CRANFIELD UNIVERSITY

E.P. Boden

An Adaptive Gridding Technique for Conservation Laws on Complex Domains

COLLEGE OF AERONAUTICS

PhD THESIS

CRANFIELD UNIVERSITY
COLLEGE OF AERONAUTICS

PhD THESIS

Academic year 1996/97

E.P. Boden

An Adaptive Gridding Technique for Conservation Laws on Complex Domains

Supervisor: Prof. E.F. Toro

May 1997

Abstract

Obtaining accurate solutions to flows that involve discontinuous features still remains one of the most difficult tasks in computational fluid dynamics today. Some discontinuous features, such as shear waves and material interfaces, are quite delicate, yet they have a profound effect on the rest of the flow field. The accuracy of the numerical scheme and the quality of the grid discretisation of the flow domain, are both critical when computing multi-dimensional discontinuous solutions. Here, the second order WAF scheme is used in conjuction with an adaptive grid algorithm, which is able to automatically modify the grid in regions of discontinuous features and solid boundaries. The grid algorithm is a combination of two successful approaches, namely Chimera and Cartesian grid Adaptive Mesh Refinement (AMR). The Chimera approach is able to accurately represent non-Cartesian boundaries, whilst the AMR approach yields significant savings in memory storage and CPU time. The combined algorithm has been thoroughly validated for convection test problems in gas dynamics. The computed solutions compare well with other numerical and experimental results. These tests have also been used to assess the efficiency of the grid adaption algorithms. Finally, the approach is applied to axi-symmetric, two-dimensional, two-phase, reactive flows in the context of internal ballistics problems. Again, the computed results are compared with other numerical and experimental results.

Acknowledgements

I would especially like to thank my supervisor, Prof. E.F. Toro, for guiding me in my research. His motivation and enthusiasm has greatly influenced me.

I am extremely grateful for the moral support and encouragement given to me by Dr. Stephen Billett and Dr. Caroline Lowe. I am also indebted to them for their knowledgeable input into the research. I would also like to thank Dr. Bill Speares, Dr. Nikos Nikiforakis and Dr. Simon Leppington for their inputs and ideas.

This research would not have been possible without the financial support of an EPSRC (Engineering and Physical Sciences Research Council) grant and the subsequent funding from the Defence Evaluation and Research Agency (DERA), Fort Halstead, U.K. I would especially like to thank Mr. Clive Woodley (DERA Research Monitor) for not only giving me the opportunity to finish the development of the computer code, but also for his valuable input.

I am grateful to Prof. J.F. Clarke and Dr. N. Qin for assuming supervisory roles during the periods when Prof. E.F. Toro was absent from Cranfield.

Finally, I would like to thank my family and friends for their love and support.

Contents

List of Figures

Notation

$\mathbf{A}$	Jacobean of a flux function
A	coefficient in a numerical scheme
$\mathrm{A}_{i,j}$	area of cell (i,j)
a	wave speed in the x direction for the linear advection equation *or* sound speed in gas dynamics
b	wave speed in the y direction for the linear advection equation *or* covolume in gas dynamics
E	total energy per unit volume
e	specific internal energy
$\mathbf{F}$	vector flux function in the x direction
$\mathbf{F}_{i+\frac{1}{2}}$, $\mathbf{F}_{i+\frac{1}{2},j}$	intercell flux in the x direction for systems of equations
$\mathbf{G}$	vector flux function in the y direction
$\mathbf{G}_{i,j+\frac{1}{2}}$,	intercell flux in the y direction for systems of equations
G_L	grid level
G_{Lmax}	maximum number of grid levels
i	cell index (x-direction for Cartesian grids)
j	cell index (y-direction for Cartesian grids)
k	index for waves in the solution a Riemann problem
L	operator for a one dimensional scheme
$l_{i+\frac{1}{2},j}$, $l_{i,j+\frac{1}{2}}$	cell side lengths
$l^{(k)}_{i+\frac{1}{2},j}$, $l^{(k)}_{i+\frac{1}{2},j}$, $l^{(k)}_{i,j+\frac{1}{2}}$, $l^{(k)}_{i,j+\frac{1}{2}}$	integration lengths for the k^{th} wave
m	number of cells in the discretisation
m_{ig}	total igniter mass
M_s	shock Mach number
N	number of equations in a system/waves in the solution of a Riemann problem
$\mathbf{n}$	unit normal vector
n	time level
P	point
p	pressure
q	some physical quantity
r	flow parameter *or* radius
S	wave velocity
$\mathbf{S}$	Source term vector
t	time
t_1, t_2	limits on time integral
t_{ig}	igniter venting time
$\mathbf{U}$	vector of conserved variables

$\widetilde{\mathbf{U}}$	solution to a Riemann problem
u	velocity in the x direction
	or conserved variable in the linear advection equation
V	grid velocity
V_{ig}	volume into which the igniter is injected
v	velocity in the y direction
W	weight in WAF intercell flux
W_g	molecular weight of gas
x	coordinate direction
y	coordinate direction
z	coordinate direction

Greek alphabet

α	pressure index for Piobert's burning law
β	burning rate coefficient for Piobert's burning law
δ	integration length
Δx, Δy	mesh spacings for Cartesian grids
Δt	time-step
Γ	refinement factor
γ	ratio of specific heats
ζ	coordinate direction corresponding to the i indices
η	coordinate direction corresponding to the j indices
λ	characteristic speed
μ	relative wave Courant number
ν	Courant number
ρ	density
σ	standard deviation
	or signature of flagged cells in a strip of cells
ϕ	limiter function
	or Gaussian distribution
ω	wave speed

Subscripts

atm	atmospheric
g	gas
ig	igniter

i, j	in grid cell (i, j)
$i + \frac{1}{2}$	at boundary $i + \frac{1}{2}$
$j + \frac{1}{2}$	at boundary $j + \frac{1}{2}$
L	left
$*L$	left star state
loc	at the current local boundary
M	middle
p	particle *or* propellant
R	right
$*R$	right star state
upw	at the nearest upwind boundary
v	vortex
G	Gaussian
x	in the x direction
	or x derivative
y	in the y direction
	or y derivative

Superscripts

(k)	k^{th} characteristic field
n	time level
(Δt)	time step for operators

Acronyms

ACR	Adaptive Cell Refinement
ANP	Adaptive Node Placement
AMR	Adaptive Mesh Refinement
CAMR	Chimera-Adaptive Mesh Refinement
CIR	Courant, Isaacson, Rees
CFD	Computation Fluid Dynamics
CMR	Complex Mach Reflection
CPU	Central Processing Unit
DMR	Double Mach Reflection
ENO	Essentially Non-Oscillatory
FCT	Flux Corrected Transport
FVS	Flux Vector Splitting

GRP	Generalised Riemann Problem
HLL	Harten, Lax, van Leer Riemann solver
HLLC	modified HLL Riemann solver which accounts for Contact waves
HPF	High Performance Fortran
MUSCL	Monotone Upstream Centred Schemes for Conservation Laws
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PLM	Piece-wise Linear Method
PPM	Piece-wise Parabolic Method
RAM	Random Access Memory
RR	Regular Reflection
SMR	Single Mach Reflection
TVD	Total Variation Diminishing
WAF	Weighted Average Flux

Word Definitions

Cell	A quadrilateral element used to discretise the domain and evaluate the solution.
Chimera *or* Chimaera	Name given to the gridding technique by Benek *et al.* [6]. Animal from Greek mythology compounded of incongruous parts.
Grid *or* Grid level	One or more mesh patches of equivalent cell resolution.
Mesh *or* patch	Rectangular (in computational space) group of cells of equivalent cell resolution.

Chapter 1

Introduction

1.1 Motivation

Over the past three decades there have been some dramatic improvements in Computational Fluid Dynamics, CFD, results. However, CFD techniques are being applied to an increasing number of diverse problems, the results of which are not always an accurate reflection of the true physics. There are several reasons why a given set of CFD results will not compare well with those of the equivalent physical experiment. Assuming the results provided by the experimental model are themselves accurate, the discrepancies are usually caused by one or more of the following: the lack of physical knowledge about the chosen problem, the inappropriate choice of the governing equations, the poor solution quality of the numerical scheme, the inappropriate use of one and two-dimensional models to represent two and three-dimensional phenomena, and the limitations of the computer hardware. Experimental results are extremely valuable for assessing numerical results. Thus, the ongoing development of numerical methods is aided by the improvements in experimental techniques, such as fibre optics, heat probes, strain gauges, laser imaging, etc. Numerical computations have the advantage that they are able to provide answers where accurate experimental results are difficult to obtain. Furthermore, once a computer model has been developed, a wide range of test problems can be simulated relatively cheaply. Experimental results, unlike computer simulations, are often affected by scaling the true physical situation. Moreover, experimental monitoring techniques, such as the inclusion of pressure gauges and heat probes in the flow field, can affect the solution and thereby lead to inaccurate results. The results of new research must always be regarded with some skepticism. However, many of the numerical techniques in CFD have been widely tested and validated, by comparing the numerical solutions with both good experimental results and analytical solutions.

The mathematical models which represent discontinuous flows are usually sys-

tems of several non-linear coupled partial differential equations (PDE's). With the exception of some simple problems, systems of this nature cannot be solved analytically. There is a class of non-linear hyperbolic systems of PDE's that are used to describe time-dependent fluid problems. The hyperbolicity of the equations combined with the non-linearity, not only permit, but may also generate discontinuous features from initially smooth data.

Computing multi-dimensional solutions to flows that exhibit discontinuous features (e.g. shock waves, material interfaces etc.), along with other associated physical phenomena, (e.g. Kelvin-Helmholtz instabilities), is a major area of research. Discontinuous solutions pose stringent demands on the numerical methods used to compute them. The main difficulties in successfully modelling discontinuous features are caused by numerical errors, which are usually a combination of both dispersion and diffusion. Dispersion errors cause unphysical oscillations to occur in the region of discontinuities. Diffusive errors tend to broaden features such as contact waves and slip surfaces to unrealistic extents. Note that the compressive characteristic nature of shocks has a tendency to limit the numerical diffusion [116].

It is difficult to represent the location and the severity (gradient) of discontinuous data, much better than the resolution of the cell discretisation, upon which the data is computed. Good numerical schemes are able to contain shock waves within two or three grid cells for many time steps. Increasing the order of accuracy of the scheme has little effect[1]; the shocks are still contained within two or three grid cells. Reconstructing discontinuities from cell averaged data also has a limited effect. Hence, for discontinuous solutions, there is a limit beyond which attempts to improve the accuracy of the numerical scheme are futile. Thus, if increased resolution of isolated discontinuities is required, beyond the capabilities of the numerical scheme, then it can only be achieved by decreasing the size of the computational grid cells. As a result, the number of cells in a typical regular grid computation will increase, which may lead to a prohibitive amount of memory storage and unrealistically long processing times. A lot of research has been focussed upon adaptively refining the computational grid in regions where the numerical error is high, in order to improve the accuracy of the representation, without incurring large computing costs. There are problems, for which current computing technology cannot provide solutions within realistic time frames, without employing some form of grid adaption. A very good argument, involving the burning of rate sticks of high energy solids, as to why grid adaption is necessary, is given in [23].

The computational grid has a direct effect upon the quality of the computed solution. When anything other than a Cartesian computational grid is employed, high quality numerical schemes introduce extra errors into the solution. Note that, non-Cartesian domains can be computed using Cartesian computational grids, by

[1]Higher order schemes do have a very noticeable effect for smooth solutions.

transforming the mathematical models, e.g. for a circular domain the model can be transformed to polar coordinates. The more the computational grid lines deviate from the orthogonal, the greater the potential error. Many problems that are of interest to scientists and engineers involve domains, that cannot be accurately represented by a Cartesian grid. The representation of dynamic boundaries is an even greater source of difficulty.

The aim of this work is to develop an adaptive gridding technique, that is capable of computing time dependent discontinuous solutions, in and around dynamic arbitrary geometries. The application motivating this research is a two-phase (gas and solid) internal ballistics problem. The problem involves a typical gun configuration in which the geometry of the flow domain changes as the projectile moves along the barrel. Movement of the projectile is caused by high pressure behind the projectile, which is generated by the combustion of the solid propellant. Modelling multi-phase internal ballistics problems poses some challenging numerical difficulties. Numerical schemes must be employed that can accurately model discontinuous features, such as shock waves and material interfaces. The numerical method must also account for the rapid motion of moving boundaries. The mathematical nature of the model and the numerical difficulties that arise, result in successful numerical methods being quite computationally expensive. A good grid adaption strategy should be able to reduce the computing costs and so enable the use of more detailed models.

1.2 Background

The modelling of fluid dynamics problems involves the mathematical representation (the PDE's, equation of state, etc.) of the physics and the numerical methods for computing the subsequent solutions. The work presented here, concentrates on the development of the numerical techniques for solving some well known problems. There are two aspects to a numerical method; the numerical scheme used to compute the solution and the computational grid upon which the solution is represented and solved. Numerical methods have evolved over many decades, resulting in a wide variety of different strategies. No single numerical method has been adequately proven to be superior to all others for solving discontinuous flows. There have been many reviews and comparisons of the various numerical strategies, accounts of which are given in references [88, 103, 132] and [134] to name just a few. In some ways, these comparisons are vital in order to advance scientific and engineering knowledge. However, if any one method achieved superiority over others, then too much attention might focus on it alone. This would almost certainly have a detrimental effect, since the diversity of knowledge gained from other techniques would be curtailed. A review of various numerical schemes, that the author considers to be most suitable for solving non-linear hyperbolic PDE's based on conservation

laws, is given in the following subsection. This is followed by a review of various suitable gridding techniques.

1.2.1 Numerical Schemes for Discontinuous Flows

The mathematical models that describe many different fluid dynamics phenomena, such as the Shallow Water, Euler and Navier-Stokes equations, have been around for a hundred years or more. For a long time, all but the simplest of problems were impossible to solve. Eventually, with the advent of computing machines, the solutions to some basic problems involving smooth flows became calculable. However, accurate solutions to fluid problems involving discontinuous features were still unattainable. It was a common belief that discontinuous flows could only be modelled, by specifically tracking the position of the discontinuities, in order to calculate the correct relationships (*Rankine-Hugoniot*) across them. Computations of this type had limited success, especially for situations involving the interaction of two or more discontinuous features, for which accurate jump conditions are difficult to apply. In 1960 Lax & Wendroff [61] reported that in the limit, a converged numerical solution, computed with a conservative scheme, yields the correct conditions across a discontinuity. There are a variety of finite difference and finite volume conservative schemes that compute the same expressions throughout the flow field. These schemes are commonly referred to as *shock capturing* schemes so as to differentiate them from *shock fitting* or *front tracking* schemes.

Numerical schemes can be split into two classes; implicit and explicit schemes. Unlike implicit schemes, explicit schemes express the solution at the next time level only in terms of the previous time level solutions. The time step for explicit schemes is restricted by the stability criteria, which varies from scheme to scheme. Thus, even though explicit updates are generally inexpensive, they must be applied frequently. Generally, implicit schemes have much larger time steps than their explicit counterparts. However, for transient problems, the time steps must also be restricted to some extent, in order to ensure temporal accuracy. For implicit schemes, the solution at every time step, is calculated by solving a system of simultaneous algebraic equations. So as to reduce as much as possible the memory requirements, these equations are rearranged to form banded (often tridiagonal) matrices. However, even with highly developed pre-conditioning techniques, the solutions to these systems of equations can be expensive in terms of both the processing time and the memory storage. Compared to explicit schemes, implicit schemes are able to attain steady state solutions relatively quickly. However, for transient problems, the time step restriction required to achieve good time accuracy may make implicit schemes overly expensive. For such problems, accurate solutions can be obtained efficiently with explicit schemes. Although, some equation models can often be solved most efficiently by using a combination of both implicit and explicit schemes. For example,

often the advection terms in the Navier-Stokes equations are solved by an explicit scheme, while the diffusion terms, which incur prohibitive stability conditions, are solved by an implicit scheme. The nature of the problems that are presented in this work, involve highly transient flows. Also some of the problems involve boundaries in relative motion, which directly affects the solution. Hence, the time accuracy of the numerical scheme is of the utmost importance. Therefore, only explicit schemes shall be considered here.

Too much numerical diffusion associated with first order explicit schemes, renders them virtually impractical for solving general fluid dynamics problems. They not only diffuse discontinuous features, such as contact and shear waves to unacceptable levels, but they are also unable to maintain the representation of smooth features over even a moderate number of time steps. First order schemes have been superseded by a whole plethora of higher order (second and above) schemes. However, even though a high order scheme may be linearly stable, it will produce erroneous solutions in the vicinity of discontinuities, in the form of unphysical oscillations. The oscillations can result in non-linear instabilities, which can cause the failure of the computation, e.g. a negative pressure. Godunov produced the theorem that all constant coefficient shock capturing schemes, with greater than first order accuracy, will admit unphysical spurious oscillations [41].

Von Neumann *et al.* [130] proposed that unphysical oscillations could be 'damped out' by adding viscous terms to the PDE's, thereby artificially increasing the viscosity of the scheme. Provided sufficient diffusion is added, discontinuous profiles appear to be free of any spurious oscillations. Schemes based on this technique, are commonly known as *artificial viscosity* schemes. They have been extended to finite volume schemes and applied to various problems for both two and three-dimensional flow domains. (Refer to work by Richtmyer & Morton [83], Jameson [58] and Rizzi & Eriksson [84].) Artificial viscosity methods are very easy to implement and are relatively inexpensive computationally. However, the parameters which govern the amount of diffusion added, need to be adjusted or 'tuned'. This is somewhat alleviated by making the parameters dependent on the local flow conditions [63], so that the amount of added diffusion varies within the problem, (e.g. greater in regions of high density gradient). For situations where two or more transient discontinuities are in close proximity, it is still difficult to adequately tune the scheme.

Since the beginning of the 1970's several different methods have been presented that, combine first and second order schemes in such a way as to reap the advantages of each. These methods attempt to employ the second order scheme in regions of smooth flow and the first order scheme in regions of discontinuities, with a blend between the two. Hereinafter, it should be taken as read that the order of accuracy of a scheme only applies to smooth solutions. Boris & Book [17] developed such a scheme that updates the solution with a mixed order scheme. A mixed order flux is calculated by combining first and second order fluxes in such a way as to minimise

spurious oscillations. Schemes of this type, commonly referred to as Flux Corrected Transport (FCT) schemes, have been presented by Harten & Zwas [53], Zalesak [135] and Löhner [66]. When applied to non-linear systems, FCT schemes can result in a 'stair-casing' effect in smooth regions of the flow [132].

The concept of Total Variation Diminishing TVD schemes was put forward by Harten [47]. Schemes of this type ensure that the total variation of the solution at any particular time level, is less than or equal to that at the previous time level. This is achieved by limiting the accuracy of the scheme, so that it varies with the local solution data. Provided the initial data is monotonic, TVD schemes will preserve the monotonicity for all scalar conservation laws. Even though the theory is only valid for scalar conservation laws, it has been successfully extended to non-linear systems of equations without any significant drawbacks. Many different schemes, including older ones that have been extended (e.g. FCT), now incorporate TVD limiting. TVD schemes are able to produce very accurate solutions to discontinuous flows. Moreover, they function automatically, in that they do not require 'tuning' or any prior knowledge of the problem.

In 1952, Courant, Isaacson & Rees [28] presented the idea of an *upwind* scheme. Their scheme, which has become known as the CIR scheme, effectively changes its stencil, based on the characteristic paths at the current time level, such that it is biased towards the data from the upstream direction of the flow. This approach, when compared to centred schemes, generally benefits from improved stability, and thus affords larger time steps to be taken. The CIR scheme is the most accurate first order scheme, in that it has the smallest leading coefficient in the truncation error analysis [106]. However, the main disadvantage of the CIR scheme is that it is not conservative, and is therefore unable to correctly predict the speeds of shock waves. Nowadays, most good shock capturing schemes employ upwinding. Schemes that are referred to as *flux vector splitting* (FVS) schemes, incorporate upwind information by splitting the numerical flux into a forwards component $\mathbf{F}^+$ and backwards component $\mathbf{F}^-$, based on the direction of the characteristics, i.e. if $\mathbf{U}_i^n$ is the solution in cell i at time level n, then the intercell flux $\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}^+(\mathbf{U}_i^n) + \mathbf{F}^-(\mathbf{U}_{i+1}^n)$. The first full mathematical description of FVS was given by Steger & Warming [98]. An improved FVS scheme was later presented by van Leer [127].

A milestone for modern shock capturing schemes came in 1959, when Godunov proposed using the solution to the Riemann problem in schemes for computing hyperbolic systems of conservation laws [41, 42]. The Riemann problem is an initial value problem for a system of PDE's. The initial conditions involve a single discontinuity between two sets of constant data. Upwind information is directly accessible from the wave structure in the Riemann problem solution. Often there is not a closed form solution to Riemann problems for non-linear systems. Thus, exact solvers often need to iterate to find the solution and may therefore be expensive. There exists a number of approximate Riemann solvers with a range of performance

characteristics; (refer to [73, 85, 115] and [119]). Toro [112] used an exact and a very cheap approximate Riemann solver adaptively to solve one and two-dimensional Euler equations. More than 99% of the Riemann problems were solved using the approximate solver, without affecting the accuracy of the computed solution. The employment of the Riemann problem enabled Godunov to extend the CIR scheme to non-linear systems [42]. Note that, for hyperbolic systems of linear equations with constant coefficients, Godunov's scheme is identical to the CIR scheme. Godunov's scheme solves Riemann problems at the interfaces of the discretised solution at time level n. Before the waves emanating from any two neighbouring Riemann problems interact, it calculates the cell solutions at the next time level, $n + 1$, by taking integral averages of the evolved Riemann problem solutions. Godunov's scheme is a conservative, first order accurate, monotone scheme and is the basis of several different higher order schemes for non-linear hyperbolic systems.

The first in a class of schemes known as MUSCL (an acronym for Monotonic Upstream-centred Schemes for Conservation Laws) was introduced by van Leer [125] and [126]. Generally, MUSCL schemes achieve second order accuracy by constructing piece-wise linear data from the discretised piece-wise constant data. TVD is incorporated by limiting the slopes of the reconstructed data. Other second order schemes in the MUSCL category include the Generalised Riemann Problem (GRP) scheme by Ben-Artzi and Falcovitz [4], the Piece-wise Linear Method (PLM) by Colella [24] and the scheme by Hancock (MUSCL-Hancock) [128]. Colella & Woodward [26] replaced the linear reconstruction in the PLM scheme with a parabolic one in order to obtain the third order accurate Piece-wise Parabolic Method (PPM). Harten *et al.* [50] and [52] used even higher order reconstructions in their schemes. MUSCL schemes extrapolate the reconstructed data in neighbouring cells to the intercell boundary. They differ in the way that the extrapolated values are used to obtain the numerical flux. For example, the MUSCL-Hancock scheme first evolves the extrapolated values by half a time step before using them to form a piece-wise constant data Riemann problem. The intercell boundary flux is then obtained from the Riemann problem solution along the t-axis. Whereas, the GRP scheme computes the numerical flux from a Taylor series expansion of the extrapolated data Riemann problem solution.

Schemes of the MUSCL type are often referred to as *slope limiter* schemes. This distinguishes them from another category of schemes that introduce limiting within the flux calculation and are known as *flux limiter* schemes. (The FCT scheme is a type of *flux limiter* scheme.) Sweby [101], Harten [46], Roe [86] and Toro [109, 114] have all introduced high order, TVD, *flux limiter* schemes based on Godunov's scheme. Toro's scheme, called the Weighted Average Flux (WAF) scheme, has its roots in an earlier scheme of Toro's, known as the Random Flux scheme [107]. The Random Flux scheme, which has been statistically proven to be second order accurate [118], first computes the Riemann problem at the intercell boundary between each pair of piece-wise constant data. The numerical flux is then taken as a random sample of

the flux across the wave structure of the Riemann solution, at the mid-point time level. The WAF scheme improves on this by integrating the flux across the structure of the Riemann solution and then dividing by the distance to get an average flux. In practice, the integration is a simple weighting for each state of the Riemann problem solution, according to the wave speeds. The WAF scheme is a second order scheme, that is made TVD by calculating a limiting function for every wave in the Riemann problem solution, that is based on the ratio of the upwind to the local data jumps across each wave. By basing the limiting on the jumps across each wave rather than across neighbouring cells, the WAF scheme is able to obtain high resolution monotone solutions. Results computed with the WAF scheme tend to yield better resolution of delicate discontinuous features, such as contact waves, than the equivalent order MUSCL schemes with standard slope limiting [57].

One-dimensional numerical schemes can easily be extended to multi-dimensions using a technique known as *space operator* or *dimensional splitting*; originally called the method of *fractional steps* [133]. This involves solving all the one-dimensional strips of cells, in all the computational dimensions, sequentially. Other improved operator splitting techniques are described by Strang [100]. Even though the TVD theory is not valid for split schemes, multi-dimensional solutions have been successfully computed using basic splitting techniques, e.g. by Woodward & Colella [132]. The development of unsplit, genuine *finite volume* schemes has further improved the quality of multi-dimensional solutions. However, Goodman & LeVeque [43] showed that for non-trivial cases, schemes that impose TVD constraints in a multi-dimensional sense, are at most first order accurate. Very effective shock capturing Riemann based unsplit schemes for multi-dimensions have been presented by Colella [25], Billett [15] and Saltzman [89]. In order to achieve second order accuracy, none of these schemes attempt to apply strict TVD limiting in all the dimensions. As a result, they are not strictly oscillation free. Schemes, known as *Essentially Non-Oscillatory* (ENO) schemes, that do permit small oscillations have been developed by Harten, Engquist, Osher, and Chakravarthy, ([48, 50] and [49]). ENO schemes selectively choose the reconstruction or interpolation stencil in order to minimise, but not remove, the oscillations. Unfortunately, because of the inherent one-dimensionality of the Riemann problem, all multi-dimensional schemes that utilise its solution are far from ideal. For this reason, many CFD codes still employ high order split schemes. Presently, truly multi-dimensional Riemann problem solvers do not exist. A great deal of interest is being focussed on schemes which take account of the direction of primary wave propagation. Schemes of this type, referred to as *Multi-Dimensional Upwind* (MDU) schemes [129], are new and relatively unproven.

When solving the systems of PDE's that model physical problems, the equations are often split into one or more homogeneous hyperbolic parts with some extra source terms. Such systems are usually solved by *time operator splitting*, whereby the solutions are updated by the homogeneous parts and source terms separately. The

homogeneous parts are solved using the numerical schemes described above, whilst a simple ODE solver, such as a two stage modified Euler solver, is often sufficient to solve for the source terms. However, if the time scales associated with the source terms are smaller than those associated with the convection of the flow, then numerical instabilities may result. Situations involving chemically reactive flows often involve very disparate time scales, which require 'stiff' solvers to solve for the source terms [64].

This section is by no means a comprehensive account of all the numerical schemes that have been used to solve discontinuous flows. The reader is referred to other techniques such as, spectral methods, finite element and front tracking schemes. The books by Hirsch [55], LeVeque [63] and Toro [116] are good sources of information about many different schemes for computing discontinuous solutions.

1.2.2 Computational Grid Techniques

The ability to model fluid interactions with geometrically complex solid boundaries, is a major stepping stone in the construction of a general research or design tool. Multi-dimensional CFD calculations generally require the spatial domain of the solution to be discretised. The discretised domain is commonly referred to as the computational grid. Not only does the storage of the solution data relate to the computational grid, but its evolution is directly affected by the geometry of the grid. Whilst the recent advances in both computer software and hardware have yielded successful numerical solutions to many two and three dimensional problems, the issues involved in how best to discretise complex flow fields and compute their solutions is still a matter of debate.

Only very simple domains can be represented by one or more Cartesian grids. Curvilinear grids, which can be regarded as Cartesian grids with the cell vertices distorted, can often be used to discretise reasonably complex (flow) domains, without too much difficulty. Curvilinear grids, like Cartesian grids, are *regular* grids, in that the locations of the cells in physical space is reflected in the computational space. Hence, individual grid vertices and cells can be directly referenced from the stored positions in a suitably dimensioned array. Given the boundary information and some other control functions, smooth curvilinear grids can be generated algebraically [92] by interpolating the coordinates of the cell vertices. Algebraic grid generation is a very fast technique that is often used for fixed grid problems. Generating algebraic curvilinear grids for domains involving complex geometries, can be a difficult task. Fairly complex domains can be discretised by solving appropriate elliptic PDE's [32]. The resulting grids are smooth and can be easily adjusted, such that the cell resolution can be varied by slightly changing the form of the PDE. Parabolic and hyperbolic PDE's have also been successfully used, although they

are not always suitable for situations involving grid discontinuities because of their tendency to produce non-smooth variations in the surrounding grid cells [131]. In two dimensions, curvilinear grids have also been generated using conformal mapping techniques [56]. However, mapping techniques cannot provide the same degree of control over the grid resolution as elliptic techniques.

Whilst structured grids have proved to be very successful in producing solutions to a wide variety of problems, they do have their drawbacks. Firstly, in order to compute solutions on such grids, the numerical scheme has to be modified so as to take account of the geometric variations. The modification requires extra geometrical data to be stored[2] and/or extra calculations (rotations, side-length calculations etc.) during every cell integration to be made. Secondly, numerical schemes for conservation laws, have a tendency to relinquish some of their accuracy when applied to anything other than a Cartesian computational grid. Hence, the computed solution is affected by the quality of the computational grid. The errors that are incurred are not too evident provided the grids are smooth, the grid cells are roughly the same size and are not too skewed. Even with multi-block techniques, in which two or more curvilinear meshes are linked together, it is not always possible to generate good quality structured grids to fit complex geometries involving multiple bodies.

An alternative approach, is to use *unstructured* grids, which are able to discretise even the most complex flow domains relatively easily. Unstructured grids have traditionally been used with finite element schemes. The two most common techniques for generating unstructured grids are the *Advancing Front* technique, developed by George [38] and Peraire [76] for two and three dimensions respectively, and the Delaunay triangulation technique [131]. Compared to structured grids, unstructured grids require extra ancillary information to be stored in order to locate the data associated with any single cell and its neighbours. Unstructured grids lack orthogonality at boundaries, which make accurate boundary conditions difficult to apply. Furthermore, they are even worse choices for computing transient discontinuous features, especially strong shock waves and thin shear waves, and are not best suited for computing boundary layer solutions.

The book by Thompson *et al.* [104] and the lecture series notes by Weatherill [131] provide more detailed information about various techniques for generating both structured and unstructured computational grids.

Flow domains with arbitrary geometries have been modelled using so called Cartesian cut cell techniques. The cells that are 'cut' by the fluid-solid boundary interfaces are updated in a special way. Cartesian cut cell techniques require no extra storage regarding the grid geometry, other than the small amounts for the cut cells. Thus, the memory requirements and the processing times are kept to a mini-

[2] A minimum of every vertex coordinate, but often to improve the CPU time efficiency the side lengths and cell areas are also stored.

mum. Other than the numerical errors associated with the scheme, the integration of the cut cell solutions is the only other source of error. As with non-Cartesian grid cells the size of the truncation error is dependent on the geometric variations (side lengths and intercell boundary orientations) in the grid. Also, the quality of the boundary representation is proportional to the resolution of the grid cells. Hence, these techniques have benefited from adaptive grid refinement techniques, which are discussed later in this section. Cartesian techniques create a number of small cut cells, which can potentially cause numerical stability problems for time dependent computations. Clarke *et al.* [22] absorbed a small cell into a neighbouring cell if its area had been reduced by a specified fraction as a result of a cut. This has become a common technique for circumventing any stability problems [36, 80], but the increased cell size reduces the (local solution) accuracy. LeVeque [62] ensured stability in the small cells by applying a large time step scheme to the boundary cells. Berger & LeVeque having already coupled this technique with grid adaption, later improved the technique by removing the large time step scheme and instead solved for interpolated data normal to the boundary cuts [11]. Both of the techniques presented were second order accurate within the flow field, but were less than second order accurate at the boundaries, although they suggested that the latter technique could be extended to second order accuracy.

Other approaches have been developed that combine two different techniques in such a way as to maximise the benefits and minimise the drawbacks of each. These approaches are often referred to as *Chimera*[3] approaches. The Chimera approach over-sets a background mesh with curvilinear boundary-fitted meshes. Usually, the boundary data for the over-set meshes is interpolated from the underlying meshes and the covered cell data of the underlying meshes is interpolated from the over-set meshes [2, 6, 19]. Although various non-conservative interpolation schemes have been used successfully, concerns about the effects of interpolating discontinuous data, have prompted the development of conservative interpolation schemes [8, 20]. Instead of interpolating the data for the overlying and underlying cells, Nakahashi and Obahashi [70] linked the meshes with slender regions of unstructured grid cells. Both techniques benefit from the high quality solutions obtained from the background meshes (usually a Cartesian mesh) and the very smooth boundary-fitted meshes. The drawbacks of both techniques are the connectivity between the different parts of the data storage and the errors that can occur at the interfaces between any two meshes.

The representation of both the solution and the boundaries of the flow domain can be improved by refining the computational grid. However, complete refinement of the grid is often far too expensive in both data storage and processing time to provide realistic solutions quickly. Adaptively refining the computational grid, such that the cell resolution varies according to the solution error, brings obvious

[3] An animal from Greek mythology compounded of incongruous parts.

advantages. Any efficient grid adaption algorithm, (i.e. minimal in comparison to the integration of the solution), when applied to problems that require high cell resolution in only a fraction of the flow field, should be capable of providing solutions more quickly and with less data storage than regular grids of comparable grid resolution. Good adaption algorithms, should also be able to produce solutions that have no significant differences to the equivalent regular grid solutions.

One problem with trying to assess the solution quality produced by one adaptive grid technique compared to another, is that low resolution solutions are rarely published. Why not? The answer lies in the fact that all adaptive gridding techniques, compare more favourably with the equivalent constant resolution grid as the resolution is increased. By way of an example, suppose a discontinuous feature, such as a shock wave, is represented by three grid cells. If the resolution were to be increased, the feature would probably still be represented by three cells, but the extent of the three cells would take up a smaller fraction of the whole flow domain. Hence, the ratio of the total number of fine cells in the adaptive grid to the number in the equivalent constant resolution grid, decreases. As a result the storage savings and the reduction in processing times for the adaptive technique appear more favourable. Hence, the lack of low resolution solutions is not surprising; none is presented here either. A fuller discussion about the relative performance of grid adaption is given in section 5.3.2.

The fundamental ideas underpinning the most established adaptive grid refinement techniques are considered here. Since the work in this thesis is motivated by unsteady problems, only those techniques that are able to dynamically change the grid structure throughout a computation are relevant. (Obviously, fixed grid refinement is only suitable for problems in which prior knowledge of the solutions are available.) There are two opposing schools of thought for adaptive grid refinement; those with fixed costs, which produce grids that have varying peak resolution and those with varying costs, which produce grids that have fixed peak resolution.

Fixed cost, Adaptive Node Placement (ANP) methods, often referred to as penalty methods, move the nodes (cell vertices) of a structured grid in such a way as to cluster them in regions of special interest, whilst still maintaining the structured nature of the grid. The result is a curvilinear grid, in which the number of cells in each computational direction remains unchanged throughout the computation. Hence, ANP techniques have the same advantages and disadvantages as regular curvilinear grids, (i.e. extra geometric data needs to be stored but the data structures are simple and easily vectorised). When adapting to several features, it is very difficult to prevent the resolution, in any area of the grid, being diminished to unsatisfactory amounts. For complex groupings of features care must be taken in order to prevent the grid cells from becoming entangled. Even when all these difficulties have been avoided the resulting grid cells may be badly distorted, which in turn adversely affects the solution quality. Another drawback of penalty methods is the fact that the solution

quality varies according to the amount of refinement, because the number of grid cells is fixed throughout a computation.

The second adaptive gridding school of thought can be split further into unstructured and structured grid refinement. Unstructured adaptive grid techniques [65] are usually, but not always, based on domain decomposition. Once a suitable domain decomposition algorithm has been set up, only minor changes to the data structure and the inclusion of the refinement criteria are required. The drawbacks of unstructured grids, whether they are refined or not, have been highlighted earlier in this section, (i.e. lower solution quality, greater storage requirements and increased processing times). Other than ANP, structured grid refinement is either based on Adaptive Mesh Refinement (AMR) or Adaptive Cell Refinement (ACR)[4]. Both of these approaches benefit from the structured nature of the grid, in that they can be easily applied to vector machines and the numerical scheme can easily be separated from the complicated workings of the rest of the code.

Berger has taken the AMR approach from the conceptual stage to the point where it is able to efficiently compute accurate solutions to unsteady, multi-dimensional, discontinuous flows. Her early algorithms [7, 12] did not constrain the refined grids to the coordinate directions of the underlying coarse grids. Most current AMR algorithms are variations on the one by Berger & Colella [9], which employs a nested hierarchy of Cartesian mesh patches. In order to prevent the stability restrictions of the finer grids from restricting the progress of the coarser grids' solutions, AMR involves temporal as well as spatial refinement. Also, by advancing the solution with time steps that are appropriate to the grid cell size, less numerical diffusion is introduced. Unlike all other refinement techniques, the memory storage is on a per mesh basis rather than per cell. Thus, the amount of memory storage that is required for any given problem, is less than that for other adaptive techniques. The biggest drawback of AMR is the complexity of the algorithm and source code. However, this has not deterred several workers from publishing new algorithms. Quirk [79] demonstrated the ability of his algorithm to compute high resolution solutions on a standard work station[5]. His algorithm has been applied to unsteady detonation waves in high energy solids [23] and has been optimised for a network of parallel processors [82]. AMR has been used to compute a variety of different problems, including atmospheric flows [90], reactive flows [23], viscous flows [33, 79] and wave propagation in non-linear solids [120]. Berger & LeVeque [10] were the first to develop a Cartesian AMR cut cell technique for arbitrary geometries. Quirk [80, 81] has also demonstrated a very robust Cartesian-AMR approach, but like Berger's & LeVeques, it is only first order accurate at the boundaries. A Cartesian-AMR approach has been extended to three dimensions [75], but again the numerical scheme reduces to first order accuracy at the boundaries.

---

[4] Unstructured adaptive grid techniques are usually a form of ACR .

[5] SPARC 1; a slow work station by todays standards.

ACR divides cells into smaller cells until the required level of refinement is achieved. ACR techniques have been applied to both Cartesian and curvilinear grids. Szmelter *et al.* [102] contrasted an ANP, a curvilinear ACR and a combined ANP and ACR technique, and concluded that the combined technique offered an improvement over the other two. De Zeeuw & Powell combined a Cartesian cut cell technique with ACR [29, 30] for steady state Euler problems. Chiang *et al.* [21] extended the approach, by including temporal refinement and a corresponding flux fix, in order to compute accurate solutions to transient problems. The storage costs for ACR are on a per cell basis and are therefore greater than for AMR, even with the quad-tree data structure described in [30].

1.3 Thesis Outline

For time dependent discontinuous flows, Riemann problem based, high order, TVD, conservative schemes, such as the WAF scheme, computed on high resolution computational grids are able to produce good quality, cost effective solutions. The work presented in this thesis employs the WAF scheme in conjunction with space and time operator splitting in order to solve two-dimensional time dependent problems.

During the last fifteen years, AMR has had a great deal of success computing discontinuous solutions. It can provide a relatively cheap alternative to a fine regular computational grid, without any noticeable difference in the solution quality. For arbitrary domains the Cartesian cut cell technique has obvious benefits. Unfortunately, high order boundary solutions are difficult to obtain. The author does not subscribe to the view that first order accuracy at boundaries is a tolerable drawback, which is somehow recovered by adaption. In order to achieve the same level of accuracy as a second order scheme, a first order scheme requires a very fine grid. For moving boundaries the Cartesian cut cell approach would need to recalculate the positions of the cuts every time step. The Chimera approach is perhaps the most ideally suited gridding technique for computing non-Cartesian boundaries that are in relative motion. The work presented here combines the Chimera approach and an AMR approach, in order to model internal ballistics problems, which involve fluid interactions with dynamic non-Cartesian boundaries. By adapting the Cartesian AMR grid at solid boundaries, the generation of extremely slender boundary-fitted grids is a simple task, (a matter of extending normals into the flow field).

The layout of the proceeding chapters of this thesis is as follows. A general background to the types of conservation laws, their solutions, the numerical schemes of interest and a description of the WAF scheme are given in chapter 2. Also described are extensions to multi-dimensions via space operator splitting and moving curvilinear computational grids. The theory behind an original interpretation of the AMR algorithm is described in chapter 3. The algorithm is roughly based on those de-

scribed by Berger & Colella [9] and Quirk [79]. The original aspects of the algorithm are discussed and contrasted with the equivalent parts of other AMR algorithms. Chapter 4 describes the Chimera approach and the changes that are required in order to couple it with the AMR algorithm that was described in the previous chapter. Some validation tests for the numerical schemes, the Cartesian AMR algorithm and the combined Chimera-AMR algorithm, are given in chapter 5. The validation tests include CPU time and profiling comparisons with equivalent resolution regular grids. The combined approach has been applied to internal ballistics problems. Chapter 6 details the two-dimensional two-phase internal ballistics model, along with the numerical techniques. The results for several internal ballistics problems are presented in chapter 7. Finally, conclusions are drawn and future developments are discussed in chapter 8.

Chapter 2

Solutions to Hyperbolic Conservation Laws

2.1 Introduction

This chapter is intended to introduce the reader to the analytical and numerical aspects underpinning the numerical methods that are used in work presented in later chapters. Section 2.2 describes the type of mathematical problems that are of interest. The linear advection equation and the Euler equations of gas dynamics are also introduced. The basic theory behind numerical methods for conservation laws is described for the one-dimensional case in section 2.3. This section describes the wave structure for the general Riemann problem and a first order scheme that utilises its solution, known as Godunov's scheme. A second order extension to Godunov's scheme, known as the WAF scheme, is described in section 2.4. Included in this section is a description of the TVD version and the extension to moving grids. Section 2.5 describes a methodology behind multi-dimensional computations based on one-dimensional schemes and the particular intricacies of the two-dimensional WAF scheme. Section 2.6 describes a basic technique for computing non-homogeneous systems of equations. Finally, section 2.7 describes the implementation of boundary conditions for non-Cartesian, moving boundaries. A much more thorough description of the schemes and techniques presented here can be found in [116] and references cited therein.

2.2 Systems of Conservation Laws in One Dimension

In one dimension, the physical sub problems of interest here are described mathematically by systems of hyperbolic conservation laws, of the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}), \tag{2.1}$$

where $\mathbf{U}(x,t)$ (for $-\infty \leq x \leq +\infty$ and $t \geq 0$), is the *vector of conserved variables*, $\mathbf{F}(\mathbf{U})$ is a vector function of the conserved variables called the *flux vector* and $\mathbf{S}(\mathbf{U})$ is the *source term vector*. Explicit time marching methods have been developed for solving homogeneous hyperbolic PDE's, given by the left hand side of (2.1) with $\mathbf{S} = \mathbf{0}$. The simplest example of (2.1) is the linear advection equation

$$u_t + a u_x = 0 \tag{2.2}$$

in which a is a constant coefficient that represents the speed of the wave propagation. Equation (2.2) is often used as a model for developing and understanding other more complicated systems. Equation (2.1), which is known as the *differential form*, can be written in *quasi-linear form* as

$$\mathbf{U}_t + \mathbf{A}(\mathbf{U})\mathbf{U}_x = \mathbf{S}(\mathbf{U}), \tag{2.3}$$

where $\mathbf{A}(\mathbf{U}) = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ is the Jacobian. The eigenvalues of $\mathbf{A}$ represent the characteristic speeds of the system. If, for all values of x, t and $\mathbf{U}$ the eigenvalues are real, then the system is hyperbolic, and if they are also distinct, the system is said to be strictly hyperbolic.

The Euler equations are a system of non-linear, conservation laws, that can be written as

$$\begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}_x = 0, \tag{2.4}$$

where ρ is the density, u is the velocity, p is the pressure and E is the total energy per unit volume, which is given by;

$$E = \frac{1}{2}\rho u^2 + \rho e, \tag{2.5}$$

where e is the specific internal energy. The Euler equations describe the dynamics of an inviscid, compressible fluid (usually a gas). Equations (2.4) and (2.5), are not sufficient to derive the full solution because there are more unknowns than there are equations. The system is closed by the inclusion of the *equation of state*, which is formulated so that e can be described in terms of the other flow variables. Often it is

not necessary for the equation of state to take account of all the physical phenomena that affects e, e.g. the van der Waal's (intermolecular) forces. For ideal gases, the internal energy is related to the temperature, which can be described in terms of the density and pressure via the ideal gas law. For the work presented here, it is sufficient to define e by

$$e(\rho, p) = \frac{p(1 - b\rho)}{(\gamma - 1)\rho}, \tag{2.6}$$

where the constant γ is the ratio of specific heats and b is the covolume. The covolume represents the finite volume occupied by the fluid molecules at high densities. Corner [27] reported that experimentally b changes insignificantly over a realistic range of densities. For the work presented here, the assumption is made that b is a constant. Many problems of gas dynamics can be modelled by the Euler equations and the ideal equation of state; (2.6) reduces to the ideal equation of state if $b = 0$. The system (2.4) is strictly hyperbolic with characteristic speeds $u - a$, u and $u + a$, where a is the sound speed given by

$$a = \sqrt{\frac{\gamma p}{\rho(1 - b\rho)}}. \tag{2.7}$$

The book by Toro [116] or any good gas dynamics text book, such as [1], will provide the reader with a more comprehensive discussion of the Euler equations and various equations of state.

2.3 Numerical Schemes for Conservation Laws

The numerical techniques of interest here are those that are appropriate for solving initial and boundary value problems for hyperbolic conservation laws of the form (2.1). Equation (2.1) is not suitable for representing discontinuous flows, because the derivatives cannot be defined at every point in the flow. The left-hand side of equation (2.1) can be rewritten in *integral form* as

$$\oint (\mathbf{U}\, dx - \mathbf{F}(\mathbf{U})\, dt) = \mathbf{0}. \tag{2.8}$$

The integral form is more relevant to nonlinear hyperbolic problems, since unlike (2.1), it permits discontinuous solutions. Solutions to (2.8) are often referred to as *weak* solutions, in order to distinguish them from the *classical* or *strong* solutions to (2.1). For any rectangular control volume in the $x - t$ plane, $[x_1, x_2] \times [t_1, t_2]$, equation (2.8) can be expanded to give

$$\int_{x_1}^{x_2} \mathbf{U}(x, t_2)\, dx - \int_{x_1}^{x_2} \mathbf{U}(x, t_1)\, dx = \int_{t_1}^{t_2} \mathbf{F}\left(\mathbf{U}(x_1, t)\right)\, dt - \int_{t_1}^{t_2} \mathbf{F}\left(\mathbf{U}(x_2, t)\right)\, dt. \tag{2.9}$$

Having discretised the solution into cells of piece-wise constant data, equation (2.9) can be used to update the solution explicitly, by treating every cell as a control volume. Consider the control volume for cell i in figure 2.1, where Δx is the cell

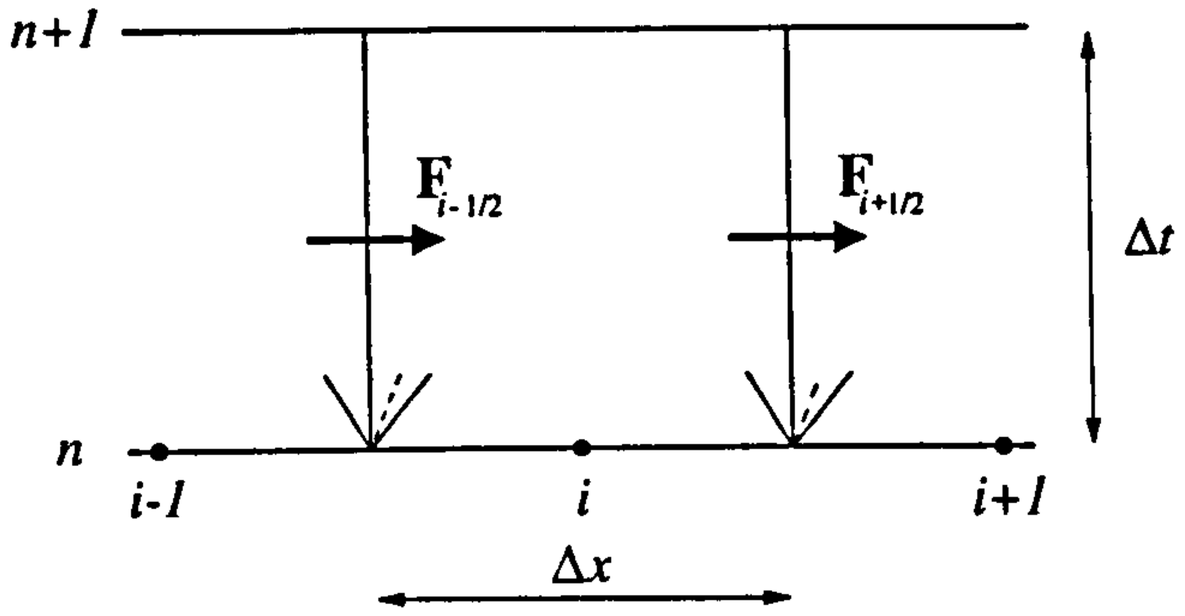


Figure 2.1: An integration control volume for cell i.

spacing, given by $\Delta x = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and $\Delta t = t^{n+1} - t^n$ is the step change in time between consecutive time levels. If $\mathbf{U}_i^n$ is taken to be the integral average of the solution in cell i at time level n, i.e.

$$\mathbf{U}_i^n = \frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}(x, t^n)\, dx \tag{2.10}$$

and $\mathbf{F}_{i+\frac{1}{2}}$ the integral average of the flux at the $i + \frac{1}{2}$ intercell boundary between time levels n and $n+1$, i.e.

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \mathbf{F}\left(\mathbf{U}(x_{i+\frac{1}{2}}, t)\right)\, dt, \tag{2.11}$$

then (2.9) reduces to the numerical conservation formula

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i-\frac{1}{2}} - \mathbf{F}_{i+\frac{1}{2}}\right). \tag{2.12}$$

Explicit conservative schemes generally differ in the way that $\mathbf{F}_{i+\frac{1}{2}}$ is approximated. Most successful conservative schemes utilise the solution of the Riemann problem in order to obtain the flux. Hence, at every intercell boundary the solutions in both the adjacent cells are used to compute the solution to a Riemann problem, which is then used to calculate the flux. Figure 2.1 indicates typical three-wave structures to Riemann problem solutions, that would appear between cell i and each of its neighbouring cells.

Consistency and convergence are two fundamental properties of any useful conservative scheme. Any scheme based on (2.12) is consistent with the conservation law (2.1), if the numerical flux function reduces to the true flux for constant flow. Lax & Wendroff [61] introduced the following theorem for convergence.

Theorem 1 *If, as* $\Delta x \xrightarrow{lim} 0$, *a numerical approximation* $\bar{\mathbf{U}}(x,t)$ *computed with a consistent, conservative scheme of the form (2.12), converges to a function* $\mathbf{U}(x,t)$, *then* $\mathbf{U}(x,t)$ *is a weak solution of the conservation law (2.1).*

However, this theorem does not ensure convergence. It is not possible to guarantee the convergence of a scheme that is not numerically stable. Unfortunately there is no general method for testing, let alone deriving, stable schemes for non-linear systems. However, it is generally accepted that a conservative scheme that is proven to be stable for (2.2), will for practical purposes be stable for non-linear systems. The stability of a scheme is usually expressed in terms of a non-dimensional parameter ν, known as the *Courant number*, which for (2.2) is defined as

$$\nu = a\frac{\Delta t}{\Delta x} \tag{2.13}$$

The Courant number can be regarded as the ratio of the flow speed a to the computational, or grid speed $\Delta x/\Delta t$. Most modern conservative schemes with any practical worth are usually stable for $|\nu| \leq 1$ or more.

2.3.1 The Riemann Problem

The Riemann problem is an initial value problem for a system of hyperbolic PDE's, in which two neighbouring states of constant data are separated by a discontinuous change. The Riemann problem has been solved for various mathematical models, using a number of different exact solvers [44, 78, 108]. For the one-dimensional Euler equations, the Riemann problem solution is an extended[1] model of a shock tube experiment, involving two regions of an inviscid gas that are separated by a diaphragm. The solutions generally agree, very well, with those of the experiment. For strictly hyperbolic systems, the number of wave families, N, in the Riemann problem solution is equal to the number of eigenvalues in the Jacobian $\mathbf{A}(\mathbf{U})$. The waves are separated by $N+1$ constant state regions. Figure 2.2 depicts the three-wave structure of the Riemann problem solution for the one-dimensional Euler equations (2.4). The extreme left and right states, are equal to the initial left and right data $\mathbf{U}_L$ and $\mathbf{U}_R$ respectively. The regions between the left and right waves are referred to as the star regions. Knowledge of the wave structure is needed to calculate the left and right star states $\mathbf{U}_{*L}$ and $\mathbf{U}_{*R}$. However, the star states are needed in order to ascertain the wave structure. Thus, what are referred to as exact solvers, actually need to iterate to obtain the solution. The wave types can be revealed by examining the characteristic fields associated with the eigenvectors of $\mathbf{A}(\mathbf{U})$. The characteristic field for the central wave is linearly degenerate, whereas those of the outer waves are genuinely non-linear [116]. Whilst the central wave is always a contact discontinuity,

[1] Unlike the shock tube experiment, the initial velocities can be non-zero.

each of the non-linear waves can be either a shock or a rarefaction wave. All the flow variables change continuously across rarefaction waves and discontinuously across shock waves. Neither the pressure nor the velocity[2] changes across contact waves. Thus, the conserved variable vectors in the four regions are:

$$\mathbf{U}_L = \begin{bmatrix} \rho_L \\ \rho_L u_L \\ E_L \end{bmatrix}; \mathbf{U}_{*L} = \begin{bmatrix} \rho_{*L} \\ \rho_{*L} u_* \\ E_{*L} \end{bmatrix}; \mathbf{U}_{*R} = \begin{bmatrix} \rho_{*R} \\ \rho_{*R} u_* \\ E_{*R} \end{bmatrix}; \mathbf{U}_R = \begin{bmatrix} \rho_R \\ \rho_R u_R \\ E_R \end{bmatrix}.$$

where ρ, u and E represent the density, velocity and total energy per unit volume respectively. The pressure p in each region, which only changes across the non-linear waves, can be described in terms of the other variables, i.e.

$$p = \left(\frac{\gamma - 1}{1 - b\rho}\right)\left(E - \frac{1}{2}\rho u^2\right) \tag{2.14}$$

The characteristic speeds of the left, middle and right waves are given by the eigenvalues $\lambda_1 = u - a$, $\lambda_2 = u$ and $\lambda_3 = u + a$ respectively, (the sound speed a is given by equation (2.7)). Generally, the characteristics on either side of the waves have different velocities relative to that of the wave. Suppose that in figure 2.2, the left

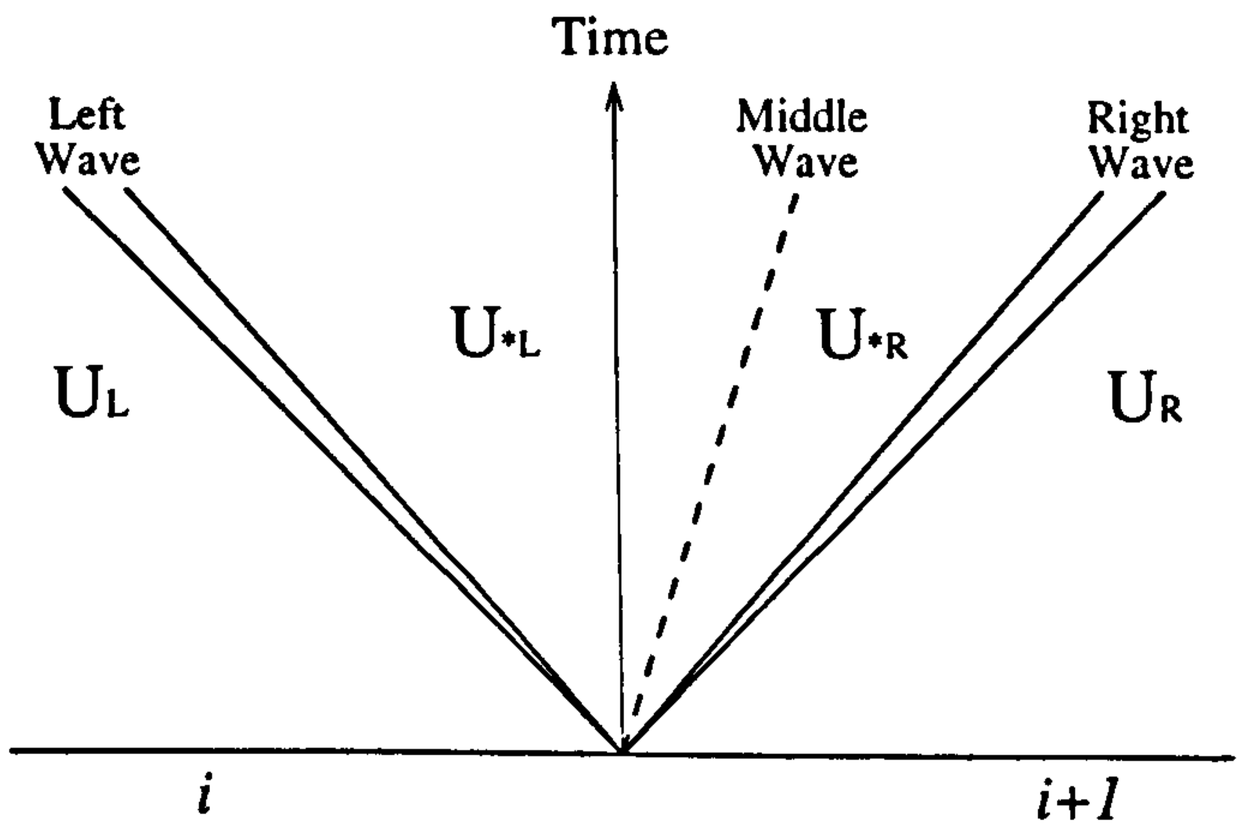


Figure 2.2: The structure of the Riemann problem for a three wave system.

wave is a rarefaction and the right wave is a shock. Across rarefaction waves the characteristics diverge and decrease in speed from head to tail. Hence, for a left rarefaction wave

$$S_H = \lambda_1(\mathbf{U}_L) \leq \frac{\hat{x}}{\hat{t}} \leq \lambda_1(\mathbf{U}_{*L}) = S_T,$$

where S_H and S_T are the head and tail velocities respectively of the rarefaction and $\hat{x}/\hat{t}$ represents the velocity of any ray within it. Conversely, the characteristics on

[2]In two dimensions, the shear velocity does change across contact waves.

either side of a shock wave focus on the wave; those ahead have slower speeds than the ones behind. Hence, for a right shock wave

$$\lambda_3(\mathbf{U}_{*R}) > S_R > \lambda_3(\mathbf{U}_R),$$

where S_R is the velocity of the shock wave. The characteristics on either side of the contact wave run parallel to the wave, i.e.

$$\lambda_2(\mathbf{U}_{*L}) = S_M = \lambda_2(\mathbf{U}_{*R}),$$

where S_M is the velocity of the contact wave. The wave velocities of rarefaction and contact waves correspond to velocities of the characteristics. Hence, for this example, the $S_M = u_*$ and the velocity at any point $(\hat{x}, \hat{t})$ in space-time inside the rarefaction wave is $S = \hat{x}/\hat{t} = \hat{u} - \hat{a}$, e.g. the head and tail velocities of the rarefaction are given by $S_H = u_L - a_L$ and $S_T = u_* - a_{*L}$ respectively. The velocity of the shock wave S_R is given by

$$S_R = u_R + a_R \sqrt{\left(\frac{\gamma+1}{2\gamma}\right)\left(\frac{p_*}{p_R}\right) + \left(\frac{\gamma-1}{2\gamma}\right)} \tag{2.15}$$

The utilisation of the Riemann problem solution, was a significant development in the computation of discontinuous flows. A number of approximate solutions to the Riemann problem have been presented [73, 85, 115] and [119], that are reasonably accurate whilst being computationally less expensive than exact solvers. A hierarchy of approximate Riemann solvers can be used, so that the accuracy of the solution and the corresponding computational effort, varies according to the severity of the flow conditions. An improvement of the HLL approximate Riemann solver [51], known as the HLLC Riemann solver [119], is used to compute the results presented in this thesis. Toro *et al.* [119] showed that solutions computed with the HLLC Riemann solver compare very well with those computed with an exact Riemann solver, whilst it was 50% faster for equations of state of the form (2.6). They also suggested that greater improvements in efficiency would result for more complex equations of state, for which exact solvers become more costly.

2.3.2 Godunov's Scheme

Godunov [41] was the first to use the solution of the Riemann problem to provide upwind information within a conservative scheme. By discretising the solution into piece-wise constant data, Riemann problems can be computed at the boundaries between every pair of neighbouring cells. Typical three-wave structures of the Riemann problems that emanate from the intercell boundaries at $x_{i-\frac{1}{2}}$ and the $x_{i+\frac{1}{2}}$ are shown in figure 2.3. The local time step is given by $\Delta t = t^{n+1} - t^n$. Taking

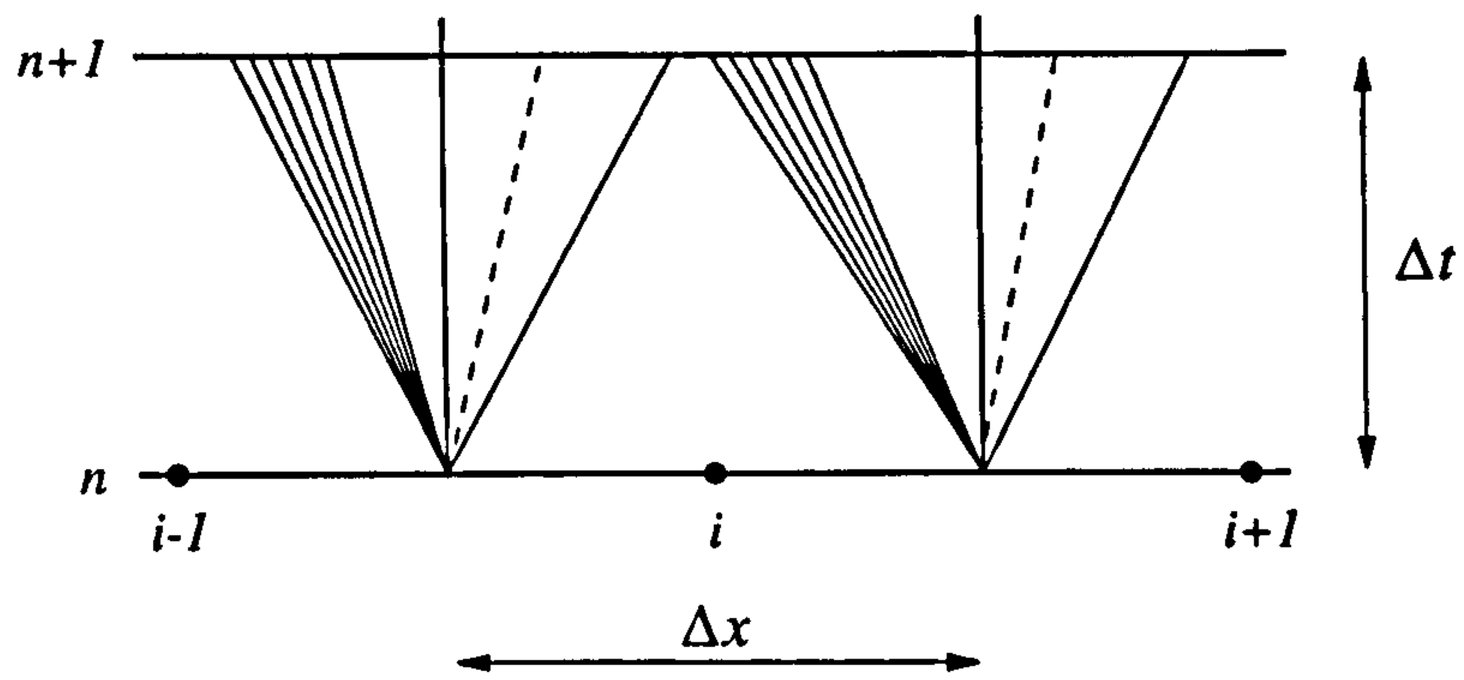


Figure 2.3: The structure of the Riemann problem solutions for each intercell boundary adjoining cell i.

the solution vector $\mathbf{U}_i^n$ to be piece-wise constant in cell i at time level n, the solution generated by Riemann problem at the $i+\frac{1}{2}$ intercell boundary, with initial left and right data $\mathbf{U}_i^n$ and $\mathbf{U}_{i+1}^n$ respectively, can be written as $\widetilde{\mathbf{U}}_{i+\frac{1}{2}}^n(\hat{x}/\hat{t})$, where $\hat{x} = x - x_{i+\frac{1}{2}}$ and $\hat{t} = t - t^n$ are the local space and time coordinates. Godunov chose the time step Δt and the corresponding time level t^{n+1}, such that none of the waves from any pair of neighbouring Riemann solutions interacted. Wave interaction is avoided for $\nu \leq \frac{1}{2}$. For one-dimensional problems, it is possible to first calculate all the wave speeds, before determining the time step. Hence, the time step is given by

$$\Delta t \leq \text{MIN}\left(\frac{\Delta x}{\omega_{i-\frac{1}{2}}^{(N)} - \omega_{i+\frac{1}{2}}^{(1)}}\right); \quad \text{for } 1 \leq i \leq m,$$

where m is the number of cells in the discretisation and $\omega_{i+\frac{1}{2}}^{(k)}$ is the velocity of the k^{th} wave of the N wave Riemann solution positioned at the $i+\frac{1}{2}$ intercell boundary. The time step may be over estimated for situations in which waves are accelerated after interacting with one another. To ensure stability, the time step is multiplied by a constant fraction between zero and one, (0.9 is a typical value). The solution in cell i at time level $n+1$, is then calculated by taking an integral average, between $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$, of the solution $\widetilde{\mathbf{U}}(x/t)$ given by piecing together the solutions of neighbouring Riemann problems, i.e.

$$\mathbf{U}_i^{n+1} = \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \widetilde{\mathbf{U}}(x/\Delta t)\, dx. \tag{2.16}$$

Instead of updating the solution by integrating the evolved Riemann solution data in space, a slight variation on (2.16) is given by applying the integral form of the conservation law to every cell control volume. Hence, the solution can be updated

via the conservative update formula,

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}\left[\mathbf{F}(\widetilde{\mathbf{U}}_{i-\frac{1}{2}}^n) - \mathbf{F}(\widetilde{\mathbf{U}}_{i+\frac{1}{2}}^n)\right] \tag{2.17}$$

where $\mathbf{F}(\widetilde{\mathbf{U}}_{i+\frac{1}{2}}^n)$ is the time integral average of the flux, computed with the solution of the Riemann problem along the $i+\frac{1}{2}$ intercell boundary, i.e. the solution along the t axis. Thus, because the solution along the t axis remains constant until a wave from a neighbouring Riemann problem reaches it, the time step is not as restricted as that for the original derivation, i.e.

$$\Delta t = C.\text{MIN}\left(\frac{\Delta x}{\omega_{i+\frac{1}{2}}^{(k)}}\right); \quad \begin{array}{ll} \text{for} & 0 \le i \le m \\ \text{and} & 1 \le k \le N \end{array} \tag{2.18}$$

where C is a constant coefficient. Indeed, it can be shown that for the model linear advection equation (2.2), Godunov's scheme is stable for Courant numbers less than or equal to unity. For non-linear systems, waves may be accelerated by interactions with other waves. It is therefore necessary to introduce a margin of safety into the time step calculation, by using a Courant number that is marginally less than one. Hence, stability is ensured by choosing a value of the constant coefficient C between zero and one (typically $C = 0.9$). Godunov's scheme is only first order accurate and is therefore not a very practical choice for solving general fluid dynamics problems. It not only diffuses discontinuous features, such as contact and shear waves to unacceptable levels, but it is also unable to maintain the representation of smooth features over even a moderate number of time steps. However, Godunov's scheme has provided the basis for several different higher order schemes.

2.4 The Weighted Average Flux Scheme

The WAF scheme is a second order extension of Godunov's scheme that was presented by Toro [109]. Like Godunov's scheme, WAF uses the solution of the Riemann problems, computed at the intercell boundaries between piece-wise constant data, to calculate the numerical flux. It achieves second order accuracy, not by reconstructing the discretised data as with MUSCL schemes [4, 24, 128], but by examining more closely the structure of the Riemann problem solution $\widetilde{\mathbf{U}}_{i+\frac{1}{2}}(x/t)$. A second order flux $\mathbf{F}_{i+\frac{1}{2}}$ can be obtained by integrating the flux in both space, from x_i to x_{i+1}, and time, from t^n to t^{n+1}, i.e.

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{\Delta t \Delta x}\int_{t^n}^{t^{n+1}}\int_{x_i}^{x_{i+1}} \mathbf{F}\left[\widetilde{\mathbf{U}}_{i+\frac{1}{2}}(x/t)\right] dx\, dt. \tag{2.19}$$

The WAF scheme approximates the time integration in (2.19) by the mid-point value. Hence,

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{\Delta x}\int_{x_i}^{x_{i+1}} \mathbf{F}\left[\widetilde{\mathbf{U}}_{i+\frac{1}{2}}\left(\frac{x}{\Delta t/2}\right)\right] dx \tag{2.20}$$

If the assumption is made that the individual N waves in the Riemann problem can be represented by a single ray, then equation (2.20) reduces to a simple weighted summation of the fluxes in the $N+1$ regions. Consider figure 2.4, which depicts a Riemann problem solution with a three-wave wave structure and the corresponding weights and solution values. The assumption that all the Riemann problem waves

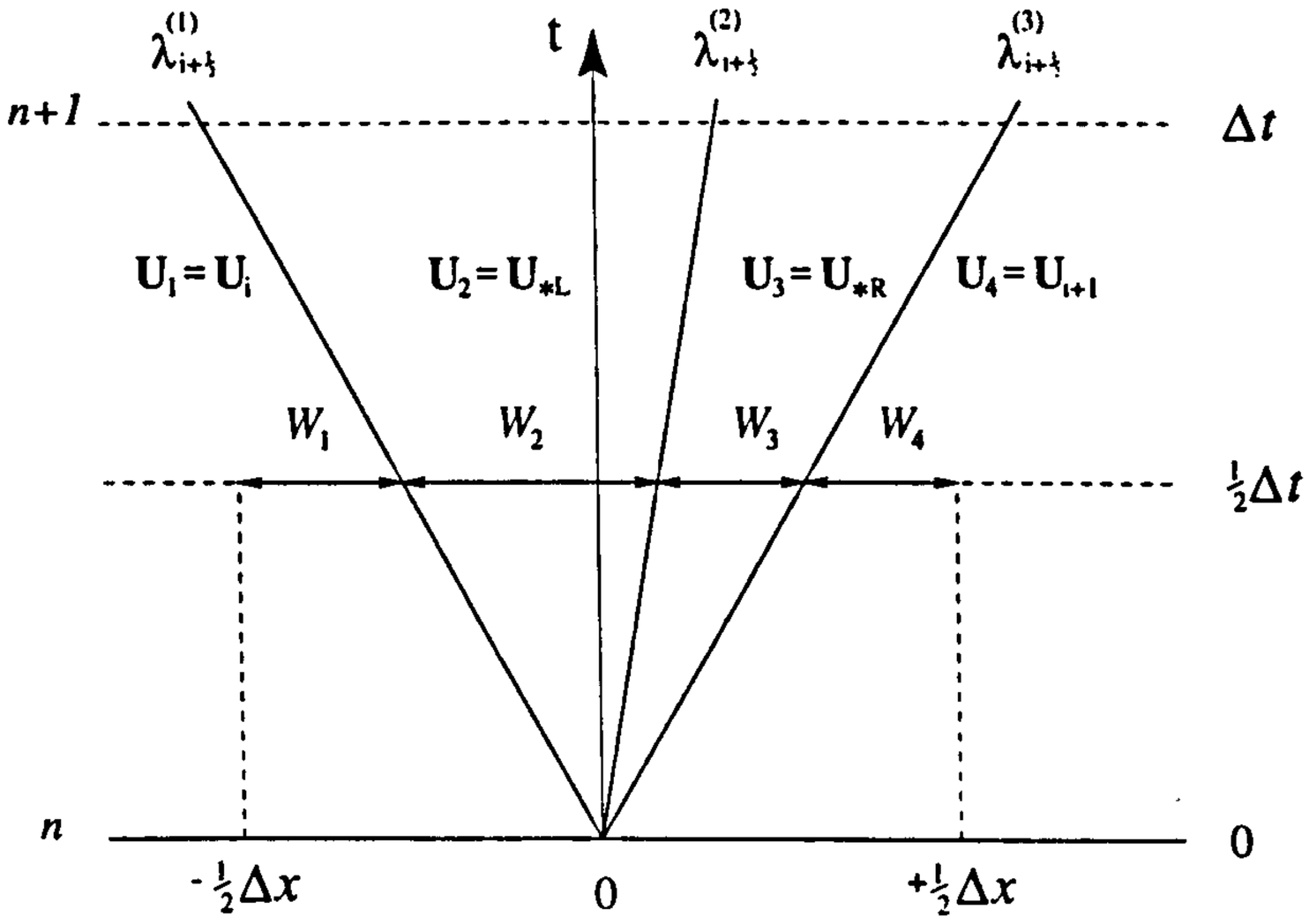


Figure 2.4: The weights W_k for the WAF integration across a three wave Riemann problem.

are single rays is perfectly correct for contact waves and shock waves, but is clearly an approximation for rarefaction waves. Approximating rarefaction waves by their leading characteristics (those with the greatest speed) has proved to be sufficiently accurate. The local Courant number for every wave in the Riemann problem, that is generated at the $i+\frac{1}{2}$ intercell boundary, is given by

$$\nu^{(k)}_{i+\frac{1}{2}} = \omega^{(k)}_{i+\frac{1}{2}}\frac{\Delta t}{\Delta x}; \quad \text{for } 1 \le k \le N \tag{2.21}$$

where $\omega^{(k)}_{i+\frac{1}{2}}$ represents the speed of the k^{th} wave. Hence, the weights $W^{(k)}$ are given by

$$W^{(k)} = \frac{1}{2}\left(\nu^{(k)}_{i+\frac{1}{2}} - \nu^{(k-1)}_{i+\frac{1}{2}}\right); \quad \text{for } 1 \le k \le N+1 \tag{2.22}$$

in which

$$\nu_{i+\frac{1}{2}}^{(0)} = -1 \text{ and } \nu_{i+\frac{1}{2}}^{(N+1)} = +1 \tag{2.23}$$

Therefore, the WAF flux can be expressed as

$$\mathbf{F}_{i+\frac{1}{2}} = \sum_{k=1}^{N+1} W^{(k)} \mathbf{F}_{i+\frac{1}{2}}^{(k)} \tag{2.24}$$

where $\mathbf{F}_{i+\frac{1}{2}}^{(k)}$ is the numerical flux in the k^{th} region of the Riemann problem solution, i.e.

$$\mathbf{F}_{i+\frac{1}{2}}^{(k)} = \mathbf{F}\left(\mathbf{U}_k\right)$$

The flux for a region containing a rarefaction wave is evaluated from the appropriate star state solution in the Riemann problem. For example, if in figure 2.4 the first wave is a rarefaction, the flux vector in the second region $F_{i+\frac{1}{2}}^{(2)}$ would be evaluated with the U_{*L} solution, i.e.

$$\mathbf{F}_{i+\frac{1}{2}}^{(2)} = \mathbf{F}\left(\mathbf{U}_{*L}\right)$$

However, for sonic rarefactions (those that span the t-axis), the solution data is taken as that along the $x/t = 0$ characteristic, i.e.

$$\mathbf{F}_{i+\frac{1}{2}}^{(k)} = \mathbf{F}\left(\widetilde{\mathbf{U}}_{i+\frac{1}{2}}(x/t=0)\right)$$

Note that, the distinction for sonic rarefactions is not needed for the HLLC Riemann solver [119]. Having obtained the WAF flux (2.24) for both the $i-\frac{1}{2}$ and $i+\frac{1}{2}$ intercell boundaries, the solution in the cell i can be updated using the conservation formula (2.12).

2.4.1 A TVD Version of the WAF Scheme

As previously stated, the WAF scheme is second order accurate in space and time. It therefore produces unphysical spurious oscillations in the vicinity of discontinuous solutions. In order for the solutions computed with it to remain oscillation free, it is necessary to limit, (depending on the severity of the data), the scheme, by reducing the order of accuracy. The WAF scheme has been made TVD for the model linear advection equation, the details of which are given in [109]. The extension of the TVD theory to non-linear systems for the WAF scheme is summarised here.

The accuracy of the scheme is limited by modifying the weights $W^{(k)}$ in the second order scheme described above, i.e.

$$W^{(k)} = \frac{1}{2}\left(\phi_{i+\frac{1}{2}}^{(k)} - \phi_{i+\frac{1}{2}}^{(k-1)}\right); \text{ for } 1 \leq k \leq N+1 \tag{2.25}$$

in which

$$\phi^{(0)}_{i+\frac{1}{2}} = -1 \quad \text{and} \quad \phi^{(N+1)}_{i+\frac{1}{2}} = +1 \tag{2.26}$$

where $\phi^{(k)}_{i+\frac{1}{2}}$ represents a limiter function for the k^{th} wave in the Riemann problem. For the WAF scheme, $\phi^{(k)}_{i+\frac{1}{2}}$ is a function of both the wave Courant number, given by equation (2.21), and the flow parameter $r^{(k)}_{i+\frac{1}{2}}$. The flow parameter is the ratio of the upwind to the local jump in some physical quantity q across the Riemann problem waves, i.e.

$$r^{(k)}_{i+\frac{1}{2}} = \frac{\Delta^{(k)}_{upw} q}{\Delta^{(k)}_{loc} q}$$

The upwind and local jumps are given by:

$$\Delta^{(k)}_{upw} q \;=\; \begin{cases} q^{(k)}_{i-\frac{1}{2}} - q^{(k-1)}_{i-\frac{1}{2}} & \text{if } \nu^{(k)}_{i+\frac{1}{2}} > 0 \\ \\ q^{(k)}_{i+\frac{3}{2}} - q^{(k-1)}_{i+\frac{3}{2}} & \text{otherwise} \end{cases}$$

$$\Delta^{(k)}_{loc} q \;=\; q^{(k)}_{i+\frac{1}{2}} - q^{(k-1)}_{i+\frac{1}{2}}$$

For models such as the Euler equations, the density has been shown to be a good choice for q.

Dispensing with the $\nu^{(k)}_{i+\frac{1}{2}}$ notation in favour of ν, the flux limiter yields a first or second order accurate scheme if, for all i and k, $\phi^{(k)}_{i+\frac{1}{2}} = \text{sgn}\,[\nu]$ or $\phi^{(k)}_{i+\frac{1}{2}} = \nu$ respectively. Generally, the flux limiter has a value somewhere between the two extremes. Three common limiter functions that have been used in conjunction with the WAF scheme are

$$\phi(r,\nu) \;=\; \text{sgn}[\nu].\begin{cases} 1 & r \le 0 \\ \\ 1 - r(1 - |\,\nu\,|) & 0 < r \le 1 \\ \\ |\,\nu\,| & r > 1 \end{cases} \tag{2.27}$$

$$\phi(r,\nu) \;=\; \text{sgn}[\nu].\begin{cases} 1 & r \le 0 \\ \\ 1 - \frac{2r}{(1+r)}(1 - |\,\nu\,|) & r > 0 \end{cases} \tag{2.28}$$

$$\phi(r,\nu) = \mathrm{sgn}[\nu].\begin{cases} 1 & r \le 0 \\ 1-2r(1-|\nu|) & 0 < r \le \frac{1}{2} \\ |\nu| & \frac{1}{2} < r \le 1 \\ 1-r(1-|\nu|) & 1 < r \le 2 \\ 1-2(1-|\nu|) & r > 2 \end{cases} \tag{2.29}$$

Limiter functions (2.27), (2.28) and (2.29) are related to the MINBEE [87], VAN LEER [123, 124] and SUPERBEE [87] flux limiters respectively.

In this form the WAF scheme has been successfully applied to a variety of problems, including shallow water [113], gas dynamics [114], magnetohydrodynamics [18] and multi-phase [110, 117] problems. Solutions computed with the TVD WAF scheme are very accurate compared to other second order schemes, especially those of the MUSCL type. The wave-by-wave limiting that is inherent with the WAF scheme, appears to provide much better resolution of discontinuous features, especially contact and shear waves. For a fuller discussion of wave-by-wave limiting refer to [57].

2.4.2 The WAF Scheme Applied to Moving Grids

Presented here is a summary of the WAF scheme applied to moving grids; refer to [14, 16] for a fuller description. A conservative extension of the WAF scheme in one dimension is derived from the integral form of the conservation laws, given by equation (2.8). Consider the control volume for cell i depicted in figure 2.5, in which a grid with constant cell spacing Δx, moves with a constant velocity V, during the time period t^n to t^{n+1} ($\Delta t = t^{n+1} - t^n$). Expanding equation (2.8) for the control

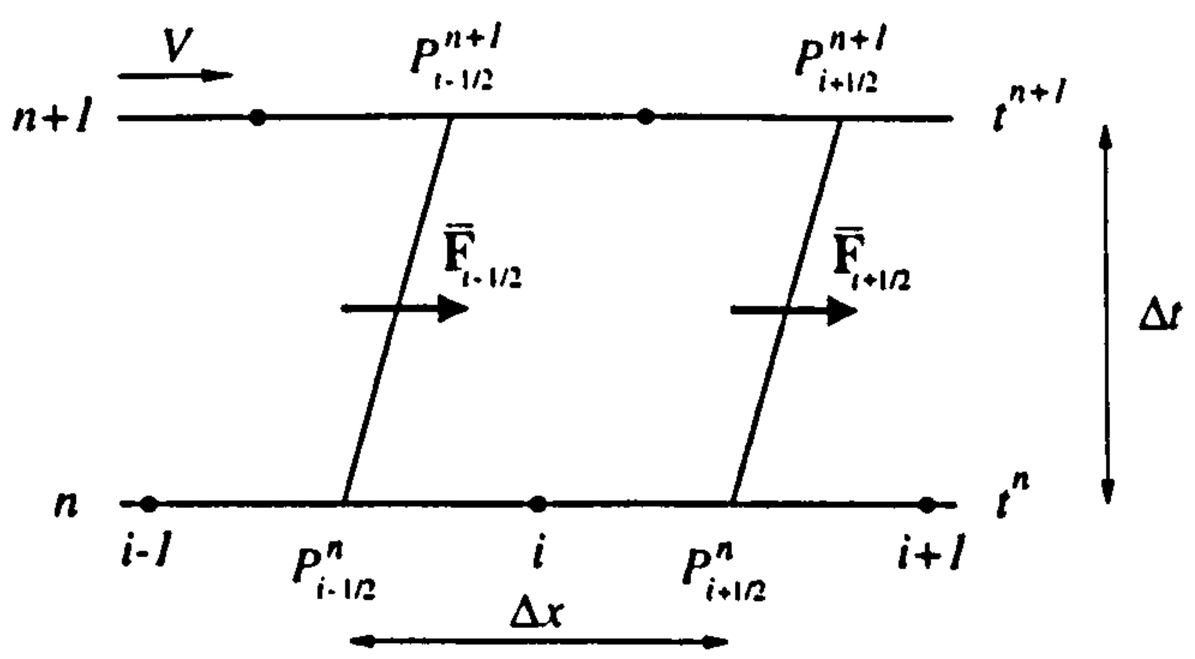


Figure 2.5: An integration control volume for cell i on a moving grid.

volume $\left[P^n_{i-\frac{1}{2}}, P^n_{i+\frac{1}{2}}, P^{n+1}_{i+\frac{1}{2}}, P^{n+1}_{i-\frac{1}{2}}\right]$, where the point $P^n_{i+\frac{1}{2}}$ represents the space-time coordinate $(x^n_{i+\frac{1}{2}}, t^n)$, gives

$$\int_{P^{n+1}_{i-\frac{1}{2}}}^{P^{n+1}_{i+\frac{1}{2}}} \mathbf{U}^{n+1}\,dx - \int_{P^n_{i-\frac{1}{2}}}^{P^n_{i+\frac{1}{2}}} \mathbf{U}^n\,dx = \int_{P^n_{i-\frac{1}{2}}}^{P^{n+1}_{i-\frac{1}{2}}} (\mathbf{F}\,dt - \mathbf{U}\,dx) - \int_{P^n_{i+\frac{1}{2}}}^{P^{n+1}_{i+\frac{1}{2}}} (\mathbf{F}\,dt - \mathbf{U}\,dx) \quad (2.30)$$

If $\mathbf{U}^n_i$ is taken to be the integral average of the solution in cell i at time level n, i.e.

$$\mathbf{U}^n_i = \frac{1}{\Delta x} \int_{x^n_{i-\frac{1}{2}}}^{x^n_{i+\frac{1}{2}}} \mathbf{U}(x, t^n)\,dx \quad (2.31)$$

and $\overline{\mathbf{F}}_{i+\frac{1}{2}}$ is taken to be the integral average, of an approximation to the flux along the moving $i+\frac{1}{2}$ intercell boundary, between time levels n and $n+1$, i.e.

$$\overline{\mathbf{F}}_{i+\frac{1}{2}} = \frac{1}{\Delta t} \int_{P^n_{i+\frac{1}{2}}}^{P^{n+1}_{i+\frac{1}{2}}} (\mathbf{F}\,dt - \mathbf{U}\,dx), \quad (2.32)$$

then the numerical conservation formula can be written as

$$\mathbf{U}^{n+1}_i = \mathbf{U}^n_i + \frac{\Delta t}{\Delta x}\left(\overline{\mathbf{F}}_{i-\frac{1}{2}} - \overline{\mathbf{F}}_{i+\frac{1}{2}}\right) \quad (2.33)$$

The integration in equation (2.32) implies a flux given by

$$\overline{\mathbf{F}}_{i+\frac{1}{2}} = \mathbf{F}(\mathbf{U}_{i+\frac{1}{2}}) - V\mathbf{U}_{i+\frac{1}{2}} \quad (2.34)$$

A second order approximation to $\overline{\mathbf{F}}_{i+\frac{1}{2}}$ can be obtained in a similar way to that for the flux in the WAF scheme. Hence, the moving grid WAF scheme approximates the flux by an integration in space at the mid-point time. The range of integration is distorted so as to reflect the movement of the grid. Hence,

$$\overline{\mathbf{F}}_{i+\frac{1}{2}} = \frac{1}{\Delta x} \int_{x_i^{n+\frac{1}{2}}}^{x_{i+1}^{n+\frac{1}{2}}} \left[\mathbf{F}\left(\widetilde{\mathbf{U}}_{i+\frac{1}{2}}(x/t^{n+\frac{1}{2}})\right) - V\widetilde{\mathbf{U}}_{i+\frac{1}{2}}(x/t^{n+\frac{1}{2}})\right] dx \quad (2.35)$$

where $\widetilde{\mathbf{U}}_{i+\frac{1}{2}}$ is the solution of the Riemann problem evaluated at the $i+\frac{1}{2}$ intercell boundary. As with the standard WAF scheme, the integral in equation (2.35) can be approximated by a simple weighted summation, given by

$$\overline{\mathbf{F}}_{i+\frac{1}{2}} = \sum_{k=1}^{N+1} W^{(k)}\left(\mathbf{F}^{(k)}_{i+\frac{1}{2}} - V\widetilde{\mathbf{U}}^{(k)}_{i+\frac{1}{2}}\right) \quad (2.36)$$

in which $\mathbf{F}^{(k)}_{i+\frac{1}{2}}$ and $\widetilde{\mathbf{U}}^{(k)}_{i+\frac{1}{2}}$ are the flux and solution vectors respectively, in the k^{th} region of the Riemann solution. The approximation assumes that each of the waves

can be approximated by single rays. For moving grids, the situation in which the head and tail velocities of a rarefaction bound the grid velocity, is analogous to the sonic rarefaction case for fixed grids. In such a situation, the solution is computed from the Riemann problem solution along the characteristic with velocity equal to that of the grid. Otherwise, the solution in a region containing a rarefaction, is taken from the solution in the appropriate star state region. Note that, this distinction is not needed for the HLLC Riemann solver [119]. Figure 2.6 depicts the weights $W^{(k)}$ for a three wave Riemann problem, centred at the $i+\frac{1}{2}$ intercell boundary. The limited weights are calculated using equations (2.25) and (2.26), but with the

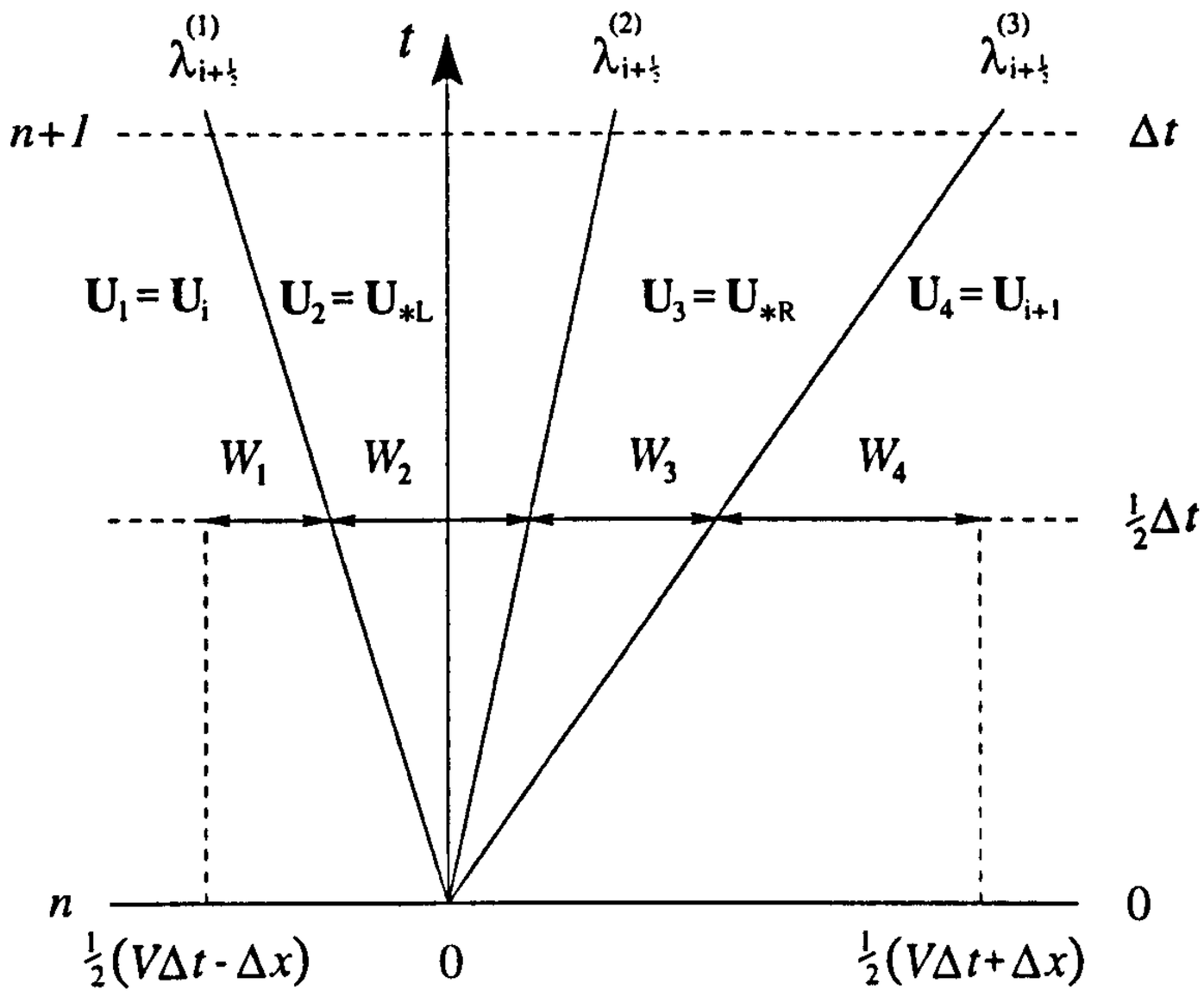


Figure 2.6: The weights W_k for the WAF integration across a three wave Riemann problem on a grid moving with velocity V.

limiter function $\phi^{(k)}_{i+\frac{1}{2}}$ altered, such that it is dependent on the relative wave Courant number $\mu^{(k)}_{i+\frac{1}{2}} = \nu^{(k)}_{i+\frac{1}{2}} - \nu_g$, i.e. $\phi^{(k)}_{i+\frac{1}{2}} = \phi(r, \mu)$, where $\nu^{(k)}_{i+\frac{1}{2}}$ is defined by (2.21) and ν_g is a similar non-dimensionalised parameter that reflects the grid movement given by

$$\nu_g = V\frac{\Delta t}{\Delta x}$$

The moving grid WAF scheme is stable providing the time step is chosen such that, for all values of i and k, $\mid \mu^{(k)}_{i+\frac{1}{2}} \mid \leq 1$.

2.5 Multi-Dimensional Computations

General physical problems cannot always be idealised by one-dimensional models. One-dimensional schemes can easily be extended to multi-dimensions by the technique known as space operator splitting [100, 133]. There are other, more involved, techniques, that do have some advantages (e.g. more accurate), such as finite volume and multi-dimensional upwind techniques. However, a valuable property of one-dimensional TVD shock capturing schemes is their ability to compute discontinuous solutions without introducing spurious oscillatory errors. Unfortunately, schemes that impose TVD constraints in multi-dimensions, lose some of their accuracy [43]. The work presented here is predominantly concerned with two-dimensional grid adaption. In order to demonstrate the grid adaption process and ensure that, as far as possible[3], the computed solutions are not marred by any unphysical oscillations, a simple space operator split TVD scheme was used.

The physical problems of interest here, can be described mathematically by systems of conservation laws, of the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = \mathbf{S}(\mathbf{U}) \tag{2.37}$$

where $\mathbf{U} = \mathbf{U}(x, y, t)$ is the vector of conserved variables, $\mathbf{F}(\mathbf{U})$ and $\mathbf{G}(\mathbf{U})$ are the flux vectors in the x and y directions respectively, and $\mathbf{S}(\mathbf{U})$ is the source term vector. For example the two-dimensional homogeneous Euler equations are

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}_x + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_y = \mathbf{0} \tag{2.38}$$

where ρ and p represent the fluid density and pressure respectively, and u and v the x and y components of velocity. The total energy per unit volume E is given by

$$E = \frac{1}{2}\rho(u^2 + v^2) + \rho e \tag{2.39}$$

in which the specific internal energy e is given by the equation of state (2.6).

Space operator splitting schemes update the solution data in each computational direction separately. Consider the two-dimensional situation in which the computational directions ζ and η, correspond to the grid lines indexed by i and j respectively. The split two-dimensional Euler equations for each of the two computational directions are

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}_\zeta = \mathbf{0} \tag{2.40}$$

[3]Note that, the TVD theory does not extend to non-linear systems.

and

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_\eta = 0 \tag{2.41}$$

where, as in equations (2.38) and (2.39), ρ, p and E represent the density, pressure and total energy per unit volume respectively, but here u and v represent the components of velocity in the ζ and η directions respectively. The operator split scheme used throughout the work presented here, involves only a single sweep in each computational direction. It can be summarised in operator form as follows:

$$\mathbf{U}^{n+1} = L_\eta^{(\Delta t)} \left(L_\zeta^{(\Delta t)} (\mathbf{U}^n) \right), \tag{2.42}$$

where the operators $L_\zeta^{(\Delta t)}$ and $L_\eta^{(\Delta t)}$ represent all the one-dimensional strip update operations, in the ζ and η directions respectively. The superscript denotes the time step Δt. Scheme (2.42) is formally only first order accurate in time, regardless of the order of accuracy of each operator. However, the differences between solutions calculated with it and those computed with second order split schemes, such as the one presented by Strang [100]:

$$\mathbf{U}^{n+1} = L_\zeta^{(\frac{\Delta t}{2})} \left(L_\eta^{(\Delta t)} \left(L_\zeta^{(\frac{\Delta t}{2})} (\mathbf{U}^n) \right) \right), \tag{2.43}$$

are less apparent than those between one-dimensional first and second order schemes. Both (2.42) and (2.43) distort the numerical solution with a directional bias. When used to calculate circular dam break problems, split schemes produce slightly non-circular wave structures [13]. The distortion in (2.43) can be alleviated to a large extent by switching the half time step direction every time step. This results in a scheme that is 50% more expensive than (2.42).

For curvilinear grids ζ and η are not generally aligned with the x and y directions. Moreover, they are parametric directions according to the grid geometry, i.e. the angles of the cell sides will generally vary with i and j. The type of schemes of interest here, utilise the Riemann problem solution at the intercell boundaries to evaluate the flux. For curvilinear grids, the components of the solution normal and tangential to the intercell boundaries form the initial conditions for the Riemann problems. The normal and tangential components are evaluated for every boundary by rotating the solution vectors in the adjacent cells. Note that, for the Euler equations, only the momentum fluxes are vector quantities, i.e. the density, pressure and energy are unaffected by the rotations. Having calculated the normal and tangential flux components for every boundary, the x and y components are calculated by a reverse rotation. Once all the fluxes in a strip have been calculated, the solution can be updated using the modified conservation formula for curvilinear grids, which

for the ζ direction updates, is written as:

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} + \frac{\Delta t}{\mathrm{A}_{i,j}} \left[l_{i-\frac{1}{2},j}\, \mathbf{F}_{i-\frac{1}{2},j} - l_{i+\frac{1}{2},j}\, \mathbf{F}_{i+\frac{1}{2},j} \right]. \tag{2.44}$$

In (2.44) $\mathrm{A}_{i,j}$ is the area of the cell (i,j) and $l_{i+\frac{1}{2},j}$ is the length of the $(i+\frac{1}{2},j)$ boundary. A similar equation to (2.44) updates the solution with the η direction fluxes $\mathbf{F}_{i,j+\frac{1}{2}}$ and intercell boundary lengths $l_{i,j+\frac{1}{2}}$. For three-dimensional split schemes the update formulas are similar, but with the cell areas and side lengths replaced with volumes and face areas respectively. Generally, the lengths of the $(i-\frac{1}{2},j)$ and $(i+\frac{1}{2},j)$ boundaries are not the same. Hence, even if the solution is constant throughout the grid, if $\zeta_{i-\frac{1}{2},j} \neq \zeta_{i+\frac{1}{2},j}$ or $l_{i-\frac{1}{2},j} \neq l_{i+\frac{1}{2},j}$, then for the general situation $\mathbf{U}_{i,j}^{n+1} \neq \mathbf{U}_{i,j}^{n}$. Thus, an error is introduced into the solution. This type of error always occurs with split curvilinear schemes. By constructing grids with only a gradual variation in the side lengths and computational directions, the splitting errors can be minimised.

An analysis of the Jacobian matrix for either (2.40) or (2.41) reveals that there are four eigenvalues, given by

$$\omega^{(1)} = u - a, \ \omega^{(2)} = \omega^{(3)} = u \ \text{ and } \ \omega^{(4)} = u + a,$$

where the sound speed a is given by equation (2.7). Hence, the split Euler equations are not *strictly* hyperbolic because, even though the eigenvalues are all real, they are not distinct. Further analysis of the characteristic fields reveals that the fourth wave represents a shear wave, across which only the tangential component of velocity changes. Therefore, the split two-dimensional Riemann solution is equivalent to the one-dimensional solution with a shear wave superimposed on the contact wave. Thus, in the Riemann problem solution for the Euler equations, the tangential component of velocity changes discontinuously across the coincident middle waves. Hence, all the flow variables except the tangential velocity change across the outer waves (non-linear), but the pressure and normal velocity remain constant across the coincident middle waves (linear).

It is not practical to calculate all the wave speeds in a multi-dimensional problem before determining the time step. A reasonably good estimate for the wave speed in a cell, is given by the characteristic speed. Hence, the time step for the Euler equations can be estimated by:

$$\Delta t = C.\mathrm{MIN}\left(\frac{\delta_{i,j}}{|V_{i,j}| + a_{i,j}} \right) \ \forall\ i,j \tag{2.45}$$

where $\delta_{i,j}$ is representative of the cell spacing, $V_{i,j}$ and $a_{i,j}$ are the total velocity and sound speed respectively, in cell (i,j). As in equation (2.18), $0 < C \leq 1$ is a constant coefficient. Note that, (2.45) is only an estimate and should be used with

care, e.g. for problems in which the initial velocity is zero, the actual emerging wave velocities will be greater than the eigenvalues and therefore an over estimate for the time step will be predicted, which may cause instabilities to occur.

2.5.1 The WAF Scheme in Two Dimensions

The WAF scheme for curvilinear grids was first presented by Speares [94], the intricacies of which are summarised here. In section 2.4.1 it was noted that the WAF scheme employs wave-by-wave limiting. The limiting for shear waves is taken to be a function of the jump in the tangential velocity and not density as is the case for the other waves. Hence, after limiting, the speeds of the combined shear and contact waves will generally be different. The two-dimensional WAF scheme takes this into account by computing the flux as a summation of five, instead of four (for the one-dimensional scheme), distinct regions; the extra region lies between the shear and the contact. The fact that the limited Courant number of the shear may be greater or less than that of the contact, is taken into account when calculating the flux weights.

For non-Cartesian grids, the WAF integration range must be altered so that the scheme remains stable for Courant numbers less than or equal to one. The alteration involves a modification to the integration weights for each region in the Riemann problem wave structure. The weights are functions of the wave Courant numbers. Thus, by redefining the wave Courant numbers that emanate from the $(i+\frac{1}{2},j)$ boundary as

$$\nu^{(k)}_{i+\frac{1}{2},j} = \omega^{(k)}_{i+\frac{1}{2},j}\frac{\Delta t}{\delta_{i+\frac{1}{2},j}} \tag{2.46}$$

the weights are automatically changed. In equation (2.46), $\delta_{i+\frac{1}{2},j}$ represents the *integration length* for the $(i+\frac{1}{2},j)$ boundary. There are a number of choices for calculating the integration lengths. Provided the grid is smooth and has little variation, the distance between the centres of the two cells adjacent to the intercell boundary (illustrated in figure 2.7), yields accurate results, i.e.

$$\delta_{i+\frac{1}{2},j} = \sqrt{(x_{i,j}-x_{i+1,j})^2+(y_{i,j}-y_{i+1,j})^2} \tag{2.47}$$

where $(x_{i,j},y_{i,j})$ are the centre coordinates of cell (i,j).

2.6 Non-Homogeneous Systems (Source Terms)

Flow phenomena cannot always be described by a system of homogeneous hyperbolic PDE's. Such problems can be solved by time operator splitting, which is a similar

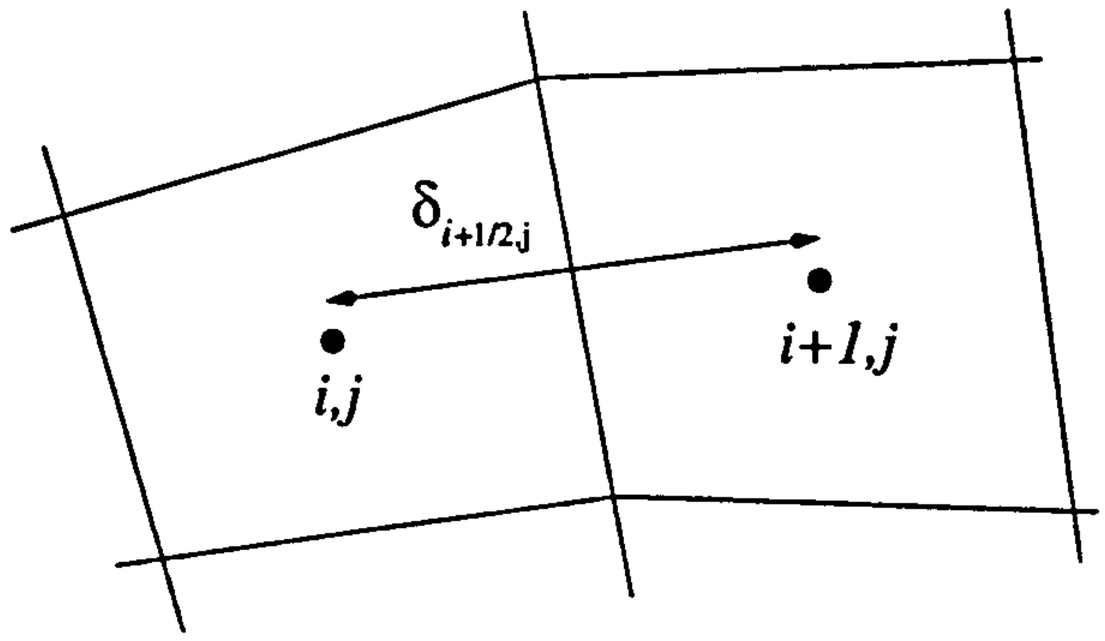


Figure 2.7: An illustration of the integration length $\delta_{i+\frac{1}{2},j}$ at the $i+\frac{1}{2},j$ intercell boundary, for the two-dimensional WAF scheme.

technique to the space splitting that was described in the previous section. The equation set is split into two or more parts and the conserved variables updated by solving each part consecutively. The problems of interest here can generally be arranged so that the equation set can be split into a hyperbolic part and some other part. In one-dimension, the systems of interest have the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}), \tag{2.48}$$

where $\mathbf{S}$ is referred to as the source term vector, which may also be a function of other parameters that are not flow variables, e.g. radial distance for axi-symmetric problems. The solution strategy of time splitting proceeds by first neglecting the source terms in order to update the left hand side of (2.48) equated to zero. The conserved variables are then updated by solving the system $\mathbf{U}_t = \mathbf{S}(\mathbf{U})$, which in the work presented here forms a system of coupled ODE's. A simple ODE solver is often sufficient to update the solution. However, there are situations where the source terms require a more advanced treatment. For example, reacting flow models often involve *stiff* ODE's [37, 60], which necessitate the use of a *stiff* ODE solver, (e.g. see [64]).

An expression for updating the solution $\mathbf{U}^n$ at time t^n, to time $t^{n+1} = t^n + \Delta t$ can be obtained with a Taylor series expansion. If terms involving second and above derivatives are neglected then

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \mathbf{U}_t \tag{2.49}$$

If $\mathbf{U}_t = \mathbf{S}(\mathbf{U})$ is substituted into (2.49) then an update formula for every cell i is given by

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \Delta t\, \mathbf{S}_i(\mathbf{U}^n) \tag{2.50}$$

Note that, the source term may vary with i and may also be a function of the global solution, not just $\mathbf{U}_i^n$, e.g. derivative terms. Equation (2.50) is a first order accurate

method, known as *Euler's Method.* If terms involving third and above derivatives are neglected then the Taylor series expansion yields:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \mathbf{U}_t + \frac{1}{2}\Delta t^2\, \mathbf{U}_{tt} \tag{2.51}$$

$$= \mathbf{U}^n + \Delta t \left[\mathbf{U}^n + \frac{1}{2}\Delta t\, \mathbf{U}_t\right]_t \tag{2.52}$$

If $\mathbf{U}_t = \mathbf{S}(\mathbf{U})$ is substituted into (2.52), then an update formula for every cell i is given by

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \Delta t \left[\mathbf{U}_i^n + \frac{1}{2}\Delta t\, \mathbf{S}_i(\mathbf{U}^n)\right]_t \tag{2.53}$$

This is a second order method, that is referred to as the *Modified Euler Method.* It can be written more conveniently by defining an intermediate state $\mathbf{U}_i^{n+\frac{1}{2}}$ and substituting it into (2.53), i.e.

$$\mathbf{U}_i^{n+\frac{1}{2}} = \mathbf{U}_i^n + \frac{1}{2}\Delta t\, \mathbf{S}_i(\mathbf{U}^n) \tag{2.54}$$

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \Delta t\, \mathbf{S}_i(\mathbf{U}^{n+\frac{1}{2}}) \tag{2.55}$$

The problems involving source terms that are presented in chapter 7, were computed using either the Euler or modified Euler methods. The time step for the ODE update was assumed to be equal to the hyperbolic PDE update. This is a potentially dangerous assumption to make and is reflected on in the final chapter of this thesis.

2.7 Boundary Conditions

Fluid domains may or may not involve impermeable solid boundaries. There cannot be any flow across such boundaries, even though the velocity of the adjacent fluid may be non-zero. Also, it is not possible to represent an infinite extent of physical space within a computational grid, without mathematically transforming the equation model. Thus, a computation must be able to replicate the presence and the absence of solid boundaries at the edges of the computational domain. The types of schemes considered here, utilise the solution of the Riemann problem to obtain numerical flux values. Therefore, the initial conditions of the boundary Riemann problems must be such that the solution is consistent with the type of boundary. Before every integration step, a number of rows of fictitious boundary cells, that are situated around the periphery of the computational grid, are primed with solution data. The number of rows of boundary cells is dependent on the stencil of the scheme, which is related to the order of accuracy of the scheme. At every intercell boundary, the WAF scheme utilises both the local and neighbouring Riemann

problem solutions to determine the numerical flux. Therefore, the second order WAF scheme requires the solution data for two rows of boundary cells.

The only boundary conditions of interest here, are *transmissive* and *reflective* conditions, which correspond to un-hindered flow and no flow across the boundary respectively. For a one-dimensional grid containing m cells with two extra boundary cells at each end, the transmissive boundary conditions at the $m + \frac{1}{2}$ boundary for the Euler equations are given by:

$$\begin{array}{llllll} \rho_{m+1} = \rho_m, & u_{m+1} = u_m, & p_{m+1} = p_m \\ \rho_{m+2} = \rho_{m-1}, & u_{m+2} = u_{m-1}, & p_{m+2} = p_{m-1} \end{array} \tag{2.56}$$

For a similar one-dimensional grid moving with velocity V, the reflective boundary conditions at the $m + \frac{1}{2}$ boundary for the Euler equations are given by:

$$\begin{array}{llllll} \rho_{m+1} = \rho_m, & u_{m+1} = 2V - u_m, & p_{m+1} = p_m \\ \rho_{m+2} = \rho_{m-1}, & u_{m+2} = 2V - u_{m-1}, & p_{m+2} = p_{m-1} \end{array} \tag{2.57}$$

The internal cell solutions can then be updated using a suitable moving grid scheme, such as the WAF scheme described in section 2.4.2. Note that, when $V = 0$ the above boundary conditions apply to fixed reflective boundaries. Similar transmissive and reflective boundary conditions can be imposed at the intercell boundary between cells 0 and 1.

For two-dimensional Cartesian grids, the shear component of velocity v, is taken from that of the relevant internal cell, i.e. for both transmissive and reflective (fixed and moving) boundaries,

$$v_{m+1} = v_m \text{ and } v_{m+2} = v_{m-1} \tag{2.58}$$

For curvilinear grids, the relevant boundary conditions are applied normal to the boundary. Hence, in (2.56), (2.57) and (2.58), u and v are replaced by the normal and tangential components of velocity respectively. The normal and tangential solutions in the boundary cells are then rotated back to the Cartesian coordinate system. Thus, when the solution data is rotated to the boundary, during the subsequent integration step, it forms a symmetrical Riemann problem, whose solution automatically enforces the desired flow condition.

Chapter 3

Adaptive Mesh Refinement

This chapter describes an adaptive grid refinement strategy for structured computational grids, known as Adaptive Mesh Refinement AMR. AMR has been successfully applied to a variety of fluid dynamics problems [3, 23, 90]. The computational effort required to solve numerical fluid dynamics problems, is focussed by varying the computational grid resolution in specific regions of the flow field. Regions of the flow domain that are likely to incur large numerical errors can be singled out, by some form of refinement criteria, as requiring increased grid resolution. In order to ensure that all the dynamic flow features in a transient problem receive the necessary level of grid resolution, AMR algorithms automatically update the grid structure during computations. Compared to regular mesh solvers, effective AMR algorithms are capable of providing large savings in processing time and memory storage, without any distinguishable difference in the accuracy of the solution.

The following sections detail the basic AMR algorithm for Cartesian grid structures, along with some original variations, which improve its overall performance. It is deemed to be important that the reader is familiar with both the AMR grid structure and the coordination of the various processes, before any particular aspect of the algorithm is described in detail. A description of the AMR grid structure is given in section 3.1, followed by an outline of the AMR approach in section 3.2. Subsequent sections, with the exception of section 3.6, describe in detail the components of the AMR algorithm. Section 3.6 summaries the various aspects of the AMR approach presented here.

3.1 AMR Grid Structure

AMR employs sets of structured computational mesh patches within a hierarchical grid system. A single grid level consists of one or more rectangular mesh patches of

equivalent resolution. At the bottom of the hierarchy is a base grid, which covers the whole flow domain with coarse cell meshes. Further grid levels[1], with increasing grid resolution, are added upon the base grid in accordance with the refinement criteria. During a computation the number, size and location of the mesh patches is automatically updated by the AMR algorithm. Generally, only a small fraction of the flow domain is covered by the finer grid levels. The efficiency of the AMR approach, as with all adaptive grid techniques, is inversely proportional to the extent of the fine level coverage. Hence, AMR is most effective when only a small amount of the computational domain requires increased grid resolution.

Extra rows of boundary cells around every mesh are primed with data prior to integrating the meshes. These cells, referred to as ghost cells, enable the mesh integration to proceed completely independently from the complicated workings of the adaption process. In order to advance a cell's solution to the next time level, an n^{th} order scheme requires a stencil of solution data of at least n adjacent cells in all computational directions. Hence, before every integration involving a second order scheme, an extra two rows of cells around every mesh must be primed with solution data. A search to find the origins of this data is done whenever the grid level's structure changes[2]. This is a relatively expensive part of the adaption process, because it requires a check to determine whether or not a ghost cell is covered by an abutting mesh. The parental mesh structure of the algorithm described here, significantly reduces the amount of checking that is required.

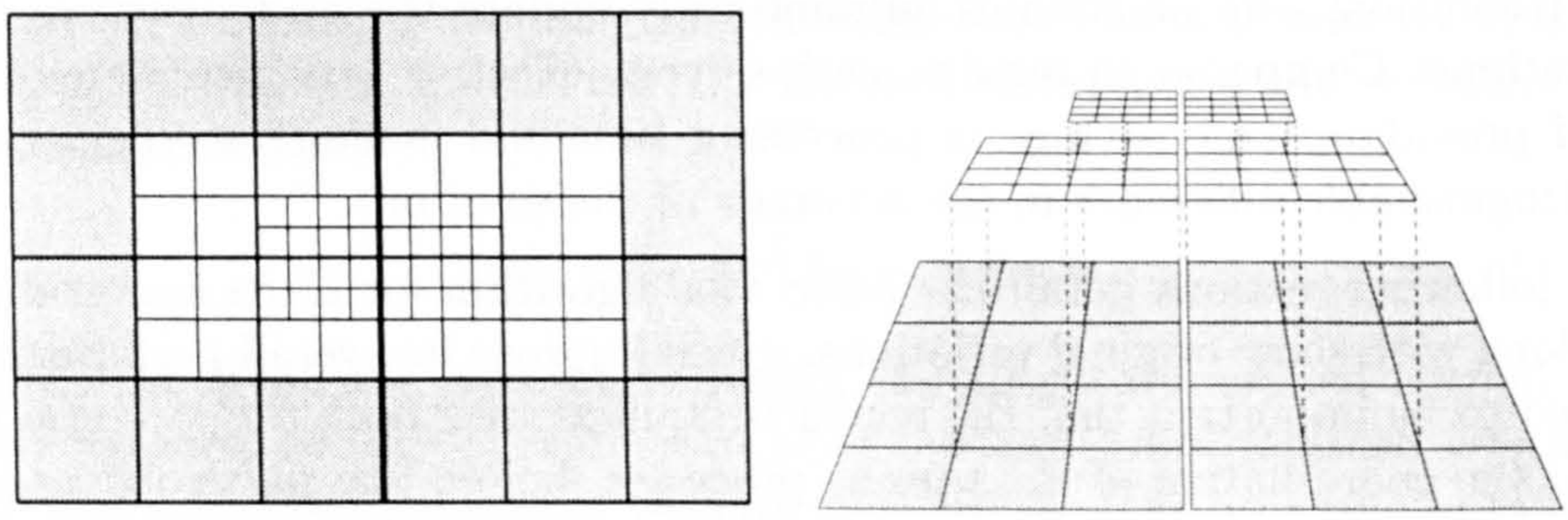


Figure 3.1: A three level grid hierarchy.

Other AMR algorithms [3, 9, 79] allow the finer mesh patches to cross underlying coarse mesh boundaries. Here we impose a far more stringent criterion for the nesting of mesh patches. Every refined mesh patch has only a single parent mesh, i.e. it is defined in terms of the cells of the underlying coarse mesh only. Furthermore, meshes of the same grid level do not overlap. The efficiency of algorithms that allow meshes to overlap are adversely affected by the extra, unnecessary, storage and integrations

[1] An upper limit on the number of grid levels is imposed by the user.

[2] A grid level's structure only changes when it reaches the same point in time as the underlying coarser grid solution, i.e. every few fine level time steps.

that result. Any coarse mesh patch may, or may not, have overlying refined child mesh patches. This *proper* parental nesting of meshes not only ensures that regions of the same grid are not integrated by more than one mesh, but also greatly speeds up the acquisition of boundary cell data. A simplistic three-tier grid hierarchy is shown in figure 3.1; there are two abutting meshes on every grid level. Any two meshes of the same grid level share at least one common ancestor. Hence, the boundary cell data is obtained by looking down the tree structure until a common ancestor is found, and then tracing back up to the relevant mesh cells. The transfer of data between various parts of the memory storage is described in section 3.4.

3.2 Outline of the AMR Approach

AMR is unique among grid adaption techniques, in that it employs temporal as well as spatial refinement. Other adaptive grid techniques advance all the grid cells with the same time step, which is often dictated by the stability criteria of the finest cells. Temporal refinement allows the coarser grid levels to advance with much larger time steps than the finer levels. In order to impose the temporal refinement, the AMR algorithm calculates the solutions of the coarse grids, even where fine level coverage exists. Updating the coarse grid cells requires little computational effort relative to that of updating the finest grid cells. To date, AMR algorithms have assumed the temporal refinement to be equal to the spatial refinement. This is likely to be problematic. For example, consider a four level grid structure in which the cell resolutions of the three finer levels are increased by a factor of five. The resulting time step of the finest grid level would be defined as $1/125^{th}$ of the coarsest time step. Unless an overly restrictive stability condition is used, the maximum wave speed of the finer grid levels could easily be underestimated. Which could lead to numerical instability at some point in the solution integration of the finer grids. Conversely, there may be situations where the solution is needlessly integrated by too many small time steps, e.g. when the maximum wave speed decelerates during a coarse time step. Such a situation will result in an unnecessary increase in the integration costs and the numerical (diffusion) error associated with the solution. The AMR approach presented here determines the time step for every grid level, from the maximum characteristic speed in its solution, immediately before it is integrated. This results in an unpredictable number of fine grid time steps in every coarse grid time step, (see figures 3.2 and 3.14). Although, over a whole computation, the ratio of the number fine to coarse time steps is approximately equal to the spatial refinement factor.

To the reader of the adaptive grid literature, the philosophy behind when or how often any adaptive grid structure should be updated, could be construed as an open issue. Several AMR algorithms, notably [3] and [9], do not update the grid

structure as often as is possible. By far, the most expensive part of any computation, is the integration of the flow field solution. Any viable adaptive grid algorithm, should spend only a tiny proportion of the overall computing time, generating the grid structure and transferring data within it. By not changing the grid structure whenever possible, adaptive grid algorithms appear to increase their efficiency, i.e. they spend less time adapting the grid structure compared with integrating the grid solution. However, the finer grid levels need to extend sufficiently far beyond the immediate vicinity of the dynamic flow features requiring refinement, in order to ensure adequate coverage during the ensuing time steps. Hence, the longer a grid structure remains unchanged, the greater the proportion of the flow domain that must be covered by fine grid cells. The increased number of cells, results in an increased number of computationally expensive cell integrations. Therefore, maximum efficiency is achieved by adapting the grid structure when and wherever possible, thereby limiting the total amount of processing time spent integrating the solution; the cost associated with the extra grid adaption is negligible in comparison. The philosophy here is in agreement with Quirk's [79], in that the AMR algorithm presented here does update the AMR grid structure whenever possible, i.e. every time a fine grid solution reaches the same time level as the underlying coarse grid. Figure 3.2 depicts a typical time evolution of a one-dimensional grid structure, for the movement of a discontinuous feature requiring refinement. Note that, only a small region of space around the feature is refined for every coarse time step. If the grid was adapted less frequently, then a larger region of space-time would be covered by the finer grid cells.

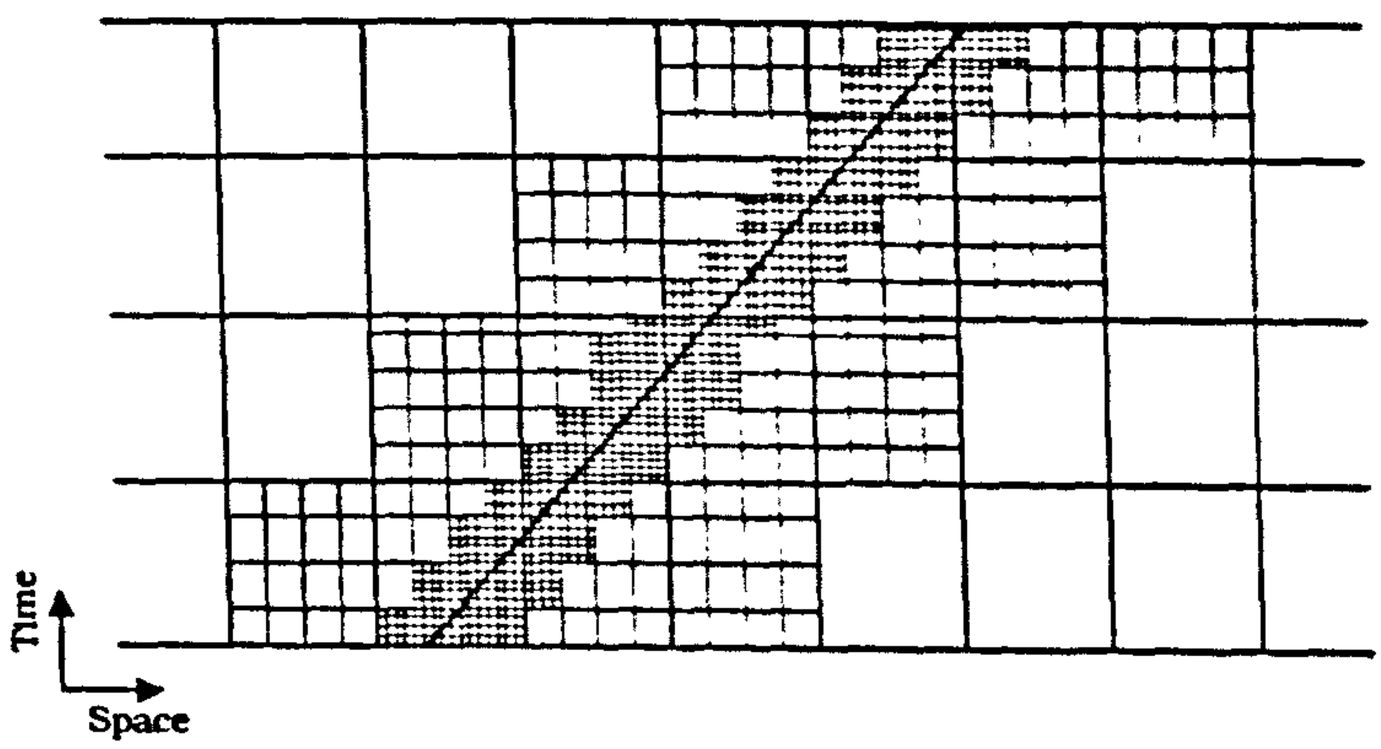


Figure 3.2: Time evolution of the AMR grid structure containing a dynamic discontinuous flow feature.

3.2.1 Coordination of the AMR algorithm

After defining the initial grid structure and priming it with solution data, the coordination of the AMR process can be regarded as a recursive sequence of procedures, that operate on every grid level until the desired point in time is reached. Figure 3.3 depicts the FORTRAN like pseudo code for the **sequence** procedure, which continually updates the AMR grid structure and its solution. The sequence of procedures are repeated until all the grid levels reach the desired output time, T_0. T_{G_L} is the time level of grid level G_L. The procedure **new_AMR_structure** regenerates the grid structure for all the fine grid levels above G_L. Procedure **time_step** estimates the largest stable time step, Δt, for grid level G_L. Before procedure **integrate_AMR_grid** integrates all the mesh patches belonging to grid level G_L, procedure **set_AMR_ghost_cells** primes all the ghost cells of grid level $G_L$with solution data. Note that, the ghost cells of the coarse grid levels (G_L=1 to $G_{Lmax}-1$) are updated during the regeneration of the grid structure, i.e. **set_AMR_ghost_cells** is called within **new_AMR_structure**. Embedded within the **integrate_AMR_grid** procedure, is the mechanism for detecting which cells require refinement. The information detailing which cells require refinement is later transferred to the **new_AMR_structure** procedure. The **update_coarse_grid_solution** procedure updates the coarse grid solution with the overlying fine grid solution, whenever the fine grid reaches the same point in time as the coarse grid.

```
PROCEDURE sequence(T_0)
    G_L=1
    DO WHILE (T_{G_Lmax}<T_0)
        IF (G_L<G_Lmax) new_AMR_structure(G_L)
        DO WHILE (T_{G_L}<T_{G_L-1})
            time_step(G_L, Δt)
            T_{G_L}=T_{G_L}+Δt
            IF (G_L=G_Lmax) set_AMR_ghost_cells(G_L)
            integrate_AMR_grid(G_L, Δt)
            IF (G_L<G_Lmax) G_L=G_L+1
        END DO
        DO WHILE ((T_{G_L}=T_{G_L-1}) AND (G_L>1))
            G_L=G_L-1
            IF (G_L≥1) update_coarse_grid_solution(G_L)
        END DO
    END DO
END PROCEDURE
```

Figure 3.3: The pseudo code for the AMR **sequence** procedure.

A typical sequence of AMR procedure calls, for a single base grid time step of a three level grid structure, is shown in figure 3.4. The sequence starts by defining the new grid structure. All the grids are then advanced[3] by their respective time steps. The finest grid level is continually updated, until it coincides with the same point in time as G_L=2. (Note that the number of fine grid time steps, within the coarse time steps, varies according to the stability criteria.) At this point the G_L=2 solution is updated by the overlying G_L=3 solution. The grid structure of G_L=3 is regenerated before the solution of grid level G_L=2 is advanced in time. The cycle

```
G_L=1                    G_L=2                    G_L=3

new_AMR_structure(1)
time_step(1,Δt1)
integrate_AMR_grid(1,Δt1)
                         time_step(2,Δt2)
                         integrate_AMR_grid(2,Δt2)
                                                  time_step(3,Δt3)
                                                  set_AMR_ghost_cells(3)
                                                  integrate_AMR_grid(3,Δt3)
                                                  time_step(3,Δt3)
                                                  set_AMR_ghost_cells(3)
                                                  integrate_AMR_grid(3,Δt3)
                                                  time_step(3,Δt3)
                                                  set_AMR_ghost_cells(3)
                                                  integrate_AMR_grid(3,Δt3)
                         update_coarse_grid_solution(2)
                         new_AMR_structure(2)
                         time_step(2,Δt2)
                         integrate_AMR_grid(2,Δt2)
                                                  time_step(3,Δt3)
                                                  set_AMR_ghost_cells(3)
                                                  integrate_AMR_grid(3,Δt3)
                                                  time_step(3,Δt3)
                                                  set_AMR_ghost_cells(3)
                                                  integrate_AMR_grid(3,Δt3)
                         update_coarse_grid_solution(2)
update_coarse_grid_solution(1)
```

Figure 3.4: A typical sequence of procedure calls for a single base grid time step.

[3]The coarser grid solutions are needed in order to update the fine-coarse boundary ghost cells, prior to integrating the finer grid levels, and to detect new, previously unrefined, regions that require refinement.

of G_L=3 integrations is then repeated until, again, it coincides with G_L=2. At this point, the time levels of all three grid levels' solutions coincide. The solution of grid level G_L=2 is first updated, by that of G_L=3, before it provides the solution data to update G_L=1. The whole sequence then starts again, by first defining the new grid structure, before advancing the grids' solutions. For any given fluid dynamics problem this sequence of processes is repeated until the desired point in the evolution of the flow solution is reached. Note that, because of the variable time step size, the sequence of procedure calls is likely to change from one coarse grid time step to another.

3.3 Generation of AMR Grid Structure

For optimum efficiency, the AMR grid structure is regenerated whenever a fine grid coincides with the same point in time as the underlying coarse grid. Figure 3.5 depicts the FORTRAN like pseudo code for the **new_AMR_structure** procedure, which sets up the new mesh structure for grid levels G_N+1 to G_{Lmax}, where grid level G_N is the finest grid that does not coincide (in time) with its underlying coarse grid. Before the new grid structure is generated the procedure **copy_AMR_structure** is called upon to store the existing grid structure and solution. Then, for every grid level from G_N to $G_{Lmax}-1$, the ghost cells are primed with data before the **new_grid_structure** procedure generates the overlying meshes and the **transfer_solution** procedure transfers the existing solution to the new grid structure.

```
PROCEDURE new_AMR_structure(G_N)

    copy_AMR_structure(G_N)
    DO FOR G_L=G_N TO G_Lmax-1
        set_AMR_ghost_cells(G_L)
        new_grid_structure(G_L+1)
        transfer_solution(G_L+1)
    END DO

END PROCEDURE
```

Figure 3.5: The pseudo code for the AMR **new_AMR_structure** procedure.

Initially, every mesh of the coarse base grid is examined in order to identify regions requiring greater grid resolution. This process, generally referred to as the flagging process, involves a switching criterion. For example, if the gradient of the solution density across two neighbouring cells exceeds some tolerance then both of the cells are 'flagged' for refinement. A clustering (grouping) process then covers the flagged cells with finer mesh patches. In order to ensure that the identified

disturbances remain within the finer mesh patches during the following time steps, the flagging process also adds extra (safety) cells around all the flagged cells. Thus, the clustering process actually groups both the flagged and the extra safety cells into refined patches.

3.3.1 Flagging for Refinement

The flagging process is a means of highlighting which cells require further refinement. Generally, the hyperbolic, non-linear nature of the system of equations will contain small scale disturbances which are likely to develop with time to form large scale features. Success, for all grid adaption techniques, is dependent on their ability to detect small disturbances, so that they can be adequately modelled as they evolve. The flagging process consists of two stages: the detection mechanism and the addition of extra safety cells. An extra one or two rows of safety cells around all the flagged cells, ensures adequate refinement coverage, whilst keeping the number of unnecessary refined cells and their subsequent integrations to a minimum. Numerical experiments involving the Euler equations suggest that only a single row of extra safety cells is required. However, when modelling flows with rapidly changing velocities, such as reactive flows, it may be prudent to flag two extra rows of safety cells. The flagging process must also include the ghost cells (mesh boundary cells), so that disturbances can be detected across common mesh boundaries. Consider figure 3.6 where AA' represents the common mesh boundary between two meshes, referred to as the left-hand (LH) mesh and the right-hand (RH) mesh. Suppose there

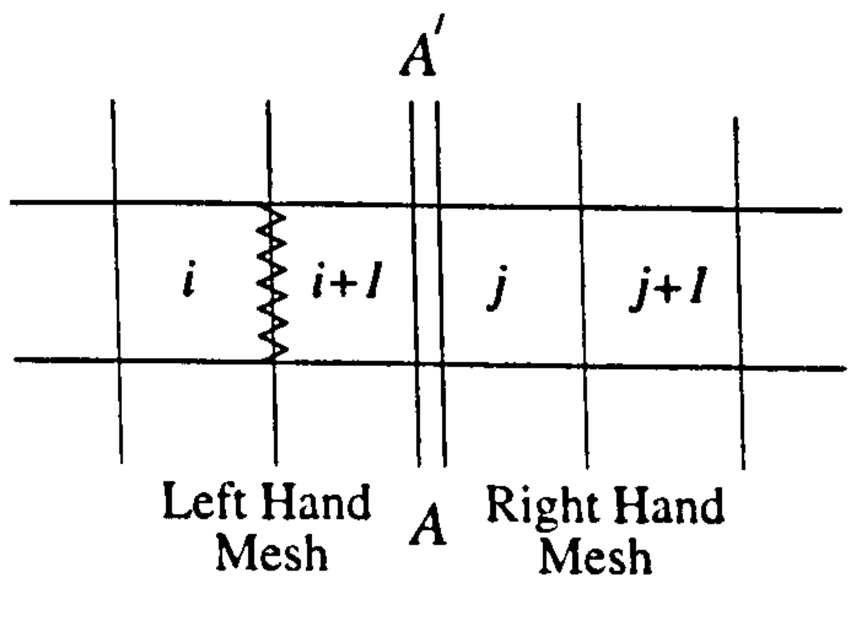


Figure 3.6: A common boundary (AA') between two meshes.

exists a disturbance between the cells i and $i+1$ of the LH mesh, such that i and $i+1$ are both flagged for refinement. The solution values of cells i and $i+1$ are not only stored in the LH mesh, but are also stored in the corresponding ghost cells of the RH mesh. So by flagging and adding the safety cells to the ghost cells belonging to the RH mesh, the internal mesh cells j and $j+1$ are also flagged, thereby ensuring the continued refinement of the disturbance across the mesh boundary. Note that,

this process requires the number of rows of ghost cells to be greater or equal to the number of rows of safety cells.[4]

Detection Mechanism

There are numerous ways of detecting when and where refinement should take place. Often, the user has some prior physical knowledge of the problem, which he or she can utilise in order to get the most benefit from an adaptive gridding computation. For example, it may be desirable to refine a particular region of the flow domain after a certain time only, (a very simple criterion to implement, involving only a couple of 'IF' statements), and all discontinuous flow features throughout the computation (requires a mechanism to detect only the designated features). There is no limit to the number of different detection criteria that can be used. All flow feature detection criteria are hinged on whether or not some specified data condition exceeds some pre-defined flagging tolerance. If this tolerance is too large then essential flow features may be poorly resolved. Conversely, if the tolerance is too small, regions of the flow field which do not introduce significant errors into the solution will be unnecessarily refined. This would incur extra computing costs, in the form of both processing time and memory storage.

Various ideologies for detecting features which require greater grid resolution have been presented in the literature. Berger *et al.* [9] used Richardson extrapolation to estimate the local truncation error, thereby indicating regions requiring refinement. Even though Richardson extrapolation is only valid for smooth data, Berger has demonstrated that it is effective for highlighting discontinuous features. Bell *et al.* [3] improved the technique in [9] by supplementing it with a spatial error estimator. Quirk [79] and Fischer [33] have had a good deal of success with very simple density difference criteria. Fischer included a further test, based on the increased difference between the pressure and density changes for non-isentropic features.

The idea of using the internal structure of the Riemann problem as a detection mechanism was first advocated, but not implemented, by Speares & Toro [97]. The Riemann problem solution is an exact local solution to the homogeneous system of equations. Thus, Riemann problem flagging is intuitively effective at highlighting local flow features. Particular features, such as shock waves, contact waves and slip surfaces, are singularly identified by considering the jumps across the associated waves within the solutions of local Riemann problems. For example, a contact wave can be detected by comparing the jumps in density across the contact waves in the local Riemann problems, with a tolerance. Generally, choosing a suitable flagging

[4]Most shock capturing schemes are at least second order accurate and therefore require a minimum of two rows of ghost cells, whilst it is unlikely that any more than two rows of safety cells will be required.

tolerance such that only the essential, error prone, discontinuities are identified is not too difficult. Indeed, for the computed flows presented here, the amount of refinement and hence the quality of the solution, is very similar for a fairly broad range of flagging tolerances.

All the various detection processes that have appeared in the literature, examine the coarse grid solution in order to create a refined grid level. It is possible that some 'delicate' features, such as Kelvin-Helmholtz instabilities, which are clearly represented on a finer grid level will not appear on the underlying coarse grid levels. This would result in the coarse grid cells not being flagged, and therefore refined, during the next updating of the fine grid structure. This situation is avoided by flagging all the grid levels upto and including the finest level. During the routine which updates the coarse grid solution with the overlying fine grid solution, the fine grid flagging information is also transfered to the coarse grid levels. Thus, a flag on the finest level, will be transfered to all the underlying cells of all the coarser grid levels.

Greater reliability and flexibility in targeting particular flow features for refinement, has been achieved by employing a Riemann problem based flagging procedure on all the available grid levels.

3.3.2 Clustering

After the cells have been singled out as requiring refinement, they must be grouped (clustered) into mesh patches. Both the size and the number of the resulting meshes are fundamental to the efficiency of the AMR algorithm. In order to achieve maximum efficiency, all the flagged cells must be covered by as few meshes as possible, without covering too many unflagged cells. The clustering process recursively divides all the coarse grid mesh patches until the refine condition for every one is satisfied. The refine condition for a particular mesh is not satisfied if either of the mesh side lengths is too large or if the proportion of flagged cells to the total number of cells is less than some prescribed tolerance, (Quirk suggests 0.6 as good choice [79]). Limiting the number of cells also limits the amount of memory storage associated with any single mesh. The limit can be chosen such that all the data that is needed to interpolate the solution of a mesh, can be transferred to, and held within, the local memory (RAM) of the computer. Hence, the computational efficiency is maximised because the computer does not need to access data from the hard disk at any point during the mesh integration. Most previous AMR algorithms use a clustering technique whereby, a mesh is simply split, in the middle of the longest side length, into two separate meshes. These clustering processes also tend to ignore the coarse mesh boundaries. This produces some fine meshes that cover cells of more than one underlying coarse mesh. This results in a grid structure that does not facilitate

the transfer of data within it. The clustering process usually creates a number of 'small' meshes (meshes that cover only a few coarse cells). The ratio of the number of ghost cells to the total number of cells is inversely proportional to the mesh size. Hence, because the ghost cells need to be primed with solution data and are partly integrated (dependent upon the integration scheme used), the extra costs associated with integrating a small mesh is relatively large. A mesh merging process is often included into AMR algorithms [9, 79]. Small meshes are merged into abutting meshes, in an attempt to improve the overall efficiency of the computation. The merging process causes meshes of the same grid level to overlap. Bell *et al.* [3] removed the merging process from their AMR algorithm, in order to reduce the amount of data storage and eliminate any unnecessary duplicate cell integrations.

It is worthwhile examining more closely the costs associated with the small meshes. Consider figure 3.7 which depicts two alternative mesh arrangements for each group of three, five and seven flagged coarse cells (marked by crosses). The three, five and seven cell groups are the three smallest groupings, that cannot be covered by a single mesh without covering one or more unflagged cells. Thus, by covering the cells with two meshes (B, D and F), it is possible to assess the costs, in terms of memory storage and processing time, relative to the single meshes (A, C and E). For a mesh with coarse cell dimensions x, y, the number of internal cells

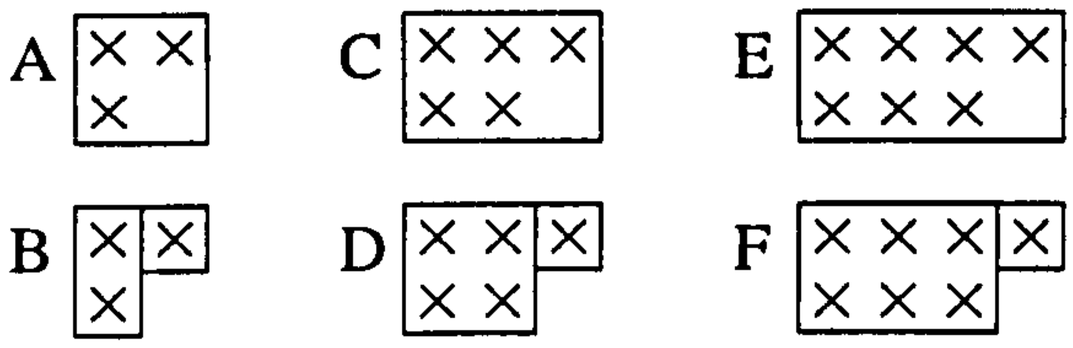


Figure 3.7: Alternative ways of clustering three (A and B), five (C and D) and seven (E and F flagged cells.

and the number of ghost cells associated with it are given by

$$x.y.\Gamma^2$$

and

$$2n_g(\Gamma(x+y)+2n_g)$$

respectively, where Γ is the refinement factor and n_g is the number of rows of ghost cells around the mesh. The total number of mesh cells (internal plus ghost cells) is given by

$$(x\Gamma+2n_g).(y\Gamma+2n_g)$$

and the number of resulting integrations, using a the operator split WAF scheme, is given by

$$2x\Gamma(y\Gamma+n_g).$$

The relative costs of the single mesh to the alternative pair of meshes, associated with each group of flagged cells, can be gleaned from the graphs shown in figures 3.8, 3.9 and 3.10. The graphs depict the ratios of the number of ghost cells, the total number of cells and the number of resulting cell integrations for a range of refinement factors, $2 \leq \Gamma \leq 6$, (n_g is taken to be two). Obviously, the ratio of the number of internal mesh cells does not vary with the refinement factor; 4/3, 6/5 and 8/7 for the 3, 5 and 7 flagged cell groups respectively.

The graph in figure 3.8 indicates that there are fewer ghost cells associated with the single meshes than with the mesh pairs, and that the difference is inversely proportional to both the refinement factor and the number of grouped cells, i.e. E/F > C/D > A/B. However, even though the total number of cells has a similar trend to the number of ghost cells, figure 3.9, the differences are less significant. In contrast the graph in figure 3.10 indicates that, compared to the single mesh, there is a reduction in the number of cell integrations for the pair of meshes. The difference is proportional to the refinement factor and inversely proportional to the number of grouped cells, i.e. E/F < C/D < A/B. Section 3.5.1 describes a pointer system that increases the amount of memory storage required by the ghost cells, but which greatly speeds up the acquisition of solution data. Therefore, it is possible to conclude that by splitting an already small mesh into two smaller meshes, one of which is the smallest possible (a single coarse cell), the memory requirements will increase but the processing time will decrease. However, these differences are not likely to be significant, since the number of cells in the small meshes are only a small fraction of the total number of cells in the whole grid structure. Hence, the small mesh situation does not have a great effect on the efficiency of AMR. Indeed, it does not appear to be prudent to try and circumvent it, especially if in doing so the AMR algorithm is further complicated and the result is greater storage requirements and processing times. Moreover, because small meshes do not require significant effort, a stringent refine condition can be used in order to further reduce the number of unnecessary integrations. Numerical experiments have revealed that for typical computations, the overall algorithm is most efficient when the refine condition tolerance is set at approximately 0.95. If a more involved operator splitting or finite volume scheme is used, like those suggested in [100] and [13] respectively, then the optimum refine condition tolerance would be even greater.

Bell *et al.* [3] proposed a new clustering technique for creating more efficient meshes. Their technique used a combination of signatures and edge detection, both of which are common techniques within the computer vision and pattern recognition fields. Every mesh is taken in turn and the signature in each row and column is taken to be the sum of the flagged cells. An edge is detected wherever the signature or its derivative is zero. Each mesh is then split along the most prominent edge; zero signature or the greatest second derivative of the signature. If an edge is not detected and a refine condition tolerance of 0.50 is not satisfied, then it is conventionally split

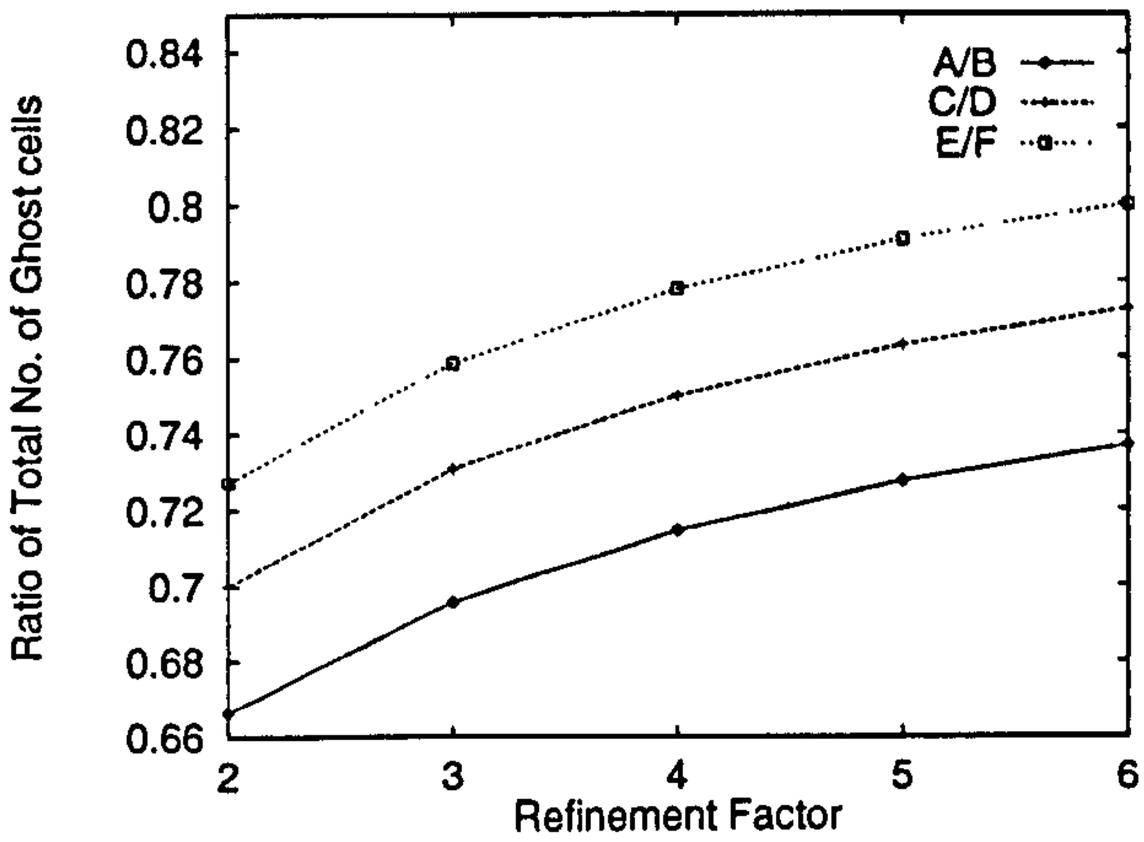


Figure 3.8: The ratios of the number of ghost cells in the single meshes (A, C and E) to those in the mesh pairs (B, D and F).

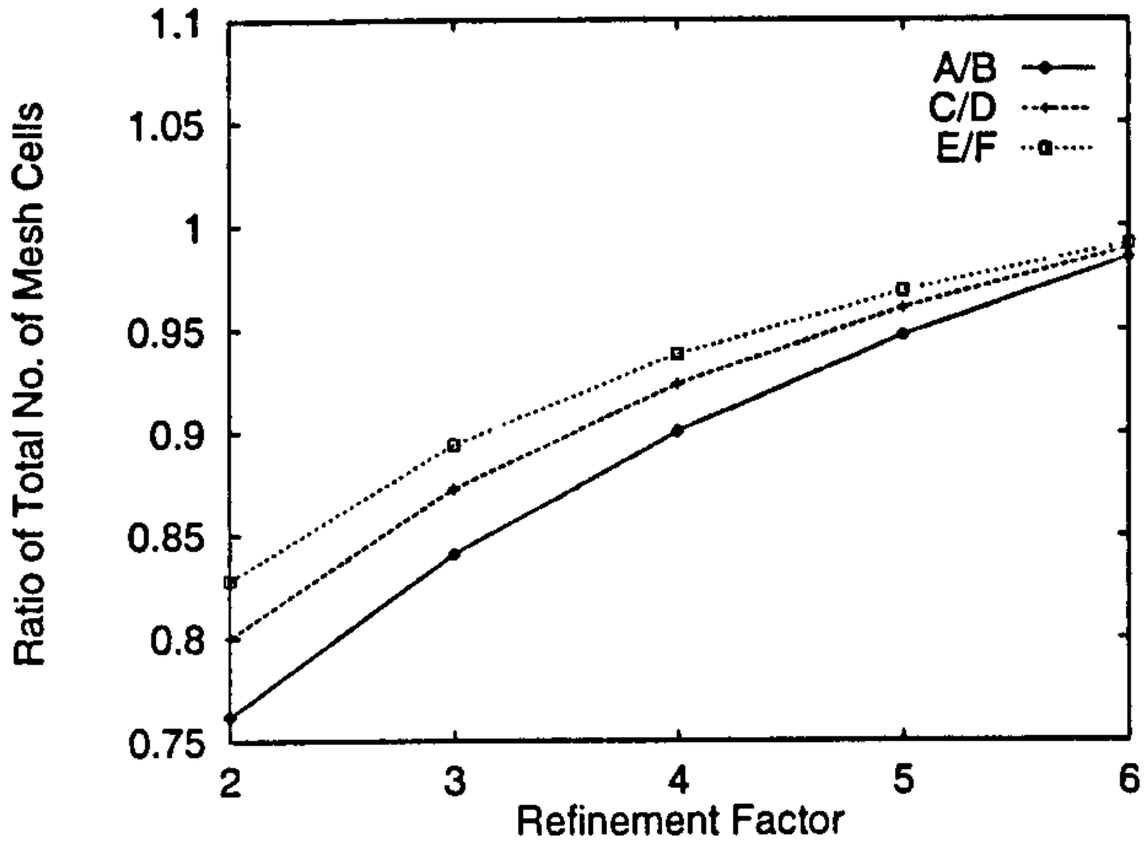


Figure 3.9: The ratios of the total number of mesh cells in the single meshes (A, C and E) to those in the mesh pairs (B, D and F).

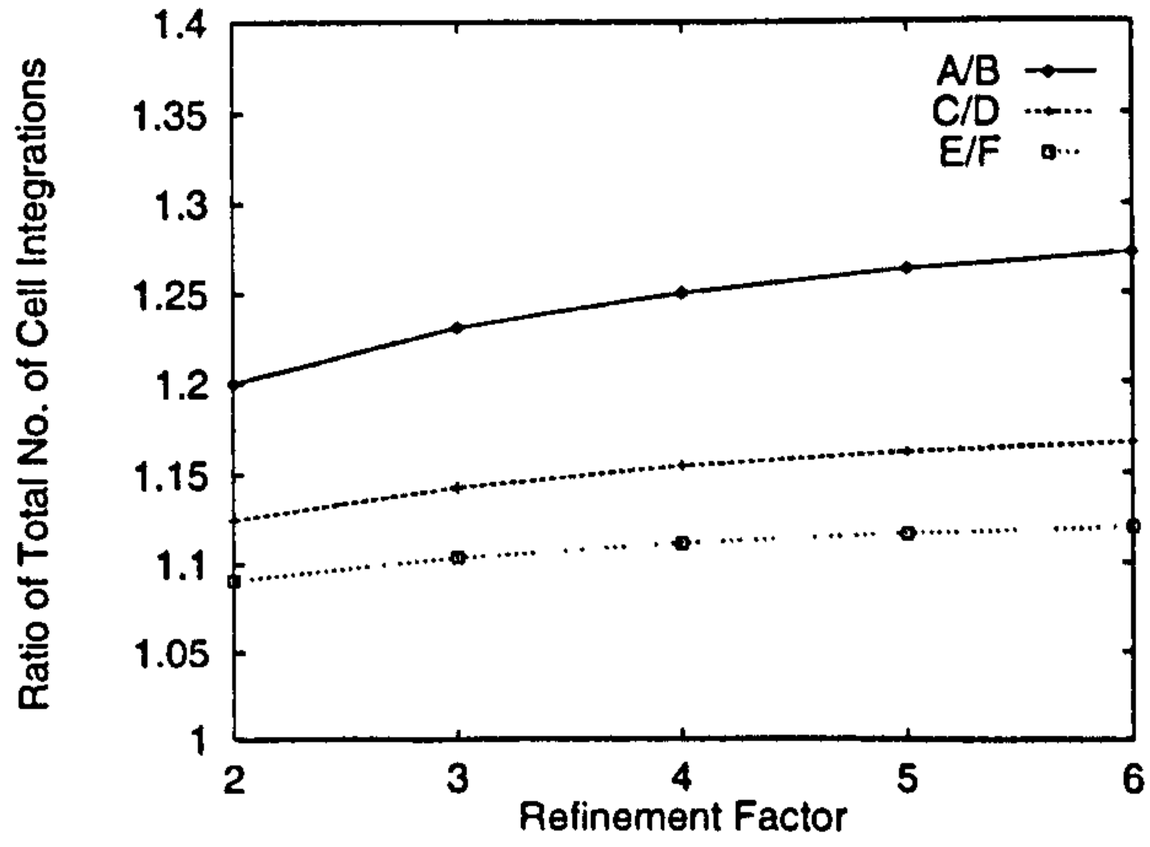


Figure 3.10: The ratios of the number of integrations for the single meshes (A, C and E) to those in the mesh pairs (B, D and F).

in the middle of the longest side length.

Whilst this method certainly improves the overall performance of AMR computations, the author has not managed to achieve the 20 percent reduction in the processing time that was claimed in [3]. There are two possible reasons why the same reduction was not achieved. Firstly, [3] does not indicate which test problem was used to assess the performance. Secondly, the overall AMR process described here may process small meshes more efficiently than the one described in [3]. The second reason is a more probable explanation. A drawback of the Bell *et al.* clustering technique is the fact that there is no account of how the flagged cells are arranged in any particular strip of cells. For example, consider figure 3.11, which depicts a single mesh (A), the resulting three meshes after Bell *et al.* clustering (B) and the ideal result of clustering (C).

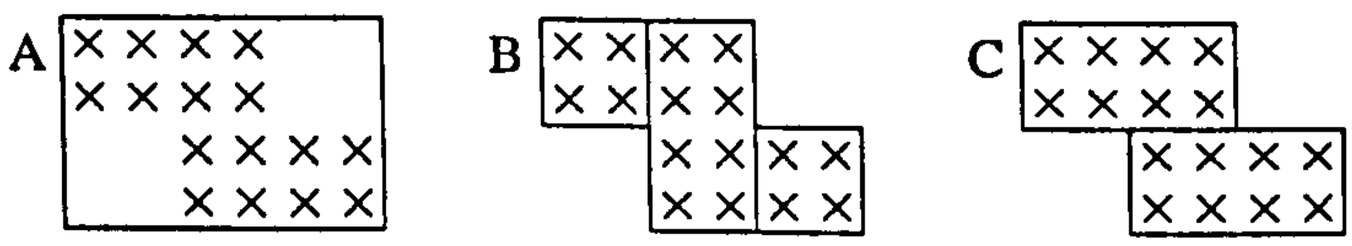


Figure 3.11: A group of flagged cells and two alternative clustered mesh structures.

In trying to develop a clustering technique that could overcome this drawback, a number of different clustering parameters were evolved, three of which are described below. As with the Bell *et al.* clustering technique described above, if the refine condition is not satisfied, then these parameters are applied to all the boundaries between every pair of neighbouring cell strips. The mesh is then split into two meshes along the boundary with the lowest parameter value. The parameters use various combinations of the number of adjacent pairs of flagged cells, one on each side of the $j + \frac{1}{2}$ boundary, $n_{j+\frac{1}{2}}$, the number of flagged cells in row j, σ_j, and the coordinates of the i^{th} flagged cell in row j, i_j. The first parameter, A, computes the ratio of $n_{j+\frac{1}{2}}$ to the number of non-adjacent flagged cells, i.e.

$$\frac{n_{j+\frac{1}{2}}}{\sigma_j + \sigma_{j+1} - 2n_{j+\frac{1}{2}}}$$

The second parameter, B, computes the ratio of $n_{j+\frac{1}{2}}$ to the extent of flagged cells in both the j and $j + 1$ rows, i.e.

$$\frac{n_{j+\frac{1}{2}}}{\mathrm{MAX}(i_j, i_{j+1}) - \mathrm{MIN}(i_j, i_{j+1}) + 1}$$

The third parameter, C, computes the ratio of the total number of flagged cells to the total number of cells within the extent of the flagged cells, in both the j and $j + 1$ rows, i.e.

$$\frac{\sigma_j + \sigma_{j+1}}{2\left(\mathrm{MAX}(i_j, i_{j+1}) - \mathrm{MIN}(i_j, i_{j+1}) + 1\right)}$$

In order to assess the performance of these parameters, consider figure 3.12 which contains three neighbouring rows of coarse cells, some of which are flagged (crosses) for refinement. This example has the flagged cells arranged so as to highlight the drawbacks of each of the first two parameter(s) relative to the proceeding one(s). The number of flagged cells in each row is given on the left-hand side of the mesh and the three parameter values are given of the right-hand side. If the mesh had

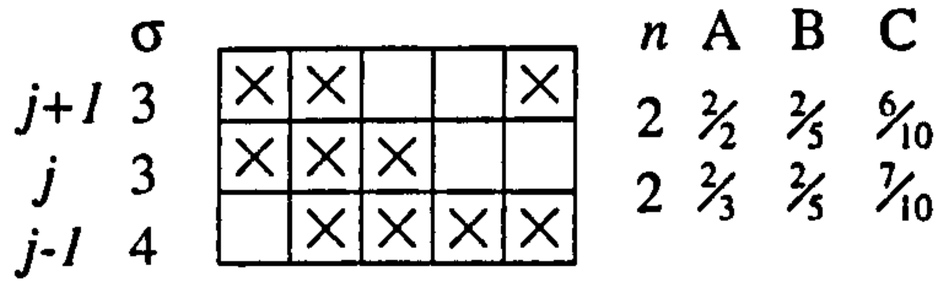


Figure 3.12: Three neighbouring rows of coarse cells.

to be divided between either j row boundary, then clearly, the $j + \frac{1}{2}$ boundary would be better. Parameter A yields a lower value for the $j - \frac{1}{2}$ boundary, and therefore, incorrectly indicates the line of division. Whereas, parameter B is unable to distinguish between the two boundaries. Parameter C yields a lower value for the $j + \frac{1}{2}$ boundary, and therefore, correctly indicates the correct line of division.

The philosophy behind parameter C could be extended to the whole mesh, not just to pairs of rows and columns, i.e. to split a mesh in such a way as the resulting two meshes are the most efficient possible. Time restraints have prevented the implementation and assessment of this extension.

3.4 Data Transfer

The term data transfer refers to the process of moving the various types of data from one part of the data structure to another. There are three situations that require the solution data within the grid structure to be transferred; the transfer of the solution data from the old to the newly generated grid structure, the updating of the coarse grid solution with the overlying fine grid solution and the updating of the ghost cells' solutions. The efficiency of the transfer processes benefit from the parental mesh structure described in section 3.1. The data storage, described in section 3.5, is such that the location of any cell within any mesh, relative to its siblings, cousins and ancestors, can easily be determined.

3.4.1 Transfer of Solution Data to the New Grid Structure

Whenever the grid structure changes, the solution data must be transferred from the old to the new grid structure. For every cell of the new grid structure that

coincides with a cell of the old grid structure of equivalent resolution, the solution data is simply transfered from the old to the new grid structure. The solutions of the remaining cells, that were previously unrefined, are interpolated from the solutions of the underlying coarse grid cells. The type of interpolation scheme is open to question. Quirk [79] noted that bi-linear interpolation does not lead to a conservative transfer of data and that it does not guarantee monotonicity of the data. Quirk suggests using a MUSCL procedure, which does not have the afore mentioned disadvantages. This interpolation scheme has been adopted in the work presented here. A piece-wise linear reconstruction of the coarse grid solution is generated using the MINMOD slope limiter, i.e. if $q_{i,j}$ represents the solution in the coarse cell (i, j), then the limited slopes for the i and j directions are given by

$$\begin{aligned} q_{i+\frac{1}{2},j} - q_{i-\frac{1}{2},j} &= \phi(q_{i+1,j} - q_{i,j}, q_{i,j} - a_{i-1,j}) \quad \text{and} \\ q_{i,j+\frac{1}{2}} - q_{i,j-\frac{1}{2}} &= \phi(q_{i,j+1} - q_{i,j}, q_{i,j} - a_{i,j-1}) \quad \text{respectively.} \end{aligned}$$

Where ϕ, the MINMOD function, is given by

$$\phi(a,b) = \begin{cases} 0 & \text{if } ab < 0 \\ \\ \text{sgn}(a)\,\text{MIN}\,(|a|,|b|) & \text{otherwise} \end{cases}$$

Given the slopes, a first order interpolation scheme can be used to obtain the fine cell data.

3.4.2 Updating the Coarse Grid Solution

Whenever the fine grid solution reaches the same point in time as the underlying coarse grid(s), the fine grid data is used to update the coarse grid solution. The process is recursively implemented (refer to section 3.2.1), in such a way that the coarse level G_L is updated before $G_L - 1$. There are two aspects to this part of the transfer process. Firstly, every covered coarse grid cell solution $\mathbf{U}_c^{m'}$ is over-written with the spatially integrated solution of the covering fine cells. This is equivalent to the summation of all the overlying fine cell solutions $\mathbf{U}_{i,j}^{m}$, multiplied by the ratio of the fine to the coarse cell areas, i.e.

$$\mathbf{U}_c^{m'} = \frac{\Delta x_{G_L} \Delta y_{G_L}}{\Delta x_{G_L-1} \Delta y_{G_L-1}} \sum_{j=1}^{\Gamma_{G_L}} \sum_{i=1}^{\Gamma_{G_L}} \mathbf{U}_{i,j}^{m}$$

where for grid level G_L, Δx_{G_L} and Δy_{G_L} are the grid spacings in the x and y directions respectively, and Γ_{G_L} is the refinement factor relative to the underlying grid. This over-writing of the solution data leads to a lack of conservation. In order to ensure conservation, the second aspect adjusts the coarse grid solution, by correcting

the necessary coarse cell fluxes. This process is commonly referred to as the *conservation flux fix* [9]. The uncovered coarse grid cells that abut fine-coarse boundaries, have previously been updated using a coarse flux which lies along the fine-coarse boundary. This flux is likely to be different to the integrated fine cell fluxes along the same boundary. Thus, by updating the solution with the difference between the coarse and the integrated fine cell fluxes, conservation can be ensured[5]. Again the integration is a simple summation, however it is a little different to other AMR flux fixes, because of the independent fine grid time step. Thus, the flux difference in the (x-direction), $\Delta\mathbf{F}$, between the coarse grid flux, $\mathbf{F}_c$, and the summation of the fine grid fluxes that lie along the same fine-coarse boundary, f_j, is given by

$$\Delta\mathbf{F} = \mathbf{F}_c - \sum_{n=1}^{q} \sum_{j=1}^{\Gamma_{G_L}} \frac{\Delta t^n_{G_L}}{\Gamma_{G_L} \Delta t^m_{G_L-1}} f_j \tag{3.1}$$

where q is the number of fine mesh time steps ($\Delta t^n_{G_L}$) within the single coarse mesh time step ($\Delta t^m_{G_L-1}$). The coarse cell solution $\mathbf{U}_c^{m'}$ is then updated to $\mathbf{U}_c^m$ by

$$\mathbf{U}_c^m = \mathbf{U}_c^{m'} \begin{matrix} - \\ + \end{matrix} \frac{\Delta t^m_{G_L-1}}{\Delta x_{G_L-1}} \Delta\mathbf{F} \tag{3.2}$$

The plus or minus sign in equation 3.2 is dependent on whether the flux lies along the $i - \frac{1}{2}$ or $i + \frac{1}{2}$ boundary. The expressions for the y-direction $\mathbf{G}$ fluxes are similar to those given in equations 3.1 and 3.2. For the results presented here, the inclusion of the flux fix does not yield noticeably different solutions. However, all the results presented here were computed with the flux fix in place.

Quirk [79] investigated how the AMR treatment of fine-coarse boundaries effects the monotonicity of the solution. For the linear advection equation, he concluded that, provided the refinement factor of the fine grid was not excessive (greater than 4), the solution computed with a first order upwind scheme would remain monotone. Unfortunately, the analysis for non-linear systems and more complicated schemes is far too involved and intractable. However, there now exists a number of AMR solutions, computed with various schemes and refinement factors, that do not appear to be impaired by grids with large refinement factors, notably [121] in which a refinement factor of 9 was used.

3.4.3 Transfer of Solution Data to the Mesh Ghost Cells

Solution data must also be transfered to the ghost cells surrounding every mesh patch. For any given ghost cell the source of the solution data is dependent upon

[5]Only for conservative homogeneous systems of equations solved with conservative numerical integration schemes.

its location within the computational domain. Figure 3.13 depicts a highlighted fine mesh patch that abuts, two external boundaries and two other fine meshes (belonging to the same grid level). Where a mesh abuts an external boundary of the problem domain, transmissive or reflective boundary conditions are imposed such that there is either undisturbed or no flow respectively across the boundary. This involves reflecting the solution data from the internal mesh cells, with the normal momentum components negated for reflective boundaries. Where a ghost cell coincides with another internal mesh cell, the ghost cell solution data is transfered directly from the internal cell. The solutions of the ghost cells that lie along fine-coarse boundaries are interpolated from the underlying coarse grid cells. The type of boundary and the source of the solution data is provided by the parental mesh structure. For example, consider a three level grid structure, such as the one shown in figure 3.1. Suppose that the number of meshes on grid levels $G_L = 1$ to 3 are 4, 20 and 100 respectively. Thus, every coarse mesh has, on average, five covering fine meshes. As in figure 3.1, some of the ghost cells may or may not coincide with the internal cells of other sibling meshes. This typically results in searching less than five meshes in order to find the correct solution data. The worst case scenario is shown in figure 3.1, when two abutting fine meshes don't share any common ancestors except for the base grid level. In this situation the search would involve a maximum of ten (on average five) meshes (zero[6] for $G_L = 1$ and five each for the other two levels). Without the parental mesh structure described in section 3.1, the transfer processes would be required to search through and test for coverage the 100, $G_L = 3$, meshes until the covering mesh is detected; on average half the meshes (50).

The ghost cells must be primed with solution data before every time step integration. (The ghost cells need to be updated even when the grid structure is unchanged.) The efficiency of this part of the transfer process is improved by constructing memory stacks of pointers during the main transfer of data to the new grid structure. For every ghost cell, the pointers point to the source of its solution data, i.e. another cell within the same grid structure. A little more information is required for those ghost cells whose solutions are interpolated from the underlying coarse cells. Figure 3.14 shows a one-dimensional view of a fine-coarse boundary, for a single coarse grid time step. The ghost cells are depicted by the dashed lines. Note that, the number of fine mesh time steps is different to the refinement factor and that the size of the time steps also varies. Each of the two fine ghost cells shown have their solutions linearly interpolated between two, spatially interpolated, coarse grid solutions before every fine grid integration. Thus, the solutions of the coarse grids must be advanced to the next time step before those of the finer grids can be updated. The spatial interpolation is the same as that for the newly generated fine cells, described in section 3.4.1. The two spatially interpolated coarse grid solutions are stored in the memory stacks for the subsequent fine grid time levels.

[6]The domain grid points directly to the relevant base mesh.

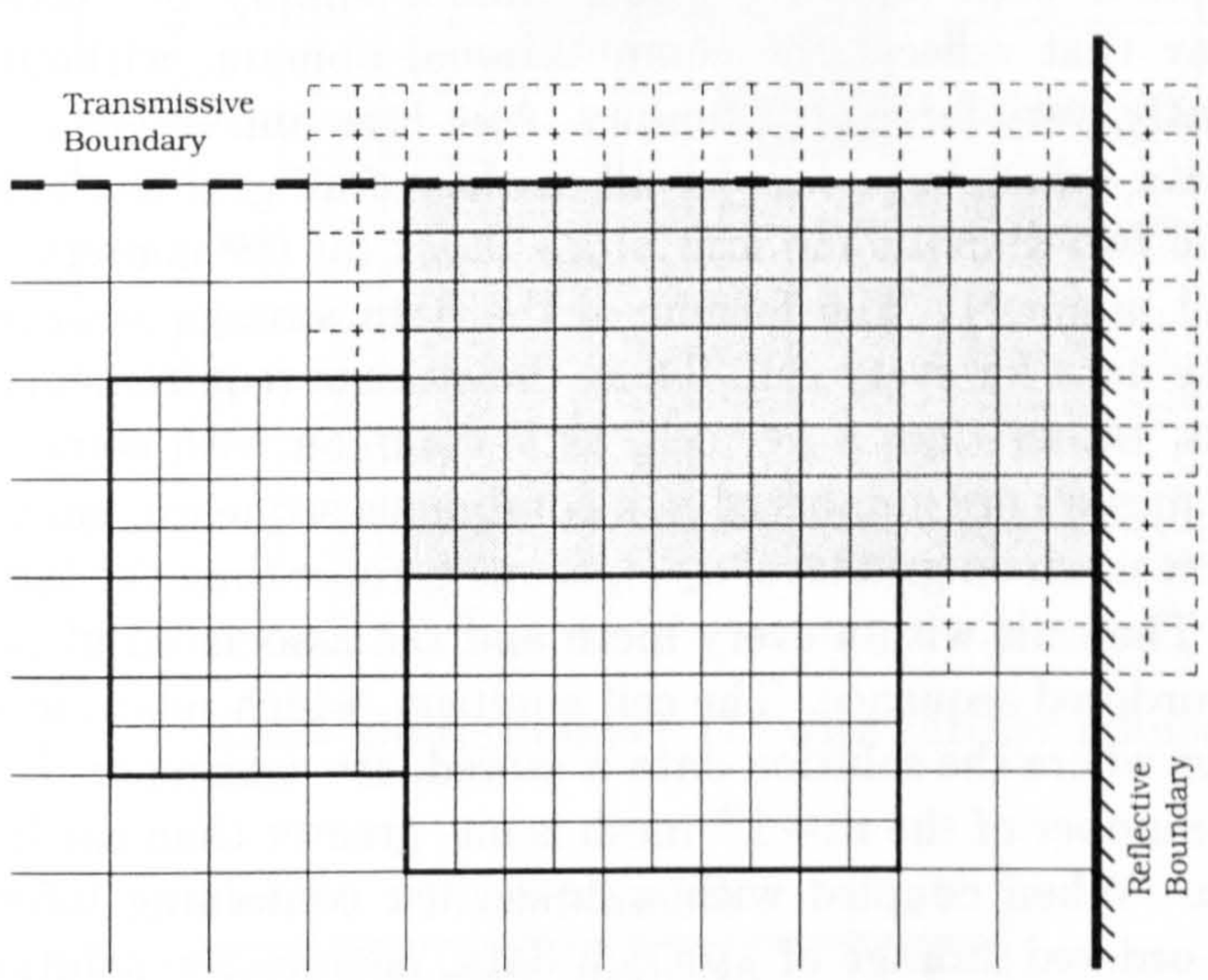


Figure 3.13: The *extra* ghost cells around a general mesh patch.

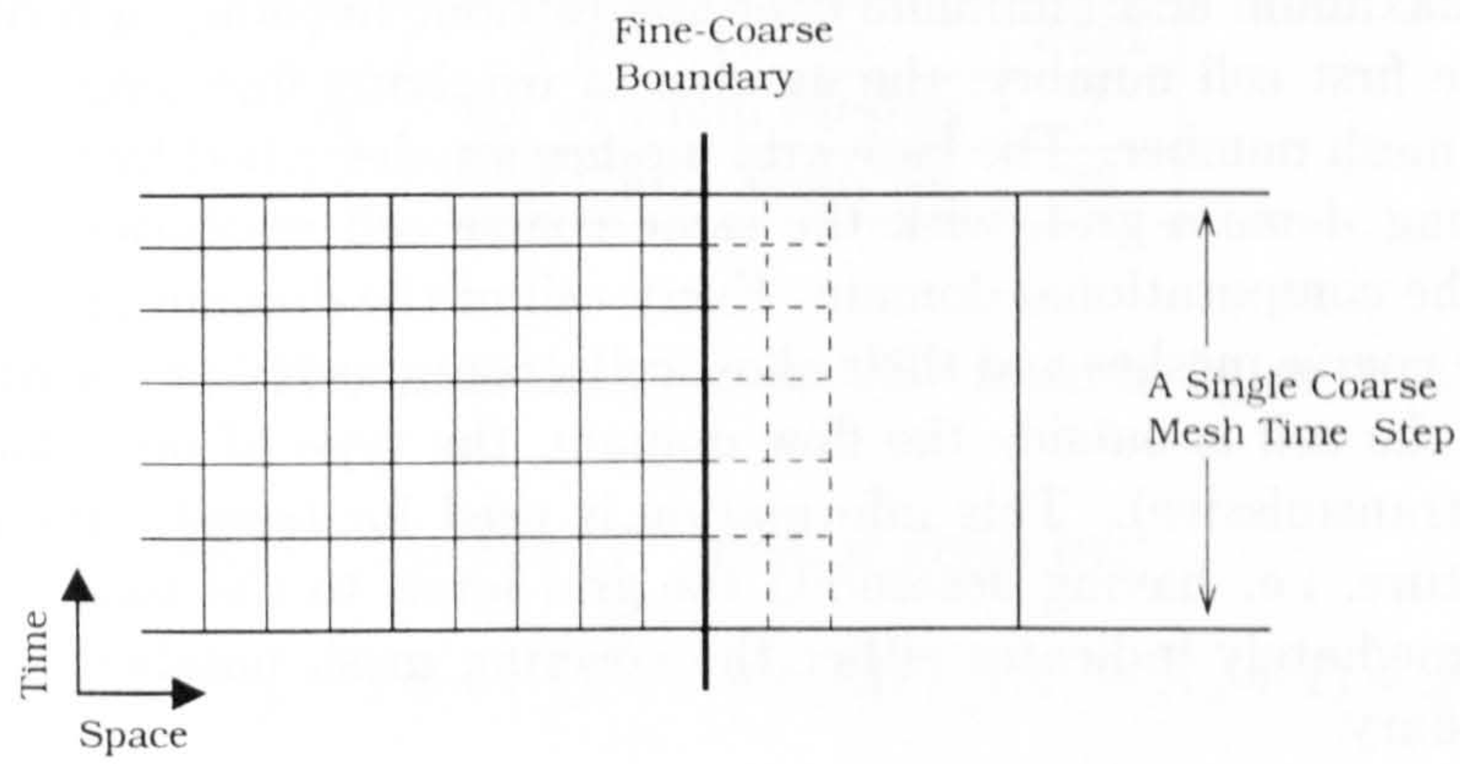


Figure 3.14: A one-dimensional view of a fine-coarse boundary.

3.5 Data Storage

The data associated with adaptive grids, cannot simply be stored in a multi-dimensional array that reflects the computational domain, without leaving most of the, subsequently, very large array empty. AMR uses linked lists to access the relevant solution data, which is stored, for all meshes of all grid levels, in a prescribed format in a single two-dimensional array (one index for the conserved variable, the other for the cell number). The format of the data storage removes the need to store connectivity data for every cell. Thus, the storage requirements of AMR are on a per mesh basis, rather than a per cell, as is common with some other adaptive techniques. The meshes are numbered in a continuous sequence, such that the mesh number of the first mesh on grid level G_L+1, is one greater than the last mesh number of grid level G_L. The cells within every mesh and the associated ghost cells are also numbered in an ordered sequence. The cell numbers, which relate to the position in the storage arrays where the solution data is stored, are continuous between meshes, i.e. the first cell number of the $m+1^{th}$ mesh is one greater than the last cell number of the m^{th} mesh. When coupled with a linked list containing information about every mesh, the ordered storage of AMR cell data, enables the solution data in any particular cell to be efficiently accessed. The amount of mesh data that is stored is a compromise between the expense of computing specific data and the expense of storing it. For the two-dimensional AMR approach presented here, every mesh in the mesh data array, stores the following eight parameters (integers): the parent mesh number, the maximum and minimum coordinates (four in total) in terms of parent mesh cells, the first cell number, the number of overlying fine (child) meshes and the first child mesh number. The base grid meshes are described by the coordinates of an underlying *domain* grid, with the same coarse cell resolution, which covers the whole of the computational domain. Every cell of the domain grid array that is covered by the coarse meshes and their ghost cells, contains either the covering mesh number or, if the cell is outside the flow domain, the type of boundary condition (reflective or transmissive). This information is used for transferring data within the grid structure, i.e. having descended the grid levels to the base grid, the base grid array immediately indicates either the covering mesh number or the type of external boundary.

Consider figure 3.15, in which the 17^{th} mesh, (the numbers in the top left-hand corner of the meshes represent the mesh numbers), is shown with its parent mesh (number 2) and its two child meshes (numbers 88 and 89). The mesh data stored for the 17^{th} mesh in figure 3.15 is given in table 3.1. If the 17^{th} mesh is assumed to be part of grid level G_L, then the refinement factors, Γ_{G_L}, for grid levels G_L and G_L+1 are 3 and 4 respectively. The number of rows of ghost cells (shown by the dashed lines), n_g, around every mesh is clearly two. Thus, using this information,

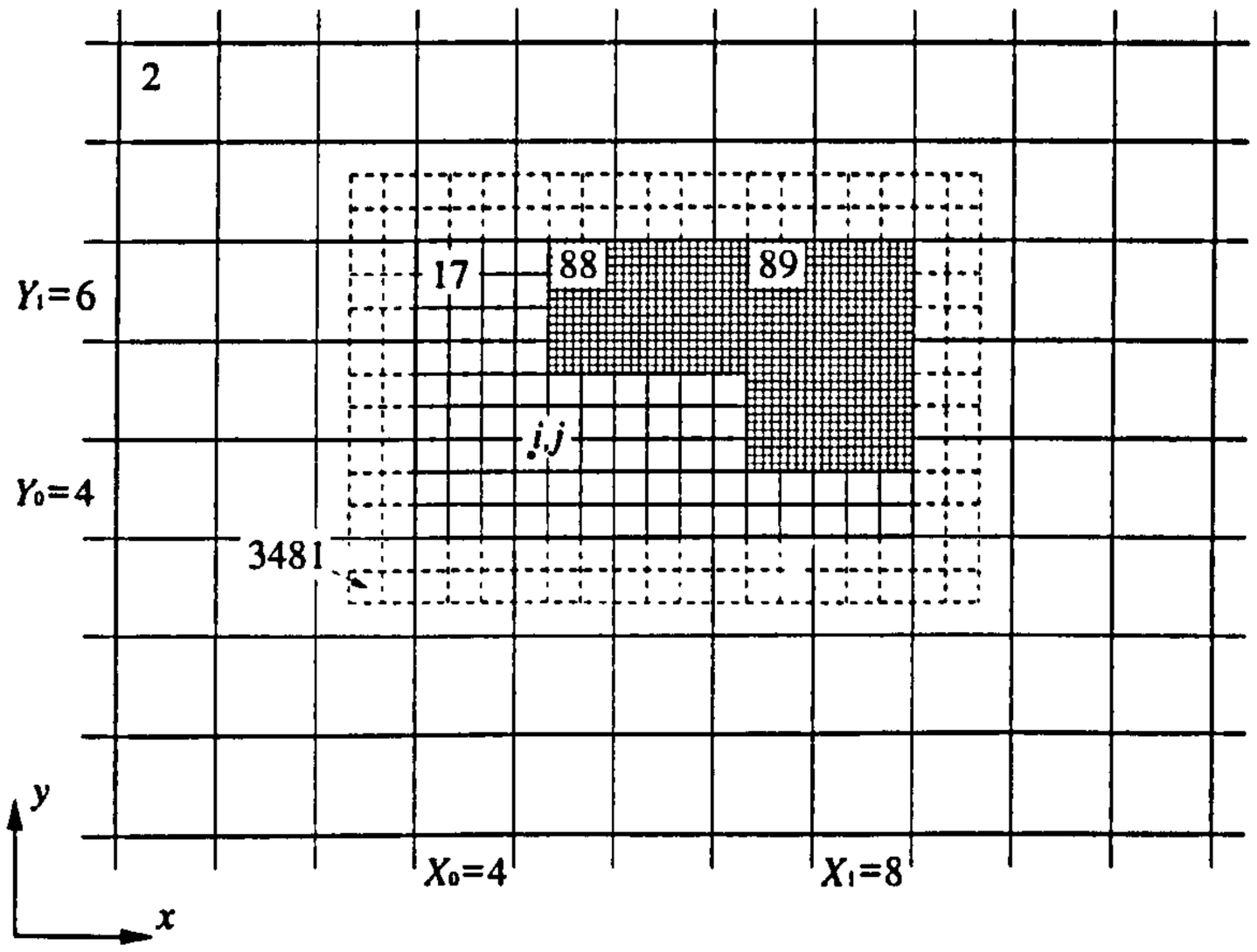


Figure 3.15: A typical refined mesh (number 17) with further refined child meshes.

m	Mesh Number	17
m_p	Parent Mesh No.	2
X_0	Minimum X-coord.	4
Y_0	Minimum Y-coord.	4
X_1	Maximum X-coord.	8
Y_1	Maximum Y-coord.	6
C_1	First Cell No.	3481
n_c	No. of Child Meshes	2
m_c	First Child Mesh No.	88

Table 3.1: The mesh data for the 17^{th} mesh in figure 3.15.

the cell number of cell $(C_{i,j})$ of the 17^{th} mesh is given by;

$$C_{i,j} = C_1 + n_g - 1 + i + (j - 1 + n_g).(\Gamma_{G_L}(X_1 - X_0 + 1) + 2n_g) \qquad (3.3)$$

i.e.

$$C_{4,3} = 3481 + 2 - 1 + 4 + (3 - 1 + 2).(3(8 - 4 + 1) + 4) = 3562$$

3.5.1 Ghost Cell Storage

Before a mesh is integrated, the surrounding ghost cells must be primed with solution data. A fine grid is usually integrated through several time steps before its structure is updated, (i.e. when its time level coincides with that of the underlying coarse grid). Furthermore, most realistic computations involve source terms, which are usually solved via operator splitting and therefore require the ghost cells to be updated beforehand. Hence, the ghost cells may need to be updated many times between changes in the grid structure. The costs associated with priming the three different types of ghost cell can be significantly reduced by storing information relating to the source of the solution. For the internal fine-fine boundary and the external boundary ghost cells, a pointer is stored which points to another internal cell of the same grid level, from which the solution data is transfered directly. The fine-coarse boundary ghost cell solutions are interpolated from the underlying coarse grid solution. For every such ghost cell, the spatially interpolated solutions at both the current and the previous coarse grid time levels are stored. The fine-coarse boundary ghost cell solutions at the intervening times, can then be linearly interpolated between the two stored solutions.

3.5.2 Flux Fix Storage

For every fine-coarse boundary, the coarse fluxes for all the conserved variables are stored in a single two-dimensional array. One index refers to the conserved variable number and the other to the fine-coarse boundary number. An array of pointers is used to locate the relevant fine-coarse boundaries. Whenever the overlying fine meshes are integrated, the peripheral fluxes are returned. Those fine fluxes that lie along fine-coarse boundaries, are then multiplied by the relevant factor, and the flux fix array updated by subtracting it from the underlying coarse flux that is stored, (see section 3.4.2).

3.5.3 Storage Requirements

For a reasonable computation the main memory costs are associated with the storage of the solution data and the cell flag data (where necessary).

Whenever the grid structure is updated, the solution data must be transfered from the old to the new grid structure. It is because the AMR approach presented here is programmed using FORTRAN 77, that for grid structures involving more than two grid levels, the old grid structure and solution must be stored. This effectively, almost doubles the memory requirements. However, if the AMR technique was programmed in a language that supports dynamic memory allocation, such as

'C' or FORTRAN 90, then this would not be necessary.

Obviously as with all adaptive gridding techniques, the memory requirements compared to a regular grid approach vary according to the proportion of the flow domain requiring refinement. However, the memory requirements of AMR is likely to be less than most other adaptive gridding techniques, because the ordered format of the solution data requires less information to be stored.

3.6 Summary

The reliance on a parental data tree structure, where a single mesh has only a single parent mesh, greatly simplifies the accessing of data between any two meshes. This is probably the most fundamental aspect of the algorithm presented here. It results in the overall code performing most efficiently with a much stricter 'refine condition'. Furthermore, by not permitting meshes to overlap one another, it prevents the unnecessary storage and integration of duplicate cells.

The mesh structure of a grid level is regenerated immediately after its parent grid level has been advanced to the next time level and not every few parental time steps. This results in a very significant reduction in the amount of unnecessary cell storage and integration, which would otherwise be required in order to guarantee the continuous refinement (capture) of features. Thus, even though a greater proportion of the processing time is spent regenerating the grid structure, the overall efficiency of the algorithm is kept to a minimum.

Since very large refinement factors can be achieved over several grid levels, merely assuming the temporal refinement to be equal to the spatial refinement almost certainly results in instability at some point in space-time, unless the time steps are overly restricted by reducing the Courant number. Conversely it may result in too many integrations of finer grid levels. Stability is ensured and wasteful integrations are avoided by employing a proper evaluation of the time step for every grid level. This benefit is most noticeable when modeling chemically reactive flows, where rapid changes in wave speed can occur.

The likelihood of missing (not refining) both small scale and developing features, is reduced by utilising a flagging technique that, examines the finest data available and is based on the internal wave structure of the Riemann problem. Greater reliability and flexibility in targeting particular flow features for refinement can be achieved.

The amount of unnecessary processing and storage is further reduced by efficiently clustering the cells requiring refinement into mesh patches. Compared with other mesh clustering techniques the new methodology yields a significant reduction in the overall processing time.

In conclusion, the methodologies presented in this chapter result in a very reliable and versatile adaption algorithm. None of the processes described here, would preclude the development to three spacial dimensions. Moreover, in three dimensions the improvements in efficiency would be even more apparent.

Chapter 4

The Combined Chimera-AMR Approach

The numerical errors that are incurred in fluid dynamics computations for any given mesh size, can be kept to a minimum by utilising Cartesian computational grids. However, Cartesian grids are not suitable for representing arbitrary shaped solid boundaries. Curvilinear grids that are fitted to the outlines of solid boundaries can yield good quality computed solutions; high order boundary conditions can easily be applied, and provided the grid cells aren't overly distorted, the numerical solution errors can be kept to a minimum. There exists several different techniques for generating curvilinear grids [104]. For situations in which the boundaries of the domain are in relative motion, the grid structure has to be altered at regular intervals, usually every time step, throughout the computation. If the computational efficiency is important, then the more expensive grid generation techniques are inappropriate. Algebraic grid generation techniques are the most inexpensive. However, formulating the necessary algebra for situations that involve large transformations of the domain, can be quite difficult and the resulting grid may be very distorted.

The so called *Chimera* approach [2, 6, 19] utilises a base grid within the main body of the flow field and overlays it with curvilinear grids fitted to any irregular boundaries. Often a Cartesian base grid is used, thereby reducing the computing costs (both storage and processing time) and maximising the accuracy of the solution. The boundary-fitted grids only extend a short distance into the flow field; far enough to cover only a few Cartesian cells. The Chimera approach exploits the advantages of both Cartesian and boundary-fitted grid types, in order to fully discretise flow domains that are bounded by solid surfaces with arbitrary geometries. The accuracy of the solutions computed with the Chimera approach are very much dependent upon the quality of the boundary-fitted grids. A high quality grid is deemed to be one that has only a gradual variation in its cell geometries. The types

of problems under investigation here, involve severe discontinuities such as strong shock waves, material interfaces and delicate slip surfaces. Numerical schemes used to compute such features, produce erroneous solutions when applied to grids with skewed or rapidly changing cell geometries. The smoothness of the boundary-fitted grids and the representation of solid boundaries, can easily be improved by increasing the resolution of the grid cells. However, if the cell resolutions of the two grid types are intended to be similar, this not only results in more boundary-fitted grid cells, but also more Cartesian grid cells. Thus, both the memory storage and the processing times for typical computations can become excessively large. What follows is a description of a way of combining the Chimera approach with the proven grid adaption technique of AMR. This combination should provide access to high quality solutions, without incurring the large computing costs associated with a conventional regular grid. Hereinafter, a boundary-fitted grid shall be regarded as being made up of one or more curvilinear meshes, which cover a single isolated body or boundary.

The layout of this chapter is as follows. A basic overview of the Chimera approach is described in section 4.1, followed by a description of the Chimera-Adaptive Mesh Refinement (CAMR) grid structure in section 4.2. An outline of the combined approach is given in section 4.3. Sections 4.5 and 4.6 detail the generation of the grid structure and the transfer of solution data within it respectively. The storage requirements are discussed in section 4.7 and a summary of the main points is given in section 4.8.

4.1 Overview of the Chimera Approach

There are several variations on what has become known as the Chimera approach. This section details a Chimera approach which was designed with the intention of incorporating it into an AMR algorithm. In order to advance the solution data of the Cartesian and the boundary-fitted grids, extra cells around the periphery of the grids need to be primed with data before every time step. For each grid type, the extra cells' solutions are interpolated from the covering cells that belong to the other grid type. The accuracy of the interpolation has a direct effect upon the quality of the computed solution. Interpolation errors can be kept to a minimum by ensuring that the cell sizes of the two mesh types are as similar as possible. This also removes the need to interpolate between the two grid types in time, as both can be advanced with equal time steps. The Chimera grid structure for a flow domain around a solid cylinder is shown in figure 4.1. Note that, the Cartesian cells are colour coded, according to their positions within the problem domain and the overlayed boundary-fitted cells are transparent. The red cells represent the cells that are cut by the outlines of the solid boundaries, the blue cells represent the

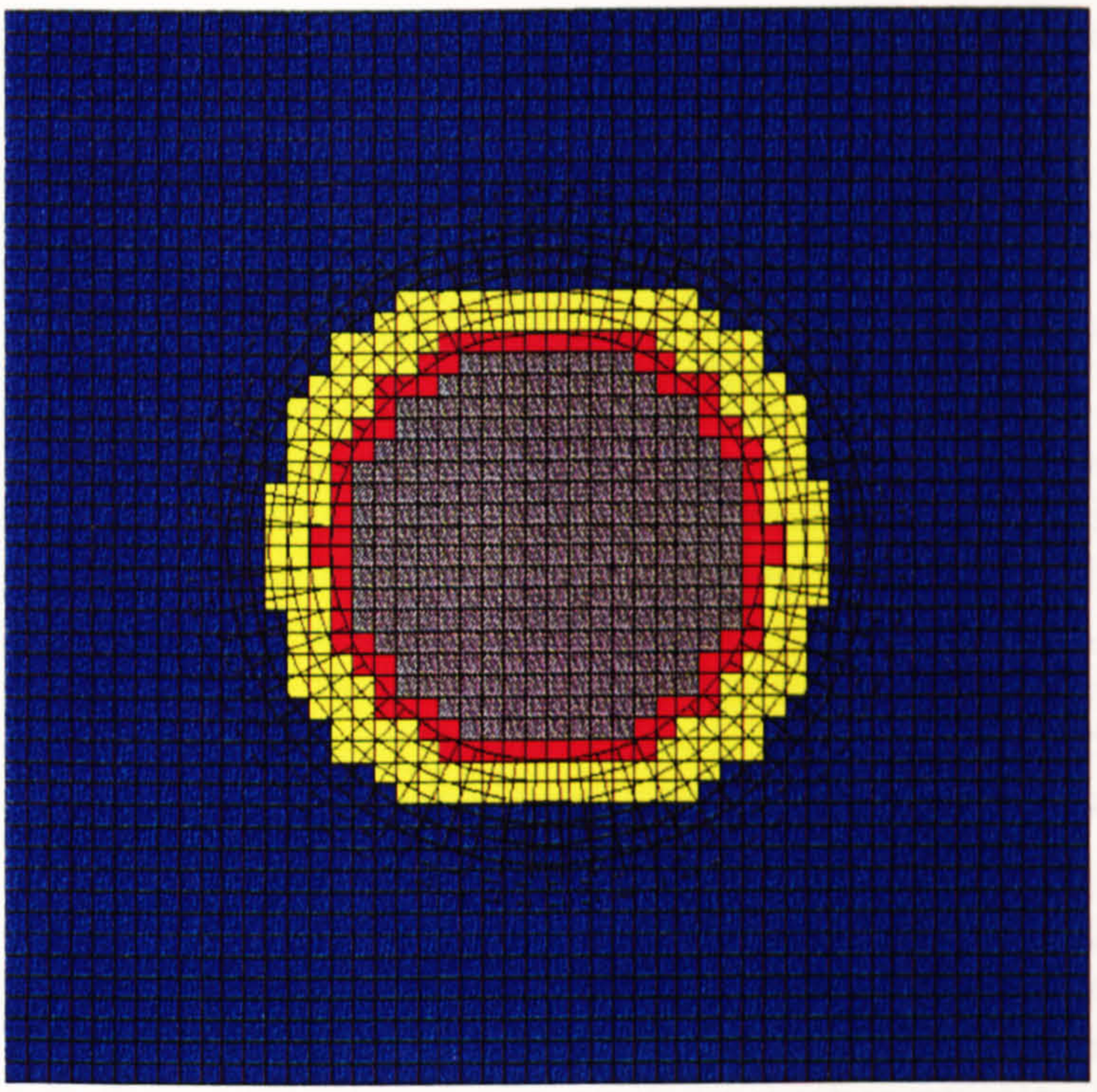


Figure 4.1: The colour coded Chimera grid structure for a solid cylinder.

cells within the flow field and the grey cells represent the cells that are enclosed within the solid boundaries. The yellow cells around the periphery of the cut cells, represent flow field cells that require their solution data to be interpolated from the boundary-fitted mesh cells before every time step. Similarly, the solutions of the boundary-fitted grid's flow field boundary cells, (shown by the dashed lines in figure 4.1), must be interpolated from the underlying Cartesian cells before every time step. In order to advance a cell's solution, an n^{th} order scheme requires the surrounding solution data from a stencil of at least n cells in every computational direction. Hence, for the $2nd$ order operator split WAF scheme, all the flow field cells within 2 cells of a cut cell are designated as being peripheral cells, (see figure 4.1). Only when all the boundary cells have been primed with the correct data can the surrounding flow field cell solutions be accurately computed.

There needs to be a sufficient number of boundary-fitted grid cells, regardless of the grid orientation and movement relative to the Cartesian grid, to ensure full coverage of the underlying peripheral cells during an ensuing time step. The extent to which moving boundary-fitted grids penetrate the flow field is generally greater than that for fixed grids. This is to ensure continuous coverage of the flow domain, i.e. all peripheral (yellow) cells remain covered by the boundary-fitted grids. The boundary-fitted grid in figure 4.1 has five cells in the direction normal to the

boundary, plus an extra two flow field boundary cells.

4.2 Overview of the Chimera-AMR Grid Structure

The CAMR grid structure encompasses both the boundary-fitted grids and the AMR hierarchy of grids. The generation of the boundary-fitted meshes around non-deformable solid surfaces only needs to be done once at the beginning of any computation. Thus, the efficiency of the boundary-fitted grid generation process is of little importance. This is in contrast to the second stage, of generating the AMR grid structure, which for moving boundaries is altered every time step. Even for fixed boundaries, it needs to be regenerated every coarse time step, i.e. for all but the finest grid level time steps.

Given details of the solid boundaries in the form of either, data points along the outlines of the solid surfaces or equations of curve segments, the grid cells are defined by extending normal vectors into the flow field. The AMR grid structure is generated by identifying regions requiring refinement. Every cell in the structure is given a particular flag according to its position within the problem. These flags are similar to those in the regular Chimera approach, described earlier. The main difference being that, for all but the finest grid level, the cut cell and surrounding peripheral cell flags serve as refinement flags. The meaning of the various cell flags is best illustrated by figures 4.2 and 4.3, which depict the structure of the finest two grid levels for a Mach reflection problem. These two figures are magnifications of the same region around the reflected Mach stem.

Both figures show the Cartesian AMR grid cells (colour coded) and the boundary-fitted grid cells (transparent). The red and grey cells indicate cells *cut* by and *enclosed* by solid boundaries respectively. In order to ensure that there is sufficient fine level coverage during boundary movement, an extra layer of cells within all moving boundaries is identified, as requiring refinement (coarse grid levels only). These *enclosed peripheral* cells are depicted by the lighter grey cells that lie adjacent to the cut cells. The remaining flow field cells can be categorised as being one of two types. The first type, the *flow field peripheral* cells (yellow) cannot be updated because a sufficient data stencil does not exist. However, the cells of this type are primed with data, so that there is a sufficient data stencil to update the second type of cells, which shall be referred to as *flow field* cells (blue). The flow field peripheral cells that belong to the finest AMR grid level and the boundary cells (dashed lines) of the boundary-fitted grids, are both treated in a similar way to those in the conventional Chimera approach. Hence, for each grid type, the necessary solution data is obtained by interpolating the solution data of the other type. Figure

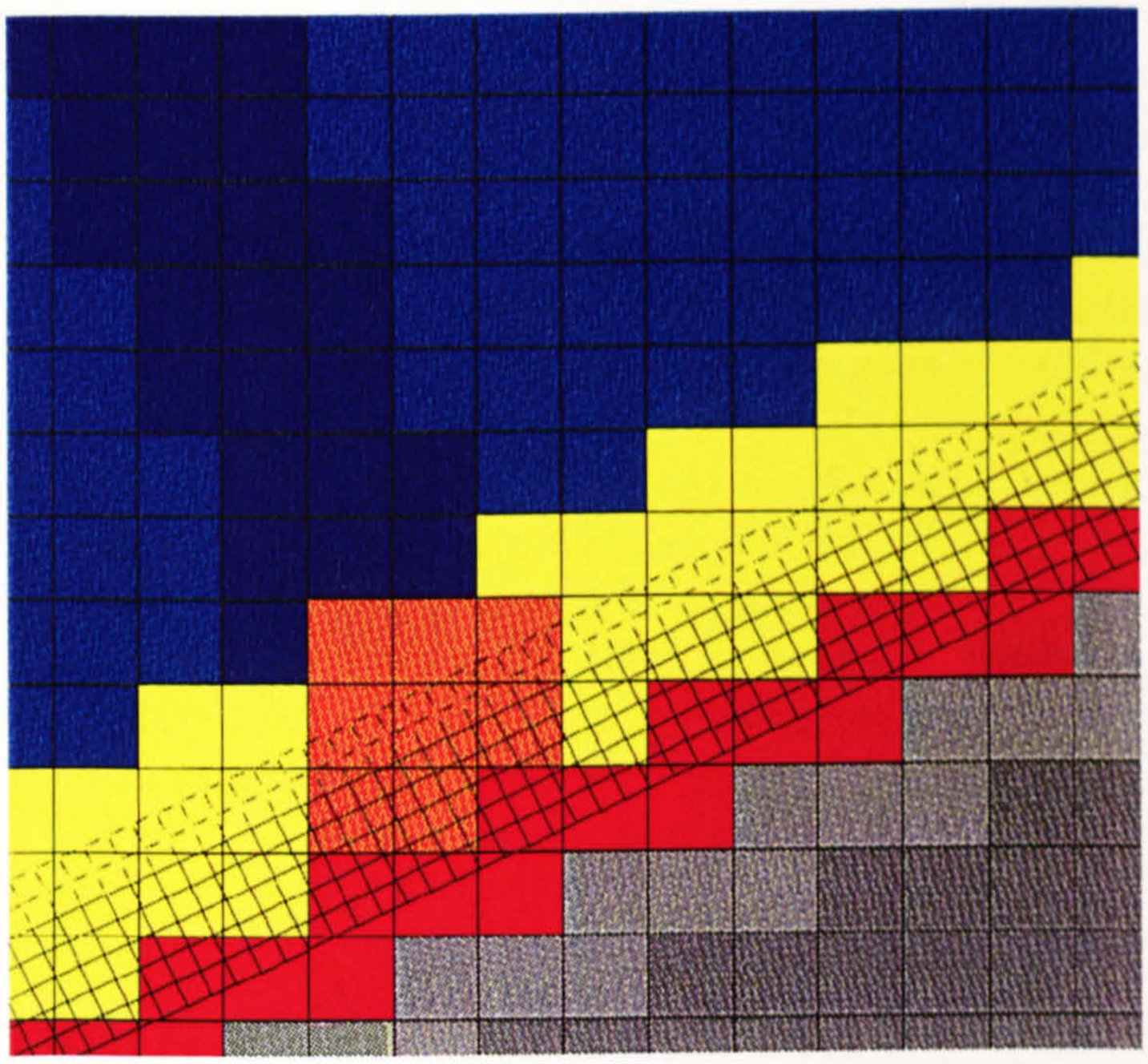


Figure 4.2: A coarse AMR grid level overset with a boundary-fitted grid.

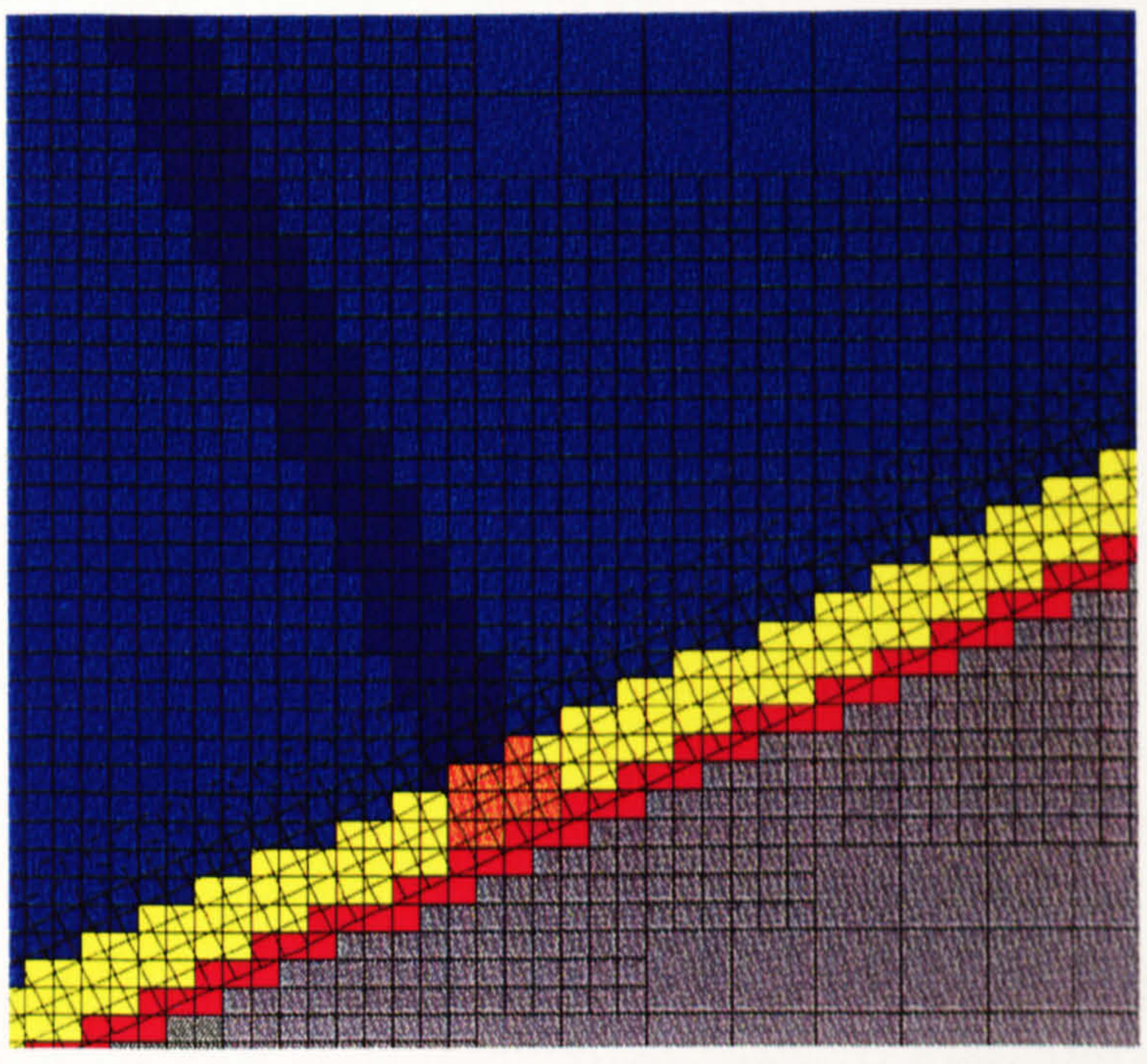


Figure 4.3: The finest AMR grid level overset with a boundary-fitted grid.

4.2 clearly shows that sufficient mutual coverage, between the two grid types, does not exist for the flow field peripheral cells of the coarser grid levels. However, having identified these cells as requiring refinement, they are covered by finer grid cells. In the normal AMR way, the coarse cell solutions can then be obtained from an integral average of the solution data of the overlying fine cells. The dark blue and the orange cells in figures 4.2 and 4.3 represent flow field and flow field peripheral cell types respectively, that have been flagged by the flow feature detection mechanism for future refinement. Thus, the Mach stem, which is normal to the boundary, is clearly apparent in both grid levels. Note that, the AMR approach presented here, flags even the finest grid level for refinement. Of course the finest level isn't actually refined further, but by transferring all refinement flags to the underlying coarse cells, extremely fine features (down to the resolution of the finest level) will not be 'missed' during subsequent changes to the grid structure. Note that, no extra safety cells are added around the cells requiring refinement, unless they are flagged as part of a flow feature that requires further refinement.

4.3 Overview of the Chimera-AMR Approach

The first step in the CAMR process is to generate the initial boundary-fitted and AMR grid structures. Generally, the boundary-fitted grids around non-deformable solids will not change during a computation. However, the AMR grid structure will almost certainly change every time step. Thus, once the initial conditions have been set up, the complexity of the approach is manifest in the efficient, continuous regeneration of the AMR grid structure and transfer of data to and from various parts of the changing data structure. The recursive sequence of processes for continually updating the AMR grid structure and solution, was illustrated by the pseudo code in figure 3.4. Figure 4.4 depicts the equivalent pseudo code for the combined CAMR approach. The sequence of procedure calls remains much the same, but for the inclusion of the **set_AMR_flags**, **set_BF_ghost_cells**, **integrate_BF_grid** and **move_BF_grid** procedures. The other procedures, (**new_AMR_structure**, **time_step**, **set_AMR_ghost_cells**, **integrate_AMR_grid** and **update_coarse_grid_solution**), differ slightly to those in the conventional AMR sequence.

The procedure **set_AMR_flags** identifies the cell flags for grid level G_L. It involves tracing around the outlines of the solid bodies, in order to determine which cells are cut by, which cells lie within and which cells lie outside the solid boundaries. Procedure **set_BF_ghost_cells** primes the boundary-fitted ghost cells with solution data before the **integrate_BF_grid** procedure advances the solution by the time step Δt. (Remember that the boundary-fitted grid time step is the same as the finest AMR grid level time step.) Note that, as there are no noticeable increases

```
PROCEDURE sequence(T_0)

   G_L=1
   DO WHILE (T_GLmax<T_0)
      IF (G_L<G_Lmax) new_AMR_structure(G_L)
      DO WHILE (T_GL<T_GL-1)
         IF (G_L=G_Lmax) set_AMR_flags(G_L)
         time_step(G_L, Δt)
         T_GL=T_GL+Δt
         IF (G_L=G_Lmax) THEN
            set_AMR_ghost_cells(G_L)
            set_BF_ghost_cells
            integrate_BF_grid(Δt)
            move_BF_grid(Δt)
         END IF
         integrate_AMR_grid(G_L, Δt)
         IF (G_L<G_Lmax) G_L=G_L+1
   END DO
   DO WHILE ((T_GL=T_GL-1) AND (G_L>1))
      G_L=G_L-1
      IF (G_L≥1) update_coarse_grid_solution(G_L)
   END DO
END DO

END PROCEDURE
```

Figure 4.4: The pseudo code for the combined CAMR **sequence** procedure.

in the computing costs, the fixed boundary-fitted grid solutions are computed by the moving grid solver, with the velocity components set to zero. After advancing the solution of a moving grid to the next time level, the **move_BF_grid** procedure updates its location, orientation and velocity components of the grid. The location and orientation of the boundary-fitted grids are needed in order to update the AMR grid structure. The velocity components are required for the time step calculation and for calculating the moving grid solution.

The **new_AMR_structure** procedure, which updates the AMR grid structure from G_L to G_{Lmax}, now takes into account, not only the flags associated with the flow field, but also those associated with solid boundaries. Hence, the inclusion of the **set_AMR_flags** procedure in the **new_AMR_structure** procedure pseudo code shown in figure 4.5. The **copy_AMR_structure** procedure copies all types of cell flags, which enables the **transfer_solution** procedure to transfer only the flow field solution data, (i.e. not the irrelevant data stored in any other flag type

```
PROCEDURE new_AMR_structure(G_N)
    copy_AMR_structure(G_N)
    DO FOR G_L=G_N TO G_Lmax+1
        set_AMR_flags(G_L)
        set_AMR_ghost_cells(G_L)
        new_grid_structure(G_L+1)
        transfer_solution(G_L+1)
    END DO
END PROCEDURE
```

Figure 4.5: The pseudo code for the combined CAMR **new_AMR_structure** procedure.

cells). The **time_step** procedure, which conventionally estimates the maximum stable time step for every one of the AMR grid levels, uses both the AMR and the boundary-fitted grid structure solutions to estimate the finest grid level time step. The boundary-fitted grid solutions can then be advanced using the same size time step. The **set_AMR_ghost_cells** procedure transfers the solution data and flag types to either the ghost cells from other internal mesh cells or to the internal mesh cells from the ghost cells. For any given ghost cell, the direction of the transfer is dependent upon whether or not it has been previously flagged and primed with solution data during the **set_AMR_flags** procedure. The **integrate_AMR_grid** and **update_coarse_grid_solution** procedures only affect those cells that are within the flow field, including the flow field peripheral cells.

4.4 The Boundary-fitted Grid Generation for the Chimera Approach

The boundary-fitted grids are generated by first tracing around the outlines of the solid boundaries, identifying equally spaced points. These points are the boundary vertices of the grid cells that are adjacent to the boundary. The spacing between the vertices can be specified prior to solving the problem, but is generally taken such that it is similar to the side lengths of the smallest Cartesian AMR cells. Normal vectors are extended into the flow field from the boundary vertices. The intercell boundaries, which form the boundary-fitted grid structure, are then constructed from discretised points along the normal vectors. The smoothness and continuity of the mesh cells are improved by distorting (rotating) the normals in regions of significant boundary curvature. Where large changes in boundary curvature occur the mesh quality can be improved further by switching from one grid configuration

to another. The resulting cells have little distortion and are all comparable in size. Figure 4.6 depicts the various convex (a,b,c) and concave (d,e) grid configurations. The boundary-fitted grids are segmented into one or more meshes, by specifying

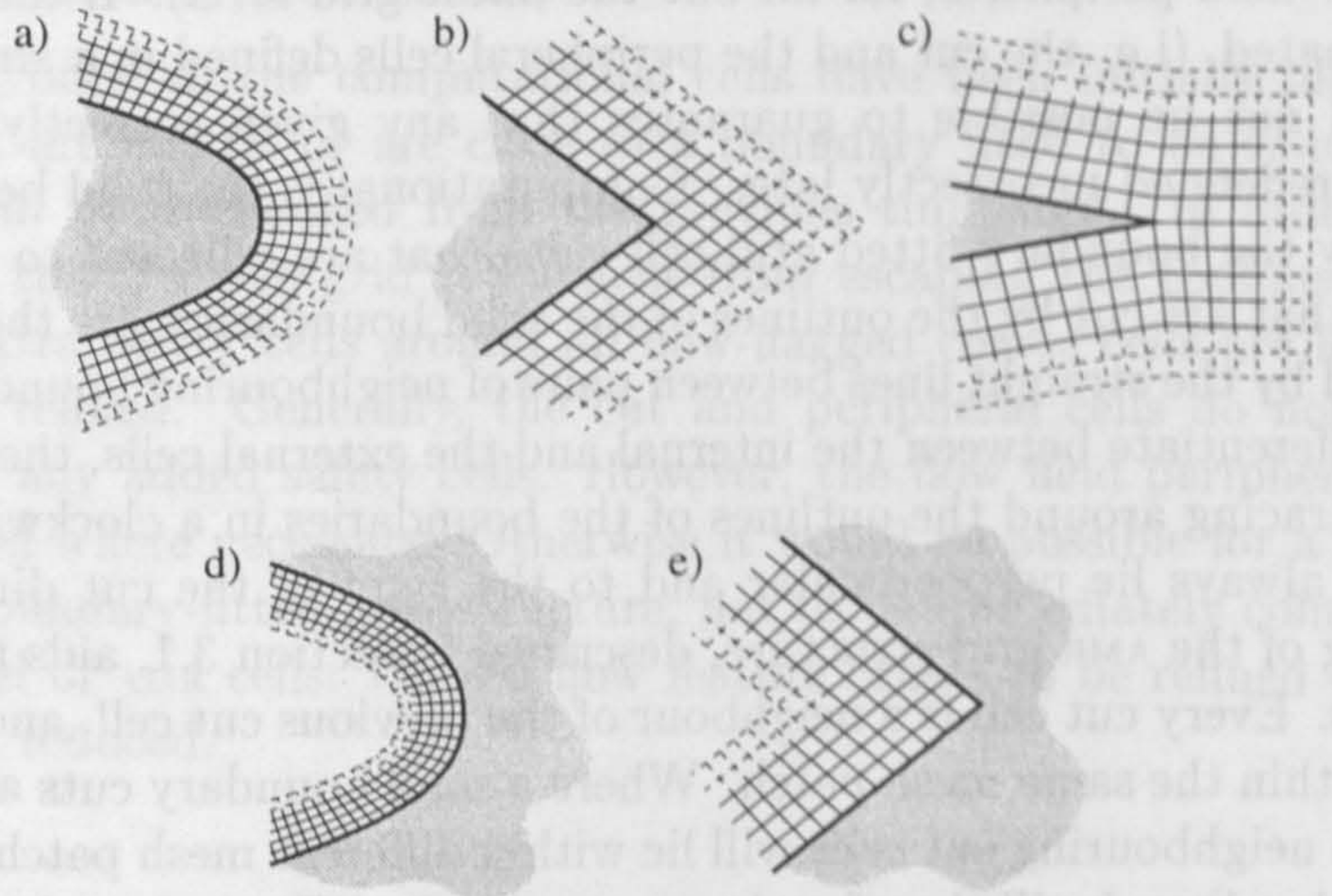


Figure 4.6: The various convex (a,b,c) and concave (d,e) boundary-fitted grid configurations.

the maximum number of mesh cells along the solid boundaries. The limit should be chosen so that all the data associated with any given mesh can be stored in the local memory (RAM) during the integration of the mesh solution. Computationally, these meshes are represented by two-dimensional rectangular mesh patches, which require the data to be rotated normal to the cell faces during the integration process. Similarly to the AMR meshes, the boundary-fitted meshes are bounded by a number of rows of ghost cells. Again the number of rows is dictated by the requirements of the numerical scheme, i.e. two rows for the second order operator split WAF scheme.

4.5 AMR Grid Generation for the Chimera Approach

For moving boundary problems, the identification of the AMR cell flags must be done every time step. Therefore, the efficiency of this process is crucial to the overall performance of the CAMR approach. It is extremely inefficient to take every AMR cell in turn and apply checks to determine its location relative to the solid boundaries. In three spatial dimensions, such a process is beyond consideration. Composing an algorithm that *positively* identifies the cell flags is not a simple task, and is exacerbated by the AMR grid and data structures.

The approach presented here, first identifies all the Cartesian AMR cells in the problem domain that are cut by the outlines of the solid boundaries. These cut cells are then retraced in order to determine the peripheral cells, (enclosed peripheral as well as flow field peripheral, for all but the finest grid level). If these two steps were amalgamated, (i.e. the cut and the peripheral cells defined in a *single* process), then it would not be possible to guarantee that any given correctly defined cell would not be redefined incorrectly later. Computationally, the solid boundaries are represented by the boundary-fitted grid cell sides that are adjacent to them. Thus, the AMR cells that are cut by the outlines of the solid boundaries, are those cells that are intersected by the straight lines between pairs of neighbouring boundary vertices. In order to differentiate between the internal and the external cells, the cut cells are identified by tracing around the outlines of the boundaries in a clockwise direction; the solid will always lie perpendicular and to the right of the cut direction. The proper nesting of the AMR grid structure, described in section 3.1, aids the efficiency of this process. Every cut cell is a neighbour of the previous cut cell, and is therefore likely to lie within the same mesh patch. Where a solid boundary cuts across a mesh boundary, the neighbouring cut cells will lie within different mesh patches (albeit on the same grid level) and will therefore have completely different locations within the storage arrays. In the same manner in which the internal fine-fine boundary ghost cell data is located (see sections 3.4.3 and 3.5), the storage locations of the cut cells can be determined. As the cut cells are identified, pointers to them are stored in a temporary memory stack, in which the first data in is the first data out (FIFO stack). The first cut cell number for every separate boundary and the relative locations of the subsequent cut cells (every cut cell lies to the north, south, east or west of the previous one) are stored. Storing this data avoids the necessity of re-identifying the cut cells during the identification of the peripheral cells. Furthermore, as the cut cells are retraced, the peripheral cells are identified according to the way in which the cells are cut. Figure 4.7 shows the various ways in which a cell can be cut by a solid boundary (the arrow indicates the clockwise direction), for a single orientation (there are three others). The clear cells represent the cut cells, the lighter

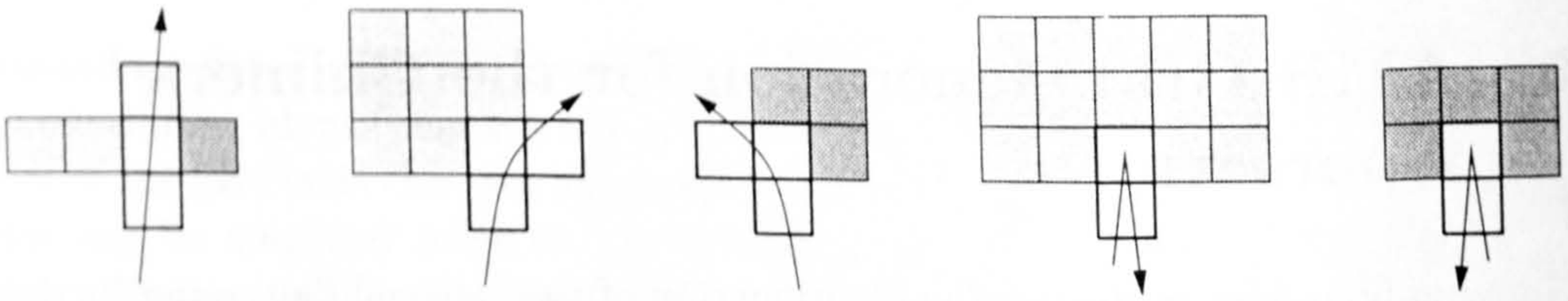


Figure 4.7: The various ways, for a single orientation, in which a cell can be cut.

and darker grey cells the flow field and enclosed peripheral cells respectively. All the ghost cells, whose flags were left unidentified at the end of the complete AMR grid generation process, copy their flags from the corresponding internal cells that belong to the relevant neighbouring mesh, during the proceeding ghost cell update

routine. However, any ghost cell that is identified as peripheral cell, copies its flag to the relevant internal cells instead. This flag information must be transfered via the ghost cells, otherwise a mesh patch that is close but not actually cut by a boundary, may not be affected by the boundary.

Generally, once all the computational cells have been initially identified, only the cells that are cut by or are close to a boundary need to be re-identified; the other cells will be unchanged from the previous time step. In order to prevent flow features that require grid refinement from escaping the confines of the finer grid levels, extra safety cells around all flow-flagged coarse cells are identified and subsequently refined. Generally, the cut and peripheral cells do not require the refinement of any added safety cells. However, the flow field peripheral cells must be flow-flagged where necessary. Otherwise it would be possible for a feature upon exiting the boundary-fitted grid structure, not to be immediately contained within the finest level of AMR cells. Once a flow feature ceases to be refined its resolution is irreversibly reduced.

4.6 Transfer of Solution Data

Before the solutions of the boundary-fitted meshes can be advanced, the surrounding ghost cells must be primed with data. Similarly, the ghost cells and the flow field peripheral cells of the AMR grid structure, must also be primed with data before advancing the AMR grid solution. The process of updating the coarse AMR grid cells that are overlayed by the finer cells was described in section 3.4.2. For the combined CAMR approach, it is only necessary to update the cells within the flow field. Section 4.2 identified the need to include the flow field peripheral cells in the process of updating the coarser grid cells. The process of priming the AMR ghost cells with solution data and flag information was described in section 3.4.3. Section 4.5 detailed the necessary extension for the combined approach, in which information can be transfered from the ghost cells to the internal mesh cells where appropriate. Thus, the only AMR cells that require specific attention here are the flow field peripheral cells of the finest grid level. These cells are primed with data immediately following their identification, during the retracing of the cut cells. This enables the search for the overlying boundary-fitted grid cells, from which the AMR cells' solutions are interpolated, to be focussed on the likeliest regions of the data storage. The solutions of the boundary-fitted ghost cells are interpolated from the underlying AMR solution. A first order interpolation scheme is used for all the results presented here. First order interpolation has the advantages that it is simple to implement and will not directly cause unphysical oscillations to appear near discontinuities. It should not be too difficult to replace the interpolation scheme with either a high order limited one, based around TVD limiting, or with a finite volume one. Better

interpolation techniques are presented in [20, 74] The transfer of data between the AMR and boundary-fitted grids is described in sections 4.6.2 and 4.6.1.

4.6.1 Transfer of Solution Data from the AMR to the Boundary-fitted Grid Structures

Consider figure 4.8 which depicts two adjacent boundary-fitted meshes, shown together and apart. The shaded cells are the internal mesh cells and the transparent cells, formed by the dashed lines, are the ghost cells. There are three types of ghost cell; the first two types lie along the mesh sides that are parallel to the boundary, whilst the third type lie along the mesh sides that are normal to the boundary. Generally the ghost cells of the first type lie within the flow field and their solutions can be obtained by interpolating the solution data of the finest level of the underlying AMR grid structure. Those of the second type lie inside the solid boundaries. Their solutions are obtained by reflecting the internal solution data so that there is no flow normal to the boundary. The picture of the separated meshes in figure 4.8, reveals the first and the third type of ghost cells between any two meshes; the second type of ghost cells are not shown. Where a neighbouring mesh exists (as shown in figure 4.8), the third type of ghost cell is primed with the solution data of the coincident cell within the neighbouring mesh. Where a mesh abuts an external boundary of the problem domain, as with open ended boundaries, (e.g. a simple wedge problem), transmissive or reflective boundary conditions are imposed such that there is either undisturbed or no flow respectively across the boundary. The identification of the underlying AMR cells is the only source of difficulty associated with updating the boundary-fitted ghost cells. The solutions of neighbouring boundary-fitted ghost cells are likely to be interpolated from the solutions of AMR cells within the same mesh. In situations where this is not the case, the relevant cells of the adjacent mesh can be identified in a similar way to one that determines the AMR ghost cells. Thus, the properly nested AMR grid structure aids the efficiency of the identification process.

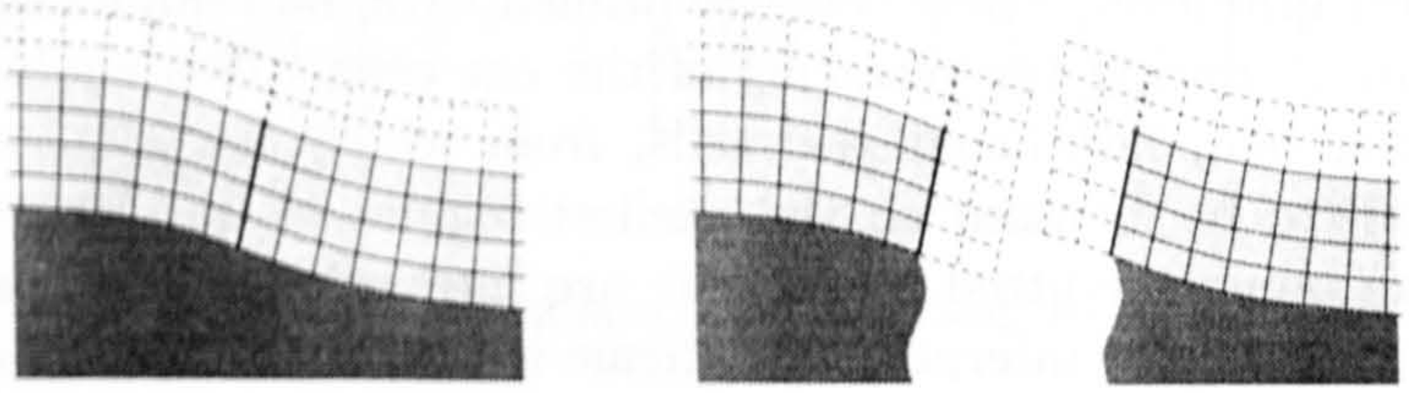


Figure 4.8: Two adjacent boundary-fitted meshes, shown together (left) and apart (right).

4.6.2 Transfer of Solution Data from the Boundary-fitted to the AMR Grid Structure

The identification of the boundary-fitted grid cells that cover the AMR peripheral cells is an iterative process. Given an initial estimate, a very fast algorithm identifies the correct covering boundary-fitted cells. The peripheral cells are identified in a clockwise sequence, during the retracing of the cut cells. Thus, for every peripheral cell, a covering cell of the previous peripheral cell provides an initial estimate. The estimate for the first peripheral cell associated with a body is taken to be the first cell of the boundary-fitted grid. Generally, the estimations are accurate, leaving only minor corrections to determine the actual covering cells.

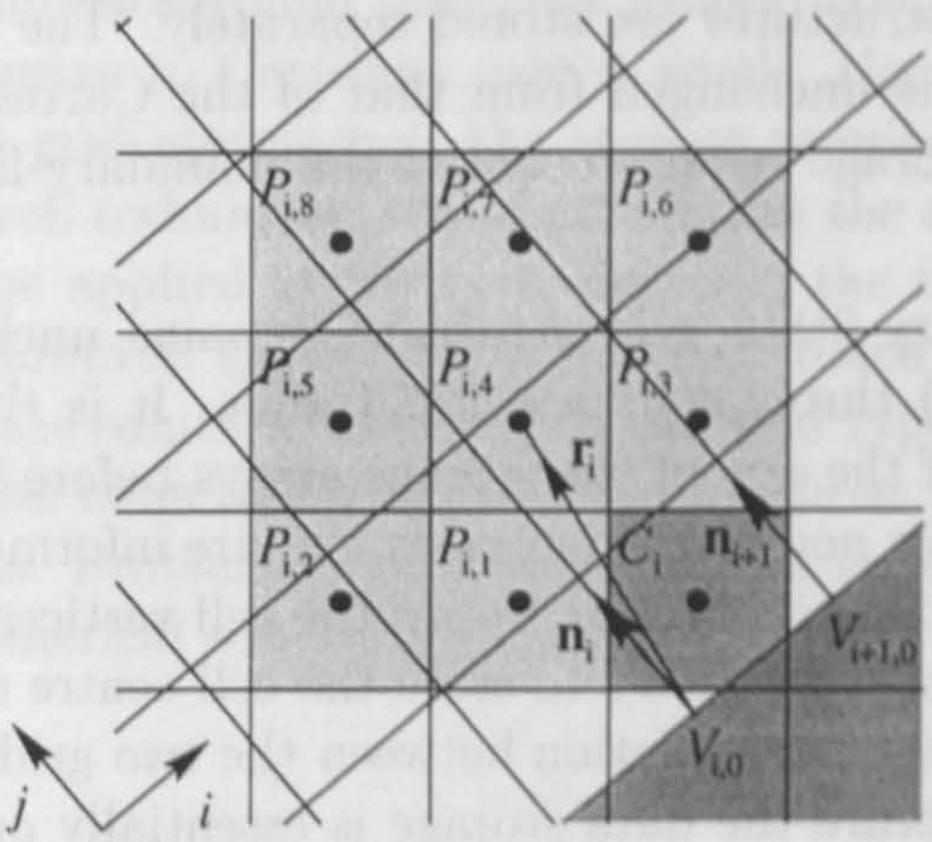


Figure 4.9: Boundary-fitted grid coverage of underlying flow field peripheral cells.

The iterative process is best illustrated by the following example. Consider the situation shown in figure 4.9, where the Cartesian cell C_i is cut by the line joining the boundary vertices $V_{i,0}$ and $V_{i+1,0}$ of the boundary-fitted grid structure. During the retracing procedure, described in section 4.5, the eight flow field peripheral cells $P_{i,n}$ $(n = 1, 8)$ are defined. The i coordinate of the covering boundary-fitted grid cell is determined by taking the vector product (cross) of the direction vectors $\mathbf{r}_i$ and $\mathbf{n}_i$. The vector $\mathbf{r}_i$ is the cell centre of $P_{i,n}$ relative to the vertex $V_{i,0}$ and the vector $\mathbf{n}_i$ is the vector along the normal intercell boundary from $V_{i,0}$. The i coordinate is incremented either positively or negatively, depending on the sign of $\mathbf{r}_i \times \mathbf{n}_i$, until the vectors $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ are both bounded by the normal vectors $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$. Hence,

$$(\mathbf{r}_i \times \mathbf{n}_i) \cdot (\mathbf{r}_{i+1} \times \mathbf{n}_{i+1}) < 0.$$

Having determined the i location, a similar vector product procedure is used, (in which tangential vectors, with origins $V_{i,j}$ for all j, along the intercell boundaries parallel to the solid boundary replace the normal vectors $\mathbf{n}_i$), to establish the j

location. Once the i, j coordinates have been found, the locations within the data structure of the relevant boundary-fitted grid cells' solutions can be easily found, (identical to that for the AMR cells; see section 3.5). The solution data for $P_{i,n}$ can then be obtained using a suitable interpolation technique.

4.7 Storage Requirements

The storage requirements for Chimera approaches with Cartesian base grids are significantly less than those for curvilinear or unstructured grids; only Cartesian cut cell techniques require less data storage. The data associated with the AMR and boundary-fitted grid structures are stored separately. The AMR storage in the combined CAMR approach is unchanged from that of the Cartesian AMR approach. This section details the storage requirements of the boundary-fitted grid structure and solution.

Generally, the boundary-fitted grid structure remains unchanged (except for translations and rotations) throughout a computation. It is therefore possible to make accurate estimates of the size of the storage arrays before initiating a computation. Numerical solvers for non-Cartesian grids, require information about the cell geometries. The algorithm presented here, stores the cell vertices and cell centre coordinates. It is not absolutely necessary to store the cell centre coordinates, but by doing so, the efficiency of the interpolation between the two grid types is improved. As with the AMR grid structure the data storage is essentially one-dimensional, i.e. all the cell solutions are stored in a single two-dimensional array, in which the solution vector (conserved variables) is referenced by one of the indices. Linked lists provide access to the cell and ghost cell solutions, vertex coordinates, etc. For every mesh, an array of pointers, indicates the locations of the first cell, vertex, etc., within the storage lists. This mesh pointer array also contains the mesh dimensions, in terms of the number of cells along the boundary, and the mesh type for every mesh. Thus, sufficient information is available, given the coordinates (i, j) within a mesh, to find all the necessary associated data, (see section 3.5 which describes the identical process for the AMR cells). For the Cartesian AMR algorithm with Riemann problem flagging, an extra array is needed to store the flags. The CAMR algorithm stores several different types of flags in the same array, without any extra storage costs.

The ratio of the number of boundary-fitted cells to the number of AMR cells, will decrease with increasing resolution. Hence, the extra storage costs associated with the combined CAMR approach will become less significant as the resolution increases. Furthermore, the proportion of the flow domain occupied by the finest AMR level and boundary-fitted cells will decrease with increasing grid resolution. Thus, the theoretical reductions in memory storage over the equivalent regular grid

are inversely proportional to the grid resolution.

4.8 Recap and Summary

An algorithm for combining the Chimera and the AMR approaches has been presented. The combined technique consists of a Cartesian background grid, that can be refined around flow and boundary features, with overlying curvilinear grids that are fitted to the irregular boundaries. The Chimera approach presented here was designed specifically for its implementation into the AMR algorithm, with the aims that it is both accurate and efficient. The representation of the boundaries improves by refining the background grid. For general computations the ratio of the number of non-Cartesian to Cartesian cells is small. Hence, compared to irregular and boundary-fitted grid approaches, the storage requirements are small. Unlike most Cartesian cut-cell techniques, the approach has the advantage that high order schemes can easily be applied at the boundary and the time step is not restricted by any small cells[1]. However, the accuracy of the approach is limited by the errors in the interpolation of the solution between the two grid types. In an attempt to minimise these errors, the boundary-fitted cells are made as similar in size as possible to the Cartesian cells. Problems that involve moving boundaries can be computed, by modifying the numerical scheme accordingly.

[1]Most, but not all, cut-cell techniques employ small cell absorption, which produce some cells that are almost 50% smaller than the uncut cells and others that are almost 50% larger.

Chapter 5

Validation and Assessment

5.1 Introduction

The aims of this chapter are to validate and assess all the aspects of the complete CAMR algorithm. The chapter is split into three separate sections. The first section contains the results for several different 1D test problems. These tests enable the various aspects of the numerical schemes to be assessed away from the more complicated algorithms, which may introduce their own errors. In section 5.3, two different 2D test problems are computed with the Cartesian AMR algorithm. The computed solutions are compared with other computational and experimental results. The validation results for the complete CAMR algorithm are presented in section 5.4. For all the test problems in this section, the WAF scheme [114] is used in conjunction with the HLLC approximate Riemann problem solver [119]. Unless stated otherwise, the SUPERBEE flux limiter is used for all waves except shear waves, for which the VAN LEER limiter is used. The performance of the AMR and CAMR algorithms are assessed by comparing the CPU times with those of an equivalent resolution regular grid. The relative costs of the various parts of the algorithms are obtained using standard profiling tools. (All the timings and profilings in this chapter make use the UNIX commands `time` and `gprof`.)

5.2 1D Test Results

This section contains the results for three one-dimensional test problems. All three tests involve an initial discontinuity between two constant states, i.e. a Riemann problem. The exact solution to the Riemann problems provide a comparison for the numerical solutions. The first two tests are used to assess the ability of the solver strategy to calculate solutions to the Euler equations, with the ideal and constant

covolume equations of state. The third test enables an assessment of solutions computed on moving grids to be made.

5.2.1 Sod's Test Problem

Sod's problem [93] is a standard test used for assessing numerical solutions to the one-dimensional Euler equations with the ideal equation of state ($b = 0$ and $\gamma = 1.4$). The domain is of unit length and is discretised by 100 computational cells. The initial conditions involve a step discontinuity, positioned at the centre of the domain, with the following left and right states

$$\begin{aligned} \rho_L &= 1.0, \quad p_L = 1.0, \quad u_L = 0.0 \\ \rho_R &= 0.125, \quad p_R = 0.1, \quad u_R = 0.0 \end{aligned}$$

The problem was computed using the WAF scheme with the HLLC approximate Rie-

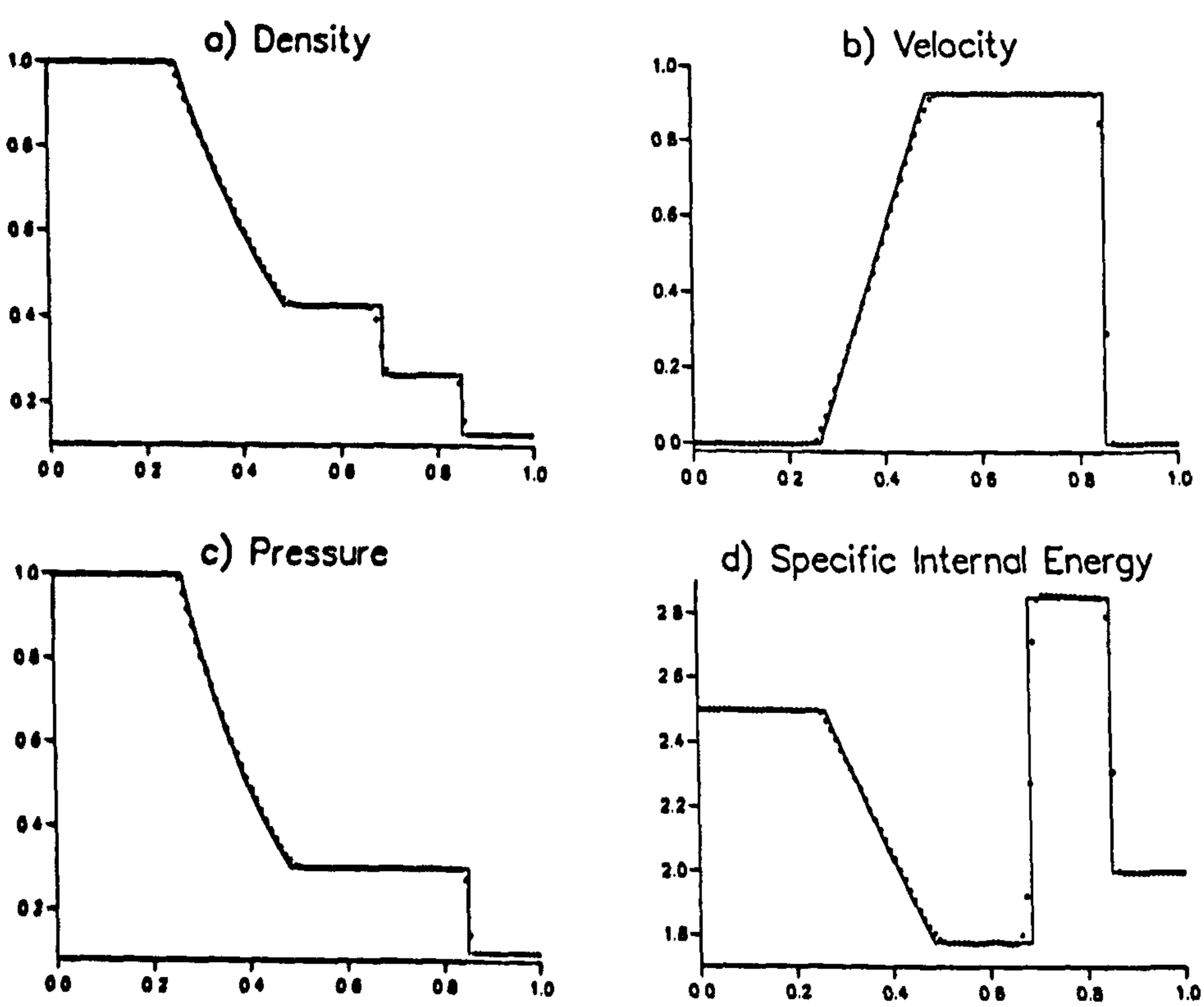


Figure 5.1: A comparison between the numerical (symbols) and the exact (full lines) solutions for Sod's problem at $t = 0.2$.

mann solver and the SUPERBEE limiter. Figure 5.1 shows four graphs representing the spatial profiles of the density, velocity, pressure and specific internal energy at time $t = 0.2$ units. The numerical results (symbols) appear to 'capture', very well, the discontinuous features of the exact solution (full lines). The shock wave is represented by just two points, which is typical of good second order Godunov-type schemes. The representation of the contact wave, which is only apparent in the density and specific internal energy graphs, always tends to be less accurate. However, the three point representation of the contact wave in these results, compares very well with those of other numerical solutions.

5.2.2 Einfeldt's Test Problem

The graphs in figure 5.2 represent the spatial profiles, at time $t = 0.2$, of the density, velocity, pressure and the specific internal energy for another shock-tube test problem. The test, which was first computed by Einfeldt [31] has the same initial

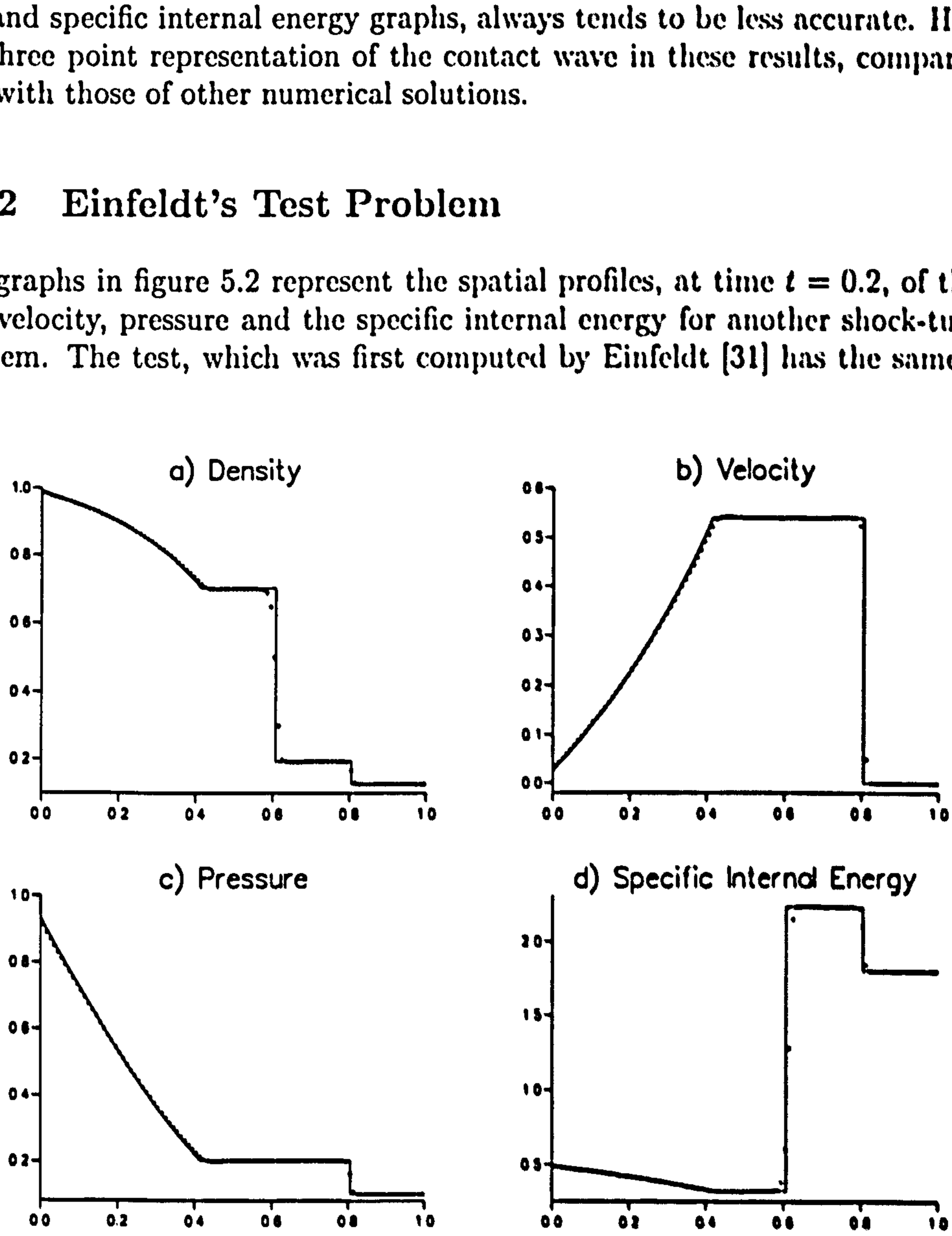


Figure 5.2: A comparison between the numerical (symbols) and the exact (full lines) solutions for Einfeldt's problem at $t = 0.2$.

conditions as Sod's problem, (computed previously), but with a constant covolume of $b = 0.8$ (compared to $b = 0$ for Sod's test). The results were computed with the same solver strategy and ratio of the specific heats $\gamma = 1.4$. There is clearly a very good comparison between the numerical solution (symbols) and the exact solution (full lines). Even though the covolume in Einfeldt's test is unrealistically large, it is interesting to compare the results in figures 5.1 and 5.2.

5.2.3 Sod's Test Problem on a Moving Grid

The graphs in figure 5.3 show the density solutions for Sod's test computed on four grids that are moving with different velocities. Again the results compare well with

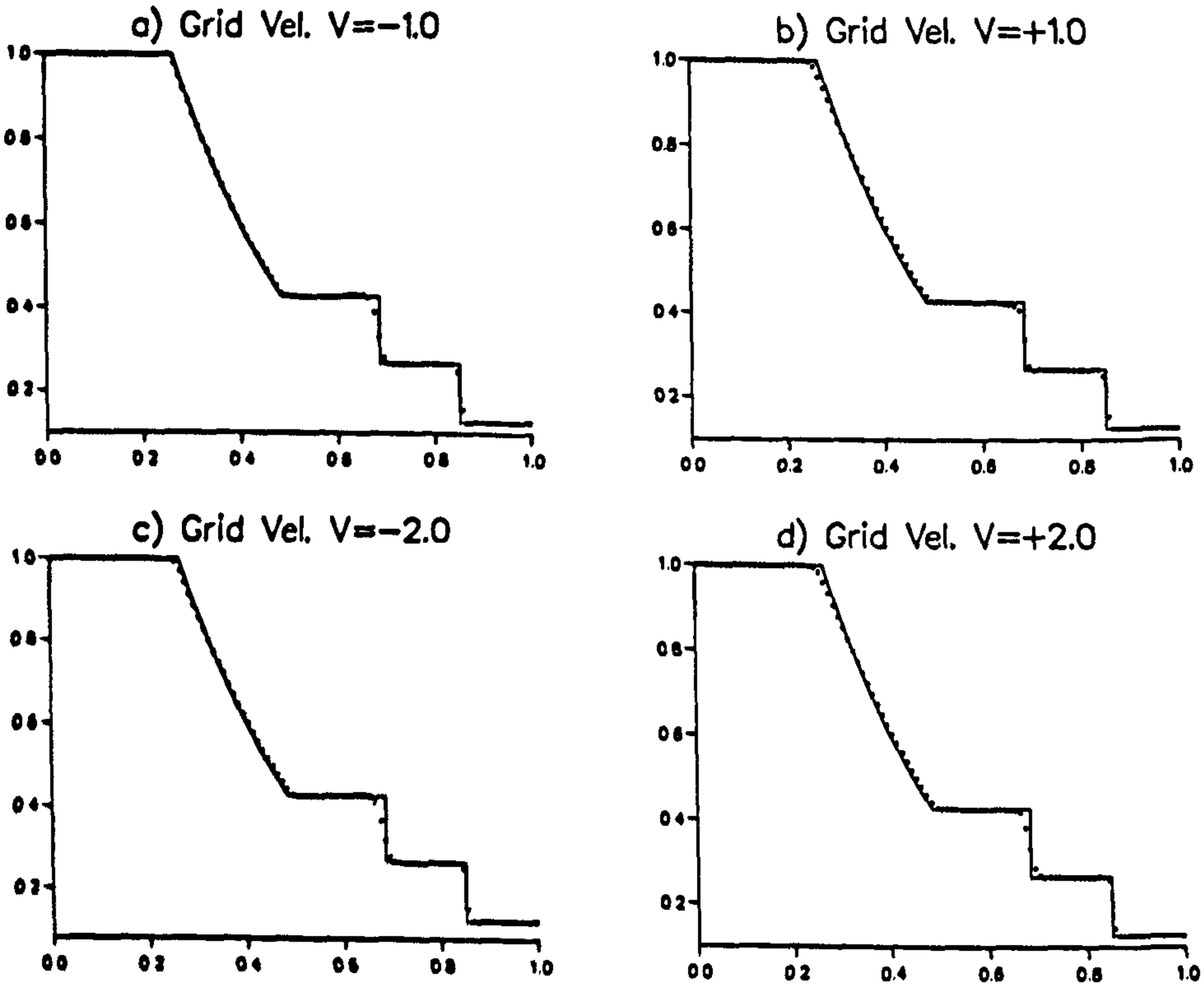


Figure 5.3: A comparison between the numerical (symbols) and the exact (full lines) density solutions for Sod's problem on a moving grid.

the exact solutions. Solutions computed with the WAF scheme vary with the difference between the wave and grid velocities. For example, linear advection solutions computed with the WAF scheme are exact when the Courant number $\nu = 0$, i.e. when

the wave propagation velocity is equal to the grid velocity. The representation of the contact wave for each graph in figure 5.3, clearly demonstrates this point. There are fewer points in the contact wave in graph b, where the wave and grid velocities are similar, than in the contact waves of the other graphs.

5.3 2D AMR Test Results

The aim of this section is to assess the AMR algorithm described in chapter 3. Two tests are presented along with the various computed results, timings and profilings. The first test involves the advection, by a vortex field, of an offset Gaussian distribution. The second test is a more realistic one of a shock wave diffraction around a corner. The timings and profilings enable the reader to properly gauge the efficiency of the algorithm.

5.3.1 Advection of a Gaussian Distribution by a Vortex Field

This test [71] involves a vortex field centred in a square domain with side lengths of 10.0 units. The vortex advects an offset Gaussian[1] distribution ϕ, which appears as a 'bell-shaped' blob, but does extend throughout the whole flow field and is described by

$$\phi = e^{-\frac{1}{2}(r_a/\sigma)^2}, \tag{5.1}$$

where r_a is the radial distance from the centre of the distribution and σ is the standard deviation.

The problem is described by the two-dimensional, variable coefficient linear advection equation, which is written as

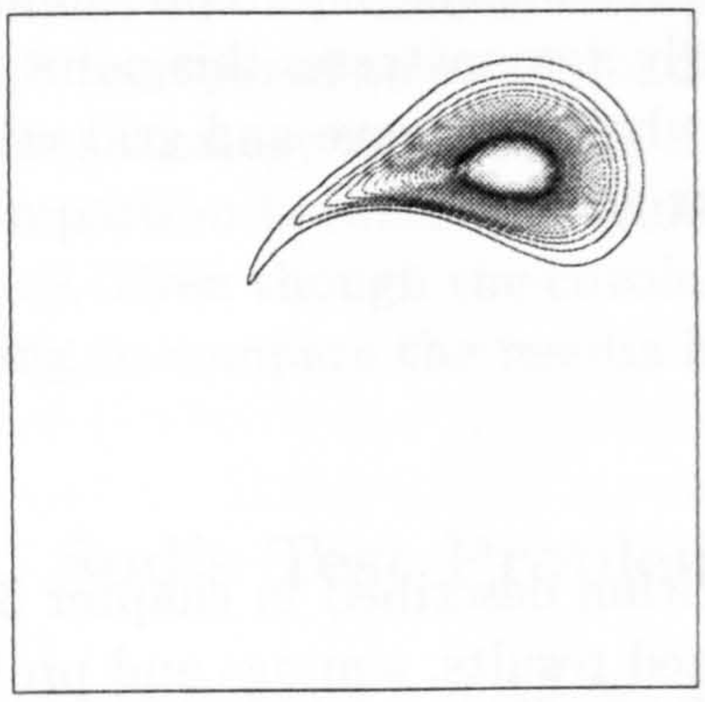
$$u_t + au_x + bu_y = 0, \tag{5.2}$$

where u represents the advected quantity equal to ϕ for this problem, and the coefficients a and b represent the wave speeds in the x and y directions respectively. The coefficients a and b at the point (x, y) within the vortex field are given by

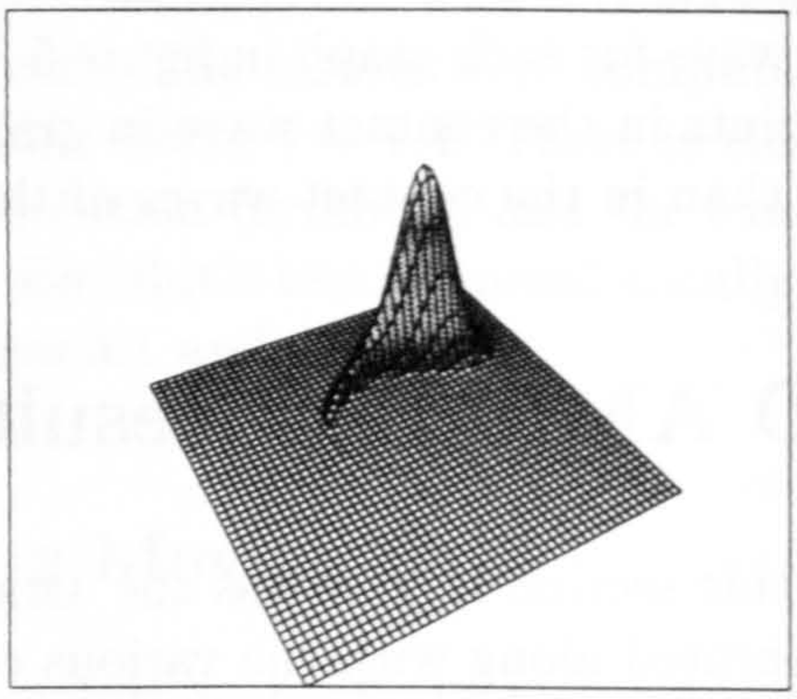
$$a = \frac{y_v - y}{rV'} \frac{\sinh(r_v)}{\cosh^3(r_v)} \quad \text{and} \quad b = \frac{x_v - x}{rV'} \frac{\sinh(r_v)}{\cosh^3(r_v)},$$

where r_v is the radial distance from the centre of the vortex at (x_v, y_v) and V' is the maximum speed. The tests were computed with $\sigma = 0.3$ and $V' = 1.0$.

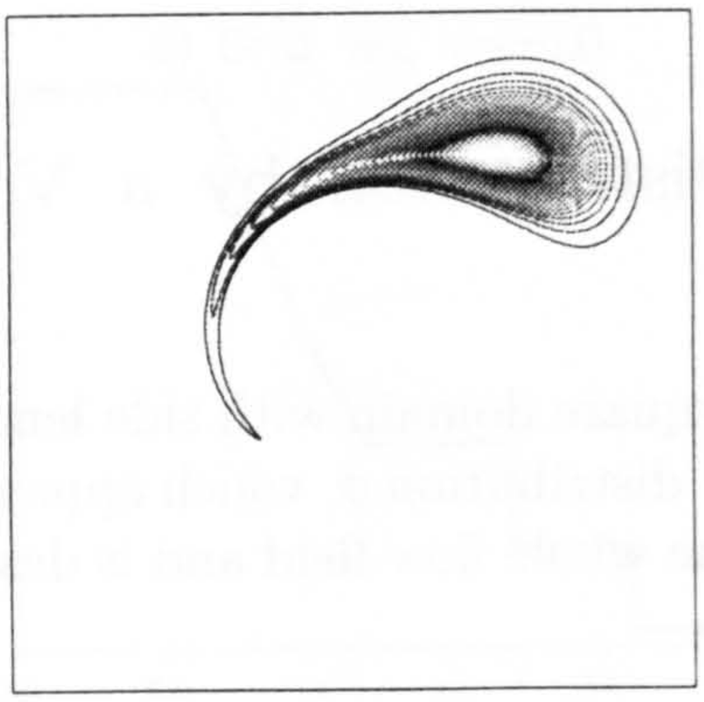
Figures 5.4 and 5.5 both show the computed solution at three different times.



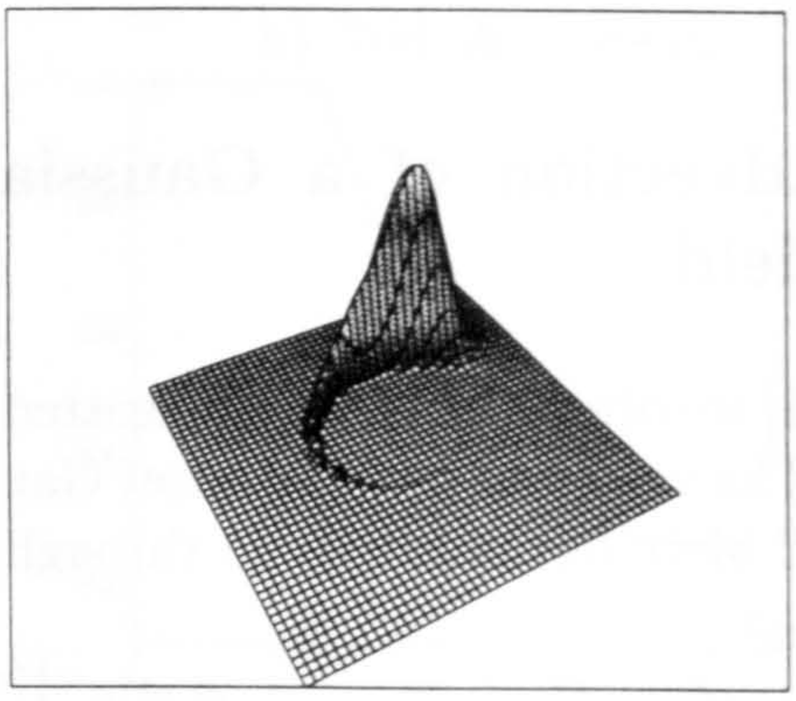
a) Solution (t=5.0)



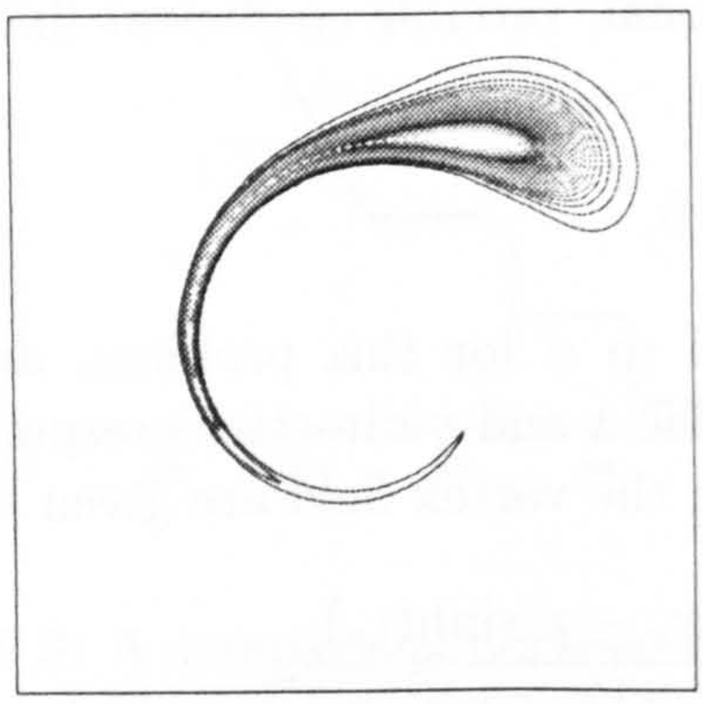
b) 3D Solution (t=5.0)



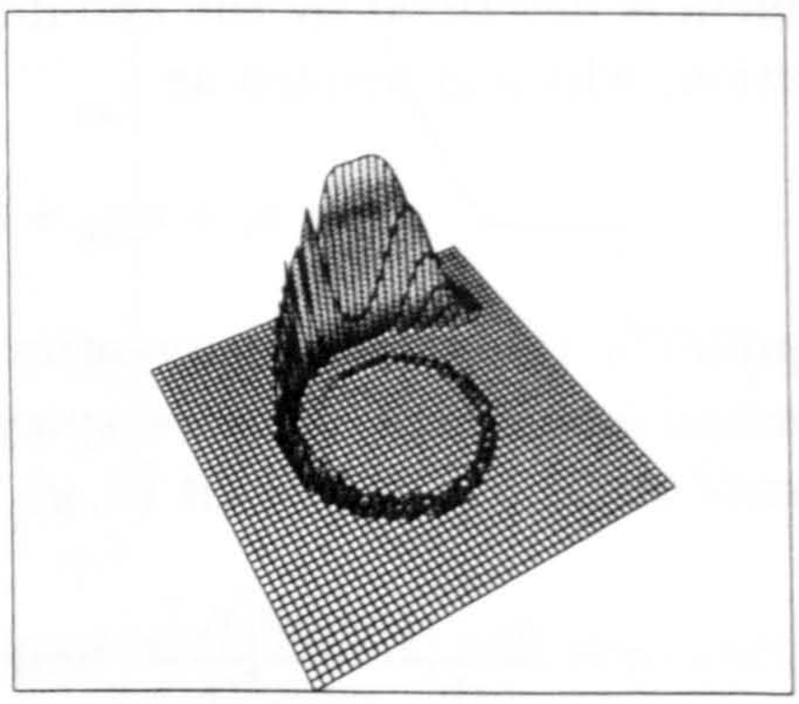
c) Solution (t=10.0)



d) 3D Solution (t=10.0)

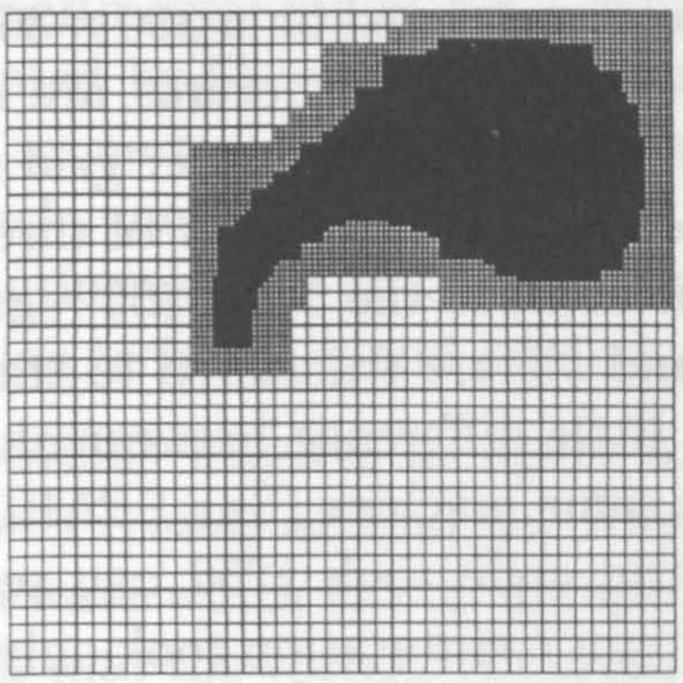


e) Solution (t=20.0)



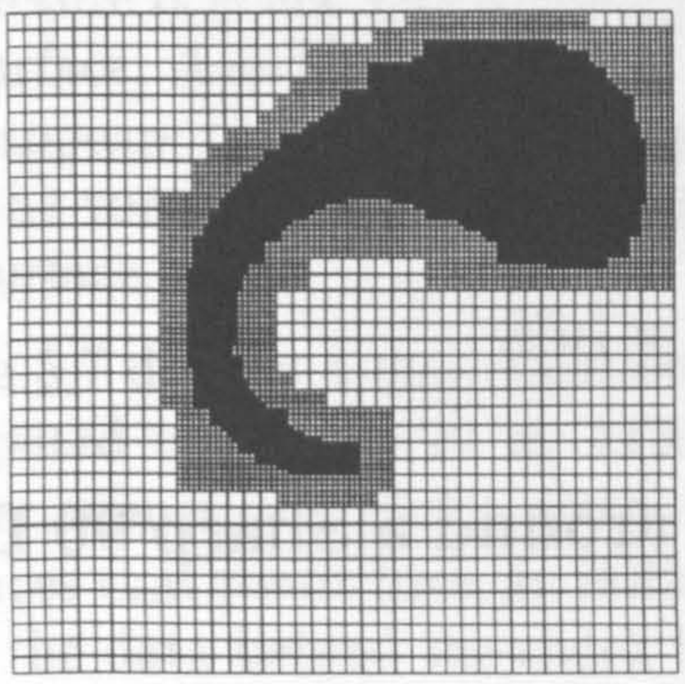
f) 3D Solution (t=20.0)

Figure 5.4: Advection of a Gaussian distribution computed with a regular grid.

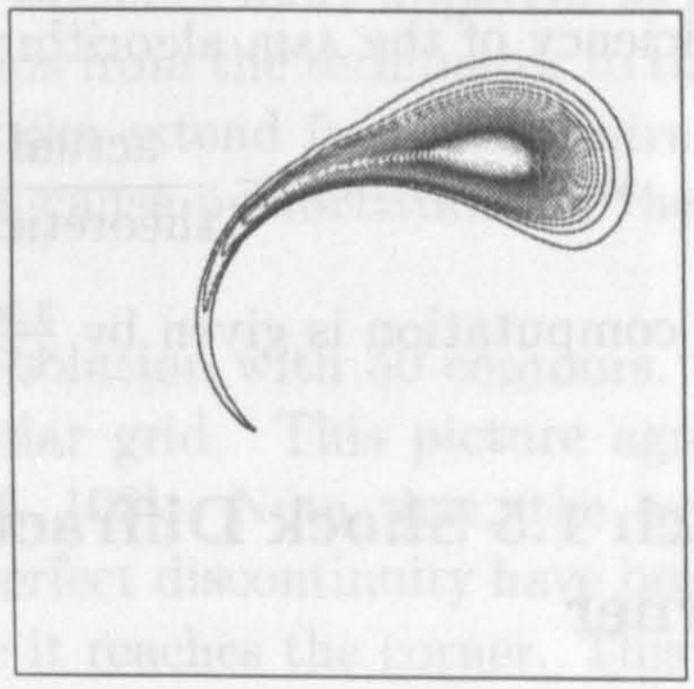


a) Grid Structure (t=5.0)

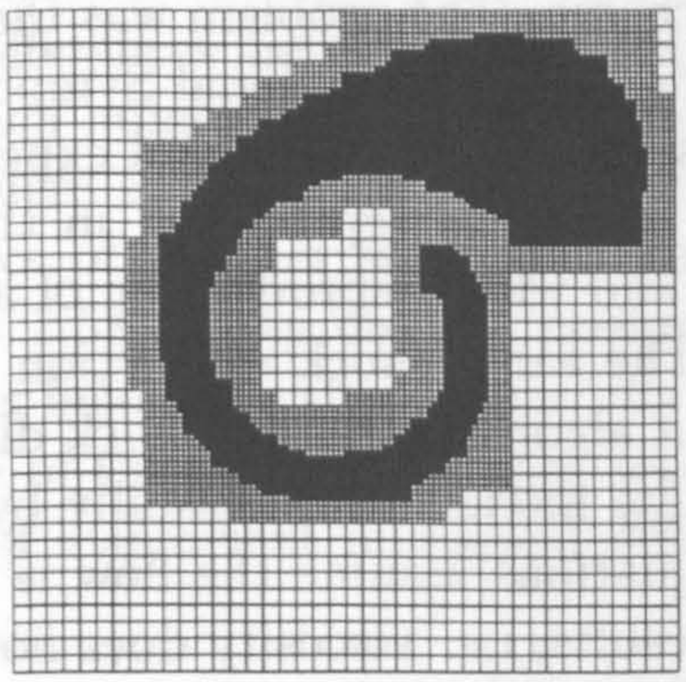
b) Solution (t=5.0)



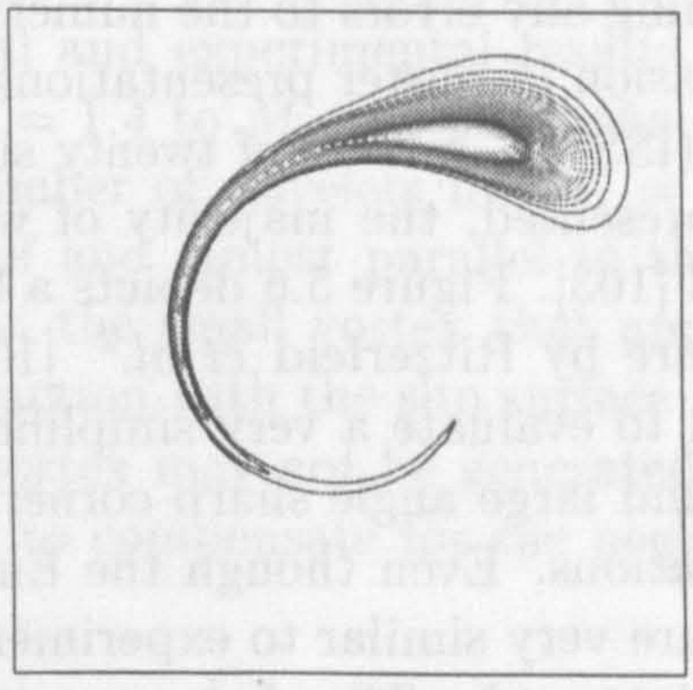
c) Grid Structure (t=10.0)



d) Solution (t=10.0)



e) Grid Structure (t=20.0)



f) Solution (t=20.0)

Figure 5.5: Advection of a Gaussian distribution computed with an AMR grid.

Figure 5.4 shows the 2D contour and 3D solutions computed on a 360×360 regular grid. The AMR solutions along with the associated grid structures are shown in figure 5.5. The AMR computations were computed with a three layer grid structure consisting of a 40 by 40 cell base mesh with the a refinement factor of three for the finer two levels, which for the finest level, is equivalent to the 360×360 regular grid. Even if the contour plots shown in figures 5.4 and 5.5 are transfered to transparent film and overlayed on each other, there is very little difference between them. The two sets of computations were computed on the same machine (Sun SPARC 2) with the same level of optimisation. The regular grid code took 1252 CPU seconds to compute the solution to time $t = 20$, whilst the AMR computation took only 373 CPU seconds. Thus, the AMR computation was 3.36 times faster than the regular grid computation. However, for any problem there is a theoretical maximum speed-up factor given by the number of regular cell integrations divided by the number of AMR cell integrations. For this problem the theoretical maximum speed-up factor is 4.42. Hence, the efficiency of the AMR algorithm can be expressed as:

$$\text{AMR efficiency} = \frac{\text{actual speed-up factor}}{\text{theoretical speed-up factor}} \times 100,$$

which for this computation is given by $\frac{3.36}{4.42}.100 \approx 76\%$.

5.3.2 Mach 1.5 Shock Diffraction around a Sharp 90 Degree Corner

The diffraction of a Mach 1.5 shock wave around a sharp 90 degree corner is an established test problem for numerical computations. It is a good test for comparing numerical schemes because the domain can be represented by a Cartesian grid, thereby confining any errors to the numerical scheme. The problem was the subject of a special session of poster presentations at the 18th International Symposium on Shock Waves (ISSW). A total of twenty six sets of computational and experimental results were presented, the majority of which were later reproduced in the Shock Waves journal [103]. Figure 5.6 depicts a black and white reproduction of an colour schlieren picture by Ritzerfeld *et al.*[2] [103]. The same shock diffraction problem has been used to evaluate a very simplified AMR algorithm [121]. Shock diffraction problems around large angle sharp corners can be modelled successfully by solving the Euler equations. Even though the Euler equations assume the flow is inviscid, the solutions are very similar to experimental results, i.e. viscosity does not appear to play a dominate role. The sharpness and the large corner angle ensures that the flow separates automatically and therefore does not require the point of separation to be specifically calculated.

[1] Note that 5.1 is a reduced form of a true Gaussian distribution.

[2] Shock Wave Laboratory, RWTH Aachen, Germany.

An experimental study for a range of initial Mach numbers and corner angles was presented by Skews [91]. More recently, Hillier [54] and Kleine *et al.* [59] have presented numerical and experimental results around 90 degree corners, for various initial Mach numbers. Hillier used a second order Godunov-type scheme and a high resolution grid to obtain very accurate solutions. Kleine *et al.* used advanced visualisation techniques and numerical results to study the small-scale features.

The obvious features in figure 5.6 are the incident shock wave, which curves as the flow expands around the corner and the vortex. At the corner there is an expansion wave which has an abrupt tail (terminator) and the start of a slip surface which rolls into the vortex. A curved acoustic wave is reflected from the corner and for this problem[3] is able to move upstream along the inlet channel. The contact wave coincides with the slip surface a little down stream of the corner and curves around the vortex up to a single juncture with the acoustic and incident shock waves. Near to the juncture it is not very visible, but it becomes more apparent as it approaches the slip surface. A second shock wave extends from the terminator to the slip surface where it appears to reflect. Two other shocks extend from the centre to the outer edges of the vortex. The interacting shocks cause perturbations in the curvature of the slip surface.

Figure 5.7 depicts a numerical density solution with 50 contours. The solution was computed using a 1120 by 1120 regular grid. This picture agrees very well with other numerical results given in [54, 103]. Note that, the weak 'start-up' disturbances that result from an initially perfect discontinuity have been reduced by resetting the data behind the shock before it reaches the corner. This is a common technique, as is using an initial shock profile that is smeared over two or three grid cells, as determined from a prior one-dimensional solution [54]. There are some slight differences between the numerical solutions and the experimental results, that appear adjacent to the slip surface, near to the corner. Kleine *et al.* [59] reported similar differences between their numerical and experimental results, for incident shock Mach numbers M_s in the range $M_s = 1.4$ to $M_s = 1.6$. In the experimental results, there appeared to be a greater number of wavelets for $M_s = 1.45$ and the appearance of another shock just ahead of and almost parallel to the terminator for $M_s = 1.58$. They did not comment on the small vortex that appears next to the corner below the slip surface. Its interaction with the slip surface could well be the cause of the differences. The second vortex may not be generated numerically, because of insufficient numerical diffusion to compensate for the neglected viscous effects.

Figures 5.8 and 5.9 depict the grid structure and density contours computed with the AMR algorithm. The resolution is equivalent to the one shown in figure 5.7; a 70 by 70 base grid and two further levels of refinement, each with a refinement factor of

[3]The post shock flow is subsonic.

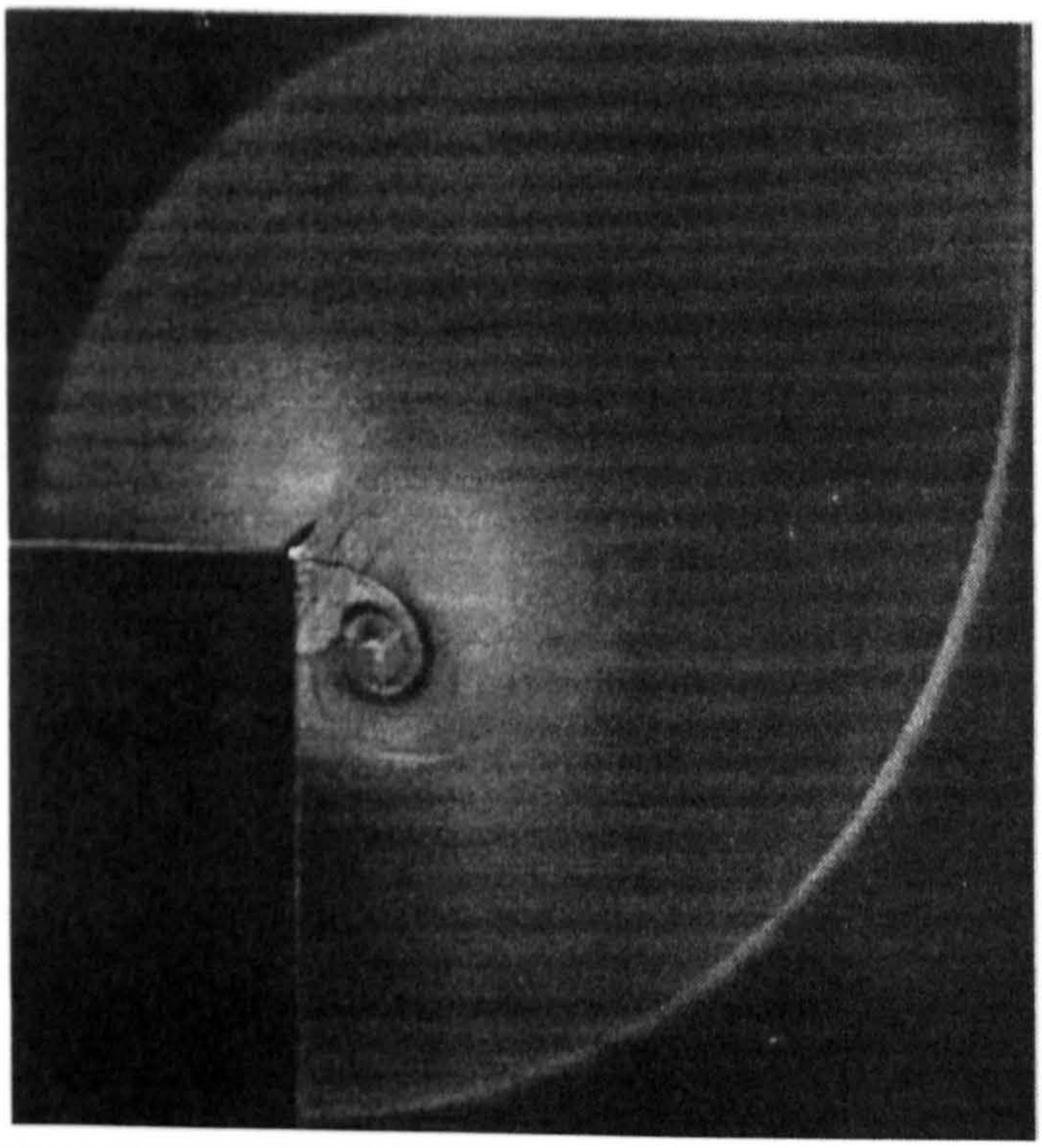


Figure 5.6: Experimental Schlieren picture of a Mach 1.5 shock diffraction.

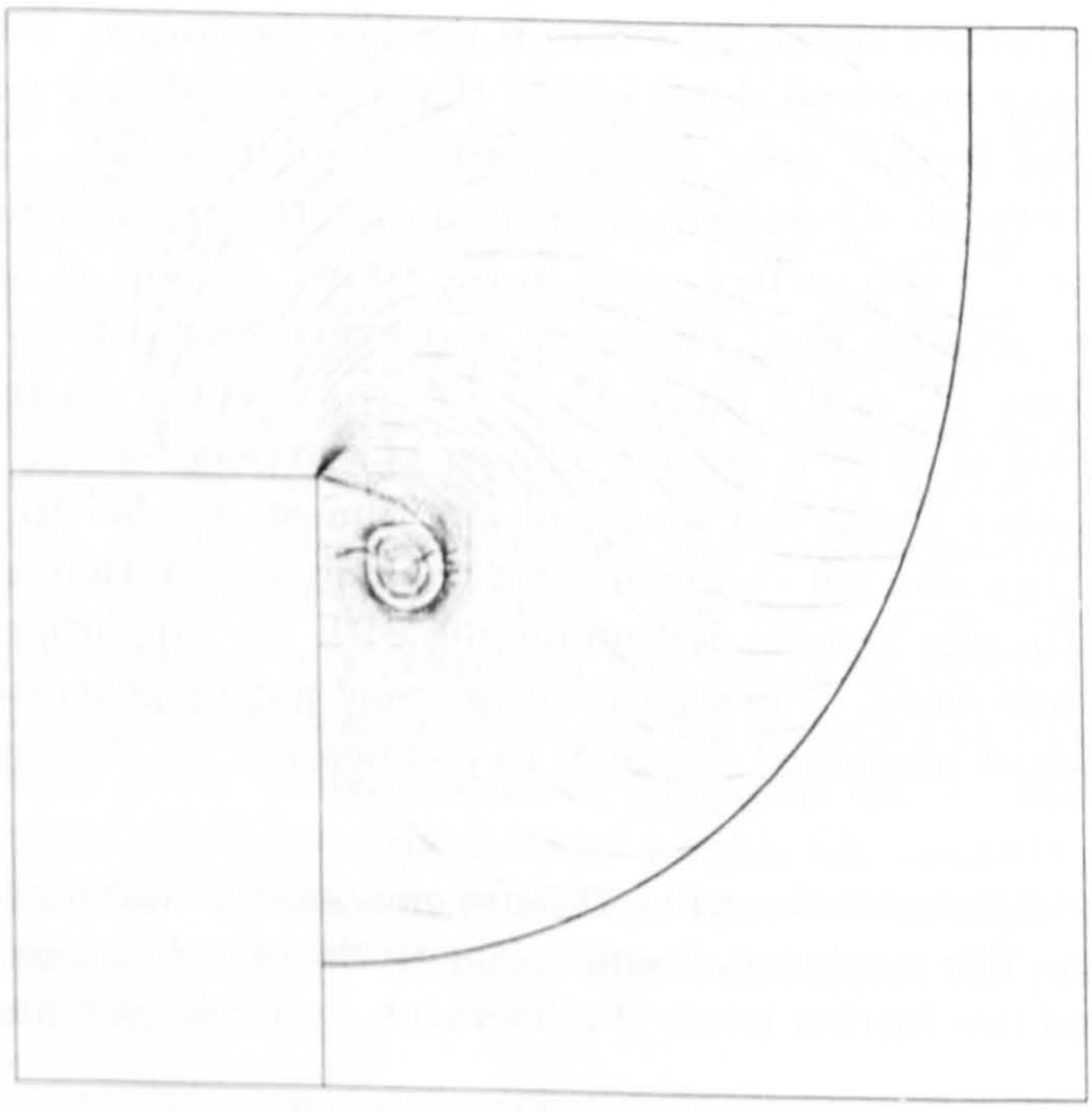


Figure 5.7: Shock diffraction density contours (50) for a regular 1120×1120 grid.

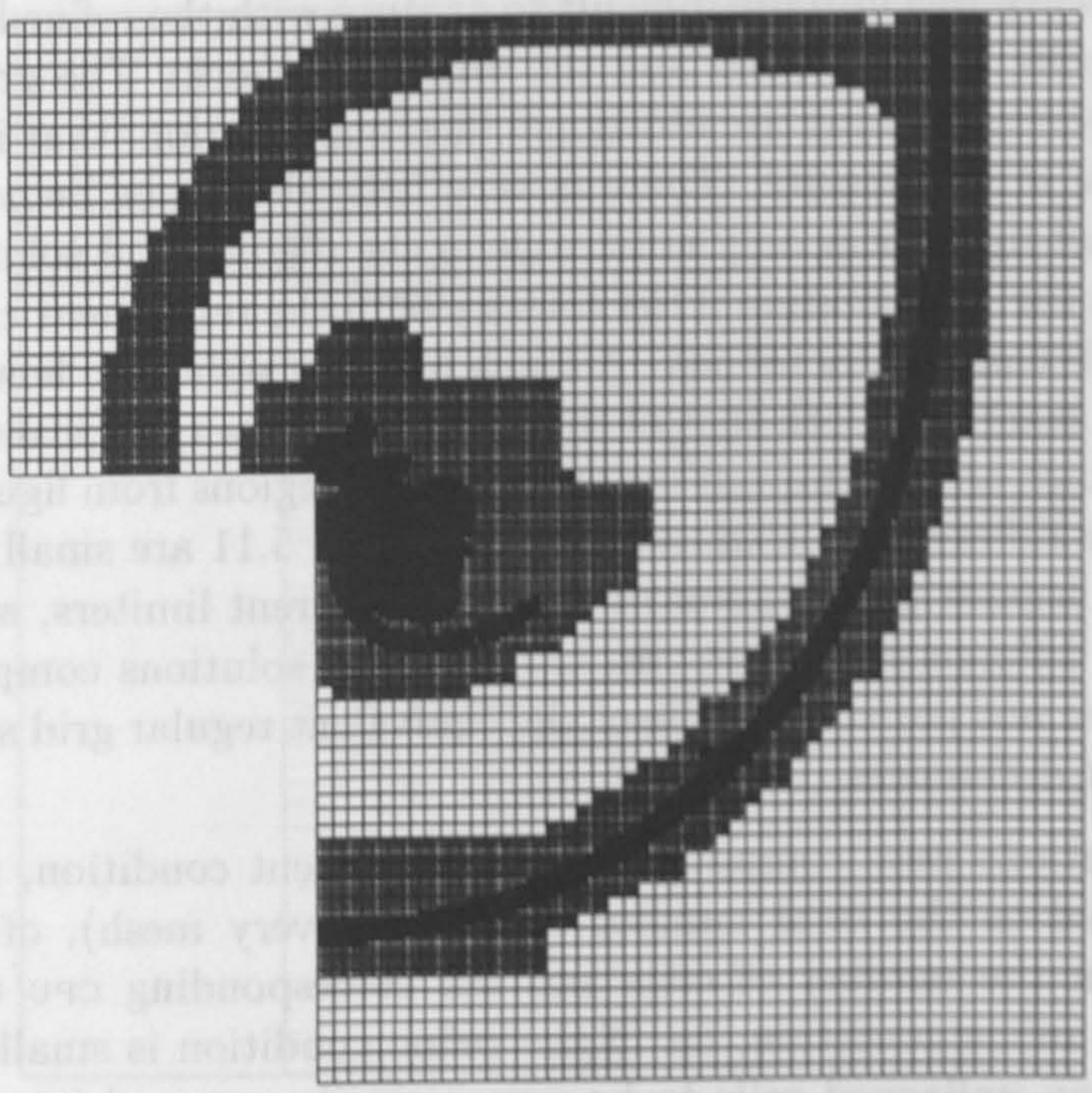


Figure 5.8: The AMR grid structure for the shock diffraction solution in figure 5.9.

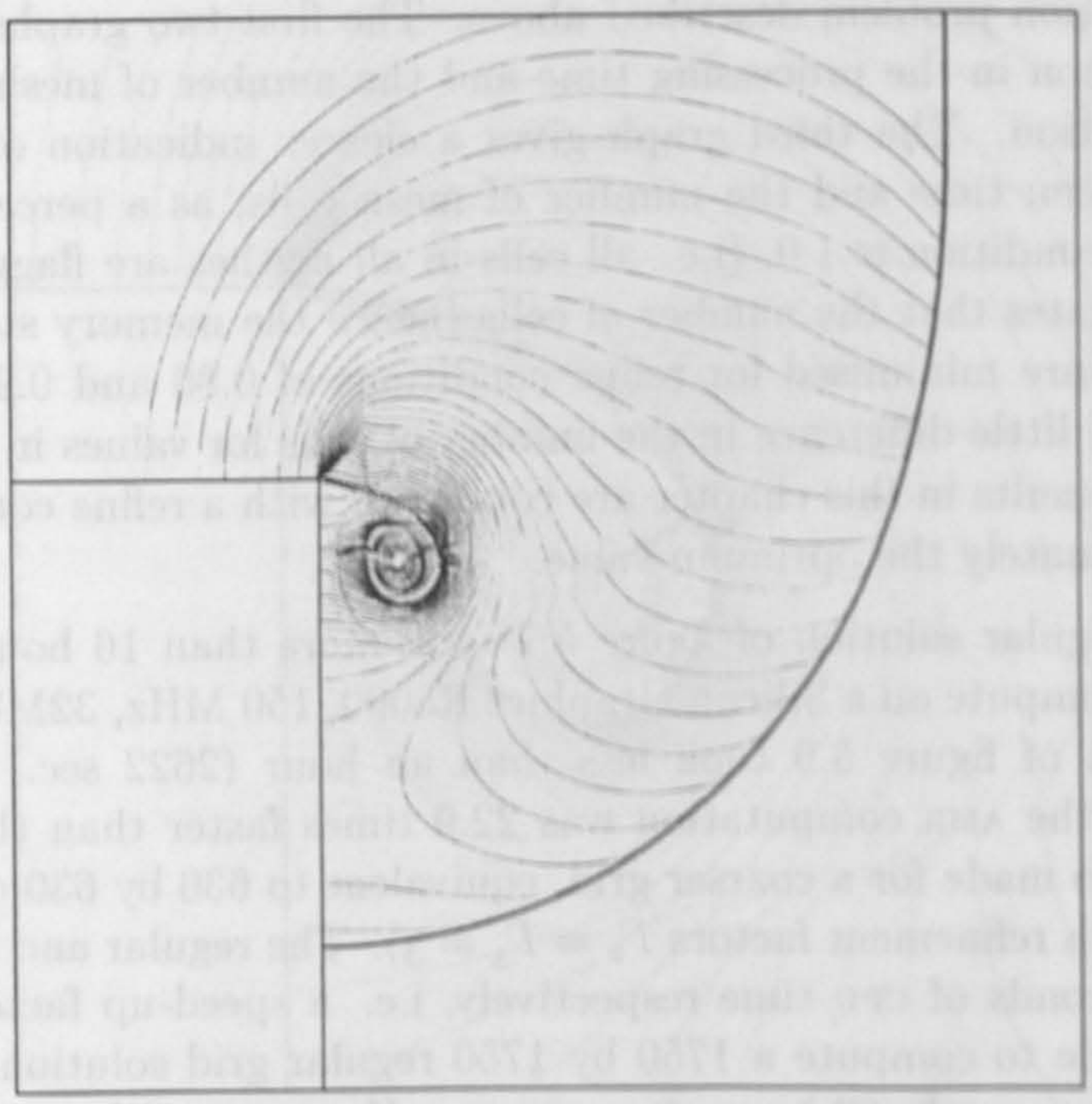


Figure 5.9: Shock diffraction density contours (50) computed on an AMR grid.

4. The acoustic wave was initially difficult to capture with the refined grids, without refining all the cells between itself and the diffracted shock. The grid structure in figure 5.8 was generated using an additional flagging criterion, that is based on the second derivative in the density. There is very little difference between figures 5.7 and 5.9. The start-up error is only noticeable in the regular grid solution. However, it does appear in the AMR solution if the tolerances in the refinement criteria are very small, so that it is continually refined. A much better insight into how the quality of the AMR solution compares with that of the regular grid, can be gleaned from figures 5.10 and 5.11, which are magnifications of the vortex regions from figures 5.7 and 5.9 respectively. The differences between figures 5.10 and 5.11 are small. It is interesting to note that regular grid computations using different limiters, would generally produce greater differences. Therefore, although the solutions computed with the AMR algorithm are not perfect replicas of the equivalent regular grid solutions, there are no significant differences.

The AMR solutions were computed with a refinement condition, (the minimum fraction of flagged to the total number of cells in every mesh), of 0.90. In any AMR computation the number of cells and the corresponding CPU time will vary according to the refine condition, i.e. if the refine condition is small, then there is potential for many unflagged cells to be unnecessarily grouped into mesh patches and subsequently integrated. The relationship between the refine condition and the computing costs (processing time and memory storage) was investigated for the shock diffraction problem described above. The first two graphs in figure 5.12 depict the variation in the processing time and the number of mesh cells with the refinement condition. The third graph gives a clearer indication of the costs. It shows both the CPU time and the number of mesh cells, as a percentage of those when the refine condition is 1.0, (i.e. all cells in all meshes are flagged cells). The third graph indicates that the number of cells (hence the memory storage) and the processing time, are minimised for refine conditions of 0.80 and 0.95 respectively. However, there is little difference in the number of cells for values in the range 0.75 to 0.90. All the results in this chapter are computed with a refine condition of 0.90, which is approximately the optimum value.

Whilst the regular solution of figure 5.7 took more than 16 hours of CPU time (59994 sec.) to compute on a Silicon Graphics R5000, 150 MHz, 32Mb work station, the AMR solution of figure 5.9 took less than an hour (2622 sec.) on the same machine. Thus, the AMR computation was 22.9 times faster than the regular one. Timings were also made for a coarser grid, equivalent to 630 by 630 cells (AMR grid: 70 by 70 base with refinement factors $\Gamma_2 = \Gamma_3 = 3$). The regular and AMR grids took 0814 and 728 seconds of CPU time respectively, i.e. a speed-up factor of 13.5. An attempt was made to compute a 1750 by 1750 regular grid solution, which should have taken approximately 69 hours to compute. However, within a few minutes it became obvious that it would take a much greater time than expected, because the

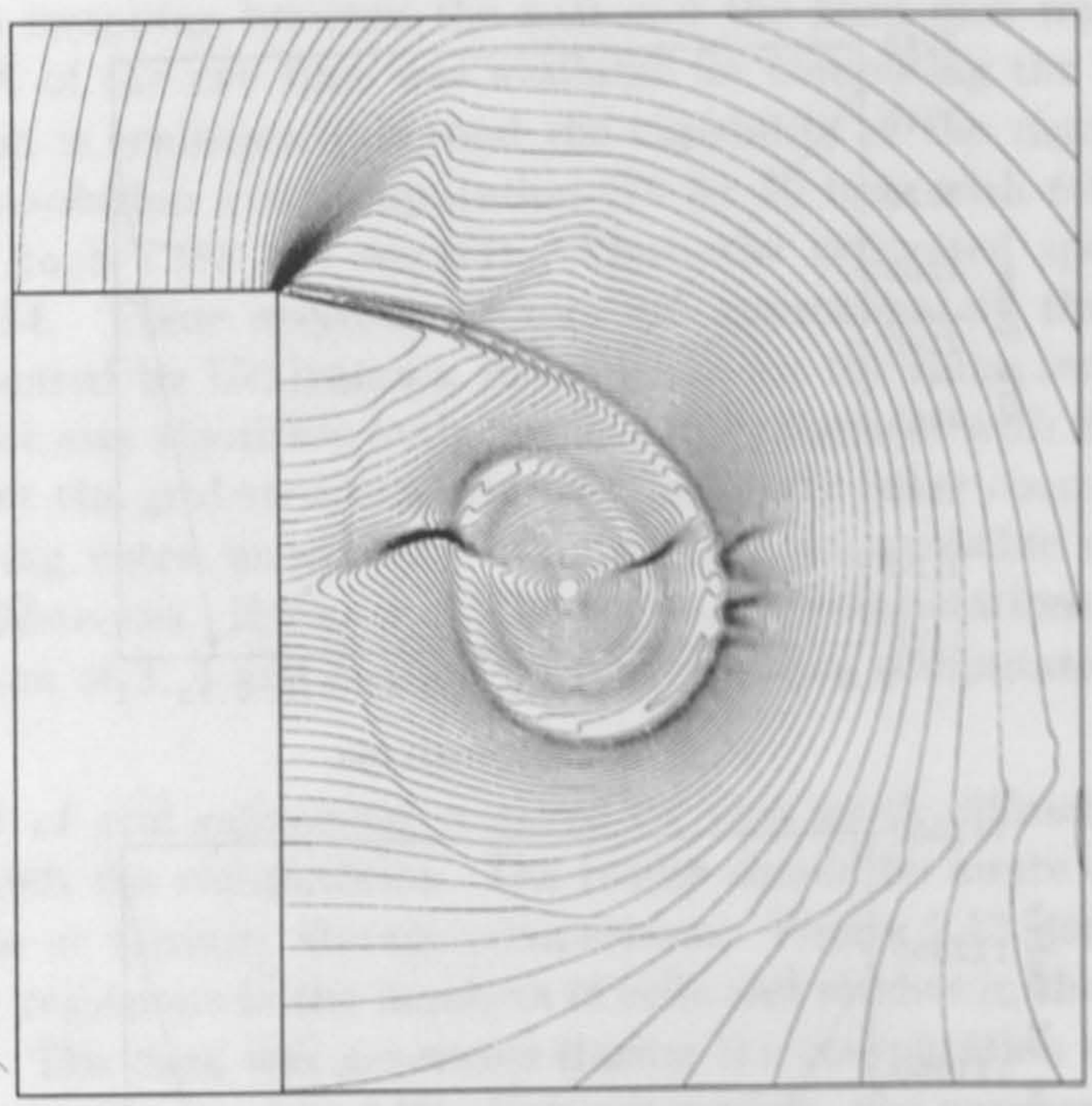


Figure 5.10: A magnified view of the regular grid vortex region (75 density contours).

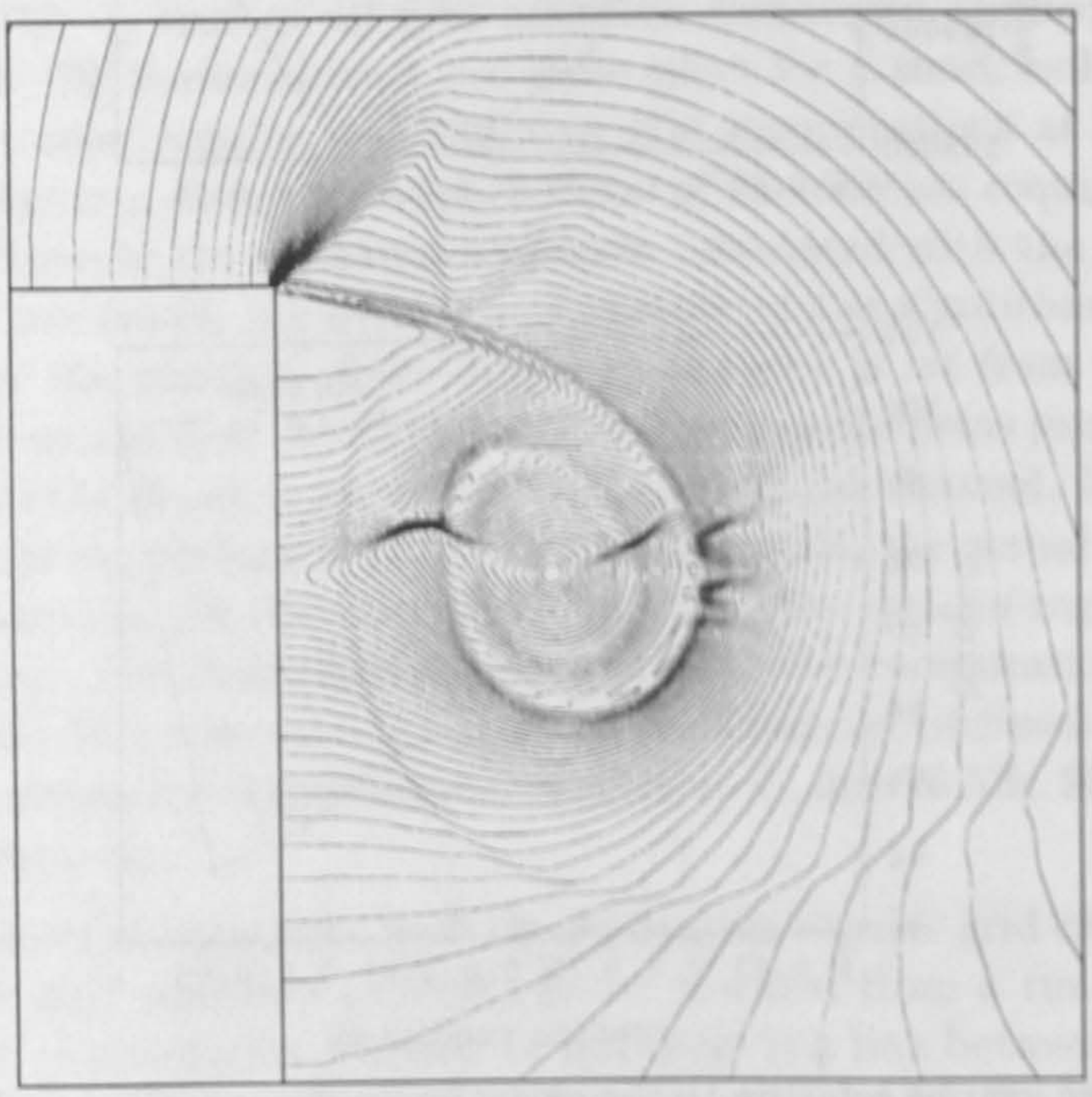


Figure 5.11: A magnified view of the AMR grid vortex region (75 density contours).

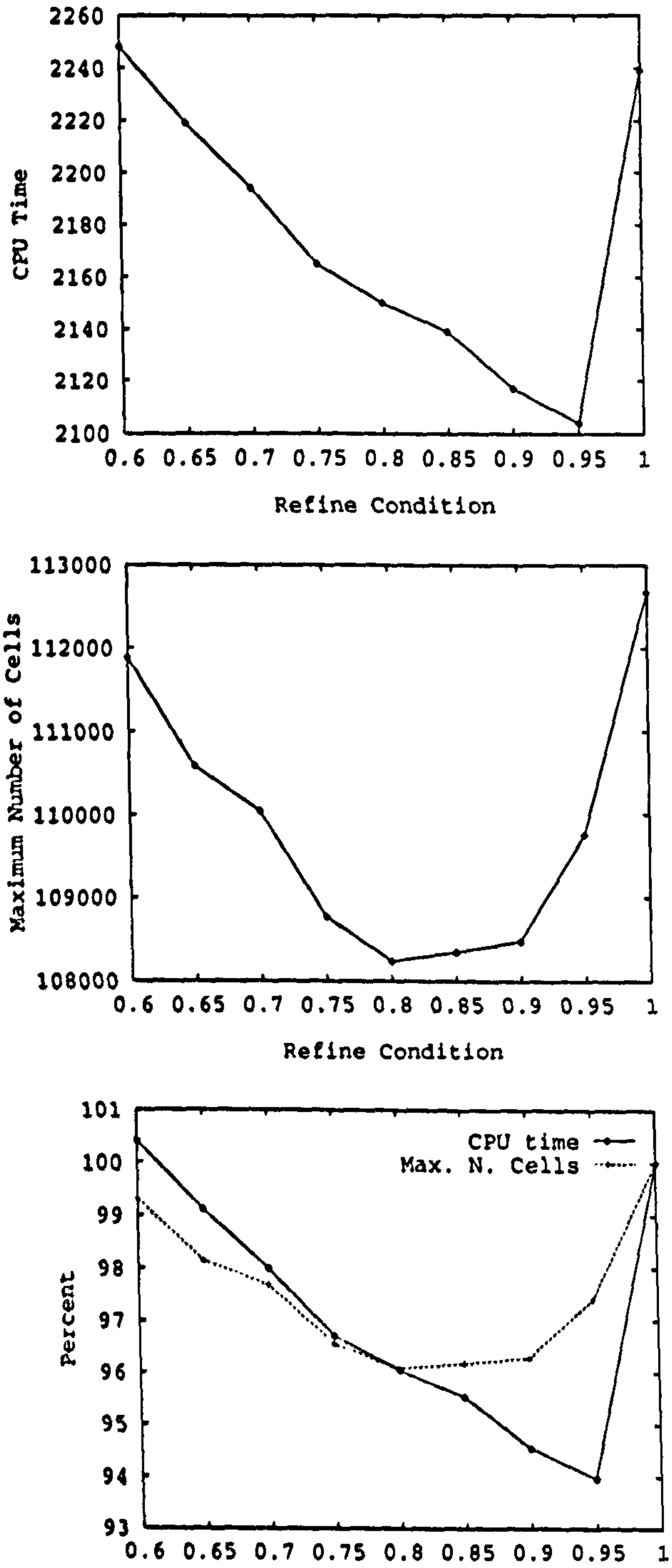
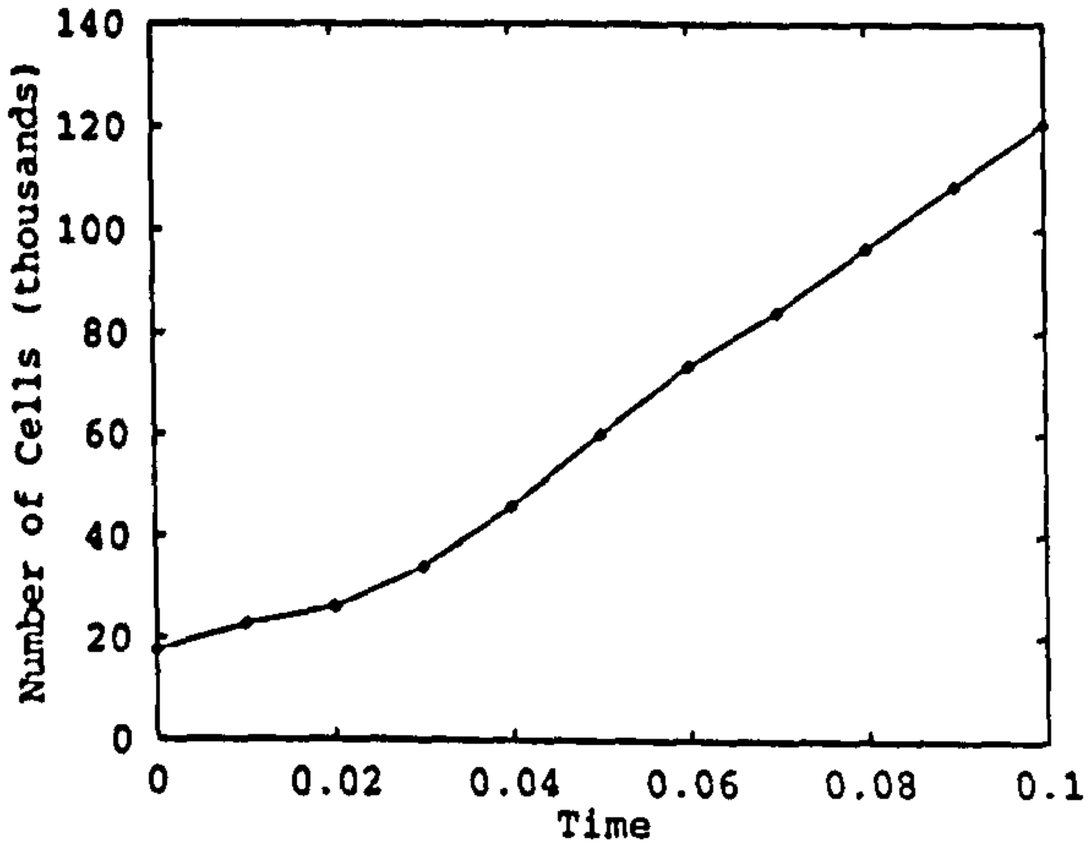


Figure 5.12: Three graphs relating to the variation in the CPU time and the maximum number of cells with the refine condition.

amount of data swapping between the RAM and the hard disk was excessive. As a result only 5% of the CPU time was available for computing the solution. Thus, this computation is realistically beyond the capability of the machine. However, the equivalent resolution AMR computation (70 by 70 base with refinement factors $\Gamma_2 = \Gamma_3 = 5$) took 7281 CPU seconds. Thus, the estimated speed-up factor is approximately 34. These speed-up factors are approximately three times larger than those presented by Uchiyama & Inoue [121], for the same test problem. The simplicity of their AMR algorithm (only one level of refinement with fixed mesh sizes), and the fact that the grid structure was updated every *other* coarse grid time step (thereby incurring extra unnecessary integrations) are probable explanations for the speed-up differences. Hereinafter, the three AMR computations with finer level refinement factors of 3, 4 and 5, shall be referred to as computations A, B and C respectively.

The amount of grid refinement required for this shock diffraction problem increases throughout the computation. The reader should be aware of the effect this has on any time or memory storage comparisons. Figure 5.13 depicts two graphs which show the variations in the numbers of cells and meshes in the AMR grid structure with time. The data was generated during the computation described above, with the refine condition set to 0.90. Not surprisingly, the numbers of meshes and cells increase with the size of the self similar features. Therefore, it is to be expected that the amount of memory storage and processing costs increase during the computation. This is true of all grid adaption techniques applied to self similar problems. Quirk [79] demonstrated the same effect for a shock reflection problem. Comparisons between regular and adaptive grid computations at early times are misleadingly flattering, because the proportion of the domain requiring adaption is small. Indeed, because the adaptive grid costs associated with the computation of the plane shock are small, the overall performance of the algorithm appears to improve the farther the position of the initial plane shock is set from the corner. The reader should be aware that in all the shock diffraction problems presented here, the initial position of the shock is set halfway along the inlet channel. In order to get a clear indication of the performance of any adaptive code, the actual speed-up factor should be compared to the theoretical one, (equal to the ratio of regular to adaptive grid integrations). The theoretical speed-up factors for computations A, B and C, are 17.6, 24.6 and 35.1 respectively. Hence, the efficiencies (defined in section 5.3.1) of the AMR algorithm for computations A, B and C, are 76.5%, 93.0% and 96.8% (estimate) respectively.

As well as direct comparisons with the equivalent regular grid computations, the efficiency of the grid adaption process can be assessed from a run time procedure profile. In order to investigate whether or not there is a link between the percentage of CPU time spent adapting the grid and the proportion of the domain requiring refinement, the AMR code was profiled for the three different grid resolution compu-



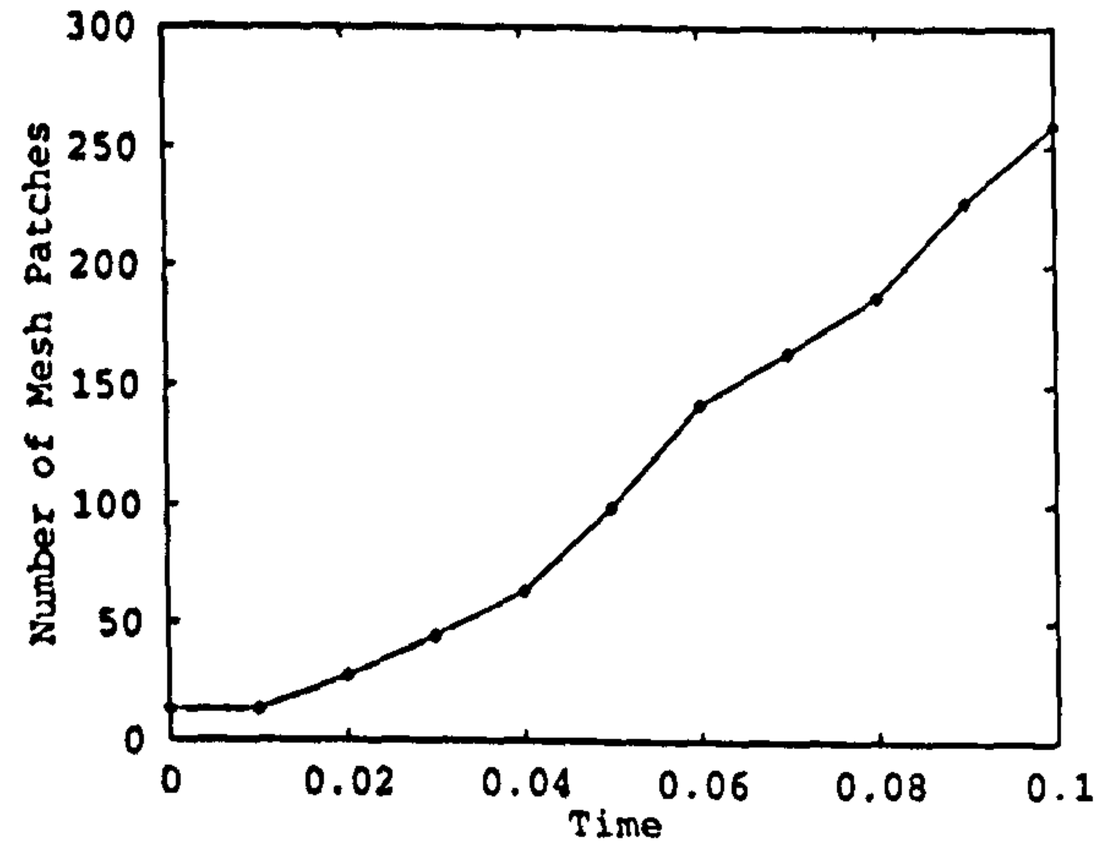


Figure 5.13: Graphs depicting the variations in the numbers of cells and meshes in the grid structure with time.

tations (A, B and C). Table 5.1 shows, for each test, the relative procedure costs as a percentage of the total CPU time. Note that, a percentage of 0.00 implies a value less that 0.005 and not zero. The total percentage incurred by the grid adaption is equal to the sum of percentages for the **new_AMR_struct**, **set_AMR_ghost_cells** and **update_coarse_solution** procedures. Thus, AMR incurs 3.63%, 2.27% and 1.55% of the total CPU time, for computations A, B and C respectively. Hence, this is a clear indication that the costs associated with AMR are inversely proportional to the level of refinement. It is important to emphasise that the AMR algorithm presented here adapts the grid whenever possible and utilises a slightly more complicated mesh generator in order to minimise the amount of integration. Moreover, the algorithm is most efficient when the refine condition is high (approximately 0.9), which increases the mesh generation CPU time. Thus, no attempt has been made to reduce the AMR overheads; to do so would increase the overall computation times.

Procedure	% of total cpu time		
	Test A	Test B	Test C
initial_setup	0.00	0.00	0.00
new_AMR_struct	2.55	1.54	1.02
time_step	1.02	1.06	1.07
set_AMR_ghost_cells	0.80	0.54	0.39
integrate_AMR_grid	94.53	96.25	97.12
update_coarse_solution	0.28	0.19	0.14
output_data	0.82	0.42	0.26

Table 5.1: Profile information for three different shock diffraction computations.

Note that, a meaningful profiling is only possible if the code is compiled without full optimisation. Also, the gprof facility does have its own overheads. Thus, it is not possible to be absolutely sure that the results presented in table 5.1 are completely accurate. However, in the absence of better alternatives, the presented results are cautiously accepted.

5.4 2D Chimera-AMR Test Results

The tests in this section are intended to give an indication of the ability of the CAMR algorithm to compute non-Cartesian domains with and without moving boundaries. The first test is a one-dimensional problem with an 'exact' solution, known as Lagrange's problem. It involves the movement of a piston under the action of a pressurised gas with constant covolume. The second test is a realistic problem in gas dynamics, in which a plane shock wave reflects from an inclined wedge. The computed CAMR solution compares very well with experimental Schlieren pictures and other regular grid solutions. Processing time comparisons with regular grid solutions and code profilings give a clear indication of the performance of the CAMR strategy.

5.4.1 Lagrange's Test Problem

Lagrange's test problem is a one-dimensional problem with an analytical solution [67]. It involves a closed piston chamber filled with a pressurised gas. The piston moves under the action of the adjacent gas pressure. Any errors in the computed solution will tend to compound with increasing time. It therefore provides a very good test for the Chimera approach. For this problem a boundary-fitted grid is attached to the rear of the piston and is allowed to move over the underlying fixed

grid which covers the full tube length.

The test is computed using the Euler equations with the constant covolume equation of state. The covolume is $b = 0.001 m^3/kg$ and the ratio of specific heats is $\gamma = 11/9$. The initial conditions within the tube are spatially constant. The gas is stationary with density $\rho = 400 kg/m^3$ and pressure $p = 621.06 MPa$. The piston has a base area of $0.01767 m^2$ and a mass of $50 kg$. It is initially at rest at a distance of $1.698 m$ from the left hand end (breech) of the tube. The pressure ahead of the piston is atmospheric $p_{atm} = 1.01325 \times 10^5 Pa$. Thus, the force on the piston is given by the pressure difference across it, multiplied by the base area. The tube, (length equal to $8m$), was discretised by a Cartesian base grid with 50 cells. A single level of refinement (factor 2) was employed throughout the region behind the piston. This increased the equivalent resolution to 100 cells covering a domain of $8m$. The boundary-fitted piston grid had only 4 cells and were equal in size to those on the finest Cartesian level.

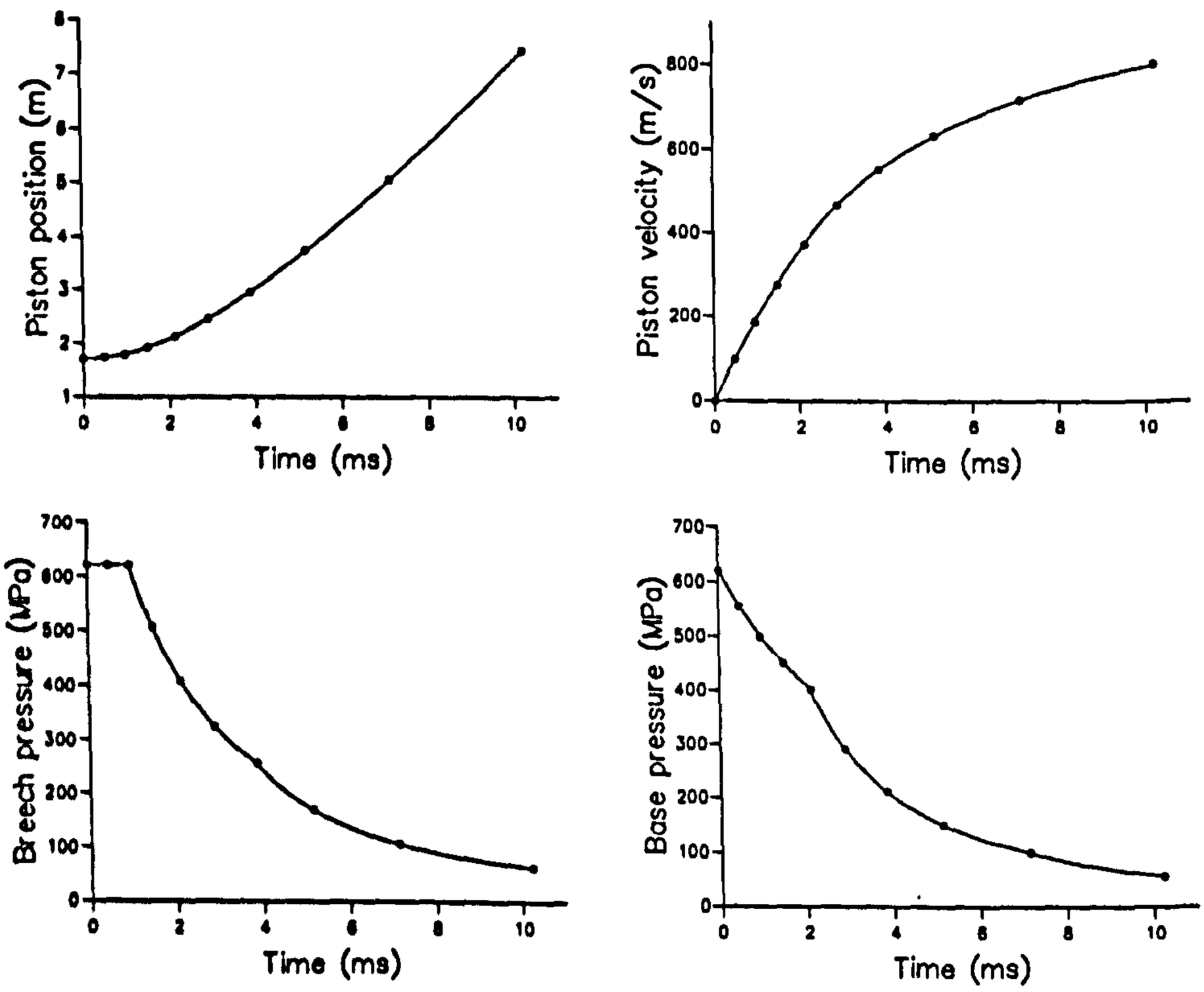


Figure 5.14: Analytical (symbols) and numerical (full lines) solution time histories for Lagrange's test problem.

The four graphs in figure 5.14 represent the time evolutions, up until time $t =$

10.23ms, of the piston position, piston velocity, breech pressure and piston base pressure. The solid lines represent the numerical results and the symbols the exact results which are taken from [67] via [16]. In all four graphs there is clearly good agreement between the exact and the numerical solutions.

5.4.2 Mach 1.7 Shock Reflection from a 25 Degree Inclined Wedge

The interaction of a plane shock wave with an inclined wedge has been extensively studied, both experimentally and computationally. The papers by Glass [39] and Glaz *et al.* [40] and the book by Ben-Dor [5] are good sources of information and reference. When a plane shock wave impinges on a fixed angle solid wedge the reflected wave structure is generally classified as being of one of four types. The first type is referred to as a regular reflection (RR) and consists of the incident shock and the curved reflected shock, both of which meet at the point of reflection on the wedge surface. The other three types are referred to as Mach reflections; single- (SMR), complex- (CMR) and double-Mach reflections (DMR). With Mach reflections the incident and the curved reflected shocks meet with another shock and a slip surface away from the wedge surface. The intersection is referred to as the triple point (three shocks). The third shock, referred to as the Mach stem, is usually[4] straight and normal to the wedge surface. All four types of two-dimensional wedge reflection are self similar, i.e. they can be described by the independent variables x/t and y/t, where t represents the time and x, y the spatial coordinates. The transition from one type of reflection to another has been extensively studied and there exists several different criteria for its prediction [72].

All four types of Mach reflection have been successfully modelled with the Euler equations. A common numerical technique is to discretise the domain with a fixed regular curvilinear grid, such as the one illustrated in figure 5.16. The merits of the CAMR algorithm can be assessed by comparing the adaptive grid solutions and processing costs with those of an equivalent resolution regular grid. Three different resolution tests (D, E and F) were computed with both the CAMR and fixed regular grids. All the solutions were obtained using the space operator split WAF scheme, in conjunction with the HLLC Riemann solver and VAN LEER flux limiter. The computations were performed on a Silicon Graphics R5000, 150 MHz, 32Mb work station. An AMR grid structure with a 50×33 cell base grid and two levels of refinement was used for all three adaptive computations. The refinement factors of both refined levels, were set to 3, 4 and 5 for tests D, E and F respectively. Again, as with test C of section 5.3.2, the regular grid computation of test F was realistically

[4]The slip surface in a DMR rolls up into a vortex which can, in some situations, interact with and slightly distort the Mach stem.

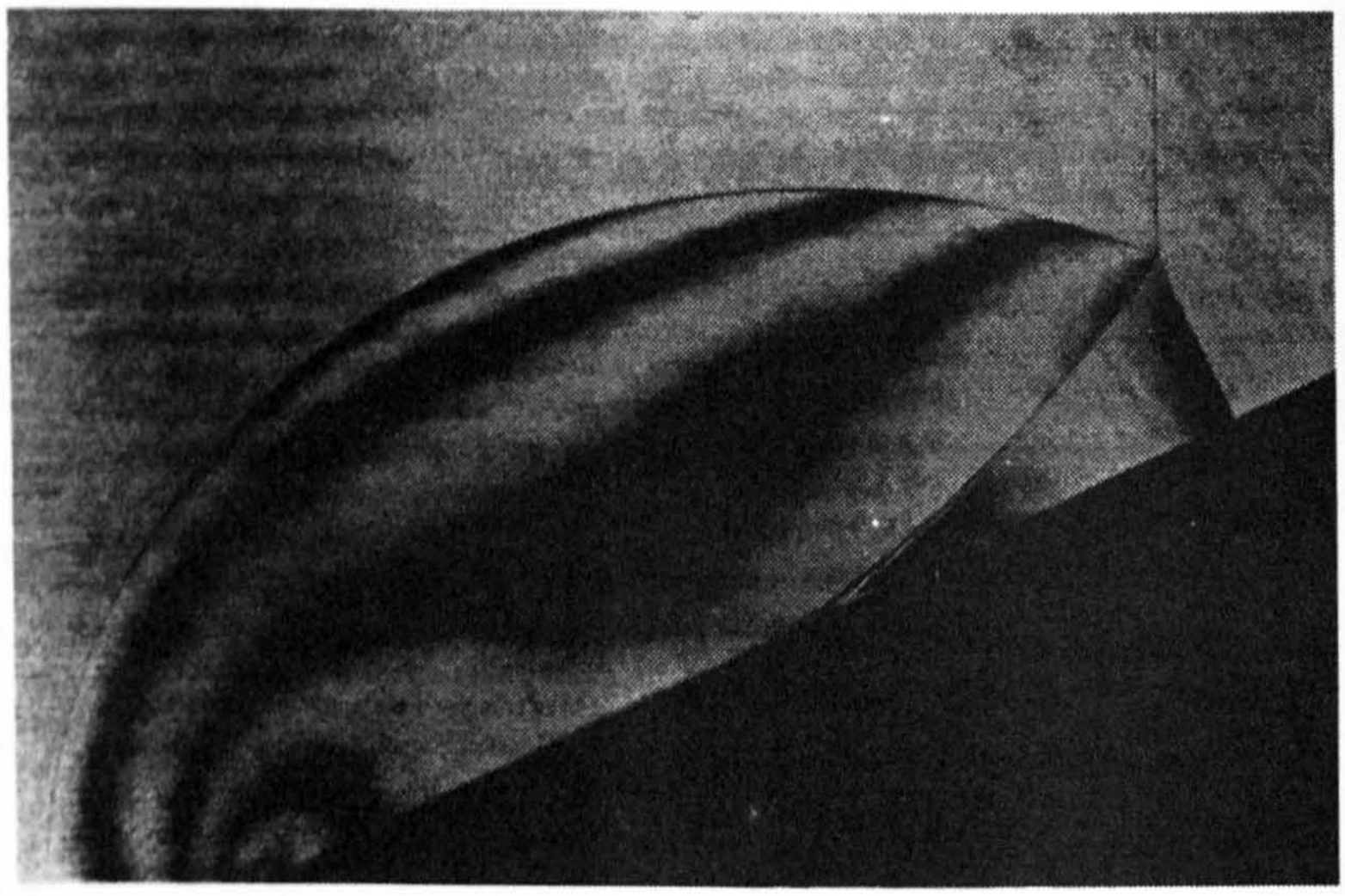


Figure 5.15: Experimental interferogram of a Mach 1.7 shock reflection from a 25 degree wedge. Courtesy of Prof. K. Takayama, Japan.

beyond the capabilities of the machine. The regular grid density solution for test E is shown in figure 5.17. The AMR grid structure and density solution for the same test are shown in figures 5.18 and 5.19 respectively. There is little difference between figures 5.17 and 5.19, both of which compare very well with the experiment. As was the case for the shock diffraction problem in section 5.3.2, the start-up error[5] is only noticeable in the regular grid computation.

For the three tests, direct CPU time comparisons have been made and the actual speed-up factors of the adaptive over the fixed regular grid computations calculated. Theoretical speed-up factors, based on the total number of cell integrations for both grid types, were used to determine the efficiency of the adaptive algorithm for the three tests. The various results are presented in table 5.2. Note that, other than the CPU time for the adaptive grid, the values for test F are estimates based on an extrapolated CPU time for the regular grid computation. At a first glance, the efficiency of the CAMR approach appears to be significantly less than that of the Cartesian AMR approach. This is a simplistic view, since a far greater proportion of the flow domain requires adaption for the wedge problem. Compared with the theoretical speed-up factors, the efficiency of the CAMR approach appears to be greater. Of course the CAMR approach incurs extra costs, but they are small compared to those of equivalent resolution curvilinear grid computations.

A clearer understanding of the costs associated with the various parts of the CAMR approach can be gleaned from run time profile information. Table 5.3 contains

[5]The data behind the incident shock was reset before it reached the wedge.

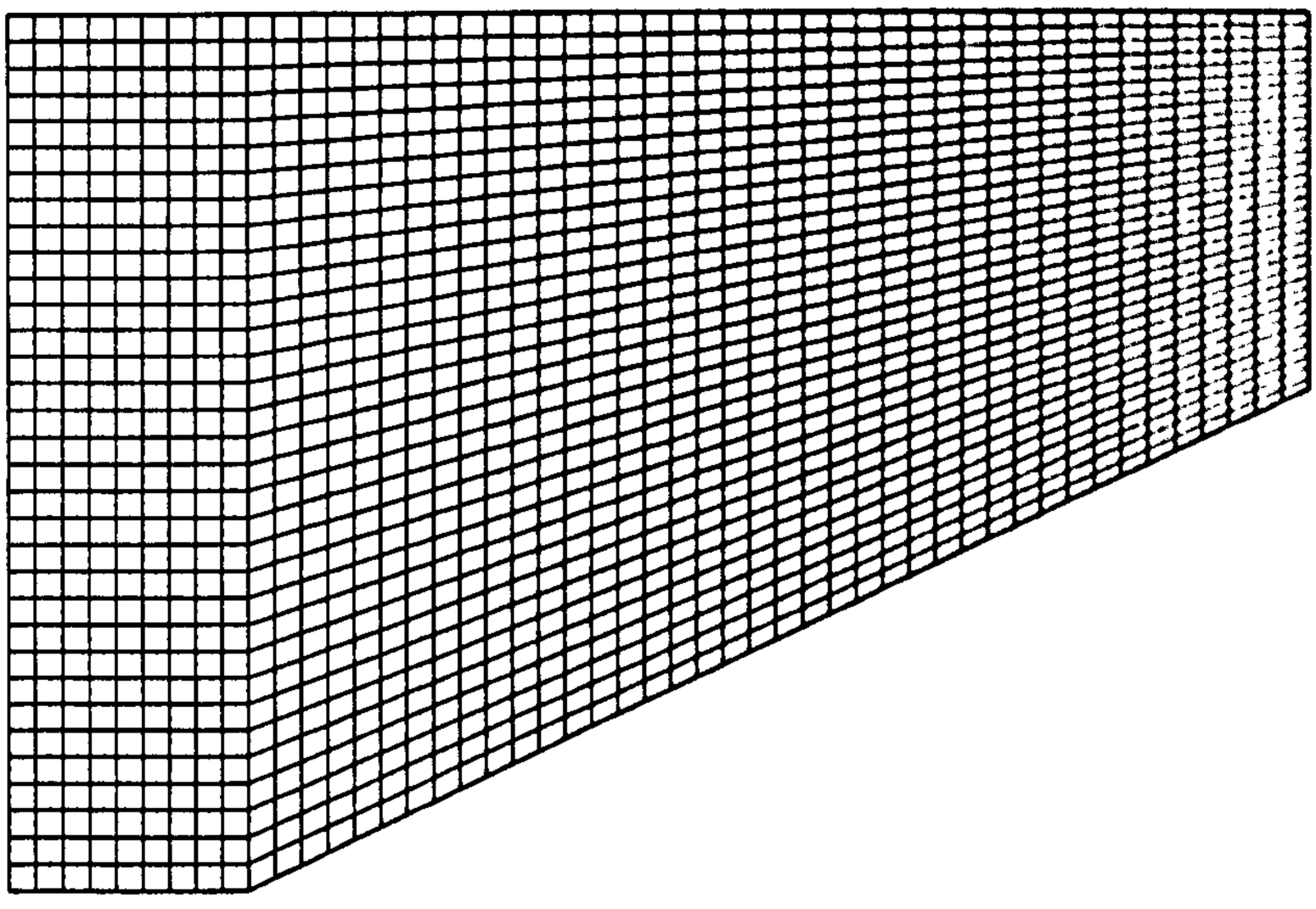


Figure 5.16: An illustration of a fixed regular grid for a wedge domain.

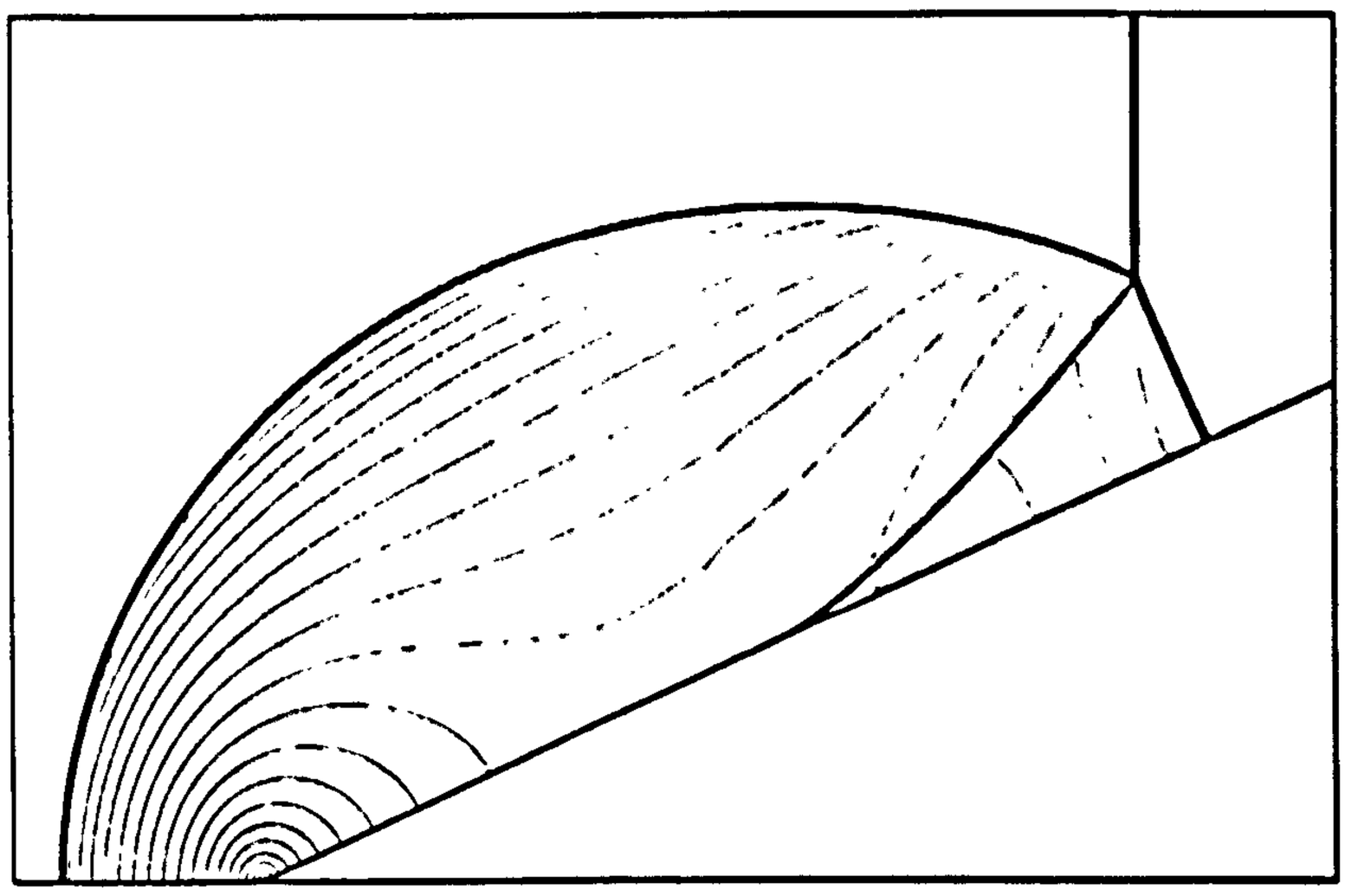


Figure 5.17: Density contours (75) computed with a 800 × 528 fixed regular grid.

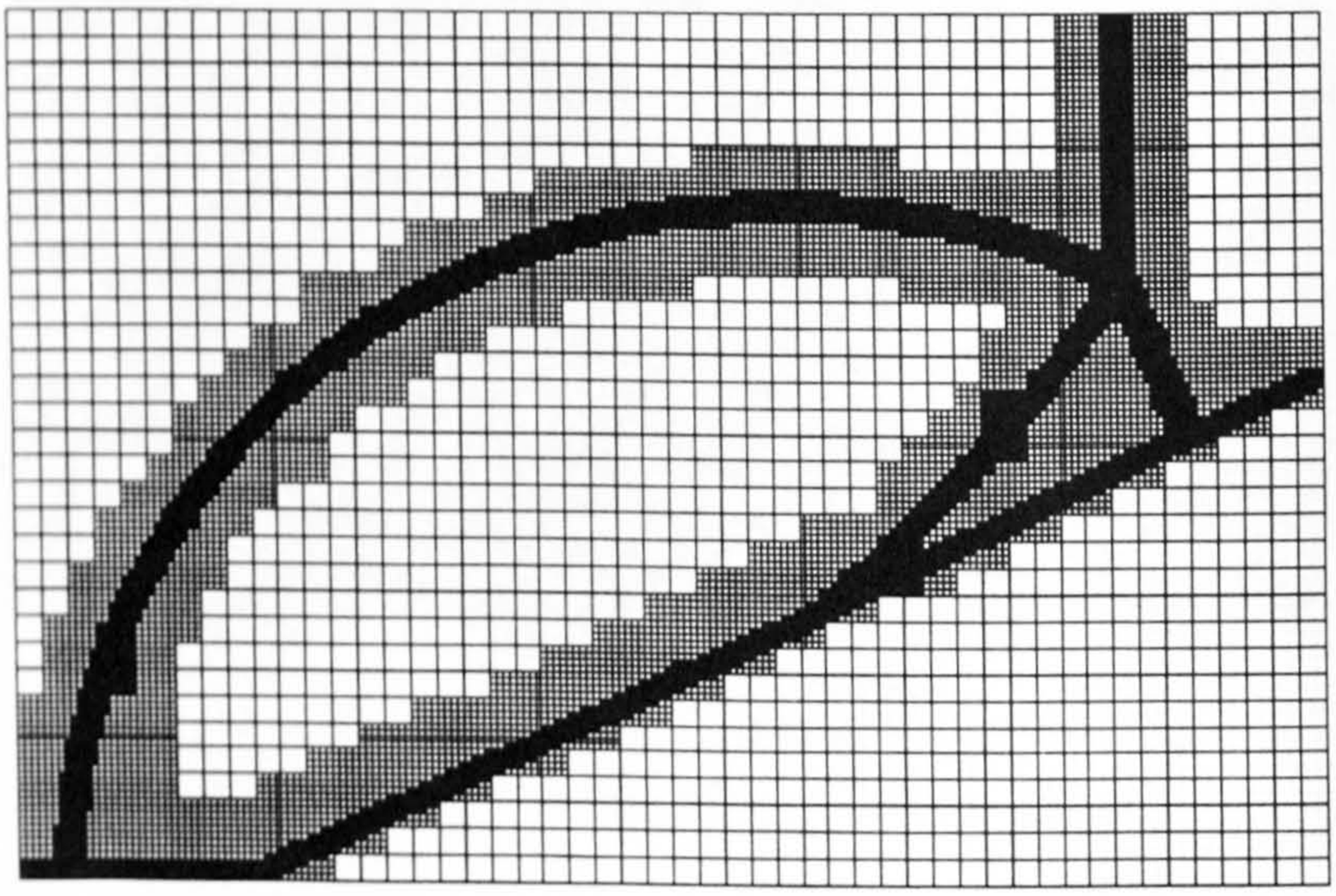


Figure 5.18: The CAMR grid structure for test E.

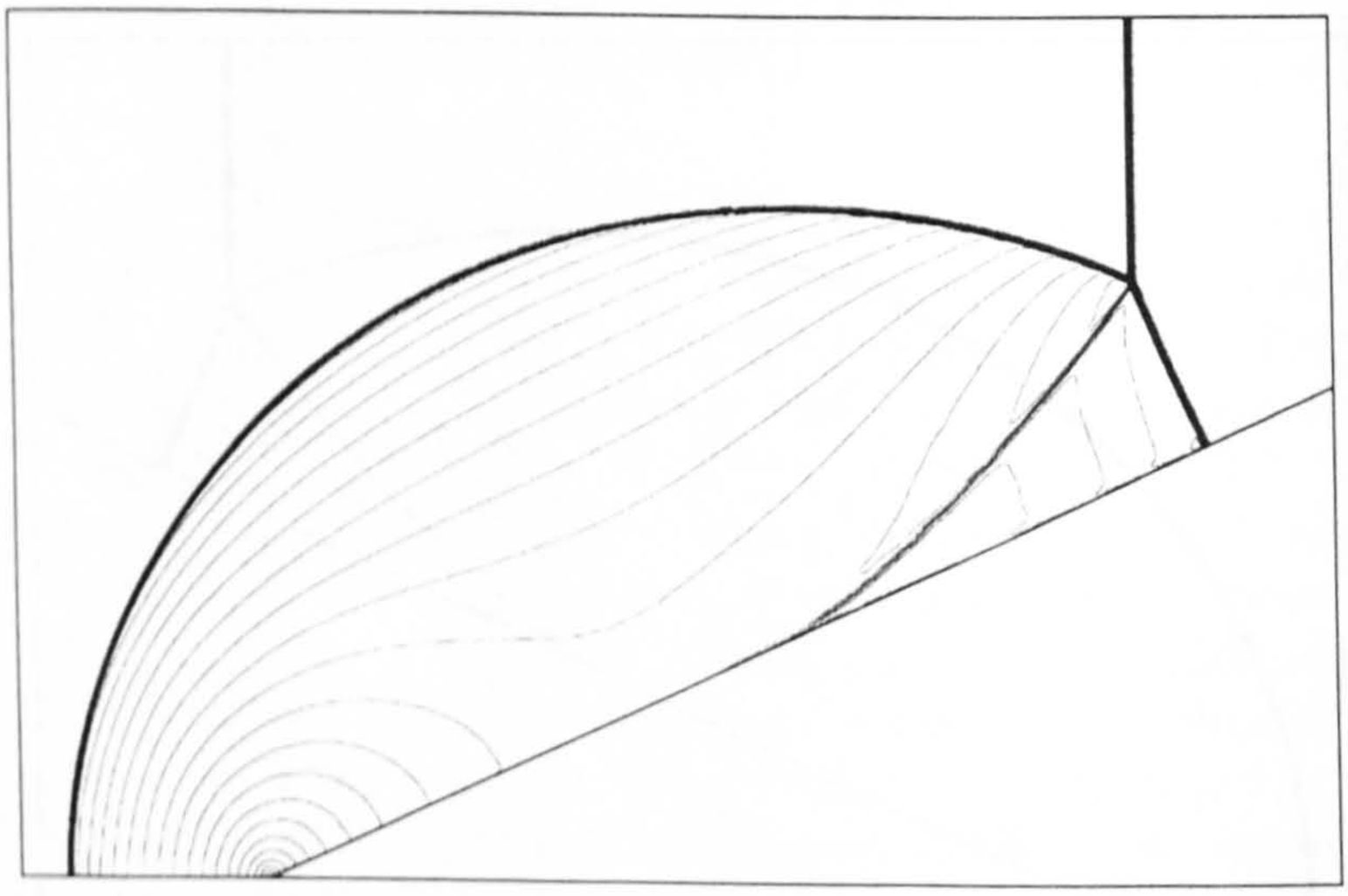


Figure 5.19: Density contours (75) computed with the CAMR grid for test E.

Test	D	E	F
Regular grid CPU time sec.	3349	22402	101739
Adaptive grid CPU time sec.	764	2379	6402
Actual Speed-up factor	4.38	9.42	15.89
Theoretical Speed-up factor	6.88	13.05	19.32
Efficiency	63.4%	72.1%	82.2%

Table 5.2: Cost comparisons for the three different resolution shock reflection tests.

Procedure	% of total CPU time		
	Test D	Test E	Test F
initial_setup	0.01	0.01	0.00
new_AMR_struct	2.39	1.58	1.10
set_AMR_flags	1.17	1.15	1.06
time_step	1.00	1.15	1.25
set_AMR_ghost_cells	0.84	0.70	0.57
set_BF_ghost_cells	0.40	0.36	0.32
integrate_AMR_grid	76.56	78.88	81.69
integrate_BF_grid	16.95	15.77	13.74
update_coarse_solution	0.14	0.10	0.08
output_data	0.54	0.30	0.19

Table 5.3: Profile information for the three different Mach reflection computations.

the profiling information for three different resolution tests (D, E and F). Again, because of the profiling overheads and the lack of full compiler optimisation, the accuracy of the results presented in table 5.3 cannot be guaranteed. The total percentage incurred by the grid adaption is again equal to the sum of percentages for the **new_AMR_struct**, **set_AMR_ghost_cells** and **update_coarse_solution** procedures. There are also some extra costs associated with the CAMR approach that would not appear in a regular Chimera approach. They are small and are incurred by the *slightly* more expensive routines that interpolate the solution between the two grid types. Neglecting these extra costs, the percentage of the CPU time incurred by the AMR algorithm for tests D, E and F are 3.37%, 2.38% and 1.75% respectively. Thus, the grid adaption costs decrease with increasing grid resolution. Conversely, a greater proportion of the computation time is spent integrating the solution. With increasing resolution, the percentage of the CPU time devoted to integrating the AMR and boundary-fitted grids increases and decreases respectively. This is because

the ratio of the number of boundary-fitted to AMR cells decreases with increasing resolution.

Chapter 6

An Internal Ballistics Application

6.1 Introduction

The study of internal ballistics almost certainly begins with the use of gunpowder as a ballistic propellant, which dates back at least to the beginning of the 14th century. Mathematical descriptions of internal ballistics phenomena have been developed over several hundred years. Today's mathematical models incorporate burning laws, chemistry, multiple phase propellants, complex ignition processes, etc. The numerical methods used to solve the mathematical models have advanced rapidly since the advent of the modern computer. However, modelling multi-phase flows is still one of the most difficult tasks in CFD today. Tractable solutions can only be obtained for models that neglect or approximate some of the details of the physics. Various numerical strategies have been tried and tested. For an up-to-date summary of interior ballistic models refer to [68, 99].

The internal ballistics problem of interest here, involves a similar configuration to that shown in figure 6.1. The chamber is packed with solid propellant granules, which combust when the ignition temperature is exceeded. Ignition is initiated by venting (radially) hot gases through the primer into the combustion chamber. As the propellant combusts gas is released and the temperature raised, which increases the gas pressure in the chamber. Peak gas pressures of $3-400\,MPa$ are typical. The shot remains stationary in the barrel until sufficient pressure is generated behind it to cause it to move. Frictional resistance is intentionally imposed on the shot to cause it to rotate as it travels along the barrel; after exiting the barrel, a spinning shot is more stable and predictable in its course. The projectile can reach speeds of more than $700\,m/s$ within $10-15\,ms$ of ignition.

Much of the research behind the internal ballistics model used here, has been sponsored by DERA, Fort Halstead and carried out by Toro [111], Speares & Toro

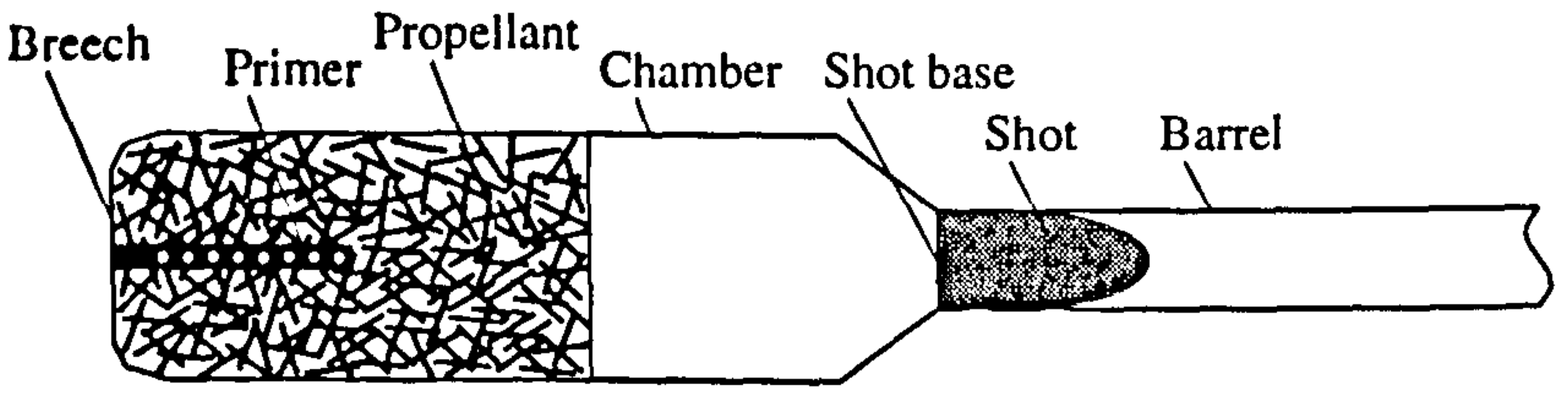


Figure 6.1: The basic configuration for an internal ballistics problem.

[96], Toro *et al.* [117] and Lowe [68]. The aim of the current research is to improve the existing numerical model, so that a greater understanding of the processes can be gleaned, in order to achieve an optimised configuration, e.g. maximum exit velocity for a given amount of propellant, whilst avoiding high damaging pressures.

The remainder of this chapter is organised as follows. Section 6.2 describes the system of PDE's that form the current mathematical model and the corresponding computational strategy. There are some mathematical and numerical difficulties with the model. Section 6.3 gives more details of the current model and describes three other ways in which the terms in the model can be rearranged or reformulated. The new formulations were designed in an attempt to resolve some of the numerical difficulties associated with the current model.

6.2 Mathematical Model

The problems of interest here, involve a macroscopic mixture of two distinct phases (gas and solid). The solid propellant granules are large relative to the size of the molecules, i.e. the mixture is not on a molecular level. Whilst interphase drag does exist between the two phases, it does not dominate the movement of the solid particles. Also, the density of the solid is very much greater than the density of the gas. Therefore, the momentum of each phase is generally very different. Thus, an accurate model must allow each phase to be transported with different velocities. If the assumption is made that the solid is incompressible, then it is also reasonable to assume that the pressure in the solid phase is equal to the pressure of the surrounding gas. The mathematical model proposed by Gough [45] and later modified by Fitt [34, 35], incorporates these traits and was reformulated by Toro [111]. The model is similar to the equal pressure model described by Stewart & Wendroff [99], but because the solid phase is incompressible, the solid phase energy equation is replaced by a particle number density equation. The axi-symmetric, two-dimensional, two-phase, internal ballistics model presented here, is the same as that presented by Spcares & Toro [95].

The mathematical model can be derived by assuming both of the phases can

be represented by a continuum. The model consists of eight PDE's, which express the balances of mass and momentum for each phase, gas energy and particle number density in terms of average values of the relevant variables (density, pressure, temperature, etc.) over control volumes. For such a mathematical description to be valid, the size of the control volumes must be large relative to the length scales of the mixture, which in this case are given by the dimensions of the propellant granules, i.e. the complicated interface between the phases is absorbed by the averaging. However, the smaller the size of the control volumes (grid cells) relative to the flow feature length scales, the more accurately the continuum model represents the physics. Stewart & Wendroff [99] noted that solving the mathematical model, that was derived with the assumption that the control volumes are large, on a refined computational grid, is effective and the solutions compare well with experimental results. The main disadvantage, is that the averaging process suppresses knowledge of the interface between the phases, which is needed for accurate modelling of the interphase drag and heat transfer.

6.2.1 Computational Strategy

The mathematical model, presented here, along with most other internal ballistics models, is of mixed hyperbolic-elliptic type and is therefore ill-posed, in that two of the eigenvalues are complex [117]. In an attempt to overcome the difficulties posed by the ill-posed model, Toro [111] advocated dividing the computational strategy into three parts via time operator splitting. The same computational strategy is adopted here. The three parts consist of a separate hyperbolic system of homogeneous PDE's for each phase and a single system of ODE's associated with the remaining source terms, i.e.

$$\mathbf{U}_t^{(g)} + \mathbf{F}(\mathbf{U})_x^{(g)} + \mathbf{G}(\mathbf{U})_r^{(g)} = 0 \tag{6.1}$$

$$\mathbf{U}_t^{(s)} + \mathbf{F}(\mathbf{U})_x^{(s)} + \mathbf{G}(\mathbf{U})_r^{(s)} = 0 \tag{6.2}$$

$$\mathbf{U}_t^{(g+s)} = \mathbf{S}(\mathbf{U})^{(g+s)} \tag{6.3}$$

where t is the time, x is the axial distance from the breech and r is the radial distance from the centre of the chamber/barrel. $\mathbf{U}$ is the vector of conserved variables, $\mathbf{F}(\mathbf{U})$ is the axial flux vector, $\mathbf{G}(\mathbf{U})$ is the radial flux vector and $\mathbf{S}(\mathbf{U})$ is the source term vector. The superscripts g and s denote the gas and solid phase variables respectively. The solution is explicitly updated to the next time level by first solving the system of gas phase PDE's (6.1). Then the solid phase system of PDE's (6.2), in which the pressure is taken to be equal to that of the gas phase, is solved. Lastly, the system of ODE's (6.3) is solved using the same time step as the two systems of PDE's.

By splitting the mathematical model in this way, the assumption is made that

the two phases do not interact with each other. Also, in order to group the terms so as to form two hyperbolic systems, it is assumed that the solution and its derivative are continuous for all space and time. Therefore, because the phases *do* interact with each other and the flow is *not* smooth, the computational strategy introduces some inaccuracies, besides the numerical errors associated with the PDE and ODE solvers. However, the two phases do communicate with each other in two ways; the volume fractions add up to one and the pressure is equal in each phase. The advantages of this strategy, are that the solutions are tractable and that the highly transient and discontinuous nature of each phase, can be accurately modelled using the WAF scheme. Furthermore, the modular computational strategy facilitates the inclusion of other phases and propellants (i.e. another hyperbolic system for every phase or propellant). This strategy has been shown to be reasonably effective for the problems considered here [68, 95, 111].

6.2.2 The Internal Ballistics Equations

Neglecting interphase drag, interphase heat exchange, heat loss to the boundaries and intergranular stress, the two-dimensional axi-symmetrical two-phase ballistic equations [95] can be written as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_r = \mathbf{S}(\mathbf{U}) \tag{6.4}$$

$$\mathbf{U} = \begin{bmatrix} \alpha_1\rho_1 \\ \alpha_1\rho_1 u_1 \\ \alpha_1\rho_1 v_1 \\ \alpha_1 E_1 \\ \alpha_2\rho_2 \\ \alpha_2\rho_2 u_2 \\ \alpha_2\rho_2 v_2 \\ N \end{bmatrix} \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \alpha_1\rho_1 u_1 \\ \alpha_1\rho_1 u_1^2 + \alpha_1 p \\ \alpha_1\rho_1 u_1 v_1 \\ \alpha_1 u_1(E_1 + p) \\ \alpha_2\rho_2 u_2 \\ \alpha_2\rho_2 u_2^2 + \alpha_2 p \\ \alpha_2\rho_2 u_2 v_2 \\ u_2 N \end{bmatrix} \quad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} \alpha_1\rho_1 v_1 \\ \alpha_1\rho_1 u_1 v_1 \\ \alpha_1\rho_1 v_1^2 + \alpha_1 p \\ \alpha_1 v_1(E_1 + p) \\ \alpha_2\rho_2 v_2 \\ \alpha_2\rho_2 u_2 v_2 \\ \alpha_2\rho_2 v_2^2 + \alpha_2 p \\ v_2 N \end{bmatrix} \tag{6.5}$$

$$\mathbf{S}(\mathbf{U}) = \begin{bmatrix} \dot{M}_{pr} + \dot{M}_{ig} - \frac{\alpha_1\rho_1 v_1}{r} \\ u_2\dot{M}_{pr} + p\frac{\partial\alpha_1}{\partial x} - \frac{\alpha_1\rho_1 u_1 v_1}{r} \\ v_2\dot{M}_{pr} + p\frac{\partial\alpha_1}{\partial r} - \frac{\alpha_1\rho_1 v_1^2}{r} \\ E_2\dot{M}_{pr} + e_{ig}\dot{M}_{ig} - p\left[\frac{\partial}{\partial x}(\alpha_2 u_2) + \frac{\partial}{\partial r}(\alpha_2 v_2) + \frac{\alpha_2 v_2}{r}\right] - \frac{\alpha_1 v_1(E_1+p)}{r} \\ -\dot{M}_{pr} - \frac{\alpha_2\rho_2 v_2}{r} \\ -u_2\dot{M}_{pr} + p\frac{\partial\alpha_2}{\partial x} - \frac{\alpha_2\rho_2 u_2 v_2}{r} \\ -v_2\dot{M}_{pr} + p\frac{\partial\alpha_2}{\partial r} - \frac{\alpha_2\rho_2 v_2^2}{r} \\ -\frac{v_2 N}{r} \end{bmatrix} \tag{6.6}$$

where t is the time, x is the axial distance from the breech and r is the radial distance from the centre of the chamber/barrel. $\mathbf{U}$ is the vector of conserved variables, $\mathbf{F}(\mathbf{U})$

is the axial flux vector, **G(U)** is the radial flux vector and **S(U)** is the source term vector. Here the subscripts 1 and 2 denote the gas and solid phase variables respectively. The fraction of the control volume occupied by each phase is denoted by α. The phase density, axial component of velocity and radial component of velocity are denoted by ρ, u and v respectively. It is assumed that the solid phase density ρ_2 is constant. The gas pressure and the particle number density are denoted by p and N respectively. The total energy per unit volume E_1 of the gas is defined as

$$E_1 = \frac{1}{2}\rho_1 u_1^2 + \rho_1 e_1 \tag{6.7}$$

where the specific internal energy e_1 is given by

$$e_1 = \frac{p(1 - b\rho_1)}{(\gamma - 1)\rho_1} \tag{6.8}$$

γ is the ratio of specific heats and b is the covolume, which is assumed to be constant. In (6.6), e_{ig} is the chemical energy per unit mass of the igniter gas and $\dot{M}_{ig}$ and $\dot{M}_{pr}$ are the rates of mass addition per unit volume of the igniter gas and the gas produced by the combustion of the propellant respectively.

6.2.3 Source Terms

The source terms are those terms that remain after grouping the other terms such that they form two separate homogeneous hyperbolic systems, one for each phase. The source terms are grouped together and are used to update the solution after the two hyperbolic systems have been solved.

The terms involving r (not including the radial derivatives) arise from the axi-symmetrical flow derivation. The axial and radial derivatives in the momentum equations (second, third, sixth and seventh equations) are the results of reformulating the derived balance equations into hyperbolic form, i.e. the original term is

$$\alpha_1 \frac{\partial p}{\partial x} = \frac{\partial \alpha_1 p}{\partial x} - p\frac{\partial \alpha_1}{\partial x}$$

Note that this reformulation assumes that the flow is smooth. The axial and radial derivatives in the energy equation (fourth equation) is the work done due to the change in the gas volume fraction.

The injection of an energetic gas into the chamber affects the mass and energy equations. If the assumption is made that the rate of mass addition per unit volume $\dot{M}_{ig}$, is constant throughout the venting time t_{ig}, then

$$\dot{M}_{ig} = \frac{m_{ig}}{V_{ig} \times t_{ig}} \tag{6.9}$$

where m_{ig} is the total igniter mass and V_{ig} is the volume into which the igniter gas is injected.

As the solid propellant burns, gas is generated. Assuming that the products of combustion are purely gaseous, the rate of decrease in the mass of the solid is equal to the rate of increase in the mass of the gas. Once conditions are such that combustion takes place (temperature greater than ignition temperature T_{ig}), the burning rate is a function of the propellant mass, the shape of the granules, the pressure and some empirical coefficients. The shape of the propellant is fundamental to the rate of combustion. Under identical conditions, two granules of equal mass, but different surface areas will burn at different rates; the one with the larger surface area will burn faster. If as the propellant combusts, its surface area increases then the rate of combustion will increase and the burning is said to be progressive. Conversely, if the surface area decreases with combustion, then the rate of combustion will decrease and the burning is said to be degressive. Here we make the assumption that all the granules have the same shape and that *parallel layer burning* occurs. Consequently, the shape of the granules is assumed to remain constant throughout the combustion process, uniformly decreasing in size and mass. The assumption that the particles burn uniformly and do not fragment is necessary for the conservation of particle number density. Experimental evidence supports this assumption for most of the burning process. Therefore, the rate of mass addition (gas) per unit volume M_{pr}, at a point (x, r, t), is only a function of the propellant mass per unit volume $m(x, r, t)$ and the fraction left to burn $Z = Z\,(f(t))$, known as the *form function*, i.e.

$$\dot{M}_{pr} = m(x, r, t)\frac{dZ}{df}\frac{df}{dt} \tag{6.10}$$

where $f(t)$ is the fractional depth of the propellant granules that remains to be burnt.

The derivative of f with respect to t is the surface regression caused by the burning and is given by Piobert's law of burning;

$$\frac{df}{dt} = -\frac{\beta p^{\alpha}}{D_{frag}} \tag{6.11}$$

where β is the burning rate coefficient, p is the pressure, α is the pressure index and D_{frag} is the least thickness required for complete combustion, referred to as the ballistic size. For multi-tubular granules[1], D_{frag} is equal to web length between adjacent holes, e.g. for a seven tube propellant:

$$D_{frag} = \frac{d_o - 3d_i}{4} \tag{6.12}$$

[1] Cylindrical granules perforated, along their lengths, by a number of evenly spaced holes.

where d_o is the outer diameter of the granule and d_i is the diameter of the internal holes.

The form function is the fraction of the solid mass that is burnt to the original solid mass, i.e.

$$Z = \frac{\text{Original mass} - \text{Mass at time } t}{\text{Original mass}} \tag{6.13}$$

$$= 1 - \frac{\alpha_2 \rho_2}{N_p m_o} \tag{6.14}$$

where m_o is the initial mass of a single granule. Corner [27] suggested the following analytical approximation for Z and its derivative:

$$Z = (1 - f)(1 + \theta f) \tag{6.15}$$

$$\frac{dZ}{df} = \sqrt{(1 + \theta^2) - 4\theta Z} \tag{6.16}$$

where θ is a constant known as the *form coefficient* that depends on the geometry of the propellant granules. Progressive and degressive burning corresponds to $\theta < 0$ and $\theta > 0$ respectively. When ($\theta = 0$), the burning is said to be neutral. However, whilst (6.15) and (6.16) are effective approximations for the majority of the burning process, they are unable to predict accurately the later stages for multi-tube propellants, when fragmentation of the granules occurs. During fragmentation the surface area of the granules can change significantly, which has an appreciable effect on the burning rate. Pike [77] computed the exact form function for seven tube granules, which takes into account grain fragmentation. There is not a closed form expression for the derivative of the form function, so given a value for Z from (6.14), tabulated results are generated and the intermediate values calculated via interpolation. Compared to Corner's approximation, Pike's approach does improve the results, although the particle number density is no longer formally conserved. The results presented here have all been generated using Pike's approach.

There are many other physical effects that can be taken into account, but which have not yet been incorporated into the model presented here. Such physical effects include the viscosity, interphase drag and transfer of heat between the phases and across the chamber walls.

6.3 Formulations of the Particle Phase Equations

This section describes four different formulations of the particle phase equations. Actually, the formulations differ only in the way that the momentum equation is expressed. All the formulations assume that the solid phase pressure is equal to

the gas phase pressure and does not vary with time during the solid phase update. Note that the pressure solution does vary with space and time during the gas phase update. The first formulation, A, and its computational strategy [111], were outlined in the previous sections of this chapter. This formulation is not valid for situations in which the solid mass is zero, because it admits infinite wave speeds (obviously non-physical). The zero solid mass situation is analagous to the vacuum situation of gas dynamics. Hereinafter, situations where the solid mass is zero, shall be referred to as vacuums. Note that none of the work presented here involves vacuums in the gas dynamics context. The second formulation, B, was an attempt to circumvent the infinite wave speed problem, by altering the hyperbolic part of the equations. This resulted in an extra source term which tended to infinity with increasing grid resolution. The third formulation, C, involves a linearisation and is non-conservative. This formulation produces good results and is valid for vacuum situations. However, it assumes that the pressure does not vary with the solid phase density ($\frac{\partial \bar{p}}{\partial \rho} = 0$). The fourth formulation, D, is similar to formulation C, but does not make the assumption that the pressure is fixed relative to the solid phase density.

The pertinent issues for the four different formulations are described consecutively in the following four subsections.

6.3.1 Formulation A of the Particle Phase Equations

The homogeneous particle-phase equations of mass, momentum and particle number, for our one-dimensional two-phase model are:

$$(\alpha_2\rho_2)_t + (\alpha_2\rho_2 u_2)_x = 0 \tag{6.17}$$

$$(\alpha_2\rho_2 u_2)_t + (\alpha_2\rho_2 u_2^2)_x + \alpha_2 p_x = 0 \tag{6.18}$$

$$(N_2)_t + (N_2 u_2)_x = 0 \tag{6.19}$$

where α_2 is the fraction of the cross sectional area occupied by the solid phase, ρ_2 (constant) is the density of the solid particles, u_2 is the particle velocity of the solid phase, N_2 is the solid particle number density and p is the gas phase pressure.

Formulation A [111] rewrites the momentum equation (6.18) as:

$$(\alpha_2\rho_2 u_2)_t + (\alpha_2\rho_2 u_2^2 + \alpha_2 p)_x = p\frac{\partial \alpha_2}{\partial x} \tag{6.20}$$

Letting $\rho = \alpha_2\rho_2$, $u = u_2$, $N = N_2$ the system of equations can be written in conservation-law form as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}) \tag{6.21}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ N \end{bmatrix} \ , \ \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + \rho P \\ u N \end{bmatrix}, \ \mathbf{S} = \begin{bmatrix} 0 \\ p\frac{\partial \alpha_2}{\partial x} \\ 0 \end{bmatrix} \tag{6.22}$$

in which P may be regarded as a sort of 'pseudo pressure', given in terms of the actual gas pressure and the inter-granular stress, R.

$$P = P(\rho) = (R + p)/\rho_2$$

and

$$p = \frac{\bar{p}}{\alpha_1} = \frac{\bar{p}}{1 - \alpha_2} = \frac{\rho_2 \bar{p}}{\rho_2 - \rho}$$

$\bar{p} = \bar{p}(x)$ is the gas pressure computed by solving the gas phase equations, for a time Δt. The intergranular stress is given in terms of empirical values [45], which complicate the mathematical analysis. Neglecting R has little effect on computed solutions. For $R \equiv 0$ we have

$$P(\rho) = \frac{\bar{p}}{\rho_2 - \rho} \tag{6.23}$$

In primitive-variable form, the homogeneous ($\mathbf{S} \equiv 0$) version of (6.21) - (6.22) is

$$\begin{pmatrix} \rho \\ u \\ N \end{pmatrix}_t + \begin{pmatrix} u & \rho & 0 \\ a^2/\rho & u & 0 \\ 0 & N & u \end{pmatrix} \begin{pmatrix} \rho \\ u \\ N \end{pmatrix}_x = 0 \tag{6.24}$$

where

$$a = \sqrt{\frac{d}{d\rho}(\rho P)} \tag{6.25}$$

is the sound speed in the solid particle phase. Use of (6.23) gives

$$a = \frac{\bar{a}}{\rho_2 - \rho} \ , \ \bar{a} = \sqrt{\rho_2 \bar{p}} \tag{6.26}$$

System (6.24) is hyperbolic, with real eigenvalues

$$\lambda_1 = u - a, \quad \lambda_2 = u, \quad \lambda_3 = u + a \tag{6.27}$$

and corresponding right eigenvectors

$$K^{(1)} = \begin{bmatrix} \rho \\ -a \\ N \end{bmatrix} \ , \ K^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \ , \ K^{(3)} = \begin{bmatrix} \rho \\ a \\ N \end{bmatrix} \tag{6.28}$$

Like the other derivative terms, the $[\alpha_2]_x$ factor in the source term is only valid for smooth flows. Any shock capturing scheme, is unable to produce perfectly discontinuous solutions. Even if a physical discontinuity in α_2 is represented numerically

by only two points, the source term would not be infinite. However, with increasing grid resolution the $[\alpha_2]_x$ term may become very large. In order to maintain stability in the ODE step, this may result in a prohibitively (for an explicit solver) small time step. Therefore, merely assuming that the explicit time step for the ODE part is equal to that of the PDE parts, is likely to produce numerical difficulties. The issues involving the stability of the ODE step will be addressed in future work.

The Riemann Problem Solution for Formulation A

Figure 6.2 illustrates the three-wave structure of the solid phase Riemann problem solution. The structure is similar to that for the gas phase. The three wave families correspond to the eigenvalues λ_1, λ_2 and λ_3. The outer non-linear waves can be either shock or rarefaction waves, whilst the middle linear wave[2] is a contact wave. The solid phase Riemann problem solution is derived by finding the density ρ_* in the

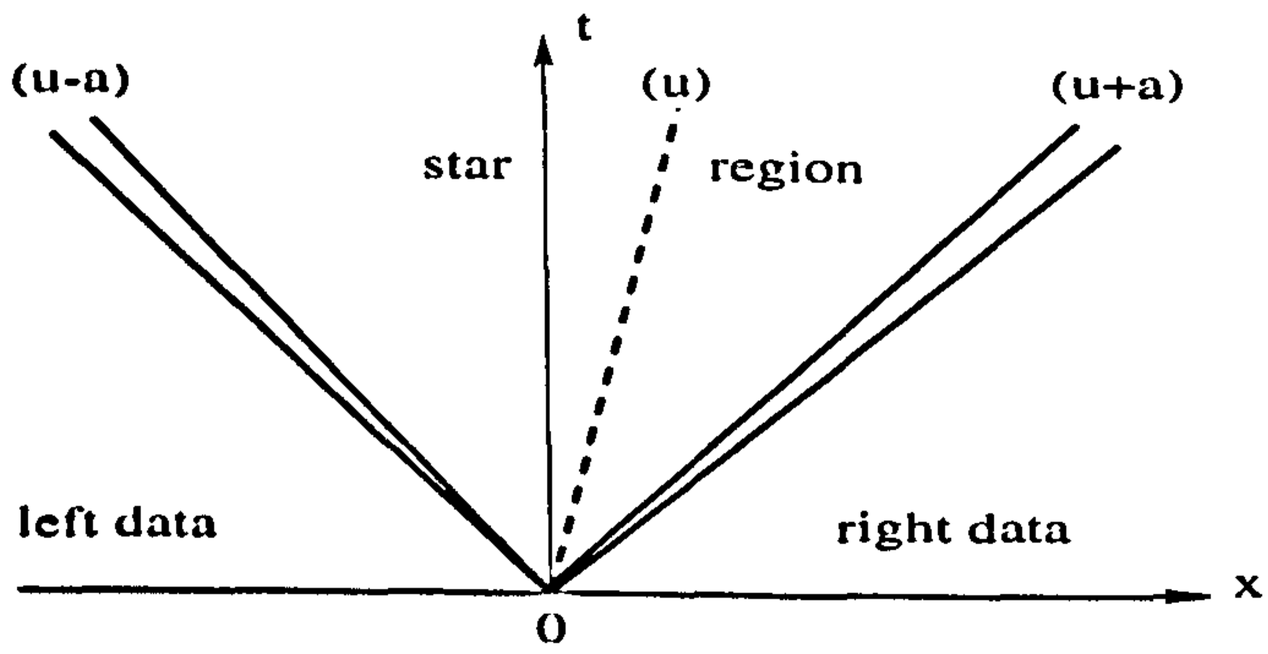


Figure 6.2: Structure of the Riemann problem solution in the $x-t$ plane

star region region between the outer non-linear waves. The exact Riemann problem solution for formulation A is summarised here; a full derivation is given in [117].

The exact solution for ρ_* between the non-linear waves is the root of the non-linear algebraic equation

$$f(\rho_*) = f_L(\rho_*) + f_R(\rho_*) + u_R - u_L = 0 \tag{6.29}$$

in which

$$f_L = \begin{cases} \bar{c}\log\left[\frac{\rho_*(\rho_2-\rho_L)}{\rho_L(\rho_2-\rho_*)}\right] & \text{if } \rho_* \le \rho_L \\ \left[\frac{(\rho_* P_* - \rho_L P_L)(\rho_* - \rho_L)}{\rho_L \rho_*}\right]^{\frac{1}{2}} & \text{if } \rho_* > \rho_L \end{cases} \tag{6.30}$$

$$f_R = \begin{cases} \bar{c}\log\left[\frac{\rho_*(\rho_2-\rho_R)}{\rho_R(\rho_2-\rho_*)}\right] & \text{if } \rho_* \le \rho_R \\ \left[\frac{(\rho_* P_* - \rho_R P_R)(\rho_* - \rho_R)}{\rho_R \rho_*}\right]^{\frac{1}{2}} & \text{if } \rho_* > \rho_R \end{cases} \tag{6.31}$$

[2] In two-dimensions the middle wave is a combined contact and shear wave.

where $\bar{c}$ is the constant

$$\bar{c} = \sqrt{\bar{p}/\rho_2}$$

Any root finding algorithm can be used to solve equation (6.29). The Newton-Raphson method generally only requires two or three iterations to find an accurate value for $\rho_\star$. An eigenvector analysis shows that $\rho_\star$ and $u_\star$ are constant in the star region, while N has two constant values $N_{\star L}$ and $N_{\star R}$.

The Solid Phase Vacuum Situation

Consider the situation in which the right-hand state is a vacuum (zero solid mass) as shown in figure 6.3. Situations of this nature, result in a single left rarefaction wave, with a contact front attached to its tail. The contact front separates a region of no vacuum (left) from a region of vacuum (right). The velocity of the non-vacuum-

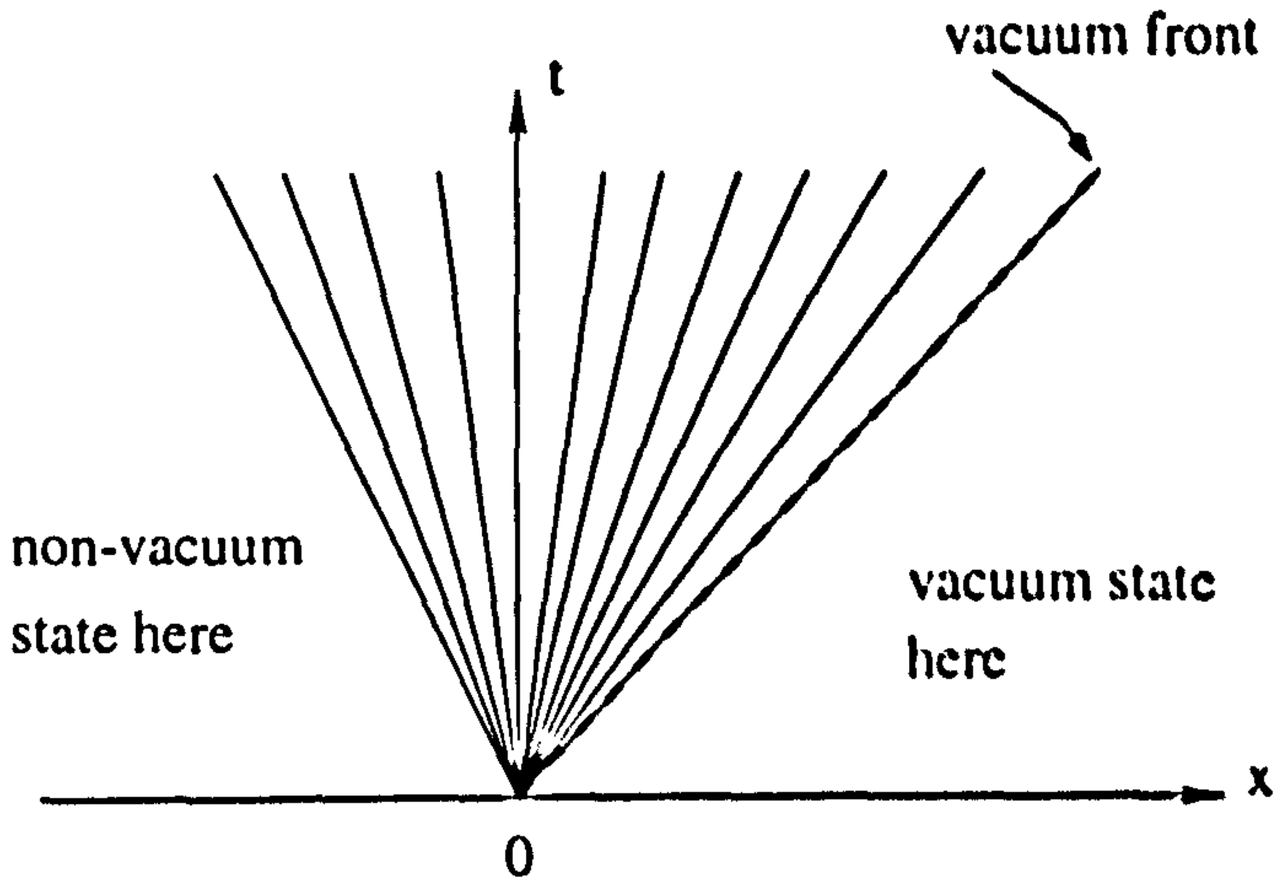


Figure 6.3: Structure of the Riemann problem solution for a non-vacuum-vacuum situation.

vacuum front is

$$S = \hat{u} - \hat{a}$$

At the front the density $\hat{\rho} = 0$. From (6.26) $\hat{a} = \bar{c}$. The particle velocity at the front $\hat{u}$ is given by

$$\hat{u} = u_L - f_L$$

where f_L is given by (6.30). However, when $\hat{\rho} = 0$, $f_L = -\infty$, i.e. the speed S of the non-vacuum–vacuum front is infinite. Similarly, the reverse situation, when the left-hand state is a vacuum, also results in an infinite wave speed at the vacuum–non-vacuum front. Movement of the solid into vacuum regions at an infinite velocity, is clearly non-physical. However, formulation A has been used for problems that involve vacuum regions, by changing the problem slightly, so that zero values of

density are replaced by a tolerance value. Even though results obtained in this way appear to be quite reasonable, a lack of robustness has been noted and attributed to the choice of tolerance for the vacuum [69].

6.3.2 Formulation B of the Particle Phase Equations

Formulation B is derived by manipulating formulation A so that the correct conditions at the non-vacuum–vacuum interface are satisfied. A pseudo pressure for the hyperbolic part of the solid phase momentum balance is chosen to be

$$P(\rho) = Q(\rho) + \tilde{P}(\rho) \tag{6.32}$$

By replacing the term $\alpha_2 p$ on the left-hand side of equation (6.20) with P, the momentum equation becomes:

$$[\rho u]_t + [\rho u^2 + Q(\rho) + \tilde{P}(\rho)]_x = p\,[\alpha_2]_x$$

Which can be written as

$$[\rho u]_t + [\rho u^2 + Q(\rho)]_x = p\,[\alpha_2]_x - [\tilde{P}(\rho)]_x \tag{6.33}$$

Thus, the pseudo pressure for the homogeneous set of equations (hyperbolic part) is

$$Q(\rho) = P(\rho) - \tilde{P}(\rho)$$

This formulation chooses $\tilde{P}(\rho)$ as a function of two parts; the first to negate $P(\rho)$ and the second to meet the requirements for the equation of state at the interface between the non-vacuum and the vacuum (see the book by Toro [116]), namely

$$Q(0) = 0,\ Q'(0) = 0 \text{ and } Q''(\rho) > 0 \text{ for } \rho > 0 \tag{6.34}$$

i.e.

$$\tilde{P}(\rho) = \frac{\rho\bar{p}}{\rho_2 - \rho} - \frac{\bar{p}}{3}\left(\frac{\rho}{\rho_2}\right)^3 \tag{6.35}$$

Note that equation (6.35) is correctly dimensionalised and that the $\frac{1}{3}$ factor in the second term conveniently simplifies the sound speed of the hyperbolic system. The momentum source term on the right hand side of equation (6.33), can be written (assuming $\frac{\partial \bar{p}}{\partial x} = 0$) as

$$S = \bar{p}\frac{\partial \alpha_2}{\partial x}\left[\frac{\alpha_2^2(1-\alpha_2)^2 - \alpha_2}{(1-\alpha_2)^2}\right] \tag{6.36}$$

This formulation yields a hyperbolic system of equations that has all the desired properties for both non-vacuum and vacuum situations. However, as with

formulation A, the $[\alpha_2]_x$ factor in the source term is likely to become very large with increasing grid resolution and may result in numerical difficulties associated with the ODE solver. At the time when this formulation was conceived, other more promising formulations (C and D) also came to light. As a result of time restraints further analysis and the implementation of this formulation has not been carried out.

6.3.3 Formulations C and D of the Solid Phase Equations

The term $\alpha_2 p_x$ in the original momentum equation (6.18), can be expanded as follows:

$$\begin{aligned} \alpha_2 p_x &= \alpha_2 \frac{\partial p}{\partial \rho}\frac{\partial \rho}{\partial x} \qquad \text{where } p = \frac{\bar{p}}{\alpha_1} = \frac{\bar{p}}{1-\alpha_2} = \frac{\rho_2 \bar{p}}{\rho_2 - \rho} \\ &= \rho\left[\frac{(\rho_2-\rho)\frac{\partial \bar{p}}{\partial \rho}+\bar{p}}{(\rho_2-\rho)^2}\right]\frac{\partial \rho}{\partial x} \end{aligned} \tag{6.37}$$

Expanding $\alpha_2 p_x$ in this way, results in a formulation that is valid for vacuum-non-vacuum interfaces. In formulation C, the assumption is made that $\frac{\partial \bar{p}}{\partial \rho} = 0$, which yields

$$\alpha_2 p_x = \frac{\rho \bar{p}}{(\rho_2 - \rho)^2}\frac{\partial \rho}{\partial x}$$

Hence, the system of equations can be written in non-conservative form as

$$\begin{bmatrix} \rho \\ \rho u \\ N \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 \\ u N \end{bmatrix}_x + \tilde{P}\begin{bmatrix} 0 \\ \rho \\ 0 \end{bmatrix}_x = 0 \tag{6.38}$$

where $\tilde{P}$ is given by

$$\tilde{P} = \frac{\rho \bar{p}}{(\rho_2 - \rho)^2} \tag{6.39}$$

As with the other formulations, this system of differential equations is only valid for smooth flows. However, by linearising P across discontinuous features, the system can be solved using a suitable numerical technique. The value $a = \sqrt{\tilde{P}}$ is analagous to the speed of sound in gas dynamics.

Formulation D is the same as formulation C, except that $\frac{\partial \bar{p}}{\partial \rho}$ is not assumed to be zero. As a result, formulation D includes an extra term that is treated as a source term. Thus, the solid phase momentum equation can be written as

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \tilde{P}\frac{\partial \rho}{\partial x} = \frac{\rho}{\rho_2 - \rho}\frac{\partial \bar{p}}{\partial x} \tag{6.40}$$

where $\tilde{P}$ is again given by (6.39) and the analagous sound speed is given by $a = \sqrt{\tilde{P}}$.

The Riemann problem solutions for formulations C and D are the same, because the homogeneous hyperbolic systems for each formulation are identical.

The Riemann Problem Solution for Formulation C

The structure of the Riemann problem solution for formulation C, is similar to that of formulation A, (illustrated in figure 6.2). Each of the outer non-linear waves can be either a shock or a rarefaction wave, whilst the middle linear wave is a contact wave. Generally, formulations A and C will have different star state solutions and corresponding wave speeds. Determining the density in the star region is the key to finding the complete Riemann problem solution. The star region density is dependent on the non-linear wave types.

Non-linear Waves are Rarefaction Waves

Assuming the non-linear waves are rarefaction waves we can connect the data states to the star region using Riemann invariants. For the left wave we have

$$\frac{d\rho}{\rho} = \frac{du}{-a}$$

or

$$du + \frac{a}{\rho}\, d\rho = 0$$

Integration in phase-space gives

$$u + \int \frac{a}{\rho}\, d\rho = \text{constant} \tag{6.41}$$

Evaluation of the integral gives

$$\int \frac{a}{\rho}\, d\rho = \bar{c} \log\left(\frac{\sqrt{\rho_2} + \sqrt{\rho}}{\sqrt{\rho_2} - \sqrt{\rho}}\right) \tag{6.42}$$

where $\bar{c}$ is the constant

$$\bar{c} = \sqrt{\bar{p}/\rho_2}$$

Therefore, the Riemann invariants for the left and right waves give

$$u + \bar{c} \log\left(\frac{\sqrt{\rho_2} + \sqrt{\rho}}{\sqrt{\rho_2} - \sqrt{\rho}}\right) = \text{constant} \tag{6.43}$$

and

$$u - \bar{c} \log\left(\frac{\sqrt{\rho_2} + \sqrt{\rho}}{\sqrt{\rho_2} - \sqrt{\rho}}\right) = \text{constant} \tag{6.44}$$

Application of (6.43) gives

$$u_\star + \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_\star}}{\sqrt{\rho_2}-\sqrt{\rho_\star}}\right) = u_L + \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_L}}{\sqrt{\rho_2}-\sqrt{\rho_L}}\right) \tag{6.45}$$

Application of (6.44) gives

$$u_\star - \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_\star}}{\sqrt{\rho_2}-\sqrt{\rho_\star}}\right) = u_R - \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_R}}{\sqrt{\rho_2}-\sqrt{\rho_R}}\right) \tag{6.46}$$

From (6.45)

$$u_\star = u_L - f_L \tag{6.47}$$

where

$$f_L = \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_\star}}{\sqrt{\rho_2}-\sqrt{\rho_\star}}\,\frac{\sqrt{\rho_2}-\sqrt{\rho_L}}{\sqrt{\rho_2}+\sqrt{\rho_L}}\right) \tag{6.48}$$

From (6.46)

$$u_\star = u_R + f_R \tag{6.49}$$

where

$$f_R = \bar{c}\log\left(\frac{\sqrt{\rho_2}+\sqrt{\rho_\star}}{\sqrt{\rho_2}-\sqrt{\rho_\star}}\,\frac{\sqrt{\rho_2}-\sqrt{\rho_R}}{\sqrt{\rho_2}+\sqrt{\rho_R}}\right) \tag{6.50}$$

Non-linear Waves are Shock Waves

Consider the left wave to be a shock wave moving with speed S_L. Transforming the frame of reference to the moving shock, yields the relative speeds

$$\hat{u}_L = u_L - S_L, \quad \hat{u}_\star = u_\star - S_L \tag{6.51}$$

Assuming a linearisation for the non-conservative system (6.38), application of the Rankine-Hugoniot conditions for each conserved variable gives

$$\rho_\star\,\hat{u}_\star = \rho_L\,\hat{u}_L \tag{6.52}$$

$$\rho_\star\,\hat{u}_\star^2 + \rho_\star\,\tilde{P} = \rho_L\,\hat{u}_L^2 + \rho_L\,\tilde{P} \tag{6.53}$$

$$N_\star\,\hat{u}_\star = N_L\,\hat{u}_L \tag{6.54}$$

We define the mass flux as

$$M_L \equiv \rho_\star\,\hat{u}_\star = \rho_L\,\hat{u}_L \tag{6.55}$$

From the momentum equation (6.53)

$$M_L = \frac{\tilde{P}(\rho_L - \rho_\star)}{\hat{u}_\star - \hat{u}_L} = \frac{\tilde{P}(\rho_L - \rho_\star)}{u_\star - u_L} \tag{6.56}$$

From equations (6.55) and (6.56)

$$M_L = a\sqrt{\rho_\star \rho_L} \tag{6.57}$$

and

$$u_\star = u_L - f_L \tag{6.58}$$

where

$$f_L = \frac{\tilde{P}(\rho_L - \rho_\star)}{M_L} \tag{6.59}$$

giving

$$f_L = \frac{a(\rho_\star - \rho_L)}{\sqrt{\rho_\star \rho_L}} \tag{6.60}$$

The speed of the shock S_L is given by equations (6.51), (6.55) and (6.57) i.e.

$$S_L = u_L - \hat{u}_L = u_L - \frac{M_L}{\rho_L} = u_L - a\sqrt{\frac{\rho_\star}{\rho_L}} \tag{6.61}$$

If the right wave is a shock wave with speed S_R, then the relative wave speeds are

$$\hat{u}_\star = u_\star - S_R, \quad \hat{u}_R = u_R - S_R \tag{6.62}$$

Applying the Rankine-Hugoniot conditions for each conserved variable, gives

$$\rho_\star \hat{u}_\star = \rho_R \hat{u}_R \tag{6.63}$$

$$\rho_\star \hat{u}_\star^2 + \rho_\star \tilde{P} = \rho_R \hat{u}_R^2 + \rho_R \tilde{P} \tag{6.64}$$

$$N_\star \hat{u}_\star = N_R \hat{u}_R \tag{6.65}$$

Now the mass flux is

$$M_R = \rho_\star \hat{u}_\star = -\rho_R \hat{u}_R \tag{6.66}$$

Algebraic manipulations lead to

$$M_R = a\sqrt{\rho_\star \rho_R} \tag{6.67}$$

and

$$u_\star = u_\star + f_R \tag{6.68}$$

in which

$$f_R = \frac{a(\rho_\star - \rho_R)}{\sqrt{\rho_\star \rho_R}} \tag{6.69}$$

and the speed of the shock S_R is given by

$$S_R = u_R - a\sqrt{\frac{\rho_\star}{\rho_R}} \tag{6.70}$$

The Complete Riemann Problem Solution for Formulation C

The exact solution for ρ_* between the non-linear waves is again the root of the algebraic equation (6.29). For formulation C the functions f_L and f_R are given by

$$f_K = \begin{cases} \bar{c} \log \left[\frac{\sqrt{\rho_2}+\sqrt{\rho_*}}{\sqrt{\rho_2}-\sqrt{\rho_*}} \frac{\sqrt{\rho_2}-\sqrt{\rho_K}}{\sqrt{\rho_2}+\sqrt{\rho_K}} \right] & \text{if } \rho_* \leq \rho_K \\ \bar{a} \frac{(\rho_*-\rho_K)}{\sqrt{\rho_* \rho_K}} & \text{if } \rho_* > \rho_K \end{cases} \tag{6.71}$$

where $\bar{a}$ is the sound speed based on the linearisation of $\tilde{P}$. Practically, computing $\tilde{P}$ from a linearisation in the density is sufficient. Thus for the Riemann problem

$$\bar{\rho} = \frac{1}{2}(\rho_L + \rho_R) \tag{6.72}$$

and

$$\bar{a}^2 = \tilde{P} = \frac{\bar{\rho}\bar{p}}{(\rho_2 - \bar{\rho})^2} \tag{6.73}$$

As with formulation A, an eigenvector analysis shows that ρ_* and u_* are constant in the star region, while N has two constant values N_{*L} and N_{*R}.

6.3.4 Numerical Implementation of formulations C and D

The solid phase system of PDE's for formulation C (6.38) is non-conservative. In fact, only the momentum equation,

$$[\rho u]_t + [\rho u^2]_x + \tilde{P}[\rho]_x = 0 \tag{6.74}$$

is non-conservative. Toro [105] has successfully computed systems of non-conservative equations, with schemes based on primitive variables. In [105] it was reported that in regions containing contact or shear waves, conservative schemes can introduce significant errors into computed solutions. Toro [105] developed an adaptive primitive-conservative method, which uses a conservative scheme in regions containing shock waves and a primitive scheme everywhere else. Formulation C can be implemented numerically, by using a mixed primitive-conservative flux to update the solution with the standard conservation formula (2.12). Hence, the conservative and primitive flux components correspond to the terms ρu^2 and $\rho\tilde{P}$ in (6.74) respectively. The conservative part is computed conventionally, with the HLLC approximate Riemann solver and the WAF method. The density ρ and the pseudo pressure (linearised) $\tilde{P}$ of the primitive part are computed separately. The pressure is computed for the cell centres and the density for the intercell boundaries. Computing the density and pressure using weighted averages (similar to the WAF scheme), provides accurate results. Computing an approximate pressure given by the mean of the pressure solutions along the two neighbouring intercell boundaries produces inferior results.

The intercell weighted average densities are computed along with the conservative flux components, with TVD limiting. Hence, the star state density components are those obtained from the HLLC solver. As with formulation A, the wave limiting is based on the jumps in variables across the the Riemann problem waves, (density for shocks and rarefactions, particle number for the contact wave). Where one of the Riemann problem states is a vacuum, computing the weights with the non-limited Courant numbers produced more consistent results. The weights are also adjusted so as to take account of the shortened integration range, (extends into the vacuum only as far as the tail of the rarefaction). Hence, instead of the weights being;

$$W_1 = \tfrac{1+\phi_1}{2}, \quad W_2 = \tfrac{\phi_2-\phi_1}{2}, \quad W_3 = \tfrac{\phi_3-\phi_2}{2}, \quad W_4 = \tfrac{1-\phi_3}{2} \quad \text{for non-vacuum,}$$

they are replaced with;

$$W_1 = 0, \qquad W_2 = 0, \qquad W_3 = \tfrac{\nu_3-\nu_2}{1-\nu_2}, \quad W_4 = \tfrac{1-\nu_3}{1-\nu_2} \quad \text{for left vacuum,}$$

$$W_1 = \tfrac{1+\nu_1}{1+\nu_2}, \quad W_2 = \tfrac{\nu_2-\nu_1}{1+\nu_2}, \quad W_3 = 0, \qquad W_4 = 0 \quad \text{for right vacuum.}$$

Where ϕ_k and ν_k are the limited and non-limited Courant numbers, respectively, associated with the k^{th} Riemann problem wave.

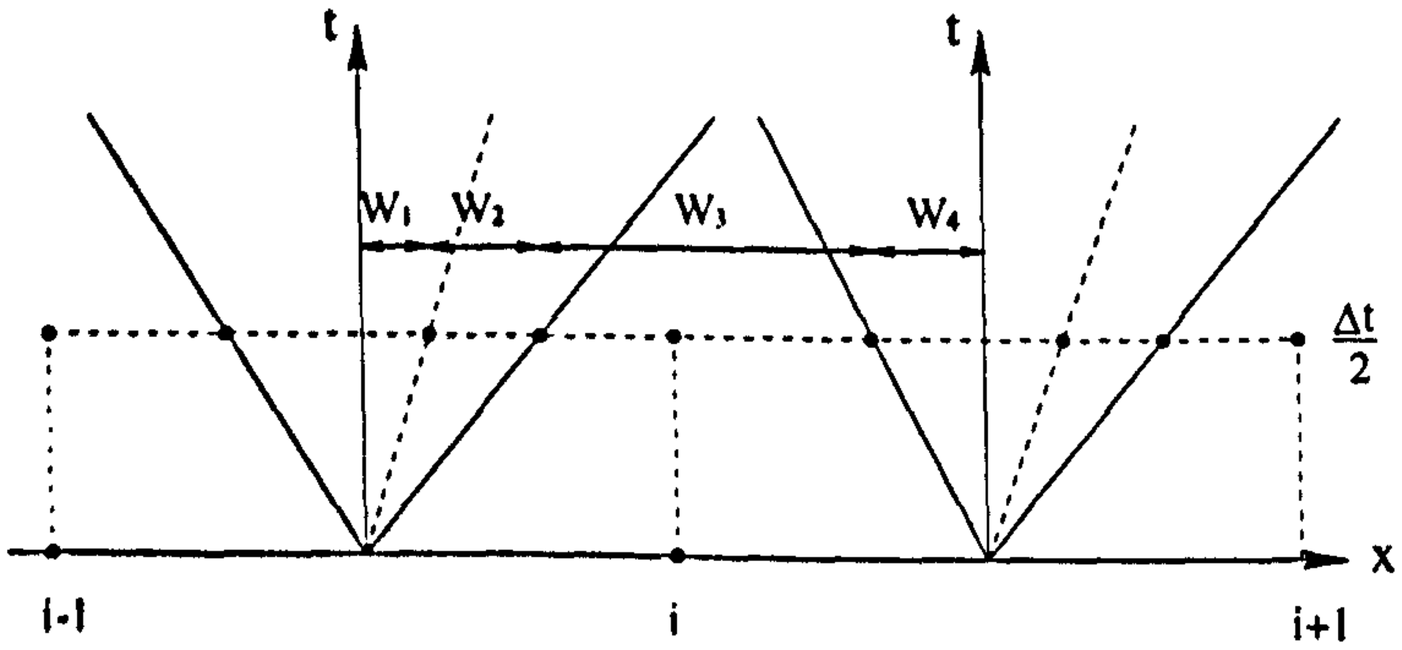


Figure 6.4: The weighted states for the primitive pressure calculation of cell i.

The pseudo pressure $\tilde{P}$ for cell i is computed from the components of the two Riemann problem wave structures that *potentially* enter cell i during the ensuing time step. Figure 6.4 depicts four pressure states for a particular pair of Riemann problem wave structures. In general, the most efficient one-dimensional algorithm calculates four different weights (some may be zero) for every 'half' Riemann problem solution. The two half Riemann problem solutions for cell i extend from $x_{i-\frac{1}{2}}$ to x_i and from x_i to $x_{i+\frac{1}{2}}$. The pseudo pressure in cell i is then calculated from the solution in eight weighted regions. Note that the pressure weights, unlike the density and flux weights, are computed using the non-limited Courant numbers. If one of the Riemann problem states is a vacuum then the weights are modified accordingly (shortened integration range).

The extra term for formulation D is used to update the solution along with all the other source terms in the ODE step, which follows the two PDE steps.

Chapter 7

Internal Ballistics Results

7.1 Introduction

The application of the CAMR approach to the internal ballistics model is straight forward. Presented here are the results for several one- and two-dimensional test problems. The tests expose some of the mathematical problems that occur with the two-phase model. Wherever possible the results obtained with the new formulations and/or the CAMR approach are compared with other numerical and experimental results. Various one- and two-dimensional results are presented in the following sections. A summary of the results is given in section 7.6.

7.2 Solid-phase Riemann Problems

Conventionally, Riemann problems with exact known solutions are used to assess the accuracy of numerical methods. Results for what is probably the most commonly solved Riemann problem test (Sod's), were presented in section 5.2.1. Exact Riemann problem solutions are not available for formulations A and D because they involve source terms. However, they can be computed numerically and should converge to the correct solution with increasing grid resolution. Formulation C, for which there is an exact solution to the Riemann problem, is equivalent to formulations A and D when the pressure is constant. (Note that, when the pressure is constant formulations C and D are identical.) Therefore, numerical solutions obtained with all three solid phase formulations (A, C and D), to a constant state Riemann problem in which the pressure is constant in space and time, should all converge to the exact Riemann problem solution given by formulation C. Figures 7.1 to 7.5 depict the density and velocity solution profiles at time 0.4ms for the Riemann problem with initial left and right data

$$\begin{aligned} \rho_L &= 1000.0\,kgm^{-3}, & \rho_R &= 100.0\,kgm^{-3} \\ u_L &= 0.0\,ms^{-1}, & u_R &= 0.0\,ms^{-1} \end{aligned}$$

The pressure $\bar{p} = 200\,MPa$ $(\forall\, x, t)$ and the solid density $\rho_2 = 1600\,kgm^{-3}$.

In figures 7.1 to 7.5 the exact solutions obtained with formulation C are represented by the solid lines and the various numerical solutions are given by the symbols. Figures 7.1 and 7.2 depict the solution profiles for formulation A, computed with 100 and 400 cells respectively. Both sets of results contain an unphysical negative spike in the velocity ahead of the shock wave. The density and velocity solutions in figure 7.2 are also affected by spurious oscillations between the rarefaction and shock waves. The results in figure 7.3 were computed on a 400 cell grid, using formulation A with a first order scheme (the WAF scheme with a limiter function that reduces it to first order accuracy). Therefore, if a first order scheme is used, then the oscillations that occured in the results shown in figure 7.2, do not appear. However, the solution becomes very diffused and a small negative velocity ahead of the shock still exists. Figures 7.4 and 7.5 depict the solution profiles for formulation C, computed with 100 and 400 cells respectively. The results obtained with formulation C appear to be a little more smeared than those obtained with formulation A. However, formulation C is much more robust than formulation A, i.e. there are no significant spurious oscillations with higher grid resolutions and no negative velocity spikes ahead of the shock wave. The cause of the error in formulation A has not been investigated, but it could well be the $p\frac{\partial \alpha_2}{\partial x}$ term in the momentum source term.

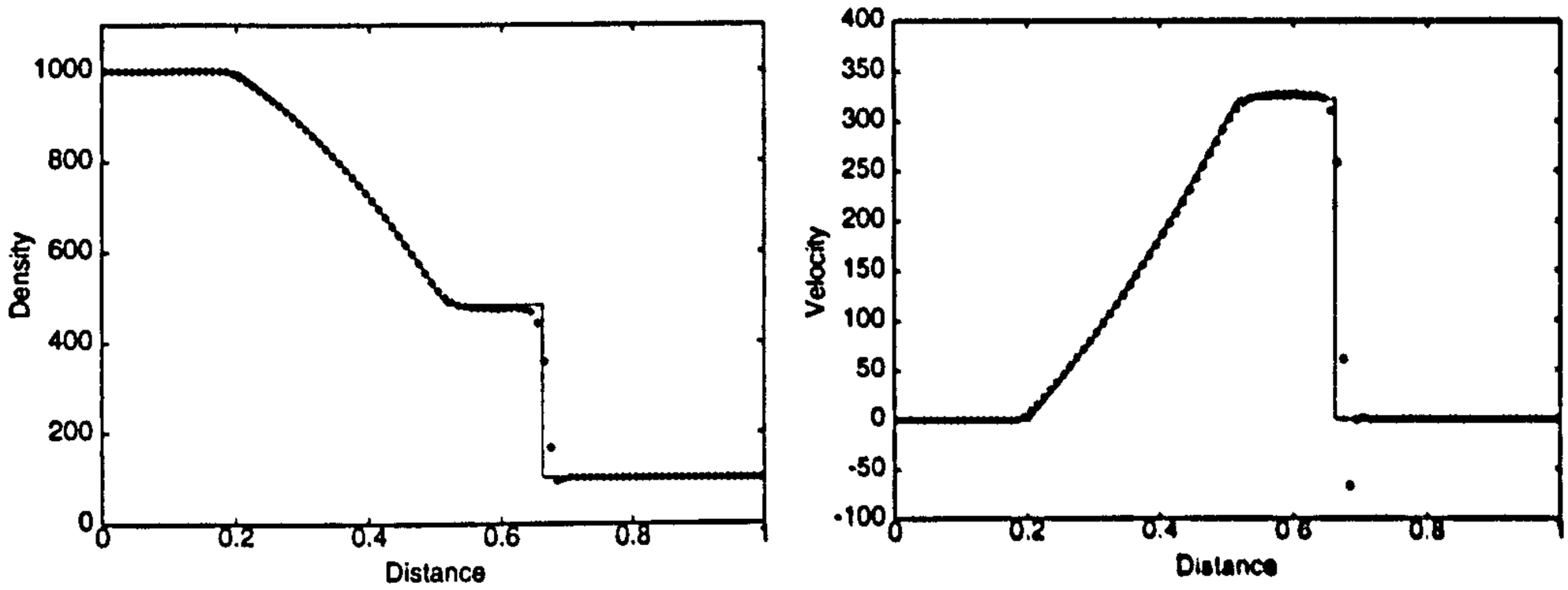


Figure 7.1: Density and velocity profiles (100 cells) at time 0.4ms, computed with formulation A.

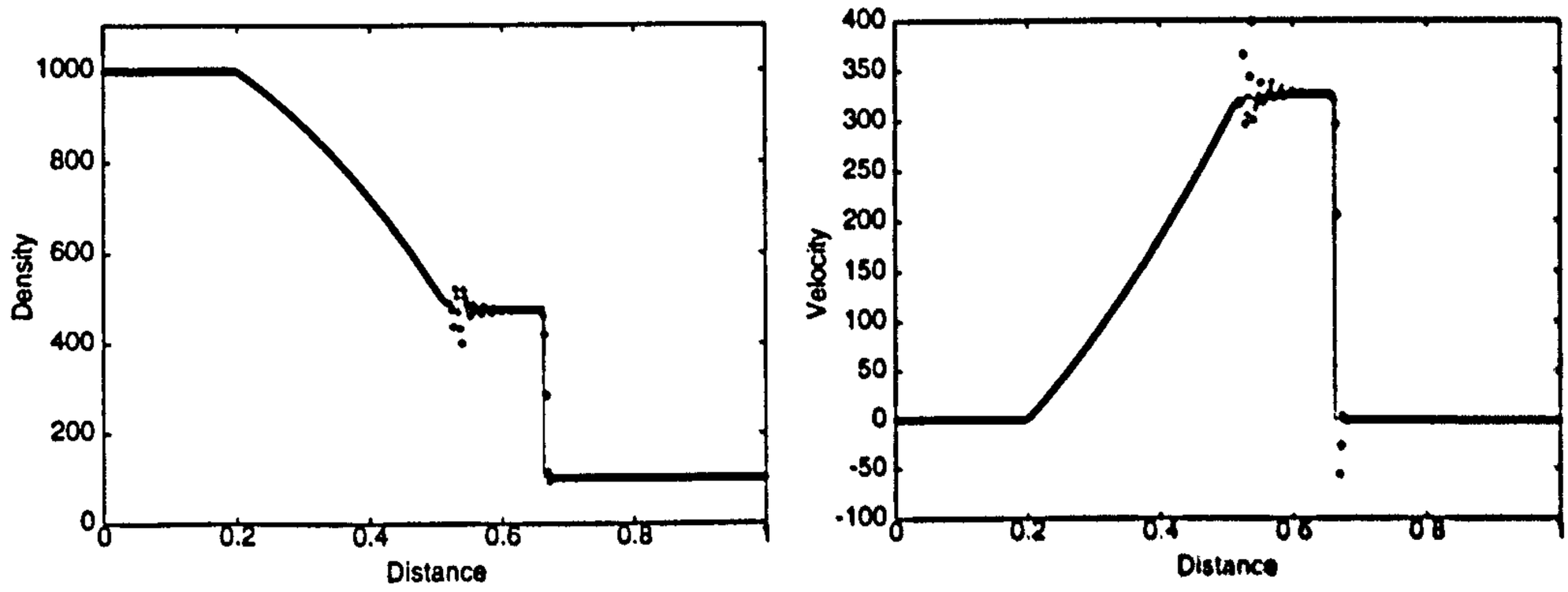


Figure 7.2: Density and velocity profiles (400 cells) at time 0.4ms, computed with formulation A.

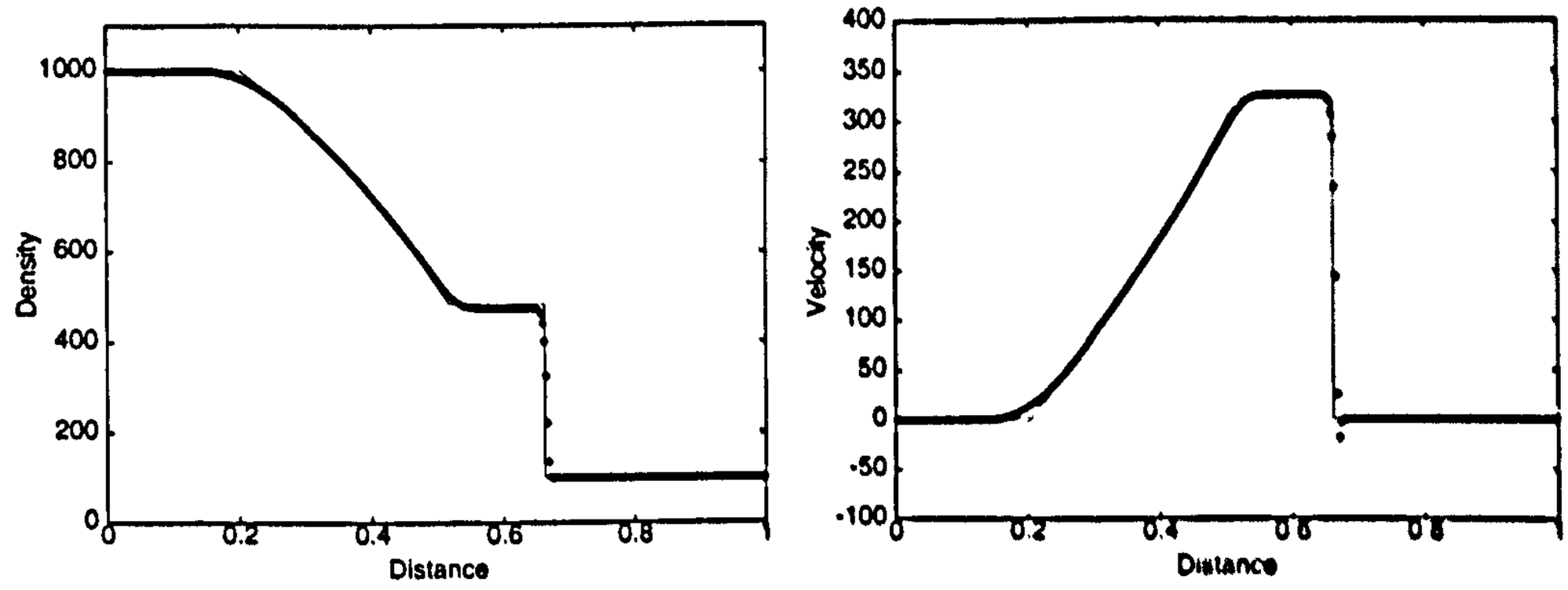


Figure 7.3: Density and velocity profiles (400 cells) at time 0.4ms, computed with formulation A using a first order scheme.

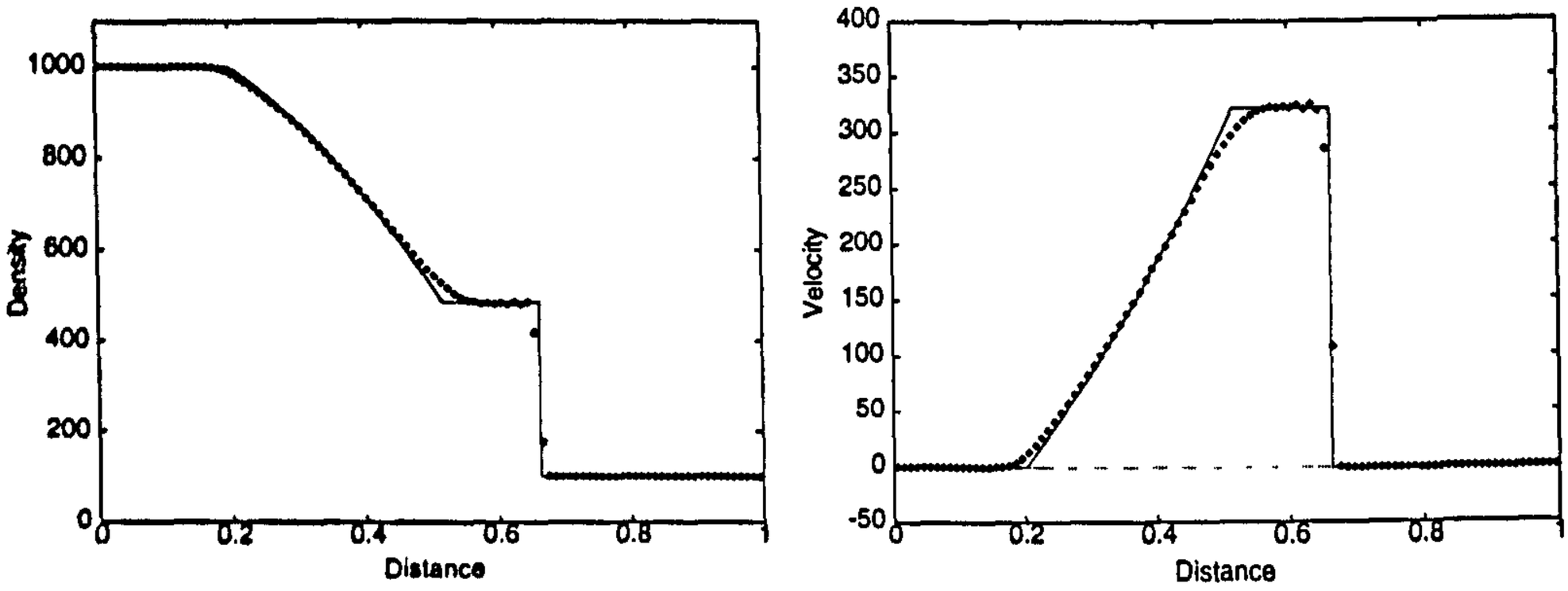


Figure 7.4: Density and velocity profiles (100 cells) at time 0.4ms, computed with formulation C.

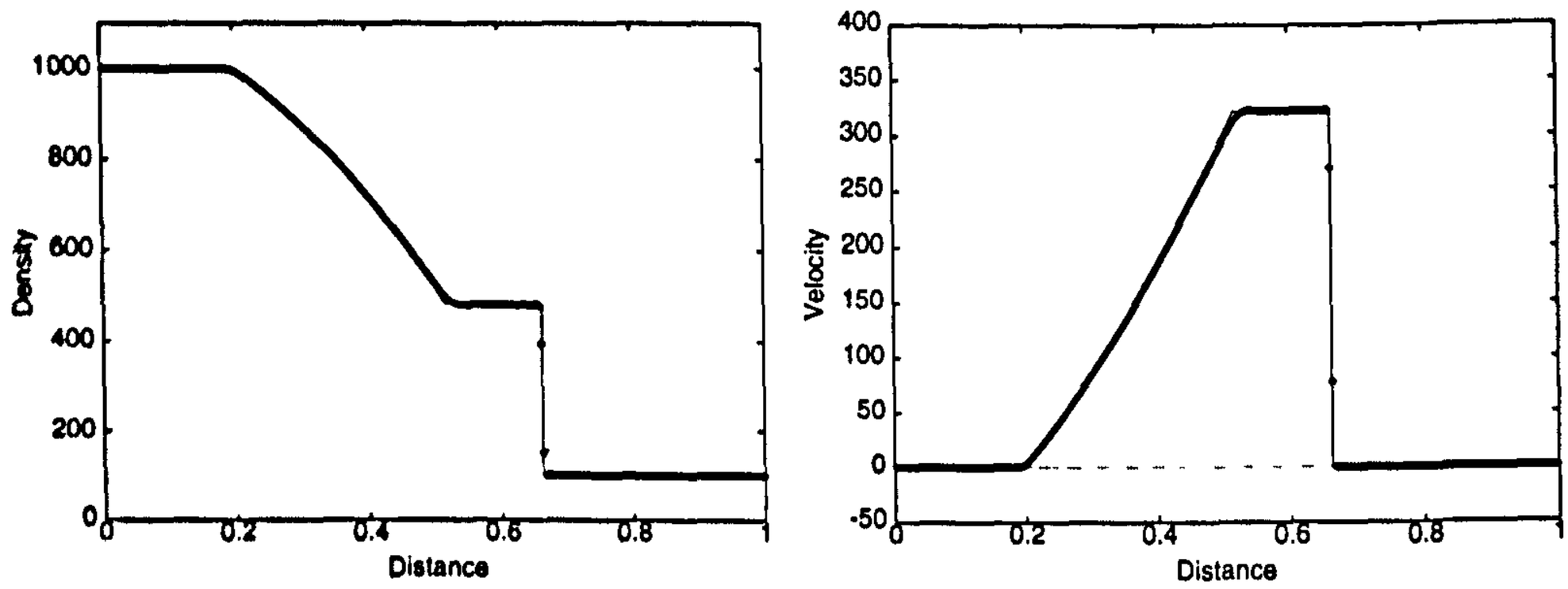


Figure 7.5: Density and velocity profiles (400 cells) at time 0.4ms, computed with formulation C.

Unlike formulation A, formulation C is valid for the non-vacuum–vacuum situations. (Remember that in the solid phase context, vacuum situations correspond to zero amounts of solid mass.) Figures 7.6 to 7.8 depict the density and velocity solution profiles at time 0.4ms, for the Riemann problem with initial left and right data

$$\begin{aligned} \rho_L &= 1000.0\,kgm^{-3}, \quad \rho_R = 0.0\,kgm^{-3} \\ u_L &= 0.0\,ms^{-1}, \quad u_R = 0.0\,ms^{-1} \end{aligned}$$

Figures 7.6, 7.7 and 7.8 depict the solution profiles for formulation C computed with 100, 400 and 800 cells respectively. The solutions are good except for the non-vacuum–vacuum front, which moves more slowly and is steeper than the exact solution. The solutions do improve with increasing grid resolution. It is unlikely that the mathematics underpinning formulation C are incorrect, since the exact solution is physically reasonable. Thus, the error at the non-vacuum–vacuum front is almost certainly due the numerical scheme adopted to compute the solution. The error could be caused by the calculation of the primitive variables at the front where the

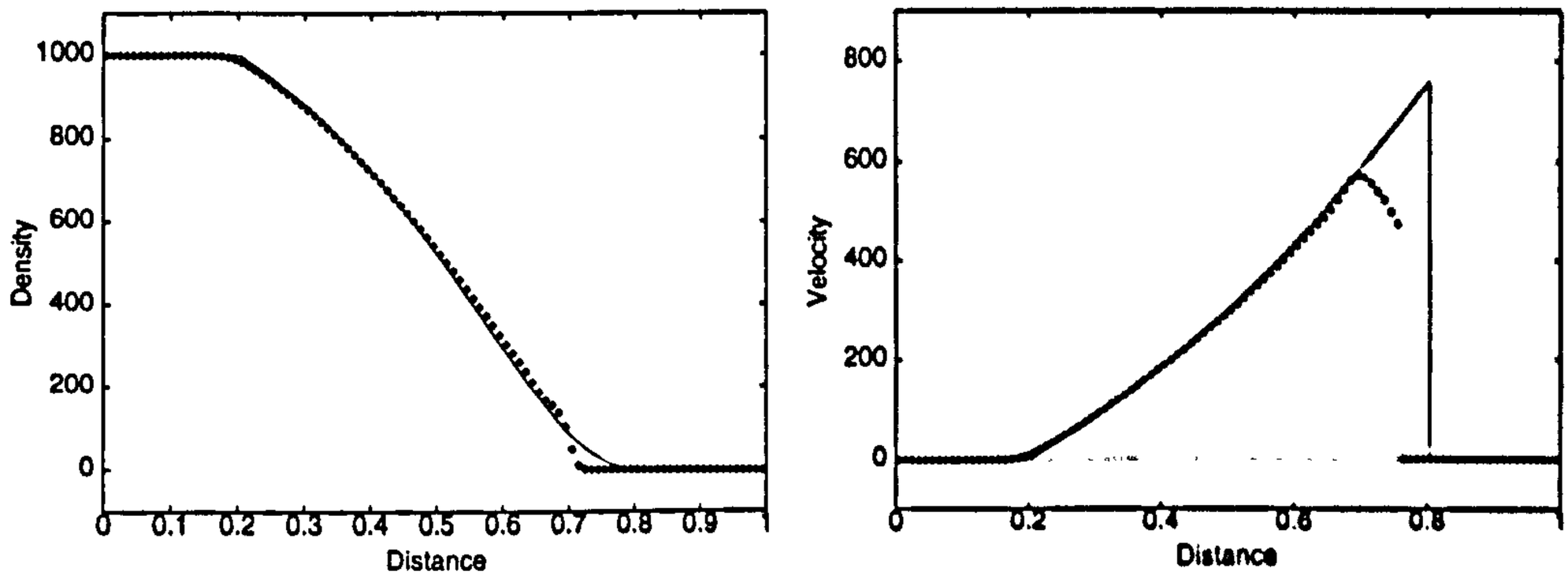


Figure 7.6: Density and velocity profiles of a vacuum state Riemann problem (100 cells) at time 0.4ms, computed with formulation C.

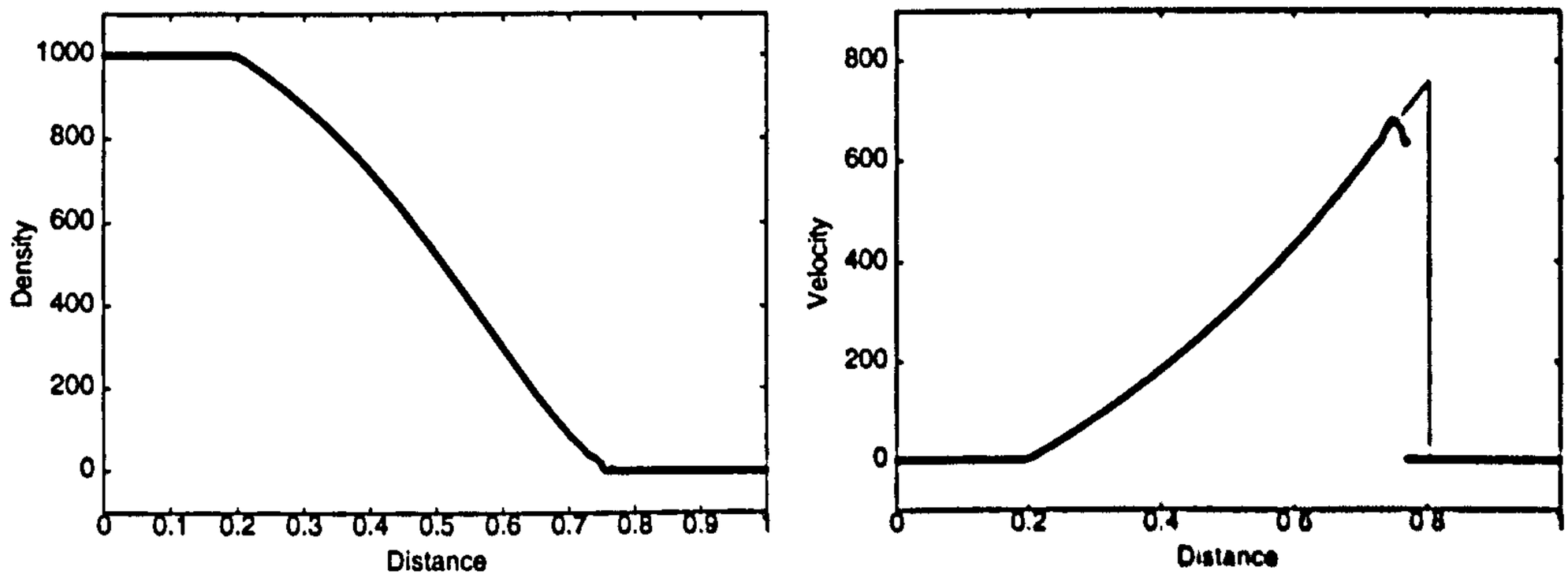


Figure 7.7: Density and velocity profiles of a vacuum state Riemann problem (400 cells) at time 0.4ms, computed with formulation C.

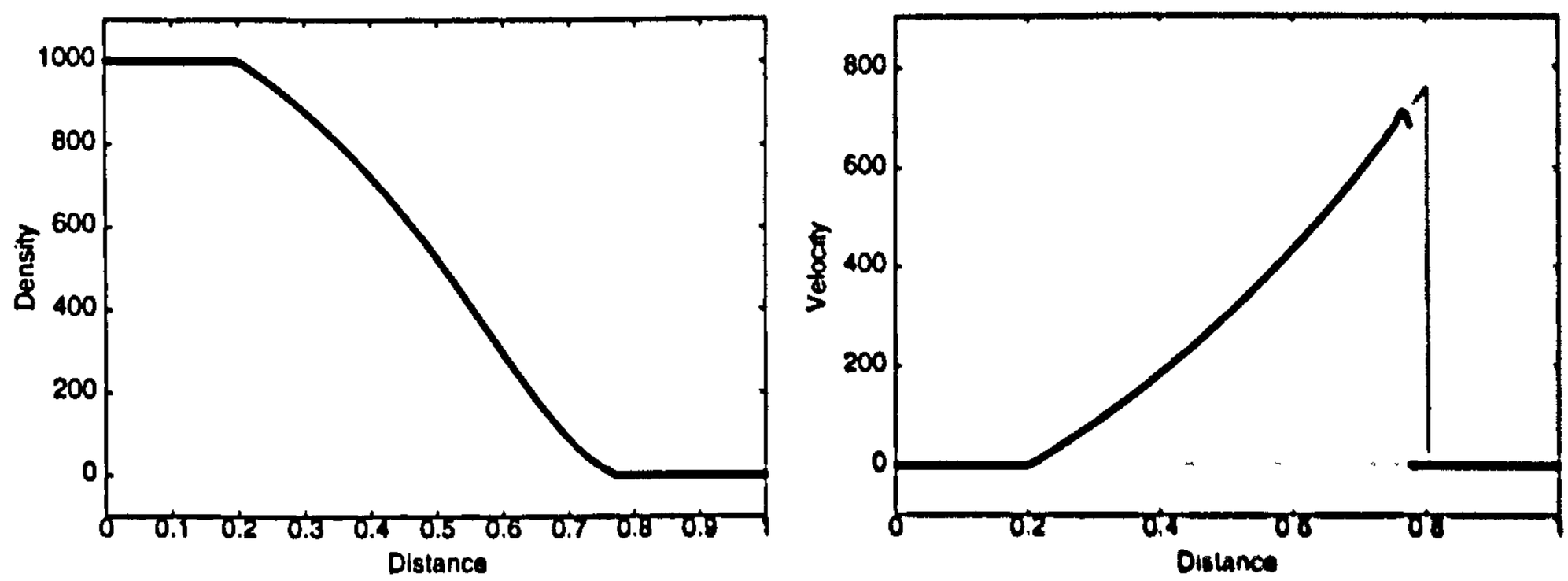


Figure 7.8: Density and velocity profiles of a vacuum state Riemann problem (800 cells) at time 0.4ms, computed with formulation C.

density is very small and the velocity very high, (i.e. the calculation of the velocity from the momentum). Thus, using a primitive scheme at the front may improve the solution. However, it could be that the primitive part of the flux in formulation C *is* the source of the error; using a primitive scheme, the author has obtained results for shallow water flow into a dry bed, which showed similar errors.

7.3 The AGARD Test Problem

A better means of assessing the various solid phase formulations is referred to as the AGARD test problem. The AGARD test is a one-dimensional internal ballistics problem, in which the shot projectile is initially positioned $0.762\,m$ from the breech. Initially, the chamber region behind the shot is at constant temperature and pressure, and is evenly distributed with seven hole solid propellant granules (tubes). Combustion is initiated by venting an energetic igniter gas between the breech and a point $0.127\,m$ from it. The shot moves as a result of the pressure acting on its base. Atmospheric pressure ($13.79\,MPa$) is the only resistance to the shot movement. The remaining initial conditions are given in table 7.1.

Presented here are the results to the AGARD test, computed with various grid resolutions for formulations A, C and D. The formulation A results are courtesy of Dr. C. Lowe, Cranfield University. In attempting to compute the AGARD solutions, it was noted that some values of the vacuum tolerance appear to be the cause of spurious pressure spikes, which often cause the source code to 'crash'. The vacuum tolerance is a numerical anomaly that is necessary to avoid computer rounding errors; the solid phase density and momentum are set to zero in grid cells whose solid density is below the tolerance. Lowe [69] also noted that the choice of the vacuum tolerance[1] similarly affects the solutions obtained with formulation A. Lowering the grid resolution appears to delay the onset of the spikes for any given vacuum tolerance. Stewart & Wendroff [99] noted that the equal pressure model is ill-posed, in that two of the eigenvalues of the Jacobian for the quasi-linear form are complex. They also noted that, by adding sufficient diffusion, the model becomes well-posed. A similar eigenvalue analysis for the mathematical model used here, was done by Dr. S.J. Billett and presented in [117]. Thus, by reducing the grid resolution, the truncation errors introduce greater numerical diffusion into the model and the system increasingly becomes well-posed. This conflicts with the aim of improving the quality of the solution by refining the grid. The amount of numerical diffusion can be increased, without lowering the grid resolution, by using a more diffusive flux limiter. Again, the idea of increasing the numerical errors in order to get more accurate results, seems to go against the whole ethos of numerical methods.

[1] Formulation A is not valid when the density is zero. Therefore, the grid cells whose density is below the vacuum tolerance, are set to the tolerance.

$45.359\,Kg$	M_p	Projectile mass
$0.0136848\,m^2$	A_p	Projectile base area
1.27	γ	Ratio of specific heats
$0.0010838\,m^3/Kg$	b	Covolume
$21.3\,Kg/mole$	W_g	Molecular weight of gas
$8313.3\,J/Kgmole$	R	Universal gas constant
$294.4\,K$	T_g	Initial gas temperature
$0.1014\,MPa$	P_g	Initial gas pressure
$9.5255\,Kg$	m_p	Total propellant mass
$0.001143\,m$	d_i	Inner diameter of propellant grains
$0.01143\,m$	d_o	Outer diameter of propellant grains
$0.0254\,m$	d_l	Propellant grain length
$1577.8\,Kg/\mathrm{m}^3$	ρ_p	Particle density
$444.4\,K$	T_i	Ignition temperature
-0.17	θ	Form function coefficient
$2.7131757\mu s/Pa^n$	β	Burning-rate coefficient
0.9	α	Pressure index
$3.7363.0\,MJ/Kg$	e_p	Propellant chemical energy
$0.2268\,Kg$	m_{ig}	Total igniter mass
$10.0\,ms$	t_{ig}	Igniter venting time
$1.57\,MJ/Kg$	e_{ig}	Igniter chemical energy

Table 7.1: The initial data for the AGARD test problem.

However, using more diffusive limiters for the solid phase, but maintaining the more accurate limiters for the gas phase, produces very similar time history results.

Breech and shot base pressure solutions for the AGARD problem were obtained using formulation C with three different solid phase flux limiters (VAN LEER, MINA and a limiter that makes the scheme first order accurate). A comparison between the VAN LEER and first order results is shown in figure 7.9. An identical comparison is shown for formulation D in figure 7.10. The results in figures 7.9 and 7.10 were obtained with quite a fine grid (80 chamber grid cells) and are typical of many results obtained with different vacuum tolerances. Of the two second order limiters, VAN LEER and MINA, VAN LEER is the least diffusive. As a result, pressure spikes are more of a problem when using the VAN LEER limiter. This is reflected in the results in figures 7.9 and 7.10. The VAN LEER limiter base pressure time histories were affected and terminated prematurely by spikes at approximately 10 and 8ms for formulations C and D respectively. Whereas the base pressure time histories obtained with the MINA limiter was terminated prematurely at 12ms for formulation C, but was unaffected for formulation D. Results were obtained with vacuum

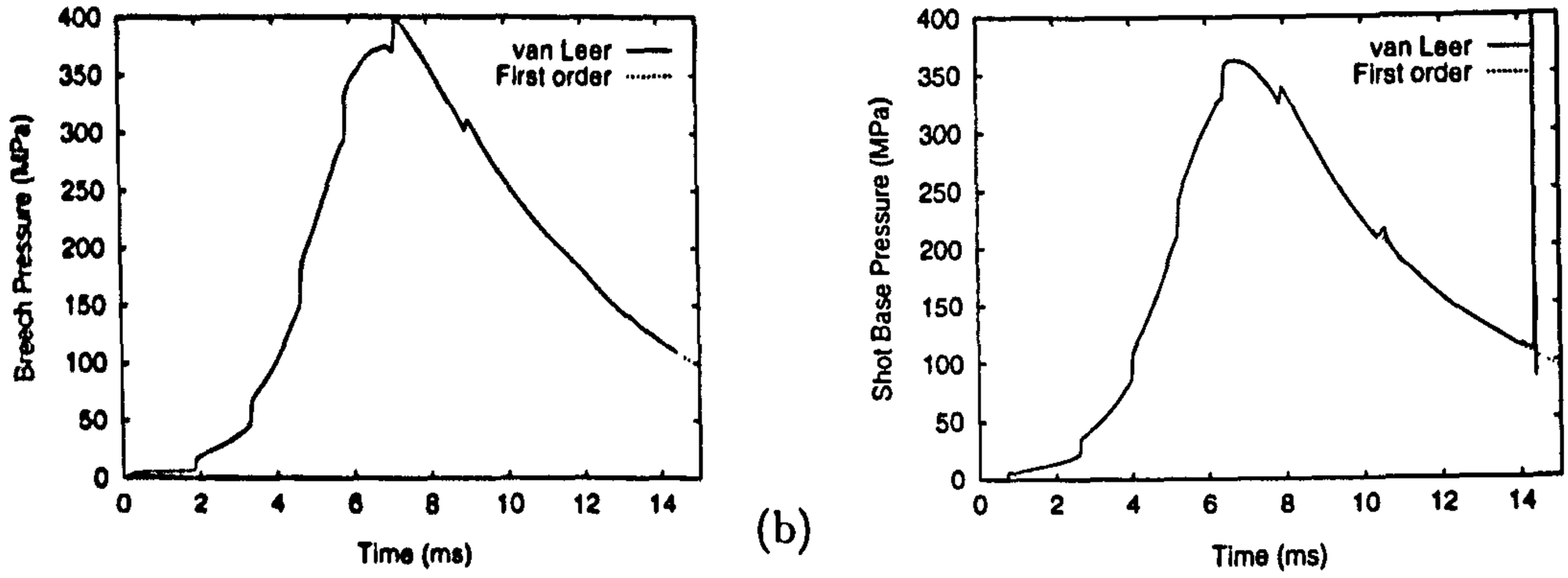


Figure 7.9: A comparison of solutions to the AGARD problem, obtained with formulation C using different solid phase limiters.

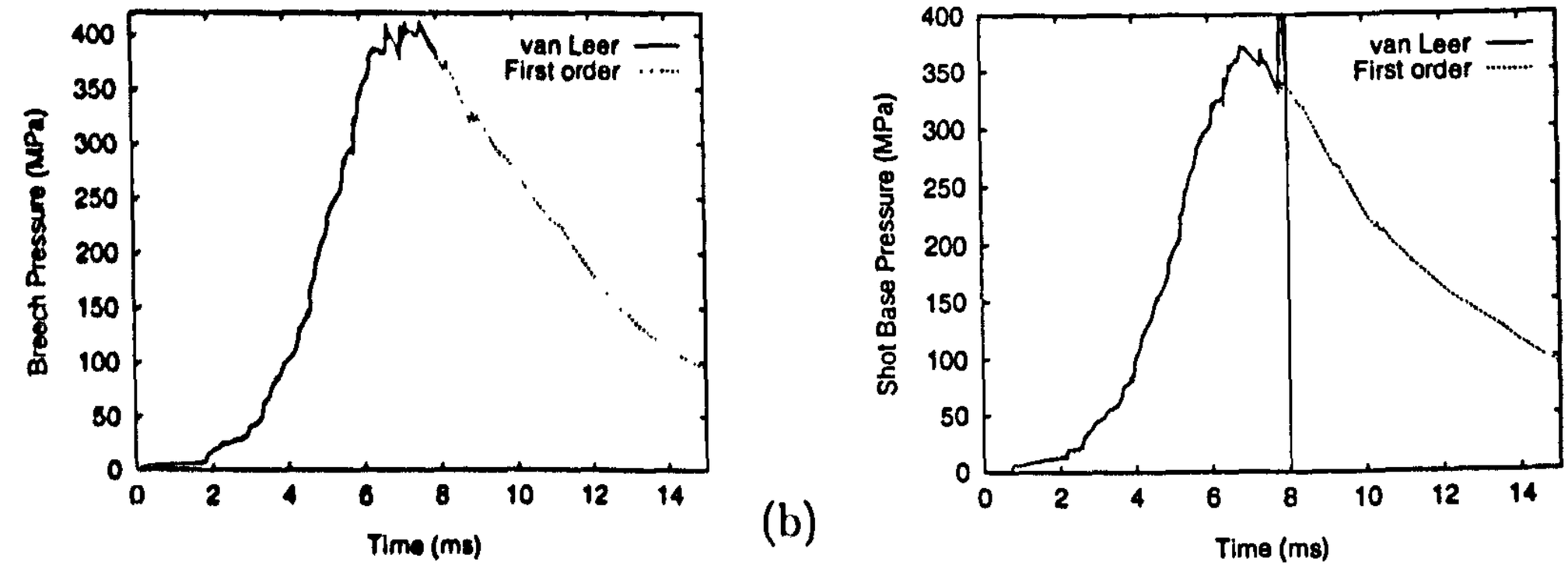


Figure 7.10: A comparison of solutions to the AGARD problem, obtained with formulation D using different solid phase limiters.

tolerances in the range 1×10^{-16} to 0.1. Pressure spikes appeared in most of the results obtained with the VAN LEER limiter. The MINA limiter also produced pressure spikes for some of the vacuum tolerances tested. No pressure spikes were found, throughout the range of vacuum tolerances, in any of the results obtained with the first order scheme. Pressure spikes were generally generated earlier with formulation D computations, compared to those generated with formulation C. It does appear that the most reliable solutions are obtained using the first order scheme for the solid phase. Hereinafter, all the results obtained with formulations C and D, have been generated using the first order scheme for the solid phase and the second order WAF scheme, with the VAN LEER limiter, for the gas phase.

Figures 7.11 to 7.13 show the breech and shot base pressure histories for the AGARD test computed with formulations A, C and D. There are no noticeable differences in the shot position and velocity histories (not shown). The results in figures 7.11, 7.12 and 7.13 were computed with grids containing 20, 40 and 80 cells in the

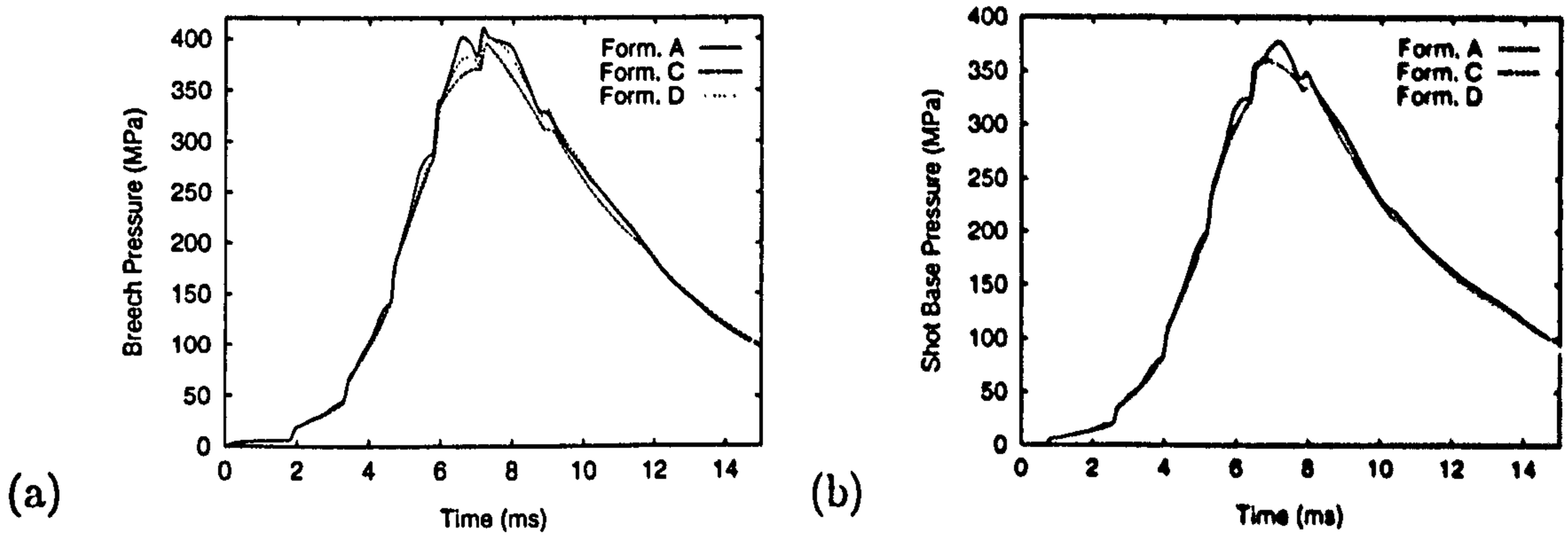


Figure 7.11: Breech (a) and base (b) pressure histories for the AGARD test, computed with formulations A, C and D, using 20 chamber cells.

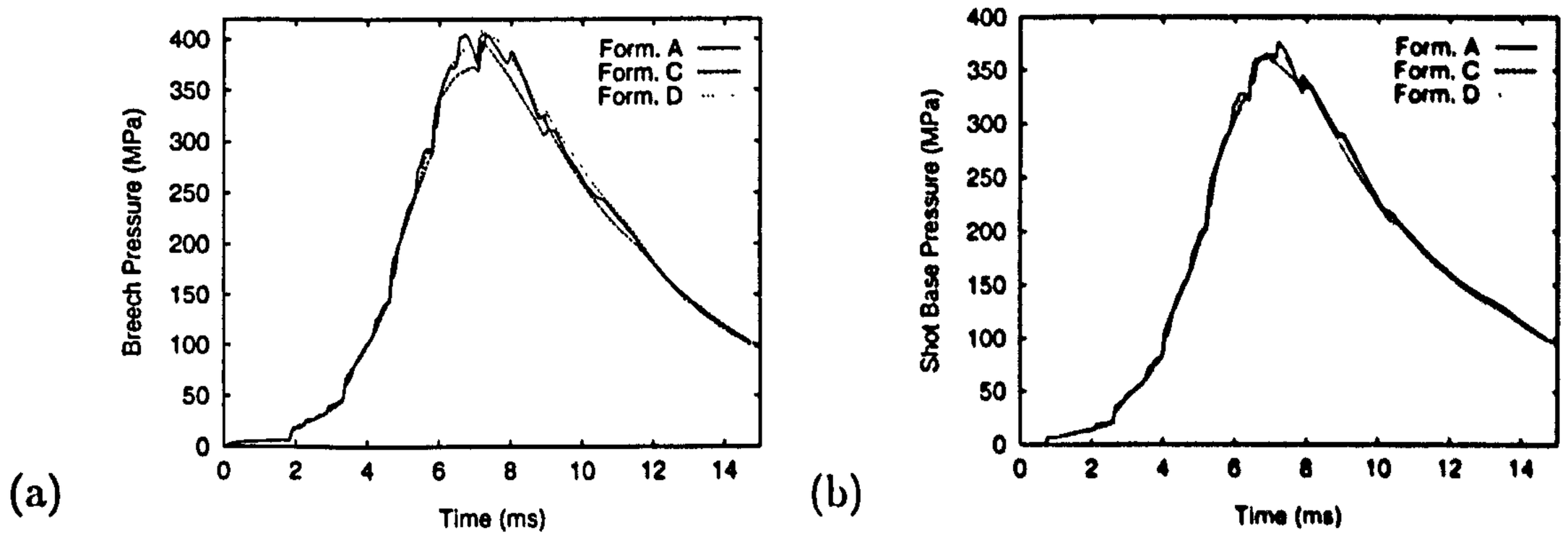


Figure 7.12: Breech (a) and base (b) pressure histories for the AGARD test, computed with formulations A, C and D, using 40 chamber cells.

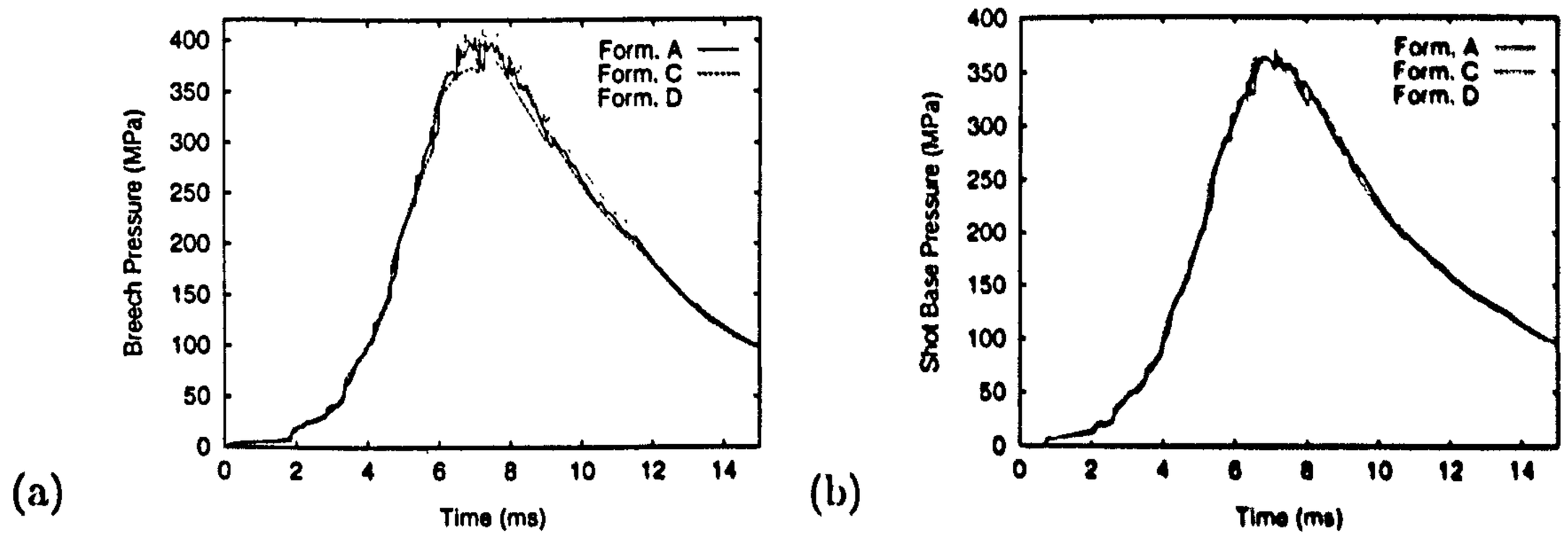


Figure 7.13: Breech (a) and base (b) pressure histories for the AGARD test, computed with formulations A, C and D, using 80 chamber cells.

chamber region respectively. The three different formulations produce roughly similar results. Generally, where differences occur, the formulation D solutions deviate away from those of formulation C, towards the solutions of formulation A. The formulation A solutions become increasingly oscillatory with increasing grid resolution. In contrast the formulation C results are much smoother and appear to converge with increasing grid resolution. The formulation D solutions are much less oscillatory than those of formulation A, but do not appear to converge with increasing grid resolution.

The three formulations differ in the way that the particle phase momentum is evaluated. Therefore, an examination of the particle velocity at an early time, is likely to reveal the causes of the time history variations shown in figures 7.11 to 7.13. Figure 7.14 shows a comparison of formulation A, C and D particle velocity solutions at time $t = 1$ms for the AGARD test computed with (a) 20, (b) 40, (c) 80 and (d) 160 chamber grid cells. The formulation C results, which assumes that the pressure does not vary with the solid phase density $\frac{\partial p}{\partial \rho} = 0$, do have non-zero particle velocities. However, they are very small in comparison to those of formulations A and D. Formulations A and D are mathematically identical, but do differ in the way that they are implemented numerically. Qualitatively, the particle velocity profiles obtained with formulations A and D are similar, but are dissimilar quantitatively. Solutions computed with either formulation A or D, do not appear to converge; as the grid resolution increases, the negative velocity spike at approximately 0.2m does appear to converge, the positive spike at approximately 0.14m from the breech continues to increases in size.

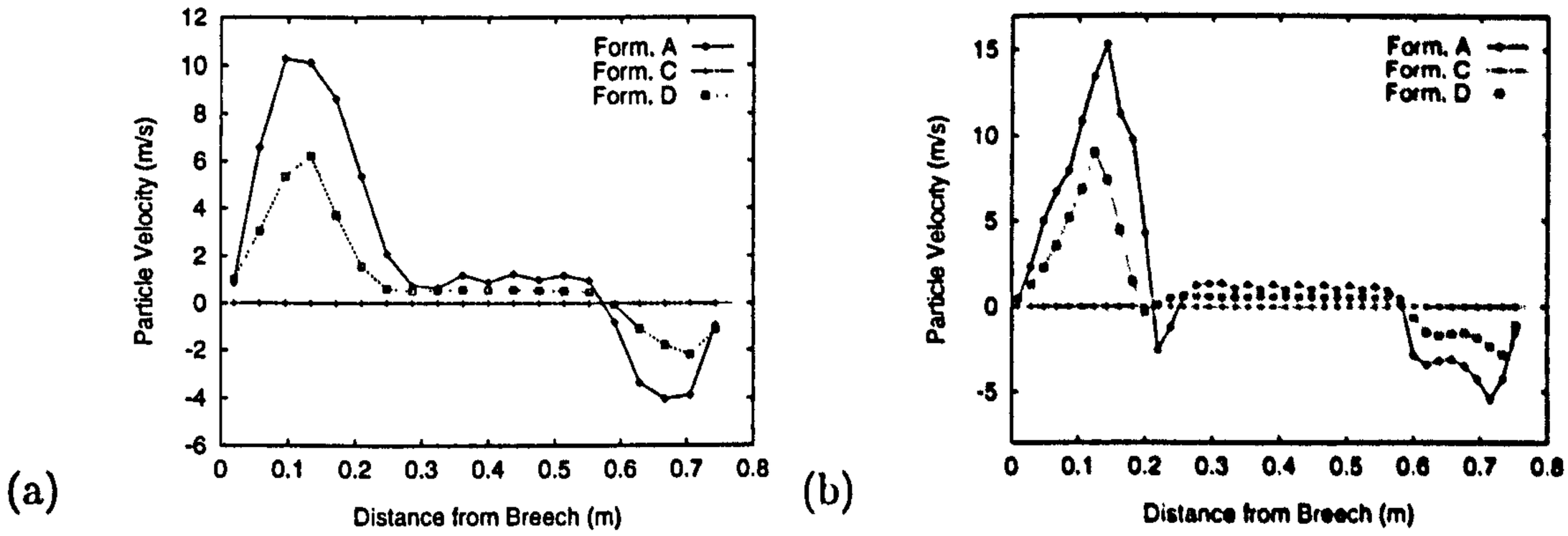


Figure 7.14: Particle velocities at t=1ms of the AGARD test solution, computed with formulations A, C and D, for four different grid resolutions (a) 20, (b) 40, (c) 80 and (d) 160 cells.

7.4 Two-dimensional AGARD Results

Presented in this section are results that have been obtained by solving the two-dimensional axi-symmetric ballistic equations, using the CAMR code. Even though the AGARD test is purely one-dimensional, a comparison of the two-dimensional CAMR solutions with the equivalent one-dimensional solutions, gives a measure of the numerical errors associated with the internal ballistics CAMR combination. The AGARD test problem is a one-dimensional problem and therfore, the flow velocity should be entirely axial, i.e. the radial component should be zero. However, it appears that numerical errors cause the two-dimensional code to produce non-zero radial velocities. The two graphs in figure 7.15 show the variation in the gas radial velocity along the chamber at t=0.1ms. The solutions in each graph were computed

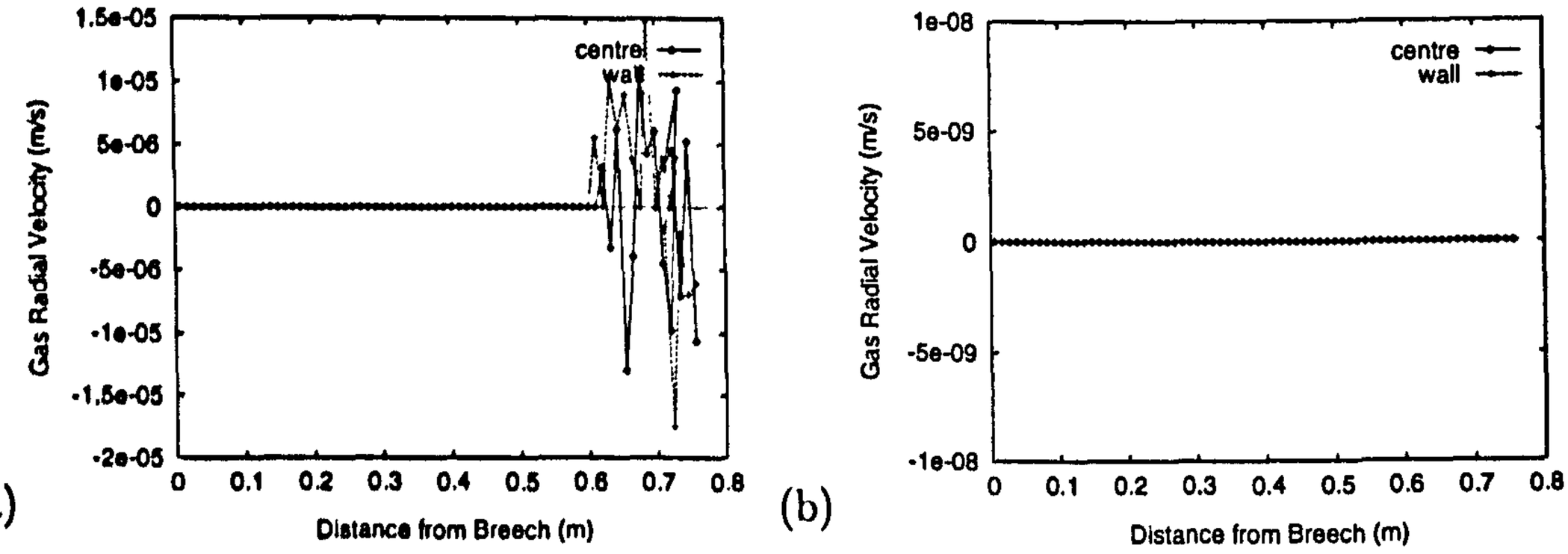


Figure 7.15: Gas Radial Velocities at time 0.1ms, computed with single- (a) and double-precision (b).

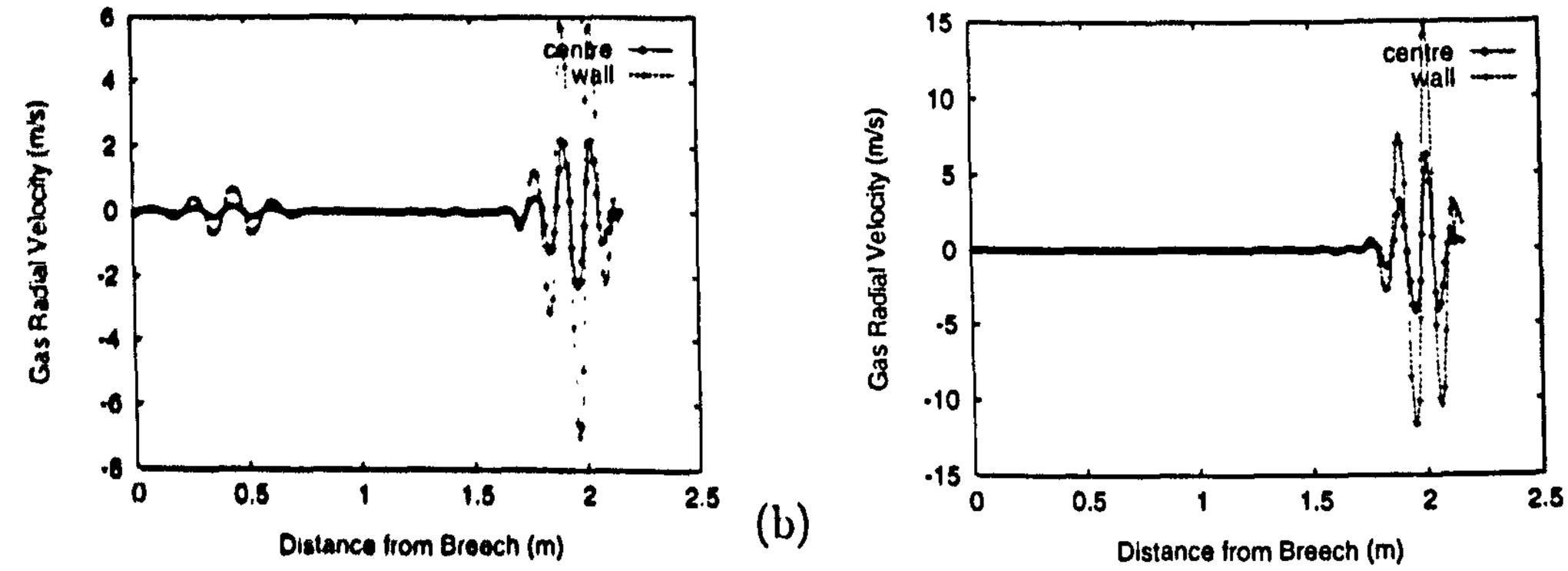


Figure 7.16: Gas Radial Velocities at time 10ms, computed with single- (a) and double-precision (b).

using single- (a) and double-precision (b) codes. The radial component of velocity

should be zero throughout the chamber. The results from the single-precision computation show errors that occur in the vicinity of the interface between the chamber and the projectile base grids, which do not appear in the results from the double-precision computation. However, even the double-precision computation is likely to produce errors over a large number of time steps. This can be confirmed by examining the radial solutions computed with different levels of precision at much later times, i.e. after many time steps. The two graphs in figure 7.16 show the variation in the gas radial velocity along the chamber at t=10ms. The solutions in the two graphs were computed using single- (a) and double-precision (b) compiler options. Both solutions show errors in the vicinity of the grid interface.

There is reasonable agreement between the 1D and the 2D CAMR time history solutions, computed using the source code compiled in double precision. The graphs in figures 7.17 and 7.18 show a comparison between the 1D and the 2D CAMR shot position (a), shot velocity (b), breech pressure (c) and shot base pressure (d) time histories. The CAMR grid structure involved a single level of refinement, which covered

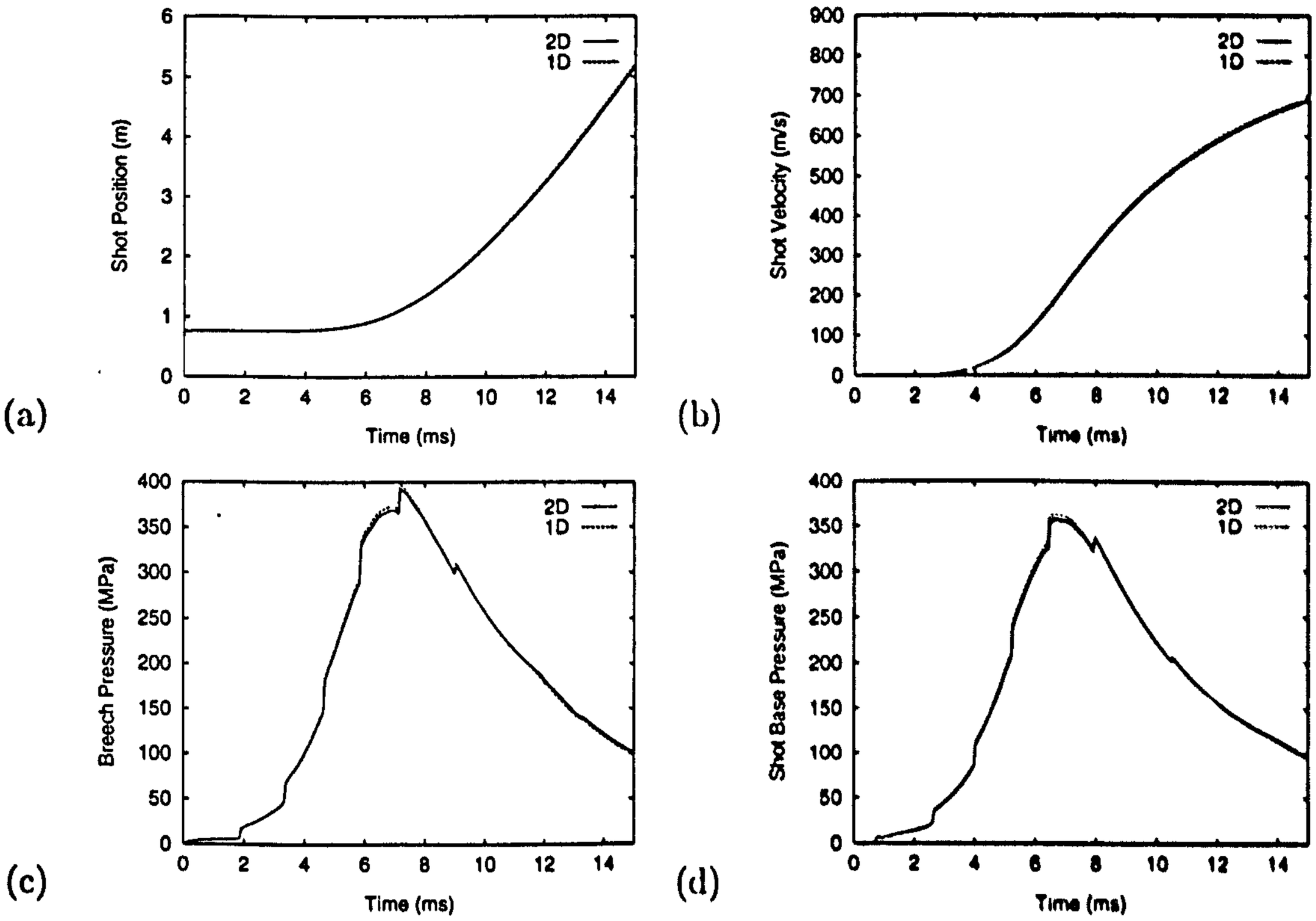


Figure 7.17: A comparison, for the AGARD test problem, between 1D and 2D CAMR time histories results computed with formulation C.

the whole of the flow domain with refined cells. The grid structure automatically changed as the movement of the shot exposed more of the barrel. The resolution

of the 1D grid was taken to be equal to that of the refined CAMR grid level. Both sets of results were obtained using the same solver strategy. A likely explanation for the cause of the deviation of the 2D results from those of the 1D, are the numerical rounding errors that generate the small radial velocities. Most probably, the errors are caused by either, the interpolation between the boundary-fitted and the Cartesian grids and/or the source term solver for the boundary-fitted grids. A full analysis to determine which lines of the CAMR source code are responsible for the rounding errors, that produce the non-zero radial velocities, is yet to be done.

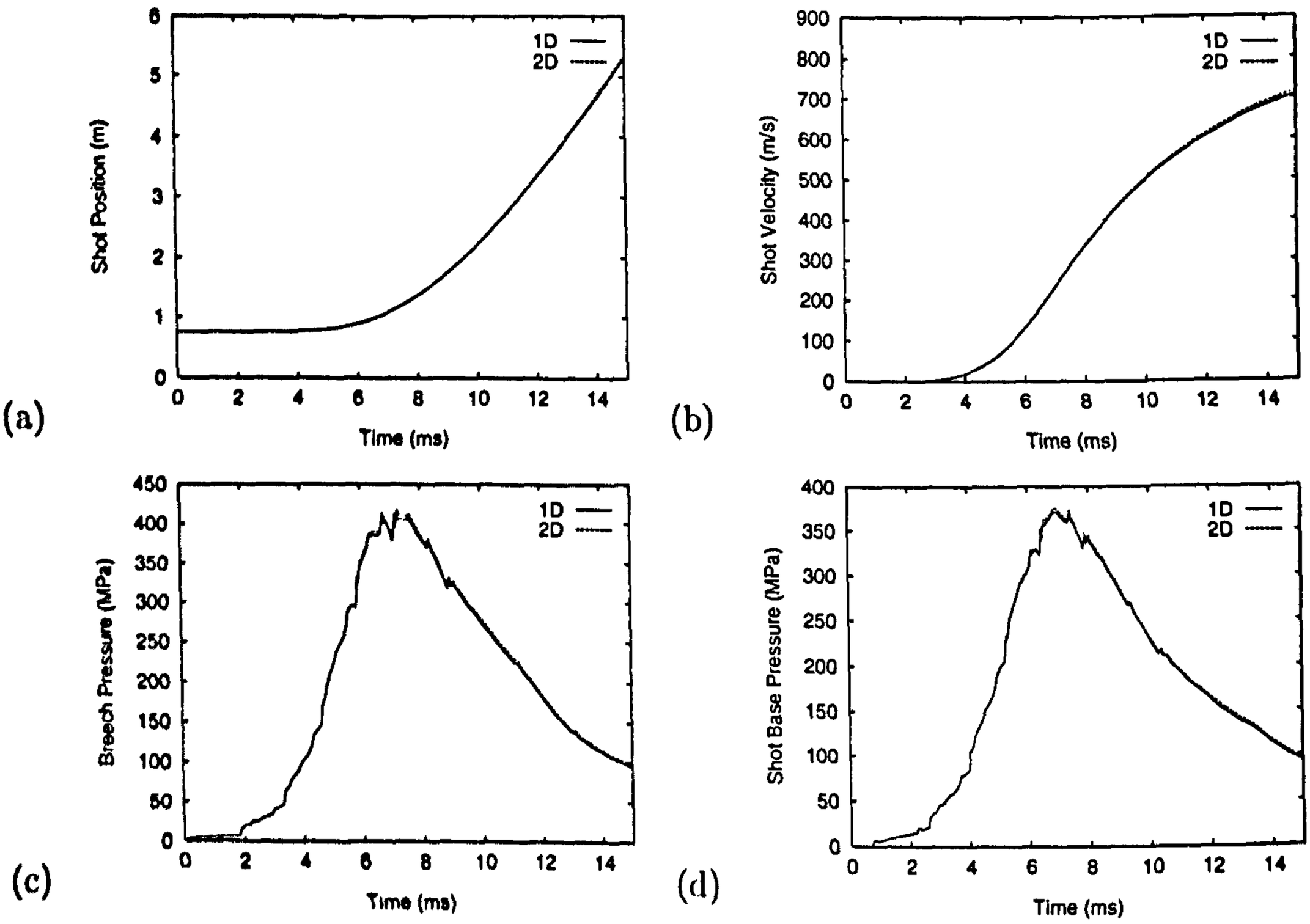


Figure 7.18: A comparison, for the AGARD test problem, between 1D and 2D CAMR time histories results computed with formulation D.

7.5 The NAVAL Test Problems

The two NAVAL problems, A and B, involve realistic gun configurations similar to that shown in figure 6.1. The geometry of the chamber is described by rotating, about the $r = 0$ axis, the straight line segments, that join the axial and radial points given in table 7.2. Initially, the shot projectile is positioned at $0.5122\,m$ from the breech. The initial temperature and pressure are constant throughout the chamber. The two NAVAL problems differ only in the initial distribution of the seven hole solid propellant granules (tubes). In the experimental NAVALB test, gravity causes the propellant to lie initially along the bottom of the chamber. Such a distribution cannot be accurately represented by an axi-symmetric model. The numerical model assumes that the propellant is evenly distributed throughout the chamber. For the NAVALA test, all the propellant is restricted to all points within $0.3615\,m$ of the breech. Combustion is initiated by the venting of the energetic igniter gas in the region between $0.1016\,m$ and $0.2718\,m$ from the breech. The remaining initial conditions are given in table 7.3. The projectile moves under the influence of the pressure acting on its base and a resistance force. The graph in figure 7.19 depicts the empirical resistance data, in the form of a resistive pressure versus the axial distance from the breech. Experimental results[2] are available for the two tests, in the form of pressure histories obtained by gauges at three different chamber locations; 0.00686m, 0.27940m and 0.49280m from the breech.

x	r
0.00000	0.03810
0.01181	0.04445
0.02934	0.047245
0.50700	0.04699
0.54220	0.03810
4.00000	0.03810

Table 7.2: Coordinate data of the chamber for the NAVAL problems.

Two-dimensional CAMR results, obtained with formulation C, are compared with both the experimental solutions and one-dimensional results, obtained with formulation A (courtesy of Dr. S.J. Billett [16]). Again, the CAMR grid structure involved a single level of refinement, which automatically changed as the movement of the shot exposed more of the barrel. Figures 7.20 and 7.21 show the (a) shot positions, (b) shot velocities, (c) breech pressures, (d) shot base pressures and (e) to (g) the three gauge pressures time histories for the NAVALA and NAVALB test

[2]Provided by the DERA, Fort Halstead.

$5.95\,Kg$	M_p	Projectile mass
$0.00456\,m^2$	A_p	Projectile base area
1.2676	γ	Ratio of specific heats
$0.001125\,m^3/Kg$	b	Covolume
$21.535\,Kg/mole$	W_g	Molecular weight of gas
$8313.3\,J/Kgmole$	R	Universal gas constant
$298.0\,K$	T_g	Initial gas temperature
$0.1013\,MPa$	P_g	Initial gas pressure
$1.87\,Kg$	m_p	Total propellant mass
$0.000361\,m$	d_i	Inner diameter of propellant grains
$0.004526\,m$	d_o	Outer diameter of propellant grains
$0.009957\,m$	d_l	Propellant grain length
$1611.0\,Kg/\mathrm{m}^3$	ρ_p	Particle density
$435.0\,K$	T_i	Ignition temperature
-0.17	θ	Form function coefficient
$0.038026\,s/Pa^n$	β	Burning-rate coefficient
0.449	α	Pressure index
$3.5127.0\,MJ/Kg$	e_p	Propellant chemical energy
$0.017037\,Kg$	m_{ig}	Total igniter mass
$14.0\,ms$	t_{ig}	Igniter venting time
$1.148\,MJ/Kg$	e_{ig}	Igniter chemical energy

Table 7.3: The initial data for the NAVAL test problems.

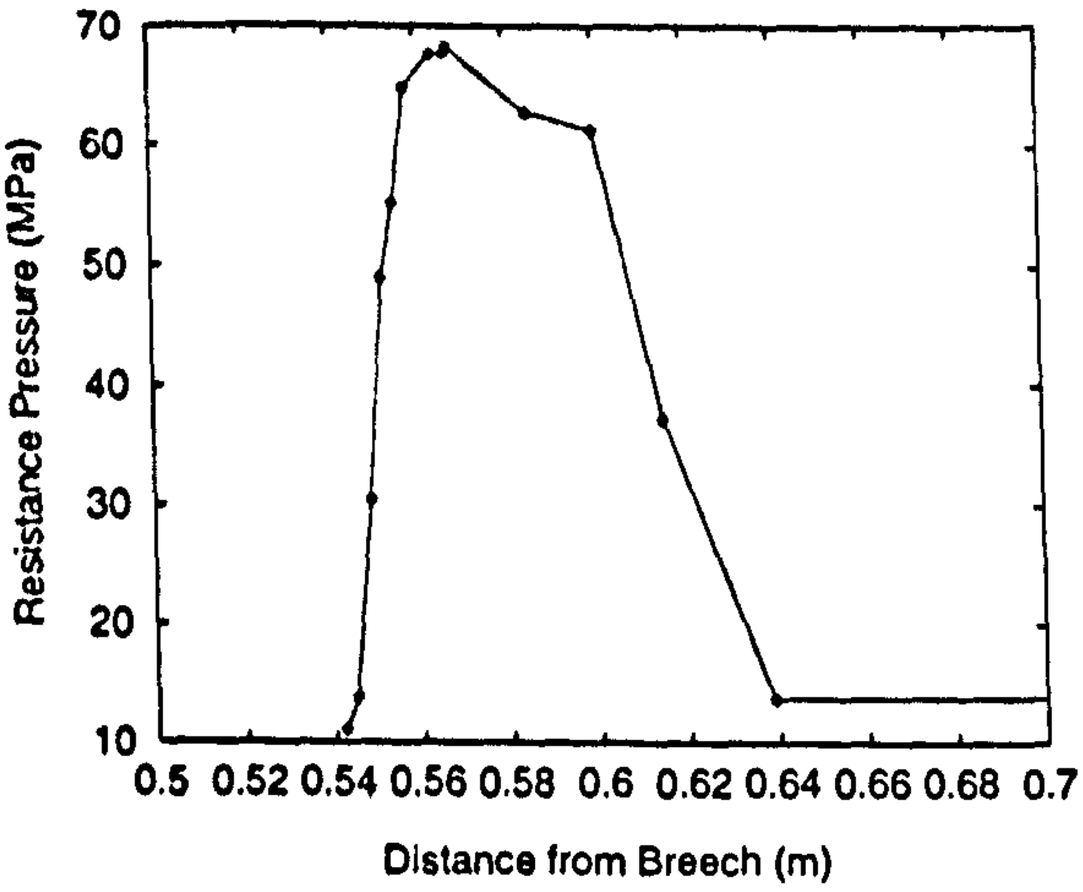
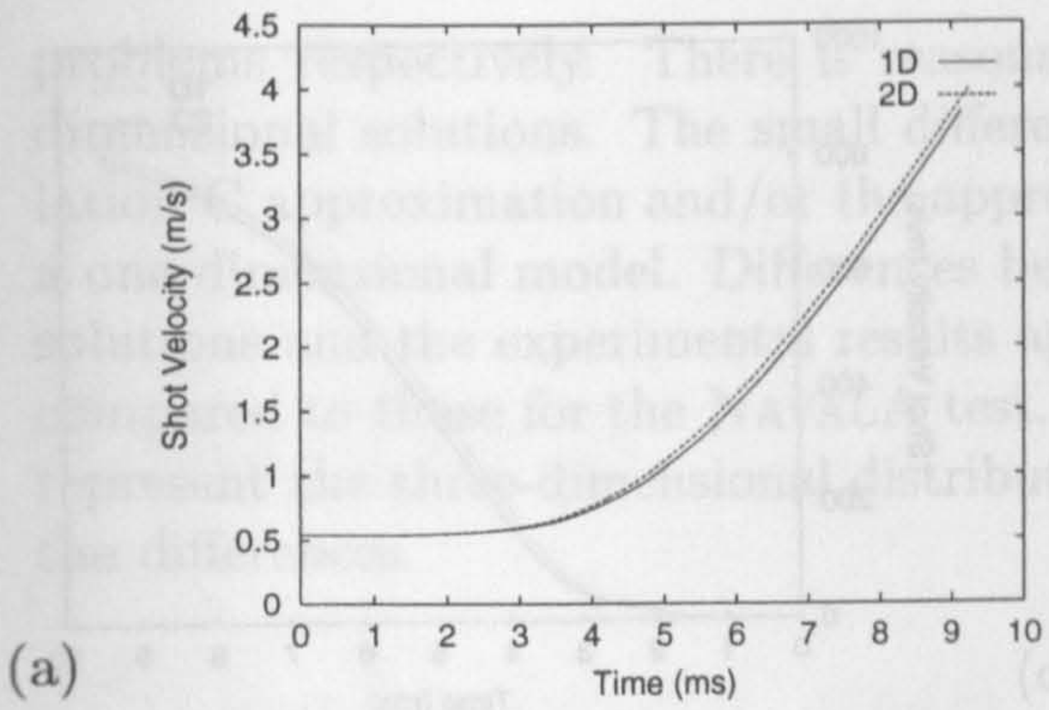
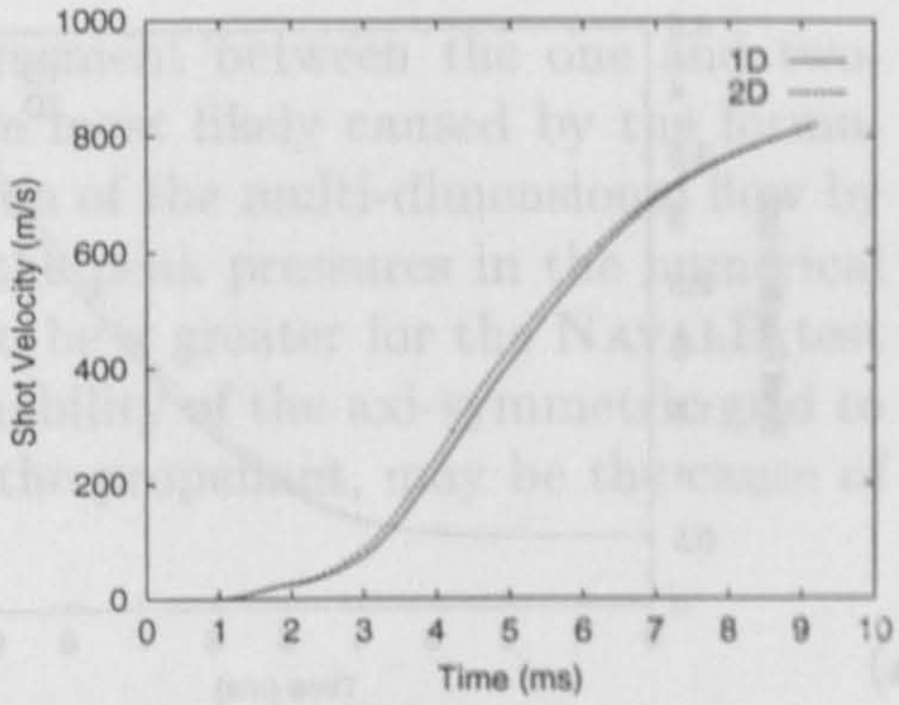


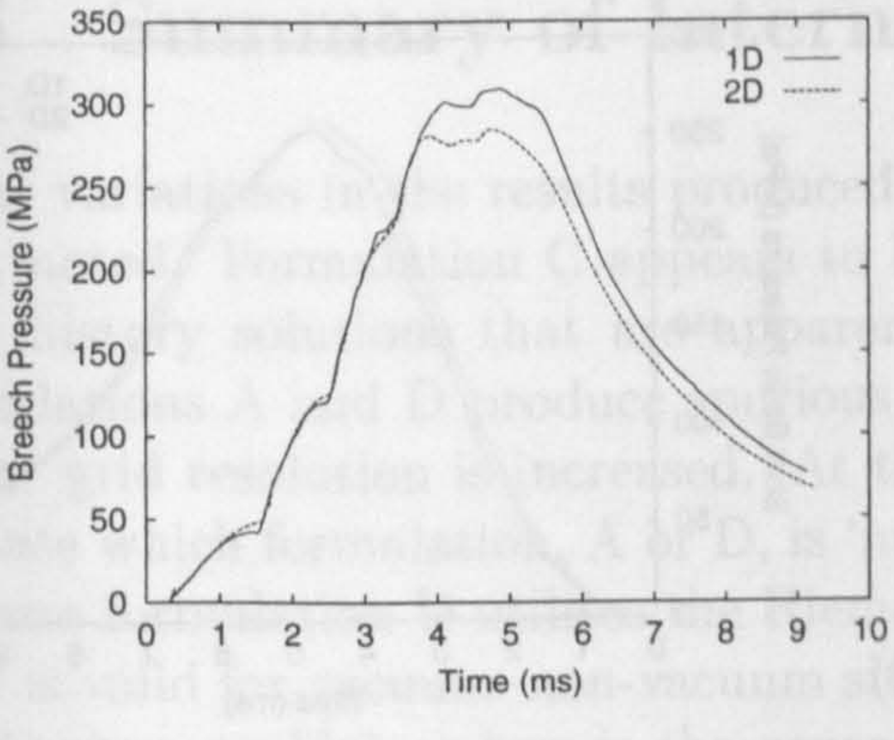
Figure 7.19: Projectile pressure resistance profile for NAVAL problems.



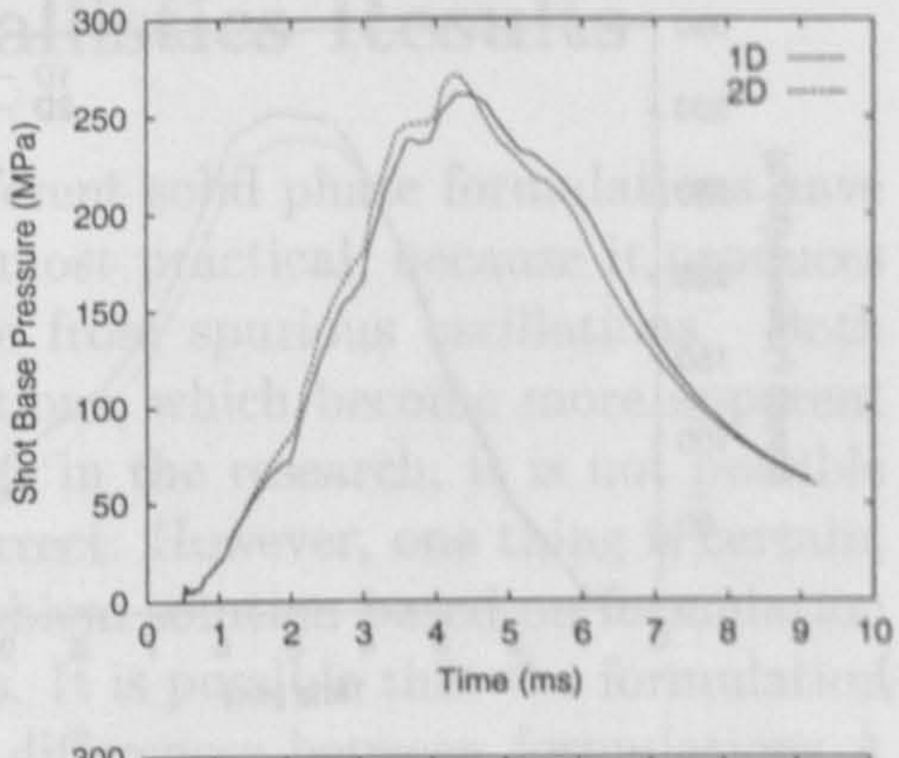
(a)



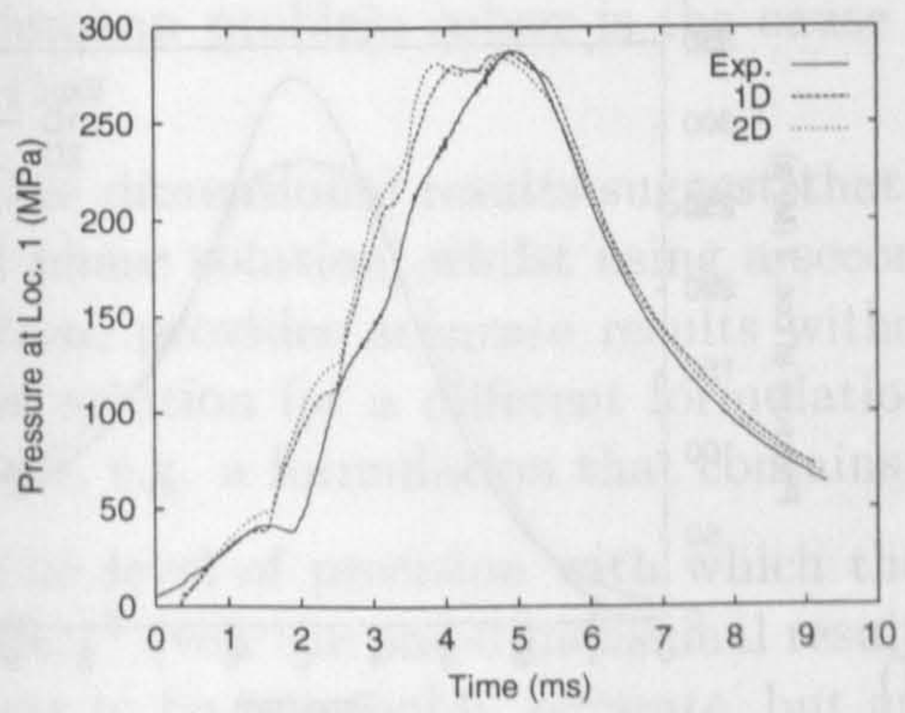
(b)



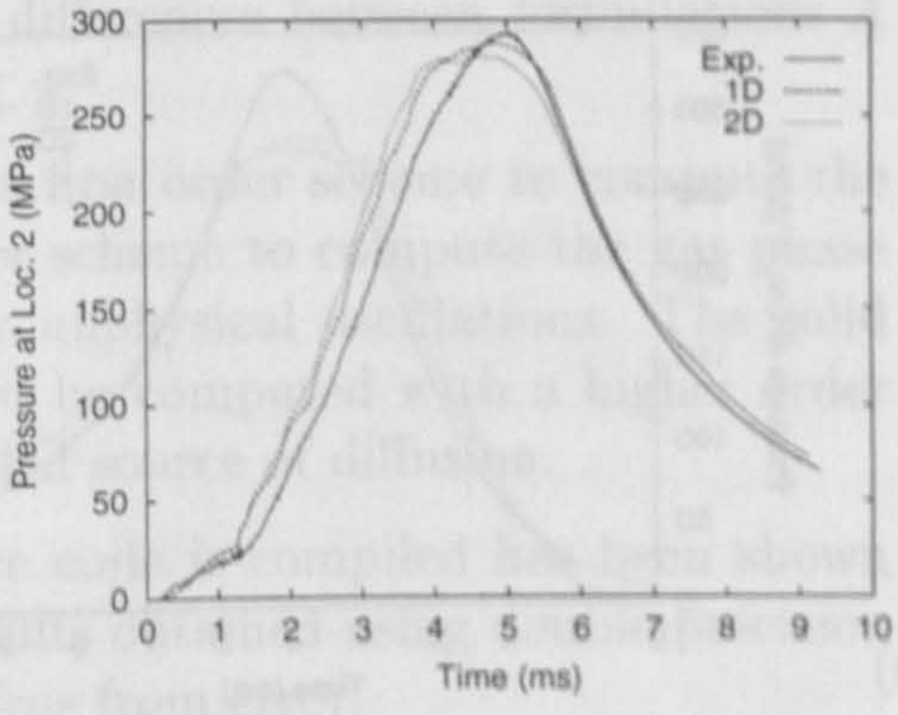
(c)



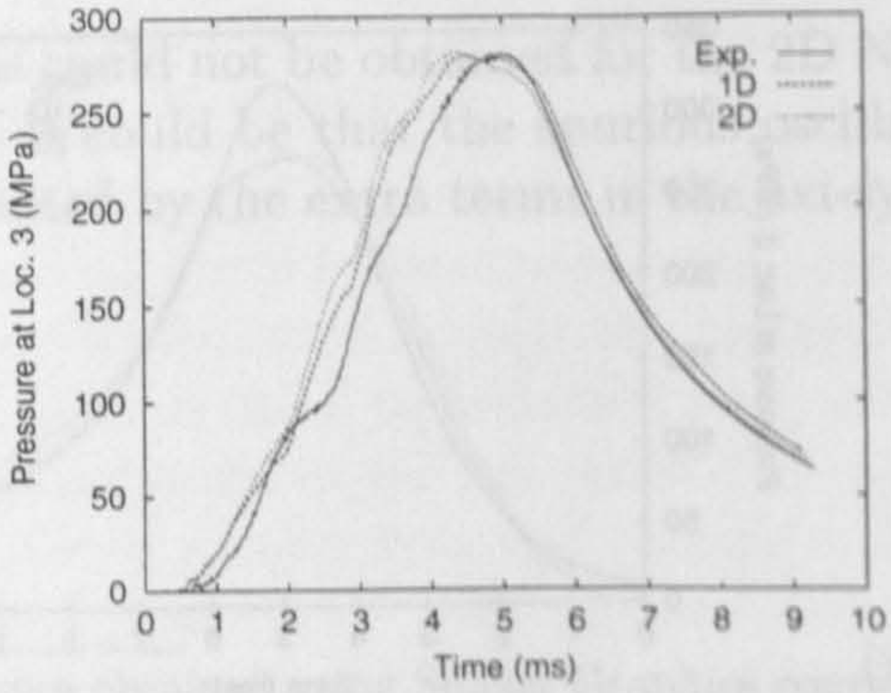
(d)



(e)

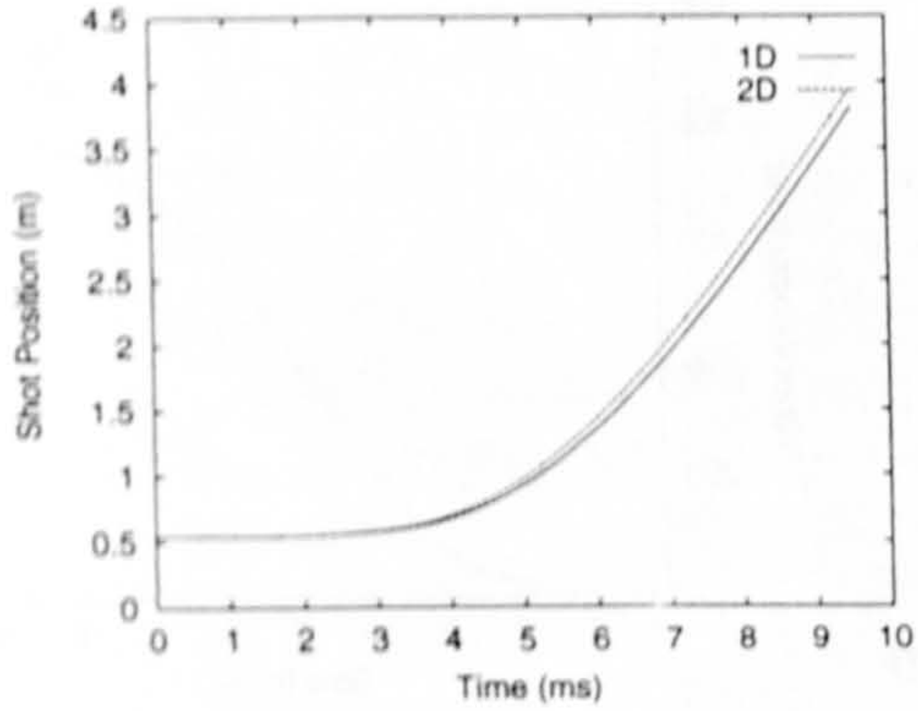


(f)

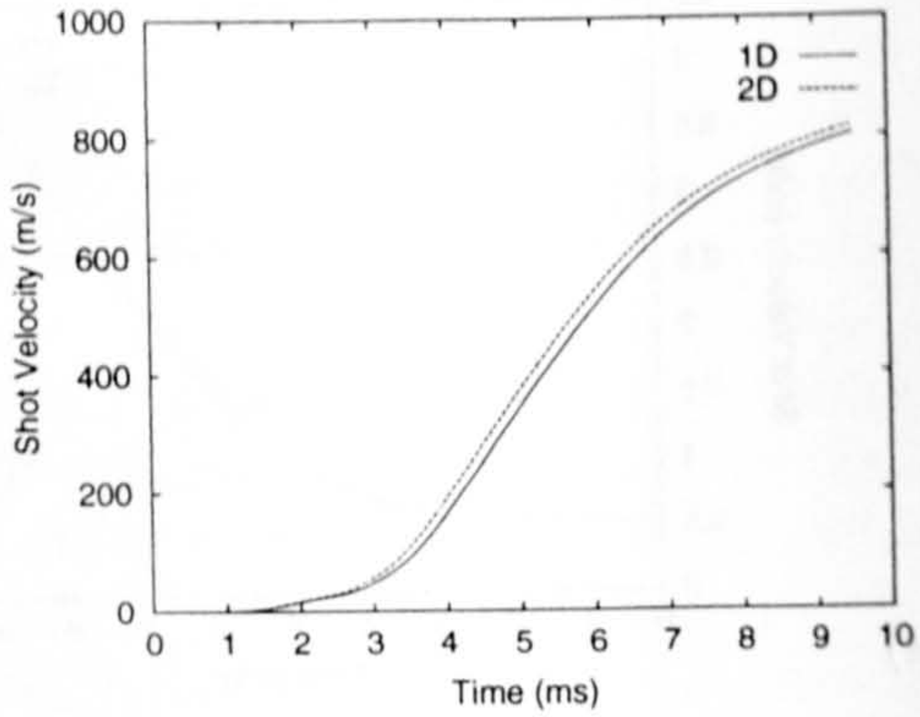


(g)

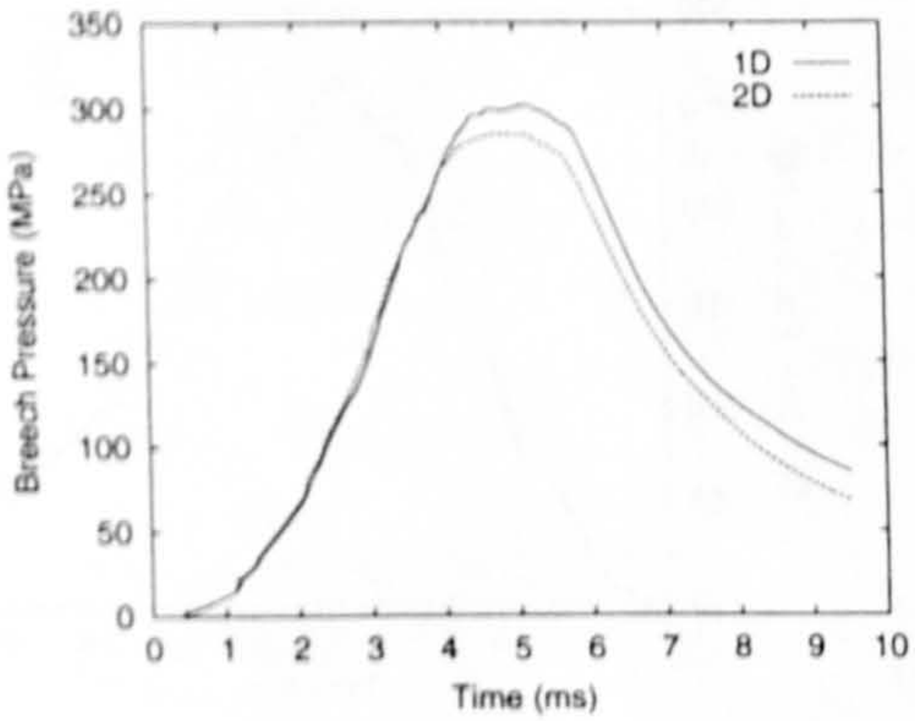
Figure 7.20: A comparison of experimental, 1D (formulation A) and 2D CAMR (formulation C) solution time histories, for the NAVALA test problem.



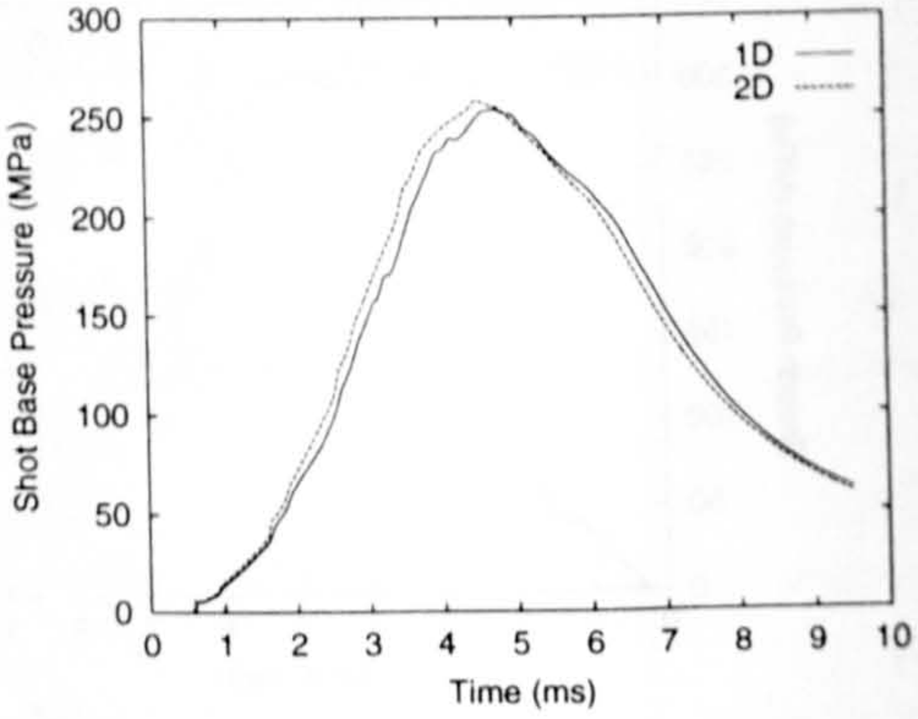
(a)



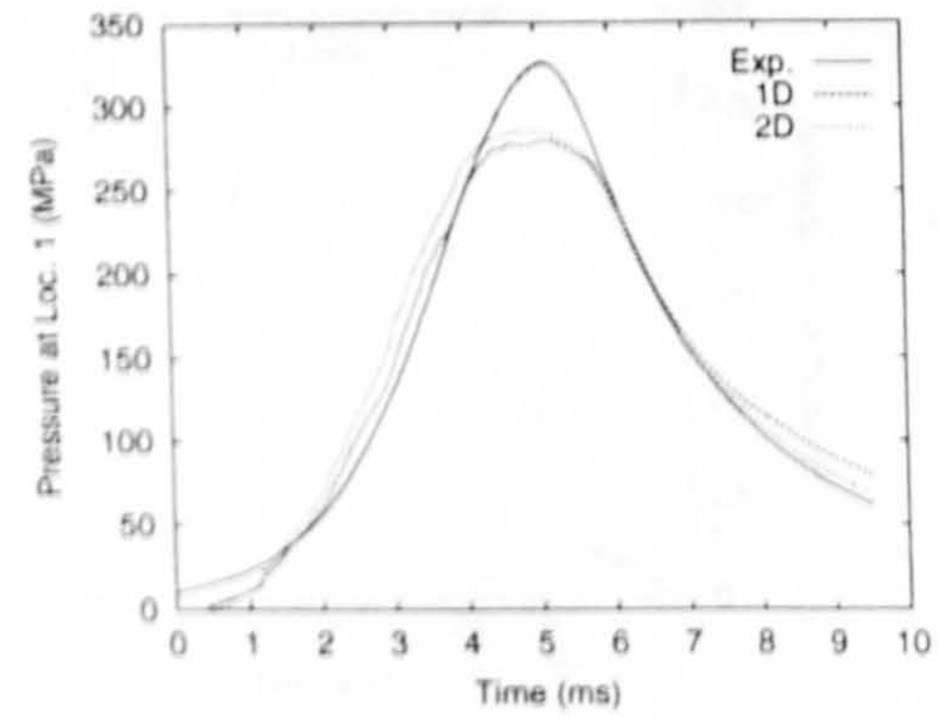
(b)



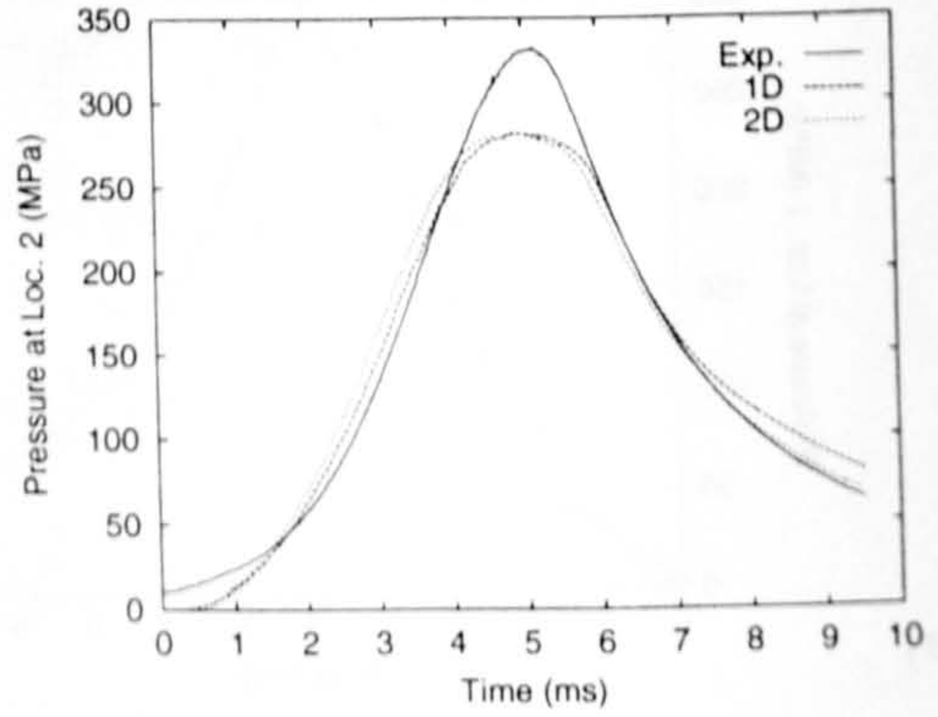
(c)



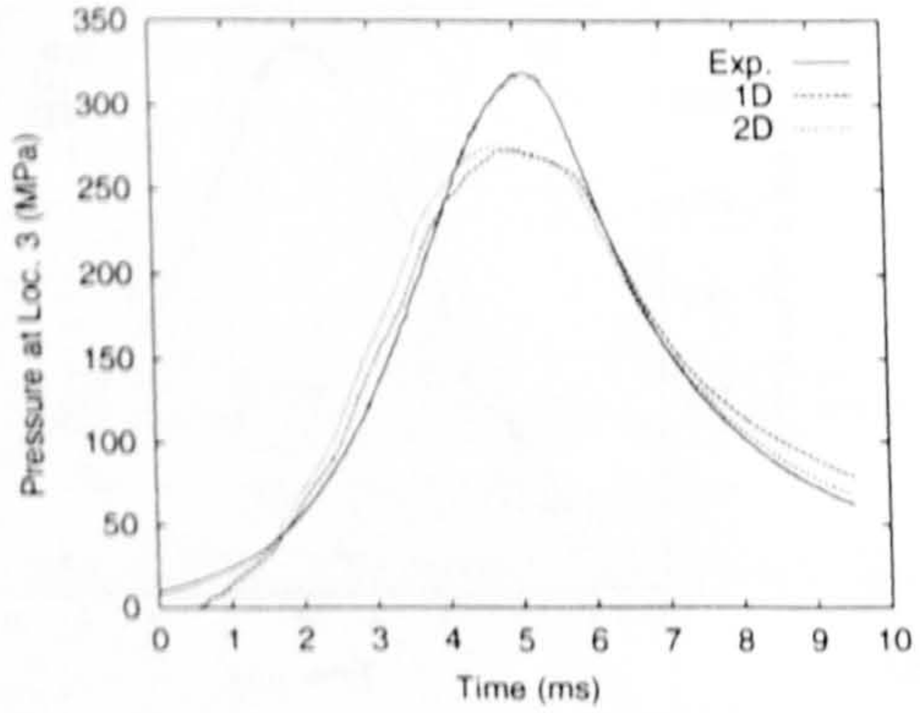
(d)



(e)



(f)



(g)

Figure 7.21: A comparison of experimental, 1D (formulation A) and 2D CAMR (formulation C) solution time histories, for the NAVALB test problem.

problems respectively. There is reasonable agreement between the one and two-dimensional solutions. The small differences are most likely caused by the formulation C approximation and/or the approximation of the multi-dimensional flow by a one-dimensional model. Differences between the peak pressures in the numerical solutions and the experimental results appear to be a greater for the NAVALB test compared to those for the NAVALA test. The inability of the axi-symmetric grid to represent the three-dimensional distribution of the propellant, may be the cause of the differences.

7.6 Summary of Internal Ballistics Results

Slight variations in the results produced by different solid phase formulations have been noted. Formulation C appears to be the most practical, because it produces time history solutions that are apparently free from spurious oscillations. Both formulations A and D produce spurious oscillations, which become more apparent as the grid resolution is increased. At this stage in the research, it is not possible to state which formulation, A or D, is 'most' correct. However, one thing is certain, because formulation D utilises the Riemann problem solution based on formulation C, it is valid for vacuum–non-vacuum situations. It is possible that the formulation C Riemann problem solver is the cause of the differences between formulations A and D.

One dimensional results suggest that using a first order scheme to compute the solid phase solution, whilst using a second order scheme to compute the gas phase solution, provides accurate results without any unphysical oscillations. The solid phase solution for a different formulation, could be computed with a higher order scheme, e.g. a formulation that contains an added source of diffusion.

The level of precision with which the source code is compiled has been shown to affect[3] even the one-dimensional results. Results obtained using double precision appear to be reasonably accurate, but are not free from error.

Full time history results could not be obtained for the 2D NAVAL problems using CAMR with formulation D. It could be that the spurious oscillations produced with formulation D, are exacerbated by the extra terms in the axi-symmetric derivation.

[3] All the erroneous results were obtained using Silicon Graphics computers. Similar tests using other platforms have not been done.

Chapter 8

Conclusions and Future Work

The theory and practical intricacies of a combined Chimera and AMR grid algorithm for solving systems of hyperbolic PDE's has been presented. A number of original aspects, which improve the accuracy and efficiency of the AMR algorithm have been presented. The Chimera approach was designed specifically for the accurate and efficient implementation into the AMR algorithm. Results for some well established test problems have been used to validate the AMR and CAMR approaches. These tests have also been used to assess the efficiency of the algorithms, by comparing the CPU times with those of regular grid approaches. The main cause for concern with the CAMR approach is the accuracy of the interpolation between the grid types. Even so, the results for the Euler equations demonstrate that the approach is a viable alternative to unstructured and Cartesian cut cell adaptive grid strategies.

There have been some difficulties in applying the approach to the internal ballistics problems described in the previous chapters. The ill-posed nature of the model is a likely cause of at least some of the difficulties. Two new solid phase formulations have been presented that are valid for vacuum situations. Results obtained with these formulations have been compared with those obtained by other researchers using a different formulation. No attempt has been made to assess the accuracy of the various formulations. Although, the results suggest that fewer numerical difficulties arise with one of the formulations.

The numerical implementation of the internal ballistics model involves solving a separate system of hyperbolic PDE's for each phase and a system of ODE's for the source terms. The approach is justifiable because the highly transient nature of the physical problem, dictates that in order to maintain time accuracy, the convection processes need to be computed with an explicit scheme. However, it is not certain whether or not the source terms can be accurately solved with an explicit scheme, that assumes the time step is given by the stability condition for the two hyperbolic systems. The author believes, with very little justification in the way of results, that

at least one of the source terms that arise from the axi-symmetric derivation,

$$\frac{-\alpha_1 v_1 (E_1 + p)}{r}$$

could cause some numerical difficulties. Even though v_1 may be small, r can also be very small (resolution dependent) and E_1 and p are generally very large. Thus, not only does this create the classical numerical problem of obtaining the product of two largely differing numbers, but may also yield a very large value source term, which could make the ODE step unstable. A fully three-dimensional derivation will not involve extra radial terms and should therefore be less problematic numerically.

There is no reason why the CAMR approach could not be extended to three dimensions. However, in three dimensions, the representation of boundary surfaces is difficult to generalise. This is a difficulty that is not specific to the Chimera approach. Ideally, the code would be able to utilise surface geometry data from a computer aided design (CAD) package.

Besides the inability of the two-dimensional model to represent three-dimensional phenomena, there are other physical aspects, such as chemistry of the ignition process, interphase drag, heat exchange, etc., that have been neglected from the internal ballistics model. A lot of ongoing research is being done into ways of improving the internal ballistics model and its numerical implementation. Many of the phenomena, such as heat loss to the solid boundaries, play a minimal role in the problems considered here and could easily be incorporated into the model. However, the role of chemistry in the ignition processes is far more significant. Lowe and Clarke are currently investigating ways of improving the ignition model. The next step in this work is to extend the model so that multiple propellants can be modelled.

There are a number of changes that could be made in order to improve the overall performance of the AMR and CAMR approaches. Presently, the interpolation between the boundary-fitted and Cartesian grids is only first order accurate. Interpolation between overlapping grids has been extensively studied. Improving the accuracy of the interpolation, while ensuring conservation should be included in future development work.

The AMR code alone (without Chimera), is very efficient and very robust. The author now considers that the CAMR code was developed with too much emphasis placed on its efficiency and not enough on its robustness. Problems occur because of the lack of generality for situations where a boundary-fitted grid, associated with one boundary, coincides with other boundaries. Presently, the interaction between the shot base and chamber grids in the internal ballistics problems, is dealt with specifically. It is possible to generalise the code to take account of interaction between two or more boundary-fitted grids, but it will adversely affect the efficiency and possibly the accuracy of the code.

The efficiency of the CAMR code could be significantly increased by adapting it for parallel processing computers. The necessary changes are usually made within the code. The mesh patch grid structure should facilitate the efficient parallelisation of the code, for either shared or distributed memory machines. The code can be rewritten using High Performance Fortran (HPF), which is a superset of Fortran 77 and 90. Utilising some features of Fortran 90, such as dynamic memory allocation and pointers, would also improve the efficiency of the code.

Directional Mesh Refinement (DMR) [122] is a recent extension of AMR, in which the refinement factor in each direction varies according to the requirements of the flow field. In [122], DMR reduced the amount of memory storage by up to a factor of five. However, the reduction in the CPU time was insignificant in comparison. An efficient implementation of DMR should yield similar savings in memory storage and CPU time. The implementation of DMR would require a significant amount of editing of the AMR and CAMR codes. The proper parental grid structure, described in section 3.1, may well aid the efficiency of the code when DMR is implemented.

Generally, solutions that are computed on grids that are aligned differently are not consistent with one another. Therefore, for general domains, the Chimera approach will produce slight differences in the solutions of the boundary-fitted and underlying grids. These differences are exacerbated if a spatially split solver strategy is employed. Moreover, interpolating the solutions between the grids would further increase the differences. One of the benefits of the CAMR algorithm, is that it is completely independent of the flow solver. Thus, any regular grid scheme, be it split or un-split, could easily be incorporated into the code. An un-split finite volume scheme, such as the one in [15], is likely to be included in the code during future development.

Bibliography

[1] J.D Anderson. *Modern Compressible Flow.* Mc Graw-Hill, 1990.

[2] E. H. Atta and J. Vadyak. A Grid Overlapping Scheme for Flowfield Computations About Multicomponent Configurations. *AIAA Journal*, 21(9):1271–1277, 1983.

[3] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws. *SIAM J. Sci. Stat.*, 15(1):127–138, 1994.

[4] M. Ben-Artzi and J. Falcovitz. A Second Order Godunov-Type Scheme for Compressible Fluid Dynamics. *J. Comput. Phys.*, 55:1–32, 1985.

[5] G. Ben-Dor. *Shock Wave Reflection Phenomena.* Springer-Verlag, 1992.

[6] J. A. Benek, J. L. Steger, and F. C. Dougherty. A Flexible Grid Embedding Technique with Application to the Euler Equations. *AIAA Paper* 83-1944, 1983.

[7] M.J. Berger. *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations.* PhD thesis, Dept. of Computer Science, Stanford University, California, 1982.

[8] M.J. Berger. On Conservation at Grid Interfaces. *SIAM J. Numer. Anal.*, 24(5):967–984, 1987.

[9] M.J. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *J. Comput. Phys*, 82:64–84, 1989.

[10] M.J. Berger and R.J. LeVeque. An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries. AIAA Paper 89-1930-CP, 1989.

[11] M.J. Berger and R.J. LeVeque. Stable Boundary Conditions for Cartesian Grid Calculations. Technical Report 90-37, ICASE, 1990.

[12] M.J. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *J. Comput. Phys.*, 53:484–512, 1984.

[13] S.J. Billett. *A Class of Upwind Methods for Conservation Laws.* PhD thesis, College of Aeronautics, Cranfield University, 1994.

[14] S.J. Billett and E.F. Toro. Numerical Methods for Overlapping Grids and Moving Boundaries. In *Sixth International Symposium on Computational Fluid Dynamics*, volume I, pages 111–116, Lake Tahoe, Nevada, U.S.A, September 1995.

[15] S.J. Billett and E.F. Toro. On WAF-Type Schemes for Multidimensional Hyperbolic Conservation Laws. *J. Comput. Phys.*, 130:1–24, 1997.

[16] S.J. Billett and E.F. Toro. Development of an Overlapping Grid Scheme for a One Dimensional Internal Ballistic Problem. Technical report, DRA, March 1995.

[17] J.P. Boris and D.L. Book. Flux-Corrected Transport. I. SHASTA, A Fluid Transport Algorithm That Works. *J. Comput. Phys*, 11:38–69, 1973.

[18] A.P. Burton. Adaptive mesh refinement with a godunov-type method applied to the one dimensional magnetogasdynamics equations. M.Sc Thesis, College of Aeronautics, Cranfield Institute of Technology, U.K., 1992.

[19] G. Chesshire and W.D. Henshaw. Composite Overlapping Meshes for the Solution of Partial Differential Equations. *J. Comput. Phys.*, 90:1–64, 1990.

[20] G. Chesshire and W.D. Henshaw. A Scheme for Conservative Interpolation on Overlapping Grids. *SIAM J. Sci. Stat*, 15(4):819–845, 1994.

[21] Y.L. Chiang, B. van Leer, and K.G. Powell. Simulation of Unsteady Inviscid Flow on an Adaptively Refined Cartesian Grid. In *AIAA 30th Aerospace Sciences Meeting*, Reno, NV, 1992.

[22] D.K. Clarke, H.A. Hassan, and M.D. Salas. Euler Calculations for Multielement Airfoils using Cartesian Grids. AIAA Paper 85-0291, 1985.

[23] J.F. Clarke, S. Karni, J.J. Quirk, L.G. Simmons, P.L. Roe, and E.F. Toro. Numerical Computation of Two-Dimensional, Unsteady Detonation Waves in High Energy Solids. *J. Comput. Phys.*, 106:215–233, 1993.

[24] P. Colella. A Direct Eulerian MUSCL Scheme for Gas Dynamics. *SIAM J. Sci. Stat. Comput*, 6:104–117, 1985.

[25] P. Colella. Multidimensional Upwind Methods for Hyperbolic Conservation Laws. *J. Comput. Phys*, 87:171–200, 1990.

[26] P. Colella and P.R. Woodward. The Piecewise Parabolic Method (PPM) method for Gas Dynamical Simulation. *J. Comput. Phys.*, 54:174–201, 1984.

[27] J. Corner. *Theory of the Interior Ballistics of Guns.* Wiley, 1950.

[28] R. Courant, E. Isaacson, and M. Rees. On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences. *Comm. Pure. Appl. Math.*, pages 243–255, 1952.

[29] D. De Zeeuw and K.G. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. In *AIAA 10th Computational Fluid Dynamics Conference*, 1991.

[30] D. De Zeeuw and K.G. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *J. Comput. Phys.*, 104:56–68, 1993.

[31] B. Einfeldt. On Godunov-Type Methods for the Euler Equations with General Equation of State. In *2nd International Conference on Hyperbolic Problems*, Aachen, Germany, 1988.

[32] P.R. Eiseman and G. Erlebacher. Grid Generation for the Solution of Partial Differential Equations. Technical Report 87-57, ICASE, 1987.

[33] J. Fischer. Self-adaptive mesh refinement for the computation of steady, compressible visous flows. *Aeronautical Journal*, pages 357–367, 1993.

[34] A.D. Fitt. Some aspects of Internal Ballistics Theory. In *Fifth Anglo-German Ballistics Meeting*, Unterluss, Germany, June 1988.

[35] A.D. Fitt. Contrasting Numerical Methods for Two Dimensional Two-phase Internal Ballistics Test Problems. In *11th International Symposium on Ballistics*, Brussels, Belgium, May 1989.

[36] A.D. French. *Solution of the Euler Equations on Cartesian Grids.* PhD thesis, College of Aeronautics, Cranfield Institute of Technology, 1991.

[37] C.W. Gear. *Numerical Initial-Value Problems in Ordinary Differential Equations.* Prentice-Hall, 1971.

[38] A.J. George. Computer Implementation of the Finite Element Method. Technical Report Stan-CS-71-208, Dept. of Computer Science, Stanford University, 1971.

[39] I.I. Glass. Some Aspects of Shock-Wave Research. *AIAA Journal*, 25(2):214–229, 1986.

[40] H.M. Glaz, P. Colella, I.I. Glass, and R.L. Deschambault. A Detailed Numerical, Graphical and Experimental Study of Oblique Shock Wave Reflections. Technical Report 285, Institute for Aerospace Science, University of Toronto (UTIAS), 1986.

[41] S.K. Godunov. A Difference Method for the Numerical Calculation of Discontinuous Solutions of Hydrodynamic Equations. *Mat. Sb.*, 47:271–306, 1959.

[42] S.K. Godunov. A Finite Difference Method for the Computation of Discontinuous Solutions of the Equations of Fluid Dynamics. *Mat. Sb.*, 47:357–393, 1959.

[43] J.B. Goodman and R.J LeVeque. On the Accuracy of Stable Schemes for 2D Scalar Conservation Laws. *Math. Comp.*, 45(21):15–21, 1985.

[44] J.J Gottlieb and C.P.T. Groth. Assessment of Riemann Solvers for Unsteady One-Dimensional Inviscid Flows of Perfect Gases. *J. Comput. Phys.*, 78:437–458, 1988.

[45] P.S. Gough and F.J. Zwarts. Modelling Hetrogeneous two-phase flow. *AIAA Journal*, 17(1):17–25, 1979.

[46] A. Harten. High Resolution Schemes using Flux Limiters for Hyperbolic Conservation Laws. *SIAM J. Numer. Anal*, 21(5):995–1011, 1984.

[47] A. Harten. On a Class of High Resolution Total Variation Stable Finite Difference Schemes. *SIAM J. Numer. Anal*, 21(1):1–23, 1984.

[48] A. Harten. Preliminary results on the Extension of ENO schemes to Two Dimensional Problems. In *International Conference on Non-Linear Hyperbolic Problems*, pages 23–51. Springer-Verlang, Berlin, 1987.

[49] A. Harten. ENO Schemes with Subcell Resolution. *J. Comput. Phys.*, 83:148–184, 1989.

[50] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly High Order Accuracy Essentially Non-oscillatory Schemes III. *J. Comput. Phys*, 71:231–303, 1987.

[51] A. Harten, P.D. Lax, and B van Leer. On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws. *SIAM Review*, 25(1):35–61, 1983.

[52] A. Harten and S. Osher. Uniformly High-Order Accurate Nonoscillatory Schemes I. *SIAM J. Numer. Anal*, 24(2):279–309, 1987.

[53] A. Harten and G. Zwas. Self-adjusting Hybrid Schemes for Shock Computations. *J. Comput. Phys*, 9:568, 1972.

[54] R. Hillier. Numerical Modelling of Shock Wave Diffraction. In *19th Internal Symposiumm on Shock Waves*, pages 17–26. Springer, 1993.

[55] C. Hirsch. *Numerical Computation of Internal and External Flows.* Wiley, 1988.

[56] D.C. Ives. A Modern Look at Conformal Mapping, Including Doubly Connected Regions. Technical Report Paper 75-842, AIAA, 1975.

[57] M.J. Ivings, D.M. Causon, and Toro E.F. On Hybrid High Resolution Upwind Methods for Multicomponent Flows. *ZAMM Math. Mech.*, 77 (5):1–24, 1997.

[58] A. Jameson. Transonic Aerofoil Calculations Using The Euler Equations. In *Numerical Methods in Aeronautical Fluid Dynamics.* Academic Press, New York, 1982.

[59] H. Kleine, E. Ritzerfeld, and H. Grönig. Shock Wave Diffraction - New Aspects of an Old Problem. In *19th Internal Symposiumm on Shock Waves*, pages 118–122. Springer, 1993.

[60] J.D. Lambert. *Computational Methods in Oridinary Differential Equations.* John Wiley and Sons, 1973.

[61] P. Lax and P. Wendroff. Systems of Conservation Laws. *Comm. Pure Appl. Math.*, pages 217–237, 1960.

[62] R.J. LeVeque. High Resolution Finite Volume Methods on Arbitrary Grids via Wave Propagation. *J. Comput. Phys*, 78:36-63, 1988.

[63] R.J. LeVeque. *Numerical Methods for Conservation Laws.* Birkhauser Verlag, 1992.

[64] R.J. LeVeque and H.C. Yee. A Study of Numerical Methods for Hyperbolic Conservation Laws with Stiff Source Terms. *J. Comput. Phys*, 86:187–210, 1990.

[65] R. Löhner and K. Morgan. Improved Adaptive Refinement Strategies for Finite Element Aerodynamic Computations. Technical Report 86-499, AIAA, 1986.

[66] R. Löhner, K. Morgan, J. Peraire, and M Vahdati. Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier Stokes Equations. *Int. J. Numer. Meth. Fluids*, 7:1093-1109, 1987.

[67] E.H. Love and F.B. Pidduck. Lagrange's Ballistic Problem. *Phil. Trans. Roy. Soc. Lond.*, 222:167–228, 1922.

[68] C. Lowe. *CFD Modelling of Solid Propellant Ignition.* PhD thesis, College of Aeronautics, Cranfield University, 1997.

[69] C. Lowe. Private communication, 1997.

[70] K. Nakahashi and S. Obayashi. FDM-FEM Zonal Approach for Viscous Flow Computations Over Multiple Bodies. In *25th Aerospaces Sciences Meeting*, Reno, 1987.

[71] N. Nikiforakis. Private communication.

[72] M. Olim and J.M. Dewey. Least Energy as a Criterion for Transition between Regular and Mach Reflection. *Shock Waves*, 1:243–249, 1991.

[73] S. Osher. Riemann Solvers, the Entropy Condition, and Difference Approximations. *SIAM J. Numer. Anal*, 21(2):217–235, 1984.

[74] E. Pärt-Enander and B. Sjögreen. Conservative and Non-Conservative Interpolation Between Overlapping Grids for Finite Volume Solutions of Hyperbolic Problems. *Computers Fluids*, 23(3):551–574, 1994.

[75] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield, and M.L. Welcome. An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions. *J. Comput. Phys*, 120:278–304, 1995.

[76] J. Peraire. *A Finite Element Method for Convection Dominated Flows.* PhD thesis, University of Wales, 1986.

[77] J. Pike. Internal Ballistics using Two Different Propellant Form Functions. Technical report, Cranfield Institute of Technology, 1990.

[78] J. Pike. Riemann Solvers for Perfect and Near-Perfect Gases. *AIAA Journal*, 31(10):1801–1808, 1993.

[79] J.J. Quirk. An Adaptive Grid Algorithm for Computational Shock Hydrodynamics. Ph.D Thesis, College of Aeronautics, Cranfield Institute of Technology, U.K, 1991.

[80] J.J. Quirk. A Cartesian Grid Approach with Hierarchical Refinement for Compressible Flows. In *Computational Fluid Dynamics 94. Proceedings of the Second European Computational Fluid Dynamics Conference*, pages 200–209. John Wiley & Sons, 1994.

[81] J.J. Quirk. An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two Dimensional Bodies. *Computers and Fluids*, 23(1):125–142, 1994.

[82] J.J. Quirk and U.R. Hanebutte. A Parallel Adaptive Mesh Refinement Algorithm. Technical Report ICASE 93-63, NASA Langley Research Center, Hampton, Virginia, USA, 1993.

[83] R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial Value Problems*. Interscience-Wiley, New York, 1967.

[84] A. Rizzi and L.E. Eriksson. Computation of Flow Around Wings based on the Euler Equations. *J. Fluid Mech.*, 148:45–71, 1984.

[85] P.L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *J. Comput. Phys*, 43:357–372, 1981.

[86] P.L. Roe. The Use of the Riemann Problem in Finite Difference Schemes. In *Proceedings of the Seventh International Conference on Numerical Methods in Fluid Dynamics*, pages 354–359, 1981.

[87] P.L. Roe. Some Contributions to the Modelling of Discontinuous Flows. In *Proceedings of the SIAM/AMS Seminar*, San Diego, 1983.

[88] P.L. Roe. Characteristic Based Schemes for the Euler Equations. *Ann. Rev. Fluid Mech.*, 18:337–365, 1986.

[89] J. Saltzman. An Unsplit 3D Upwind Method for Hyperbolic Conservation Laws. *J. Comput. Phys.*, 115:153–168, 1994.

[90] W.C. Skamarock and J.B. Klemp. Adaptive Grid Refinement for Two-Dimensional and Three-Dimensional Nonhydrostatic Atomospheric Flow. *Mon. Wea. Rev.*, 121:788–804, 192.

[91] B. W. Skews. The Perturbed Region Behind a Diffracting Shock Wave. *J. Fluid Mech.*, 29(4):705–719, 1967.

[92] R.E. Smith. Algebraic Grid Generation. In *Numerical Grid Generation*. North-Holland, 1982.

[93] G.A. Sod. A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. *J. Comput. Phys.*, 27:1–31, 1978.

[94] W. Speares. A Finite Volume Approach to the Weighted Average Flux Method. MSc thesis, College of Aeronautics, Cranfield Institute of Technology, U.K., 1991.

[95] W. Speares and E.F. Toro. WAFBC2 Ballistic Code Manual. Technical report, Cranfield Institute of Technology, U.K., April 1992.

[96] W. Speares and E.F. Toro. Adaptive Code User Manual. Technical report, Cranfield Institute of Technology, U.K., July 1993.

[97] W. Speares and E.F. Toro. An Adaptive Gridding Approach to the Computation of Reactive Two-phase Flows in Two Dimensions. In *The 19th International Symposium on Shock Waves*, Marseille, France, 1993.

[98] J.L Steger and R.F. Warming. Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite Difference Methods. *J. Comput. Phys*, 40:263–293, 1981.

[99] H.B. Stewart and B. Wendroff. Two-Phase flow: Models and Methods. *J. Comput. Phys.*, 56:363–409, 1984.

[100] G. Strang. On the Construction and Comparison of Difference Schemes. *SIAM J. Numer. Anal.*, 5(3):506–517, 1968.

[101] P.K. Sweby. High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM J. Numer. Anal.*, 21:995–1011, 1984.

[102] J. Szmelter, M.J. Marchant, A. Evans, and N.P. Weatherill. Two-Dimensional Navier Stokes Equations with Adaptivity on Structured Meshes. *Proceedings of the Second Workshop on Reliability and adaptive meshes in Computational Mechanics*, 1991. to be published in a separate issue of *Computer methods in Applied Mechanics and Engineering.*

[103] K. Takayama and O. Inoue. Shock Wave Diffraction Over a 90 Degree Sharp Corner. *Shock Waves*, 1:301–312, 1991.

[104] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation: Foundations and Applications.* North-Holland, 1985.

[105] E. F. Toro. Defects of Conservative Methods and Adaptive Primitive-Conservative Schemes for Computing Solutions to Hyperbolic Conservation Laws. Technical Report MMU–9401, Department of Mathematics and Physics, Manchester Metropolitan University, UK, 1994.

[106] E.F. Toro. Private communication.

[107] E.F. Toro. A New Numerical Technique for Quasi-Linear Hyperbolic Systems of Conservation Laws. Technical Report 8708, Cranfield CoA, 1986.

[108] E.F. Toro. A Fast Riemann Solver with Constant Covolume Applied to the Random Choice Method. *Int. J. Numer. Meth. Fluids*, 9:1145–1164, 1989.

[109] E.F. Toro. A Weighted Average Flux Method for Hyperbolic Conservation Laws. *Proc. Roy. Soc. Lond.*, A423:401–418, 1989.

[110] E.F. Toro. Riemann-Problem Based Techniques for Computing Reactive Two-Phase Flows. In Dervieux and Larrouturrou, editors, *Proc. Third. Intern. Confer. on Numerical Combustion*, number 351 in Lecture Notes in Physics, pages 472–481, Antibes, France, May 1989.

[111] E.F. Toro. WAFBC1 Internal Ballistic Code. Technical report, Cranfield Institute of Technology, U.K., 1989.

[112] E.F. Toro. A Linearized Riemann Solver for the Time-Dependent Euler Equations of Gas Dynamics. *Proc. Roy. Soc. Lond.*, A434:683–693, 1991.

[113] E.F. Toro. Riemann Problems and the WAF Method for Solving Two-Dimensional Shallow Water Equations. *Phil. Trans. Roy. Soc. Lond.*, A338:43–68, 1992.

[114] E.F. Toro. The Weighted Average Flux Method Applied to the Time-Dependent Euler Equations. *Phil. Trans. Roy. Soc. Lond.*, A341:499–530, 1992.

[115] E.F. Toro. Direct Riemann Solvers for the Time-dependent Euler Equations. *Shock waves*, 5:75–80, 1995.

[116] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer-Verlag, 1997.

[117] E.F. Toro, S.J. Billett, and E.P. Boden. Advanced Modelling Techniques for Propulsion Systems. In *Proceedings of Conference on Energetic Materials and Propulsion Technology*, Salsbury, Australia, April 1996.

[118] E.F. Toro and P.L. Roe. A Hybridised High-Order Random Choice Method for Quasi-Linear Hyperbolic Systems. In *Proc. 16th Intern. Symp. on Shock Tubes and Waves*, pages 701–708, Aachen, Germany, July 1987.

[119] E.F. Toro, M. Spruce, and W. Speares. Restoration of the Contact Surface in the HLL-Riemann solver. *Shock waves*, 4:25–34, 1994.

[120] J.A. Trangenstein. Adaptive Mesh Refinement for Wave Propagation in Nonlinear Solids. *SIAM J. Sci. Comput.*, 16:819-839, 1995.

[121] N. Uchiyama and O. Inoue. On the Performance of Adaptive Mesh Refinement Computation. *Shock Waves*, 2:117–120, 1992.

[122] U. Uphoff, C.H. Thill, and D. Hänel. Structure Mesh-refinement Techniques for Reactive and Multi-phase Flow. In *The Proceedings of the Third EC-COMAS Computational Fluid Dynamics Conference*, pages 287–292, Paris, France, September 1996.

[123] B. van Leer. Towards the Ultimate Conservative Difference Scheme I. The Quest for Monotonicity. *Lecture Notes in Physics*, 18:163–168, 1973.

[124] B. van Leer. Towards the Ultimate Conservative Difference Scheme II. Monotonicity and Conservation Combined in a Second Order Scheme. *J. Comput. Phys*, 14:361–370, 1974.

[125] B. van Leer. Towards the Ultimate Conservative Difference Scheme IV. A New Approach to Numerical Convection. *J. Comput. Phys*, 23:276–299, 1977.

[126] B. van Leer. Towards the Ultimate Conservative Difference Scheme V. A Second Order Sequel to Godunov's Method. *J. Comput. Phys*, 32:101–136, 1979.

[127] B. van Leer. Flux-Vector Splitting for the Euler Equations. Technical Report 82-30, ICASE, NASA Langley Research Center, 1982.

[128] B. van Leer. On the Relation between the Upwind-Differencing Schemes of Godunov, Enguist-Osher and Roe. *SIAM J. Sci. Stat. Comput.*, 5(1):1–20, 1984.

[129] B. van Leer. Progress in Multi-Dimensional Upwind Differencing. Technical Report CR-189708/ICASE 92-43, NASA, Sept. 1992.

[130] J. von Neumann and R.D. Richtmyer. A Method for the Numerical Calculation of Hydrodynamic Shocks. *J. Appl. Phys.*, 21:232–257, 1950.

[131] N.P. Weatherill. Numerical Grid Generation. Lecture series 1990-06, von Karman Institute for Fluid Dynamics, 1990.

[132] P. Woodward and P. Colella. The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks. *J. Comput. Phys*, 54:115–173, 1984.

[133] N.N. Yanenko. *The Method of Fractional Steps.* Springer Verlag, New York, 1971.

[134] H.C. Yee. A Class of High-Resolution Explicit and Implicit Shock-Capturing Methods. von Karmen Institute for Fluid Dynamics, Lecture Series 1989-04, 1989.

[135] S.T. Zalesak. Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids. *J. Comput. Phys.*, 31:335–362, 1979.