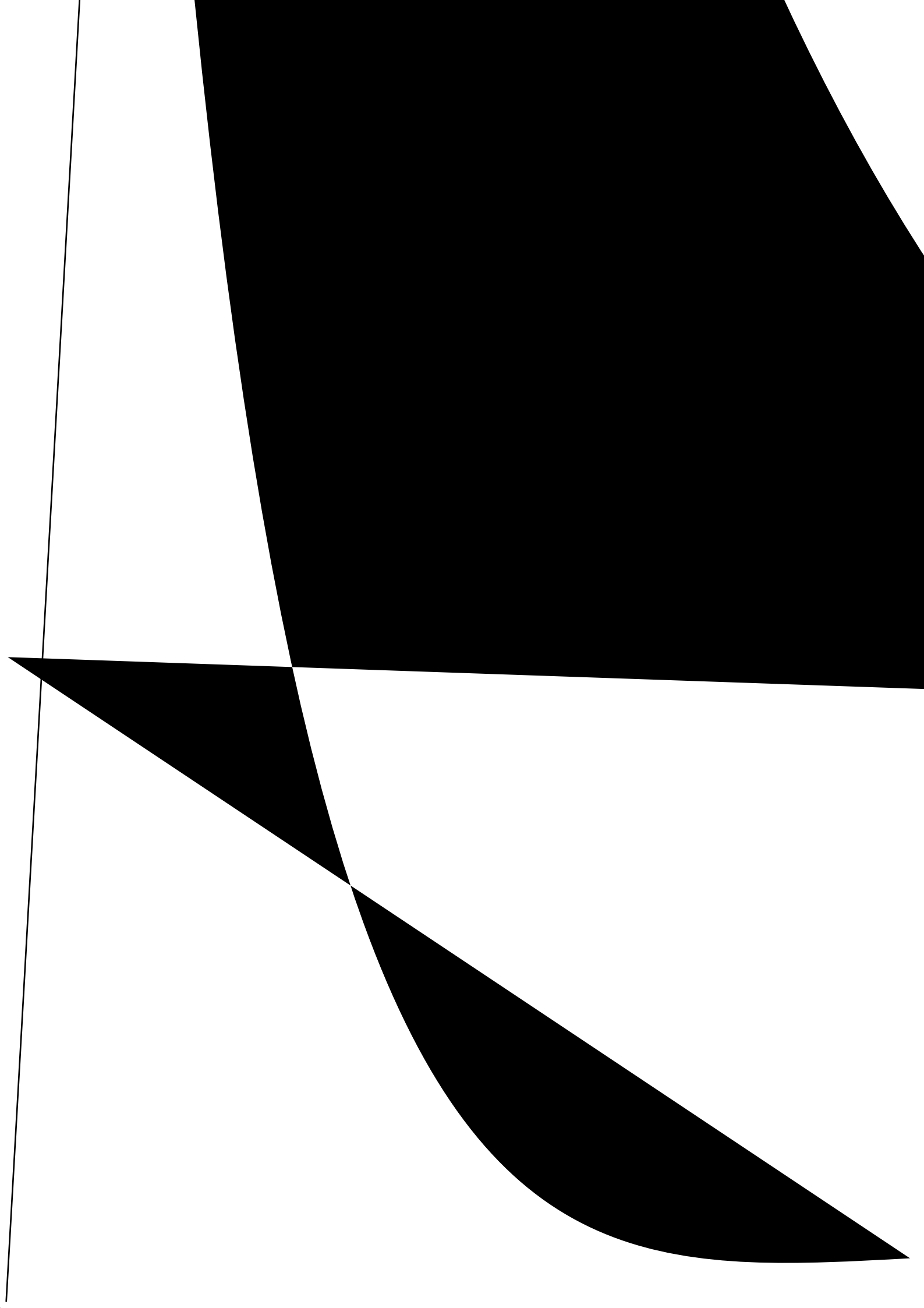
from the fine regions. To remove this problem bu□er cells were introduced. The results then showed one region of fine cells encasing both the rarefaction fan and the contact discontinuity and another region surrounding the shock. The density was used for the refinement criteria because it would vary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

The shock width and relative density error were comparable to a 100 $\times$ 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 $\times$

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.


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



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

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 bu□er 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

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 e□cient, 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 di□erential equations”, Journal of Computational Physics, 53, 484-512, 1984.

[10] M. J. Berger. Adaptive mesh refinement for hyperbolic partial di□erential 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 di erence 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.

7(f)-221.709(C)0.271[26]-(M)0.245657(r)-0.089541(p)-0.312447(p)-0.3