A THESIS

entitled

DYNAMIC GRID ADAPTION USING THE LPE EQUATION

by

Christopher Randle Bennett BEng. MSc.

submitted in partial fulfilment of the
requirements for the award of the
Degree of Doctor of Philosophy

School of Mathematics, Statistics and Scientific Computing,
Faculty of Technology,
University of Greenwich,
London.

MARCH 1999

PREFACE

The aim of the thesis was to develop a technique for dynamic grid adaption based on the LPE method of Catherall (1991) that can be used on 1D, 2D and 3D problems in a form that can be attached to computational fluid dynamics (CFD) packages with the minimum of additional coding. To this end the adaptive technique was implemented to run on real flow problems within the commercial CFD code PHOENICS (Spalding 1989), on a single problem with the research code PHYSICA (Cross et al 1995),and independently on a range of simple test functions.

Most, though not all of the successful work on structured grid adaption has been achieved working with inviscid codes that are tolerant of high degrees of grid distorsion, and involve problems with distinctive flow features such as very strong shocks which make the task of finding something to move the grid to much easier. PHOENICS is a viscous code that is not quite so tolerant of poor grid quality and some of the problems looked at dont have such clearly defined features. This makes the tasks addressed by the current work much harder.

A major requirement of this adaptive technique is that, as it is being developed for use with but independent of CFD packages, there should be the minimum of information transfer between and disruption to the flow solver. This is a great constriction on the development of the technique, as the information that can be reliably passed between the adaption tool and the solver consists of the grid coordinates and the values of solved variables. Grid cells cannot reliably be added, so adaption must take place by redistribution not refinement. Grid cell renumbering may not be possible, so boundary conditions must always apply to the same group of cells, regardless of how much the mesh distorts. In addition there is no allowance for any prior knowledge of the original problem domain. All of the features of a particular geometry can only be calculated from the shape and point distribution in the initial grid. Finally, the benefits of variations in grid density with the adapted grid must be able to outweigh the errors due to grid distortion when solved over by the CFD code.

The main aim, the development of the adaptive technique, has been achieved. However, some important features of the technique have not been fully tested. The most important of these is adaption in three dimensions. Testing in three dimensions was difficult because of the lack of suitable cases with reliable data, and the difficulty of comparing results from fixed and from adapted meshes.

Another feature that needs more work concerns weight functions. Many different forms were found in literature and several of them were implemented in the course of the development of the technique. Unfortunately there was not enough time to fully explore the differences, if any, between them.

Within the range of grid quality that can be tolerated by the solver the adaptive grid can be an effective tool in improving accuracy, given that there is some strong gradient or flow feature in the problem domain, and that the starting grid is neither too coarse to pick up the feature nor too fine to render grid redistribution unnecessary.

ABSTRACT

This thesis describes the development and implementation of a dynamic adaptive grid method for general two and three dimensional static and transient fluid flow problems solved over structured grids. The technique automatically manipulates the location of grid points within the domain of interest to concentrate cells in regions of high solution activity, thus aiming to improve the accuracy of the overall simulation for a given number of initial grid cells. To achieve this aim the Laplace Poisson Equidistribution equation is used. Furthermore, the work also covers different types and treatment of weight functions needed to represent areas of high solution activity and a range of techniques necessary to make the use of adaptive grids practical, including geometry modelling and grid quality control. The technique is implemented on simple functions and within the commercial CFD code PHOENICS, on fluid flow problems ranging from convection driven flows to shock capturing. The ability of the technique to be used for general grid manipulation is demonstrated by using it to couple PHOENICS with a stress code in the simulation of a deflecting beam in a uniform flow. In addition, a novel technique to adapt grids to solution phenomena using neural nets is demonstrated.

ACKNOWLEDGEMENTS

I would like to thank my main supervisor, Dr. Mayur Patel, and my second supervisor, Professor Koulis Pericleous, for their time, ideas, and encouragement. On the technical side of things other people who deserve thanks include Steve Rickman of BAe, Jun Zhao for neural networks stuff, Avril Sloane for the FET code used in the fluid structure interaction stuff and Nick Croft for being an all round clever so and so. There must be others as well, but this could go on too long.

The most important part of any research is the funding, and all the sources need listing. I was lucky enough to have an EPSRC CASE award to cover me for the first three years of the PhD. Sponsorship, ideas and test cases came from the aerodynamics department at British Aerospace Military Aircraft Division down at Farnborough. With the kind efforts of Steve Rickman of BAe and the accounts department of the University of Greenwich the last sponsorship cheque was only a year late, even with the University's habit of holding on to the money a little longer than I would have liked. In this my fourth year the University has funded me through a bursary and through teaching for most of the year, with my Father covering the holes in the funding in the middle.

Finally I would like to thank my family and friends for their support and encouragement, and especially Lorna for reminding me there is a life outside the computer labs and 'encouraging' me to get on and finish so that I can experience a little more of it.

Free time? What is that?

CONTENTS

APPENDIX C: PSEUDO CODE FOR ADAPTIVE MODULES

SYMBOL KEY

θ	..	..	Angle
$\lambda_L,\lambda_P,\lambda_E$		..	Weights on the Laplace, Poisson, and Equidistribution terms of the LPE equation respectively
ξ	..	..	Curvilinear coordinate
ϕ	..	..	Adaption Variable
a	..	..	Equation constant
$\bar{g}^{ij}$	..	..	Modified form of contravariant metric tensor
g^{ij}	..	..	Contravariant metric tensor
g	..	..	Jacobian
k	..	..	Solution curvature $k = \dfrac{\lvert\phi_{xx}\rvert}{\left(1+(\phi_x)^2\right)^{\frac{3}{2}}}$
P	..	..	Control function
p,q,r	..	..	general space coordinates
S	..	..	Source term
s	..	..	Arc length
u,v	..	..	Parametric values
WI, WJ, WK		..	Weight function values in X, Y, Z coordinate directions respectively
w	..	..	Weight function
Z	..	..	Parametric equation

Subscripts $_x$, $_\xi$ refer to partial differentiation with respect to x, ξ. Double subscripts, such as ϕ_{xx} refer to the second derivative of ϕ with respect to x.

CHAPTER 1: INTRODUCTION

1.1 Introduction to Grid Adaption and CFD

The aim of this chapter is to introduce the concept of grid adaption, to briefly explain the different techniques found in literature, and to describe the Laplace Poisson Equidistribution algorithm used in the current work.

Computational fluid dynamics (CFD) encompasses the simulation of physical phenomena through the use of mathematical equations. It exists because it allows physical behaviour to be determined in situations and geometries that could not otherwise be measured or explored analytically, and it allows rapid modifications to be made to geometrical designs in the virtual world of the computer that would take much longer if modifying 'real' models.

Many systems of mathematical equations used in computational fluid dynamics involve the solution of partial differential equations on discrete elements or cells that define the physical domain. These cells form a grid, and its shape and resolution has a profound effect upon the accuracy of the simulation. The finer the grid, the higher the resolution and accuracy, but the higher the computational cost in time for the computation of the solution and in memory used to store information for each element.

In addition the length scales of various phenomena may vary dramatically across the physical domain. If, for instance, a shock wave is being modelled then the required grid resolution to accurately fit the shock will be very high. If the grid is uniformly distributed, then either an extremely large number of cells will be required or the phenomena will be dissipated or lost. An extreme example is given by Quirk (1991) for the case of a detonation wave in solid explosives where for the required grid resolution within the reaction zone the cell thickness should be in the order of 0.002 mm in a sample which may measure 100 mm in length by 100 mm in diameter. If the location of such phenomena is known beforehand, then the generation of the computational grid can be modified to produce a higher resolution in the important

regions. If their positions are not known, their shapes complex, or in transient cases their position moves, the only way to accurately and efficiently model them is to use a grid that dynamically adapts to them.

The two techniques used for dynamic grid adaption are grid refinement (figure 1.1), which involves the addition and possible subtraction of grid nodes to the original grid, and redistribution (figure 1.2), where the positions of the original grid nodes are modified. Refinement provides the least cost in computational resources for a prescribed level of accuracy. Redistribution provides the best accuracy for a proscribed level of cost. The two techniques are not mutually exclusive. The current work is concerned with redistribution.

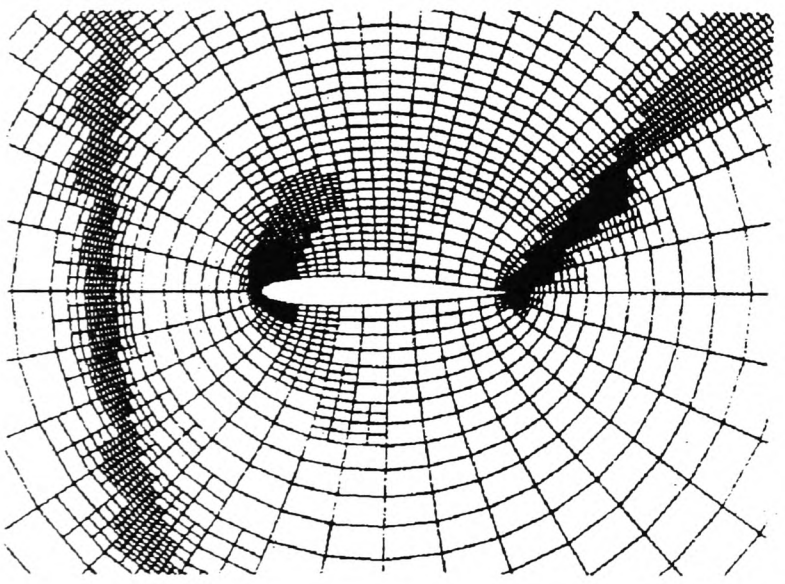


Figure 1.1 Refinement (Dannenhoffer 1991)

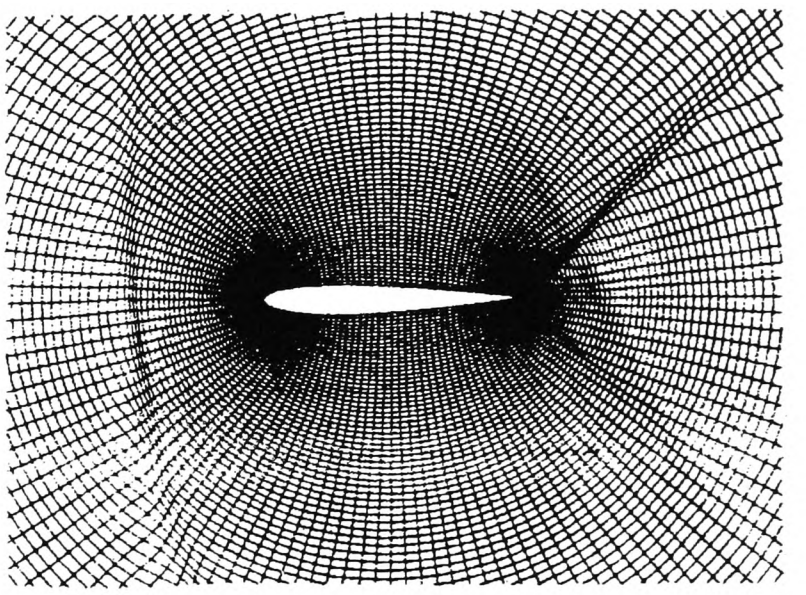


Figure 1.2 Redistribution (Dannenhoffer 1991)

Another approach to see why grid adaption can be useful is to consider truncation error due to the Taylor expansion which can be used to discretise partial differential equations. The Taylor expansion creates solvable algebraic expressions for the first and second order derivatives that consist of terms involving discrete points and a truncation error which is a function of the distance between the discrete points and the change in value over the points. The truncation error is least when either the distance between the points or the change in value is small. By clustering points where the rate of change is high and reducing the number of points where the rate of change is low a dynamically adapting grid can achieve a constant level of truncation error over the whole domain thus minimising the total error.

The type of grid covered in the current work is the structured, body fitted coordinate grid, or BFC for short. The BFC grid is formed by the intersection of families of curvilinear lines that coincide with the shape of the physical domain at its boundaries (figure 1.3). It can be viewed as a regular, rectangular grid that has been pulled and stretched until it fits the dimensions of the physical domain. A few other examples of grid include cartesian, where the grid is formed by the intersection of three sets of mutually perpendicular parallel planes; polar coordinate, where the grid is formed by planes intersecting an axis, perpendicular to the axis, and in concentric cylindrical surfaces around the axis; and unstructured, where elements may in theory have any number of sides and the grid connectivity is controlled by the use of adjacency tables.

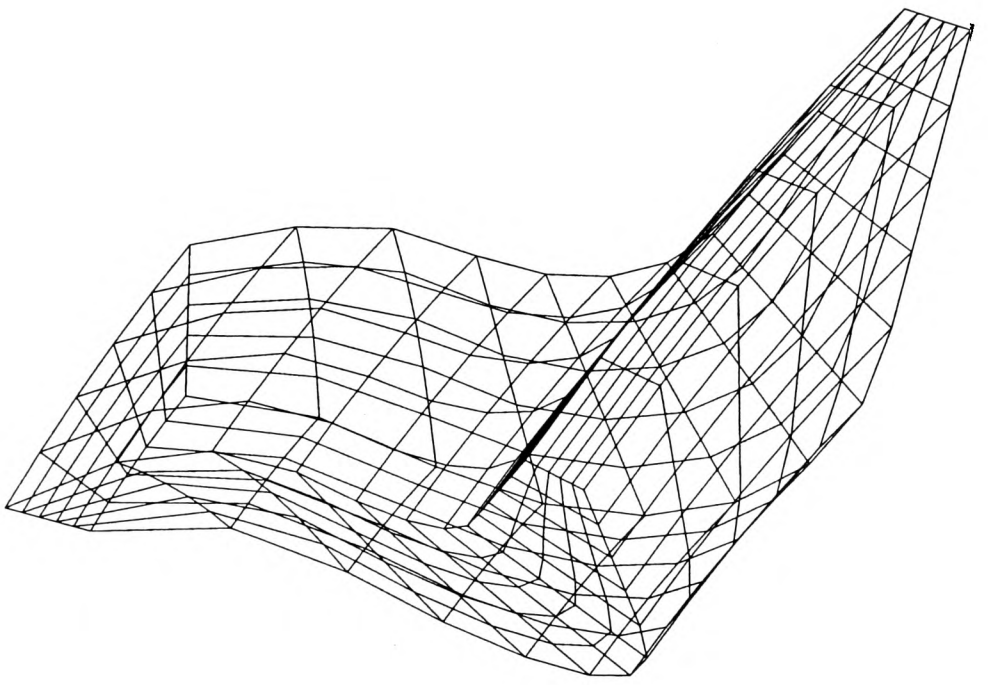


Figure 1.3 Example BFC grid

1.2 False Diffusion

False diffusion is a major source of error in CFD techniques. It results from the treatment of fluid flow between neighbouring cells in multidimensional grids as one dimensional in the basic discretisation of the conservation equations used to model the fluid. When the local grid is misaligned with the direction of fluid flow the contributions from convection from the neighbouring cells are of poor accuracy. The error can be seen as an artificial diffusion term. A much fuller description of false diffusion is given by Patankar (1980).

False diffusion has little effect when the grid is aligned with the fluid flow and when the ratio of convection to diffusion is low. Otherwise false diffusion can be reduced by using complex discretisation schemes on the conservation equations, such as SUCCA (Carey et al. 1993), or CUPID (Patel et al. 1988), or by using grid adaption. Grid adaption by redistribution can align the grid cells with the local flow direction. Grid adaption by both redistribution and refinement can also reduce the local grid cell size where false diffusion is worst.

1.3 Grid Quality

Grid quality is difficult to measure because it depends so much upon the problem and the capability of the CFD solver which is used to solve it. The basic requirements of a good structured grid are that it:

- Accurately defines the geometry of the problem, including internal and external boundaries.
- Allows a solution to be generated to the required level of accuracy for the minimum use of resources. This can involve variation in grid density.

The degree to which the grid can be skewed to accommodate boundaries and increased density in regions of high solution error is governed by the way in which grid properties contribute towards error. The two main grid properties that contribute towards grid error are:

- Orthogonality. An orthogonal system is defined by Thompson et al. (1985) as one where a vector normal to a coordinate surface is parallel to the tangent of a coordinate line that crosses that surface. These two vectors are known as the contravariant and covariant base vectors respectively and are important in the derivation of the metric tensors used to transform grid coordinates from physical to curvilinear space (see appendix A). Structured grid orthogonality is based on the arrangement of grid nodes.

- Smoothness, defined as the rate of change of the angles between grid cells and the rate of change of grid cell volume.

The general requirements for grid quality given by Jacquotte (1991) are orthogonality the grid lines close to the boundaries, angles between lines not exceeding 45°, deviation of lines from one cell to the next of not more than 10 to 15°, and a rate of change of volume not greater than 30%.

Grid cell aspect ratio can contribute to grid quality problems, particularly when combined with poor smoothness and orthogonality.

Though there is much literature available on mesh optimisation, with or without grid adaption, (Lehtimäki 1995, Knupp 1992, Jacquotte 1992, 1991, Chawner and Anderson 1991, Lu and Eiseman 1991, Lee et al. 1990 and Thompson et al. 1985), research on the actual contributions of grid properties of solution error have been limited (Huang and Prosperetti 1994, Lee and Tsuei 1992b, Shirayama 1991). The most interesting of these papers is possibly Huang and Prosperetti (1994) who determine formulae for the truncation error due to the diffusion term in the Navier Stokes equations, and then use the formula to test the influence of grid angle, smoothness and boundary orthogonality. Their main conclusions are that grid angle is only significant when very small, the importance of grid smoothness is related to the shape of solution, with very regular solutions needing smoother grids, and that orthogonality is less important than grid density. In other words orthogonality can be disadvantageous when it reduces the local grid concentration.

1.3.1 Measuring Grid Quality

Grid skewness is very similar to grid orthogonality and can be measured relatively easily using the technique described in Lehtimäki (1995). Here skewness is defined as the amount of deformation of the individual grid cells. In two dimensions it is calculated for each grid cell by using the areas of both pairs of triangles which are formed by splitting it along either of its two diagonals.

If the areas of the triangles are t_{11} and t_{12} for the diagonal from *i,j* to *i+1,j+1*; t_{21} and t_{22} for the diagonal from *i+1,j* to *i,j+1*; and the maximum lengths of the edges of the cell in each curvilinear direction are e_1 and e_2 (figure 1.4) then the skewness ρ can be calculated using equation 1.1.

$$\rho = 2\frac{\max\{\min\{t_{11}, t_{12}\}, \min\{t_{21}, t_{22}\}\}}{e_1 e_2} \qquad \textbf{(1.1)}$$

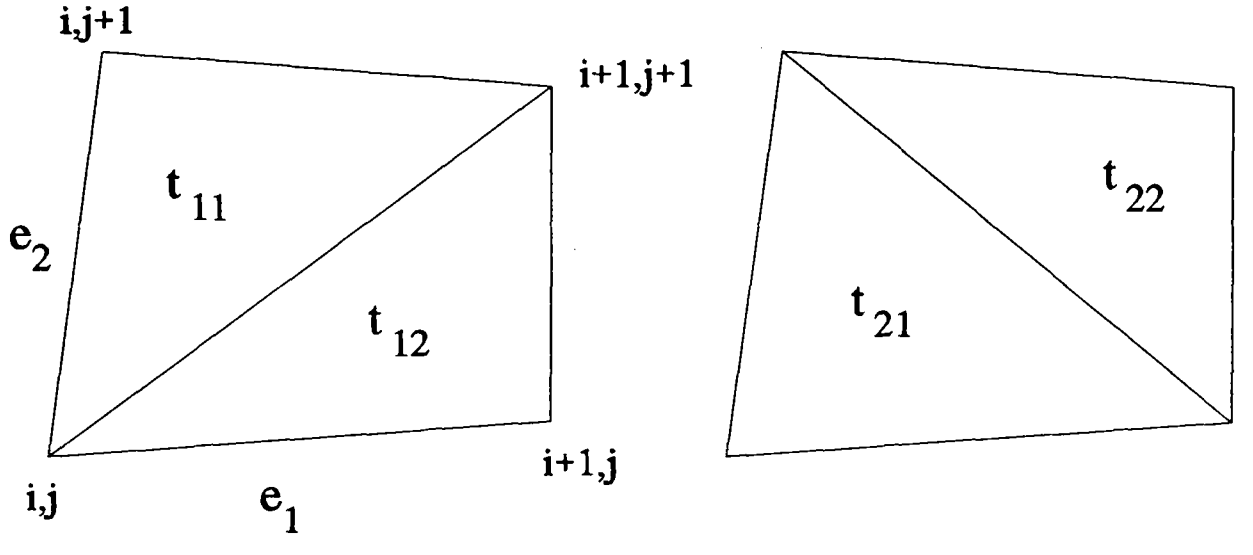


Figure 1.4 Definition of Terms in Equation 1.1

A value of 1 for the skewness ρ indicates a quadrilateral cell. As the cell becomes more skewed ρ decreases. If the cell becomes folded then ρ becomes negative.

In the system described in figure 1.5 the orthogonality *o* of the point $r_{i,j}$ can be determined using the formula given in Lehtimäki (1995):-

$$o = \frac{1}{4}\left[(\delta r_{i+1,j} \cdot \delta r_{i,j+1})^2 + (\delta r_{i,j-1} \cdot \delta r_{i+1,j})^2 + (\delta r_{i-1,j} \cdot \delta r_{i,j-1})^2 + (\delta r_{i,j+1} \cdot \delta r_{i-1,j})^2\right] \qquad \textbf{(1.2)}$$

Where

$$\delta r_{i+1,j} = \frac{r_{i+1,j} - r_{i,j}}{\left|r_{i+1,j} - r_{i,j}\right|} \qquad \textbf{(1.3)}$$

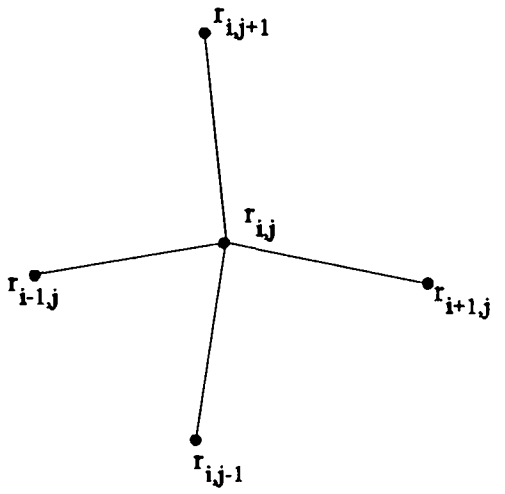


Figure 1.5 Grid Orthogonality

1.4 Techniques

This section presents examples of the range of adaption techniques found in the literature. Features which are common to all adaption techniques are briefly discussed first.

All adaption techniques consist of a method to determine where in the physical domain extra computational resources are required. The value calculated by this method is commonly known as the weight. The weight is usually some function that relates to solution error which may be some measure of the truncation error using a second coarse mesh, or more simply based on the gradients of chosen solution variables. The weight may also be linked to geometrical properties of the grid to assist in grid generation. The principal aim of adaption techniques is to evenly distribute the weight over the domain so as to evenly distribute and minimise the overall solution error by avoiding large peaks and troughs.

A fuller description of methods used to calculate the weight is presented in chapter 2.

In the case of refinement more grid cells are added in regions where the weight exceeds some predefined tolerance. In redistribution techniques the grid cell spacing is, depending on the technique used, inversely proportional to the weight distribution. This is driven by the equidistribution equation, which in one dimension is

$$w\Delta x = constant \tag{1.4}$$

Where w is the weight, and Δx is the grid point spacing.

A second feature common to all adaption techniques is that the final grid itself will be able to provide information on the shape of the solution through the distribution of its cells. An example of this can be seen in figures 1.1 and 1.2 above where the calculated position of pressure waves around the aerofoil can be clearly seen in the grid.

1.4.1 Mesh Redistribution Methods

Mesh redistribution methods involve the movement of nodes in, or the recalculation of the original grid with the same number of cells, so as to give a better grid distribution for the problem.

The main problem in redistribution techniques is balancing the demands of equidistribution of error against grid quality. Errors due to grid quality will increase and overtake the error modelled by the weight function if the grid is stretched too far.

Grid quality can be measured in terms of orthogonality or skewness, which can be looked on as a function of the internal angles of the cell, and smoothness, which is the rate of change of cell size. Maintaining grid quality also means avoiding grid line cross over and cells which are too small or large. There is some published work concerned with optimising grid quality alone. Examples include Lehtimäki (1995), Huang and Prosperetti (1994), Lu and Eiseman (1991), Saoub and Vandromme (1991), and Shirayama (1991).

The main advantages of grid redistribution techniques include

- Continuous response to solution behaviour.

• Fixed storage requirements, though allowances may need to be made to store old grids.

• Easier to implement, particularly for three dimensional grids. The flow solver may not need to have any knowledge of adaptivity.

• Alignment of grid with the direction of local flow and physical phenomena such as shock waves can help to reduce numerical diffusion

Disadvantages include

• Loss of grid quality. Computational effort and coding complexity may have to be tied up with monitoring quality.

• Maintenance of internal and external grid boundaries and grid features. This is more straightforward where the grid is regenerated using known geometrical information about the physical domain, but less so when grid points are moved only.

• Disruption to flow solver. If adaption takes place dynamically, that is during the course of a single run of the flow solver, then the stored values for the solved variables in each cell become invalid once the grid has moved. Dannenhoffer (1991) assumes that as the movement in each adaption will be small that the solution carried forward to the new grid will be a good guess to the new solution. The alternative is to interpolate the solution between grids, which may be expensive if the grid is large and particularly for three dimensional grids.

1.4.1.1 Equidistribution

The equidistribution equation on its own is the simplest method of grid adaption, as well as being the basis for grid movement in other techniques.

There are a range of methods based on the equidistribution equation. The method used by Patel, Pericleous and Baldwin (1995) involves equidistribution along lines. Each grid line is treated separately, and cell distribution is based on the arc length. Equation (1.4) is rewritten as

$$\Delta s = \frac{c}{w} \tag{1.5}$$

where Δs is the arc length. This is modified to allow for zero values of weight giving

$$\Delta s_n = \left(\frac{c}{w + c} \right) \Delta s_o \tag{1.6}$$

where Δs_n is the new arc length, Δs_o, the original arc length and c, some predefined function that is fixed for each line. After all of the new arc lengths are calculated they are scaled to fit within the limits of the original grid line. Additional terms may be added to control the amount of movement in each adaption.

Equidistribution on its own may be very fast, but additional steps, such as Laplace smoothing, may be needed in the adaption algorithm to enhance grid quality. Such steps may also help to link the movement between neighbouring grid lines that may otherwise lead to highly skewed grids.

Examples of equidistribution techniques in literature include Adams and Conlisk (1995), Patel et al. (1995), Lin and Wu (1993), Lee and Tsuei (1992a), (1992b), Shyy (1992), (1991a), (1991b), (1990), (1986), Harvey et al. (1991), Chang and Shyy (1991), Chao and Liu (1991), Pao and Abdol-Hamid (1991), Bockelie et al. (1990), Lawal (1990), Mattheij and Smooke (1989), Seibert et al. (1989), Anderson (1987a),

Eiseman (1987), Eiseman and Bockelie (1987), Coyle et al. (1986), Sanz-Serna and Christie (1986), Dwyer (1985) (1984), Rai and Anderson (1982).

1.4.1.2 Poisson Equation

A common method for structured grid generation involves the solution of the Poisson equation, where the Laplace equation common in elliptic grid generators is equated to some control function. The basic form of the equation is

$$\nabla^2 \xi^i = Q_i \quad i = 1,3 \tag{1.7}$$

where Q_i is a control function and ξ_i are the system of coordinates that define curvilinear space. This equation is commonly inverted to make the physical coordinates the dependent variables, and thus takes the form

$$\sum_i \sum_j g^{ij} r_{\xi^i \xi^j} + \sum_k P_k g^{kk} r_{\xi^k} = 0 \quad i,j,k \ cyclic \tag{1.8}$$

where g^{ij} is the contravariant metric tensor, r is the general space coordinate, and P_k is the transformed control function. The value of the control function is used to control the shape of the grid and the concentration of grid cells. A control function of zero means that the equation defaults to the Laplace equation and the resulting grid is equi-spaced. A full description of the use of control functions can be found in Thompson et al. (1985).

This method can be easily modified to allow for adaption by modifying the control function term to include an adaptive term that is related to solution activity. By including and weighting both the term to generate the original grid and the term to adapt the grid the amount of adaption can be controlled. If the weighting on the terms is allowed to vary during the run, then the original grid can be recovered by reducing the weight on the adaptive term (Roache et al. 1991). The adaptive form of the Poisson equation can be written as

$$\sum_i \sum_j g^{ij} r_{\xi^i \xi^j} + \sum_k \left[(1 - c_w) P_k + c_w W_k \right] g^{kk} r_{\xi^k} = 0 \qquad \textbf{(1.9)}$$

where c_w controls the amount of adaption, and W_k is some adaptive weight function.

The Poisson equation is widely used for grid adaption on both two and three dimensional grids (Hall and Zingg 1995, Kwon and Jeong 1995, Shen et al. 1993, Catherall 1991, Dannenhoffer 1991, Hsu and Lee 1991, Anderson 1990, 1987a, 1987b, Tu and Thompson 1990, Noack and Anderson 1989, Kim and Thompson 1988, Matsuno and Dwyer 1988, Thompson 1985, Holcomb 1984).

1.4.1.4 Spring Analogy

This method is in more common usage for finite element problems, and in particular mixed fluid and structure interactive problems where the mesh may deform about an elastic structure (Farhat and Lin 1990, Lesoinne and Farhat 1995). In such problems the stress solver required to modify the grid may already be available. The grid can be looked at as a pseudo structural problem with each node connected to its neighbours by a spring. For grid adaption a force is applied to each node in the mesh relating to the value of some function. For moving mesh type problems a range of nodes are given an initial displacement and the system allowed to reach a new equilibrium. The problem can also be looked at as the minimisation of the energy in the system of springs as depicted in figure 1.6.

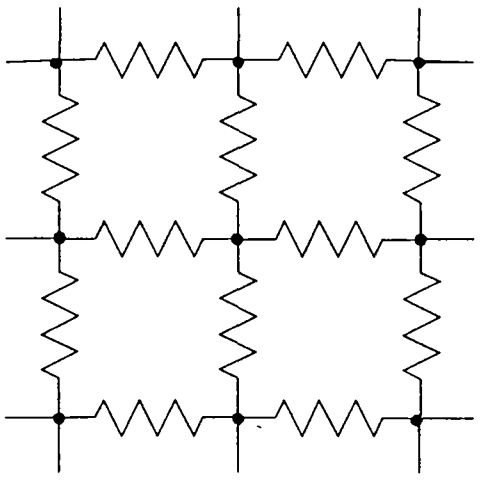


Figure 1.6 Tension spring analogy

To prevent excessive grid skewness a torsion spring concept (figure 1.7) may be used about each node (Catherall 1991, Harvey et al. 1991, Nakahashi and Diewert 1987, 1986, 1985).

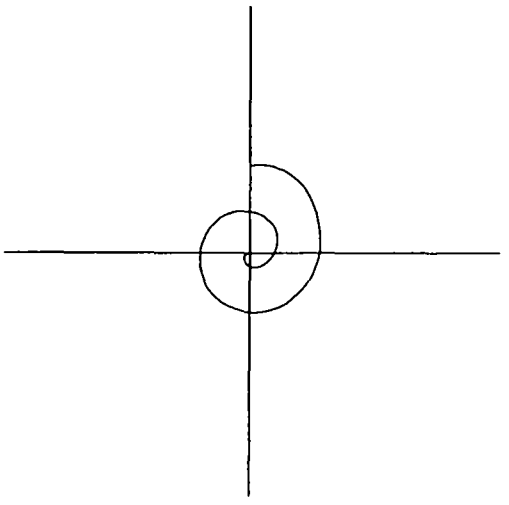


Figure 1.7 Torsion Spring Analogy

An advantage of this approach is that as it only depends upon a system of nodes and links the same technique can be used for both structured and unstructured grids (Farhat and Lin 1990).

Other examples of the use of the spring analogy in literature include Hu (1998) at the University of Greenwich, Ramakrishnan and Singh (1994), Harvey et al. (1993), Davies and Venkatapathy (1992), Palmerio (1992), Niederdrenk (1991), Gnoffo (1983). It is also being used in ongoing work at the University of Greenwich in coupling structural and CFD codes (Slone 1997).

1.4.1.4 Variational Technique

The variational technique involves the evaluation and minimisation of grid properties and solution error, commonly using Euler-Lagrange equations from the calculus of variations. The properties commonly checked for include cell volume or area, smoothness and orthogonality. The final solved for equation is a weighted combination of the above properties.

The integral form of the three terms for orthogonality, smoothness, and solution behaviour used by Kim (1987) and others are:-

Smoothness

$$I_S = \int\int\int \sum_{i=1}^{3} \left(\nabla \xi^i \cdot \nabla \xi^i\right) dx \tag{1.10}$$

Orthogonality

$$I_O = \int\int\int \sum_{i=1}^{3} \left(\nabla \xi^j \cdot \nabla \xi^k\right)^2 g^{\frac{2}{3}} dx \tag{1.11}$$

Solution behaviour

$$I_W = \int\int\int w^2(x)\sqrt{g}\, dx \tag{1.12}$$

The main advantage of variational techniques is that the individual terms reflect physically significant quantities that are easy for the user to understand. However the final equations may be more costly to evaluate whilst not producing significantly different results from, say, Poisson based schemes (Kim 1987).

Other examples of the use of variational techniques are included in Brackbill (1993), Castillo (1991), Desbois (1991), Hsu and Tu (1987), Jacquotte and Coussement (1992), Palmerio (1992), Saouab and Vandromme (1991), Singh, Kumar and Tiwari (1991), Jeng and Liou (1989), Giannakopoulos and Engel (1988), Kim and Thompson (1988), and Brackbill and Saltzman (1982).

1.4.1.5 Mapping Techniques

Mapping techniques involve the adaption of the grid in some form of transformed or parametric space which is then mapped back to the physical domain. The parametric space may be much simpler than the physical domain and may even be uniform. This allows the use of simpler adaption equations, that will in turn probably behave better than on a skewed mesh. In addition the grid shape and grid quality can be closely controlled as the grid points are mapped back to the physical domain.

The main strength of mapping techniques is in handling complex geometry, where it is much easier to maintain high grid quality and prevent grid cross over.

The weaknesses of these techniques include the time taken for transfer of data from parametric space to real space, and the need to know a lot of information about the initial grid. It may be necessary to completely regenerate the original grid, so the technique may only be practical when a new grid can be generated rapidly, as in the case of Allen (1995) where transfinite interpolation is used to generate the grid.

Examples of the use of monitor surface include Allen (1995), Hagmeijer (1994), Brackbill (1993), Benson and McRae (1991), Pao and Abdol-Hamid (1991), Bockelie et al. (1990), Arina (1989), (1988), Eiseman (1987), (1985a), (1985b), and Eiseman and Bockelie (1987).

1.4.1.6 Others

Greenburg (1983) used a linear unimolecular chemical kinetic analogy to determine grid point movement.

Another technique is presented by Petzold (1987) who uses a one dimensional moving grid scheme to minimise the time rate of change when modelling moving fronts.

1.4.2 Grid Refinement

In grid refinement extra cells are used to resolve the solution in flagged, commonly rectangular, regions within the original grid. Regions are flagged where some sort of solution error indicator exceeds a defined tolerance level. The solution error indicator may be very similar in form to those used for weights in mesh redistribution.

For structured grids there are two principal techniques, grid embedding or grid division.

In grid division (figure 1.8) the grid is held in a complex data structure such as a quad tree or oct tree for three dimensions. Flagged cells spawn sub cells in a level below them in the grid hierarchy that are formed by subdividing the region covered by the original cell. Starting from an initial grid that is already refined it may be possible to coarsen regions where solution activity is small. A common restriction on this kind of grid is that neighbouring cells can only be one level different. The grid may be solved in a partly unstructured way.

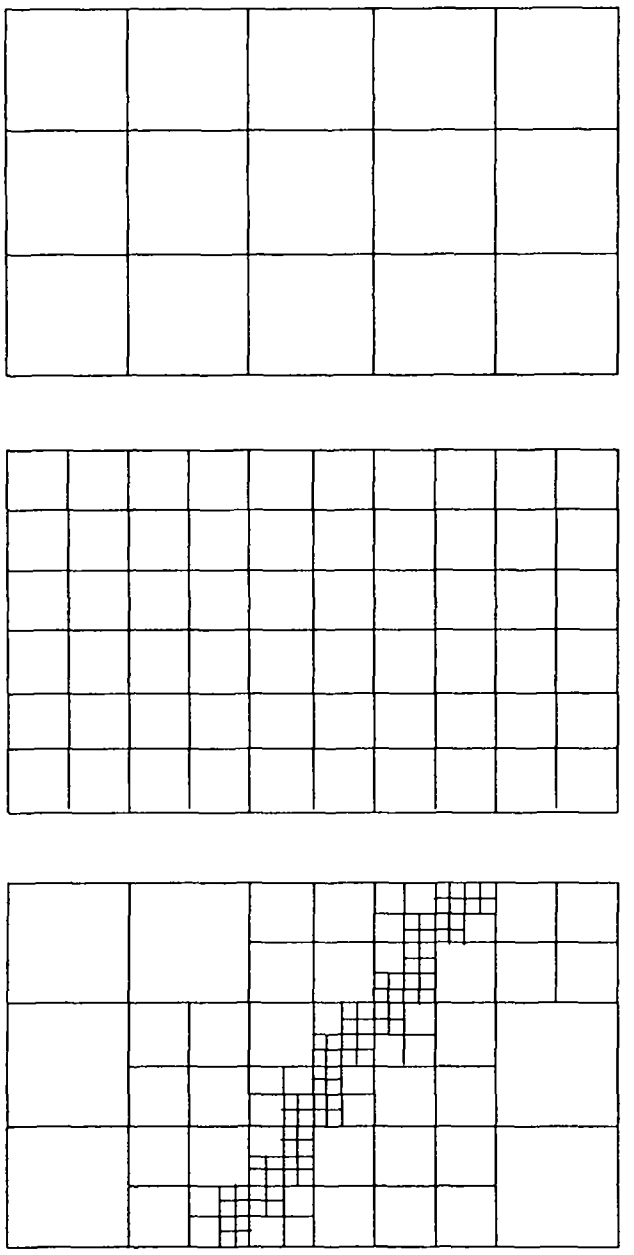


Figure 1.8 Top level, initial, and adapted grids

The two main problems with this kind of adaption are due to the restrictive data structure involved, which must be built into the underlying flow solver from an early stage, and due to the problem of dealing with discretisation and conservation at hanging nodes.

Examples of grid division techniques are included in Davies and Dannenhoffer (1994), De Zeeuw (1992), Dannenhoffer (1991), (1987), (1985), Carey et al. (1988), and Carey (1987).

Grid embedding (figure 1.9) involves the use of a new separate grid to cover the flagged region. The embedded grid is solved as a separate problem, with boundary conditions coming from the original grid, possibly in the form of single layer of surrounding cells known as halo cells. The solution computed on the embedded grid is either stored in its own right or transferred back to the initial grid. The initial grid may have embedded regions either be blocked or blanked out, or solve for those regions using data interpolated back. The overall data structure of the grid may be invisible to the underlying flow solver (Quirk 1991) allowing the choice of solver to be much more flexible than would be the case using grid division. At the same time the problems of hanging nodes are greatly reduced as no grid solved for contains them. They only contribute to the definition of the boundary conditions, the calculation of which is invisible to the underlying solver. There is also more flexibility over the type and number of cells in the refined region, allowing very high resolution to be achieved instantly to model rapidly changing length scales in the solution, if required.

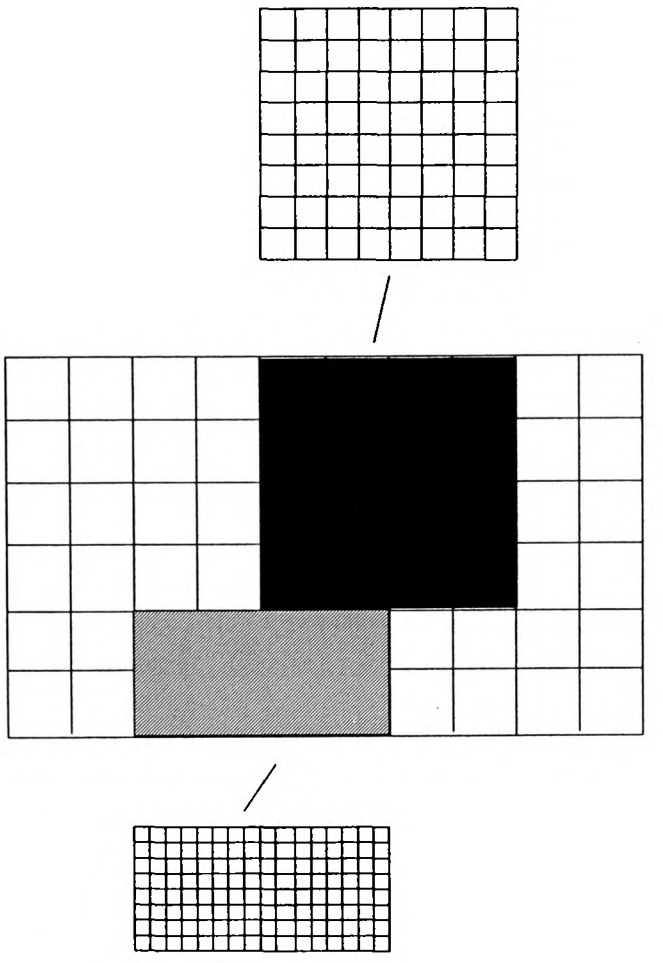


Figure 1.9 Initial and embedded grids

Grid embedding is close to multigrid and multiblock methods. The separation of the solution procedure for the different embedded regions may allow the process to be easily parallelised. Though the data structure may be much simpler than for grid division, data storage for the whole problem may be difficult, especially if the position of the embedded regions is to be modified.

Examples of grid embedding techniques are included in Pascau and Gaspar (1995), Wu (1995), Blom and Verwer (1994a), (1994b), (1994c), Quirk (1991), Thompson and Ferziger (1989), Berger (1986), and Berger and Colella (1986).

The main advantages of grid refinement techniques include

- Grid quality maintained. If only rectangular regions are refined the quality of the final grid will be no worse than that of the original. In techniques such as Local Uniform Grid Refinement (Blom and Verwer 1994a, 1994b, 1994c) the grid is always regular, and the underlying solver can take advantage of this.

• Grid geometry maintained. Subdivision of grid cells does not require knowledge of the geometry of the whole domain.

The main disadvantages of grid refinement include

• Data structures and storage. The storage requirement will be constantly varying, demanding some outside limit of space, or the use of some form of dynamic memory allocation.

• Visualisation of results. This is tied in with the implementation, but may be especially awkward if the solution is transient.

• Discrete reaction to phenomena. Cells are only subdivided or embedded at set threshold values of the error indicator. The subdivisions or embedded cells are of fixed sizes.

1.4.3 Hybrid Grids

Hybrid adaption techniques combine grid refinement and redistribution (figure 1.8). Hybrid methods potentially offer very fast and efficient adaption to solution phenomena, though at the same time need both the data structures required by refinement methods and the controls needed to maintain geometry and grid quality required for redistribution methods. Because both methods of adaption are available the actual amount of movement that occurs due to redistribution may only need to be small, so that the problems with excessively skewed grids may be avoided. When the grid is refined by halving the flagged grid cells a modest reduction in cell size through redistribution will reduce the number of refinement levels necessary to achieve the desired accuracy.

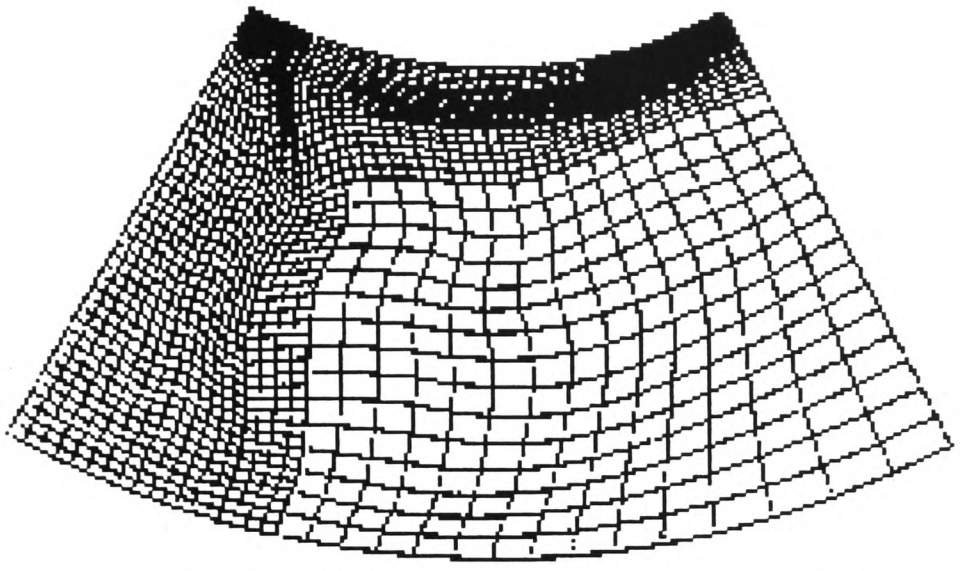


Figure 1.10 Hybrid grid adaption (Lee and Tsuei 1993)

The two choices are refine then redistribute, used by Acharya and Moukalled (1990) or redistribute then refine used by Lee and Yeh (1993). Both techniques have their advantages. The choice may depend on the type of grid refinement available.

Acharya and Moukalled (1990) use a modified grid embedding technique where the refined structured grid is generated using weighting based on solution behaviour. The unusual aspect of their work is that the refined region does not have to have a regular shape and as a result the embedded grid may be very skewed, particularly at its boundaries.

Examples of refinement followed by redistribution in literature can be found in Moukalled and Acharya (1991), and Acharya and Moukalled (1990).

Examples of redistribution followed by refinement in literature can be found in Arney and Flaherty (1990), Biswas et al. (1993), Lee and Yeh (1993), Lee and Tsuei (1993), and Szmelter et al. (1992). Seibert et al. (1989) adapt on the allowable grid then coarsen the resulting grid by removing grid points.

1.4.4 Other Methods

Mavriplis (1992) discusses adaption of spectral elements. Beyond redistribution and refinement the order of the equations for the cells in question may also be increased.

1.4.5 Unstructured Grids and Finite Elements

Though much is applicable to any adaptive grid method, the current work is solely concerned with structured grids, though it could be easily extended to rectangular elements, or 'pseudo' structured grids, so the coverage of adaption for unstructured grids is by necessity very light. Adaption for unstructured grids is arguably more common than for structured, and there are numerous papers available on unstructured grid adaption. No attempt has been made to produce an exhaustive review of unstructured grid adaption techniques.

Unstructured grid techniques usually have the advantage of fast grid generation, particularly for triangular elements. They may also have a more flexible number of grid nodes. These factors make the easiest method of adaption refinement by simply adding or redistributing grid nodes to important areas, then recalculating the grid. Transfer of data between grids may be more awkward. What has been classed as grid redistribution, where the grid nodes retain their connectivity between applications of the adaption algorithm, is much more rare, with the main examples being in finite element problems where the grid may be moving anyway.

A few examples of adaption in unstructured meshes include Franca and Haghighi (1994), Palmerio (1992), Sweby (1988), Carey (1987), and Oden et al. (1986).

1.5 The Laplace-Poisson-Equidistribution Equation

The Laplace-Poisson-Equidistribution, or LPE, equation is the adaptive method implemented in this thesis. It is based on the Poisson equation used for grid generation and adaption (see 1.4.1.2). As suggested by its name it includes three principal terms, a Laplace smoothing term, a Poisson term that promotes the original grid distribution and an Equidistribution term that drives the grid adaption. The terms are weighted by the user to control how much or how little adaption takes place. The LPE equation can be written in the form

$$\lambda_L(L) + \lambda_P(P) + \lambda_E(E) = 0 \tag{1.13}$$

Where λ_L, λ_P, λ_E are the weights on the equations, and *(L)*, *(P)*, *(E)* represent the Laplace, Poisson, and Equidistribution equations respectively. A full mathematical description of the terms is given in chapter 2.

The Laplace term is more commonly encountered in elliptical grid generators where it is used to produce an evenly spaced grid. No account is made of solution activity and when applied to a pre-generated grid it will produce the most orthogonal grid point distribution possible. The one characteristic which may not be so useful is a tendency for grid points to gather around convex and away from concave surfaces (Thompson 1987). However, this characteristic may be of advantage when developing grids over aerofoils, where the curve of the aerofoil surface will attract grid points to its leading edge, a potential area of high solution activity. The popularity of aerofoils as test cases in the open literature suggests that the negative side of the use of the terms in the Laplace equation may not commonly be encountered.

The Poisson term includes control functions that will recreate the original grid if the weights on the other two terms are set to zero. This term may be important if the original grid is skewed for any special reason, as, in the absence of a strong weight to drive the grid, the Laplace term will smooth it.

The Equidistribution term used encourages an even distribution of solution activity across all grid points, to the exclusion of all considerations of grid smoothness. The end result of applying the Equidistribution term alone is a potentially highly skewed and unusable grid. Coupled with the other terms it emphasises areas of high solution activity in the final grid.

The Laplace-Poisson-Equidistribution equation was first used by Catherall (1991) for two dimensional cases. The two dimensional form was implemented and the three dimensional form was developed by the author as part of an MSc thesis (Bennett 1992).

1.6 Requirements of the Adaptive Grid Algorithm

The main aim of the thesis is the development of an adaptive algorithm based on the LPE equation that is three dimensional, code independent and contains all of the tools needed to dynamically adapt structured grids. The LPE equation is used because it contains terms that maintain grid quality as well as adapt the grid to solution phenomena.

A particular feature of the technique described in this thesis is that it assumes that there is no prior knowledge of the grid or the physical domain that it encompasses. This is important in making the algorithm independent of other codes. This means that all geometrical information used to maintain the physical geometry during adaption must be generated within the algorithm, using the initial grid only.

Additional methods which are needed to make the adaptive algorithm work include tools for

• Maintenance of geometry by modelling lines and surfaces using cubic splines and bicubic patches respectively. The splines and patches need to be generated using the original grid and points need to be mapped back onto them using a location algorithm and interpolation to recover point parameters.

• Recovering control functions needed to recreate the original grid from it.

• Interpolating data between previous and new adapted grids. The principal method used in the current work is Shepard's interpolation (Shen et al. 1993).

• Determining weight functions. A range of methods for calculating weights which reflect different aspects of solution behaviour have been implemented. In addition techniques have been developed to smooth and modify the weight function to improve the adapted grid.

- Maintaining the integrity of grid cells by checking for grid overlaps.

- Displaying the behaviour of the adaption algorithm. A measure of movement in terms of a percentage of distortion of the original grid is used to show how much adaption has taken place. In addition properties relating to the adaption are stored and displayed using a CFD code post processor.

- Accessing solution variables and the grid within the underlying CFD code. This is code specific.

The algorithm is not restricted to the whole grid and can instead be used individually on a number of subdomains; this extends its applicabilty to block structured type meshes.

1.7 Structure of Thesis

The mathematics of the LPE equation are described in chapter 2.

Chapter 3 describes the additional tools needed to make adaption viable.

Chapter 4 describes general issues which affect the implementation of adaption outside the algorithms described in chapters 2 and 3. Examples of its implementation are also described within the commercial fluid dynamics code PHOENICS (Spalding 1989) and the research code PHYSICA (Cross et al 1995).

Chapters 5 and 6 are used to present results produced using adaption within PHOENICS and in an independent implementation in two and three dimensions respectively.

Chapter 7 describes how the current grid adaption method can be used to interface between a fluids code and a stress code to allow the modelling of fluid structure interaction.

Chapter 8 describes a first attempt on an alternative method of grid adaption using neural nets.

Chapter 9 contains possible future work and conclusions.

1.8 Closure

This chapter has introduced CFD, grid quality, the concept of grid adaption and the range of techniques used in literature to achieve it. In addition the Laplace-Poisson-Equidistribution equation has been introduced and the range of techniques needed to make it work listed.

In summary grid adaption is concerned with manipulating the distribution and shape of the cells that define the solution domain to achieve the best possible grid for the simulation.

CHAPTER 2: MAIN ALGORITHM

2.1 Introduction

The purpose of this chapter is to show the derivation and discretisation of the full LPE equation. The weight term used to determine grid movement is also discussed.

Only the discretisation of two dimensional form of the LPE equation will be presented in full in this chapter, though the full three dimensional partial differential equations will be listed. The discretisation of the one and three dimensional forms is given in appendix A.

2.2 Computational Molecule

The differential equations that make up the LPE algorithm are solved by using finite difference approximation.

The basic finite difference approximations, using central difference formulae are

$$\frac{\partial r_{i,j}}{\partial \xi_1} = \frac{r_{i+1,j} - r_{i-1,j}}{2} \tag{2.1}$$

$$\frac{\partial^2 r_{i,j}}{\partial \xi_1^{\,2}} = \left(r_{i+1,j} - 2r_{i,j} + r_{i-1,j}\right) \tag{2.2}$$

$$\frac{\partial^2 r_{i,j}}{\partial \xi_1 \partial \xi_2} = \frac{\left(r_{i+1,j+1} - r_{i-1,j+1}\right) - \left(r_{i+1,j-1} - r_{i-1,j-1}\right)}{4} \tag{2.3}$$

The numbering convention used for the following equations is shown in figure 2.1.

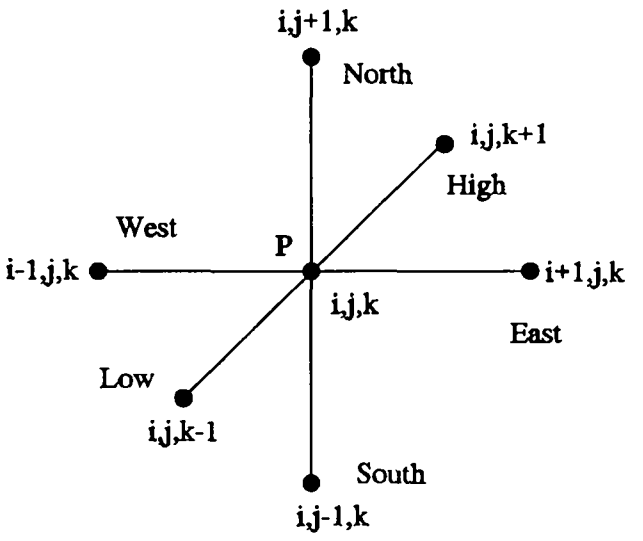


Figure 2.1 Computational Molecule

2.3 Metric Tensors

The transformation of the curvilinear grid into cartesian space is governed by the contravariant metric tensors.

A derivation of the metric tensors in three dimensions is given in Appendix A. Here, the three distinct metric tensors (noting $g^{12} = g^{21}$) are simply listed.

$$g^{11} = \frac{1}{g}(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2}) \tag{2.4}$$

$$g^{22} = \frac{1}{g}(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1}) \tag{2.5}$$

$$g^{12} = -\frac{1}{g}(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2}) \tag{2.6}$$

Where $\underline{r}_{\xi_i}$ is the gradient of the line ξ_i = constant, and g is the Jacobian. The Jacobian cancels out of the full equations

$$\underline{r}_{\xi_i} = \frac{\partial x}{\partial \xi_i}\underline{i} + \frac{\partial y}{\partial \xi_i}\underline{j} \tag{2.7}$$

The three dot products, $(\underline{r}_{\xi i}\,\underline{r}_{\xi j})$, are

$$\begin{aligned}\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_1} &= \left(\underline{i}\frac{\partial x}{\partial \xi_1} + \underline{j}\frac{\partial y}{\partial \xi_1}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_1} + \underline{j}\frac{\partial y}{\partial \xi_1}\right)\\ &= \left(\frac{\partial x}{\partial \xi_1}\right)^2 + \left(\frac{\partial y}{\partial \xi_1}\right)^2\\ &= \frac{1}{4}\left((x_{i+1,j} - x_{i-1,j})^2 + (y_{i+1,j} - y_{i-1,j})^2\right)\end{aligned} \tag{2.8}$$

$$\begin{aligned}\underline{r}_{\xi_2}\cdot\underline{r}_{\xi_2} &= \left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2}\right)\\ &= \left(\frac{\partial x}{\partial \xi_2}\right)^2 + \left(\frac{\partial y}{\partial \xi_2}\right)^2\\ &= \frac{1}{4}\left((x_{i,j+1} - x_{i,j-1})^2 + (y_{i,j+1} - y_{i,j-1})^2\right)\end{aligned} \tag{2.9}$$

$$\begin{aligned}\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_2} &= \left(\underline{i}\frac{\partial x}{\partial \xi_1} + \underline{j}\frac{\partial y}{\partial \xi_1}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2}\right)\\ &= \frac{\partial x}{\partial \xi_1}\frac{\partial x}{\partial \xi_2} + \frac{\partial y}{\partial \xi_1}\frac{\partial y}{\partial \xi_2}\\ &= \frac{1}{4}\left((x_{i+1,j} - x_{i-1,j})(x_{i,j+1} - x_{i,j-1}) + (y_{i+1,j} - y_{i-1,j})(y_{i,j+1} - y_{i,j-1})\right)\end{aligned} \tag{2.10}$$

Using the terminology *E,W* for *i+1,i-1*, *N,S* for *j+1,j-1*, gives

$$\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_1} = \frac{1}{4}\left((x_E - x_W)^2 + (y_E - y_W)^2\right) \tag{2.11}$$

$$\underline{r}_{\xi_2}\cdot\underline{r}_{\xi_2} = \frac{1}{4}\left((x_N - x_S)^2 + (y_N - y_S)^2\right) \tag{2.12}$$

$$\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2} = \frac{1}{4}\left((x_E - x_W)(x_N - x_S) + (y_E - y_W)(y_N - y_S)\right) \tag{2.13}$$

2.4 Derivation of Laplace Term

The Laplace term is used in the LPE equation to promote orthogonality in the adapted grid.

The Laplace system in three dimensions is

$$\nabla^2 \xi_i = 0 \qquad i = 1,3 \tag{2.14}$$

When modified to make the x, y and z co-ordinates the dependent variables It takes the following form

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{\xi_i \xi_j} = 0 \tag{2.15}$$

Where r represents the co-ordinate direction being solved for, either x, y or z.

In two dimensions this simply reduces to

$$g^{11} r_{\xi_1 \xi_1} + g^{22} r_{\xi_2 \xi_2} + 2 g^{12} r_{\xi_1 \xi_2} = 0 \tag{2.16}$$

Discretised, using equations 2.1, 2.2 and 2.3, this becomes

$$g^{11}(r_E - 2r_P + r_W) + g^{22}(r_N - 2r_P + r_S) + \frac{g^{12}}{2}\left[(r_{EN} - r_{WN}) - (r_{ES} - r_{WS})\right] = 0 \tag{2.17}$$

Separating the terms in equation 2.17 leads to the algebraic equation

$$a_P^L r_P = a_E^L r_E + a_W^L r_W + a_N^L r_N + a_S^L r_S + s^L \tag{2.18}$$

With the source term s^L formed by using the corner coefficients

$$s^L = a_{NW}^L r_{NW} + a_{NE}^L r_{NE} + a_{SW}^L r_{SW} + a_{SE}^L r_{SE} \tag{2.19}$$

where

$$a_P^L = a_E^L + a_W^L + a_N^L + a_S^L \tag{2.20}$$

$$a_E^L = a_W^L = g^{11} \tag{2.21}$$

$$a_N^L = a_S^L = g^{22} \tag{2.22}$$

$$a_{NE}^L = a_{SW}^L = \frac{g^{12}}{2} \tag{2.23}$$

$$a_{NW}^L = a_{SE}^L = \frac{-g^{12}}{2} \tag{2.24}$$

2.5 Derivation of Poisson Term

The Poisson term is used to promote the original grid distribution in the adaption.

The Poisson equation has been discussed in section 1.3.1.2. The three dimensional form shown in equation 1.4 is repeated here.

$$\nabla^2 \xi_i = P_i \qquad i = 1,3 \tag{2.25}$$

Where P_i represents a control function used to modify grid point locations.

The Poisson equation modified to make x, y and z co-ordinates the dependent variables takes the form

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{\xi_i \xi_j} + \sum_{k=1}^{3} g^{kk} P_k r_{\xi_k} = 0 \tag{2.26}$$

In two dimensions this reduces to

$$g^{11}\left(r_{\xi_1\xi_1}+P_1 r_{\xi_1}\right)+g^{22}\left(r_{\xi_2\xi_2}+P_2 r_{\xi_2}\right)+2g^{12}r_{\xi_1\xi_2}=0 \tag{2.27}$$

Discretised, equation 2.27 becomes

$$\begin{aligned} &g^{11}(r_E-2r_P+r_W)+g^{22}(r_N-2r_P+r_S)+\frac{g^{12}}{2}\left[(r_{EN}-r_{WN})-(r_{ES}-r_{WS})\right] \\ &+\frac{g^{11}}{2}(r_E-r_W)P_1+\frac{g^{22}}{2}(r_N-r_S)P_2=0 \end{aligned} \tag{2.28}$$

The full details of the transformation of the control function are not important in the current work as they are extracted directly from the original grid using equation 2.28. They are found by forming simultaneous equations using the initial grid values.

Separating the terms in equation 2.28 leads to the algebraic equation

$$a_P^P r_P=a_E^P r_E+a_W^P r_W+a_N^P r_N+a_S^P r_S+s^P \tag{2.29}$$

As the only differences to the Laplace equation are the control functions all terms that do not contain them are the same

$$a^P \equiv a^L \tag{2.30}$$

Only the source term is different

$$s^P=s^L+\frac{g^{11}}{2}P_1(r_E-r_W)+\frac{g^{22}}{2}P_2(r_N-r_S) \tag{2.31}$$

2.6 Derivation of Equidistribution Term

The equidistribution term is used to apply a weight to alter the grid distribution.

The equidistribution term is derived from the basic equidistribution equation (1.1) in each of the three co-ordinate directions, which can be written in the form

$$W_i \frac{\partial S_i}{\partial \xi_i} = constant \qquad i = 1,3 \tag{2.32}$$

Where W_i is the weight function and S_i is the arc length along the i varying line.

The three equations in the system are differentiated to remove the constant, multiplied by the term $g^{ii}\underline{r}_{\xi i}$ and combined to give

$$\sum_{i=1}^{3} g^{ii} r_{\xi_i} \left(\frac{W^i{}_{\xi_i}}{W^i} + \frac{x_{\xi_i} x_{\xi_i \xi_i} + y_{\xi_i} y_{\xi_i \xi_i} + z_{\xi_i} z_{\xi_i \xi_i}}{x_{\xi_i}{}^2 + y_{\xi_i}{}^2 + z_{\xi_i}{}^2} \right) = 0 \tag{2.33}$$

The multiplier puts the equidistribution term into the same form as the second term in the Poisson term as shown in equation 2.26.

In two dimensions this reduces to

$$g^{11} r_{\xi_1} \left(\frac{WI_{\xi_i}}{WI} + \frac{x_{\xi_1} x_{\xi_1 \xi_1} + y_{\xi_1} y_{\xi_1 \xi_1}}{x_{\xi_1}^2 + y_{\xi_1}^2} \right) + g^{22} r_{\xi_2} \left(\frac{WJ_{\xi_i}}{WJ} + \frac{x_{\xi_2} x_{\xi_2 \xi_2} + y_{\xi_2} y_{\xi_2 \xi_2}}{x_{\xi_2}^2 + y_{\xi_2}^2} \right) = 0 \tag{2.34}$$

Where WI and WJ are the weight functions in the x and y coordinate directions respectively.

The discretisation of the equidistribution equation can be simplified by first seeing that the denominator of the second term in the brackets is identical to vector dot products used to evaluate the metric tensors, which is calculated elsewhere and can be substituted in 2.34.

$$\underline{r}_{\xi_i} \cdot \underline{r}_{\xi_i} \equiv x_{\xi_i}^2 + y_{\xi_i}^2 \tag{2.35}$$

Equation 2.34 discretised gives

$$g^{11}\frac{(r_E-r_W)}{2}\left[\left(\frac{WI_E-WI_W}{2WI_P}\right)+\frac{1}{(\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_1})}\left(\frac{(x_E-x_W)}{2}(x_E-2x_P+x_W)+\frac{(y_E-y_W)}{2}(y_E-2y_P+y_W)\right)\right] \tag{2.36}$$

$$+g^{22}\frac{(r_N-r_S)}{2}\left[\left(\frac{WJ_N-WJ_S}{2WJ_P}\right)+\frac{1}{(\underline{r}_{\xi_2}\cdot\underline{r}_{\xi_2})}\left(\frac{(x_N-x_S)}{2}(x_N-2x_P+x_S)+\frac{(y_N-y_S)}{2}(y_N-2y_P+y_S)\right)\right]=0$$

Where r is the variable being solved for.

Separating the terms in equation 2.36 leads to the algebraic equation

$$a_P^E r_P = a_E^E r_E + a_W^E r_W + a_N^E r_N + a_S^E r_S + s^E \tag{2.37}$$

where

$$a_P^E = a_E^E + a_W^E + a_N^E + a_S^E \tag{2.38}$$

$$a_E^E = a_W^E = \frac{g^{11}}{4}\frac{(r_E-r_W)^2}{(\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_1})} \tag{2.39}$$

$$a_N^E = a_S^E = \frac{g^{22}}{4}\frac{(r_N-r_S)^2}{(\underline{r}_{\xi_2}\cdot\underline{r}_{\xi_2})} \tag{2.40}$$

$$s^E = g^{11}\frac{(r_E-r_W)}{2}\left[\left(\frac{WI_E-WI_W}{2WI_P}\right)+\frac{1}{(\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_1})}\left(\frac{(p_E-p_W)}{2}(p_E-2p_P+p_W)\right)\right] \tag{2.41}$$

$$+g^{22}\frac{(r_N-r_S)}{2}\left[\left(\frac{WJ_N-WJ_S}{2WJ_P}\right)+\frac{1}{(\underline{r}_{\xi_2}\cdot\underline{r}_{\xi_2})}\left(\frac{(p_N-p_S)}{2}(p_N-2p_P+p_S)\right)\right]$$

2.7 The LPE Equation

The LPE equation is formed by combining the Laplace, Poisson and Equidistribution terms together (equations 2.17, 2.29 and 2.37) into one equation with the user controlled constants λ^L, λ^P, λ^E, to weight each term respectively.

This leads to the algebraic system

$$a_P^{LPE} r_P = a_E^{LPE} r_E + a_W^{LPE} r_W + a_N^{LPE} r_N + a_S^{LPE} r_S + s^{LPE} \tag{2.42}$$

where, using equation 2.30 to remove the Poisson coefficients

$$a_P^{LPE} = a_E^{LPE} + a_W^{LPE} + a_N^{LPE} + a_S^{LPE} \tag{2.43}$$

$$a_E^{LPE} = a_W^{LPE} = (\lambda^L + \lambda^P) a_E^L + \lambda^E a_E^E \tag{2.44}$$

$$a_N^{LPE} = a_S^{LPE} = (\lambda^L + \lambda^P) a_N^L + \lambda^E a_N^E \tag{2.45}$$

$$s^{LPE} = \lambda^L s^L + \lambda^P s^P + \lambda^E s^E \tag{2.46}$$

2.8 Calculation of Weight Functions in the Equidistribution Term

The weight function W_i used in equation 2.32 drives the grid adaption. When the value of the weight functions at a point is higher than its neighbours, those neighbours will move towards it.

The requirements of the weight function in the current work are

- That its value should represent some measure of solution behaviour or error that varies significantly across the domain.

- That it should not go to zero, so as to avoid division by zero in equation

2.34.

• That it should be calculated from a normalised variable so that its behaviour can be predictable making the choice of the user controlled functions easier.

The typical form for most weight functions is

$$W = a_0 + \sum_k a_k M_k \tag{2.47}$$

where a is a constant and M_k is some function of the solution variables, commonly some combination of the first and second derivatives of a chosen variable along grid lines. Another form for weight equations is

$$\overline{\overline{W}} = \left(a + |\phi_x^n|\right)^{\frac{1}{2n}}$$

where ϕ is the solution variable and n is a constant.(Nakahashi (1987), Eiseman (1987)).

The most common weight function used is based on the gradient of the chosen variable. The gradient is probably the easiest measure of solution activity. With no extra terms the weight should lead to a constant gradient in the solution over each interval in the grid. This can be seen by substituting $W=\phi_x$ in the equidistribution equation 1.1, which gives $\phi_x \propto \Delta x$. Unfortunately, this form risks the weight on a point becoming zero, and the associated problems of division by zero in the LPE equations. Adaption to ϕ_x forces the grid to align perpendicular to the gradient of ϕ (Matsuno and Dwyer 1988).

The gradient as a weight function can be found in Jeng and Liou (1993), Lin and Wu (1993), Dannenhoffer (1991), Niederdrenk (1991), Lawal (1990), Tu and Thompson (1990), Arina (1989), Anderson (1987a), Hsu and Tu (1987) and Thompson et al. (1985). Ramakrishnan and Singh (1994), Benson and McRae (1991), and Chao and Liu (1991) use a combination of the normalised gradients of a range of variables.

Jeng and Liou (1992b) combine the gradient with a term for the arc length so that as the weight term goes to zero the grid distribution will go towards the predefined value.

$$W_i = \frac{1}{\Delta S_i} + a_1 \phi_x \tag{2.49}$$

where ΔS_i is the initial grid spacing.

2.8.1 Implemented Weight Functions

In the current work a number of different methods have been implemented, though only a few have been seriously tested. Which one is used depends upon the type of problem and the choice of variable. In all cases the chosen variable is interpolated onto the grid nodes from the cell centres using a series of simple linear interpolations and then non-dimensionalised to allow the user defined weight parameters to be independent of the magnitude of the chosen variable.

2.8.1.1 $a_1+a_2\phi$

This is the simplest weight function. There is no variation in the weights calculated at a point in different directions. This form has been used for grid generation for coastal configurations, using depth as the dependent variable (Kim and Thompson 1990), and using relative pressure for three dimensional grids over wings (Roache et al. 1991), (Kim and Thompson 1990), (Tu and Thompson 1990). In the current work it is used on the magnitude of velocity for convection problems such as the driven cavity flow (section 5.3.2) where there are no strong gradients and solution error is driven by the advection scheme.

2.8.1.2 $a_1+a_2\phi_x+a_3\phi_{xx}$

The term ϕ_{xx} takes account of the curvature of the solution. The inclusion of the second derivative helps to concentrate grid points about solution extrema, but may cause grid cells to oscillate if the solution is very irregular. Terms involving the

second derivative ϕ_{xx} are highly desirable as they are good indicators of solution activity, but unfortunately tend to be unstable. Examples of the use of this term in literature include Shyy (1992), (1991a), (1991b), (1990), Harvey (1991), Acharya and Moukalled (1990), Acharya and Patankar (1985), Thompson et al. (1985), (1986), and Brackbill and Saltzman (1982).

2.8.1.3 $a_1+a_2\phi_x+a_3k$

This is similar to the previous method, but with an improved approximation to the solution curvature, k, where

$$k = \frac{|\phi_{xx}|}{\left(1+(\phi_x)^2\right)^{\frac{3}{2}}} \qquad \textbf{(2.50)}$$

Including this extra term allows for solution extrema and also for areas where the solution is locally stationary. As the gradient goes to zero, so k will increase. In such areas the solution activity may be high, even though the gradient tends to zero. This is the main weight function used in the current work. The default constants used are a_1=1, a_2=1, a_3=0.01, as in Catherall (1991). Higher values of a_3 tend to destabilise the solution, though it may not have much effect set at this level. Examples of this weight function in literature include Kwon and Jeong (1995), Eiseman (1985a), Haase et al. (1985), Thompson (1985), and Noack and Anderson (1990).

2.8.1.4 $a_1+a_2\phi_x^2$

This form helps to contrast high and low gradients. It has been used by Roache et al. (1991), Holcomb (1984), and Thompson (1985).

2.8.1.5 $(a_1+a_2|\phi_x^n|)^{1/2n}$

Putting a_1=1, a_2=1 and n=1 and substituting into the equidistribution equation 1.1 gives $d\phi^2+dx^2=constant$, which implies equal arc lengths along the solution curve, and

for higher n is related to the truncation error (Nakahashi (1987), Catherall (1991), Hagmeijer (1994)).

2.8.1.6 $a_1+a_2\sqrt{(\phi_{xx})}$

$\sqrt{(\phi_{xx})}$ can be related to the truncation error in second order accurate solutions to first order differential equations, where truncation error is proportional to the second derivative multiplied by the grid spacing squared (Catherall (1991)).

2.8.1.7 $(1+a_1\phi_x^2)^{1/2}(1+a_2\phi_{xx}^2)^{1/2}$

This form is an alternative way of using the second derivative ϕ_{xx} to (2.8.1.2) used by Matsuno and Dwyer (1988).

2.8.2 Choice of Weight Constants

For the majority of the test cases the constants are chosen empirically.

Trial runs and user experience are important, but as long as there is a strong gradient to push the grid adaption a ratio of one to one for the first neutral constant to the main movement term will generally behave sensibly. Higher values can either be used to maintain grid quality or to encourage adaption.

Making the second term in functions 2.8.1.2 and 2.8.1.3 high will increase the effect of local gradients, but if too high will emphasise minor perturbations within the solution which may be as much a feature of the local grid shape as the actual solution. This can cause distorted cells in relatively quiet parts of the grid where little movement is expected.

An alternative is to set a given ratio size R for the smallest to the largest step size in each coordinate direction to determine the constants in the weight equation. (Niederdrenk (1991), Nakahashi and Deiwert (1985), (1986), (1987)).

Using the relationship

$$R = \frac{\delta x_{\min}}{\delta x_{\max}} = \frac{w_{x\,\max}}{w_{x\,\min}} \tag{2.51}$$

and starting from the basic equidistribution equation 1.1

$$\delta x\, w_x = constant \tag{2.52}$$

it is possible to determine the constant a_x in the weight equation

$$w_x = 1 + a_x |f_x| \tag{2.53}$$

by using

$$a_x = \frac{R - 1}{|f_{x\,\min}| - R\,|f_{x\,\max}|} \tag{2.54}$$

The weight along each family of grid lines is calculated using the same value a. Then a is updated at each call to the weight calculation routines

This method has the advantage of replacing two or three abstract numbers with one that has a physical relationship to the problem.

2.8.3 Weight Function Modifications

The weight function is optionally smoothed to reduce small perturbations and extend the influence of major solution phenomena by fitting the weight values to curves in the direction of action of each weight, defined by simple power laws, between major peaks and troughs.

In areas where the weight function is smooth small perturbations may kill the grid as it starts to kink towards the minor variations.

It has been shown (Bennett 1992) that smoothing of the weight function may improve

results by smearing the effect of high gradients over a larger number of cells.

In the current implementation there is an option to make the final weight value at a grid point equal to the average of the values calculated from one of the above equations on the point and the two neighbouring points.

2.8.3.1 Smoothing by Local Averaging

In order to spread the effect of solution phenomena an attempt has been made to extend the simple smoothing or 'smudging' operation carried out previously.

The available options are to smooth just the weight functions in each of their respective directions, as before, using

$$W_i^{new} = \frac{(W_{i-1} + 2W_i + W_{i-1})}{4} \tag{2.55}$$

Or to smooth the non-dimensionalised dependent variable, ϕ, using

$$U_{ij}^{new} = \frac{(nU_{ij} + U_{i-1j} + U_{i+1j} + U_{ij-1} + U_{ij+1})}{(n+4)} \tag{2.56}$$

in the 2D case, or,

$$U_{i,j,k}^{new} = \frac{(nU_{i,j,k} + U_{i-1,j,k} + U_{i+1,j,k} + U_{ij-1,k} + U_{i,j+1,k} + U_{i,j,k-1} + U_{i,j,k+1})}{(n+6)} \tag{2.57}$$

in the 3D case.

One, the other or both can be used. This smoothing step can be repeated many times.

The factor n used when smoothing the variable is controlled by the user.

In practice the smoothing step has on occasion only made an impact when the smoothing operation has been repeated 10, 20 or more times, when it begins to make

an impact on the time taken by the adaptive module to run. Smoothing by local averaging is used by Hall and Zingg (1995), Hagmeijer (1994), Benson and McRae (1991), Jeng and Liou (1989), and Seibert et al. (1989). Hall and Zingg (1995) also use weight clipping, whereby extreme weight values are reduced to an average of their neighbours.

2.8.3.2 Exponential Smoothing

Another option available is to use an inverse exponential function to smooth each weight function along the grid lines in the co-ordinate direction in which it operates. This allows solution phenomena to affect weight values some distance away.

The solution equation can be expressed as

$$W_i^{final} = W_i + a \sum_{k=i-\frac{n}{2}}^{i+\frac{n}{2}} W_k e^{-|x_i - x_k|} \qquad k \neq i \qquad \textbf{(2.58)}$$

where n is the number of points in that co-ordinate direction, a is a weight controlled by the user, and x_i-x_k is the distance between the main point and the one producing an additional influence.

The contribution to the weight value from points whose index are beyond the limits of the grid are calculated as if the grid was surrounded by mirror images. This has been done to ensure that points in the centre of the grid do not have larger weights than at the boundary solely because they are closer to more points.

This method is loosely based on the weight function used by Lee and Tsuei (1992a).

The smoothing produced by this method allows more emphasis to be put onto the equidistribution term in the main algorithm. This term is powerful yet very sensitive to minor variations in local weights and can lead to very distorted grids. The smoothing function helps to prevent this.

2.8.3.3 Power Law Smoothing

In an attempt to increase the range over which major solution phenomena have an effect and to smooth out small local perturbations the weight functions can be recalculated by fitting them along grid lines to a series of simple curves between local function maxima and minima. The curves are calculated by using a power law or exponential.

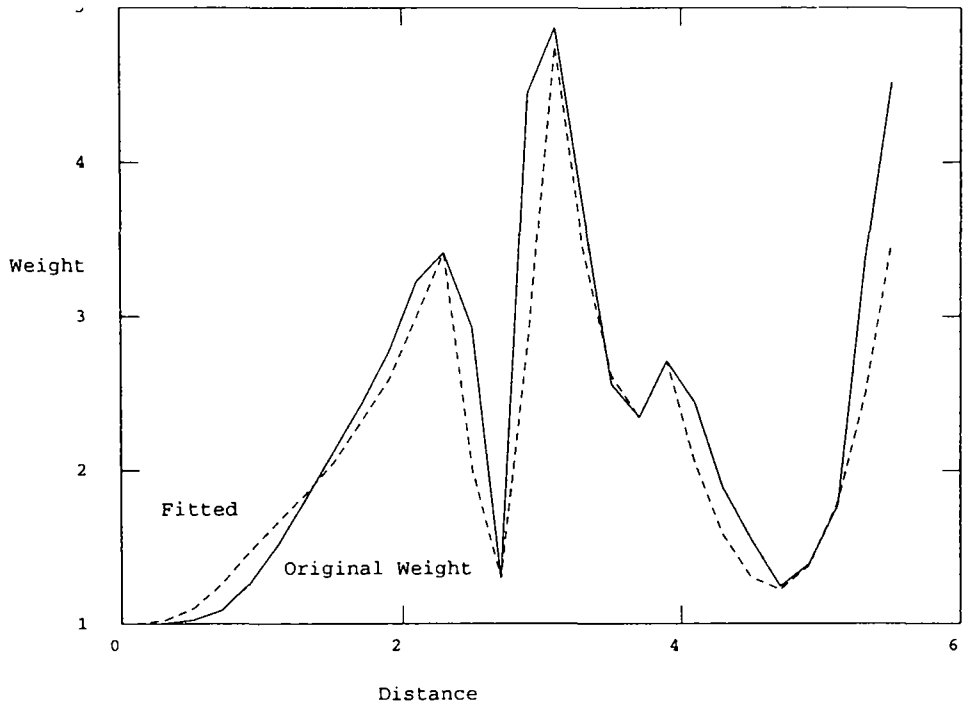


Figure 2.2 Weight function before and after fitting in 'real' case

Figure 2.2 shows how the smoothing function works in practise, the dotted line showing the final weight used in the LPE equation for a supersonic wedge cascade case where there are a series of reflected shocks that cross the grid line of interest.

For the purposes of this technique local break points are also defined as being at the centre of regions where the variation of the weight function is less than a given tolerance.

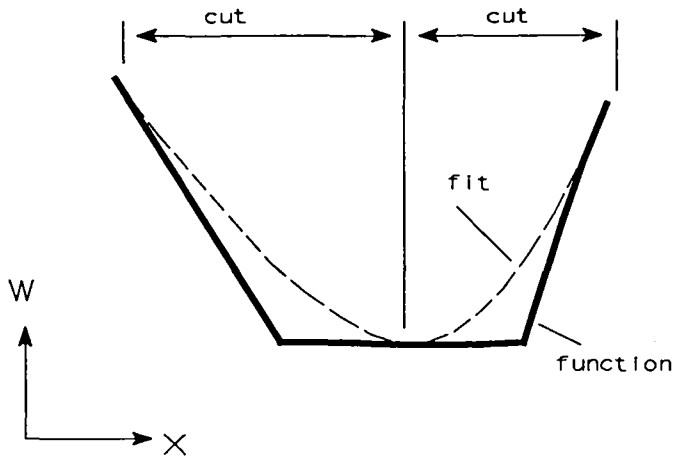


Figure 2.3 Break point at centre of 'flat' region in function

This function generates smooth weight gradients leading from areas of low activity to regions of high activity, which in turn leads to smooth variation in the grid spacing. Its use makes other smoothing unnecessary. The smoothness of the new weight functions allows more extreme parameters to be used within the main algorithm.

As the driving force behind grid point movement is the local gradient of the weight function the use of the power law enables the user to either promote the movement of grid points out of regions of low activity by using a power of less than one, or to concentrate on major phenomena by using a high power. Grid points will tend towards being distributed according to the power law, as the algorithm is driven by equidistribution.

As each weight function relates directly to one family of grid lines in the structured grid, this process of smoothing along grid lines is straightforward. The same technique may not be so applicable to unstructured grids unless weight surfaces could be built up. Such a technique might be very powerful, though could be complex, especially for three dimensional grids.

For power law smoothing of the x direction weight function along a grid line which varies predominately in the x direction the equation used to calculate the new weight value w_{xi}^{n} is

$$w_{x_i}{}^{n} = (w_{x_i} - w_{x_f})\left(\frac{x_i - x_f}{x_l - x_f}\right)^{a} + w_{x_f} \tag{2.59}$$

where w_{xf}, w_{xl} are the first and last weight values, x_f, x_l, x_i are the first, last and current x displacements, and a is a user controlled parameter that can control the grid variation. As the weight values are linked to physical dimensions not curvilinear space, x displacements are used not arc lengths.

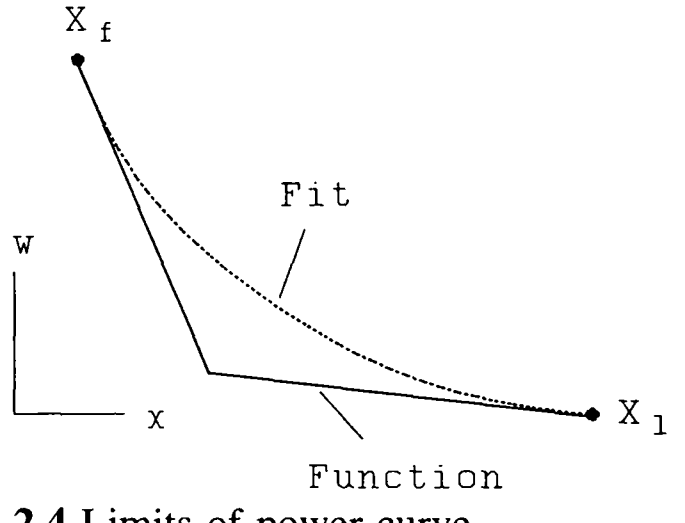


Figure 2.4 Limits of power curve

For completeness the exponential fit has also been implemented, though the power law is cheaper and more straightforward to use to fit the function variation. The equivalent equation to 2.59 above is

$$w_{x_i}{}^{n} = w_{x_f} e^{\left[\left(\ln(w_{x_l}) - \ln(w_{x_f})\right)\left(\frac{x_i - x_f}{x_l - x_f}\right)\right]} \tag{2.60}$$

2.9 Closure

This chapter has presented the full discretisation of the LPE equation in one, two and three dimensions using central difference formulae. In addition the formulation of the weight function used in the equidistribution term to drive grid adaption is presented. All adaption methods depend on a weight function that models solution behaviour. The range and treatment of these functions varies greatly between different cases and different methods. The novel technique presented here to impose a smooth function over the weight distribution using a power law or exponential curve is a fast method to produce a smooth grid.

In the full implementation of the LPE equation only its three dimensional form is coded. Two and one dimensional problems are dealt with by cancelling the extra terms and using the appropriate forms of the metric tensors.

The final discretised equations are solved using a point by point scheme with relaxation. Their format would however allow the implementation of a tri diagonal matrix algorithm or similar fast solver instead. As the algorithm is designed to be attached to a full CFD code it may be possible to use the solvers within that code instead.

CHAPTER 3: ADDITIONAL TOOLS

3.1 Introduction

The purpose of this chapter is to introduce and describe the additional tools needed to support grid adaption. These tools are used to manipulate the data from the grid and the solution before using the LPE algorithm, and to manipulate the data emanating from the LPE algorithm. They are used to refine the main algorithm. Each tool represents a necessary part of the grid adaption process. A modular structure ensures that the methods each tool uses might be replaced if better techniques become available or the criteria upon which they have been chosen change.

The bulk of this chapter describes surface and curve fitting algorithms used to maintain the geometry of body fitted coordinate grids whilst allowing grid nodes to move on grid boundaries. Other tools described cover interpolation techniques used to transfer data from grid cell centres to grid nodes and between grids, a method to prevent grid crossover and movement limiting.

3.2 Surface and Curve Fitting

Though it is easy to allow grid points to move on square boundaries by just ignoring components of the calculated movement vector, movement along curved and angled surfaces and lines can only be allowed if equations to recreate them are known or can be calculated.

Allowing grid points to move on boundaries can be a crucial part of the adaptive process, as often the boundary regions are the most important parts of the grid. In any case leaving the grid fixed in such places whilst allowing movement to take place in adjacent nodes can lead to dangerously skewed cells. In addition intelligent use of fixed curves and surfaces can be used to maintain important features of the original grid. In cases such as the RAE 2822 aerofoil (section 5.3.1) fixed curves are used to

allow movement along grid lines parallel to the aerofoil surface so as to maintain the initial skewed distribution of grid cells.

As the current work is designed to be independent of CFD codes and in particular grid generators, no prior knowledge can be assumed about the grid geometry. All information about geometry has to be computed using the original grid point locations. Equations for all curves and surfaces designated by the user as important have to be developed using assumptions based on the location of the grid points that lie on them in the original grid.

In the current work user defined surfaces and curves are modelled using bicubic patches and cubic splines respectively. Each curve and surface is broken up into a series of segments or patches and a parametric cubic equation is developed for each one, using the location of the grid points to calculate the coefficients.

The technique used to model surfaces is based on the work of Agbormbai (1991). As the current work is concerned with structured grids only, curves are defined as grid lines with two indices fixed and one varying, and surfaces as areas with one index fixed and two varying.

In the code which accompanies the current work the user can define any number of curves and surfaces within the grid. These will be maintained regardless of the local grid movement and with the restriction that interior points cannot move beyond the original surface boundaries. A feature of the current work is the automatic manipulation of user defined surfaces to preserve important geometrical features such as peaks and troughs, and discontinuities.

This section describes how the parametric equations for the curves and surfaces are developed, and how they are used to recreate the geometry after grid movement has taken place.

3.2.1 Curve Fitting

In the structured grid a curve is defined as a number of connected grid points with only one index varying. A curve is modelled by representing it as a series of cubic splines covering individual segments or patches of up to four points (figure 3.1).

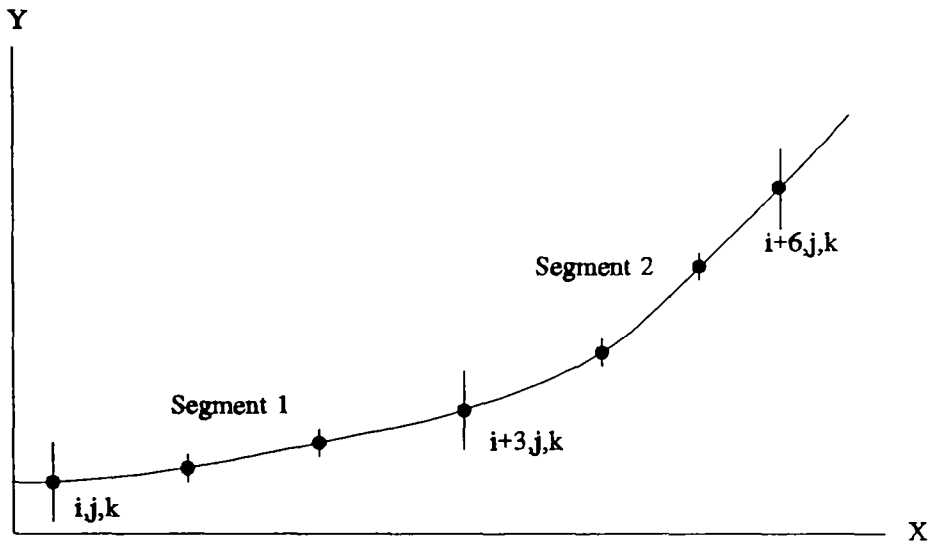


Figure 3.1 Curve split into segments

The coefficients of the cubic splines are obtained by using the original position of the grid points that define the curve. Each segment that defines the curve is treated individually and simply by using parametrised space (figure 3.2).

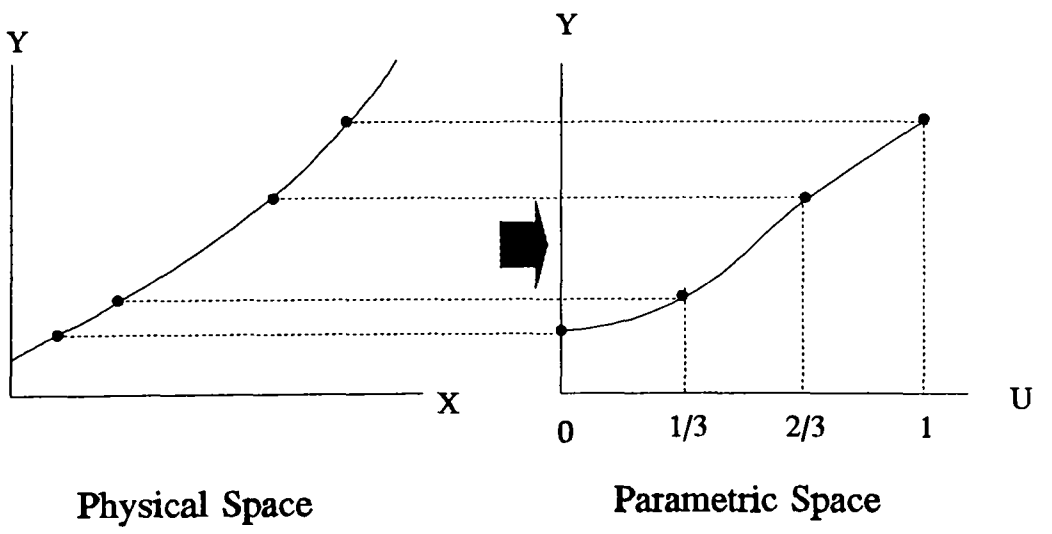


Figure 3.2 Parametrised Space

Each segment is defined by the following three equations:-

$$x(u) = a_{x1}u^3 + a_{x2}u^2 + a_{x3}u + a_{x4} \tag{3.1}$$

$$y(u) = a_{y1}u^3 + a_{y2}u^2 + a_{y3}u + a_{y4} \tag{3.2}$$

$$z(u) = a_{z1}u^3 + a_{z2}u^2 + a_{z3}u + a_{z4} \tag{3.3}$$

The curve has an associated axis which is implied by the index of its grid points which are allowed to vary. This should also correspond to the direction in which it stretches most. During adaption each point that lies on the curve is allowed to move freely along this axis up to the limits of the curve. Each point is then mapped back onto the curve by using this component of its position vector to determine the segment it lies in and a parameter to be fed back into equations 3.1-3.3. As one component of the position vector is known prior to mapping only two of the above equations will be needed. For example for the curve from $X_{i,j,k}$ to $X_{i+n,j,k}$ only the coefficients to equations 3.2 and 3.3 are calculated.

The values that need to be stored to fit a curve are the component of the original grid coordinates that relates to the index which varies to allow parameter recovery and two sets of four coefficients for each segment which relate to the two parametric equations.

3.2.1.1 Derivation of Cubic Coefficients

The generalised parametric equation $Z(u)$ can be written as

$$Z(u) = a_1 u^3 + a_2 u^2 + a_3 u + a_4$$

$$= \begin{bmatrix} u^3 u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{3.4}$$

Assuming that in parametrised space the points in the segment lie at intervals of 1/3 between 0 and 1, then the coefficients of the spline can be calculated by building up four equations for the four unknowns, given by

$$\begin{bmatrix} Z(0) \\ Z(\frac{1}{3}) \\ Z(\frac{2}{3}) \\ Z(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.037 & 0.111 & 0.333 & 1 \\ 0.296 & 0.444 & 0.667 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{3.5}$$

Inverting the main matrix to solve for the unknown coefficients gives,

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -4.5 & 13.5 & -13.5 & 4.5 \\ 9 & -22.5 & 18 & -4.5 \\ -5.5 & 9 & -4.5 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z(0) \\ Z(\frac{1}{3}) \\ Z(\frac{2}{3}) \\ Z(1) \end{bmatrix} \tag{3.6}$$

Though this system will work best for a regular grid it will still give good results for an irregular grid if the calculation of the parameter is accurate enough. Using this system does however prevent the need to invert a four by four matrix for each set of coefficients for every segment. Though this would not necessarily be too great an overhead just for curve fitting, especially as the process only has to take place once for each initial grid, for surface fitting each matrix is 16 by 16. Inversion of such large matrices for each surface patch is not practical.

3.2.1.2 'Light' Segments

As segments may occur at the end of curves which have fewer than four points the full cubic equation cannot be generated unless extra points are interpolated into the region, or overlapping segments are used which do not guarantee continuity in the line model. Instead simpler forms of the above equations can be used to generate the coefficients for the quadratic and linear equations that can be generated from three point and two point segments. These coefficients are treated exactly the same way as for the full cubic equation when the curve is recovered, just with the higher order terms set to zero.

3.2.1.2.1 Three Point Segment

The three point segment shown in figure 3.3 is parametrised using a quadratic spline with coefficients calculated in the same manner as for the cubic spline above.

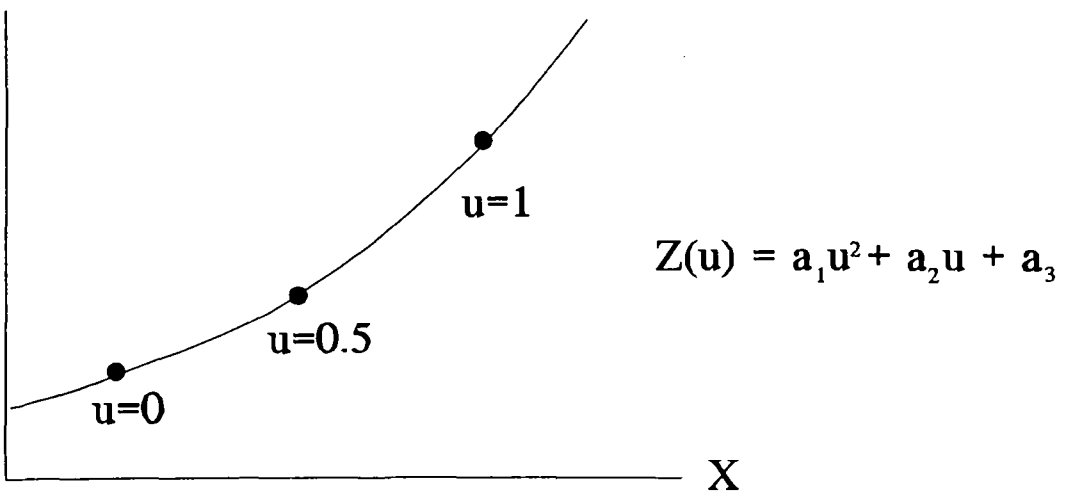


Figure 3.3 Three Point Segment

The parametric function Z(u) is written:-

$$Z(u) = a_1 u^2 + a_2 u + a_3$$

$$= \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad (3.7)$$

Assuming the three points occur at the parameter values 0, 1/2 and 1 it is possible to solve for the three constants $a_{i,i=1,3}$, as given by

$$\begin{bmatrix} Z(0) \\ Z(\frac{1}{2}) \\ Z(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.25 & 0.5 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.8}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z(0) \\ Z(\frac{1}{2}) \\ Z(1) \end{bmatrix} \tag{3.9}$$

3.2.1.2.2 Two Point Segment

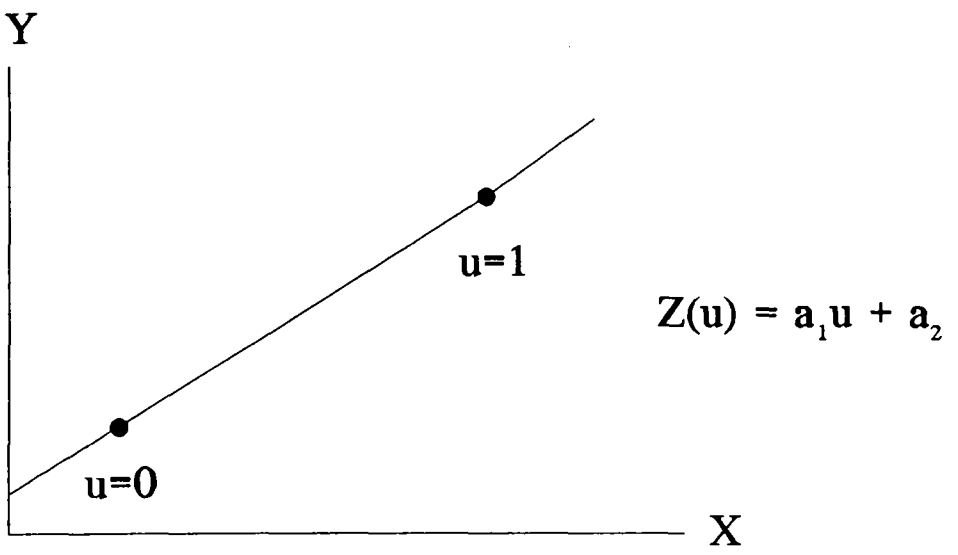


Figure 3.4 Two point segment

The two point segment shown in figure 3.4 is parametrised using the linear equation.

$$Z(u) = a_1 u + a_2 \tag{3.10}$$

Where,

$$\begin{aligned} a_1 &= Z(1) - Z(0) \\ a_2 &= Z(0) \end{aligned} \tag{3.11}$$

3.2.1.3 Curve Recovery

Curve recovery is the mapping of a point back onto a curve after it has been moved in one direction by manipulating the remaining two components of its position vector. To map the point back onto the curve firstly its location relative to the cells in the original grid must be found so that the patch in which it lies can be determined and then secondly a parameter has to be recovered by comparing its position to the original grid.

The coordinates of the original curve are stored and the patch positions are known implicitly.

The location of the point is determined by using a simple binary search pattern, comparing the directional component that relates to the index of the curve which varies to the equivalent components of the original points that define the curve. The point is compared to the half way point of a domain whose limits are initially defined by the start and end points of the curve and then by the centre point and whichever limit is on the other side of it until it can only lie in one cell.

The parameter is determined by assuming that it varies in a linear profile between the original points of the patch, and by knowing that those points occur at equal intervals between 0 and 1.

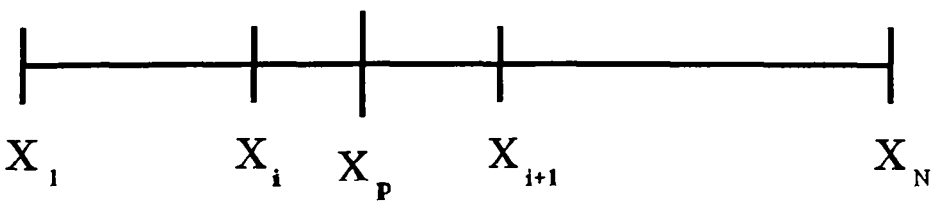


Figure 3.5

In figure 3.5 the parameter v of the point X_p that lies between the original grid points $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ in a patch of N points where i varies from 1,N is found by using

$$v = \frac{x_p - x_i}{N(x_{i+1} - x_i)} + \frac{i}{N} \tag{3.12}$$

Note that the points at the ends of the curve are not allowed to move at all.

3.2.2 Surface Fitting

A surface is modelled as a series of individual patches covering up to four by four points. Each patch is modelled using a bicubic equation that contains sixteen coefficients, calculated using the location of the original sixteen points.

Figure 3.6 shows a grid broken into patches

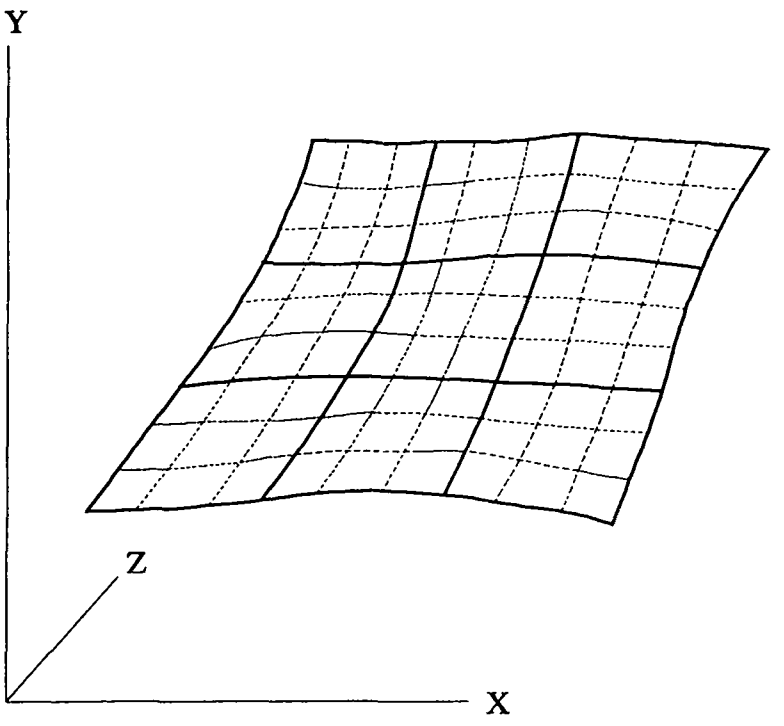


Figure 3.6 Surface split into nine patches

As for curve modelling each patch is fitted in parametric space (figure 3.7).

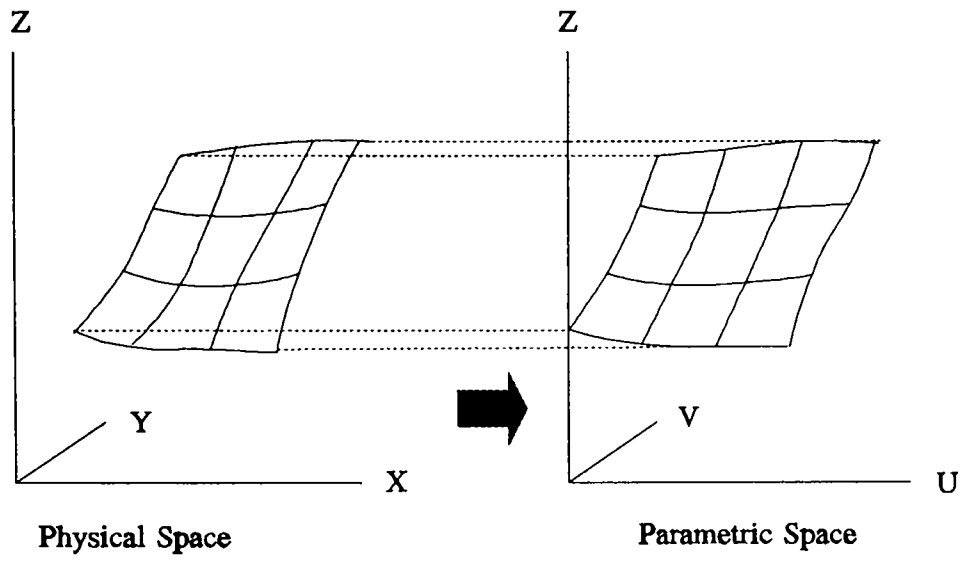


Figure 3.7 Patch in Parametric Space

The generalised parametric bicubic equation can be written in the form

$$Z(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \tag{3.13}$$

A point is mapped back on the surface after adaption by calculating a new value for the component of its position vector that matches the index of the surface that does not vary. This is achieved by calculating parameters to pass into the bicubic equation using the position of the new point relative to the position of the points in the original grid.

Only one set of coefficients need to be generated for each surface patch, as only one direction component is effected.

3.2.2.1 Derivation of Bicubic Coefficients

The sixteen unknown coefficients can be found by using the location of the sixteen points within the patch. If equation 3.13 is written in the form

$$Z(u,v)=[P][C] \tag{3.14}$$

where *[P]* is the parameter matrix,

$$[P]=[u^3v^3,\ u^3v^2,\ u^3v,\ u^3\ ,\ u^2v^3,\ u^2v^2,\ u^2v,\ u^2\ ,\ uv^3,\ uv^2,\ uv,\ u\ \ ,\ v^3\ ,\ v^2\ ,\ v\ ,\ 1] \tag{3.15}$$

And *[C]* is the coefficient matrix, written as:-

$$[C]=[a_{11},\ a_{12},\ a_{13},\ a_{14},\ a_{21},\ a_{22},\ a_{23},\ a_{24},\ a_{31},\ a_{32},\ a_{33},\ a_{34},\ a_{41},\ a_{42},\ a_{43},\ a_{44}]^T \tag{3.16}$$

Then, assuming each point in the patch occurs at intervals of 1/3 between 0 and 1 it is possible to build up the system

$$[Z]=[P][C] \tag{3.17}$$

Where *[Z]* is given by

$$[Z] = \begin{bmatrix} Z(0,0) \\ Z(0,\frac{1}{3}) \\ Z(0,\frac{2}{3}) \\ Z(0,1) \\ Z(\frac{1}{3},0) \\ Z(\frac{1}{3},\frac{1}{3}) \\ Z(\frac{1}{3},\frac{2}{3}) \\ Z(\frac{1}{3},1) \\ Z(\frac{2}{3},0) \\ Z(\frac{2}{3},\frac{1}{3}) \\ Z(\frac{2}{3},\frac{2}{3}) \\ Z(\frac{2}{3},1) \\ Z(1,0) \\ Z(1,\frac{1}{3}) \\ Z(1,\frac{2}{3}) \\ Z(1,1) \end{bmatrix} \tag{3.18}$$

And the coefficients can be found by taking the inverse of *[P]*

$$[C] = [P]^{-1}[Z] \tag{3.19}$$

The assumption of the position of the original points in parametric space is important to avoid having to invert a 16 by 16 array for each patch. Instead one standard inverted array can be used and it is assumed that the parameters can be accurately estimated. The parameter matrix *[P]* in table 3.1 and its inverse $[P]^{-1}$ in table 3.2 are given on the following page.

$$[P] = \begin{bmatrix}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0370 & 0.1111 & 0.3333 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2963 & 0.4444 & 0.6667 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0370 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \\
0.0014 & 0.0041 & 0.0123 & 0.0370 & 0.0041 & 0.0123 & 0.0370 & 0.1111 & 0.0123 & 0.0370 & 0.1111 & 0.3333 & 0.0370 & 0.1111 & 0.3333 & 1.0000 \\
0.0110 & 0.0165 & 0.0247 & 0.0370 & 0.0329 & 0.0494 & 0.0741 & 0.1111 & 0.0988 & 0.1481 & 0.2222 & 0.3333 & 0.2963 & 0.4444 & 0.6667 & 1.0000 \\
0.0370 & 0.0370 & 0.0370 & 0.0370 & 0.1111 & 0.1111 & 0.1111 & 0.1111 & 0.3333 & 0.3333 & 0.3333 & 0.3333 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.2963 & 0.0000 & 0.0000 & 0.0000 & 0.4444 & 0.0000 & 0.0000 & 0.0000 & 0.6667 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \\
0.0110 & 0.0329 & 0.0988 & 0.2963 & 0.0165 & 0.0494 & 0.1481 & 0.4444 & 0.0247 & 0.0741 & 0.2222 & 0.6667 & 0.0370 & 0.1111 & 0.3333 & 1.0000 \\
0.0878 & 0.1317 & 0.1977 & 0.2963 & 0.1317 & 0.1975 & 0.2963 & 0.4444 & 0.1975 & 0.2963 & 0.4444 & 0.6667 & 0.2963 & 0.4444 & 0.6667 & 1.0000 \\
0.2963 & 0.2963 & 0.2963 & 0.2963 & 0.4444 & 0.4444 & 0.4444 & 0.4444 & 0.6667 & 0.6667 & 0.6667 & 0.6667 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\
0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \\
0.0370 & 0.1111 & 0.3333 & 1.0000 & 0.0370 & 0.1111 & 0.3333 & 1.0000 & 0.0370 & 0.1111 & 0.3333 & 1.0000 & 0.0370 & 0.1111 & 0.3333 & 1.0000 \\
0.2963 & 0.4444 & 0.6667 & 1.0000 & 0.2963 & 0.4444 & 0.6667 & 1.0000 & 0.2963 & 0.4444 & 0.6667 & 1.0000 & 0.2963 & 0.4444 & 0.6667 & 1.0000 \\
1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000
\end{bmatrix}$$

Table 3.1 Parameter Matrix

$$[P]^{-1} = \begin{bmatrix}
20.25 & -60.75 & 60.75 & -20.25 & -60.75 & 182.25 & -182.25 & 60.75 & 60.75 & -182.25 & 182.25 & -60.75 & -20.25 & 60.75 & -60.75 & 20.25 \\
-40.50 & 101.25 & -81.00 & 20.25 & 121.50 & -303.75 & 243.00 & -60.75 & -121.50 & 303.75 & -243.00 & 60.75 & 40.50 & -101.25 & 81.00 & -20.25 \\
24.75 & -40.50 & 20.25 & -4.50 & -74.25 & 121.50 & -60.75 & 13.50 & 74.25 & -121.50 & 60.75 & -13.50 & -24.75 & 40.50 & -20.25 & 4.50 \\
-4.50 & 0.00 & 0.00 & 0.00 & 13.50 & 0.00 & 0.00 & 0.00 & -13.50 & 0.00 & 0.00 & 0.00 & 4.50 & 0.00 & 0.00 & 0.00 \\
-40.50 & 121.50 & -121.50 & 40.50 & 101.25 & -303.75 & 303.75 & -101.25 & -81.00 & 243.00 & -243.00 & 81.00 & 20.25 & -60.75 & 60.75 & -20.25 \\
81.00 & -202.50 & 162.00 & -40.50 & -202.50 & 506.25 & -405.00 & 101.25 & 162.00 & -405.00 & 324.00 & -81.00 & -40.50 & 101.25 & -81.00 & 20.25 \\
-49.50 & 81.00 & -40.50 & 9.00 & 123.75 & -202.50 & 101.25 & -22.50 & -99.00 & 162.00 & -81.00 & 18.00 & 24.75 & -40.50 & 20.25 & -4.50 \\
9.00 & 0.00 & 0.00 & 0.00 & -22.50 & 0.00 & 0.00 & 0.00 & 18.00 & 0.00 & 0.00 & 0.00 & -4.50 & 0.00 & 0.00 & 0.00 \\
24.75 & -74.25 & 74.25 & -24.75 & -40.50 & 121.50 & -121.50 & 40.50 & 20.25 & -60.75 & 60.75 & -20.25 & -4.50 & 13.50 & -13.50 & 4.50 \\
-49.50 & 123.75 & -99.00 & 24.75 & 81.00 & -202.50 & 162.00 & -40.50 & -40.50 & 101.25 & -81.00 & 20.25 & 9.00 & -22.50 & 18.00 & -4.50 \\
30.25 & -49.50 & 24.75 & -5.50 & -49.50 & 81.00 & -40.50 & 9.00 & 24.75 & -40.50 & 20.25 & -4.50 & -5.50 & 9.00 & -4.50 & 1.00 \\
-5.50 & 0.00 & 0.00 & 0.00 & 9.00 & 0.00 & 0.00 & 0.00 & -4.50 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
-4.50 & 13.50 & -13.50 & 4.50 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
9.00 & -22.50 & 18.00 & -4.50 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-5.50 & 9.00 & -4.50 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{bmatrix}$$

Table 3.2 Inverted Parameter Matrix

3.2.2.2 'Light' Patches

There is no guarantee that any defined surface will split into patches of exactly four points by four points, so a method must be developed to deal with the other cases which may occur at its end. The options are to use a modified form of the inverted matrix in table 3.2 or to interpolate extra points using the data available to get a 'full' patch.

Using a full patch which overlaps the previous patch does not guarantee continuity and may cause problems when it comes to locating the position of points which may move into this region relative to the points.

A modified form of the matrix had been used in previous work (Agbormbai 1991) but involves the storage of a number of matrices for use in specific cases which are rare. For surface definition there are eight extra special cases and each matrix for storing surfaces is 16 by 16. This when coded gives a considerable overhead to storage requirements and is a potential source of error.

For simplicity simple interpolation is used to create the correct number of 'artificial' points to define a normal patch in the current work. The accuracy of the parametric equation can only depend upon the information that is available when its coefficients are calculated. What inefficiency this method may introduce is balanced by the fact that the coefficients only have to be generated once for a particular surface.

Simple linear interpolation is used to generate extra points where there are just two known points in a particular direction, and a quadratic spline is used where there are three.

3.2.2.3 Surface Recovery

Surface recovery is the process by which a point on a surface is moved back to the surface after adaption. To allow the point to move back the patch on the original grid

in which it lies is found then two parameters are estimated based on its position within the patch.

It is assumed that the patch locations are given implicitly by the layout of the grid.

The task of locating the point is simplified by the knowledge that it must lie in the same subsurface that it was in before adaption, as movement beyond those limits is not permitted. The patch location is found using a binary search method, dividing the original surface into successive quarters and testing the relative position of the point each time until there is only one patch it can lie in.

The cell in the patch in which the point lies is found using the same method. If the cell lies on the boundary of the patch then the point is tested against the lines that define the edge of the cell to check that it is within the patch. If it is not, then the point is tested against the neighbouring patch in the appropriate direction.

The parameters for the patch equation are calculated by using bilinear interpolation over the cell in which the point lies, the parameter values being known at the corner points. The equations to recover the parameters u and v are,

$$u(x,y) = \begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} a_{u1} & a_{u2} \\ a_{u3} & a_{u4} \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix} \tag{3.20}$$

$$v(x,y) = \begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} a_{v1} & a_{v2} \\ a_{v3} & a_{v4} \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix} \tag{3.21}$$

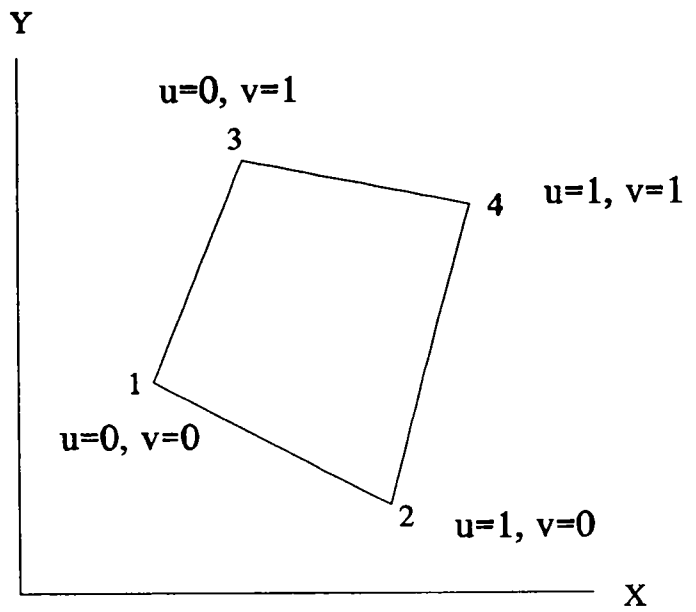


Figure 3.8 2D Interpolation Problem

Taking the general case in figure 3.8 where the parameters vary between 0 and 1 formulae for the coefficients can be built up by inverting the matrix that can be assembled up with the information available,

$$\begin{bmatrix} x1y1 & x1 & y1 & 1 \\ x2y2 & x2 & y2 & 1 \\ x3y3 & x3 & y3 & 1 \\ x4y4 & x4 & y4 & 1 \end{bmatrix} \begin{bmatrix} a_{u1} \\ a_{u2} \\ a_{u3} \\ a_{u4} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{3.22}$$

$$\begin{bmatrix} x1y1 & x1 & y1 & 1 \\ x2y2 & x2 & y2 & 1 \\ x3y3 & x3 & y3 & 1 \\ x4y4 & x4 & y4 & 1 \end{bmatrix} \begin{bmatrix} a_{v1} \\ a_{v2} \\ a_{v3} \\ a_{v4} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{3.23}$$

The result of the interpolation can be scaled to account for the range of values of the parameters that the cell covers.

3.2.3 Surface and Curve Breaking

Surface and curve breaking refers to the process by which user defined surfaces and curves are automatically broken into smaller units around important geometrical features such as turning points and discontinuities (figures 3.9-3.11). These are places where the location of a single point may be vital to the local geometry, and where a patch or spline may not be able to adequately model the shape. Points at the end of curves and the edges of surfaces do not move. These important geometry defining locations will always be maintained.

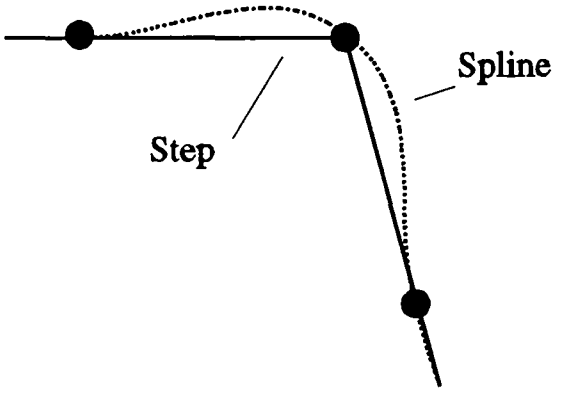


Figure 3.9 Problems at Discontinuity

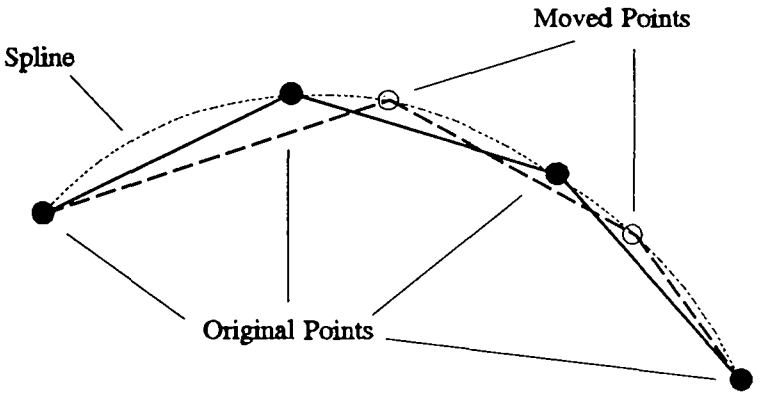


Figure 3.10 Problems at turning points

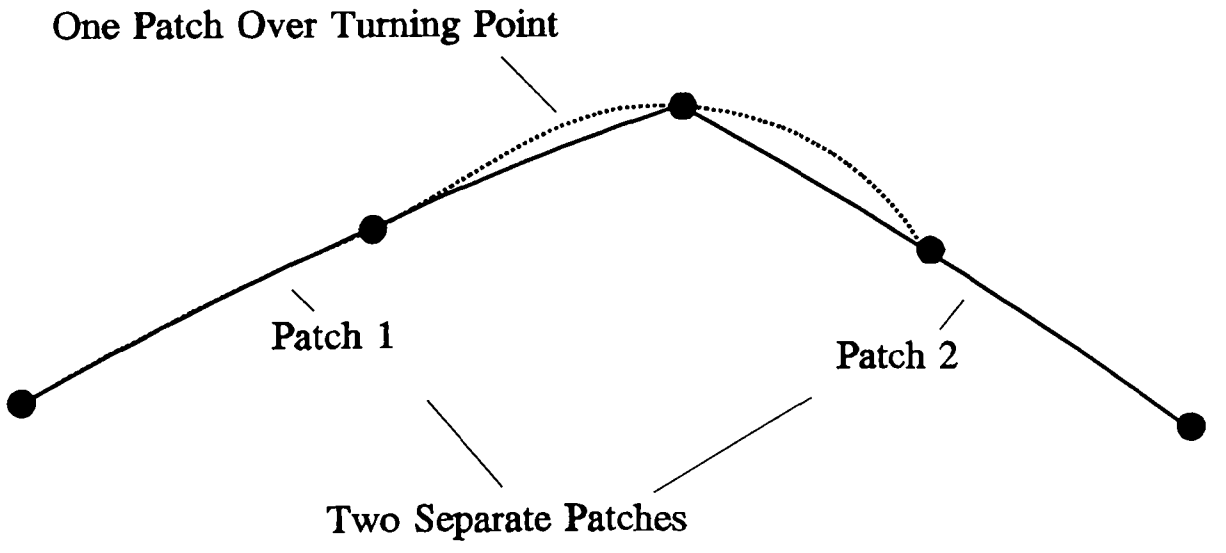


Figure 3.11 How surface breaking can help

The process has the advantage of reducing the complexity of any curve or surface that a patch has to model and therefore it may reduce possible errors although the number of light patches may be increased.

The process is essentially the same for both curves and surfaces.

Curves and surfaces are broken up at turning points by using the direction of the slope and at discontinuities by using its rate of change. The slope is monitored by using the first and second derivatives of the components of the grid point position vector that relate to the fixed indices of the grid relative to the components that vary. The derivatives are calculated using second order central difference formula.

In the code that accompanies the current work the user can switch the different methods of cutting on and off and set the tolerances used to determine where it takes place. No cut is allowed within one patch distance of another. If a conflict occurs then the tolerance used is increased and the checking routine run again. Cutting also takes place if at any point the gradient becomes infinite.

Simple examples of surface breaking are given in figures 3.12-3.15.

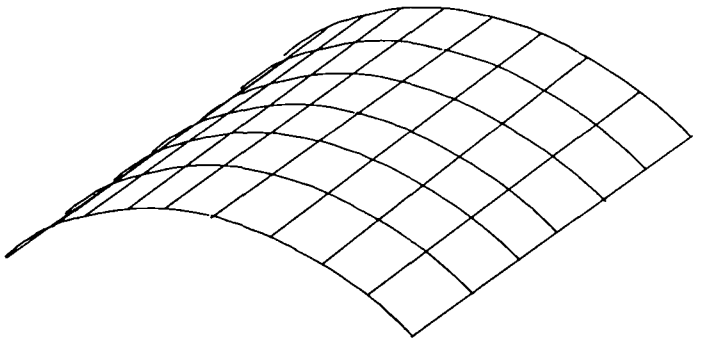


Figure 3.12 Surface with turning point

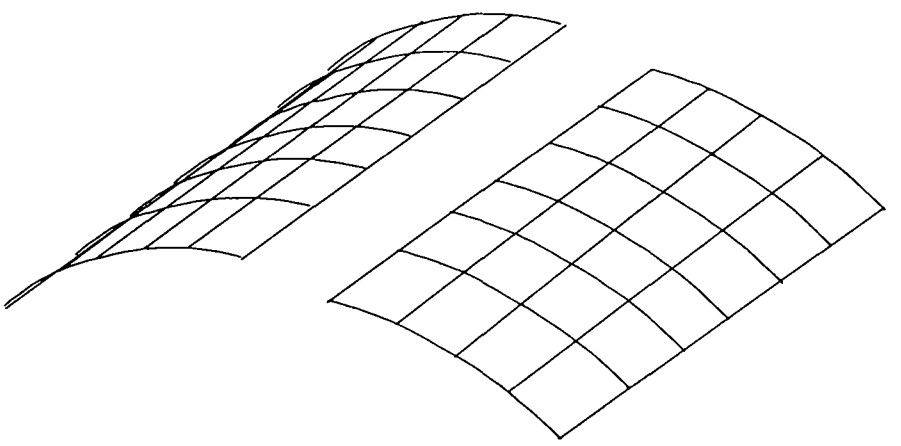


Figure 3.13 Surface with turning point after cutting

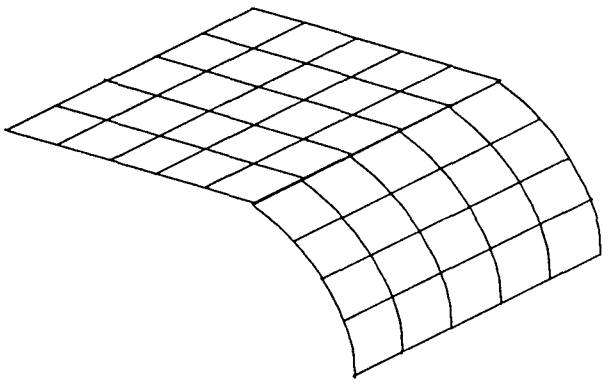


Figure 3.14 Surface with discontinuity

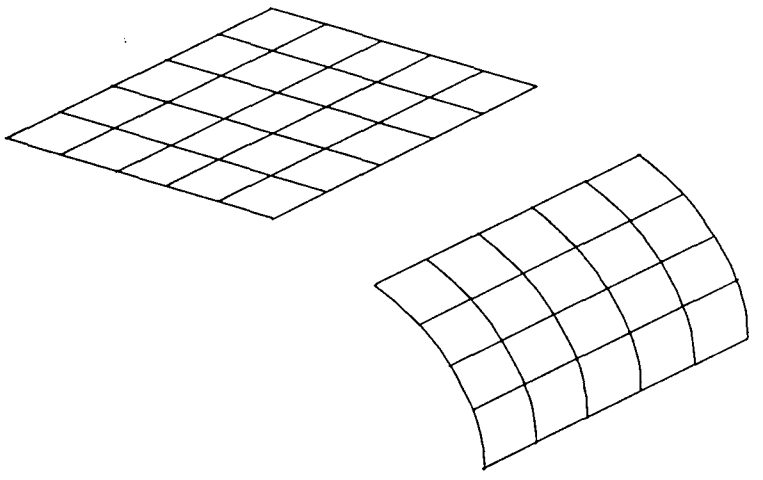


Figure 3.15 Surface with discontinuity after cutting

3.3 Interpolation

Interpolation routines are used to give the value at a point of a function from a discrete set of data at known locations. Interpolation is an important part of grid adaption, as the value of any data that vary over a grid may be required at any point within its limits, either to allow the solution to be transferred between old and new grids, or transfer data from cell centres to cell vertices. The type of interpolation that is used depends upon the characteristics of each problem. In every case, the interpolation technique must be efficient in time and the use of resources.

Interpolation within curve and surface fitting for calculating the parameters in the cubic and bicubic splines has already been described (3.2.1.3 and 3.2.2.3). In both cases the parameters are known to vary over a fixed range between known grid points over each line or surface. The location of the new grid point is known with respect to the known data as the patch in which it lies has to be calculated. Its position will however vary between the extremes of the patch and any interpolation scheme must

give accurate results over the whole range of the patch. For curves, the new point must lie between two known data points thus the problem is one dimensional, so a simple linear variation is assumed. For surfaces the new point must lie between four known data points thus the problem is two dimensional, and a bilinear equation is used. The bilinear equation requires the inversion of a four by four matrix, which is made possible by knowing the range over which the parameters vary and can be manipulated to give an algebraic equation which just requires the location of the original grid points.

The adaption algorithm relies upon knowing data at grid points and the current work has been implemented within a finite volume CFD code. Interpolation is needed to transfer data from the cell centres, the nominal variable position within the control volume, and the grid points. In this case the interpolation technique used must be three dimensional, but may also be used for two or one dimensional data. The interpolation point is never close to any known data location and will tend to lie equidistant from the known data. Its location relative to the known data is known implicitly from its index. Multiple one dimensional interpolation is used for this problem.

To allow multiple iterations of the adaption algorithm it is necessary to transfer data between successive grids. As the adaption algorithm only relies upon the values at grid points it is only necessary to transfer the data at those points. The new grid point is likely to be close to the previous point, but the location is not known implicitly, therefore Shepard's interpolation (section 3.3.2) can be used to give the new data.

After each adaption the grid and therefore the solution moves, so it may be necessary to interpolate the solution from the old grid to the new. Whatever happens adaption may give a kick to the solution. The simplest option is to assume that the grid has not moved significantly and that the previous solution is a good guess for the new grid (Danenhoffer 1991). Alternatively the solution can be transferred using the same reasoning as for the transfer of data between grid points assuming that cell values lie at the cell centres. Shepard's interpolation can be used to give the new data. The only

additional step is the calculation of the location of the cell centres of the old and new grid.

One other approach to interpolating data between grids for steady state calculations which is not discussed here is to use a pseudo time step to convert the grid movement into a velocity, which is then used with the transformation metrics to give the new values of the flow variables. This approach is covered by Benson and McRae (1991).

3.3.1 Multiple One Dimensional Linear Interpolation

Multiple one dimensional linear interpolation is used to calculate data at grid points from known values at cell centres. This technique is a combination of successive one dimensional linear interpolations used to reduce the three dimensional problem first to a two dimensional problem then finally to a one dimensional problem which will give the answer (figure 3.16).

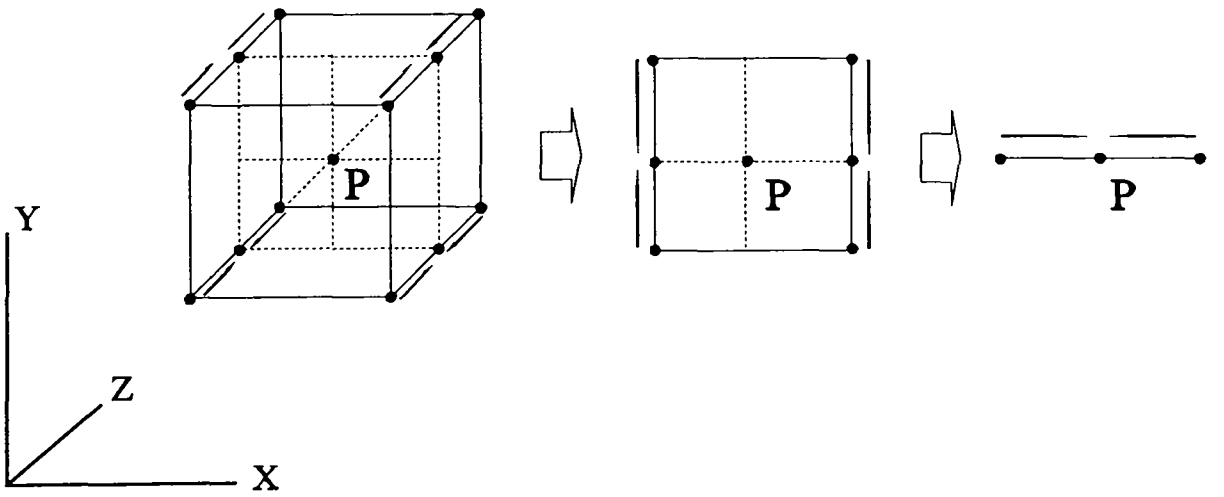


Figure 3.16 Multiple One Dimensional Interpolation

The technique uses eight surrounding cells which touch at the point, and eight cell corners as the location of the data (figure 3.17). The initial eight points are reduced to four by interpolating four pairs in one dimension, then to two by interpolation in another.

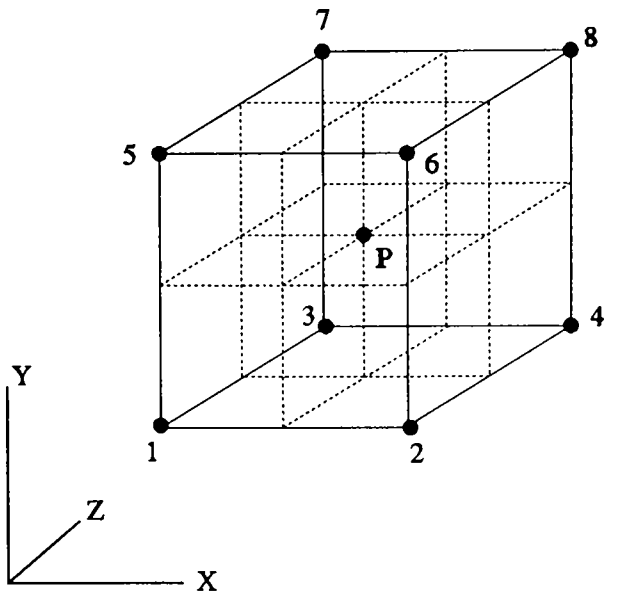


Figure 3.17 Point Definition

The first step is,

$$\phi_{13} = \phi_1 + \frac{(\phi_3 - \phi_1)}{(z_3 - z_1)}(z_p - z_1)$$

$$\phi_{24} = \phi_2 + \frac{(\phi_4 - \phi_2)}{(z_4 - z_2)}(z_p - z_2) \qquad \textbf{(3.24)}$$

$$\phi_{57} = \phi_5 + \frac{(\phi_7 - \phi_5)}{(z_7 - z_5)}(z_p - z_5)$$

$$\phi_{68} = \phi_7 + \frac{(\phi_8 - \phi_7)}{(z_8 - z_7)}(z_p - z_7)$$

Followed by,

$$\phi_{1357} = \phi_{13} + \frac{(\phi_{57} - \phi_{13})}{(y_{57} - y_{13})}(y_p - y_{13}) \qquad \textbf{(3.25)}$$

$$\phi_{2468} = \phi_{24} + \frac{(\phi_{68} - \phi_{24})}{(y_{68} - y_{24})}(y_p - y_{24})$$

And,

$$\phi_P = \phi_{1357} + \frac{(\phi_{2468} - \phi_{1357})}{(x_{2468} - x_{1357})}(x_p - x_{1357}) \tag{3.26}$$

Boundaries are dealt with by assuming mirror conditions.

This technique is easy to implement and can be easily reduced to fewer dimensions. Though it suffers some drawbacks in that it relies upon the interpolation of interpolated data, which may increase numerical errors, it is fast. The end result is used to give an indication of solution behaviour and absolute accuracy is not a prime requirement.

3.3.2 Shepard's Interpolation for Transfer of Data between Grids

Shepard's interpolation is a distance weighted scheme. Its principal advantages are that the location of the point at which a new value is required relative to the grid does not have to be calculated, and that it is safe, in that it will not produce extreme values.

Shen et al. (1993) discuss Shepard's method for interpolation of scattered data, and a modification of it for use on solution-adaptive grids.

For the function **f** with values $\mathbf{f}_k$ at nodes (x_k,y_k) for k = 1..N, the basic method is expressed as

$$F(x,y) = \frac{\sum_{k=1}^{N} W_k(x,y) Q_k(x,y)}{\sum_{k=1}^{N} W_k(x,y)} \tag{3.27}$$

With

$$W_k = \frac{1}{d_k^{\mu}}, \qquad Q_k(x,y) = f_k \tag{3.28}$$

This becomes

$$F(x,y) = \begin{cases} f_k & (x,y) = (x_k,y_k) \textit{ for some } k \\ \dfrac{\sum_{k=1}^{N} \dfrac{f_k}{d_k^{\mu}}}{\sum_{k=1}^{N} \dfrac{1}{d_k^{\mu}}} & \textit{otherwise} \end{cases} \tag{3.29}$$

Where d_k is the distance from the function point (x_k,y_k) to (x,y)

$$d_k = \left((x-x_k)^2 + (y-y_k)^2 + (z-z_k)^2\right)^{\frac{1}{2}} \tag{3.30}$$

Setting $\mu=2$ was recommended in Shen et al. (1993) and means that the square root need no longer be evaluated.

Though this is not the form recommended in the paper for solution adaptive grids it is easy to adapt this method to allow it to be used by carefully controlling the number of data points used to evaluate the new point. In the current work the default number of points used is three in each coordinate direction up to the limits imposed by boundaries around the original point. The new point is unlikely to have moved out of this region.

3.3.3 Comparison of Shepard's and Multiple 1D Interpolation

The accuracy of Shepard's and Multiple 1D Interpolation are compared by using them to calculate values on an offset 2D grid based on a known function. The original grid is a uniform square with 21x21 points and the function is given in equation 3.31

$$function = \frac{\sin(r)}{r} \tag{3.31}$$

$$r = 20\sqrt{\left((x-0.5)^2 + (y-0.5)^2\right)}$$

The offset grid was shifted by a fixed value up to half a cell length in X and Y from the original grid and contained 20x20 points so that it did not overlap the dimensions of the original grid. The total error is calculated by adding up the sums of the differences between the correct and interpolated values divided by the correct values. The total error against the shift is given in figure 3.18.

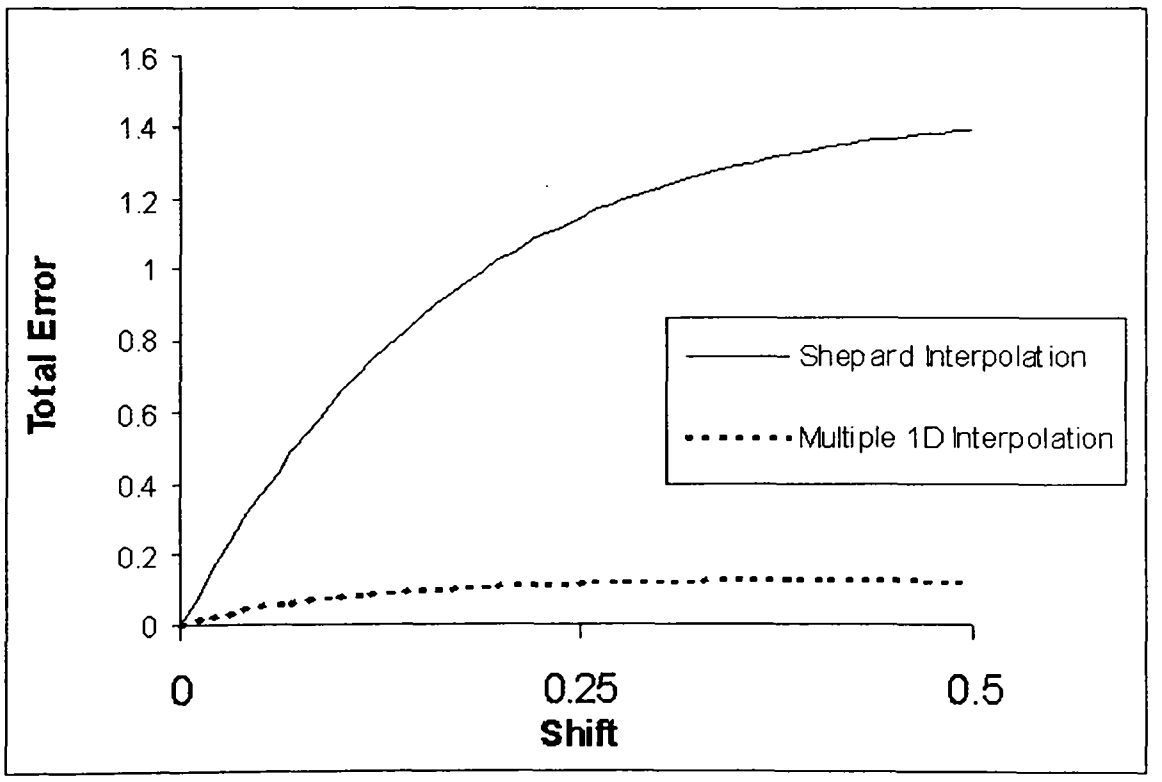


Figure 3.18 Interpolation error against shift

A plot of the correct function along the centre of the grid with the interpolated values for comparison is given in figure 3.19.

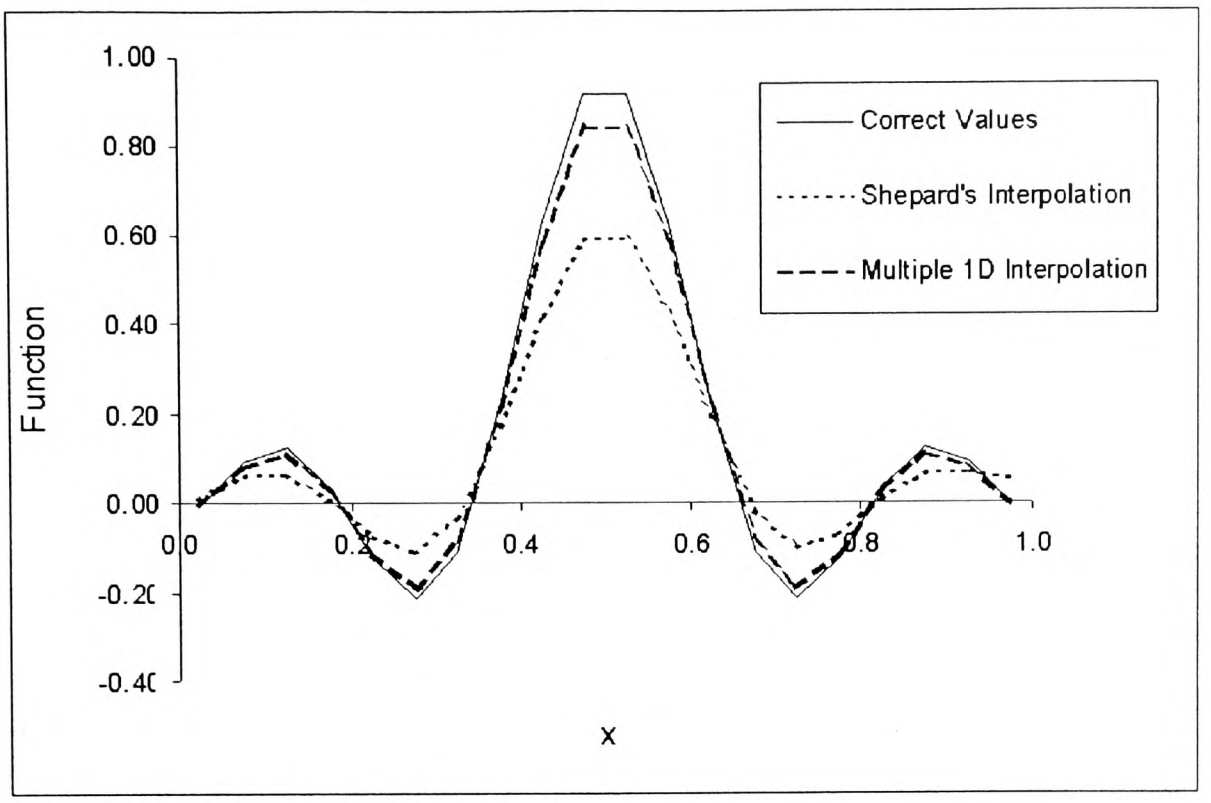


Figure 3.19 Correct and Interpolated Function Values

These graphs show that the linear interpolation scheme is more accurate. It is also in the order of five times faster than the Shepard's scheme for this grid.

Shepard's interpolation is used in the main algorithm to interpolate values to points whose relative location to known function values is not accurately known. Multiple 1D interpolation requires that the interpolation point can be accurately located in the grid as the new value depends on its direct known function neighbours and is therefore used where they are inherent to the problem. Both schemes are robust and smooth the given function.

3.3.4 Volume and Area Weighting

Volume and area weighting techniques, though not used in the current work were investigated as an alternative, particularly to the multiple 1D interpolation discussed in section 3.3.1. All of the following techniques weigh the influence of the surrounding cells on a point by using a function of the areas or volumes of those cells. The main restriction on the use of these techniques is the time spent on calculating those areas and volumes. If these values can be extracted from the underlying CFD code these methods might be more attractive, though that would tie the adaption

algorithm closer to the code. The multiple 1D technique appears to be much faster and at least as accurate.

Jacquotte and Coussement (1992) cover a range of different interpolation techniques, with a particular emphasis on their use in grid adaption and recovery of values at cell vertices from cell centres. They suggest the following approximations of linear interpolation, using the areas or volumes of the surrounding cells as a weight:-

$$\phi_M = \frac{\sum_{i=1,4} A_i^{-1} \phi_i}{\sum_{i=1,4} A_i^{-1}} \tag{3.32}$$

For areas, or

$$\phi_M = \frac{\sum_{i=1,8} V_i^{-1} \phi_i}{\sum_{i=1,8} V_i^{-1}} \tag{3.33}$$

For volumes.

Or alternatively the following approximation to L^2- least square minimisation:-

$$\phi_M = \frac{\sum_{i=1,4} A_i \phi_i}{\sum_{i=1,4} A_i} \tag{3.34}$$

For areas, or

$$\phi_M = \frac{\sum_{i=1,8} V_i \phi_i}{\sum_{i=1,8} V_i} \tag{3.35}$$

For volumes.

The following expressions are given to determine the values at boundary points, based on linear interpolation, where N is a point that butts onto cell 1 and 4, and P is at the corner of cell 1 (figure 3.20).

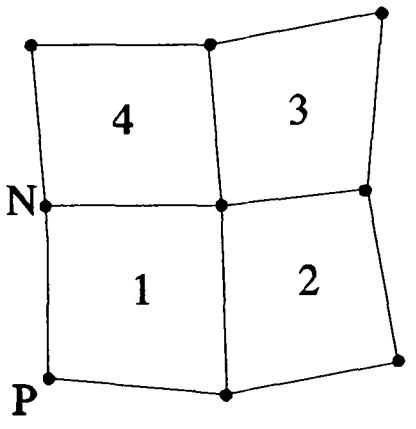


Figure 3.20 Boundary Interpolation

$$\phi_N = \frac{(A_1^{-1} + 2A_2^{-1})\phi_1 - A_2^{-1}\phi_2 + (2A_4^{-1} + A_3^{-1})\phi_3 - A_4^{-1}\phi_4}{\sum_{i=1,4} A_i^{-1}} \qquad (3.36)$$

$$\phi_P = \frac{(A_1^{-1} + 2A_2^{-1} + 2A_3^{-1} + 2A_4^{-1})\phi_1 - A_2^{-1}\phi_2 - A_3^{-1}\phi_3 - A_4^{-1}\phi_4}{\sum_{i=1,4} A_i^{-1}} \qquad (3.37)$$

Jacquotte and Coussement (1992) do not give other routines, but using the same techniques it would not be difficult to develop them.

Further techniques, including a cell search algorithm, given in the same paper could be adapted to interpolating new values for successive adaptions, or evaluations of new values in PHOENICS.

3.4 Cell Integrity Control

A method is necessary to control minimum cell size, as the LPE equation which drives the grid adaption does not prevent cells from collapsing in its current form. Collapsed grid cells invalidate the grid and cause the solution to diverge or give unpredictable

results. In the current work a series of geometrical tests are used to prevent grid lines from crossing over after each iteration of the LPE equation. An alternative option might be to solve the LPE equation using some form of whole field solver or tridiagonal matrix algorithm, with suitable modifications to the source term rather than the point by point solution used here. It is possible to weight the LPE equation such that a corrupted grid is extremely unlikely, but this may prevent effective grid movement.

The method used to check against grid cross over involves checking if the calculated movement vector applied to an original grid point overshoots either of its neighbouring points in one dimension, any line linking any two neighbouring, adjacent points in two dimensions, or any surface defined by four neighbouring, adjacent points in three dimensions (figure 3.21).

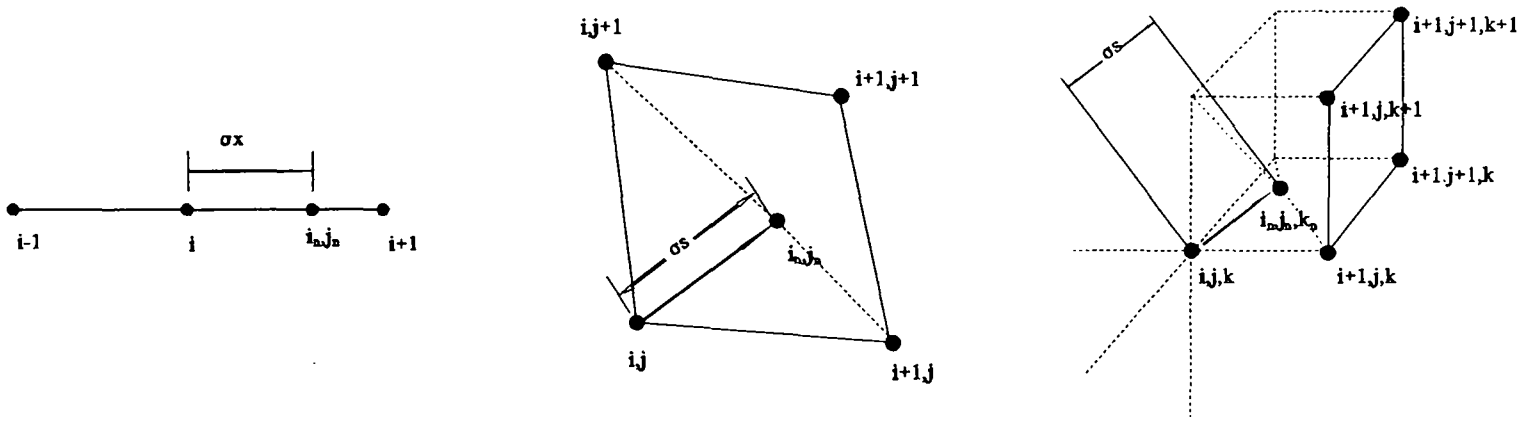


Figure 3.21 1D Movement, 2D Movement and 3D Movement

The points against which the moved point is checked are modified using a clearance vector. The clearance vector is evaluated by determining the smallest cell clearance in each coordinate direction and multiplying by a user defined ratio. Cell clearance in each co-ordinate direction is calculated as the physical distance between points along the corresponding grid lines in computational space (figure 3.22).

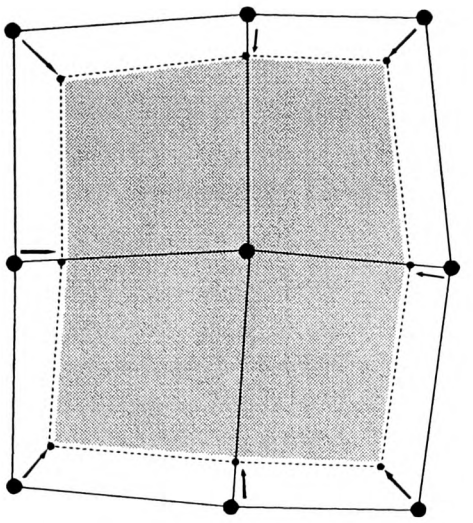


Figure 3.22 Possible Movement in 2D

If the movement vector added to the clearance vector goes past a line that connects any two surrounding points in 2D, or any face connecting four points in 3D then the vector is scaled back until it coincides with the line or surface (figure 3.23).

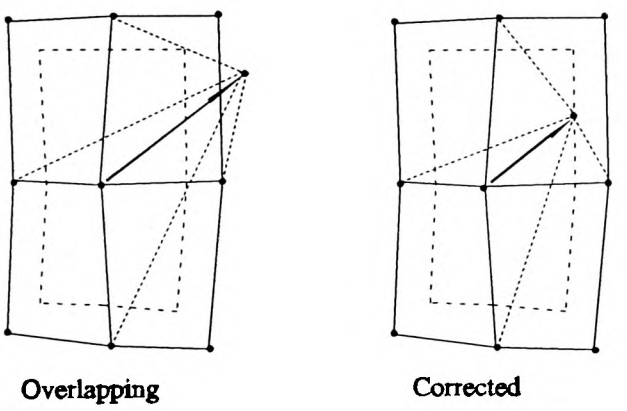


Figure 3.23 Position Correction

Each grid node location is updated after its movement vector has been checked. The movement vector is checked against the updated grid with information from the movement of the preceding grid nodes.

To reduce the possible amount of work, the length of the movement vector is first checked against the distance from the moving point to all of its neighbouring points to test for the chance of an overlap.

The geometrical tests were based on methods in Bowyer and Woodwark (1983).

3.4.1 2D Grids

The movement vector is checked against the eight lines by using the equation of each line relative to the original position of the point checked.

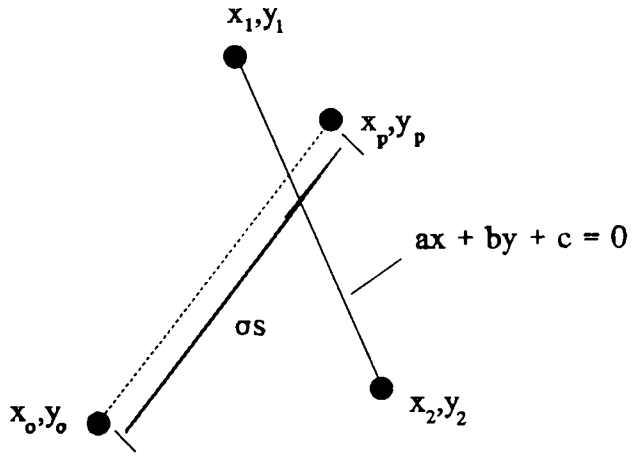


Figure 3.24 Problem Definition

The equation of the line in figure 3.24 is simply,

$$ax + by + c = 0 \tag{3.38}$$

Where,

$$\begin{aligned} a &= y_1 - y_2 \\ b &= x_2 - x_1 \\ c &= x_1 y_2 - x_2 y_1 \end{aligned} \tag{3.39}$$

If the coordinates were normalised the distance of the point from the line would be,

$$s_P = a x_P + b y_P + c \tag{3.40}$$

However, the distance is not important, only the side on which the new point lies, and all that is important from equation 3.40 is the sign of the result. If the sign of the distance s_p is different from c in equation 3.39, then the point has moved too far and must be scaled back.

If the grid point does overlap then its movement vector is scaled back by finding the intersection of a line from the original grid point position to its new position and the line it crosses. This is achieved by writing the grid movement line in the form of a parametric equation, relative to its original position, which gives,

$$\begin{aligned} x &= x_P t \\ y &= y_P t \end{aligned} \tag{3.41}$$

The correct parameter at the intersection is,

$$t = \frac{-c}{a x_P + b y_P} \tag{3.42}$$

3.4.2 3D Grids

The easiest entity to check the movement vector against in 3D is a plane defined by three points. Each four point face **1234** (figure 3.25) is divided into four such planes **12C**, **23C**, **34C**, **14C**, (figure 3.26) by finding a centre point **C**. **C** is defined by taking the average of the position vectors of the four known points.

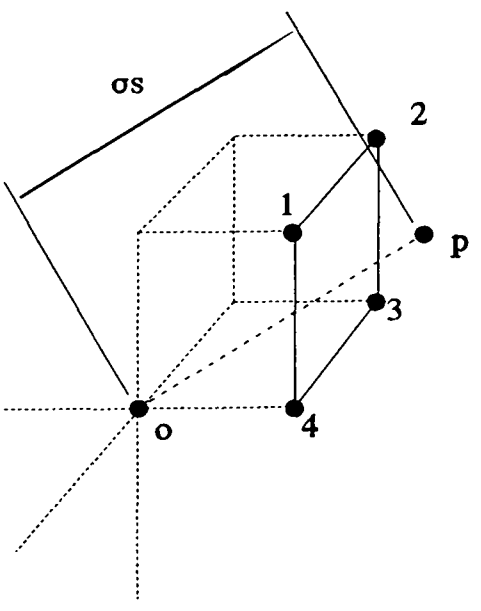


Figure 3.25 Face Definition

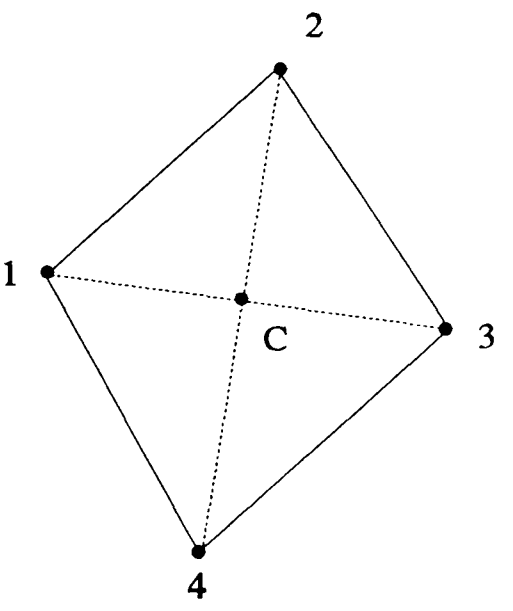


Figure 3.26 Splitting of Face

The point is checked against all four planes for all 24 faces, though this only takes place if a grid overlap is likely. This is achieved by calculating the equations of the planes relative to the original position of the point, allowing for the minimum cell clearance.

The equation of an arbitary plane *JKL* (figure 3.27), is,

$$ax + by + cz + d = 0 \tag{3.43}$$

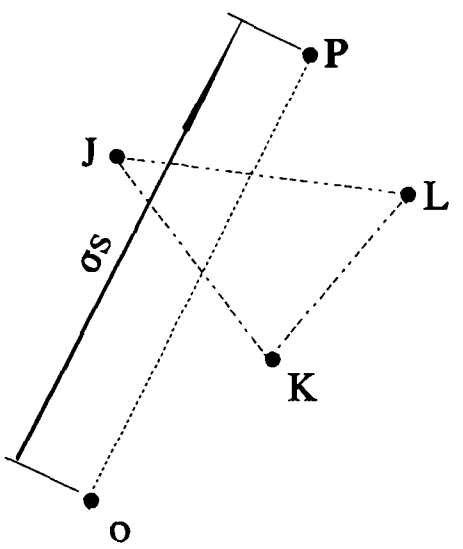


Figure 3.27 Plane Definition

As a vector from point *J* to any other which lies on the plane must be perpendicular to a vector normal to the plane's surface, the vector product of *JK* and *JL*, the equation can be found using the expression

$$\vec{JP} \cdot \left(\vec{JK} \times \vec{JL} \right) = 0 \tag{3.44}$$

Which can be written as the determinant of the matrix,

$$\begin{vmatrix} x - x_J & y - y_J & z - z_J \\ x_K - x_J & y_K - y_J & z_K - z_J \\ x_L - x_J & y_L - y_J & z_L - z_J \end{vmatrix} = 0 \tag{3.45}$$

which gives,

$$
\begin{aligned}
a &= (y_K - y_J)(z_L - z_J) - (z_K - z_J)(y_L - y_J) \\
b &= (z_K - z_J)(x_L - x_J) - (x_K - x_J)(z_L - z_J) \\
c &= (x_K - x_J)(y_L - y_J) - (y_K - y_J)(x_L - x_J) \\
d &= -(a x_K + b y_K + c z_K)
\end{aligned}
\tag{3.46}
$$

In the same manner as for the two dimensional problem the distance of the point from the surface is found using

$$
s_P = a x_P + b y_P + c z_P + d \tag{3.47}
$$

If the sign of the distance s_p is different from the constant d then the point is the wrong side of the plane and the magnitude of the movement must be reduced. This is achieved by determining the intersection of a line from the original grid point to its new calculated position and the plane.

If the line is expressed in its parametric form

$$
\begin{aligned}
x &= x_P t \\
y &= y_P t \\
z &= z_P t
\end{aligned}
\tag{3.48}
$$

Then the correct parameter at the intersection is,

$$
t = \frac{-d}{a x_P + b y_P + c z_P} \tag{3.49}
$$

3.5 Movement Limitation

In the current work a filter is used to screen the grid point movements calculated with the adaption algorithm. Using different values for each co-ordinate direction the movements in each direction can be restricted to a user defined range. The range is determined using scaling factors calculated for the original grid.

Movement greater in magnitude than the upper limit can either be reduced to the upper limit, or to zero. Movement less than the lower limit goes to zero.

This filter is useful in some cases to prevent any movement in one or more directions. It is also useful in situations when the relative speed of some grid points is too fast, as can be the case when the solution is sensitive, or when frequent grid movement is imposed to allow the solution to recover faster between adaptions.

3.6 Cylindrical Polar type Grids

There is a class of two dimensional, axisymmetric, BFC style grids that are generated by distorting polar coordinate grids. These grids are typically used to model bodies of revolution such as flow over a rocket with a circular cross section or pipe flow problems. An example of a polar coordinate grid and the BFC equivalent in PHOENICS (Spalding 1989) is shown in figures 3.28 and 3.29.

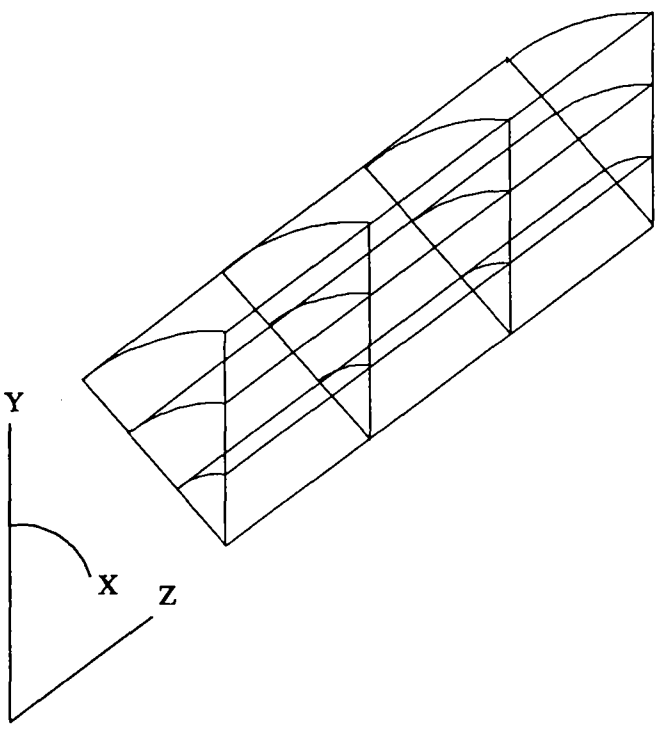


Figure 3.28 Polar Grid

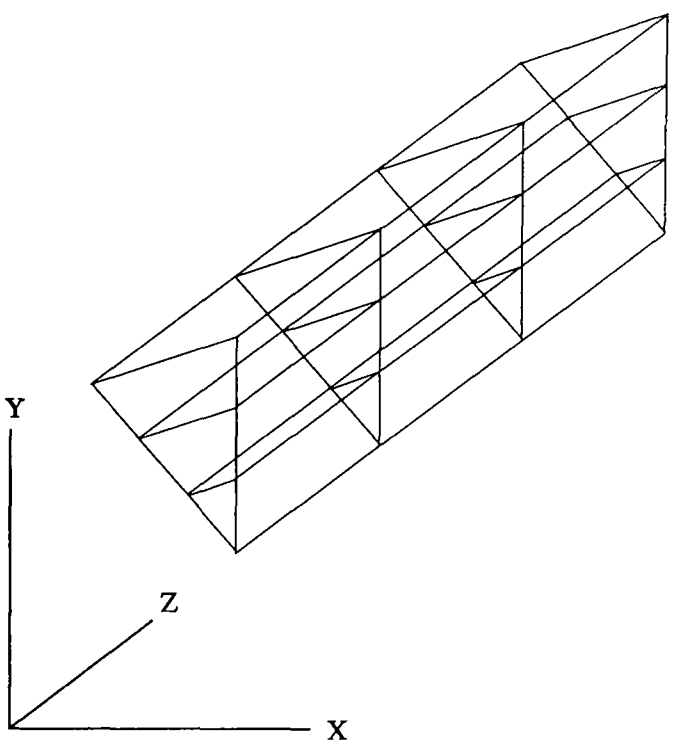


Figure 3.29 BFC Polar Grid

These meshes provide a problem for dynamic adaption. To move points in a purely two dimensional manner invalidates the back surface of the grid. To treat the grid as a purely 3D case creates problems as the front and back surfaces would have to be fitted using the surface modelling technique (section 3.2) and the grid is only one cell thick, so that the partial differential equations depend upon mirror image boundary conditions and identical movement on both faces cannot be guaranteed.

In the current work this problem is resolved by adapting the upright face of the grid only and then rotating it to resolve the back, slanted face (figure 3.30). The point from and the angle through which the front surface is rotated is calculated from the original grid. The movement of the front face is calculated exactly as if it was a 2D problem.

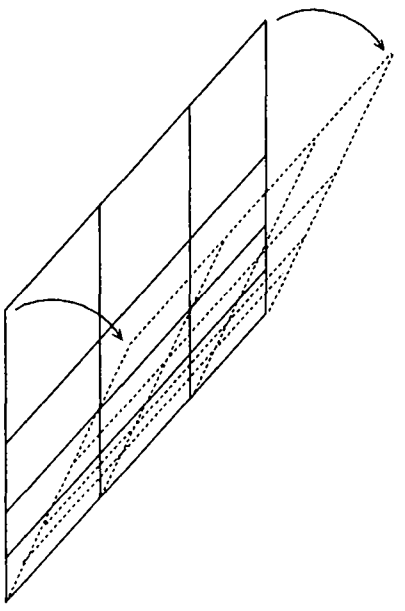


Figure 3.30 Rotating Planes

3.7 Closure

This chapter has described the remainder of the tools not covered in chapter 2 needed to make dynamic grid adaption practical. There are other methods, not part of the present algorithm, available for most of the techniques available here, particularly for curve and surface modelling. Other methods include using gradients to determine parameters, Bézier curves and Coons patches.

CHAPTER 4: ASPECTS OF IMPLEMENTATION

4.1 Introduction

This chapter concerns issues which affect the implementation of grid adaption outside of the main algorithm discussed in chapters 2 and 3. These include the organisation and control of the adaption algorithm and the interface between the adaption code and the CFD code to which it is attached.

The aim of this thesis is to create general purpose grid adaption code that is as independent as possible from the calling CFD code. The adaption algorithm needs to be treated as far as possible as a black box which reads grid and variable data in and sends new grid data out with the minimum of communication. Ideally the user should not have to be an expert in grid adaption, and should therefore be able to use default adaption parameters with a chance of getting a reasonable adapted grid.

Adaption takes place over defined regions within the problem domain. Each region is treated independently. This allows the adaption module to be potentially used for multiblock problems.

The adaption module has been implemented in the commercial CFD code PHOENICS (Spalding 1989), the in house multi physics code PHYSICA (Cross et al. 1995) and independently, to run over simple example problems. These implementations are described in this chapter.

More detailed descriptions of the structure and tasks carried out in the adaption module, with specific references to the PHOENICS implementation, are given in appendices B and C.

4.2 Adaption Organisation

The Adaption code is organised into two modules which concern all tasks which only

need to take place once, and those which take place many times respectively. The tasks in the first module, which is known as the adaption initialisation module, cover the determination and calculation of grid geometry. The tasks in the second module, known as the adaption calculation module, cover the main adaption algorithm. Adaption control is split between these two modules to reduce their dependence on each other.

The adaption initialisation module is called only once for each problem. The adaption calculation module is called at intervals in the execution of the calling CFD code solver defined through its sweep or iteration number.

Independence of the adaption modules from the calling CFD code is primarily achieved through the use of an independent data structure and a separate solver for the LPE equations. As far as possible all work done by the adaption modules is done in their own space, with all information necessary for adaption outside of the flow variables solved by the CFD code held outside it. All necessary grid data is read in once in adaption initialisation and modified internally at each call to the adaption calculation module.

4.2.1 Adaption Initialisation

Adaption initialisation concerns the definition of regions of the main problem domain over which adaption needs to take place and features within those subregions that are needed to maintain their geometry. Features include surfaces and curves which are defined using bicubic patches and cubic splines respectively, as described in section 3.2 and blanks. Blanks are subregions where no movement is permitted through the main adaption algorithm. They allow for obstructions without breaking variable data between two separate regions and the implementation of any grid movement that needs to take place independently of the main algorithm such as the fluid stress interaction discussed in chapter 7.

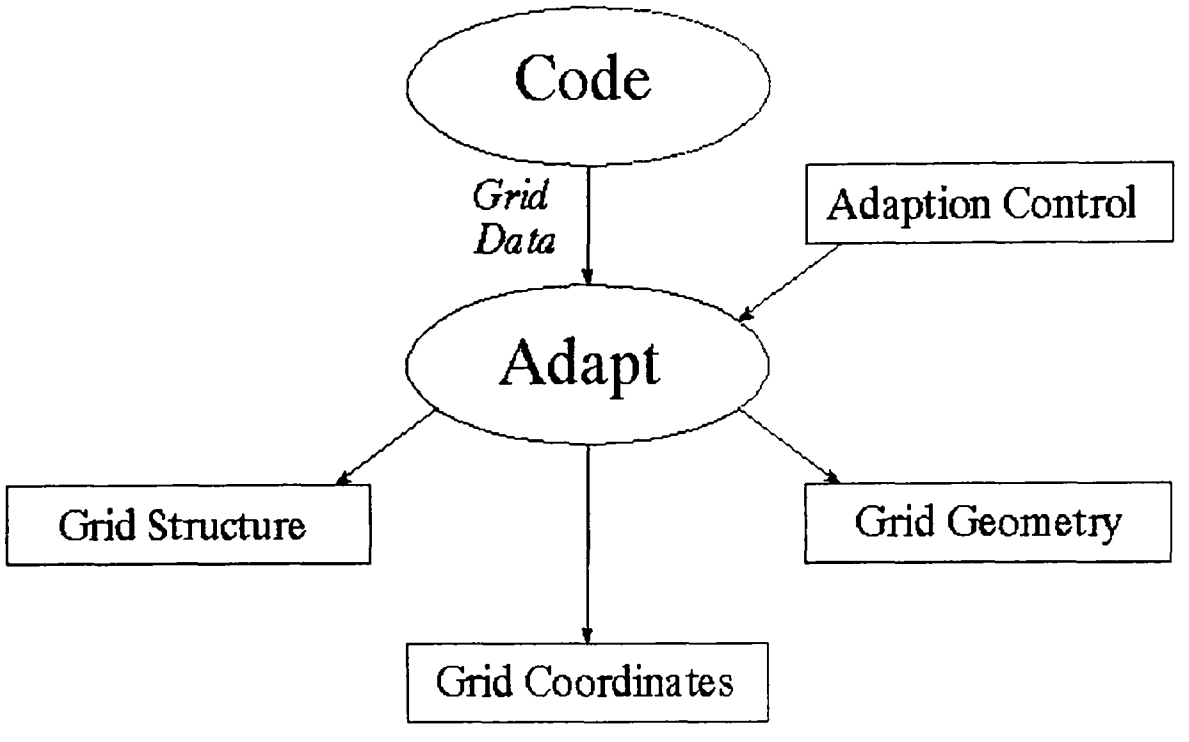


Figure 4.1 Data flow in adaption initialisation

This module generates grid data files that contain the coordinates of the regions to be adapted, parametric data to allow the recovery of defined surfaces and curves, and control function data to enable the original grid distribution to influence the main adaption algorithm as described in section 2.5.

The original grid is used to generate all data used in the adaption module. This allows it to be independent of grid generation software, but means that the definition of geometrical features in the adapted grid can only ever be as good as in the original grid.

4.2.1.1 Initialisation Control

Adaption initialisation is controlled though the use of a single input file that defines multiple adaption regions, surfaces, curves and blanks on the original grid. Control over surface and curve fitting is included in the form of tolerances that influence surface and curve cutting (section 3.2.3).

4.2.2 Adaption Calculation

Adaption calculation consists of all parts of the adaption algorithm not covered by the adaption initialisation module. This is principally the generation of a new grid based on the grid coordinates and geometry of the old grid and a given data set, and is described in figure 4.2.

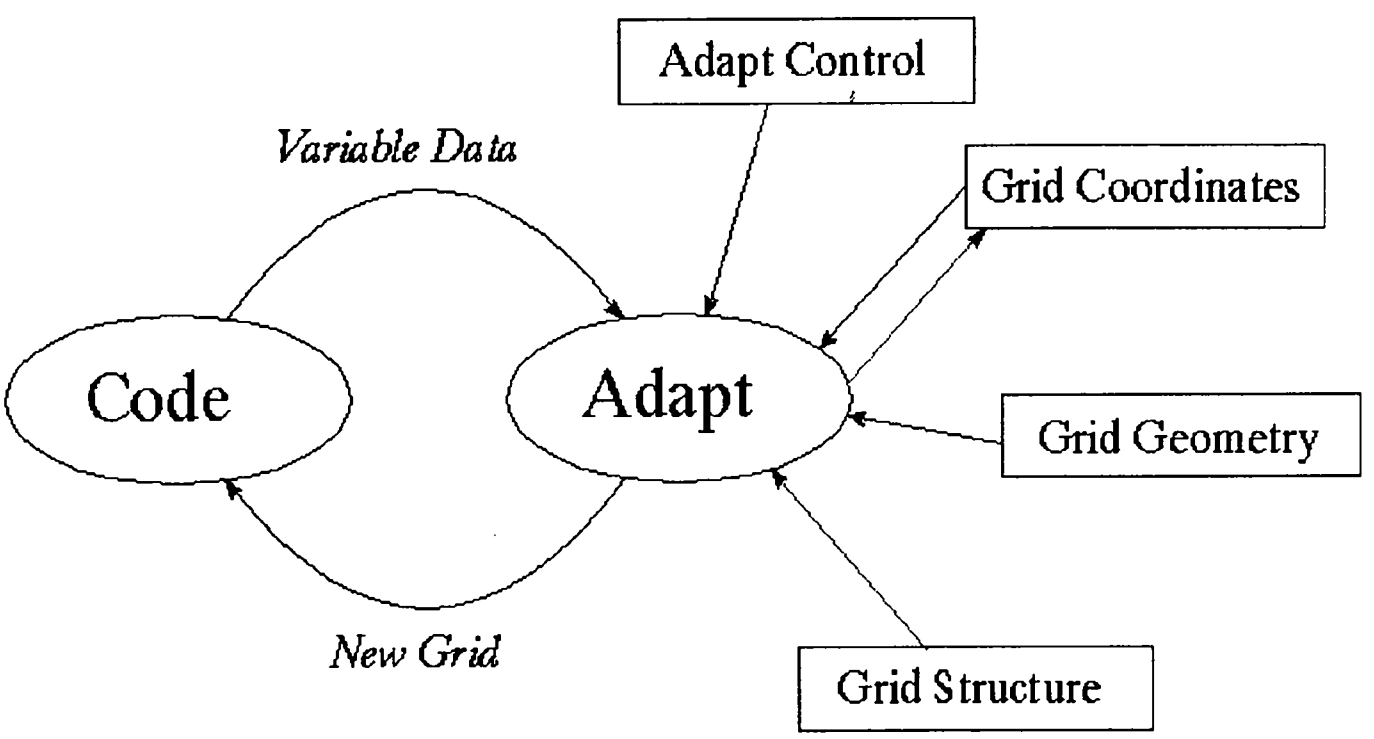


Figure 4.2 Data flow for adaption execution

The tasks in the adaption calculation module are split into three parts, calculation initialisation, main calculation and bookkeeping.

4.2.2.1 Calculation Initialisation

Calculation initialisation involves the reading of an adaption control file and checking it against the flow problem definition.

4.2.2.2 Main Calculation

The main calculation concerns the generation of a new grid. The main tasks are summarised in figure 4.3.

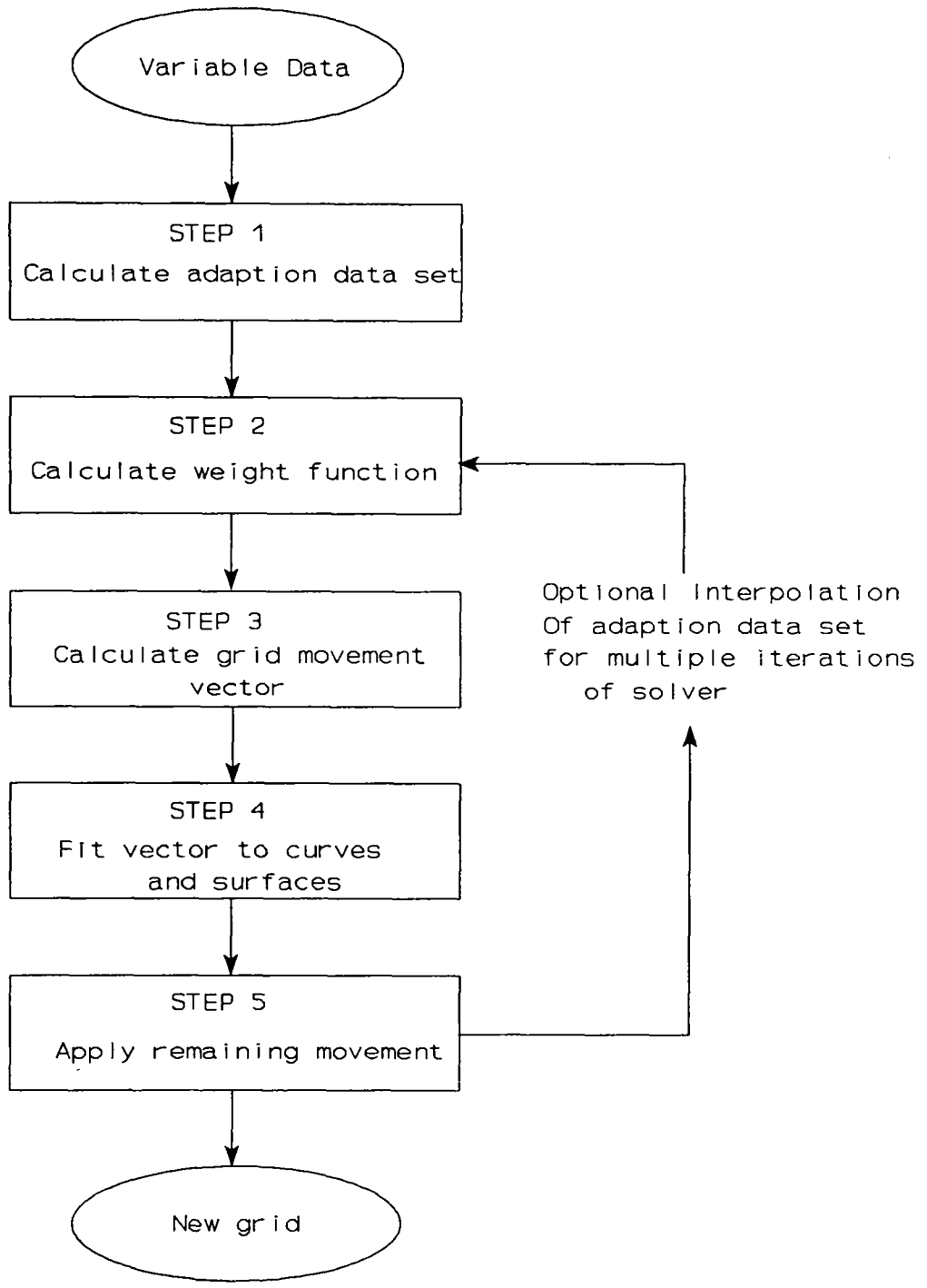


Figure 4.3 Main Calculation Process Flow

The adaption data set generated in step 1 used for adaption can be a function of many flow variables and external data sets. Smoothing can also be carried out on it prior to the calculation of the weight function (section 2.8.3).

The weight functions in step 2 are described in section 2.8. The weight functions may be smoothed using techniques described in section 2.8.3.

The grid movement vector in step 3 is calculated using the LPE algorithm described in chapter 2.

In step 4 grid movement vectors are first applied to all geometric curves and surfaces defined in the adaption domain as described in sections 3.2.1.3 and 3.2.2.3 respectively. Movement is checked to prevent grid overlapping using the technique described in section 3.4.

In step 5 the remaining grid movement vectors are applied.

It is possible to loop between steps 2 and 5 through interpolation of the adaption data set between the old and new grids. This allows a relaxation to be used in the scheme used to solve the adaption algorithm.

4.2.2.3 Bookkeeping

Bookkeeping involves the calculation of total grid movement, removal of extra adaption files and forcing the calling code to dump a final grid file.

4.2.2.4 Adaption Calculation Control

The primary calculation control is through a file that controls when the adaption module is called and all parameters that influence the main algorithm, the weight function and the interface between the module and the calling code on a region by region basis.

This control is supplemented by a number of automatic tools which cover scaling of internal tolerances to the grid and flow data, and monitoring of grid and flow solver convergence. The adaption data set used to generate the weight function is nondimensionalised to allow for consistent adaption parameters to be used over a wide range of flow problems with little modification.

4.3 Interface between CFD Code and Adaption Modules

The interface primarily concerns CFD code specific coding in the adaption modules.

It is split into three areas covering the calling of the adaption module and its control, data transfer, and several miscellaneous tasks.

The interface also covers the interpolation of the old solution onto the new grid to reduce the impact of rapid grid movement on the solution process. Implementation of interpolation within the adaption module is not desirable from the software engineering point of view as it requires much more data transfer than is needed by adaption alone. This leads to reduced data protection and more CFD code specific code. An alternative is to treat the flow problem as transient and to rely on grid velocity terms within the CFD code, if available, to conserve the flow variables. Tools to carry out interpolation are discussed in section 3.3.2.

4.3.1 Calling the Adaption Modules

The adaption initialisation module can be called any time after the initial grid is generated and before the solver of the calling CFD code is started. Its activation is controlled through either a command coded into the calling CFD code or through a switch in the adaption initialisation control file.

The adaption calculation module is called at the end of each iteration or sweep of the solver in the calling CFD code. Its activation is controlled through either a command coded into the calling CFD code or a switch in the adaption calculation control file, and by information passed through to the module from the CFD code. This information includes the current sweep number and any flags indicating the convergence or failure of the CFD code solver to allow the adaption module to execute bookkeeping. The sweep number is used to determine activation of the adaption module by being checked against criteria defined in the adaption calculation control file. The first call to the module always activates calculation initialisation.

4.3.2 Data Transfer

In adaption initialisation data transfer consists of reading the original grid in from the calling CFD code together with any pointers required to interpret the data.

In adaption calculation data transfer consists of reading in flow variable data and pointers and passing out the adapted grid. In addition it may be desirable to write adaption data back to the CFD code for visualisation purposes in post processing. Such data covers grid movement, grid quality measures and the weight function. This requires specific storage space to be set aside in the problem definition in the calling CFD code and access to pointers to that storage from within the adaption module.

Another aspect of data transfer are the filters that are needed inside the adaption module to translate data into the local adaption format. In the adaption modules all work is done on grid nodes. If flow variable data is held in cell centres, or cell faces in the case of velocity components on staggered grids, then it needs to be interpolated to grid nodes. This is achieved using linear interpolation as described in section 3.3.1.

4.3.3 Miscellaneous Tasks

The interface can also contain the following components:-

- Code to force recalculation of internal CFD code geometrical constants after adaption has taken place.
- Code to force a dump of final adapted grid to file. It may also be advantageous to dump general data output files at other points in the solution process to provide more information about the course of the adapted solution.
- Code to output information at run time about adaption. This may involve output of a measurement of movement, or, if some form of run time graphical user interface (GUI) is available, a display of parts of the moving grid.
- Code to output macro files linked to flow visualisation tools to automatically mark out areas of adaption.

4.4 Implementation Examples

The CFD flow solvers to which adaption is most likely to be applied use iterative schemes to solve complex systems of partial differential equations that represent the physics of 'real' problems. The general format of a CFD code that uses an iterative solver with adaption is shown in figure 4.4.

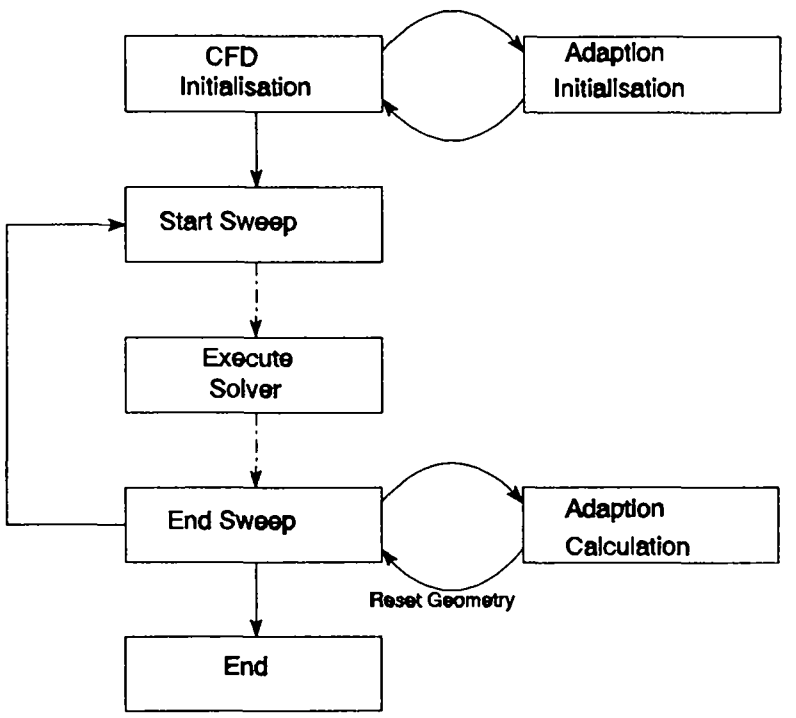


Figure 4.4 Flow chart for adaption in a general iterative CFD code

4.4.1 PHOENICS Implementation

PHOENICS (Spalding 1989) is a well established general purpose finite volume code used to simulate fluid flow, heat transfer and related phenomena. It has the ability to use cartesian, polar co-ordinate and curvilinear (body fitted co-ordinate ,or BFC) grids. BFC grids can be significantly non orthogonal and are therefore the only ones considered for adaption by grid movement. A non orthogonal solver is available if the grid is poor, though its use involves a higher computational cost than the standard solver.

PHOENICS uses a staggered grid for the calculation of the components of flow velocity at cell boundaries. Other flow variables are calculated at cell centres.

The majority of the test cases in the current work beyond those concerned with fitting the grid to simple functions have been run using PHOENICS versions 1.6, 2.0 and 2.1. Most of the development of the adaption algorithm has been achieved with PHOENICS as the calling CFD code.

PHOENICS is divided into two main programs, a preprocessor called SATELLITE and a solver called EARTH. The adaption initialisation module is linked to SATELLITE and the calculation module is linked to EARTH.

The data interface is simplified by using the same format for data storage within the adaption modules as within PHOENICS. Interpolation of variable data between old and adapted grids is implemented but does not completely prevent the disruption to the solver that the grid movement causes.

For convenience the two adaption control files can be incorporated into the PHOENICS control file Q1 as comments.

PHOENICS has a run time GUI. This has been manipulated to plot grid data to the screen to allow visualisation of the grid movement.

4.4.2 PHYSICA Implementation

PHYSICA (Cross et al 1995) is a multiphysics unstructured code under development at the University of Greenwich. It is designed to be a framework for multi physics continuum mechanics modelling and involves the use of finite volume techniques to solve coupled flow and stress problems. A single test case has been run on a grid that can be treated as structured for the purposes of adaption (section 5.3.2).

PHYSICA stores flow variables at cell centres and uses Rhie and Chow (1983)

interpolation to determine convective fluxes at cell boundaries. This is an important difference from PHOENICS. As PHYSICA is also able to solve over irregular shaped cells it is generally less sensitive to grid quality than PHOENICS.

PHYSICA consists of one main executable to carry out all tasks that are split between SATELLITE and EARTH in PHOENICS so that the structure of the code is the same as that given in figure 4.4. The adaption modules are accessed through PHYSICA using the built in user module.

The two main problems in transferring the adaption modules from PHOENICS to PHYSICA are the interface and the need to use a structured style grid in any sub regions where adaption is applied if the current code is to be reused. Implementation of an unstructured adaptive grid technique using the same method is beyond the scope of this thesis, though the neural net adaption technique discussed in chapter 8 is unstructured.

As the adaption technique is designed for structured grids it is necessary to define the grid in PHYSICA using nodes that are in structured form. This means that hexahedral elements only can be used in regions of adaption and that they be organised in the same manner as a structured grid. Extra information is required on the dimensions of the 'structured' part of the grid as this is not stored naturally in PHYSICA.

4.4.3 Independent Implementation

The independent implementation is used to adapt grids to fixed functions of a type often used to demonstrate features of grid adaption (Biswas et al 1993, Bennett 1991, Dvinsky 1991, Acharya and Moukalled 1990, Anderson 1990, 1987, 1987, Arney and Flaherty 1990, 1986, Lee and Loellbach 1989, Carey et al 1988, Matsuno and Dwyer 1988, Holcomb and Hindman 1984, Greenburg 1983, and Brackbill and Saltzman 1982).

The implementation is split into two separate parts, an initialisation program and a

calculation program. The implementation depends mainly upon the adaption modules already implemented within PHOENICS with additional code to generate the grids used in the initialisation phase. In the calculation phase the main flow solver is replaced with simple routines to generate the variables used for adaption. As there is no need to leave the adaption module during run time the data files used to store temporary information are redundant. However the grid is output at regular intervals in a format suitable for viewing using PHOENICS visualisation software, or using a built in GUI.

4.5 Closure

This chapter has described general issues in the organisation and implementation of adaption. The adaption algorithm has been implemented successfully in the CFD codes PHOENICS, PHYSICA and independently, though with some limitations in PHYSICA.

A more detailed discussion of the implementation in PHOENICS is given in appenicies B and C.

CHAPTER 5: 2D VERIFICATION TESTS

5.1 Introduction

The aim of this chapter is to present examples of the use of the adaptive algorithm on a range of two dimensional cases including both simple mathematical functions and complex heat and fluid flow problems.

The improvements in results due to the use of dynamically adapted grids can be measured by comparing results obtained using coarse meshes, coarse meshes with adaption, fine meshes and experimental results or high quality numerical results. In the more complex cases such as the 10 degree supersonic intake different grid resolutions are not available but the adapted grid is shown to offer improved results.

The choice of the initial mesh has to be careful. If the initial number of cells is too small then though the mesh will move well the resolution of the grid will not have improved enough to significantly better the results. If the number of cells is too high then little movement will take place and adaption will not improve the results

5.2 Static Cases

The following cases were generated using the independent implementation of the adaptive code. In each case the grids were adapted against a static function to demonstrate how the LPE method will work in particular circumstances. Most of the grids are compared with published results.

With some weights the LPE equation may be unstable in that the grid will converge up to a point then diverge. In these cases the adaption has been stopped at the point where the grid starts to diverge. Otherwise the problems have been run to convergence. Most of the movement occurs in the first five or ten iterations of the adaption solver.

As all grids in the following section are initially uniform, the Poisson weight in the LPE equation is set to zero. This is because the Poisson control functions which shape the grid will all be zero and the Poisson term will be the same as the Laplace term.

In most of the following cases the functions involve an instantaneous change in value at certain positions in the grid and no change elsewhere. To propagate the effects of the functions out into the rest of the grid and prevent unpredictable results where a point is on the border of the function smoothing is used on either the variable used internally to calculate the weight functions in the LPE equation, or on the weight functions themselves.

It is important to realise that there is not one single solution to each problem. It depends upon what the user wants. To demonstrate this several results are produced for the first three cases.

5.2.1 Attraction to a Line

This problem is used to demonstrate basic grid movement. The grid is attracted to the centre line by using the simple step function,

$$\phi = \begin{cases} 0 & x<0.5 \\ 1 & x\geq 0.5 \end{cases} \qquad \textbf{(5.1)}$$

This problem is very similar to the attraction to a line case used in Divinsky (1991).

A 10x10 grid is used, covering a unit square domain.

The problem is run with three separate sets of parameters to demonstrate their effect upon the final grid. In all cases the weight function used is

xit

$$w = 1 + \phi_s + 0.01 \frac{|\phi_{ss}|}{\left(1 + (\phi_s)^2\right)^{\frac{3}{2}}} \qquad \textbf{(5.2)}$$

Attraction to a Line						
	λ_L	λ_E	Power Law Factor	Number of Moves	Total X move %	Total Y move %
Case 1A	1	1	2	20	63.4	0
Case 1B	1	3	3	16	133.5	0
Case 1C	1	10	3	20	165.8	0

Where λ_L is the weight on the Laplace term in the LPE equation, and λ_E is the weight on the Equidistribution term. The power law factor governs the slope of the curve of the weight distribution (See 2.10.3.3). Movement is based on the sum of the movement of the individual points compared with the sum of the initial distances between all individual points and their neighbours in the appropriate coordinate direction in the original grid.

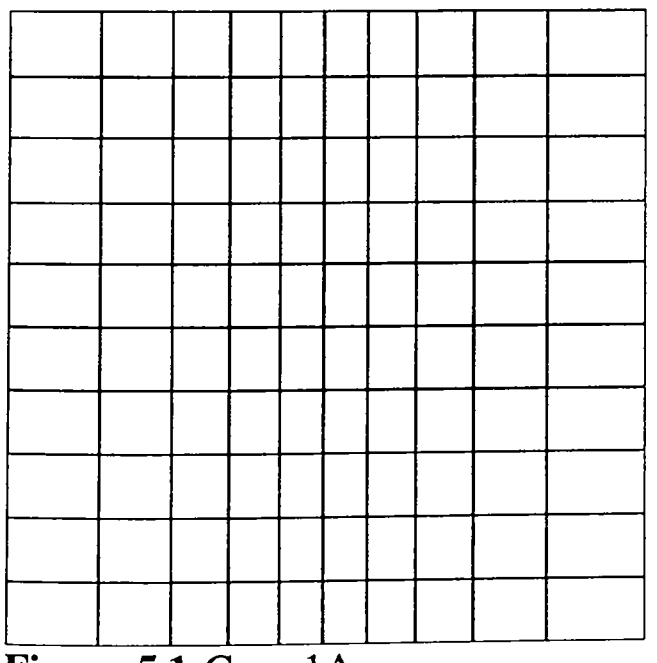


Figure 5.1 Case 1A

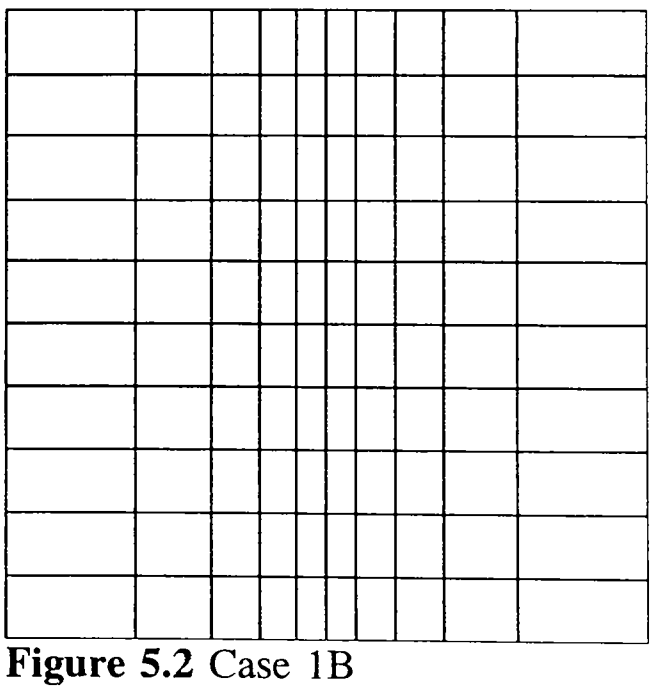


Figure 5.2 Case 1B

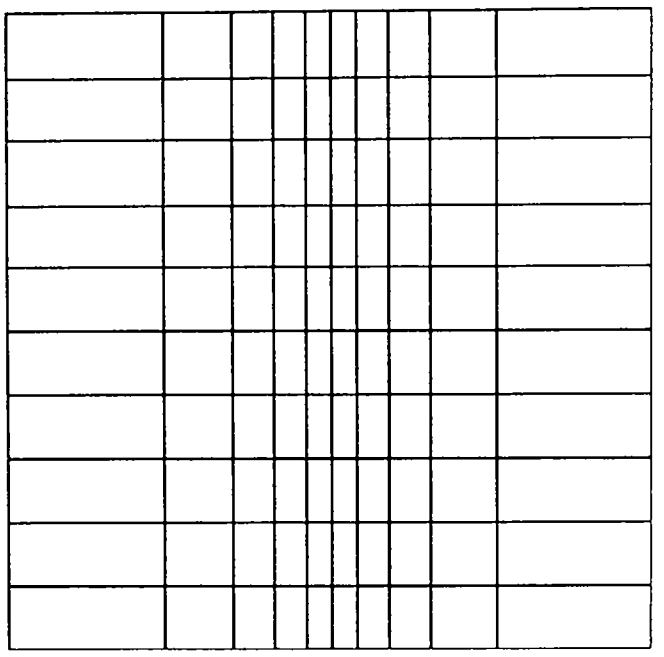


Figure 5.3 Case 1C

Movement is uniform across domain in all cases

The equidistribution term controls how much clustering occurs around the line. The power law controls the change in distribution of cells towards the centre line. The three cases have an increasing equidistribution weight.

5.2.2 Angled 'Shock' Like Function

This problem is used to demonstrate how the LPE algorithm will adapt a grid to a stylised 'shock' like function that has a similar shape to those which occur within numerical models of transonic and supersonic flows. The function is a graded, angled step, where

$$\phi = \begin{cases} 0 & 0.5(y+14) < x \leq 24 \\ 0.5(7-x)+0.25y & 0.5(y+10) < x \leq 0.5(y+14) \\ 1 & 0 \leq x \leq 0.5(y+10) \end{cases} \qquad \textbf{(5.3)}$$

This function is used by Anderson (1987a) (figure 5.8, figure 5.9) (1990) (figure 5.10)

A 24x24 grid is used. The domain measures 24x24 units, each cell initially being a unit square.

The problem is run with four power law factors to demonstrate its effect upon the final grid. In all cases the weight function used is

$$w = 1 + 5\phi_s + 0.05 \frac{|\phi_{ss}|}{\left(1 + (\phi_s)^2\right)^{\frac{3}{2}}} \tag{5.4}$$

The weight parameters are designed to accentuate the gradients of the function.

Angled 'Shock' Like Function						
	λ_L	λ_E	Power Law Factor	Number of Moves	Total X move %	Total Y move %
Case 2A	1	2	3	50	102.0	21.6
Case 2B	1	2	2	50	102.9	24.5
Case 2C	1	2	1.5	50	98.7	25.6
Case 2D	1	2	No Power Law	40	69.7	9.6

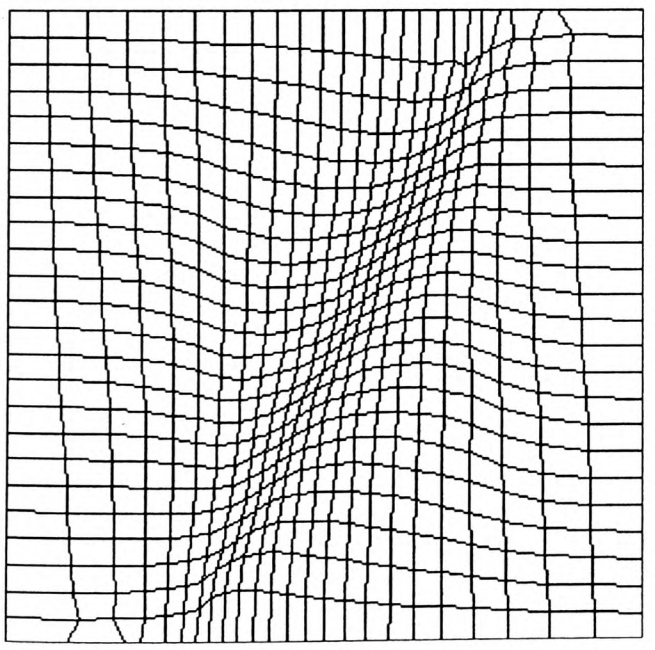


Figure 5.4 Case 2A

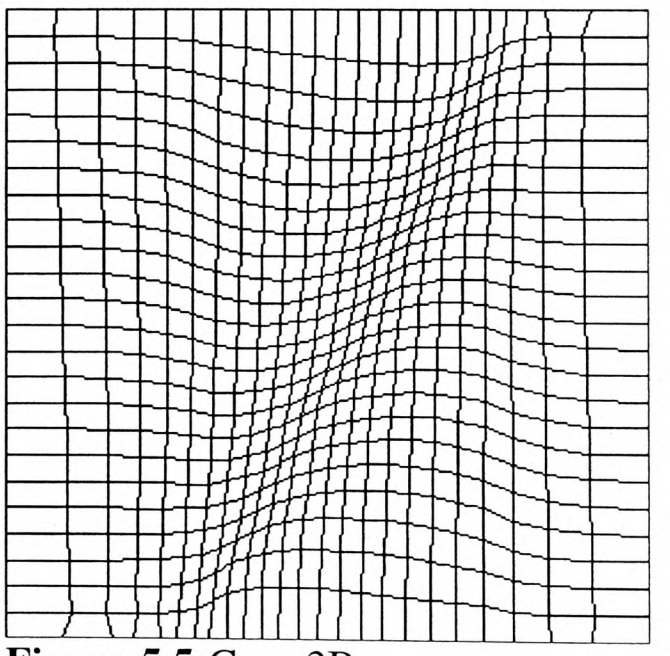


Figure 5.5 Case 2B

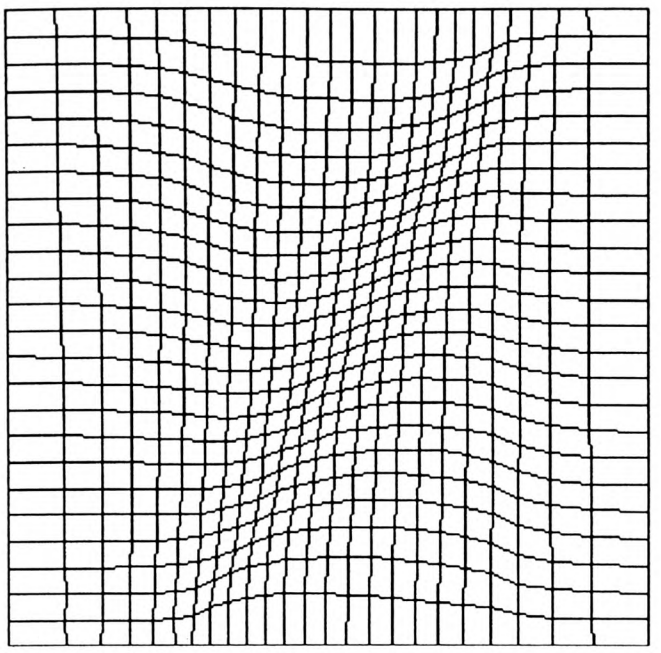


Figure 5.6 Case 2C

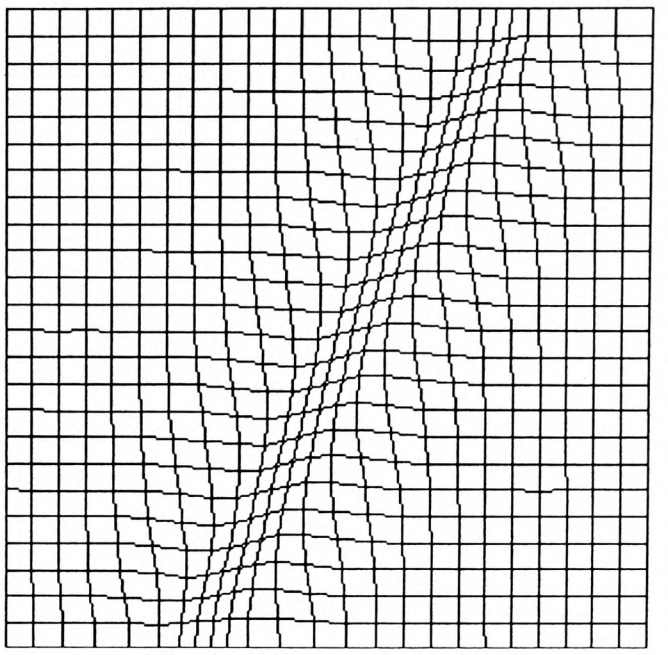


Figure 5.7 Case 2D

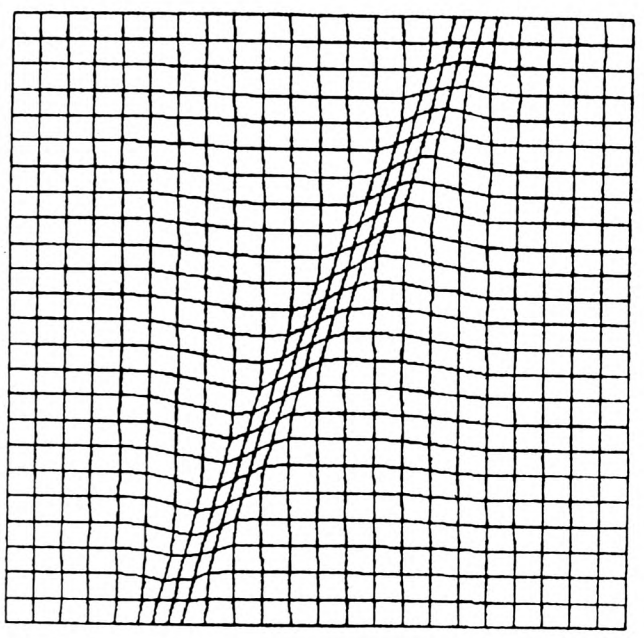


Figure 5.8 Anderson 1987a Equidistribution

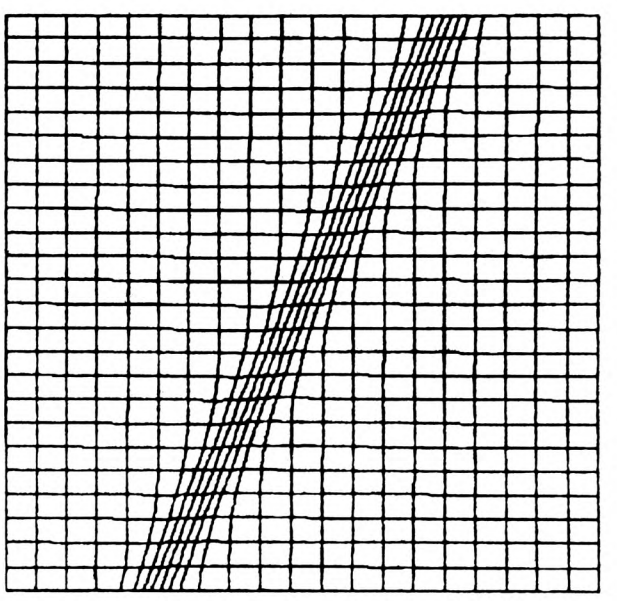


Figure 5.9 Anderson 1987a Poisson

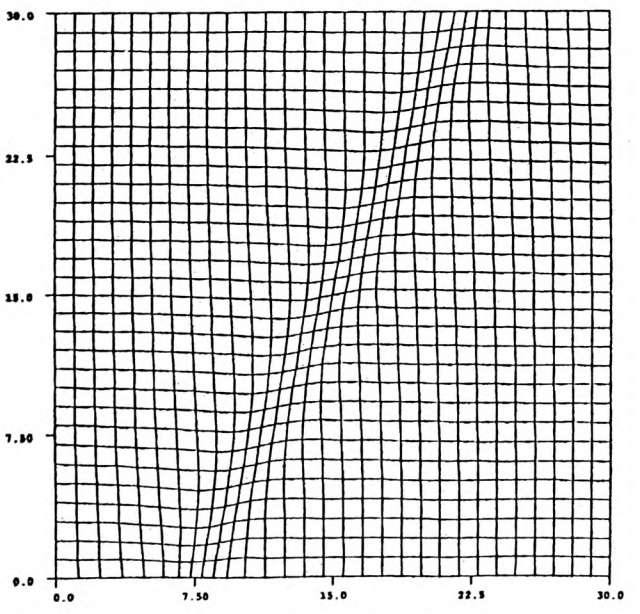


Figure 5.10 Anderson 1990 Poisson

Anderson (1987a) used this case to compare equidistribution (figure 5.8) and Poisson

(figure 5.9) grid adaption schemes and shows that the Poisson scheme is better.

The function is used here to compare the influence of power law fitting to control how much clustering occurs around the shock and how far its effect can be propagated towards the boundaries of the domain. Figures 5.4 through 5.6 show a steadily decreasing power law producing less clustering around the shock like function, but more movement at the grid boundaries. In all three of these cases the grids are smooth with a gradual change in cell size towards the function. Figure 5.7 shows the results without power law fitting. The clustering within the step is much higher than in the other cases, and its influence has been propagated through the entire domain but the change in grid size is much more sudden. This result is close to Anderson (1987a) (figure 5.9) but shows much more movement away from the step. The overall movement in this case is much less than in the other three cases.

5.2.3 Sine Wave

The purpose of this test is to demonstrate how the LPE equation can move a grid to a generalised curved function. The function is a graduated step in the shape of a sine wave.

$$\phi = \begin{cases} 0 & 0 \le y \le 11 + 4\sin(\pi x/12) \\ 0.5(y - 11\,4\sin(\pi x/12) & 11 + 4\sin(\pi x/12) < y \le 13 + 4\sin(\pi x/12) \\ 1 & 13 + 4\sin(\pi x/12) < y \le 24 \end{cases} \quad \textbf{(5.5)}$$

This function is used by Anderson (1987a) (figure 5.14, figure 5.15)

A 24x24 grid is used. The domain measures 24x24 units, each cell initially being a unit square.

In all cases the weight function used is

$$w = 1 + 2\phi_s + 0.02 \frac{|\phi_{ss}|}{\left(1 + (\phi_s)^2\right)^{\frac{3}{2}}} \qquad \textbf{(5.6)}$$

Sine Wave						
	λ_L	λ_E	Power Law Factor	Number of Moves	Total X move %	Total Y move %
Case 3A	1	2	2	30	18.6	15.2
Case 3B	1	3	3	10	15.2	40.1
Case 3C	1	2	No Power Law	20	12.5	39.3

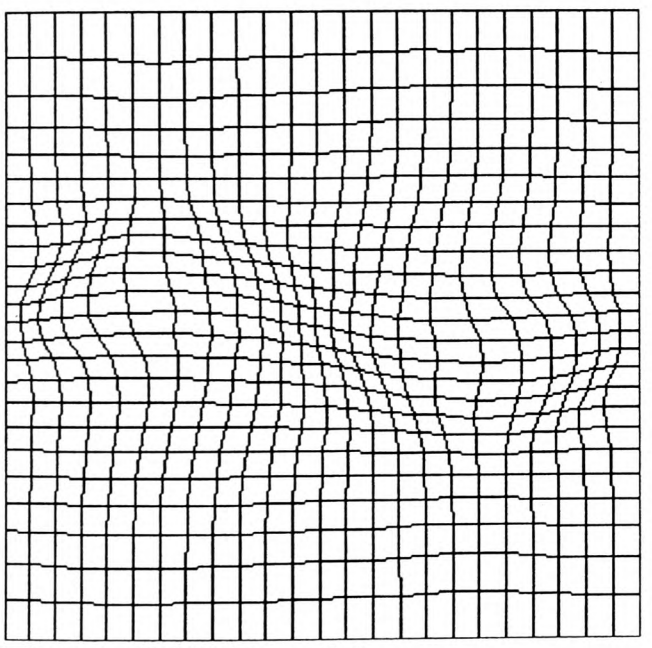


Figure 5.11 Case 3A

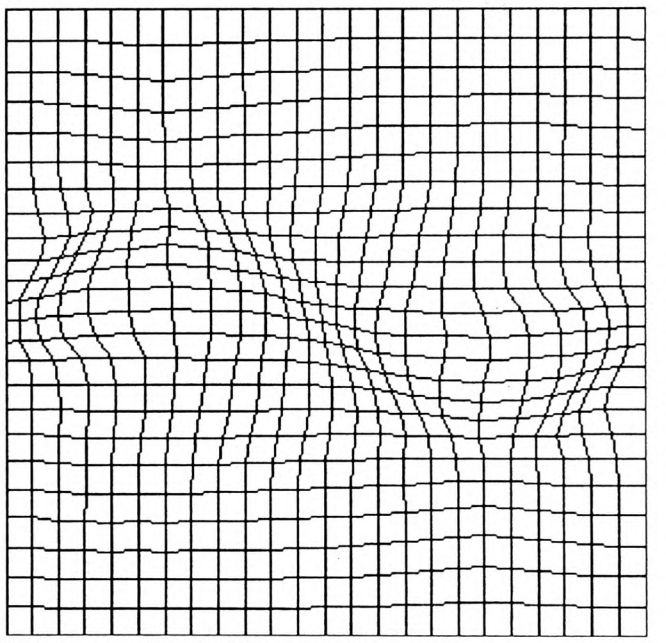


Figure 5.12 Case 3B

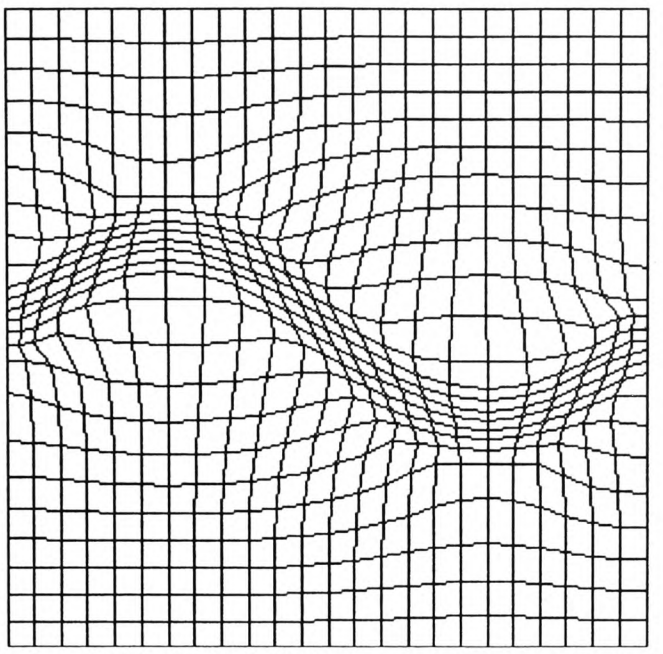


Figure 5.13 Case 3C

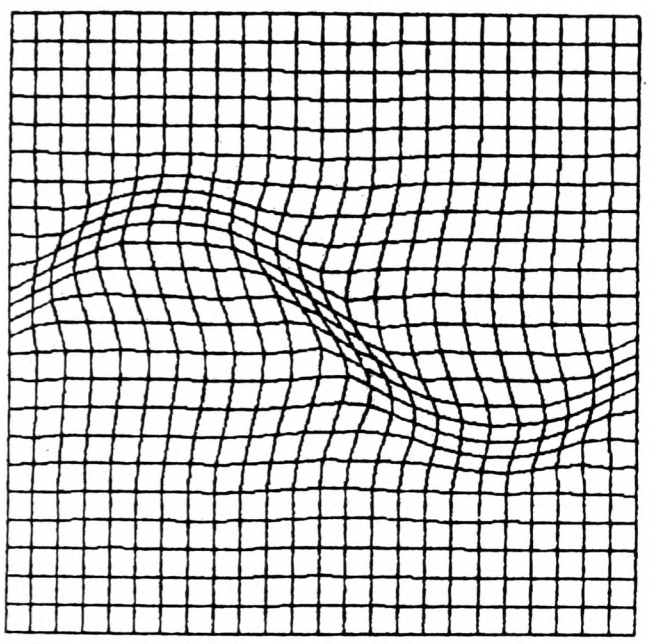


Figure 5.14 Anderson 1987a Equidistribution

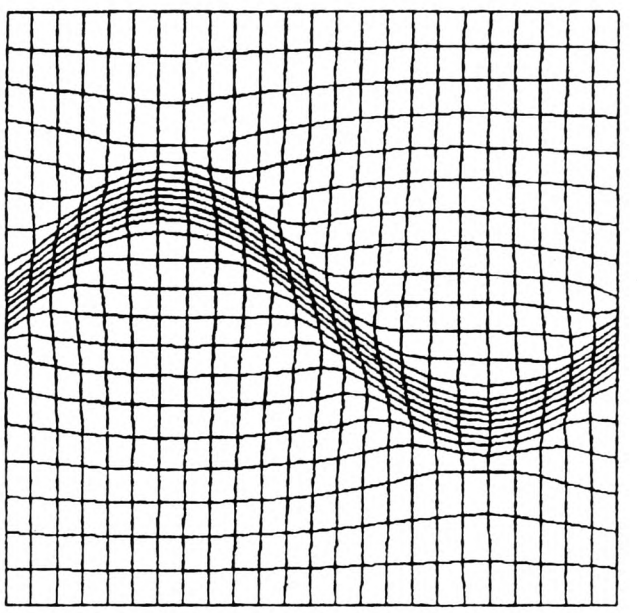


Figure 5.15 Anderson 1987a Poisson

As for the angled shock Anderson (1987a) used this case to demonstrate the shortcomings of equidistribution alone for adaption (figure 5.14) against the Poisson scheme (figure 5.15).

Case 3B shows that a final grid can be reached more quickly with an increased equidistribution term, though at the cost of some instability in the solution.

Case 3C shows a good fit of the grid to the step, though points are pulled away from the areas in the concave part of the function, leaving the area light in cells. This is less of a problem in the first two cases where the power law weight smoothing has spread the impact of the function. This distribution could also be improved by

constraining the horizontal movement of the grid points.

5.2.4 Hat Function

This function shows how the LPE algorithm can move a grid to two competing phenomena. The function used is

$$\phi = \begin{cases} 0 & y<x \\ 0.5 & y \geq x, \quad x^2+y^2>0.3 \\ 1 & y \geq x, \quad x^2+y^2 \leq 0.3 \end{cases} \tag{5.7}$$

The function is based on that used by Anderson (1987a) (figure 5.17) to show strength of his Poisson scheme, especially where the function shape switches between straight and circular.

A 40x40 grid is used over a unit square.

In this case the weight function used is

$$w = 1 + \phi_s + 0.01 \frac{|\phi_{ss}|}{\left(1+(\phi_s)^2\right)^{\frac{3}{2}}} \tag{5.8}$$

Hat Function						
	λ_L	λ_E	Power Law Factor	Number of Moves	Total X move %	Total Y move %
Case 4	1	1	No Power Law	15	21.7	21.8

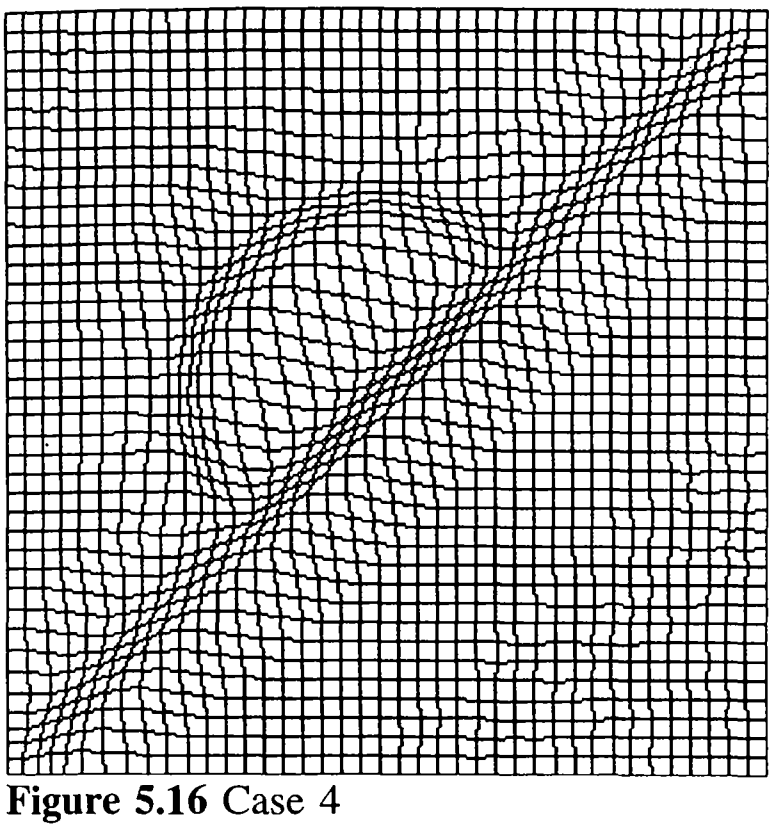


Figure 5.16 Case 4

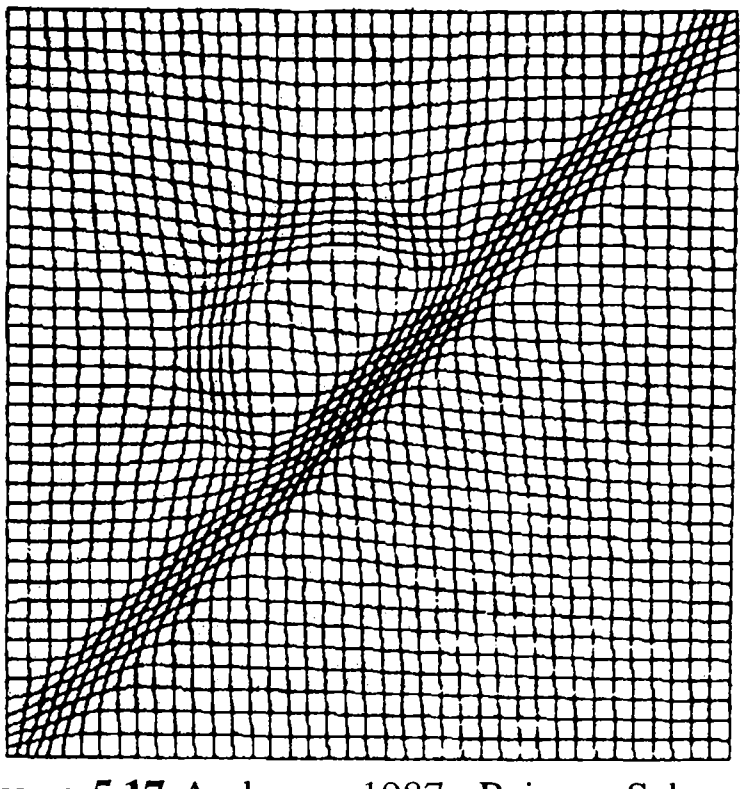


Figure 5.17 Anderson 1987a Poisson Scheme

The grid is distorted within the semi circle or hat, as cells are drawn out from this region towards its boundaries. It would work better if there were more cells in this region initially. The grid at the points where the semi circle and the line meet have retained good orthogonality.

5.2.5 Pill Box

A circular function is a frequently used test case for adaption. The form used here is the graded step found in Anderson (1990) (see figure 5.21) and (1987b) (figure 5.19 and figure 5.20). Divinsky (1991) attracts the grid to a circle (figure 5.22), and Brackbill and Saltzman (1982) solve a partial differential equation with a circular shaped solution.

The function used is

$$\phi = \begin{cases} 0 & 0.35 \le r \\ 1.75 - 5r & 0.15 < r < 0.35 \\ 1 & 0 \le r \le 0.15 \end{cases} \tag{5.9}$$

$$r = \sqrt{\left((x - x_c)^2 + (y - y_c)\right)}$$

Where the subscript *c* refers to the value at the centre of the grid.

A 30x30 grid is used over a unit square domain.

The weight function used is

$$w = 1 + \phi_s + 0.01 \frac{|\phi_{ss}|}{\left(1 + (\phi_s)^2\right)^{\frac{3}{2}}} \tag{5.10}$$

Pill Box						
	λ_L	λ_E	Power Law Factor	Number of Moves	Total X move %	Total Y move %
Case 5	2	1	No Power Law	20	15.8	15.8

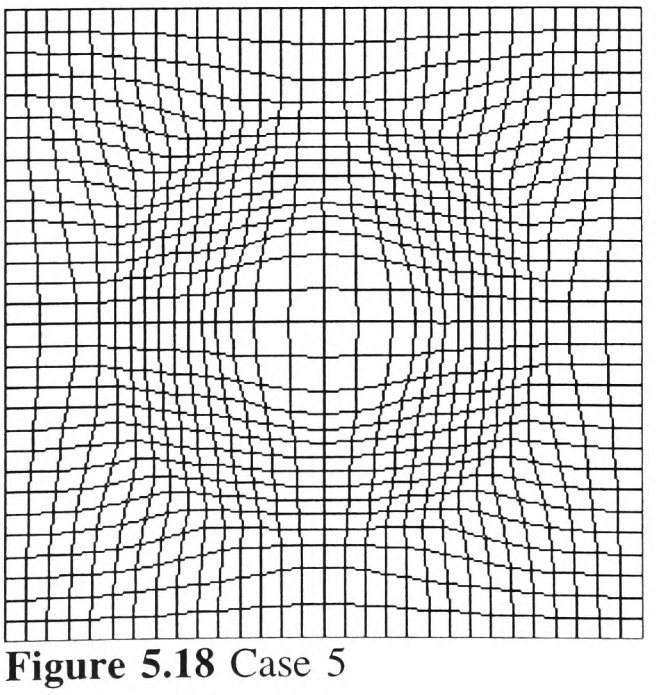


Figure 5.18 Case 5

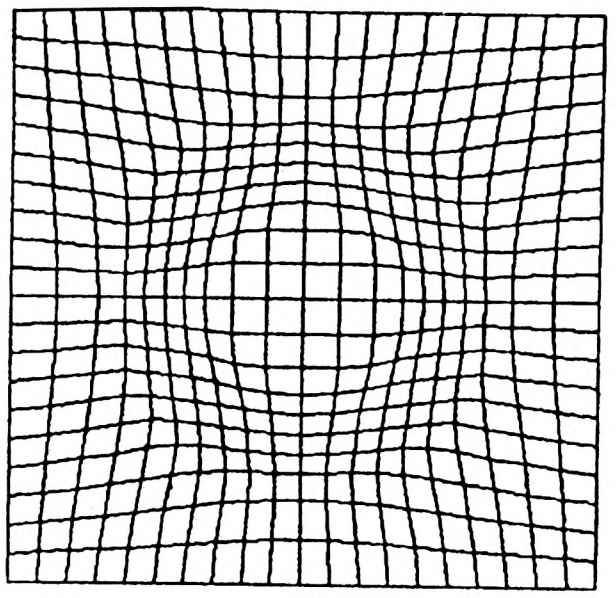


Figure 5.19 Anderson 1987b Poisson with Area Control

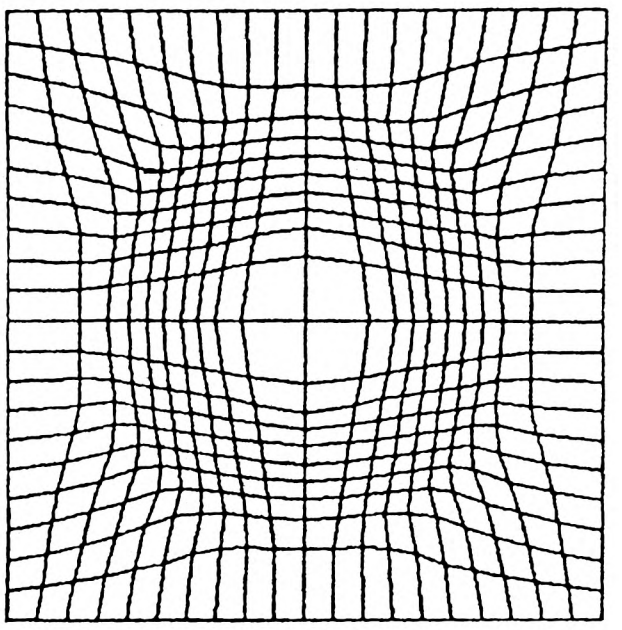


Figure 5.20 Anderson 1987b Equidistribution

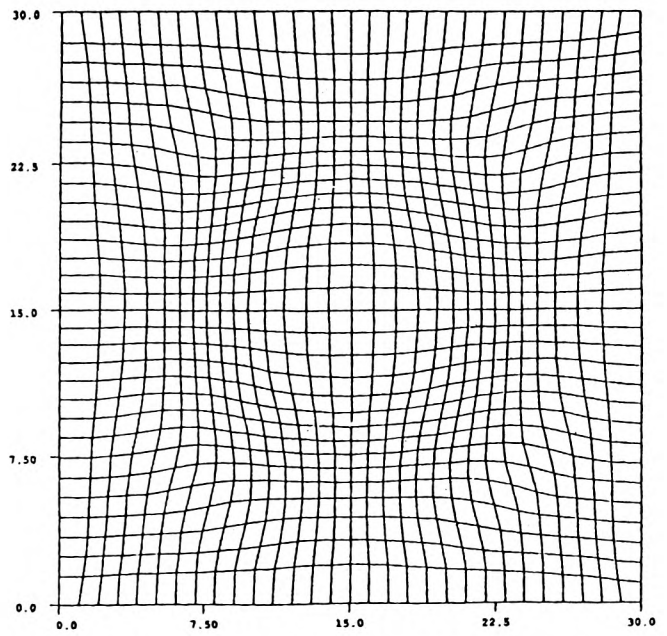


Figure 5.21 Anderson 1990 Poisson

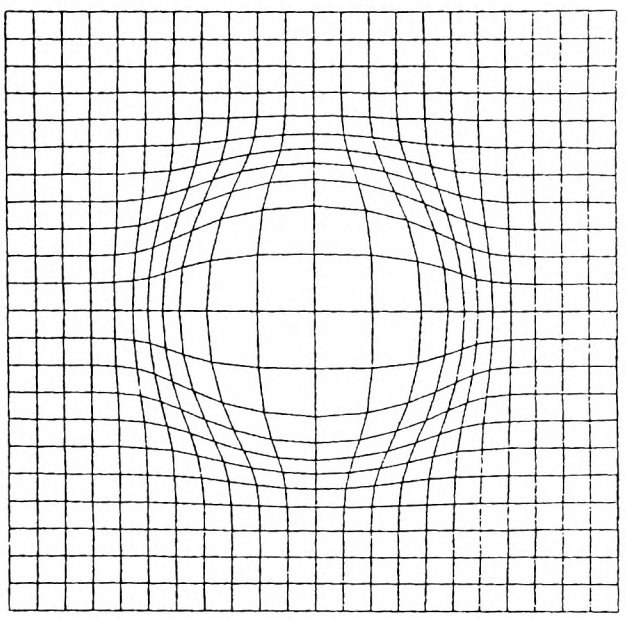


Figure 5.22 Divinsky 1991 Harmonic Map

5.2.6 Parabola

This function is based on that used by Divinsky (1991) (figure 5.24) where grid is attracted to the same parabolic line used here to define the function.

The function used is a parabola shaped step

$$\phi = \begin{cases} 0 & y > 3(x-0.5)^2 \\ 1 & y \leq 3(x-0.5)^2 \end{cases} \qquad \textbf{(5.11)}$$

A 21x21 grid is used, covering a unit square.

The weight function used is

$$w = 1 + \phi_s + 0.01 \frac{|\phi_{ss}|}{\left(1 + (\phi_s)^2\right)^{\frac{3}{2}}} \qquad \textbf{(5.12)}$$

Parabola						
	λ_L	λ_E	Number of Smoothing Steps	Number of Moves	Total X move %	Total Y move %
Case 6	1	1	5	17	50.5	23.7

In this and the following case power law fitting of the weight function is not used. Instead the variable used to calculate the weights is smoothed by setting its value at each point to the weighted average at the point and its neighbours (see 2.10.3.1). This process is repeated five times to further even out the function.

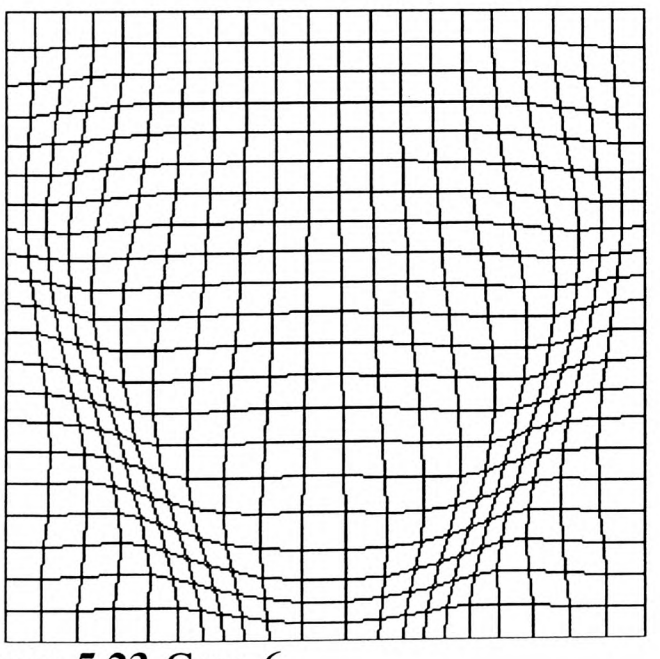


Figure 5.23 Case 6

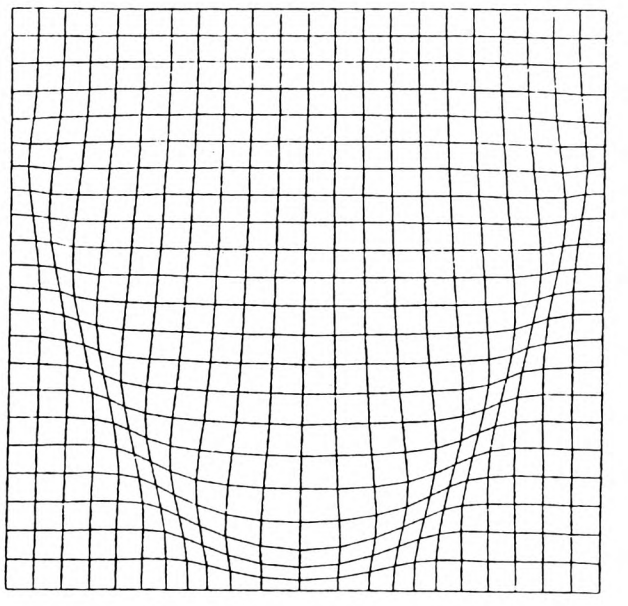


Figure 5.24 Divinsky 1991 Harmonic Map

5.2.7 Parabola with Line

This problem is designed to show how the LPE Algorithm can adapt a grid to two overlapping features. The function used is a parabola shaped step based on that used - by Divinsky (1991) (figure 5.26) where the grid is attracted to a line and the parabola.

$$\phi = \begin{cases} 0 & y>3(x-0.5)^2,\ y>x \\ 0.5 & y\le 3(x-0.5)^2,\ y>x \\ 0.5 & y>3(x-0.5)^2,\ y\le x \\ 1 & y\le 3(x-0.5)^2),\ y\le x \end{cases} \tag{5.13}$$

A 21x21 grid is used, covering a unit square.

The weight function used is

$$w = 1 + \phi_s + 0.01 \frac{|\phi_{ss}|}{\left(1+(\phi_s)^2\right)^{\frac{3}{2}}} \tag{5.14}$$

Parabola with Line						
	λ_L	λ_E	Number of Smoothing Steps	Number of Moves	Total X move %	Total Y move %
Case 7	1	1	5	25	33.4	28.1

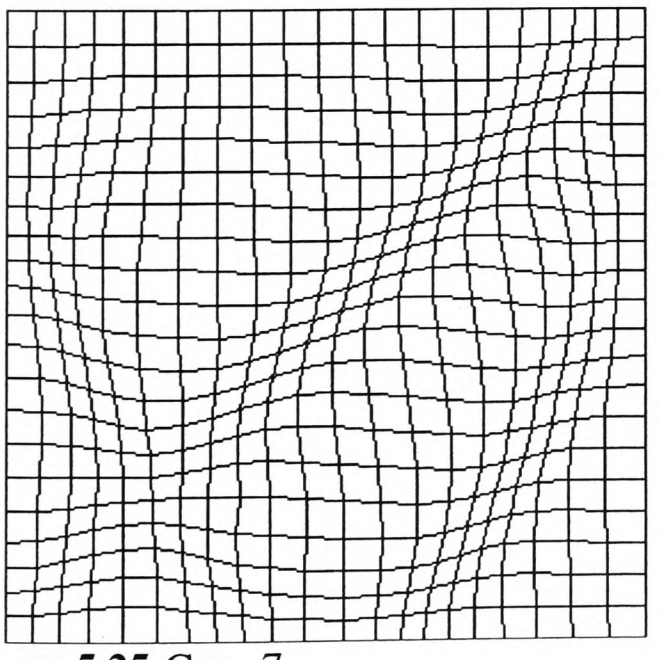


Figure 5.25 Case 7

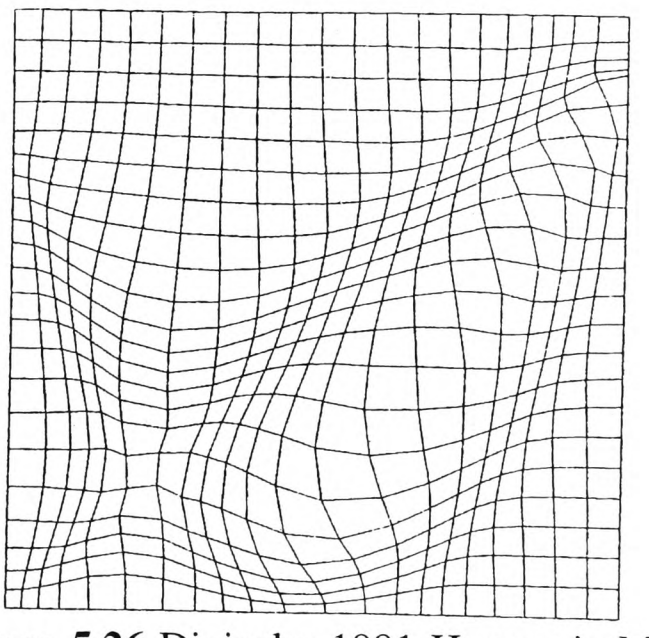


Figure 5.26 Divinsky 1991 Harmonic Map

5.3 Two Dimensional Flow Examples

The following cases were run using PHOENICS versions 1.6, 2.0 and 2.13 on PC's and Sun UNIX workstations. Case 5.3.2, which features driven cavity flow, was also run using PHYSICA version 2.

5.3.1 RAE2822 Aerofoil

The RAE2822 aerofoil test case is widely used for validation of compressible CFD methods as experimental data for pressure over the surface is readily available.

The case considered here is case 9, where a free stream Mach number of 0.730 generates a weak shock on the upper surface of the aerofoil at an angle of incidence of 3.19 degrees.

5.3.1.1 Boundary Conditions

The angle of incidence of the aerofoil is corrected to 2.79 degrees to allow for sidewall effects in the wind tunnel. The free stream flow has a Reynolds number of 6.5 million. The grid for the problem is shown in figures 5.26 and 5.27, and consists of 253x29 cells wrapped around the shape of the aerofoil to form an O grid.

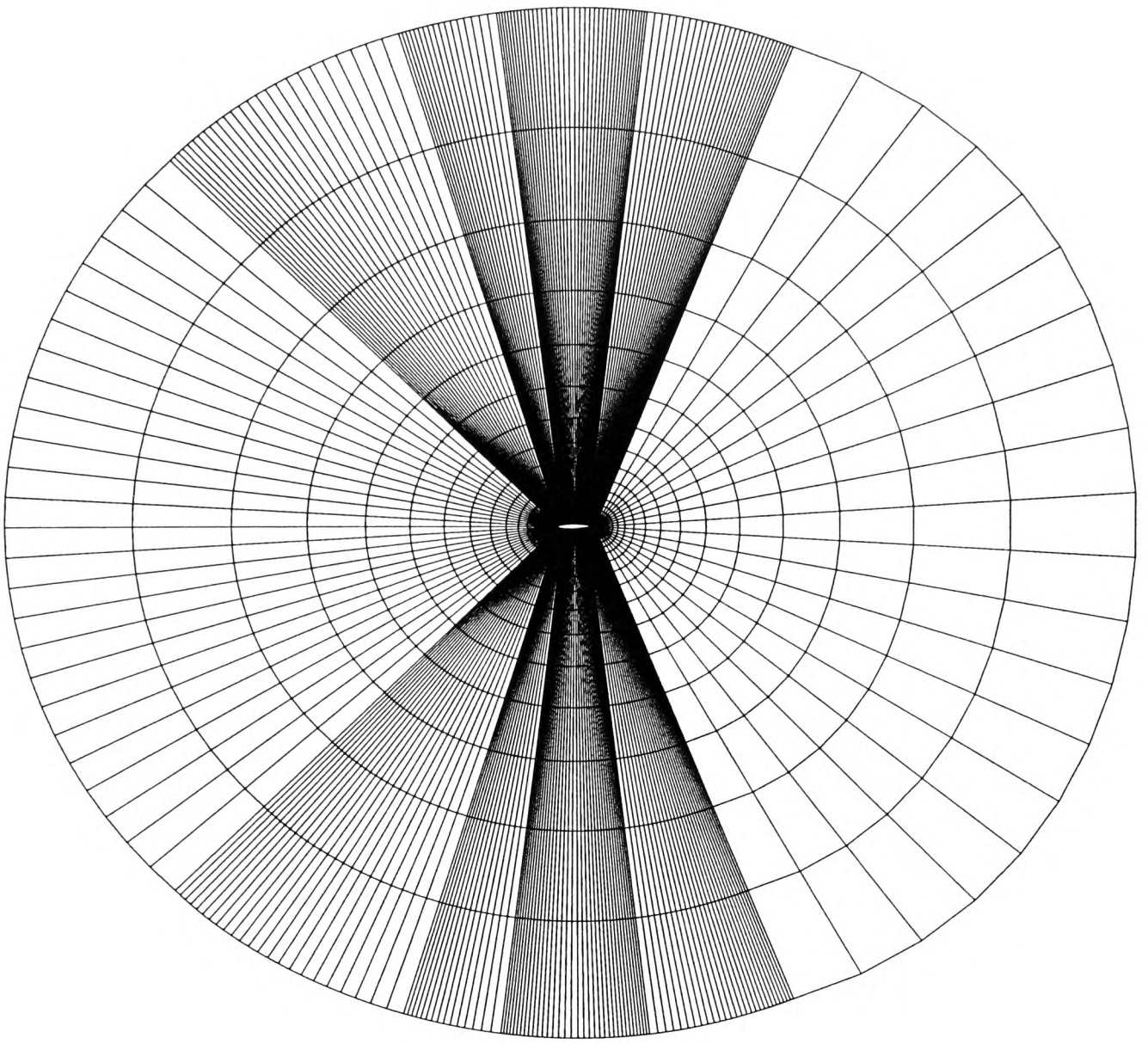


Figure 5.27a Initial Grid

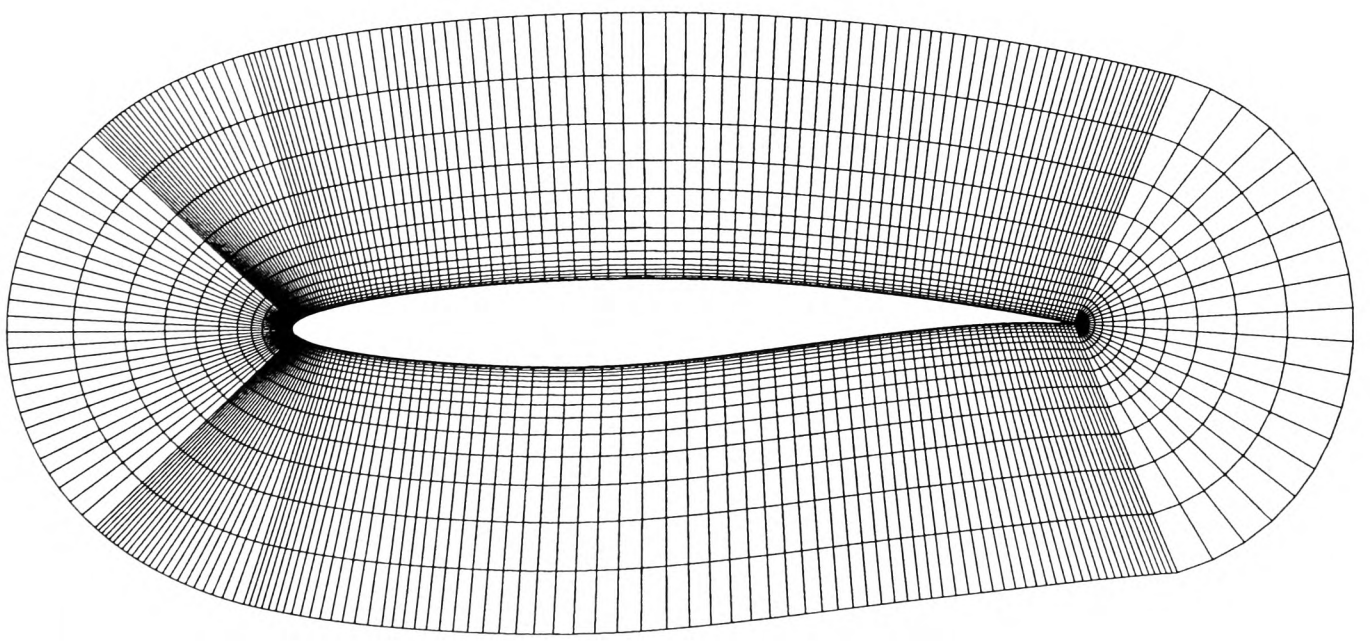


Figure 5.27b Initial Grid - Detail Around Aerofoil

The standard case is run until convergence. The adapted case is run for 1000 iterations

before adaption starts and for 500 iterations afterwards.

The under relaxation needed for the solution of the pressure distribution with adaption has to be increased from that used for the standard case. In this case the solution tends to need time to settle after the grid is modified and can diverge without relaxation.

5.3.1.2 Adaption Parameters

Two adaption domains are used in the problem area, one covering the upper surface from about 0.2 chord to the trailing edge, and the other the lower surface from about 0.1 chord to the trailing edge. The overall grid is very large with a wide variation in grid cell size and there is little point in trying to adapt it all. The distance away from the aerofoil surface that the domains should stretch is governed by the range of the perturbations in the flow caused by the aerofoil and the rate at which the cell spacing increases. Movement is limited to the points sliding along grid lines parallel to the aerofoil surface. There are no strong gradients to pull distant points on lines normal to the aerofoil towards the aerofoil.

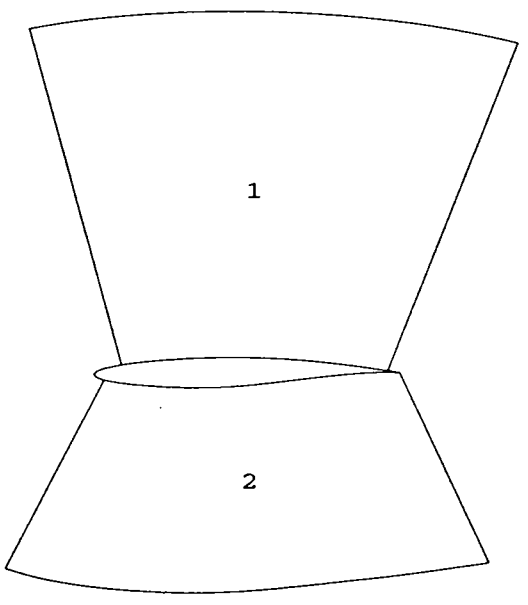


Figure 5.28 Domain Choices

The high concentration of grid cells near the surface of the aerofoil must be artificially maintained as there is no strong gradient in the flow to force the grid cells to remain there. Whatever gradient there is for movement will be aligned with the shock wave,

if present, and therefore at a steep angle to the aerofoil surface. The Laplace term will then dominate the adaption equations and will try to force the cells away from the surface. This artificial control is achieved by extensive use of the curve fitting routines in the domains to be adapted. This will mean that near the surface of the aerofoil the grid points will only be allowed to move over the grid lines defined by the initial grid.

The high variation in pressure in a small area at the leading edge of the aerofoil, coupled with a highly skewed local grid prevents effective grid adaption here. Any adaption domains defined for the grid must start far enough away from the leading edge of the aerofoil to negate any influence it may have. The main area of interest as regards grid adaption is the weak shock wave on the upper surface of the grid, so the adaption domains can be tailored to fit.

At the trailing edge of the aerofoil the grid is also skewed. There are also potential problems dealing with the grid cut and the point where the grid cells collapse. Though there should be no strong gradients here to influence adaption elsewhere, the effective limit to the adaptive domains is where the grid lines perpendicular to the aerofoil surface start to slope.

The extent of the domains is also influenced by the region in the grid above and below the trailing edge where the cell size changes rapidly and the cells themselves have a very high aspect ratio. In these regions use of adaption can easily lead to grid lines kinking and even overlapping due to minor perturbations in the flow.

The extensive use of curve fitting helps to maintain grid smoothness.The fixed curve patches are displayed in figure 5.29.

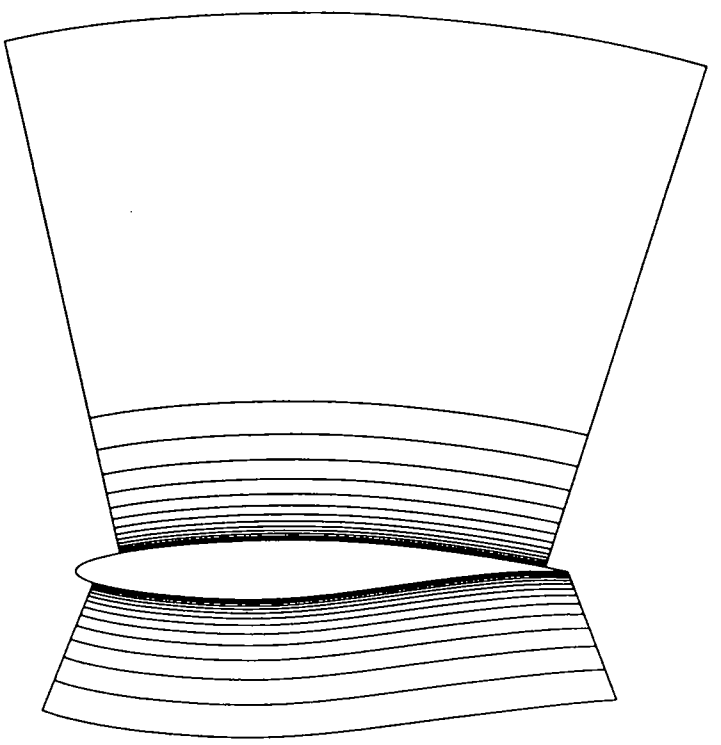


Figure 5.29 Fixed curves in the adaption domains

The most important part of the grid contains the aerofoil boundary layer cells and this is the region for which experimental data is available. It was found in early runs that the movement of the grid nodes right on the boundary was more prone to fluctuations than in the rest of the flow and more importantly tended to lag behind the movement of the points away from the surface. Preventing any movement at the surface was undesirable as it left the boundary layer grid cells highly skewed as they leaned towards the computed shock. To maintain the orthogonality of the boundary layer cells the movement of the boundary nodes was set to that of the nodes on the adjacent grid line inside.

The adaption parameters are

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
1000	1900	100	10	0.8

The constants used for adaption are given in the following table.

	LPE Equation			Variable	Weight Function
Domain 1 Upper Surface	λ_L	λ_P	λ_E	Pressure	$1+\phi_x+0.01k$ x^3 power law smoothing
	1	2	5		
Domain 2 Lower Surface	λ_L	λ_P	λ_E	Pressure	$1+\phi_x+0.01k$ x^3 power law smoothing
	1	1	2		

The grid is moved to pressure in both domains, with a higher weight on the equidistribution term on the upper domain. There is not a particularly strong gradient in either domain and it is weaker in the lower domain. To have the same parameters for the lower domain, domain 2, causes problems as minor pressure perturbations have more effect. To maximise the effect of the pressure gradient power law smoothing is used for the weight term, with the weights assumed to increase at a rate proportional to the distance cubed up to the local maximum. This use of smoothing allows high values to be used for the equidistribution term, allowing faster grid movement whilst keeping the LPE equation stable.

5.3.1.3 RAE2822 Aerofoil Case 9 Results

The grid has moved to concentrate grid cells both above and below the aerofoil. The grid concentration on the upper surface has occurred downstream of the location given in the experimental results.

Figure 5.33 shows that though there is little difference in the accuracy of the solution on the underneath of the aerofoil, the shock on the upper surface is picked out much more clearly and is of the same magnitude as the experimental results. The calculated shock occurs downstream of the experimental shock, though it is in the centre of the smeared shock predicted by PHOENICS on the original grid. The adaption has changed the PHOENICS results and shown the potential to give better results, but also possible limitations with PHOENICS.

Figure 5.34, showing surface pressure coefficients for standard, globally refined and adapted refined grids generated by Dannenhoffer (1985) is included to show that the apparent change in position of the shock wave due to adaption is not limited to the current work alone.

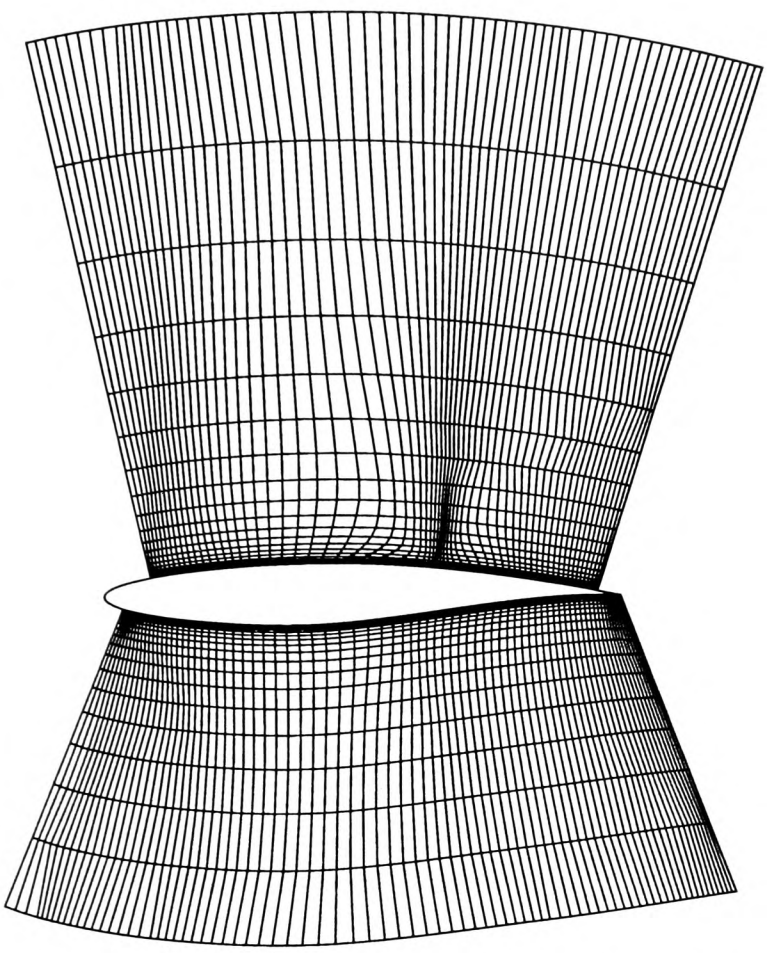


Figure 5.30 Adapted Grid

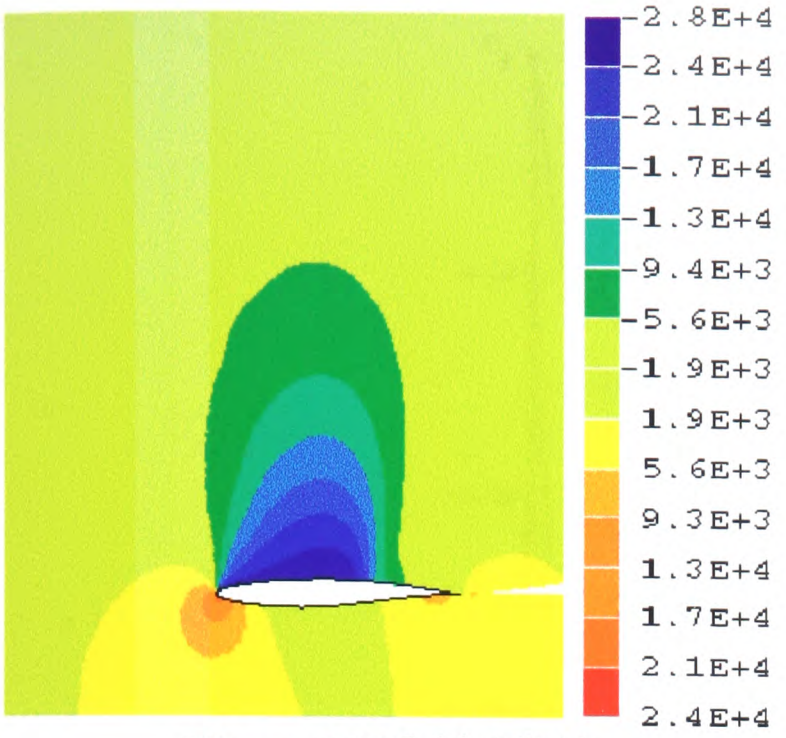


Figure 5.31 Initial Grid

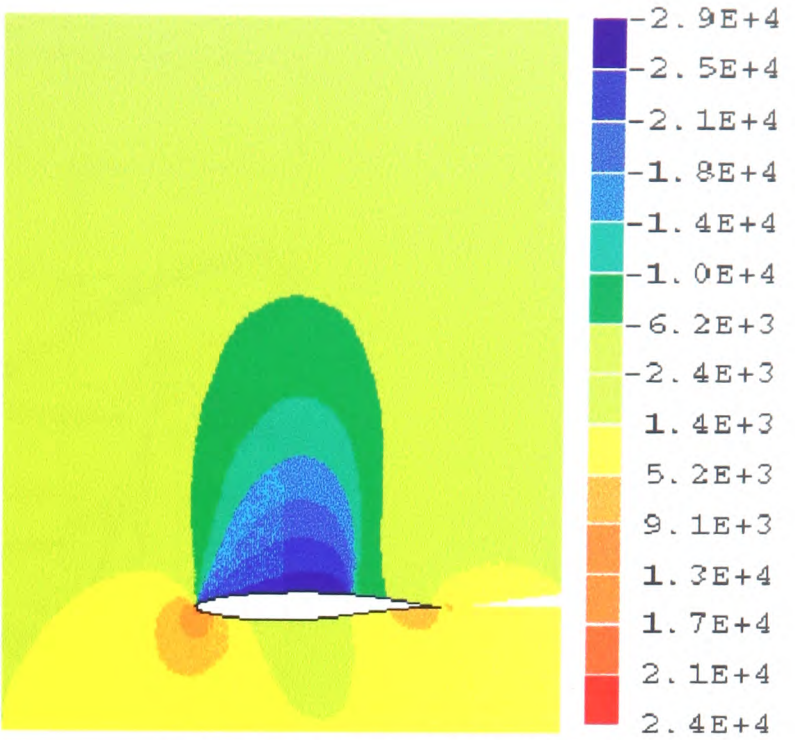


Figure 5.32 Adapted Grid

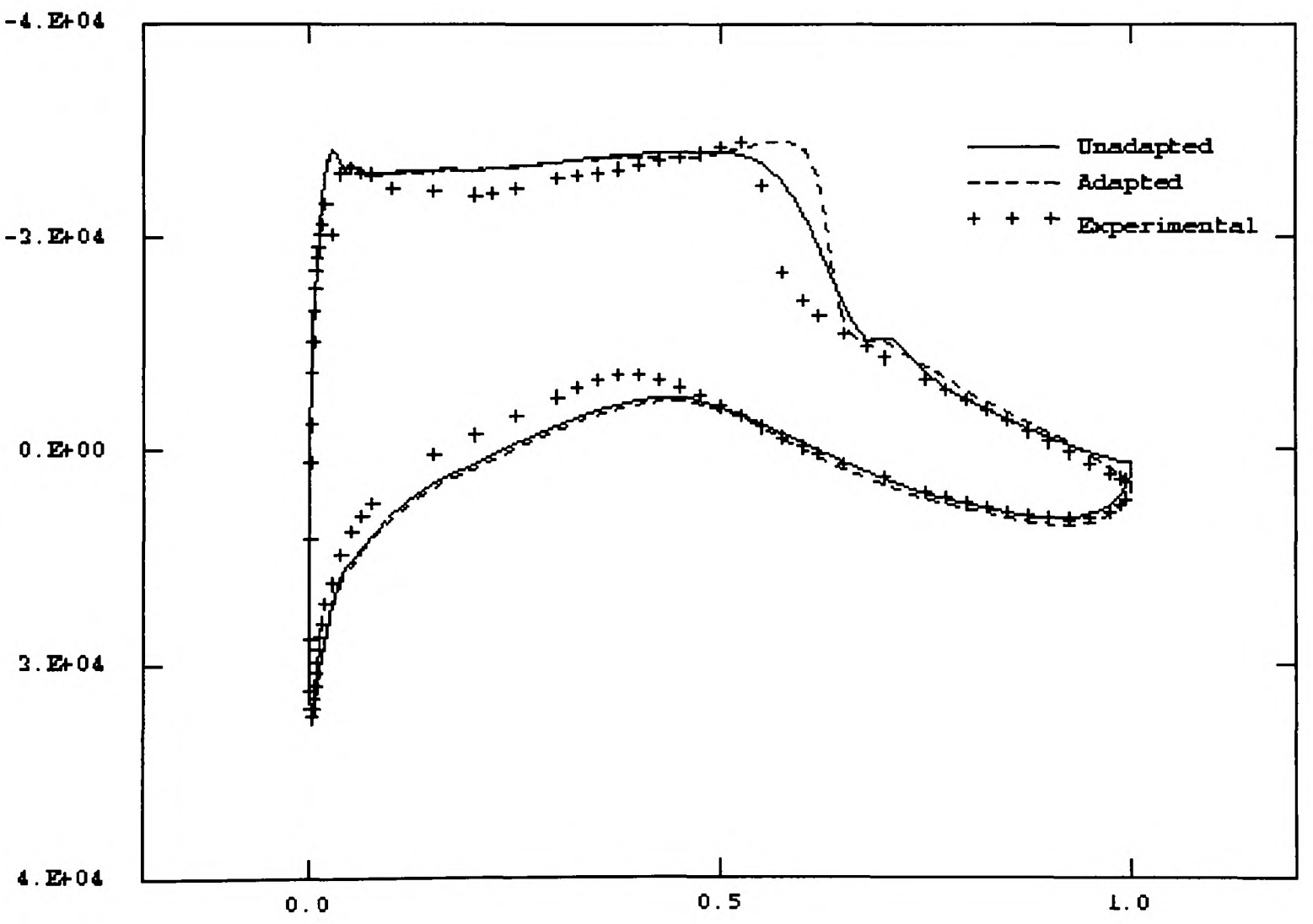


Figure 5.33 Surface Pressure Plots

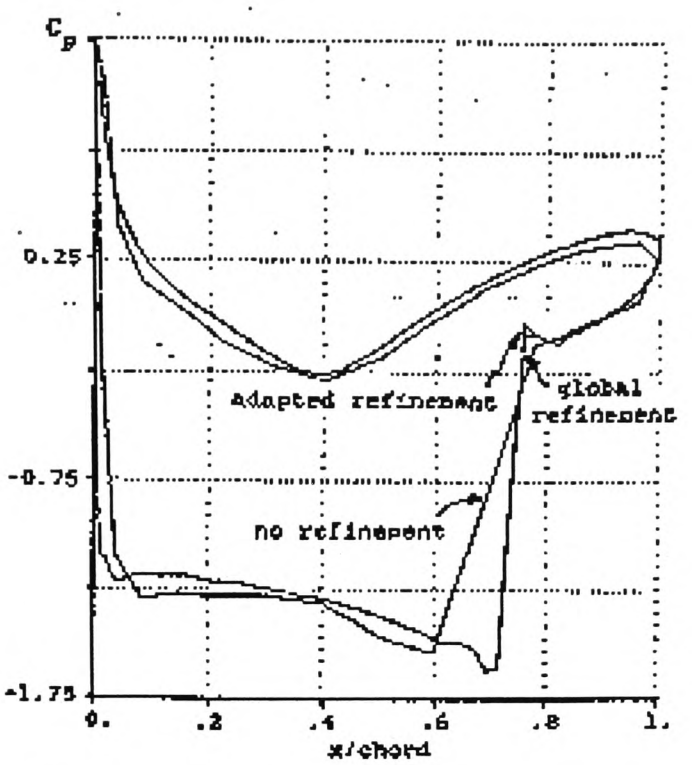


Figure 5.34 Surface Pressure calculated with Adapted Grid by Dannenhoffer (1985)

5.3.2 Driven Cavity Flow

The driven cavity problem is widely used as a test of numerical methods (Ghia 1982) and has also been used as an example in literature covering grid adaption (Lin and Wu 1993, Mavriplis 1992, Moukalled and Acharya 1991, Shen and Reed 1993, Thompson and Ferziger 1989).

The dominant source of error in this case is the hybrid advection scheme used within the finite volume solver. The accuracy of this scheme depends upon the local Reynolds number which is proportional to the local velocity and the cell spacing. Reducing cell spacing and aligning the grid to local flow conditions through adaption will improve results calculated on coarse grids.

This case is used to demonstrate adaption within PHYSICA as well as for PHOENICS.

5.3.2.1 Boundary Conditions

A wall moving at 1 ms^{-1} creates a circulation in a square domain measuring 1x1m. The problem is dependent on the Reynolds number, and the results presented are for a Reynolds number of 1000.

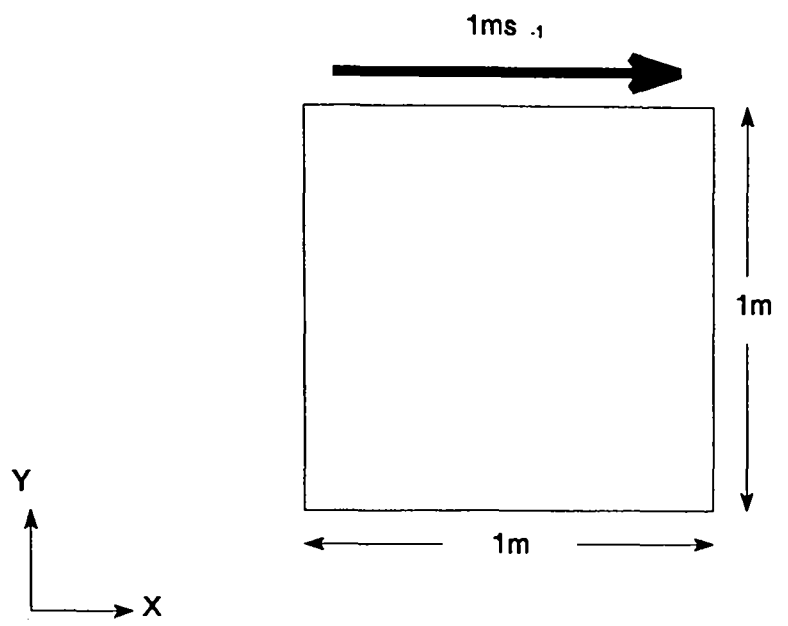


Figure 5.35 Driven Cavity Definition

The problem was run on 40x40 and 60x60 uniformly distributed and adapted grids in both PHOENICS and PHYSICA. For comparison of the effects of adaption the same case is also presented on a finer 80x80 uniform grid without adaption in PHOENICS. The results were similar enough between PHYSICA and PHOENICS for the 60x60 uniform case (figure 5.38) to make running the 80x80 PHYSICA case unnecessary.

The PHOENICS cases were run for 2500 sweeps. The PHYSICA cases were run until convergence was reached as defined by the default set up file.

With a very fine grid (250x250 cells) PHOENICS reproduced the Ghia results for a Reynolds number of 1000 exactly.

5.3.2.2 Adaption Parameters

One domain is used that covers the whole problem area. Curve patches are set up along all of the boundaries of the adaption domain except the top, moving, wall to

allow boundary points to move.

The adaption parameters are

	First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
PHOENICS	300	600	100	8	1.0
PHYSICA	80	400	40	10	0.9

	LPE Equation			Variable	Weight Function
	λ_L	λ_P	λ_E	Magnitude of Velocity	$1+\phi$ x^3 power law smoothing
PHOENICS	1	0	3		
PHYSICA	1	0	3		

Magnitude of velocity is used to drive the grid as there are no other strong gradients that exist throughout the whole domain. The form of weight function here is used as the only strong gradients within the flow except at the moving wall itself which lie at the edges of the main eddy produced by the moving wall. Moving the grid to these gradients will pull cells away from where the flow is greatest and needs most resolution.

To prevent the flow at the boundary layer dominating all movement, the magnitude of ϕ is limited to a maximum of 0.6. All values above 0.6 are treated as 0.6, giving a high constant weight across the top of the domain which resists movement but lacks the change in weight which drives the grid.

The weight for the Poisson term was set to zero as for a uniform grid the control functions will be zero and the Poisson term will collapse to the same as the Laplace term.

The different parameters used for PHYSICA and PHOENICS came about because of the different response from PHYSICA to adaption.

5.3.2.3 Driven Cavity Results

The PHOENICS run using adaption clocked 4045 seconds on a Sun SPARC 5 workstation for 5000 sweeps, against 6734 seconds for the fine grid and 3834 seconds for the coarse grid.

Figures 5.36 and 5.37 show that the grid cells have concentrated on and are more aligned to the main eddy in the cavity in both PHYSICA and PHOENICS.

Figures 5.38 and 5.39 plot the component of flow velocity parallel with the moving lid on the centre line of the box for all the flow cases run with the Ghia results (1982). They show that significant improvements are offered by adaption within PHOENICS over the plain 60x60 grid for Re=1000 and even over the 80x80 grid, though the improved results at the most important features, the moving wall and the main eddy, are balanced by a mild loss of accuracy elsewhere.

The PHYSICA results show a less marked improvement over the static grid results than for PHOENICS, though with less loss of accuracy away from the low velocity peak. There is less movement overall in the PHYSICA adapted grid.

Figure 5.40 presents the grid cell spacing along the centre line for the PHOENICS adapted grid alongside the grid density for the 60x60 and 80x80 grids. It shows that grid density peaks at the moving lid boundary condition and at the centre of the main eddy about 0.18 above the base of the cavity at the expense of neighbouring regions.

Figures 5.41-44 present the velocity vectors for the flow field with the different grids. The eddy appears to be better defined in the adapted grids than the unadapted grids. The eddy in the adapted PHYSICA results is particularly well defined.

Figures 5.45-46 present records for the rate of convergence of pressure on the 60x60 grids in PHYSICA and PHOENICS, measured using the residuals or errors in the governing equations during the solution procedure. The magnitude of change for each record are not comparable between the different cases as the way the residuals are calculated in PHYSICA and PHOENICS is different. The PHOENICS test cases were considered to have converged when the sum of the absolute values of the residuals reduced below 1.e-4. Using this criteria the adapted grid converges faster than the unadapted grid despite the large disruption to the curve caused by adaption.

The PHYSICA cases also use a tolerance value of 1e-4 to determine convergence. Here the adapted case does not converge faster. The effect of adaption on the residual error is also much less than in the PHOENICS cases.

Adaption parameters for PHYSICA will be different than those from PHOENICS because of a number of reasons. These include:-

- Reduced sensitivity of PHYSICA to grid quality. This is because it uses Rhie and Chow (1983) interpolation against the staggered grid used in PHOENICS.
- Calculation of adaption data set. In PHYSICA the data set used to calculate the weight function is stored at the cell centre and calculated prior to entry to the adaption module, where it is interpolated to grid nodes. In PHOENICS the data set is calculated directly at grid nodes by interpolating and combining individual velocity components from cell faces.
- Calculation of velocity at cell centres IN PHYSICA as opposed to cell faces in PHOENICS means that velocities at boundary conditions are potentially better defined in PHYSICA as in PHOENICS they will be further away. In PHYSICA this may lead to a steeper velocity gradient at the moving lid which will influence the way in which the adaption algorithm operates over the remainder of the domain, reducing the effect of the eddy in the

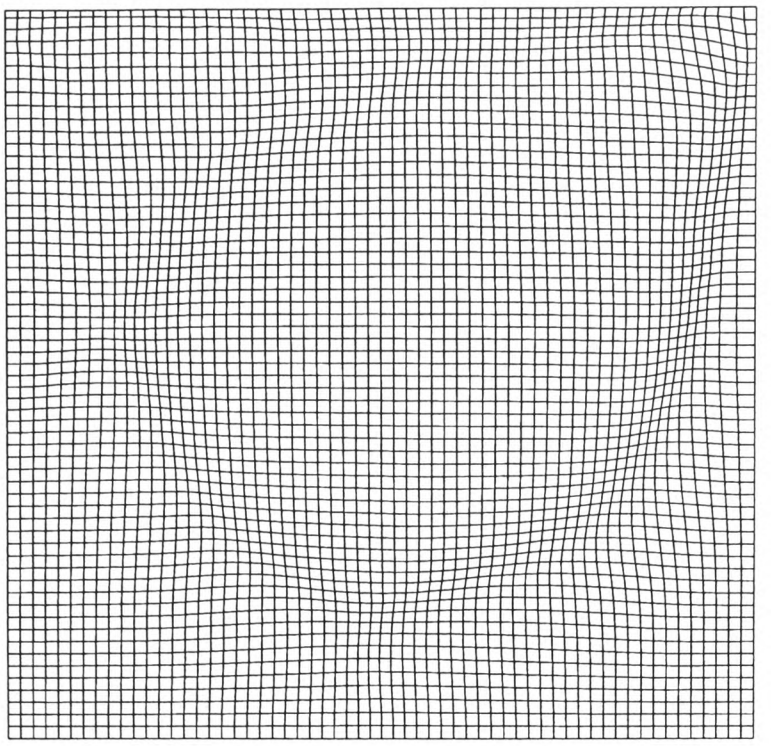


Figure 5.36 Grid adapted in PHOENICS 60x60 Cells

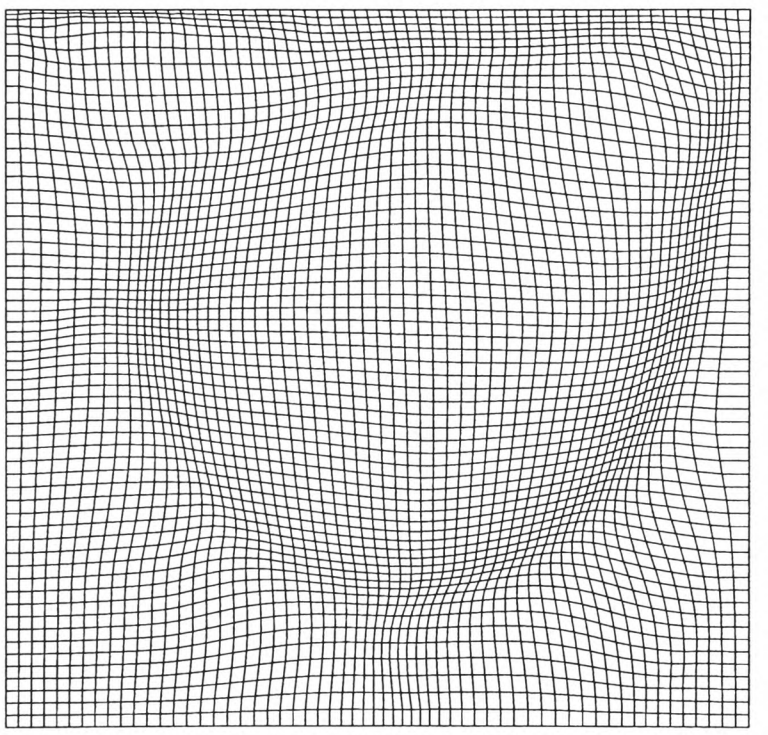


Figure 5.37 Grid adapted in PHYSICA 60x60 Cells

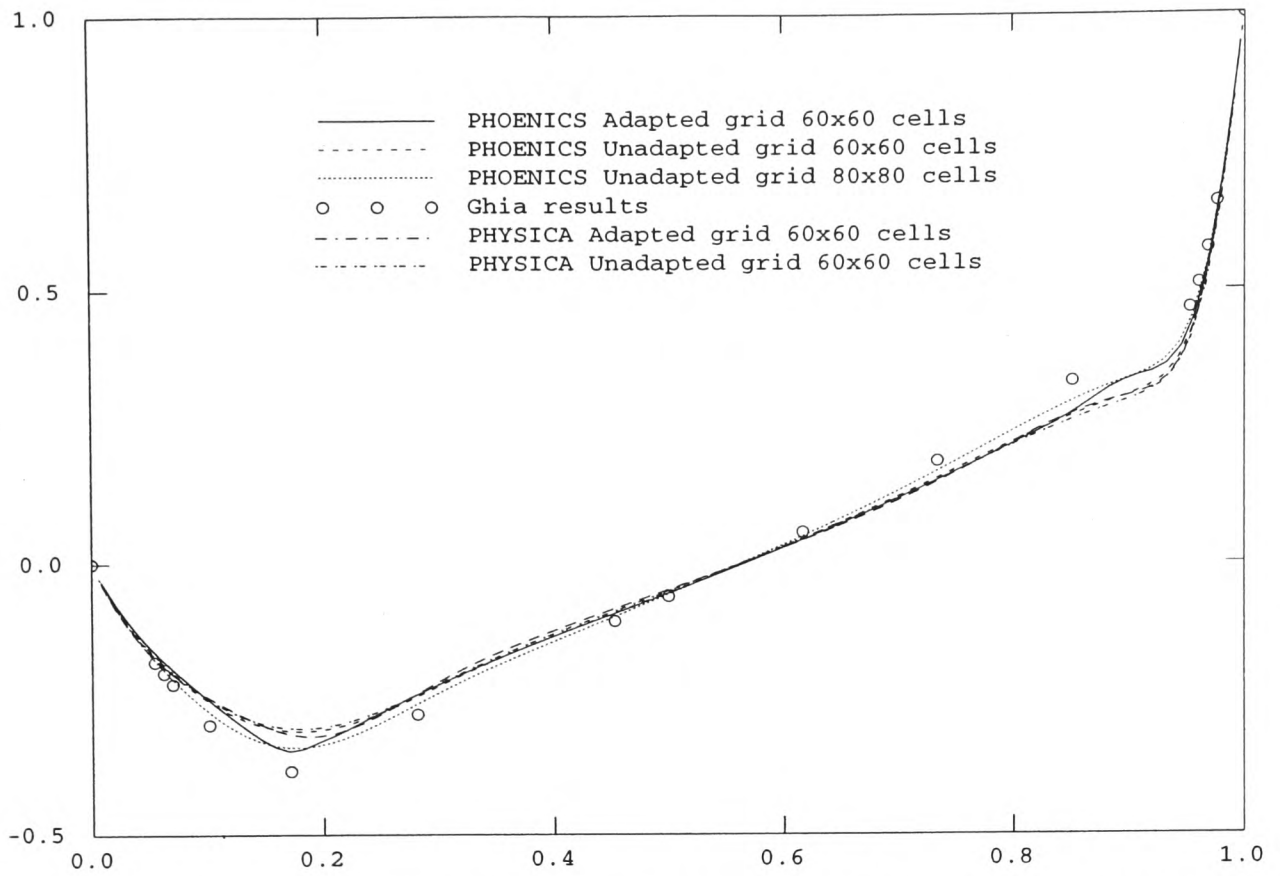


Figure 5.38 U Component of Velocity on Centre Line (Velocity against Distance)

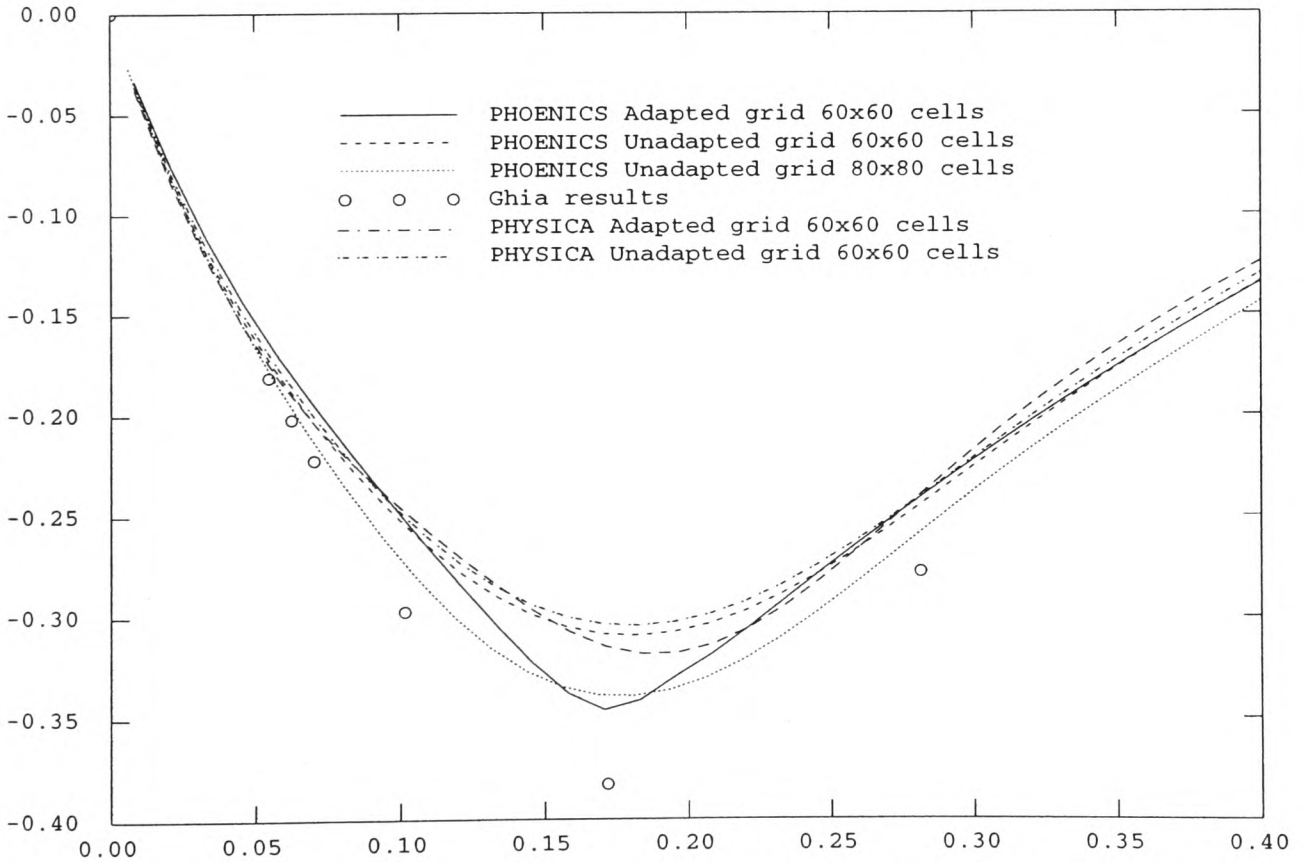


Figure 5.39 Detail of figure 5.38, U Component of Velocity on Centre Line (Velocity against Distance)

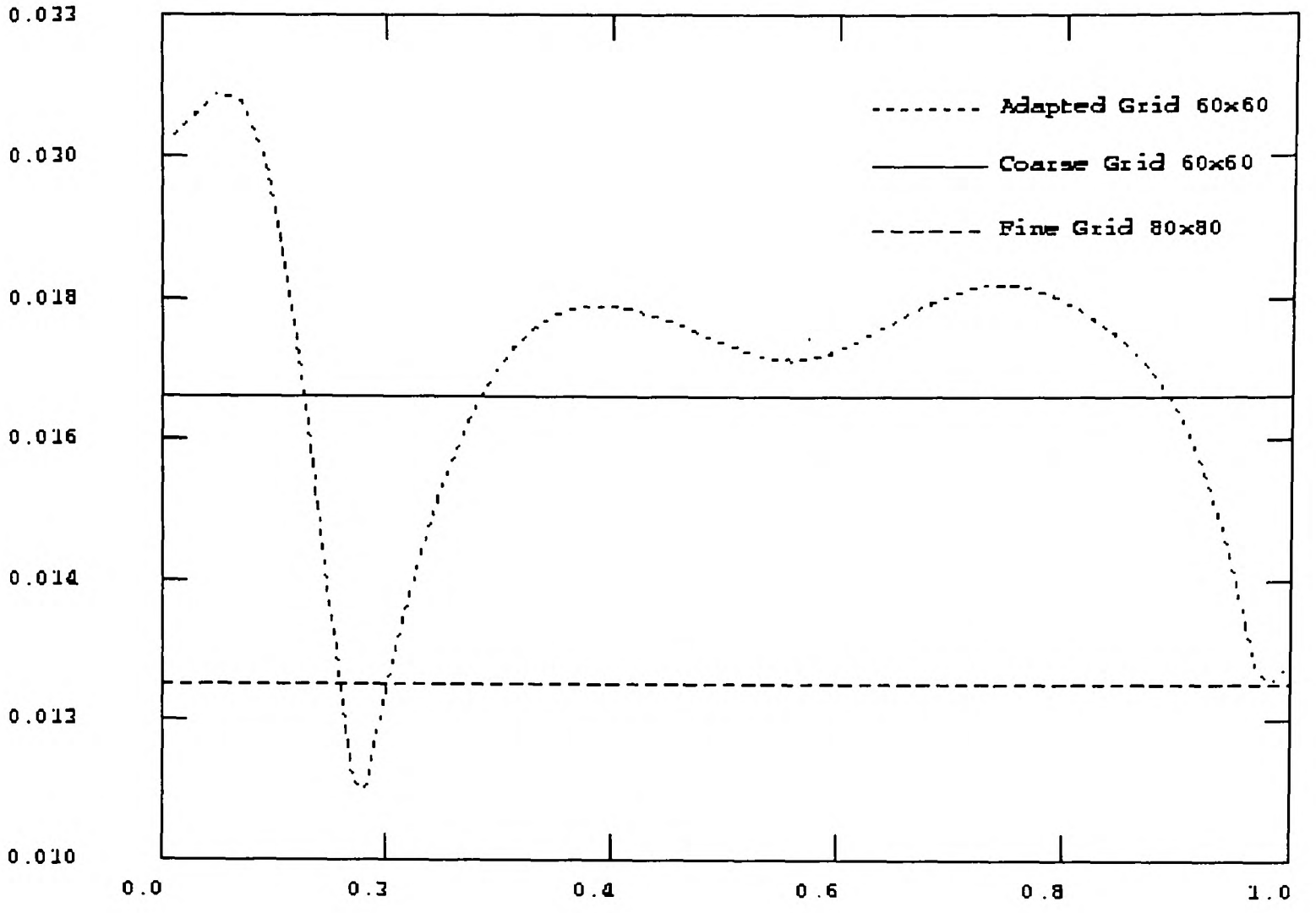


Figure 5.40 Cell Spacing along Centre Line after adaption in PHOENICS

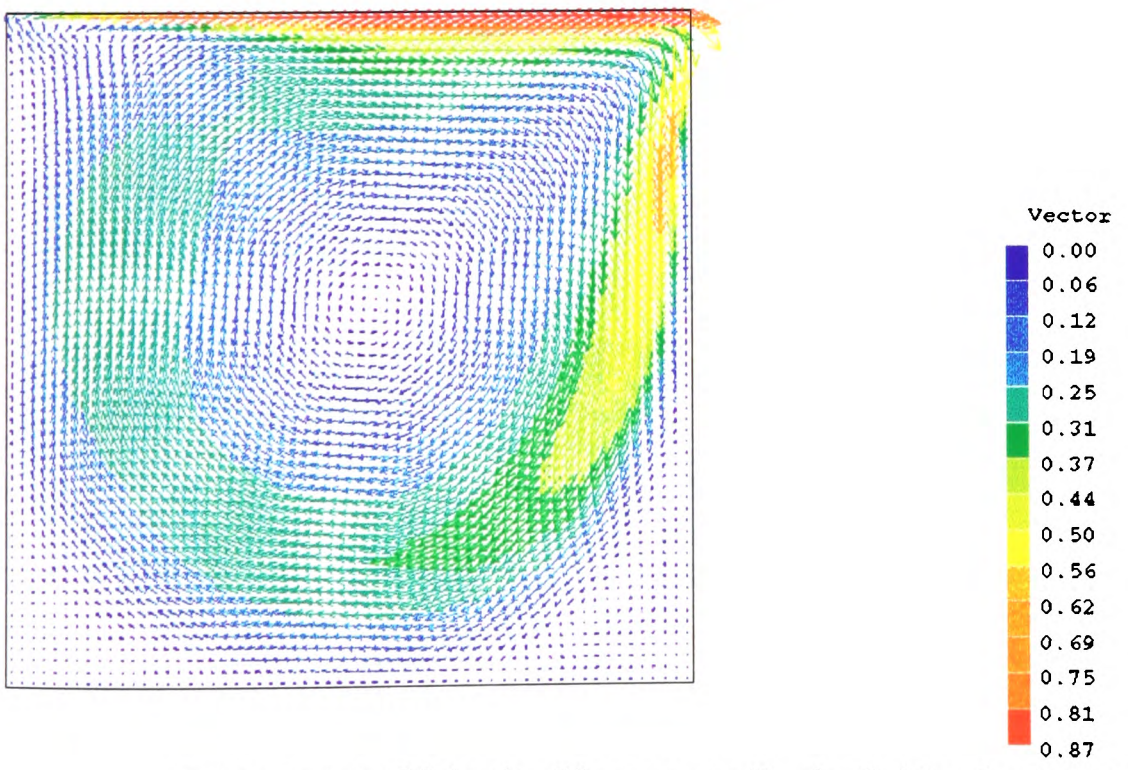


Figure 5.41 Velocity Vectors on Basic Grid calculated in PHOENICS

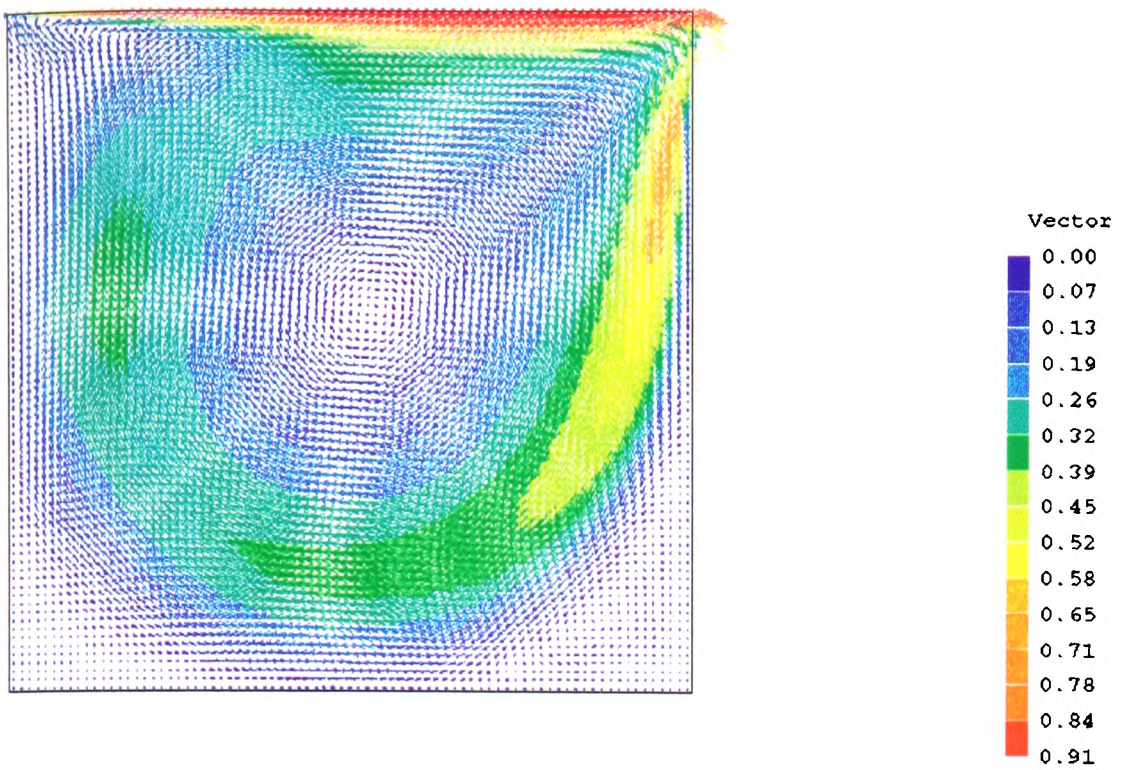


Figure 5.42 Velocity Vectors on Fine Grid calculated in PHOENICS

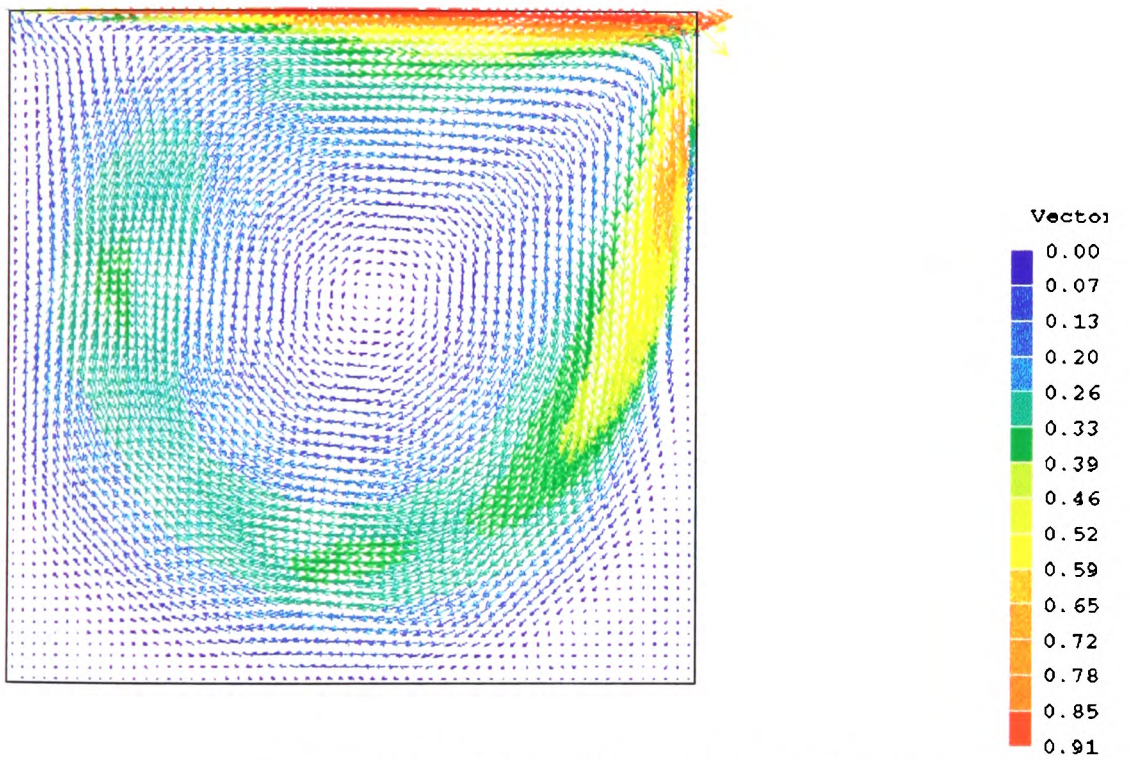


Figure 5.43 Velocity vectors on Grid adapted in PHOENICS

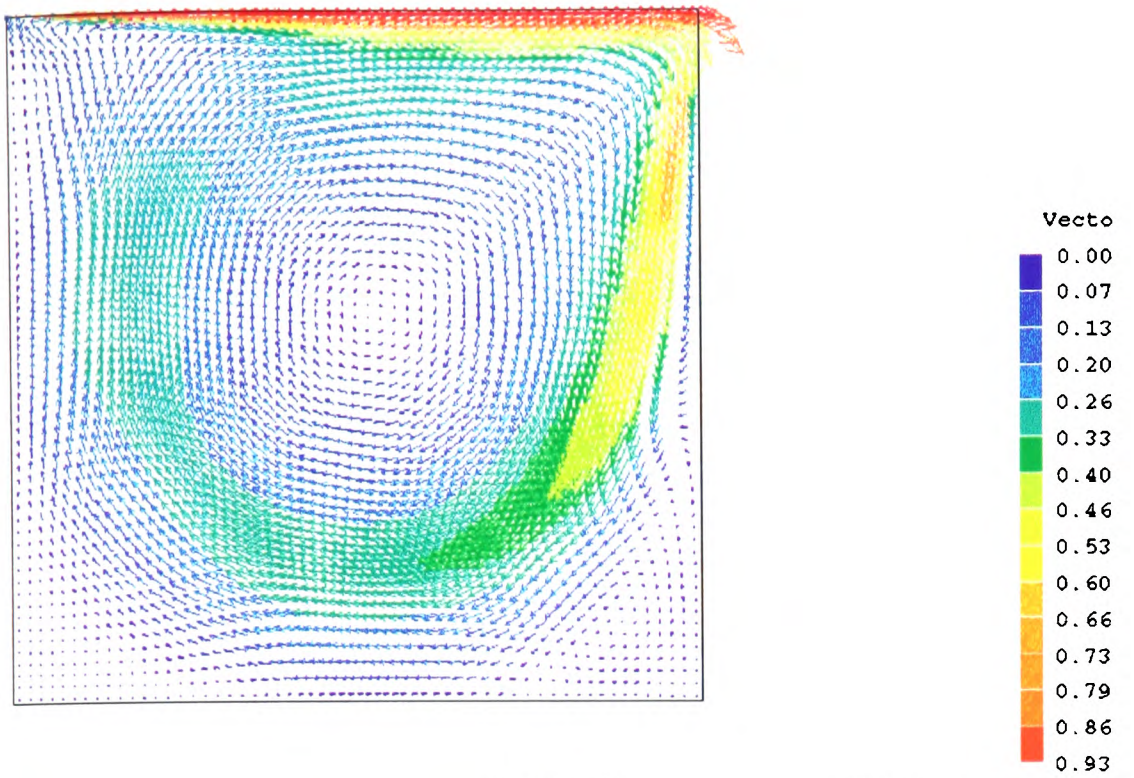


Figure 5.44 Velocity vectors on Grid adapted in PHYSICA

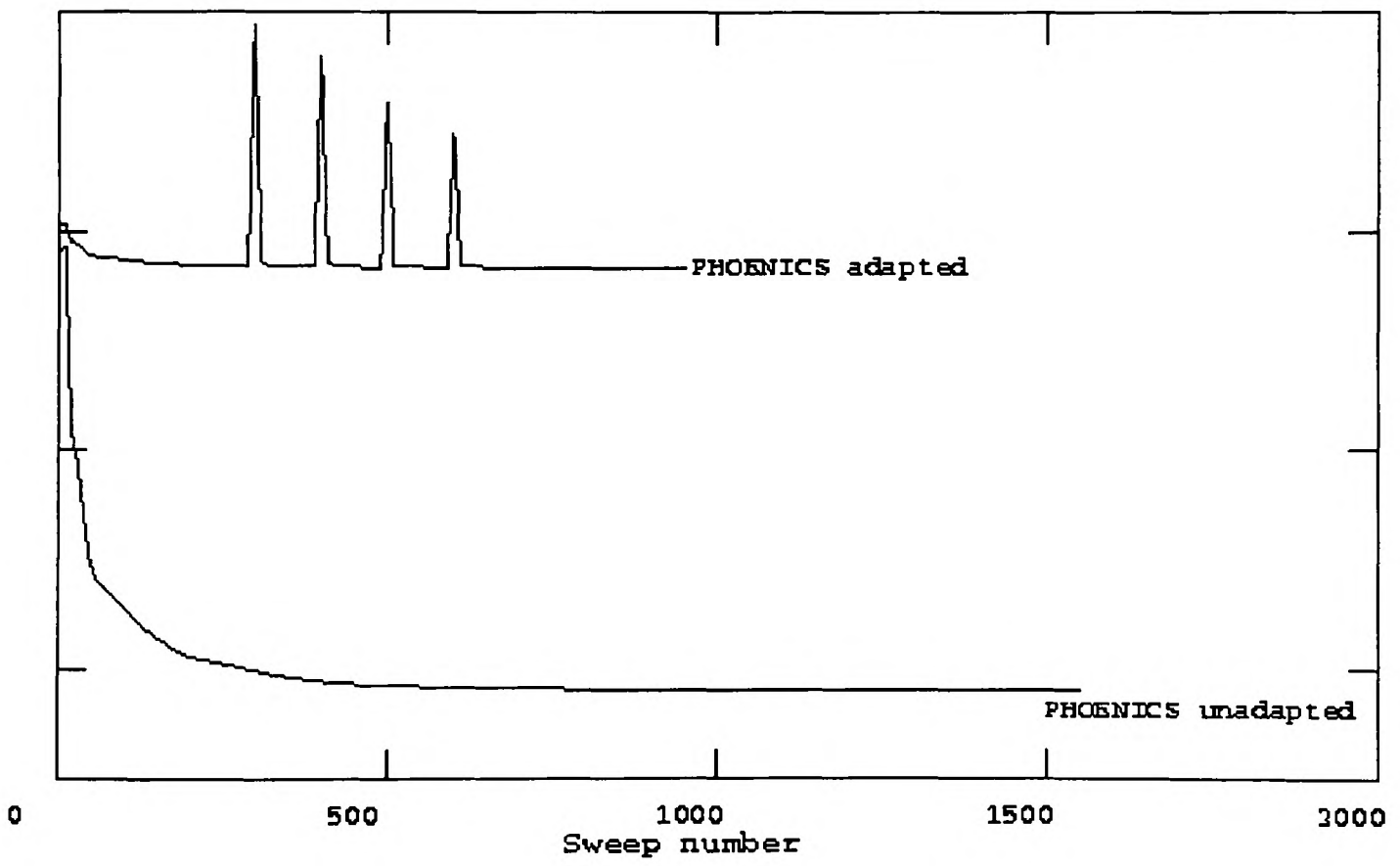


Figure 5.45 Rates of Pressure Convergence for PHOENICS with and without Adaption

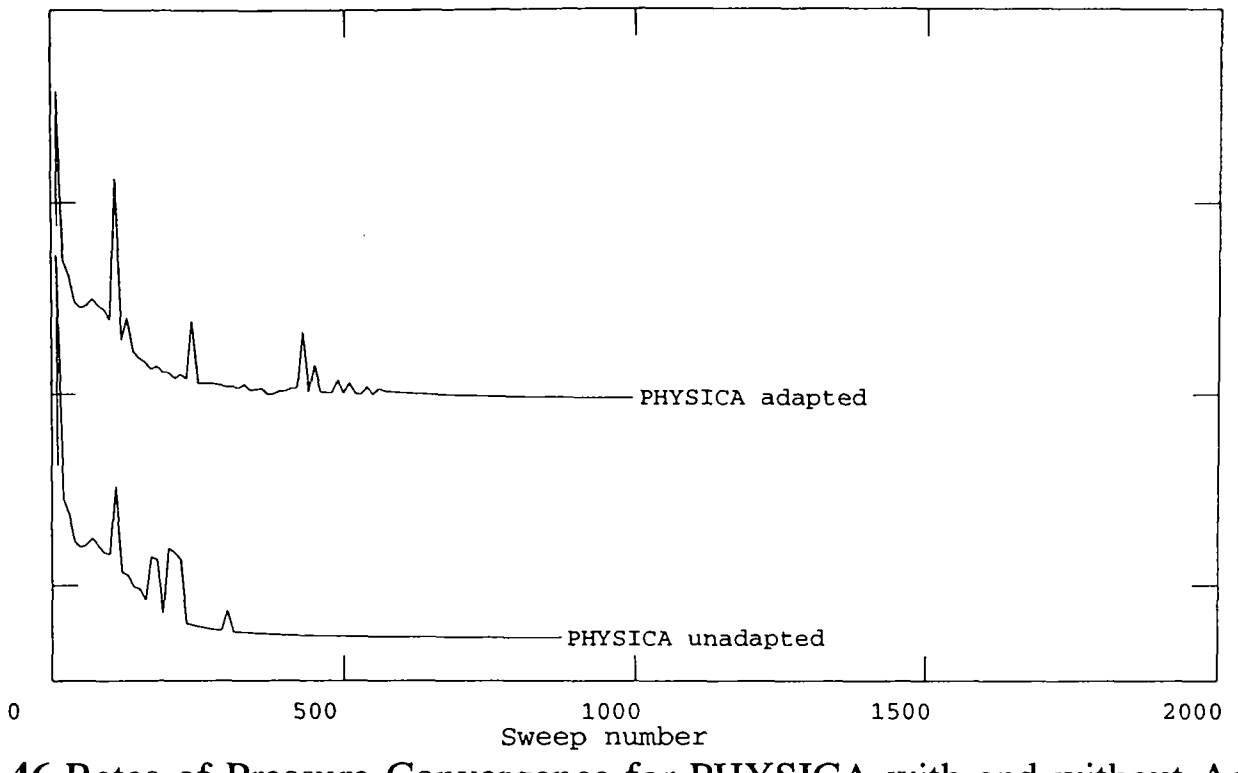


Figure 5.46 Rates of Pressure Convergence for PHYSICA with and without Adaption

5.3.3 10 Degree Supersonic Wedge Intake

This is a 2D supersonic case with two obstructions, a 10° ramp and a lip, that define an idealised jet engine intake.

The flow out from the duct formed by the two obstructions is subsonic. A strong shock wave is formed at the mouth of the duct where the flow decelerates. This case is used to demonstrate both shock capturing and geometry maintaining features of the adaptive algorithm.

5.3.3.1 Boundary conditions

The free stream mach number is 2.0 and the pressure at the duct outlet is constrained to a value calculated using a duct outlet mach number of 0.664.

The computational domain is made up of 128x99 cells, with obstructions set up to represent the lip and the ramp. The geometry is shown in figure 5.47. The curved surface of the leading edge of the lip is modelled by using a high local concentration of distorted cells.

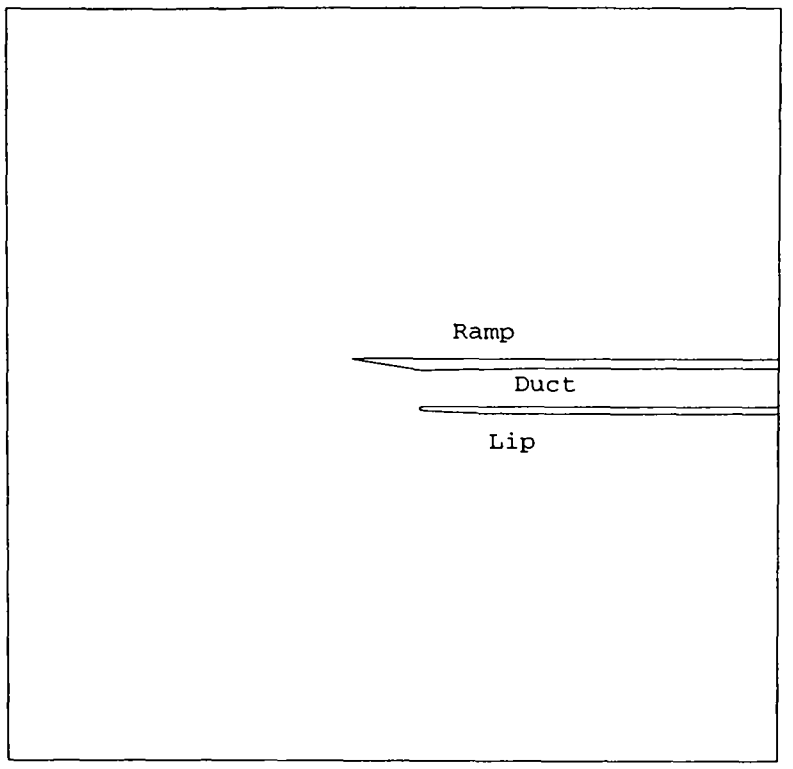


Figure 5.47 Initial Problem

5.3.3.2 Adaption Parameters

Two domains are used. Their positions are shown in figure 5.48 This case involves capturing the two shock waves that dominate the flow. The sole reason for using two domains is to pick up the leading edge shock more clearly, as it is so weak in comparison to the bow wave.

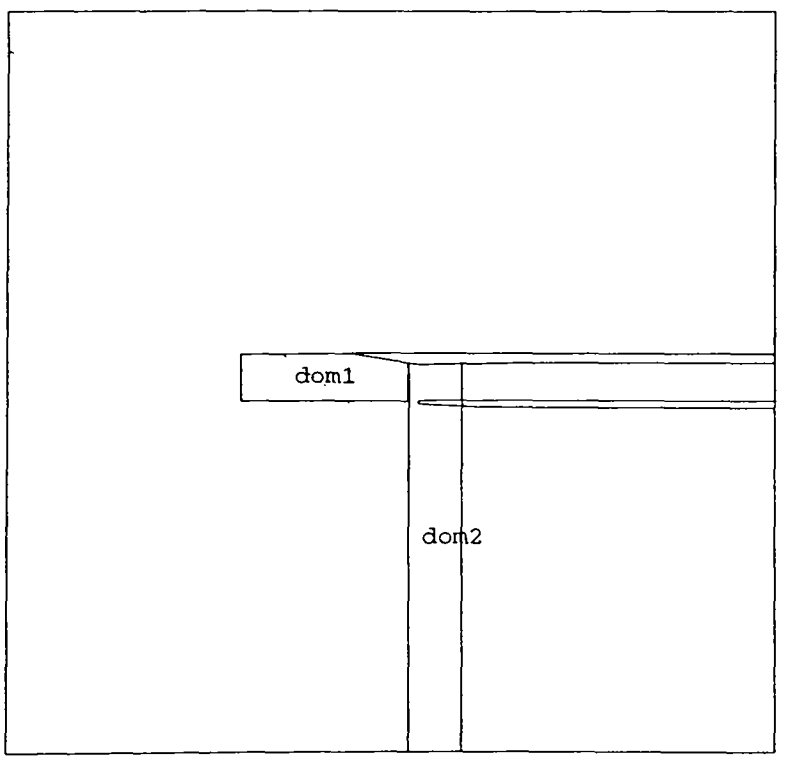


Figure 5.48 Adaption Domains

There is no pressure gradient above the ramp and sufficiently far in front or behind

the entrance of the intake, so any effort in adapting the grid is wasted here.

An allowance is made for the lip by using a 'blank' patch. This is set up to cover the obstruction and a boundary layer of one cell and will block out uncontrolled movement in this region and retain its geometry. At the curved leading edge the local grid distribution is irregular and has to be maintained to retain the integrity of the obstruction. Careful use of curve patches is used to maintain the local grid. The geometry protecting curve and blank patches are shown in figure 5.49

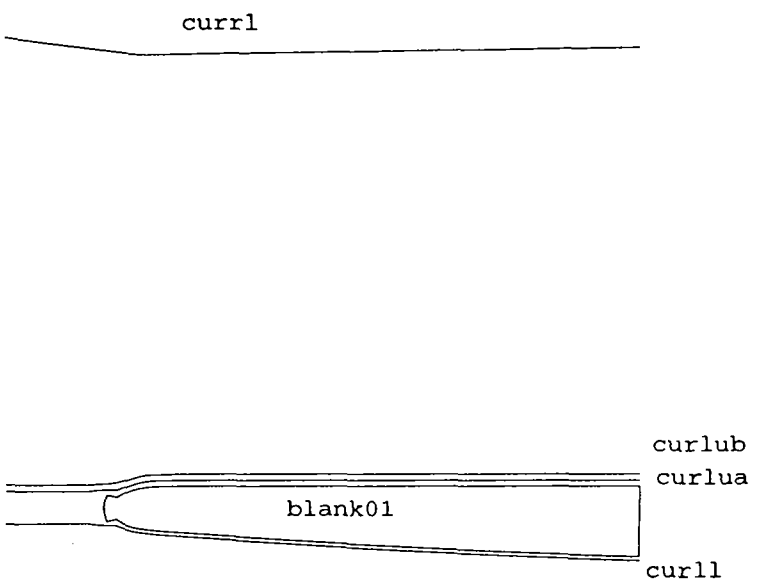


Figure 5.49 Geometry Patches

It is necessary to initialise the variable chosen for adaption in the area within the obstruction, as the adaptive code will read the relevant values from PHOENICS regardless of whether they are solved for or not, and will normalise all values between the limits it reads.

The adaption routines are called at

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
10	120	10	1	1.0

The first domain uses the parameters

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Pressure	$1+\phi_x+0.01k$ 0.05 Weighted exponential smoothing
2	0	3		

whereas the second domain uses the parameters

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Pressure	$1+1\phi_x+0.01k$ 0.05 Weighted exponential smoothing
1	0	1		

The pressure change in the first domain is much smaller than in the second, therefore the equidistribution weight is increased to emphasise the effect of the gradient that is available.

The different schemes available for manipulating the weight function were tested and the one that seemed to work the best was exponential smoothing. Because there is such a strong discontinuity in the flow care must be taken to prevent the grid from overlapping in the shock.

5.3.3.3 10 Degree Supersonic Wedge Intake Results

Figures 5.50 and 5.51 show that the resultant grid moves well and shows the shapes of the shock waves clearly, though there are still some problems at the leading edge of the lip where the lack of a strong gradient or any other confinement allow the Laplace term to push the grid out, causing some skewed cells at the start and end of this region. These appear to have little effect on the results however. The geometry of the obstructions is maintained.

The pressure contour plots show that the point at which the main shock wave connects

to the lower surface of the ramp appears to have moved back to the lowest point of the ramp. The contours are noticeably tighter for the weak leading shock and around the front edge of the lip.

The most striking difference between the results on the unadapted and adapted results come however on the plots (Figures 5.54, 5.55 and 5.56) taken on lines of grid cells at the centre of the duct and at the upper and lower surface. There may be some discrepancy as to the exact comparison as the adapted grid lines will be in different positions from the original, but what all show is that the pressure on the adapted grid rapidly settles to the value of the outlet pressure condition whereas the standard grid does not. The spike where the prediction overshoots though still present is smaller and the oscillations much reduced.

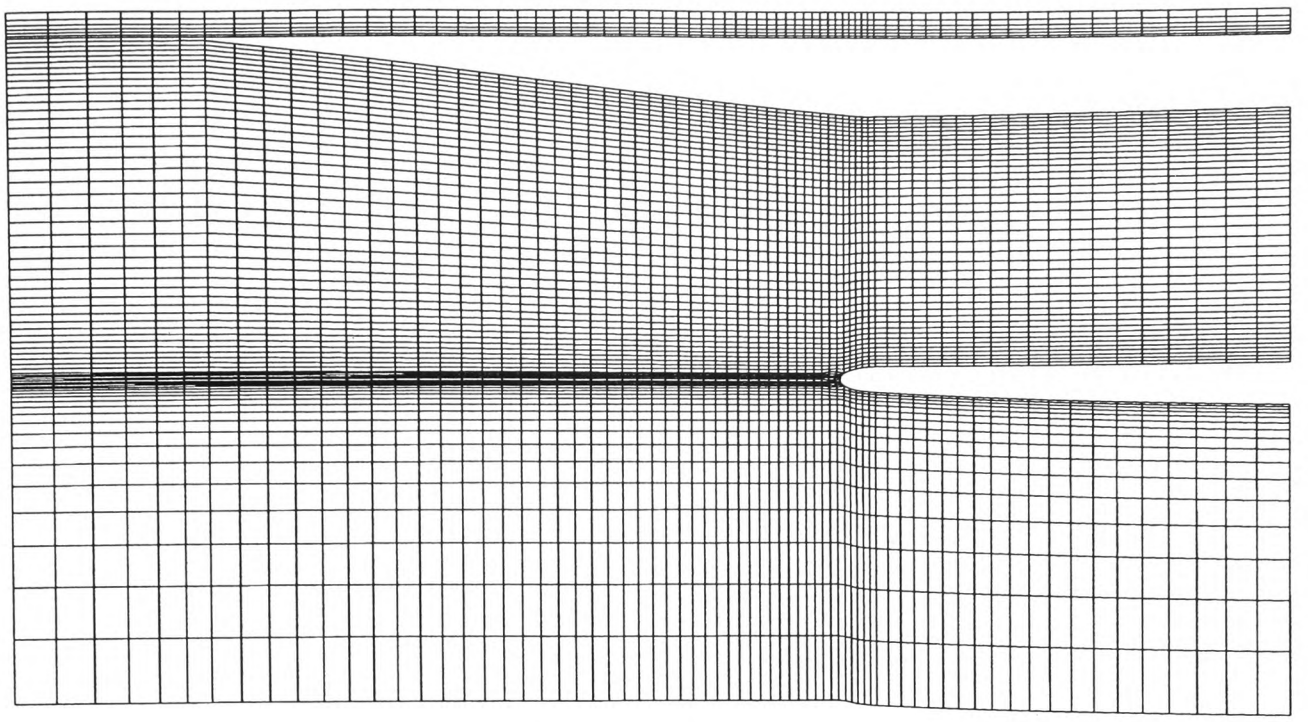


Figure 5.50 Original Grid

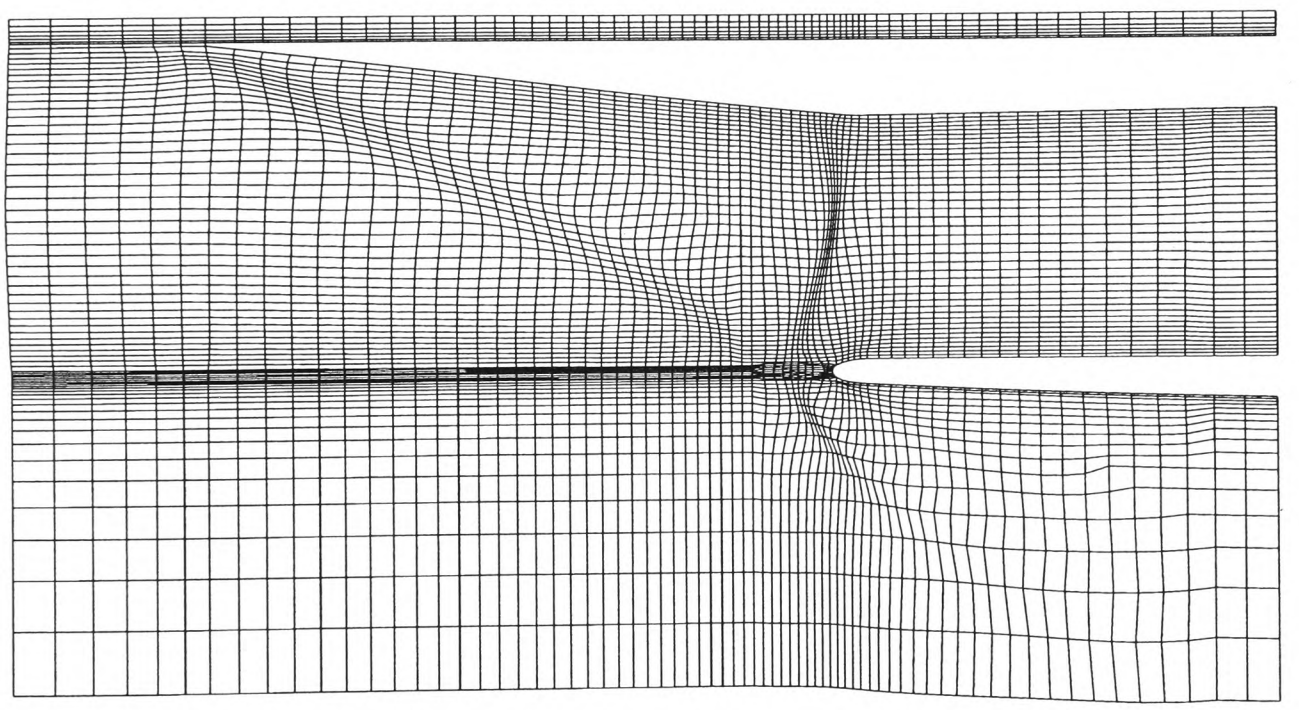


Figure 5.51 Adapted Grid

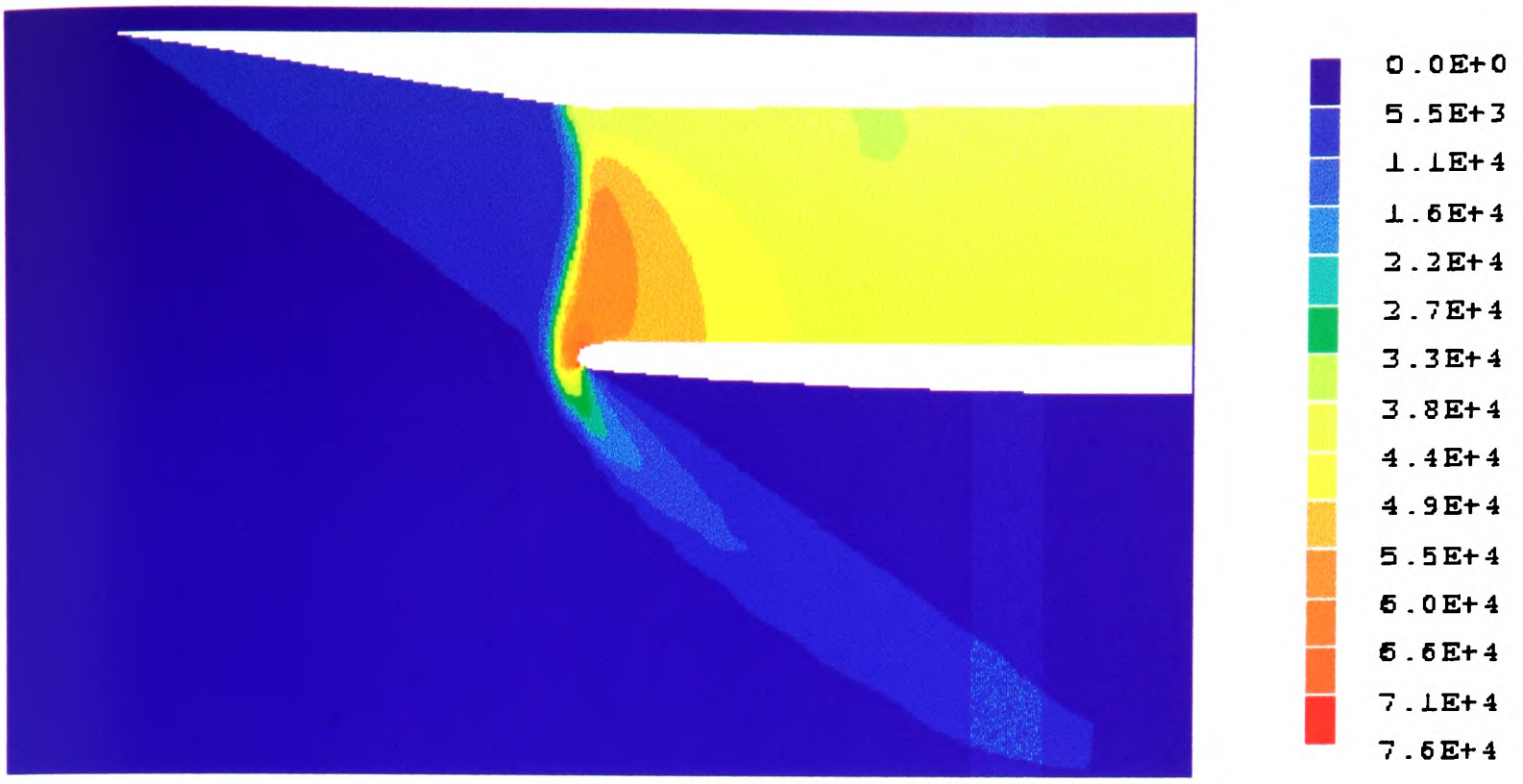


Figure 5.52 Pressure Contours on Fixed Grid

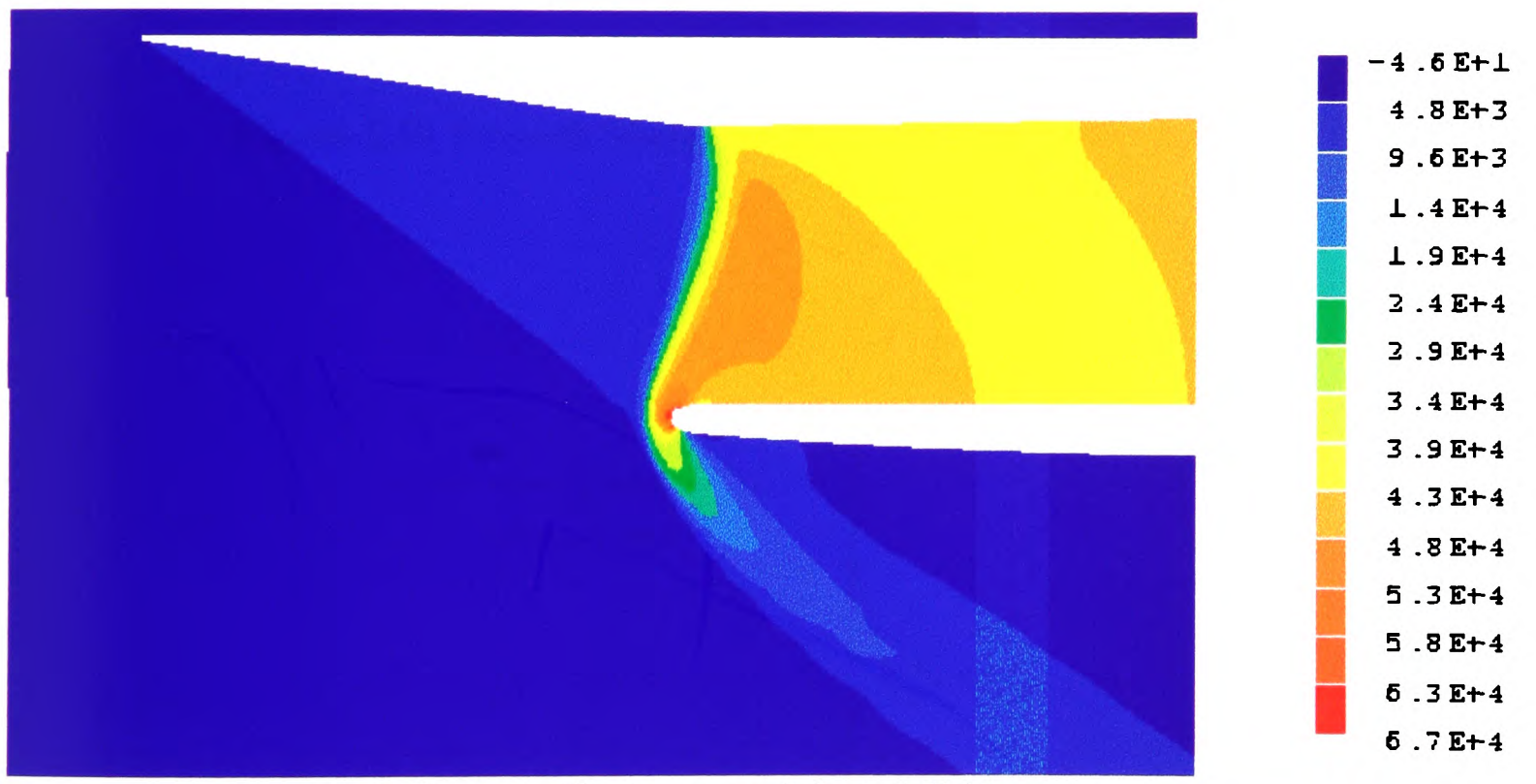


Figure 5.53 Pressure Contours on Adapted Grid

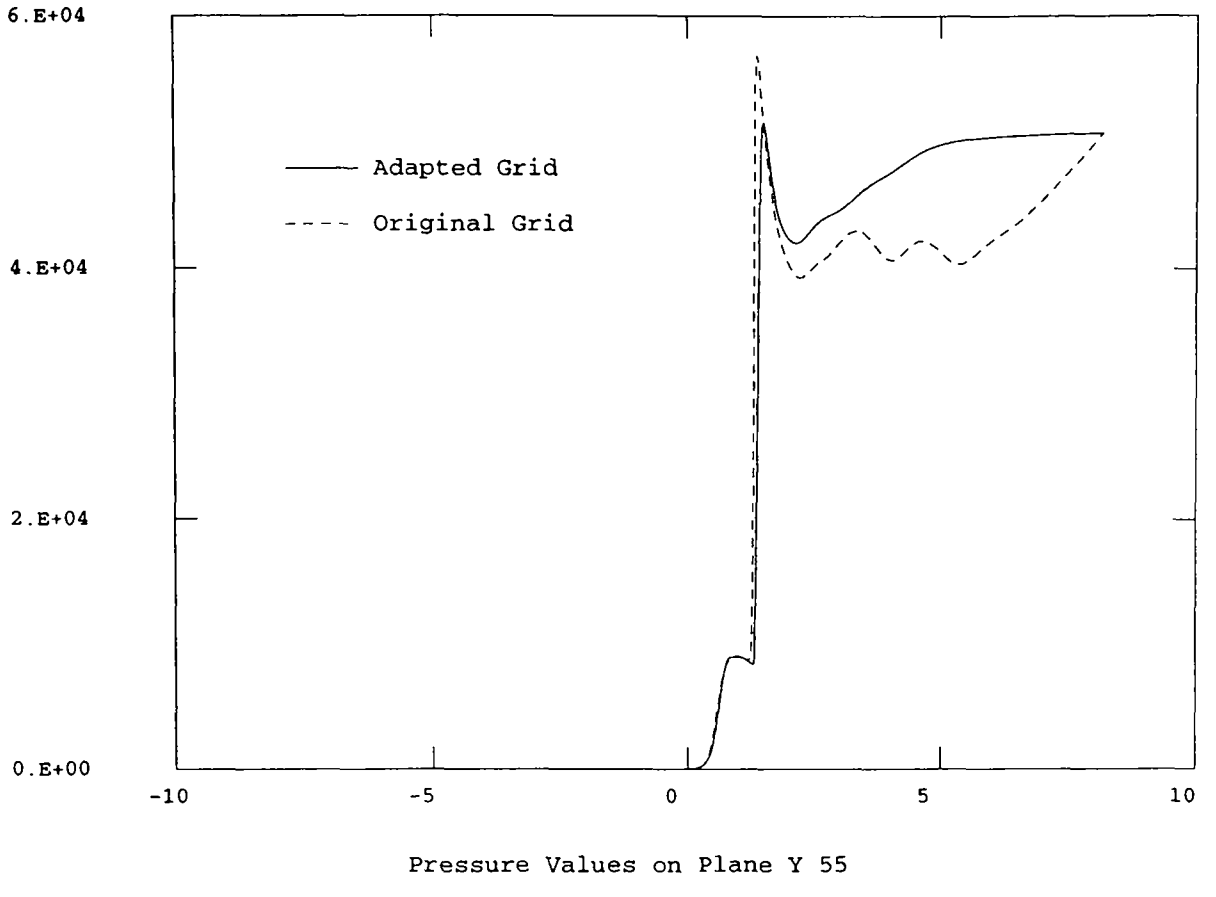


Figure 5.54 Pressure Variation on Centre Line Through Duct

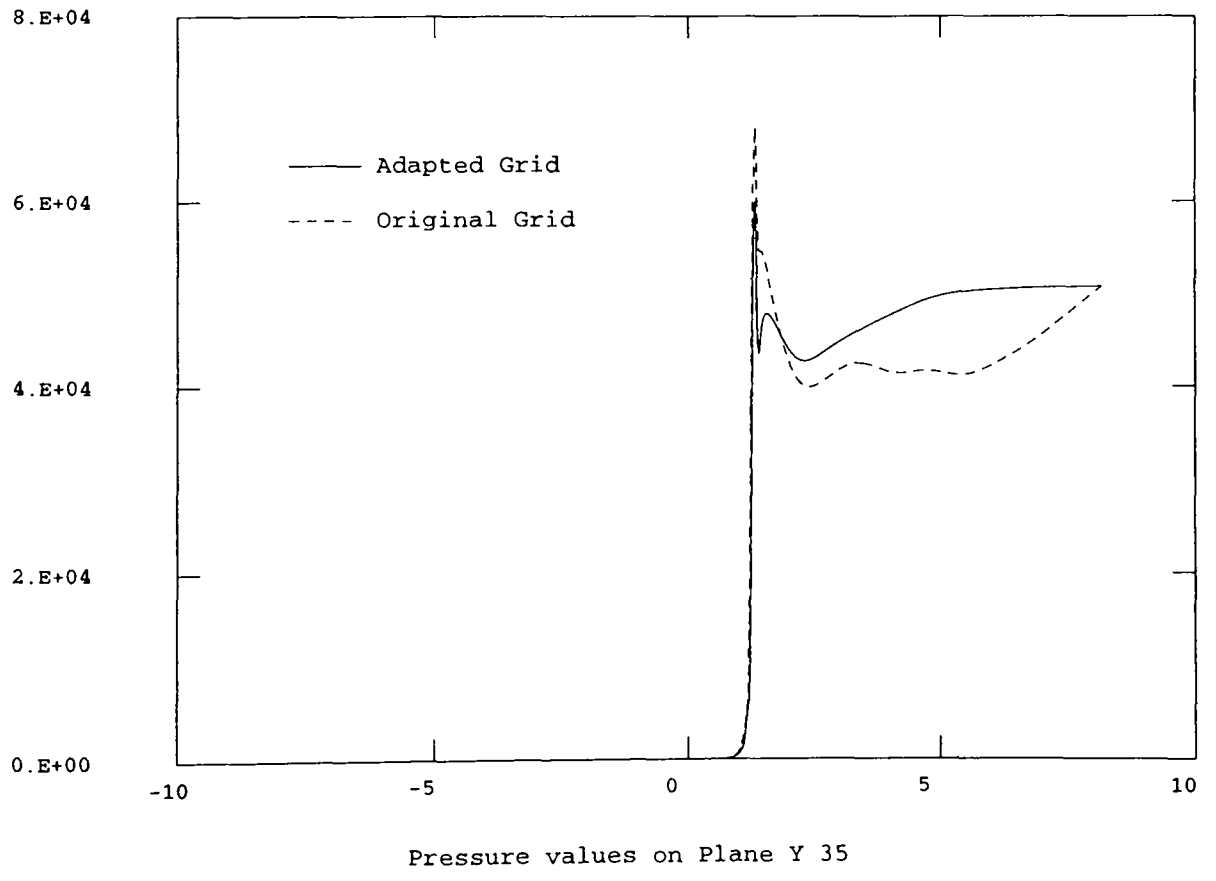


Figure 5.55 Pressure Variation on Line Running over Lip

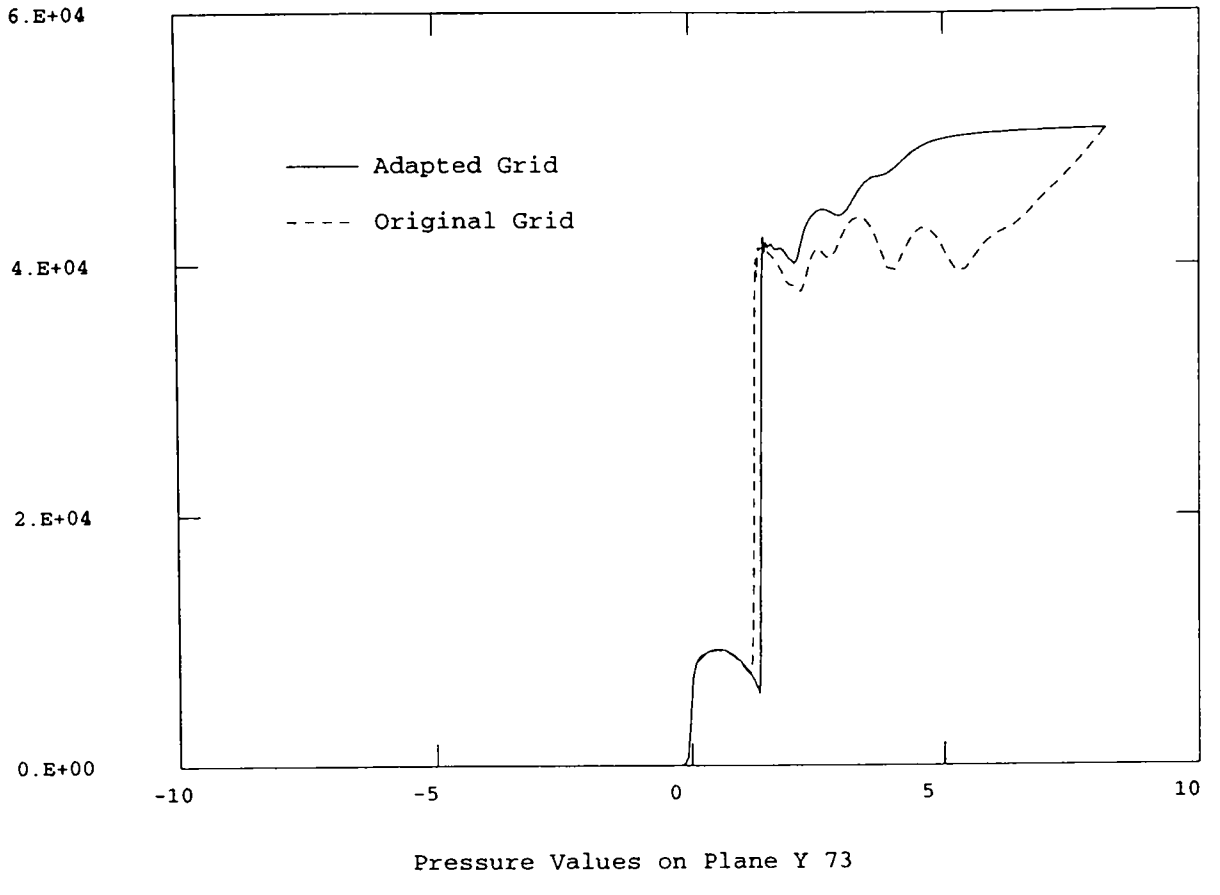


Figure 5.56 Pressure Variation over Line Underneath Ramp

5.3.4 Laval Nozzle

The De Laval nozzle case concerns transonic flow in a convergent-divergent nozzle with the exit conditions modified to set up a standing shock wave in the divergent part of the nozzle.

The adaptive grid will allow better resolution of the shock through increasing the local grid concentration and therefore reducing the change in pressure over individual cells.

5.3.4.1 Boundary Conditions

The inlet Mach number is 0.5. To create the standing shock wave the outlet pressure is set to 0.75 of the stagnation pressure, taken from tables for the design conditions as 7.83e5. The position of the shock can be calculated analytically at 60% chord (Patel, Pericleous and Baldwin 1995).

The geometry is set up with the throat situated at one-third of the length of the duct and uses symmetry. The adapted case was run with 60x5 cells. In addition the same case was run without adaption, and also with a finer mesh of 100x7 cells.

The problem is run to convergence.

5.3.4.2 Adaption Parameters

One domain was used covering the whole problem area. To retain the character of the original grid the grid cells were only allowed to move in the axial direction, sliding along the original grid lines. These lines were maintained by defining curve patches along all axial grid lines.

The main adaption parameters are

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
50	350	50		

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Pressure	$1+2\phi_x+0.02k$
2	1	2		x^2 power law smoothing

There is a strong gradient to move the grid to but too great a loss in grid orthogonality and smoothness can quickly lead to the loss of the solution, so care has to be taken to keep movement smooth.

5.3.4.3 Laval Nozzle Results

This case has previously been used to demonstrate adaption by Patel et al. (1995) and it is identical except for the method of adaption.

The solution converged after 1285 sweeps in a time of 100 seconds running on a Pentium P100 computer, compared to times of 61 seconds for the unadapted solution to converge at 1117 sweeps, and 231 seconds for the fine grid to converge at 1759 sweeps.

Figure 5.57 shows the fine, adapted and unadapted grids. Figure 5.58 shows the final axial grid density of the three cases. Figure 5.59 shows the axial pressure, which is uniform perpendicular to the axis.

The final adapted grid shows how a relatively small movement can have a beneficial effect upon the results, which are better than for the same sized unadapted grid. Figure 5.59 shows that for the adapted grid the pressure change is much steeper than for the unadapted grid and starts in the same area as the fine grid.

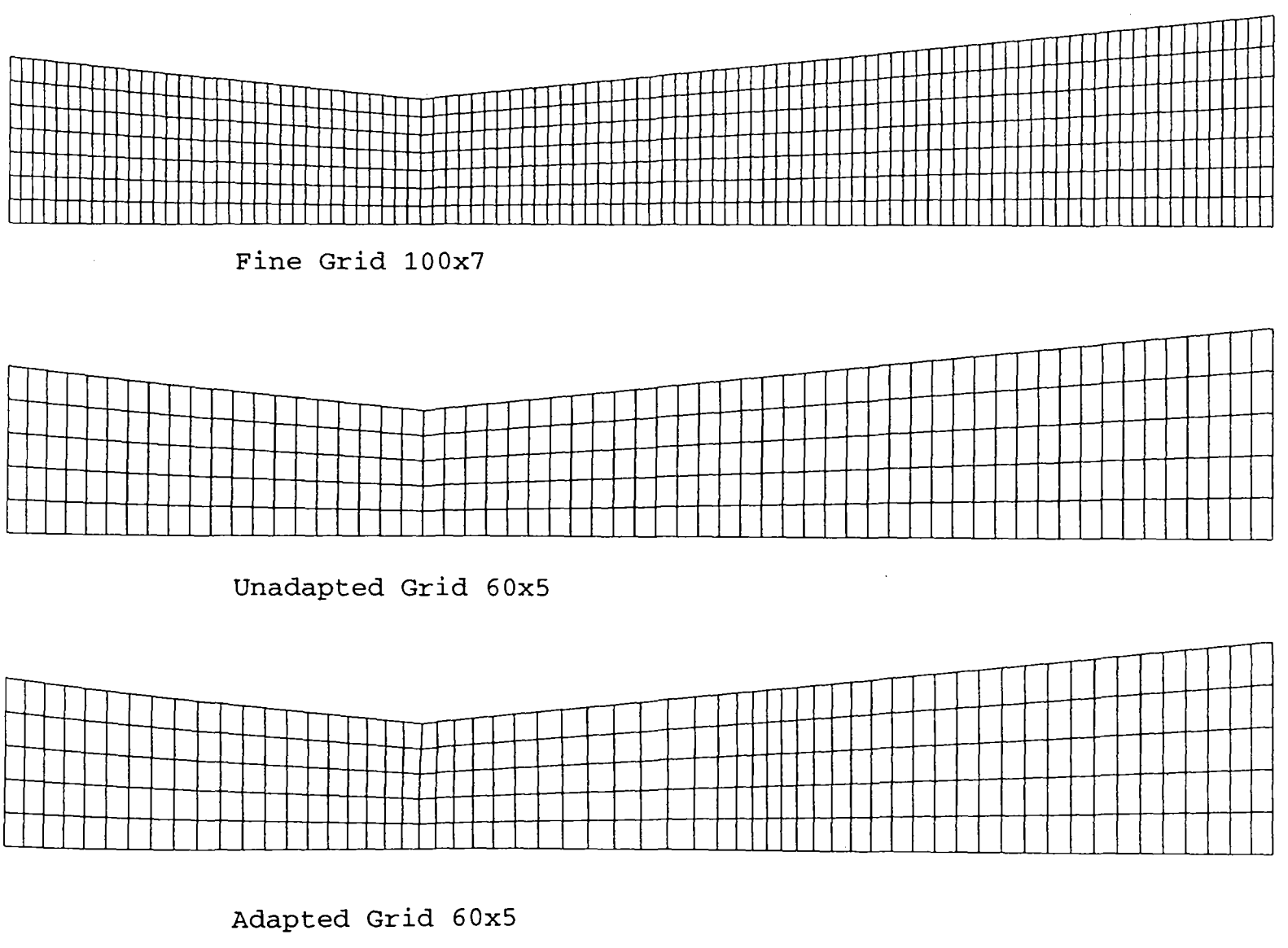


Figure 5.57 Laval nozzle grids

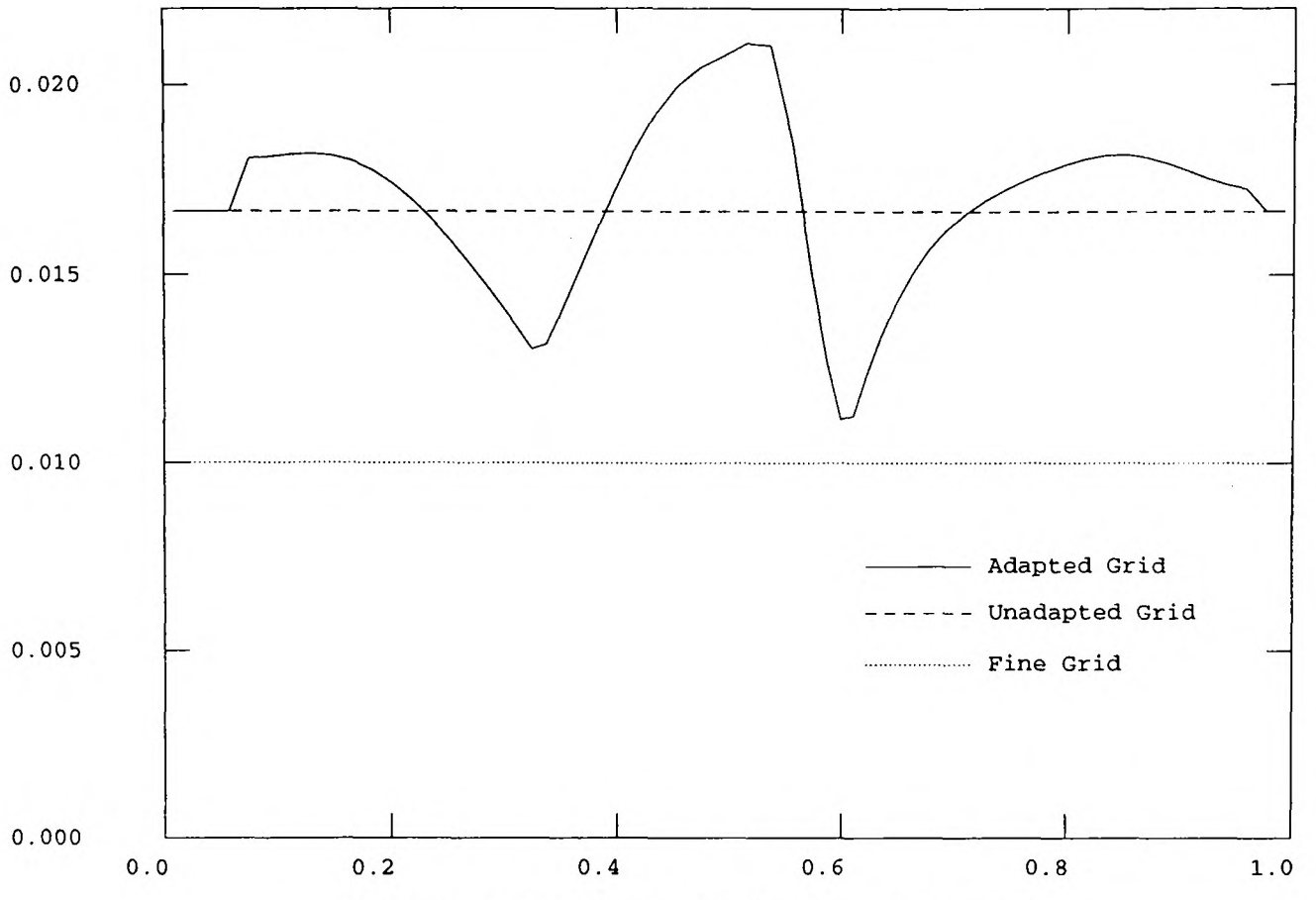


Figure 5.58 Grid cell density along axis

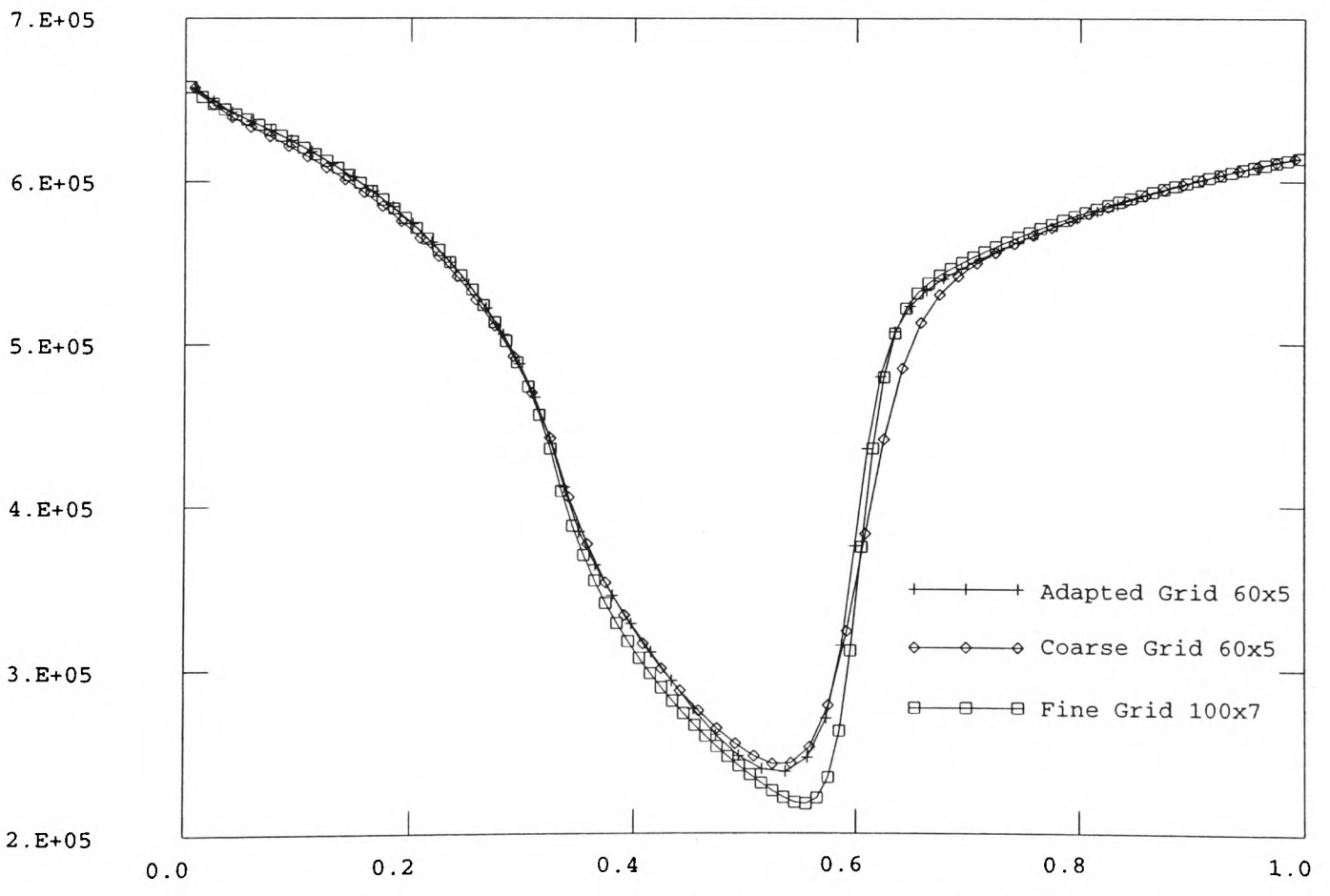


Figure 5.59 Pressure along axis (Pressure against distance)

5.3.5 Supersonic Flow Through A Wedge Cascade

This case consists of a completely supersonic flow through a wedge cascade. There is a shock wave from the leading edge which is cancelled by its reflection at the upstream corner of the cascade, giving a uniform parallel flow between the corners, a weak expansion wave off the downstream corner, and two compression waves at the trailing edge. Numerical smearing means that the initial shock wave is not completely cancelled and reflects down the narrow part of the flow domain.

The adaptive algorithm is used to better resolve the shocks throughout the model.

5.3.5.1 Boundary Conditions

The cascade has a chord of four units, with the leading ramp, at 18.5°, measuring two units, and the parallel section and trailing ramp measuring one each.

The adapted case was run with a grid measuring 112x30 cells. For comparison the same case was run without adaption and also with a finer mesh of 168x45 cells, this being the fineness of grid suggested by Palacio et al (1990) required for the solution to become grid independent.

The Mach number at the inlet at the low end of the domain is set to 3.0. The outlet has a fixed pressure calculated from gas-dynamic theory.

Cyclic boundary conditions are applied upstream and downstream of the cascade.

The flow is isentropic.

All three cases were run for 1000 sweeps, which was sufficient for all of them to converge.

This case is based on the PHOENICS library case 523 and Palacio et al (1990).

5.3.5.2 Adaption Parameters

One adaption patch was used to cover the whole domain. The strong shock wave on the leading ramp makes the adaption easier, as the pressure gradients here are strong.

The adaption parameters used are given in the following tables

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
50	500	50	5	1.0

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Pressure	$1+1\phi_x+0.01k$ x^2 power law smoothing
0	1	1		

The Poisson term was used in preference to the Laplace term to balance the equidistribution weight as the Laplace term causes the grid lines over the ramps to bow out away from the constriction.

The power law smoothing allows the effect of the shock waves on adaption to be propagated out into the grid.

Grid nodes were allowed to slide along the upper and lower boundaries using curve patches. The automatic curve breaking feature was used to break the lower curve patch into five, covering each section of the domain.

5.3.5.3 Wedge Cascade Results

The use of adaption involved an additional overhead of about 25% in CPU time, 1000 sweeps of the coarse grid taking 991 seconds running on a Sun SPARC 5 workstation against 1230 seconds for the adapted grid and 1862 seconds for the fine grid.

Figure 5.60 shows the fine, adapted and unadapted grids. Figure 5.61 shows the pressure contours for the three cases. Figure 5.62 shows the pressure over the lower surface of the wedge.

The final adapted grid clearly shows the location of the primary shock, its reflection, and the weak expansion waves from the upstream corner of the cascade, as the cells contract and align themselves with this important flow phenomena. The grid cells also concentrate at the downstream corner of the cascade.

The main drawback of the adapted grid is that the shock wave on the leading edge pulls grid nodes away from the first ramp, something which accounts for the difference in the pressure distribution over this surface as calculated by the different methods. It can be seen from the plots of the pressure contours for the three runs that the adapted grid produces much sharper gradients over its shocks than the other two cases. However, the reflection of the first shock wave now concentrates slightly downstream of the upstream corner on the lower surface, rather than being smeared across it, which accounts for the increase in magnitude of the shock waves inside the parallel section and the difference in the position of the main pressure peak in figure 5.62.

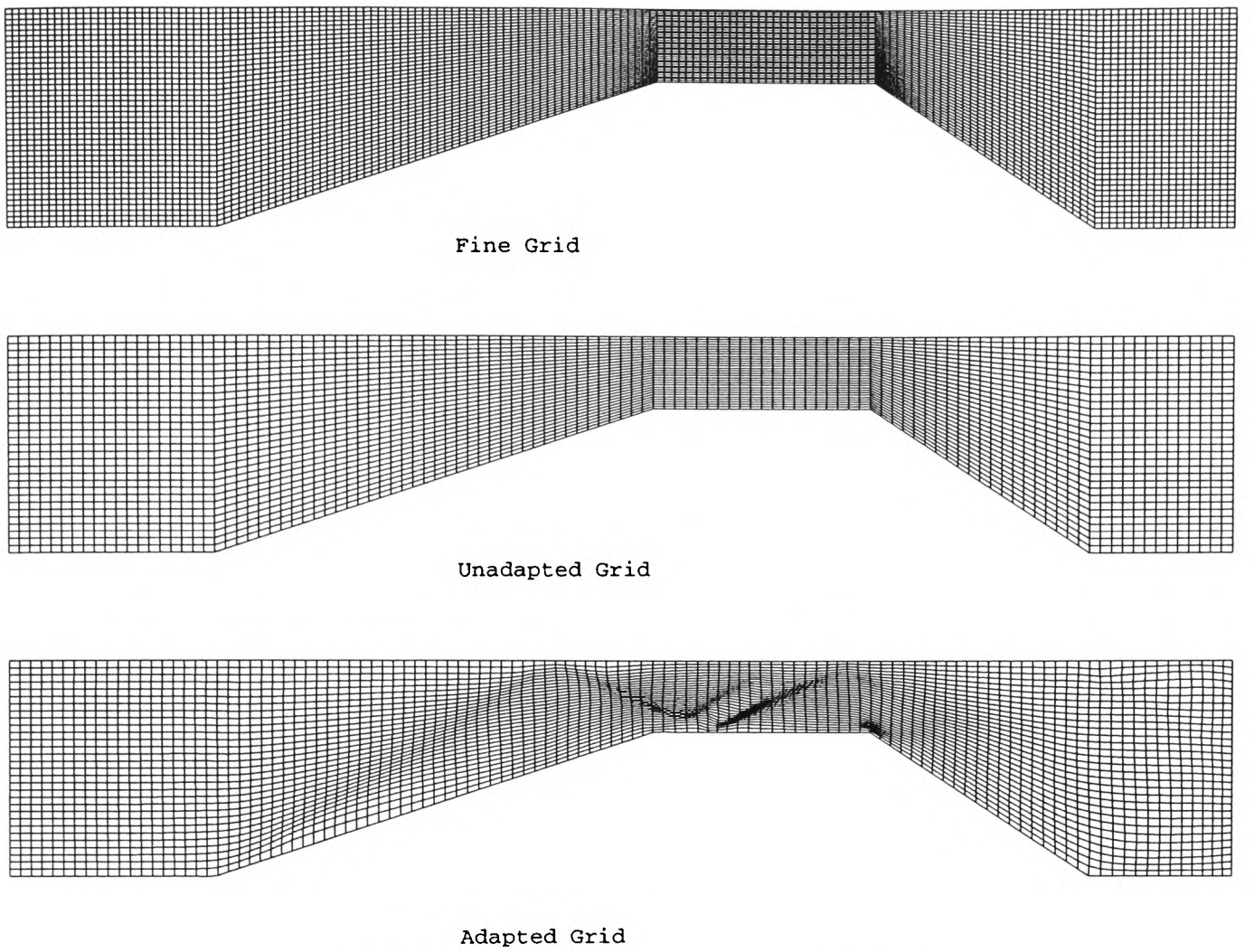


Figure 5.60 Wedge Cascade Grids

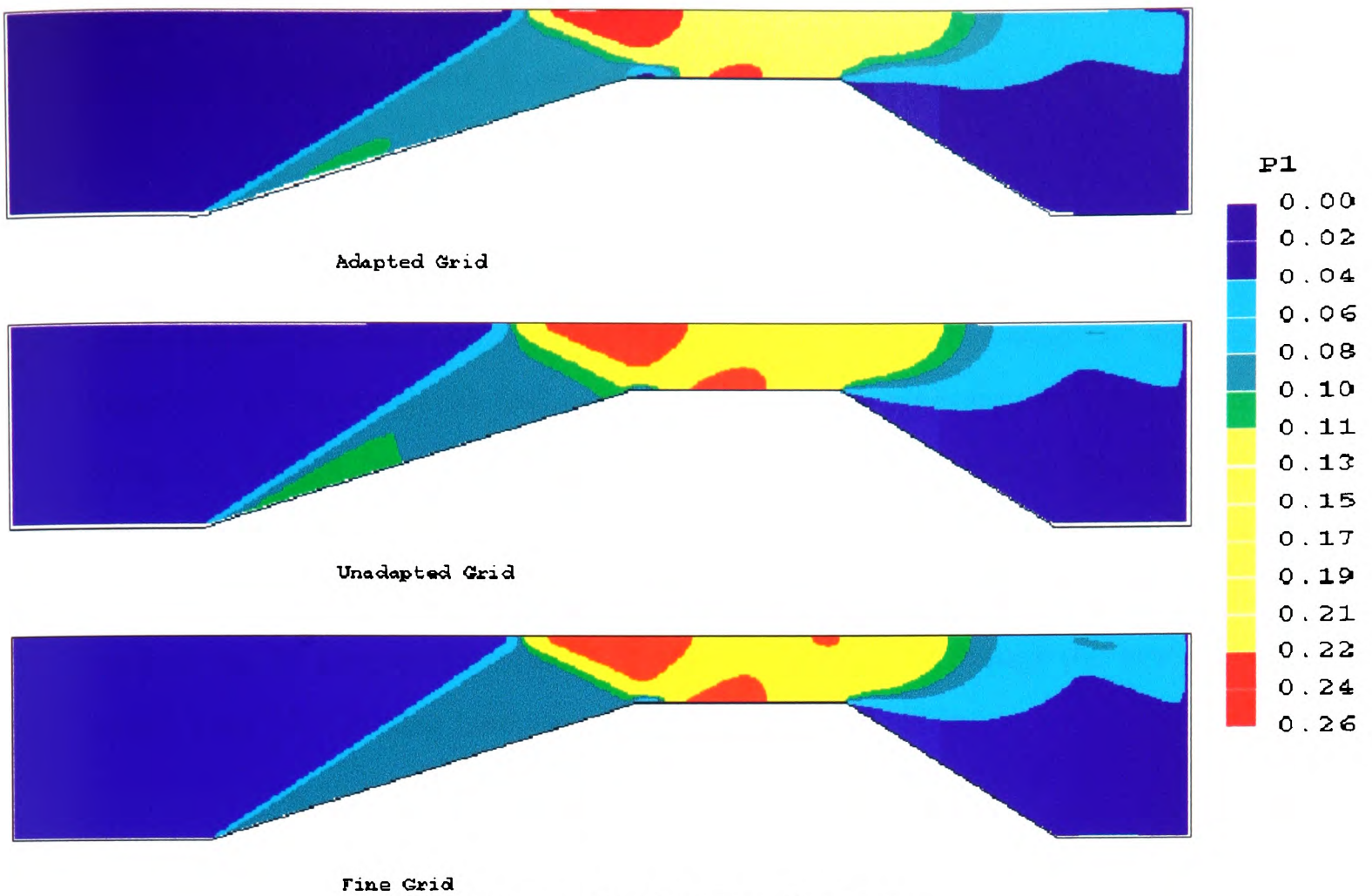


Figure 5.61 Pressure Contours

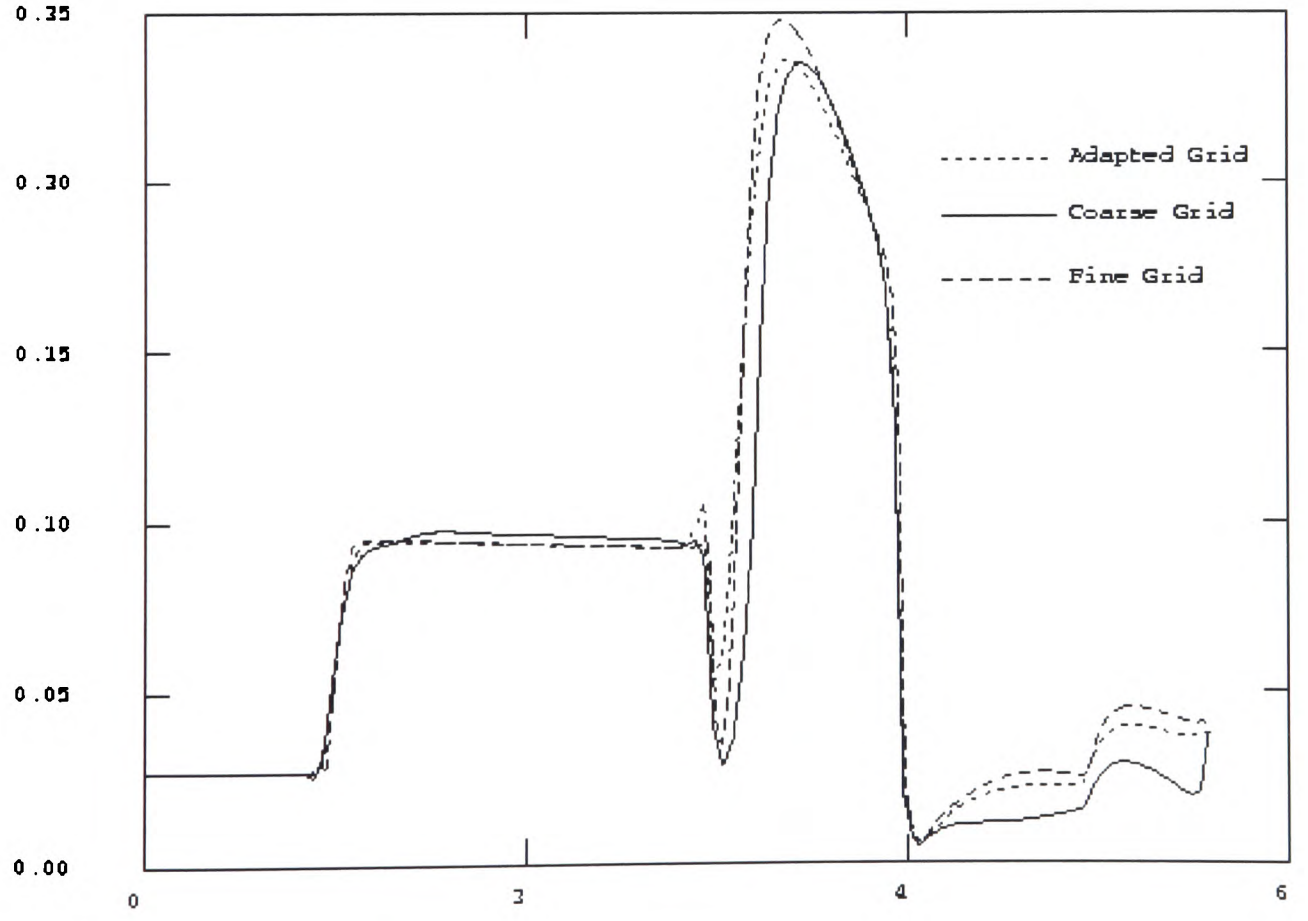


Figure 5.62 Pressure Over Lower Surface (Pressure against distance)

In figure 5.62 non dimensionalised units are used for pressure and distance. The pressure is expressed as P/P_o where P_o is the stagnation pressure.

5.4 Closure

This chapter has successfully demonstrated the use of the adaptive algorithm on a range of two dimensional test cases. However it is necessary to tailor the algorithm using the weight function and relaxation to get the best results in each case. There must also be some sort of solution feature that can be used to produce a weight. The cases which generate the most movement are those which involve strong shock waves, such as the 10 Degree Supersonic Wedge Intake (5.3.3), though the low speed driven cavity case (5.3.2) also behaves well with its strong eddy.

Large grid distortion through adaption is not always necessary to get the best results, and may instead cause instability in the solution. The Laval nozzle case (5.3.4) shows how a low level of smooth movement can produce an improved solution.

CHAPTER 6: 3D VERIFICATION TESTS

6.1 Introduction

The aim of this chapter is to present examples of the use of the adaptive algorithm on a range of three dimensional cases including both simple mathematical functions and complex fluid flow.

There are difficulties in presenting large three dimensional cases as well as comparing them to results produced elsewhere. For this reason only three cases are presented here.

6.2 Three Dimensional Test Function

The following case was run independently of PHOENICS, adapting the grid to a prescribed mathematical function.

6.2.1 Circular Function

This is a test case that was used by Matsuno and Dwyer (1988). who used the same function to show how their method worked in one, two and three dimensions. The function is circular and similar to the pill box case in section 5.2.5.

Two separate sets of results are produced to show the effect of the method of weight smoothing using a power law (section 2.10.3.3).

6.2.1.1 Boundary Conditions

The function the grid is adapted to is

$$\phi = [1 + sign(r - r_c)\{1 - e^{(-a\rho^2 + 1/2)}\}]/2 \qquad \textbf{(6.1)}$$

Where

$$r = \left((x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2\right)^{\frac{1}{2}}$$

$$sign(r - r_c) = 1 \qquad r \geq r_c$$

$$sign(r - r_c) = -1 \qquad r < r_c$$

$$\rho = \frac{1}{(2a)^{\frac{1}{2}}} + |r - r_c|$$

$$a = 500, \quad r_c = 0.25$$

Contours of the function through the centre of the domain are shown in figure 6.1.

The case is solved on a 30x30x30 cell uniform grid over a unit cube.

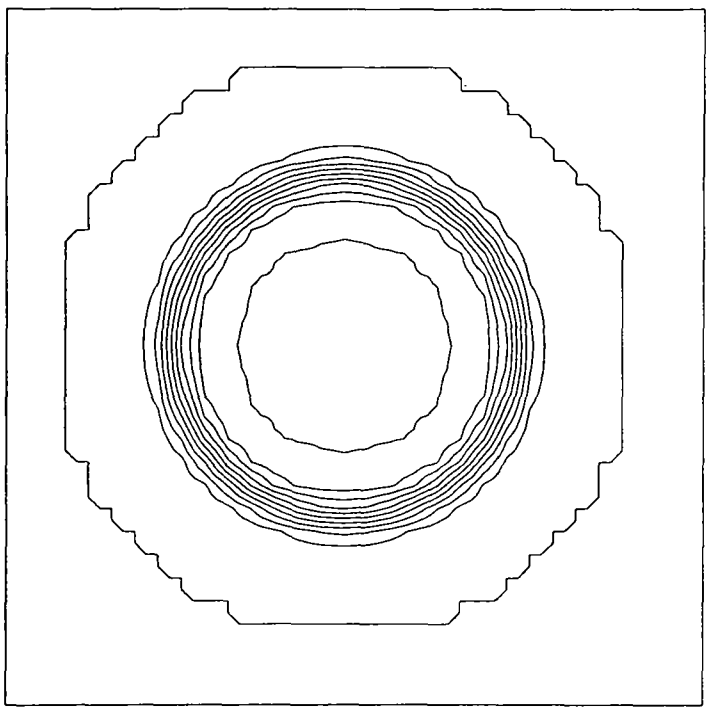


Figure 6.1 Contours of test function

6.2.1.2 Adaption Parameters

The grid is adapted over the whole domain. The points on the domain boundaries are allowed to slide but show little or no movement due to the circular shape of the function. In both cases the grids are moved ten times, with a relaxation factor of 0.9.

In case 1 the parameters used are

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Prescribed	$1+\phi_x+0.01k$
2	0	1		

In case 2, with weight smoothing, the parameters are

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Prescribed	$1+\phi_x+0.01k$
3	0	5		x^2 power law smoothing

In the second case surfaces have been set up over the three grid planes that pass through the centre of the grid. This is necessary to enforce some symmetry on the grid, as the weight smoothing algorithm fits modified weights to an asymmetric curve based on which point it determines that the location of the peak of the function lies. In the case of a semicirclular or similar function a point at its peak could move to either side unless it is constrained to the centre. Otherwise neighbouring points at right angles to the function may move in different directions, giving a badly skewed mesh. As the majority of real flow cases are not symmetric in this way and the cells are still concentrated at the areas deemed important this is not a major problem. This asymmetry can be shown in the figures 6.2 and 6.3 which present the function and the weight at the end of the run for the two cases.

The use of power law smoothing allows the grid to remain stable with a much higher weight on the Equidistribution term in the LPE equation. This allows the grid to move faster.

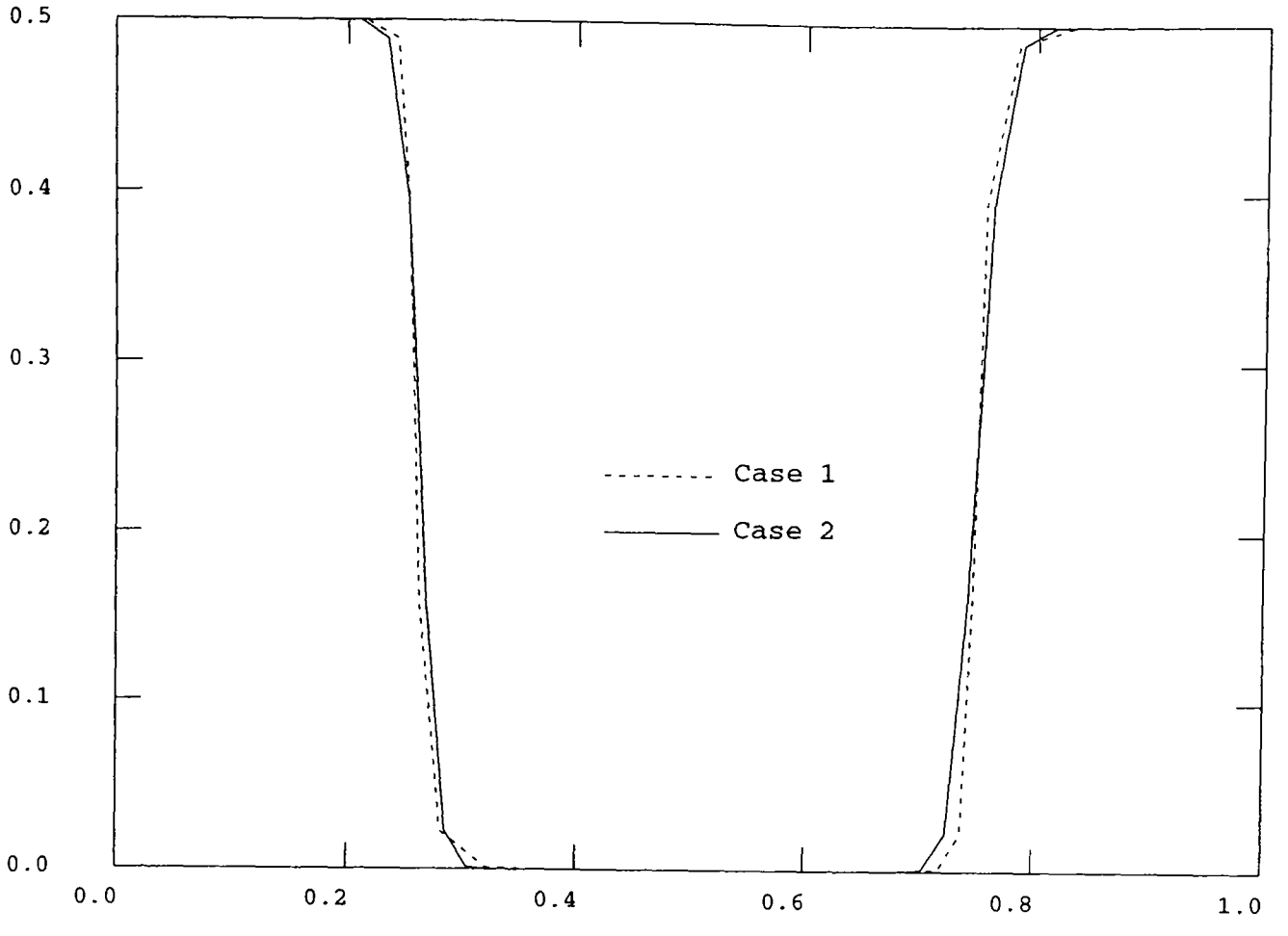


Figure 6.2 ϕ against distance across centre of grid

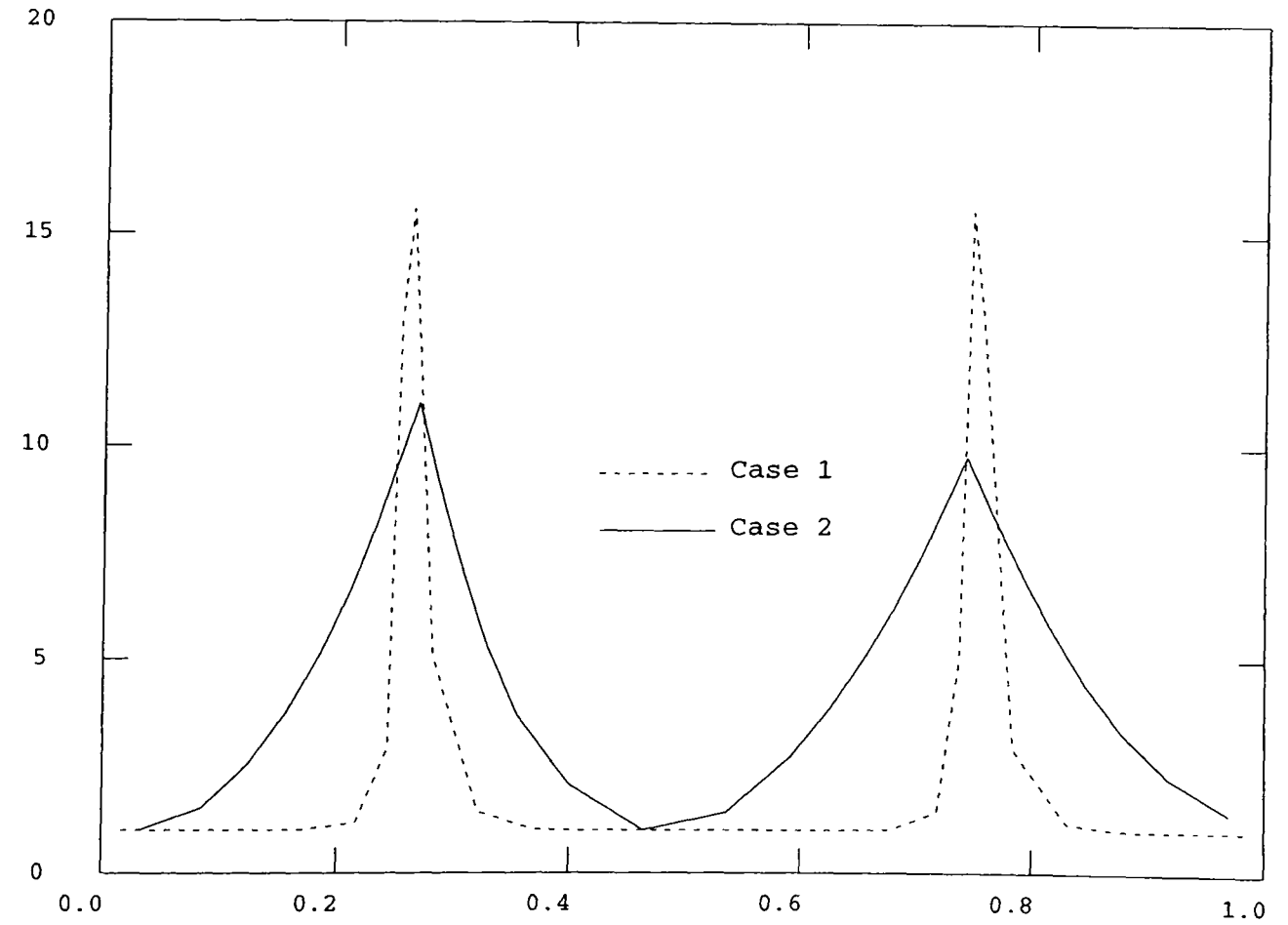


Figure 6.3 Example weight function along centre line of grid

6.2.1.3 Results

Figures 6.4 through 6.9 show different views of the final grids produced by the parameters used in the two cases. Figures 6.10 through 6.12 show the same views of the grid produced by Matsuno and Dwyer (1988).

The results from case 1 are similar to those produced by Matsuno and Dwyer (1988) with coarser cells adjacent to the fine region and little influence on the parts of the grid farthest from the function contours. For case 2, in comparison, the influence of the function has been propagated throughout the grid and the most coarse cells are those farthest away from the function contours. The exceptions are the eight corners where the weight is not propagated because it is based on grid lines which do not cross the function. Case 2 has a smooth variation in cell size from the fine region to the far parts of the grid, though there is some asymmetry, as mentioned earlier.

The grid from Matsuno and Dwyer (1988) took 38 iterations to converge.

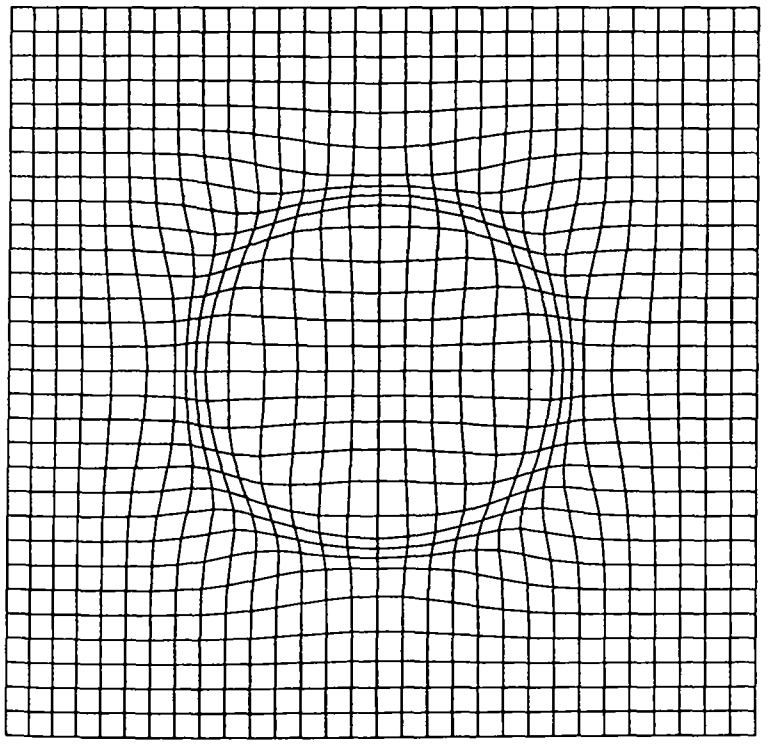


Figure 6.4 Case 1

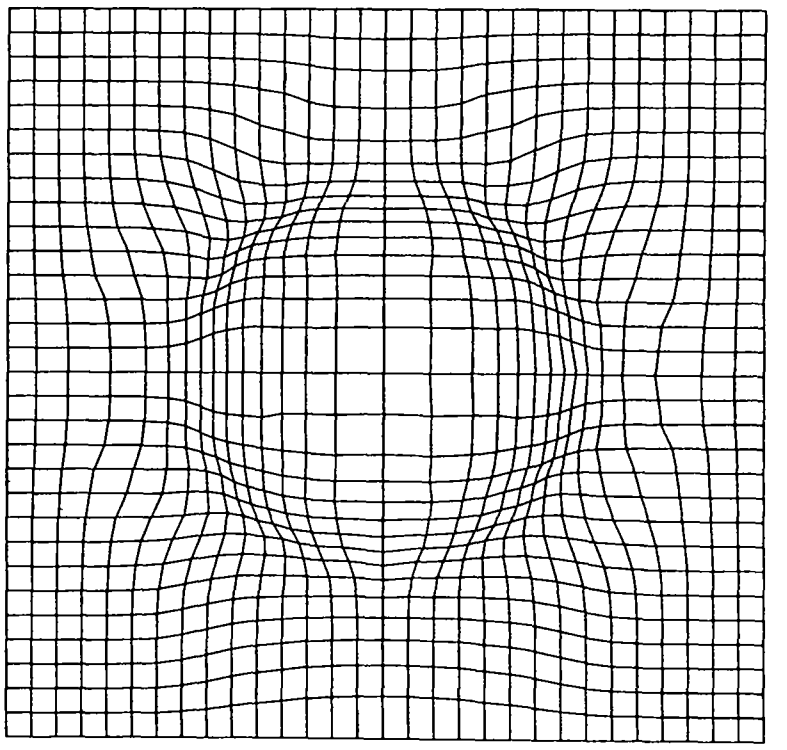


Figure 6.5 Case 2

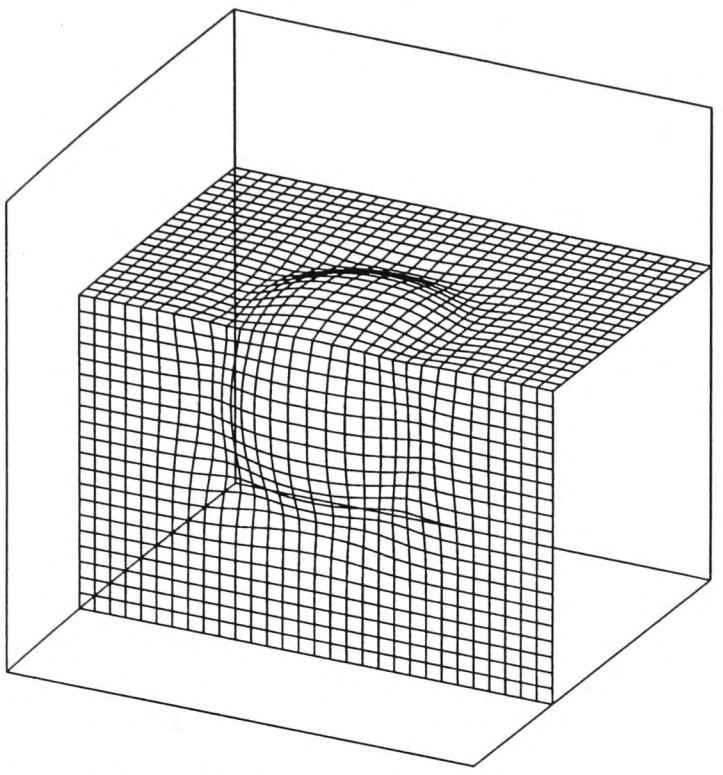


Figure 6.6 Case 1

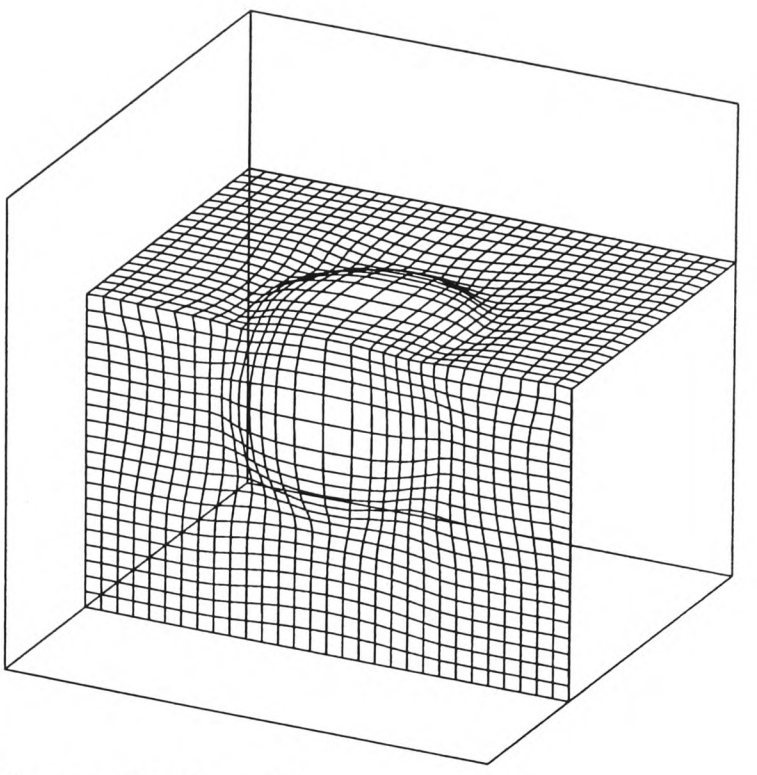


Figure 6.7 Case 2

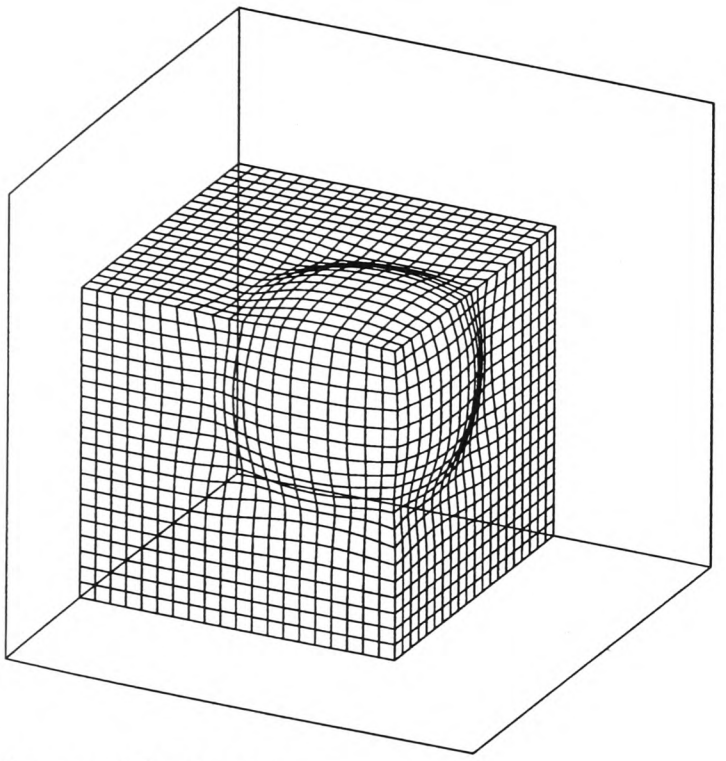


Figure 6.8 Case 1

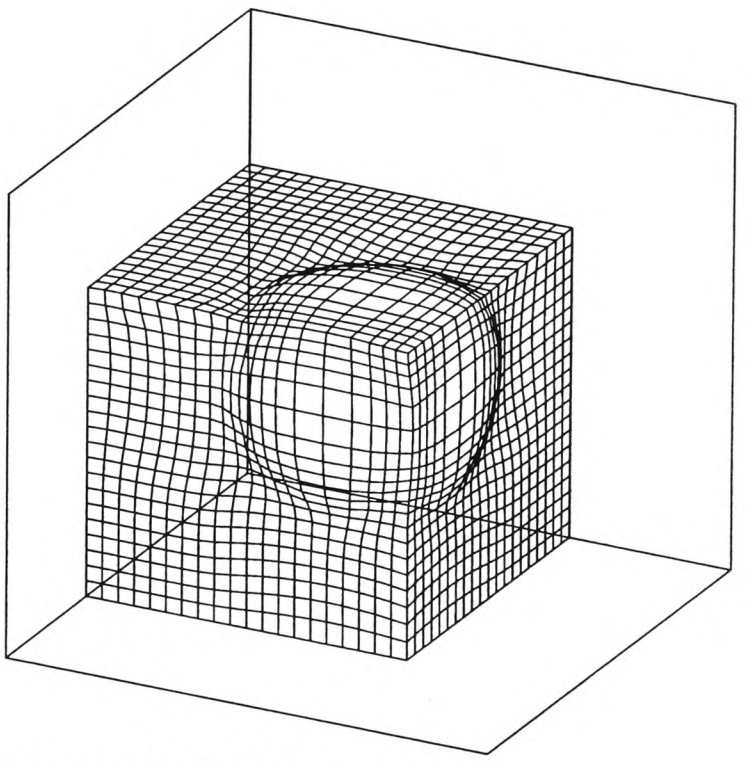


Figure 6.9 Case 2

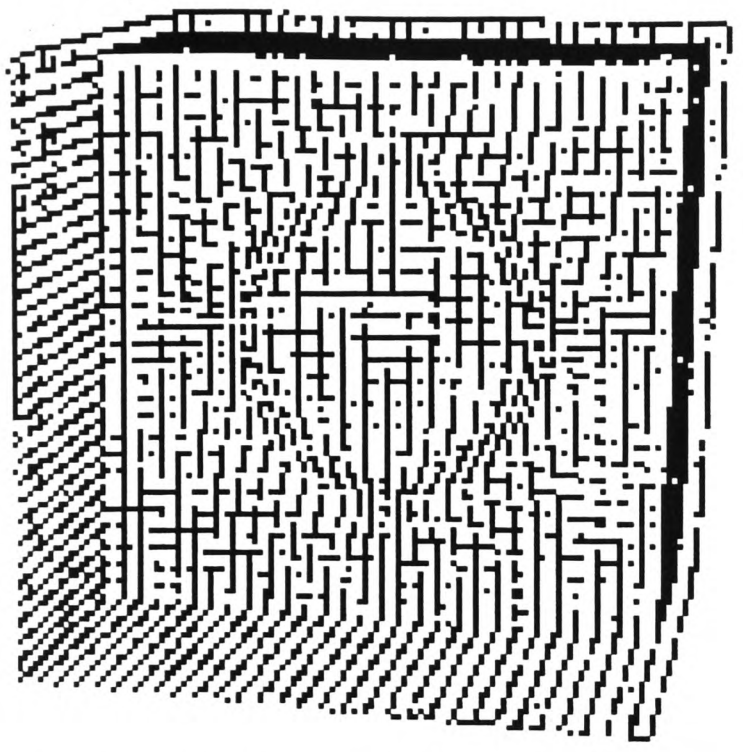


Figure 6.10 Matsuno and Dwyer (1988)

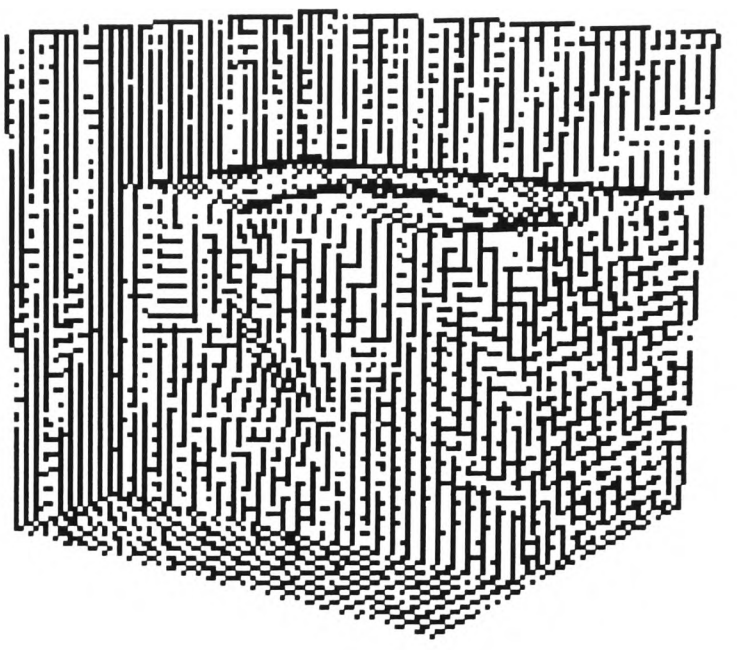


Figure 6.11 Matsuno and Dwyer (1988)

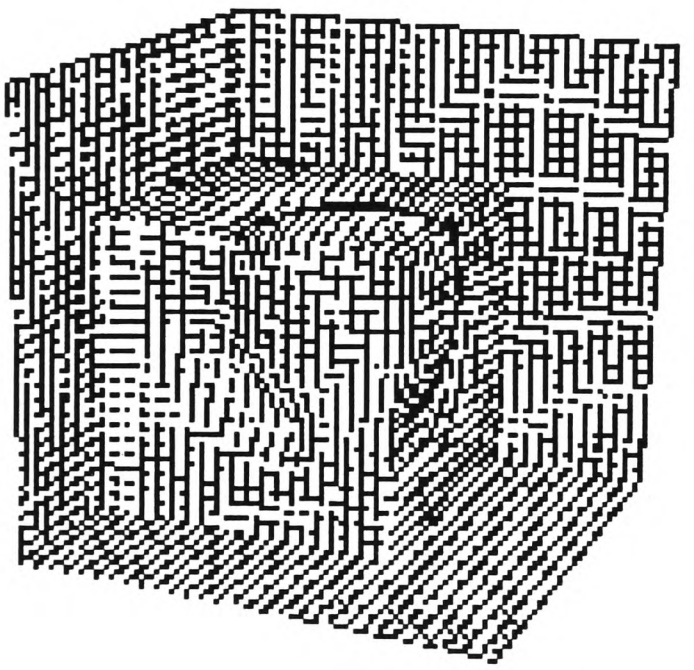


Figure 6.12 Matsuno and Dwyer (1988)

6.3 Three Dimensional Flow Examples

The following cases were run using PHOENICS version 2.0 on Sun SPARC workstations.

6.3.1 3D Skewed Wedge Cascade

This case consists of isentropic supersonic flow over a double ramp.

This case is a modified version of the two dimensional wedge cascade (see case 5.3.4), stretched out into a third dimension with the central section skewed at a small angle to give angled ramps. The leading angled ramp in turn will produce an angled shock wave.

This case is used to demonstrate the ability of the adaptive code to use surface patches in the same way as the curve patches demonstrated in Chapter 5.

6.3.1.1 Boundary Conditions

The geometry of the problem in the first Y-plane is identical to the two dimensional case (5.3.4), with the centre section swept back at an angle of 13.5°. The geometry is shown in figure 6.13.

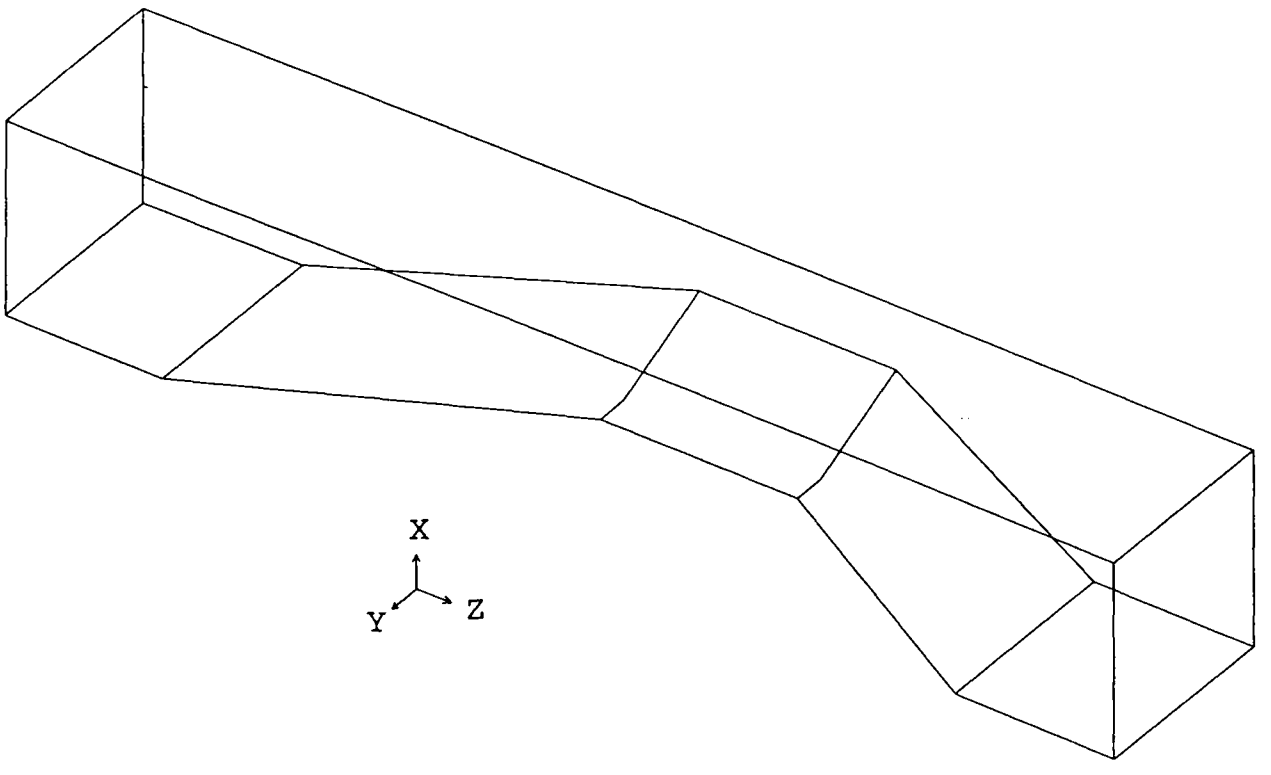


Figure 6.13 Wedge geometry

Cyclic boundary conditions are applied upstream and downstream of the problem. The velocity at the inlet, which lies at the low end of the domain is calculated based on

a Mach number of 3.0. The outlet, which lies at the high end of the domain, has a fixed pressure calculated from gas-dynamic theory.

The domain is defined by 10 by 6 by 28 cells in the i, j, k planes respectively.

The problem was run over 500 sweeps, though this was probably unnecessary as it converges fairly quickly.

6.3.1.2 Adaption Parameters

One adaption patch is used to cover the whole problem domain, with surface patches set up on the north, south, west and east faces. The automatic surface breaking routines cut the west surface that covers the ramps into five sections to allow its geometry to be maintained. The breaks are marked in figure 6.19. In addition curve patches are set up along the four edges that run from low to high on the domain. In other words the grid at the inlet and outlet conditions and on the lines which define the geometry of the west face are fixed.

The other adaption parameters are described in the following tables.

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
20	200	20	5	0.9

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Pressure	$1+\phi_x+0.01k$ x^2 power law smoothing
1	2	2		

The problem converges quickly which allows frequent adaption. There is a series of definite changes in pressure throughout the domain which makes the adaption easier.

6.3.1.3 3D Skewed Wedge Cascade Results

The total movement in each of the three coordinate directions x, y and z was 30.3%, 1.5%, and 11.5% respectively, with the grid fairly well converged. The adaption involved an overhead of approximately 60% for the fifty iterations of the adaptive solver carried out. The adaptive algorithm is more expensive for three dimensional problems than for two dimensions as the number of terms in the solved equations is related to n^3 as opposed to n^2. The adaption overhead could be reduced by using less iterations with terms in the LPE equation optimised to make the grid move faster.

Figures 6.14 and 6.15 show the Y-planes of the original and adapted grid. The adapted grid has picked up the main shock on the front ramp, its reflection onto or near the start of the constriction, the second reflection from the start of the constriction and the start of an expansion wave near the end of the constriction. Grid points have moved on all surfaces and the problem geometry has been maintained

The adapted grid has moved to become more orthogonal than the original. Most of the east west lines are considerably straighter than in the original grid. This orthogonality has had an effect upon the smoothness of the west surface over the ramp where the grid, movement has been restricted, as can be seen in figure 6.19, though this should not be large enough to damage the results.

Figures 6.16 and 6.17 show pressure contours in the Y-plane going through the centres of the six cells of the original and adapted grid. Throughout the grid the pressure contours are tighter and the features they describe are better defined.

Figure 6.18 shows surfaces of constant pressure at the same value for the original and adapted grid. The shock wave is resolved over a much shorter distance which must improve the accuracy of the simulation.

Figure 6.19 shows where the bottom surface is cut into five sections by the surface cutting algorithm (3.2.3).

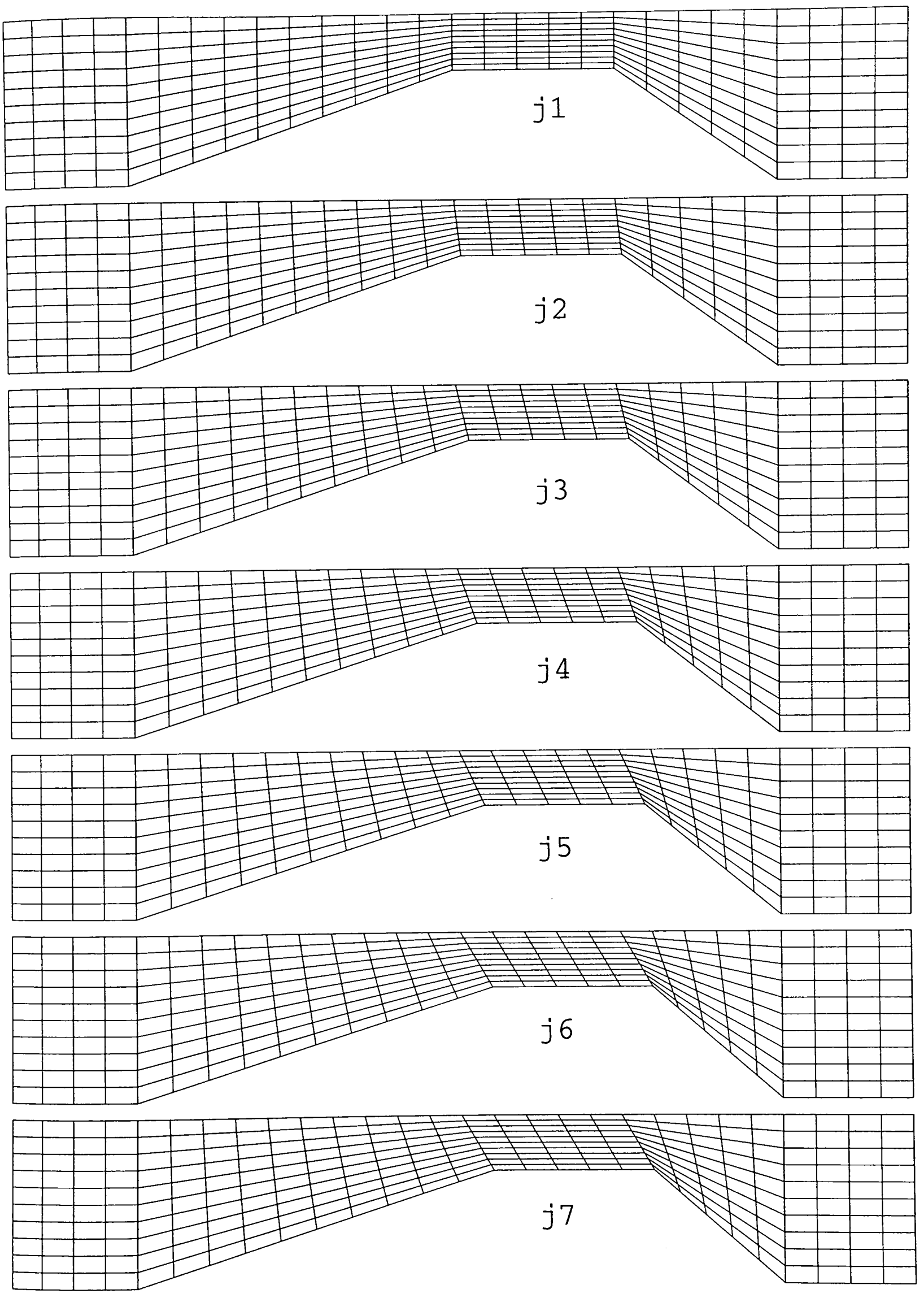


Figure 6.14 Y-planes of original grid

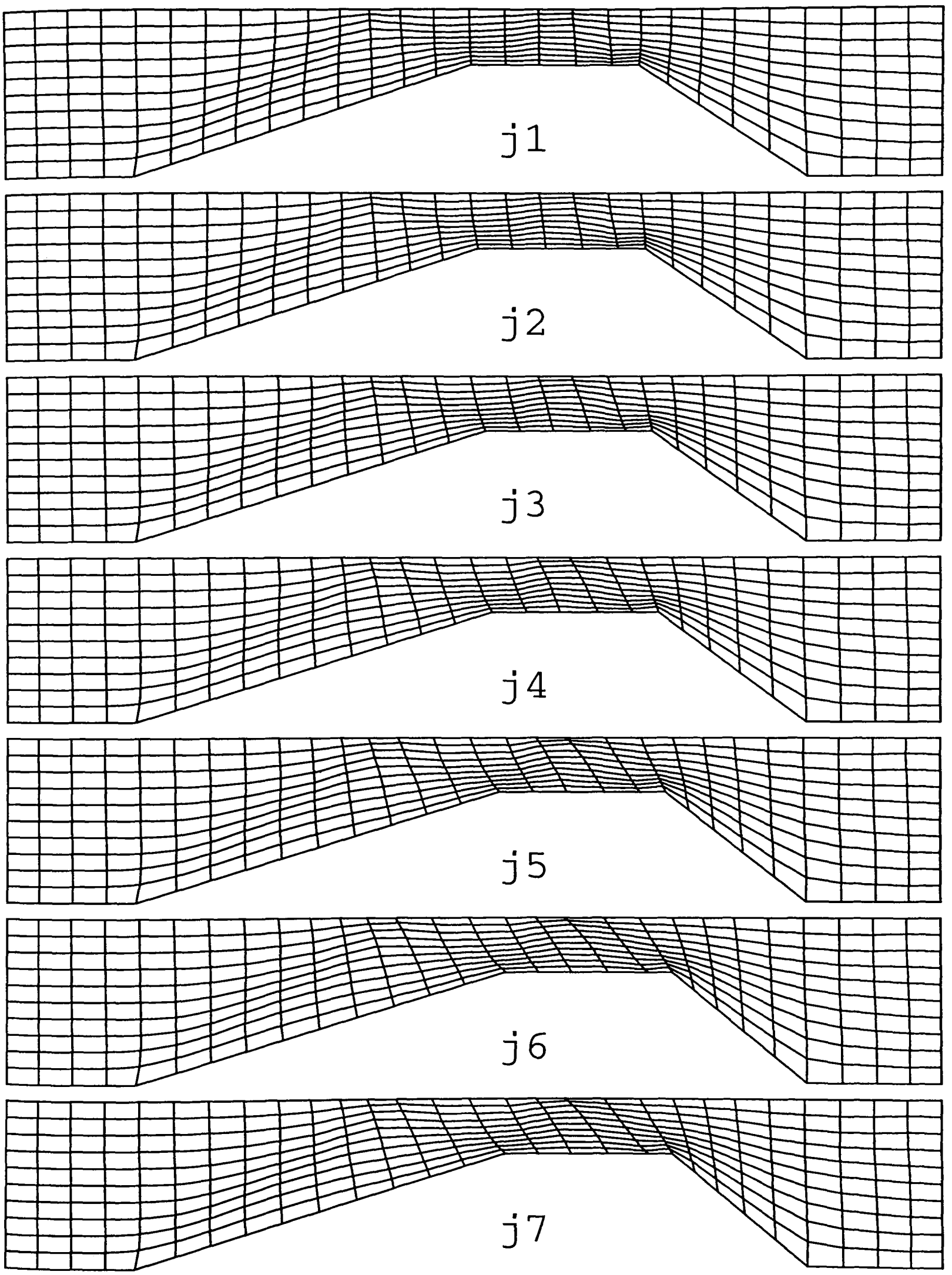


Figure 6.15 Y-planes of adapted grid

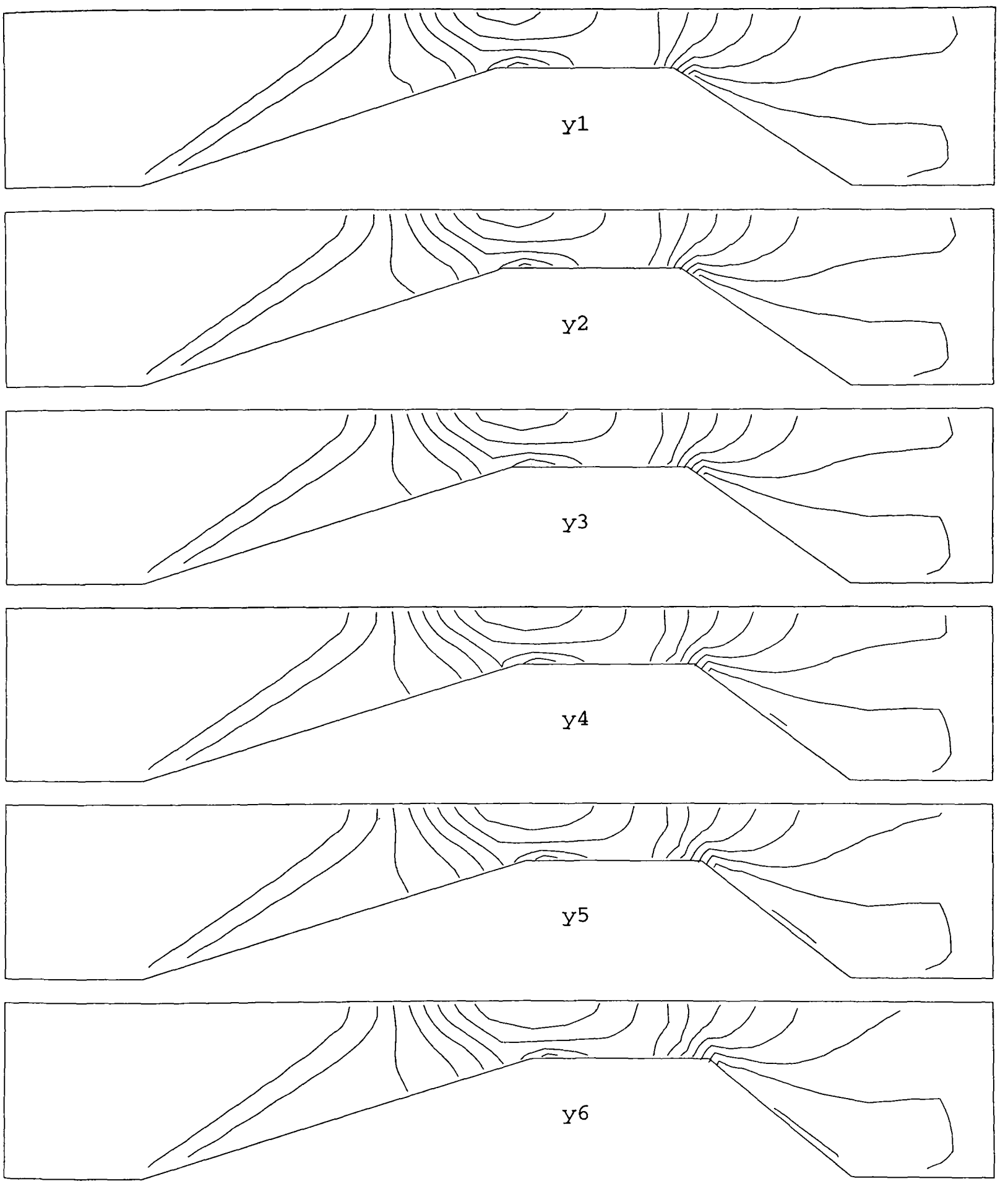


Figure 6.16 Pressure contours on Y-planes of original grid

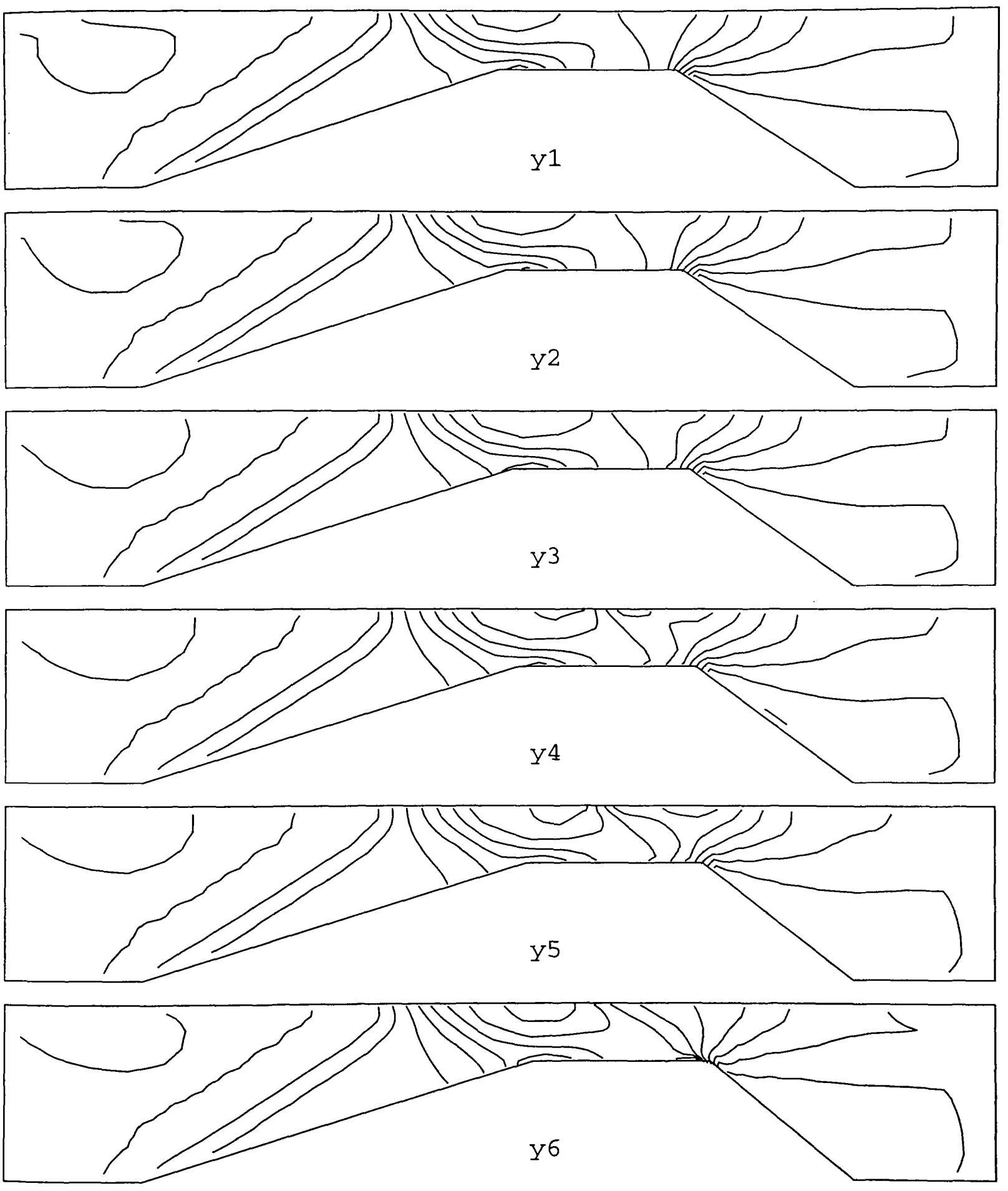


Figure 6.17 Pressure contours on Y-planes of adapted grid

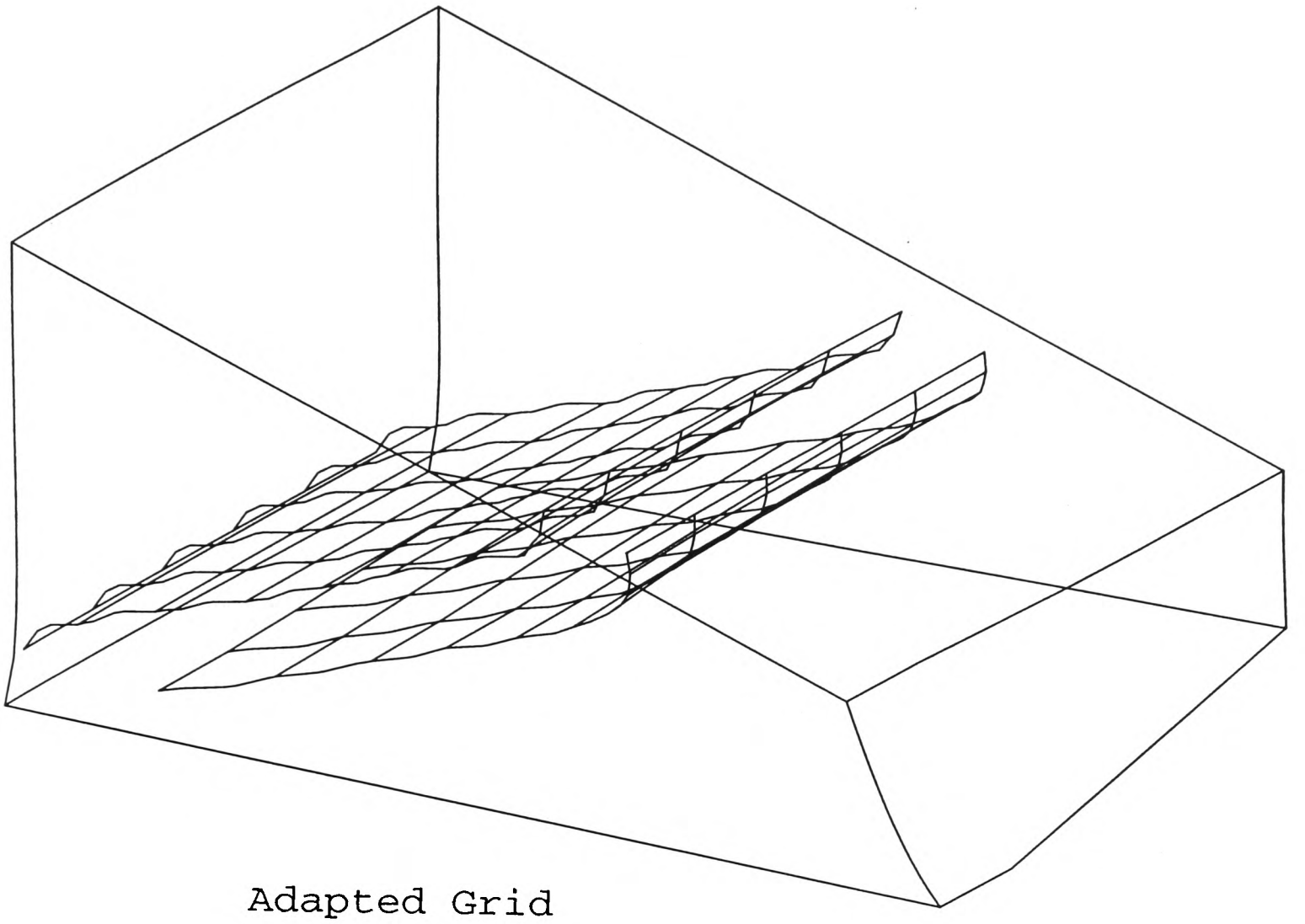


Figure 6.18 Comparison of the width of the shocks on the leading ramp

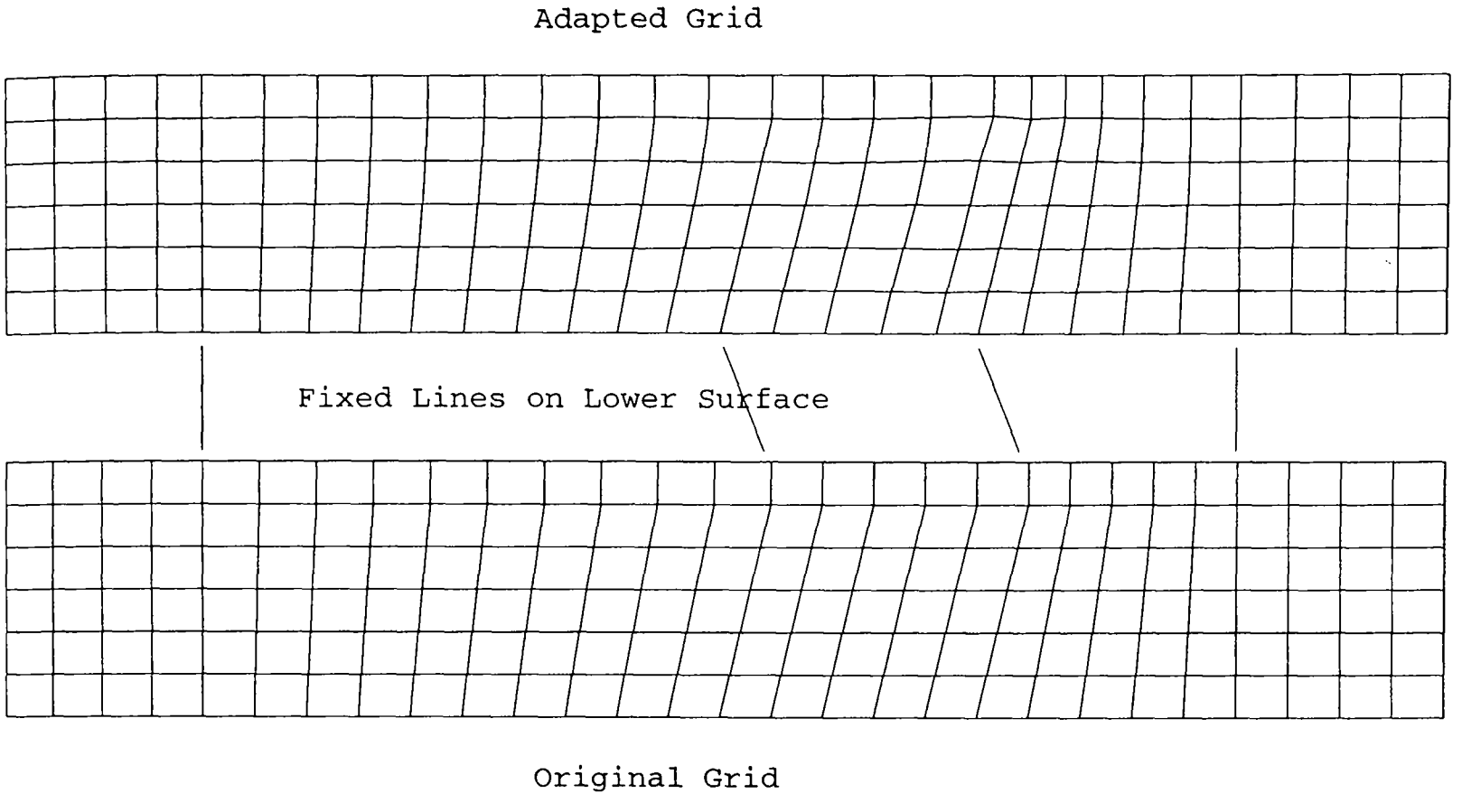


Figure 6.19 Fixed surfaces and curves on bottom surface of original and adapted grids

6.3.2 3D Driven Cavity Flow

This case is a three dimensional version of case 5.3.2. It has been used as a test case in literature (Guj and Stella 1993).

6.3.2.1 Boundary Conditions

This problem consists of a 1x1x1m cube with a lid moving at $1ms^{-1}$ from west to east. The fluid has a Reynold's number of 1000. As in case 5.3.2 the case is restarted from a run with a Reynolds number of 400, which was in turn restarted from a run with a Reynolds number of 100.

The cube is split into 60x60x30 uniform cells. 60x60 cells are the minimum used in case 5.3.2 to produce reasonable results. Only 30 cells are used on the Z-plane to prevent the problem from becoming prohibitively large.

The case is run for 5000 sweeps.

6.3.2.2 Adaption Parameters

One domain was used to cover the whole grid. Surfaces were set up on all boundaries except the top moving wall. Curve patches were set up along the edges of the grid except those that butted onto the moving wall. This allows all grid points to move except for those which define the main boundary condition on the top wall.

The adaption parameters used are

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
200	2000	200	5	0.9

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Magnitude of Velocity	$1+2\phi$ x^3 power law smoothing
1	0	2		

ϕ was limited to a maximum of 0.6 to avoid the velocity at the moving wall swamping other flow features in the domain.

6.3.2.3 3D Driven Cavity Results

Figures 6.20 through 6.24 show different planes of the final grid. The grid is shown at intervals of 6 planes in the X and Y directions and 3 in the Z direction, being divided up into 61x61x31 planes in all.

Figures 6.25 and 6.26 show plots of vectors indicating the local strength and direction of flow for the unadapted and adapted grids respectively. Figures 6.27 and 6.28 show surfaces representing the shape of the flow at a constant magnitude of 0.3 for the

unadapted and adapted grids respectively.

The flow is faster and more developed with the adapted grid, particularly at the low and high walls, suggesting that numerical dissipation has been reduced.

The grid has moved to concentrate in the area covered by the main eddy and smoothly varying across the domain.

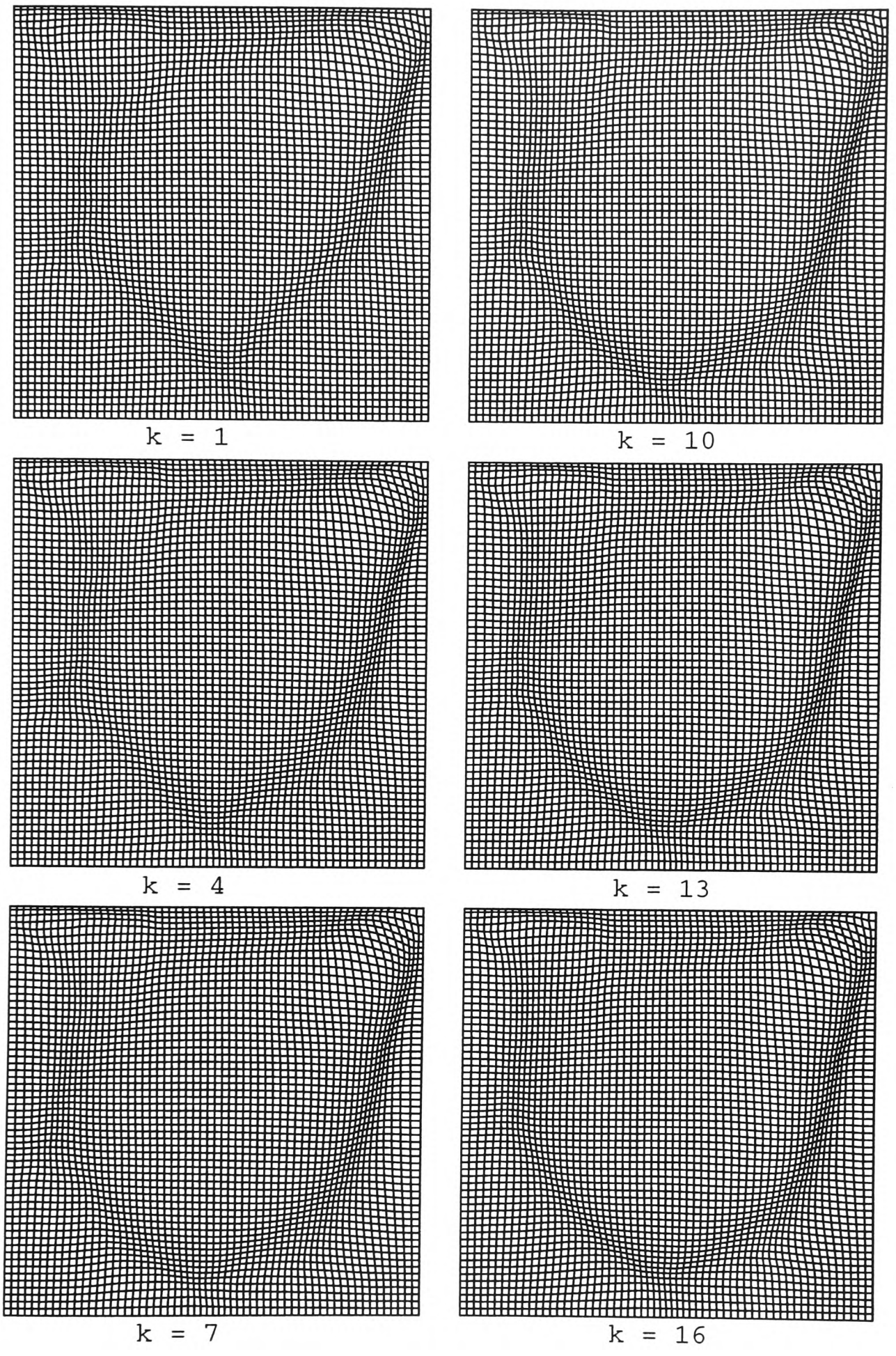


Figure 6.20 View of Z-planes of adapted grid (grid is symmetrical)

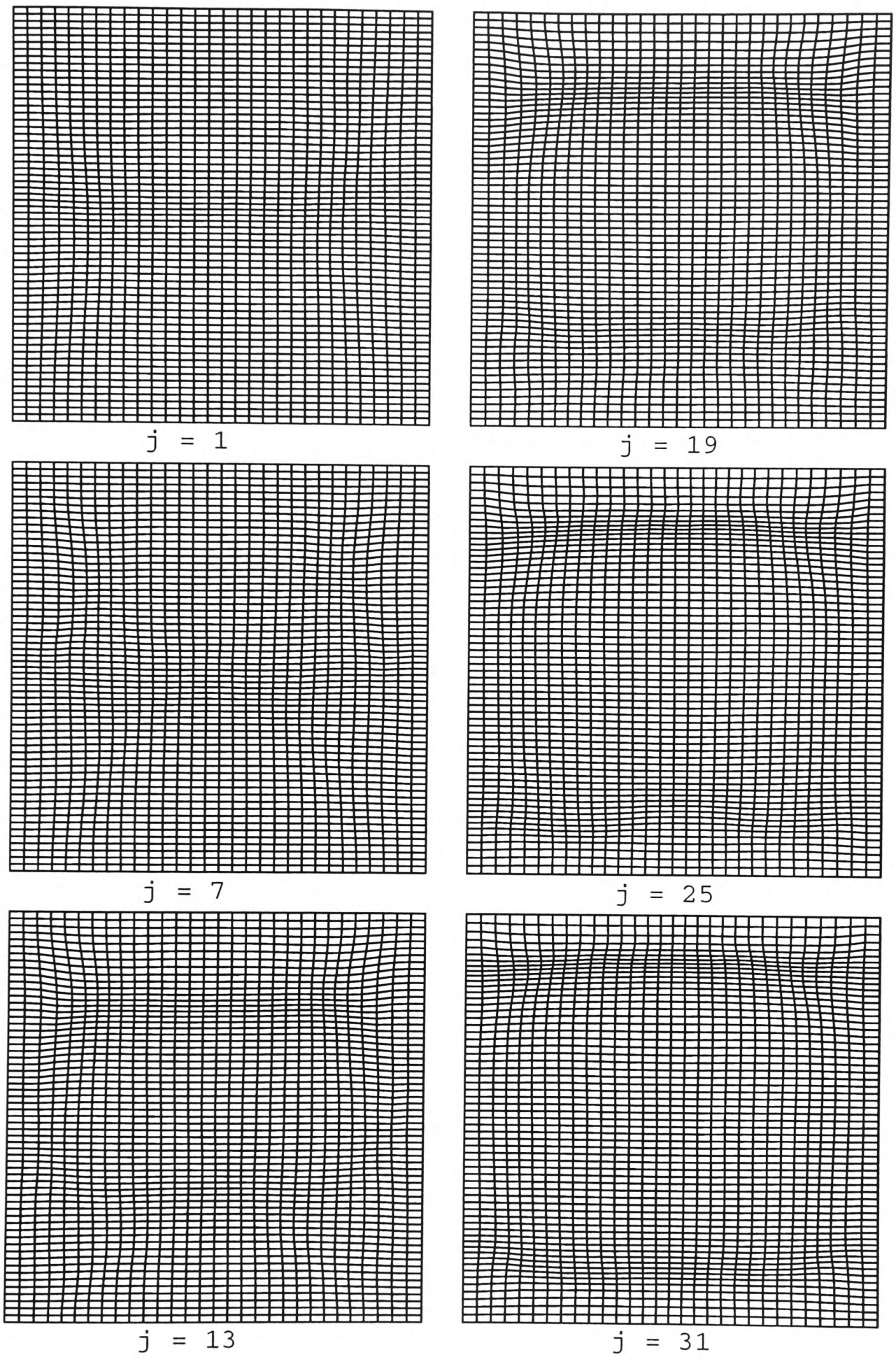


Figure 6.21 View of low Y-planes of adapted grid

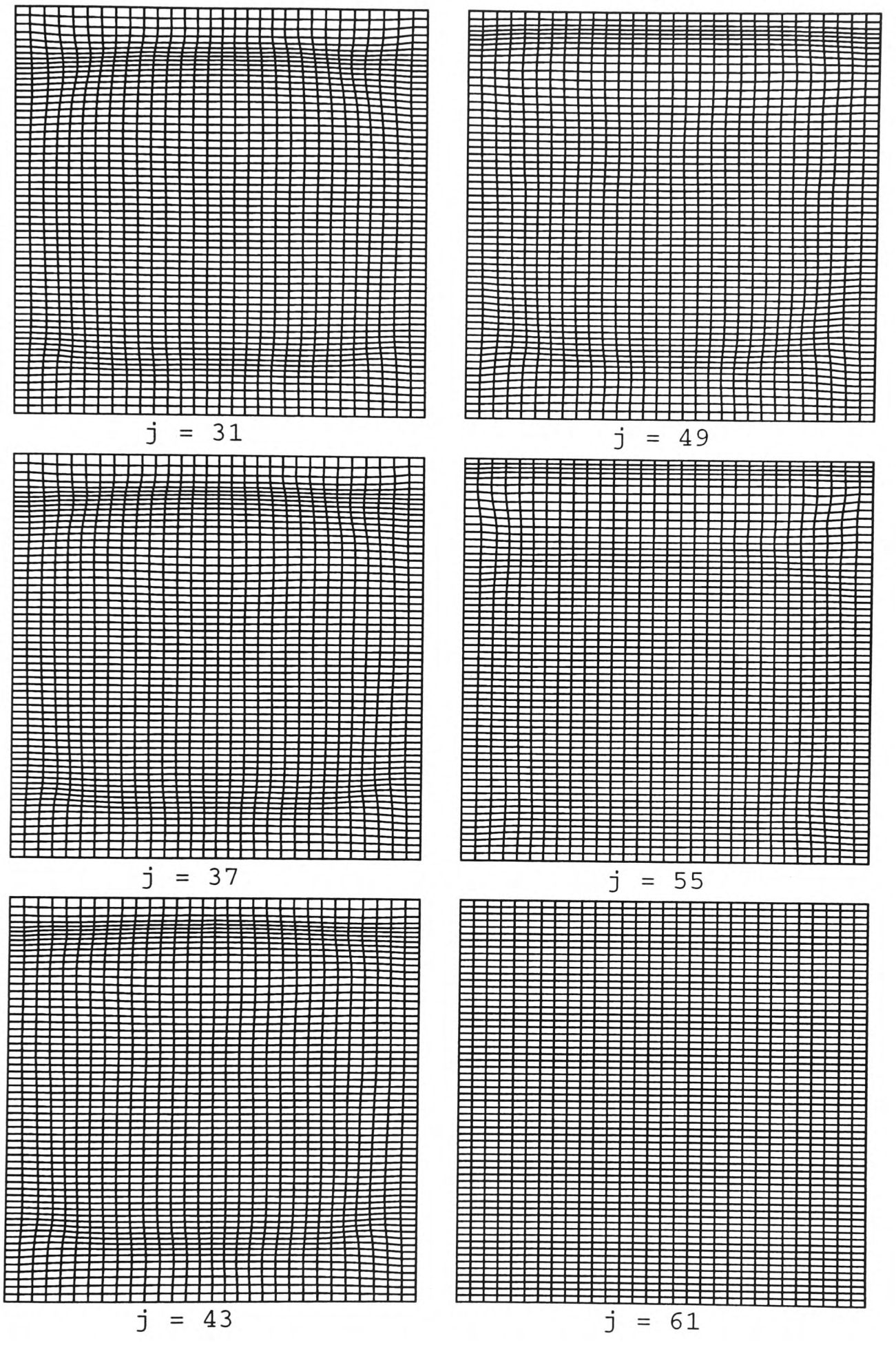


Figure 6.22 View of high Y-planes of adapted grid

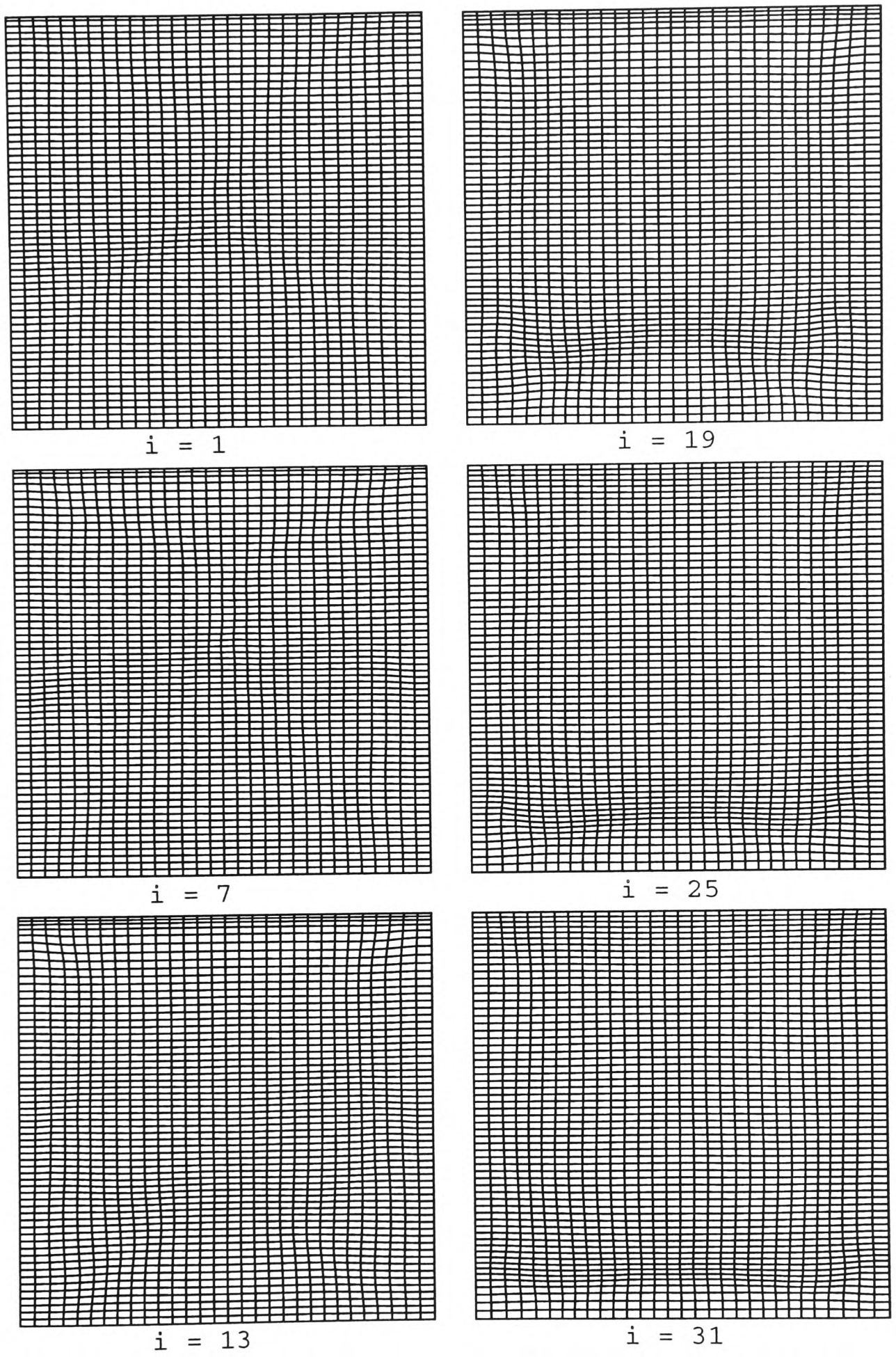


Figure 6.23 View of low X-planes of adapted grid

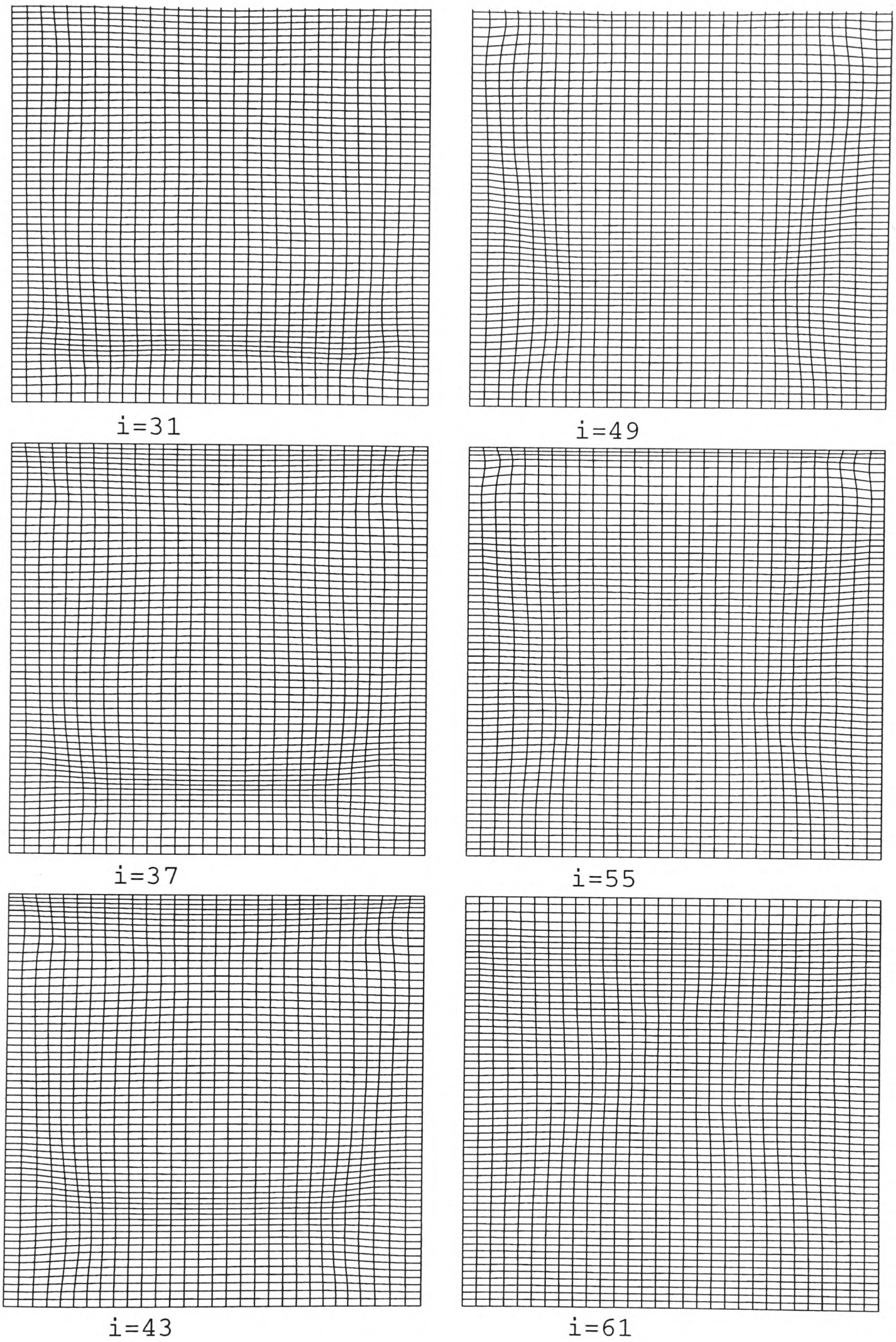
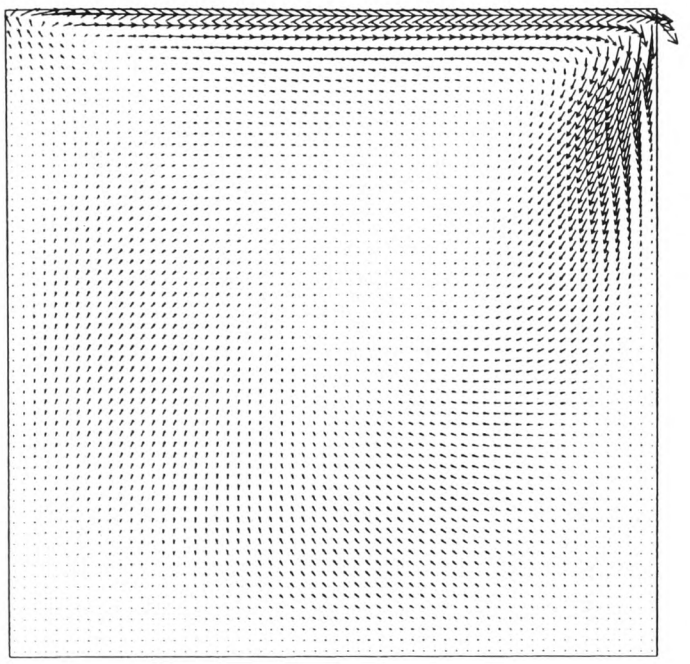
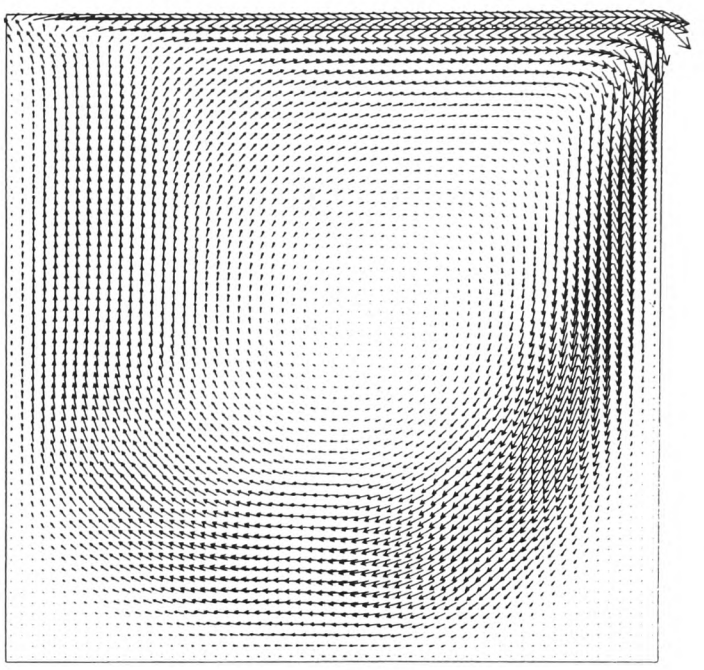


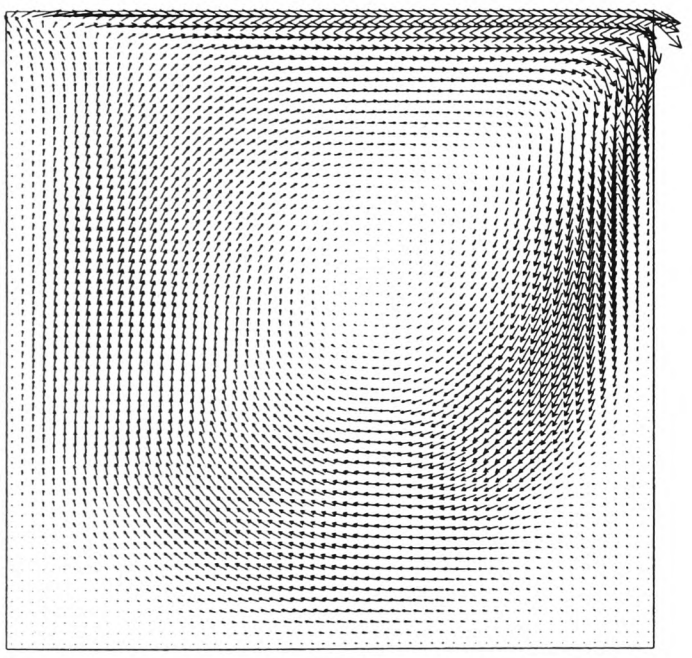
Figure 6.24 View of high X-planes of adapted grid



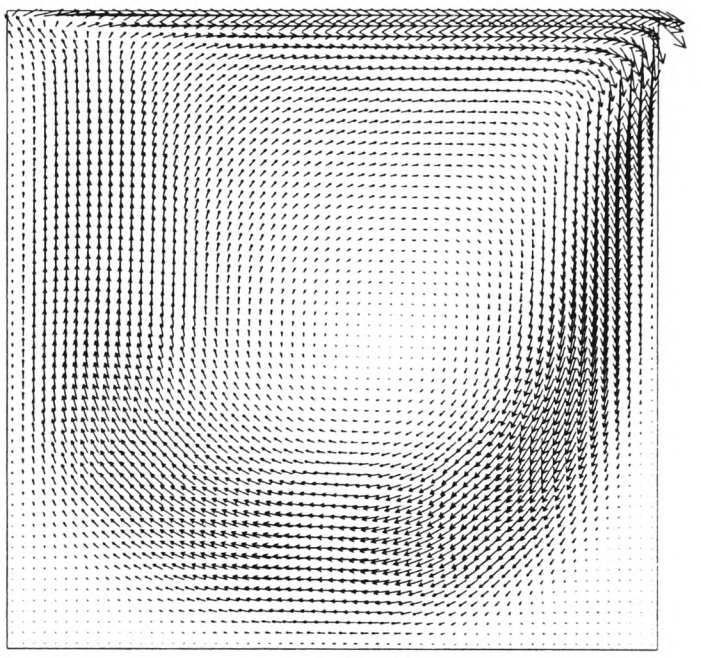
z = 1



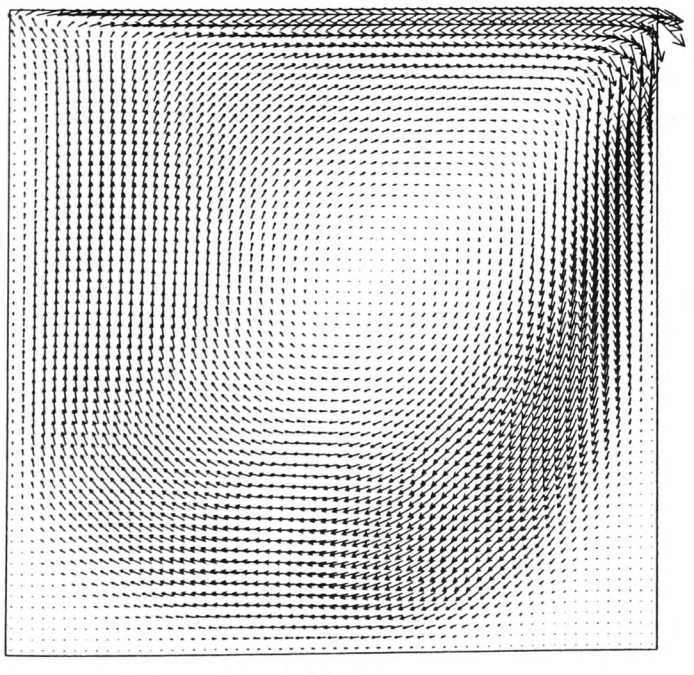
z = 9



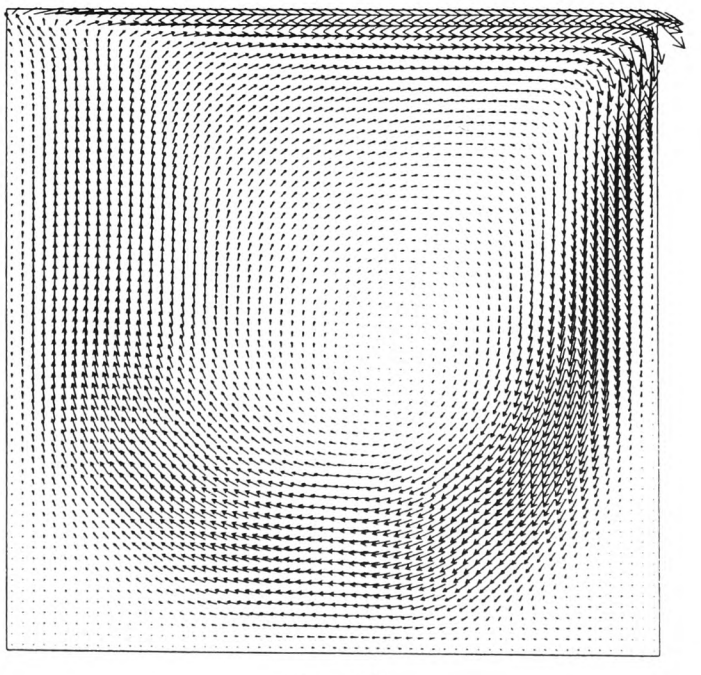
z = 3



z = 12



z = 6



z = 15

Figure 6.25 Velocity plots for original grid

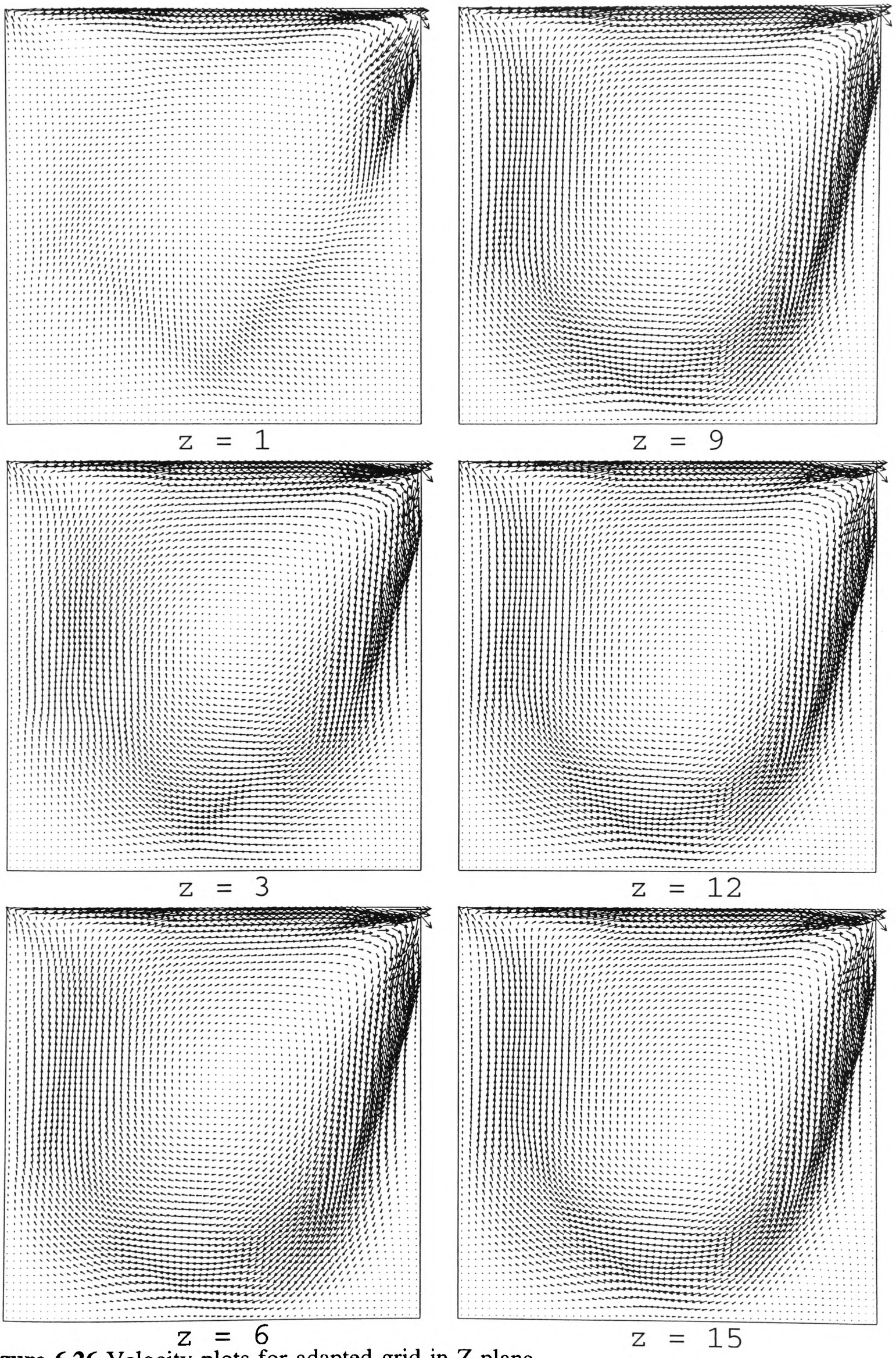


Figure 6.26 Velocity plots for adapted grid in Z plane

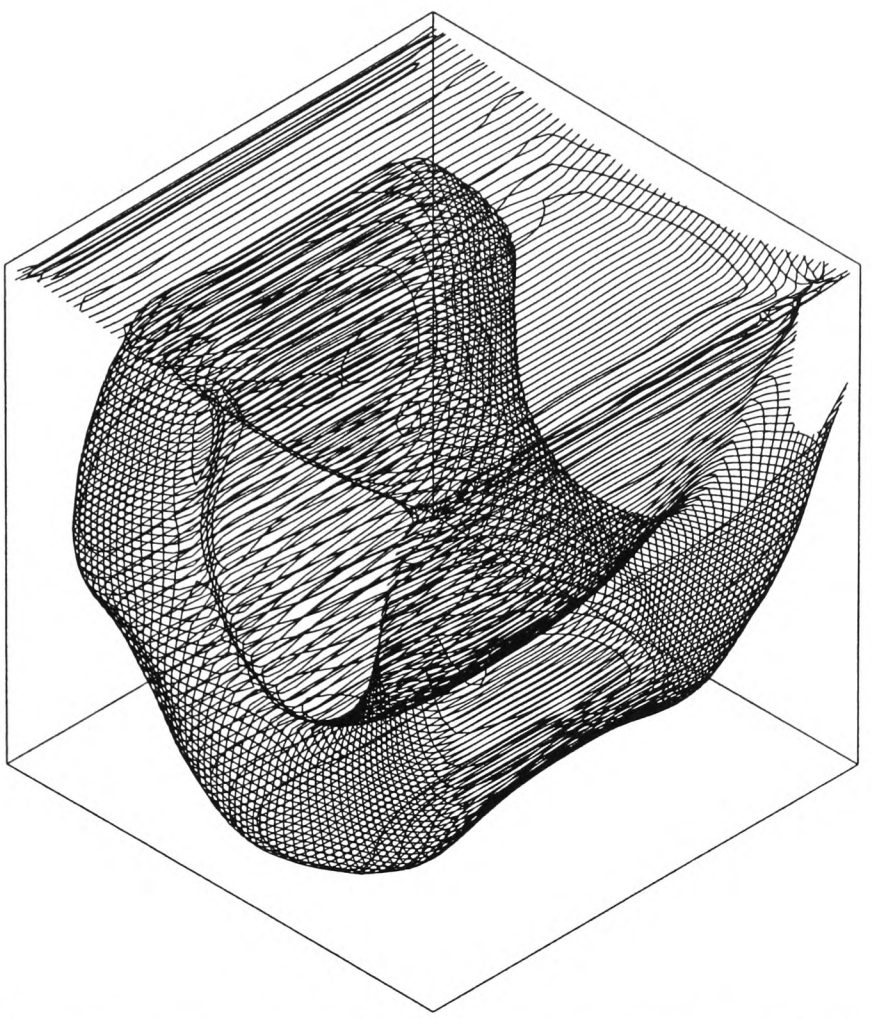


Figure 6.27 Flow in adapted grid - contour surface of velocity at 0.3ms^{-1}

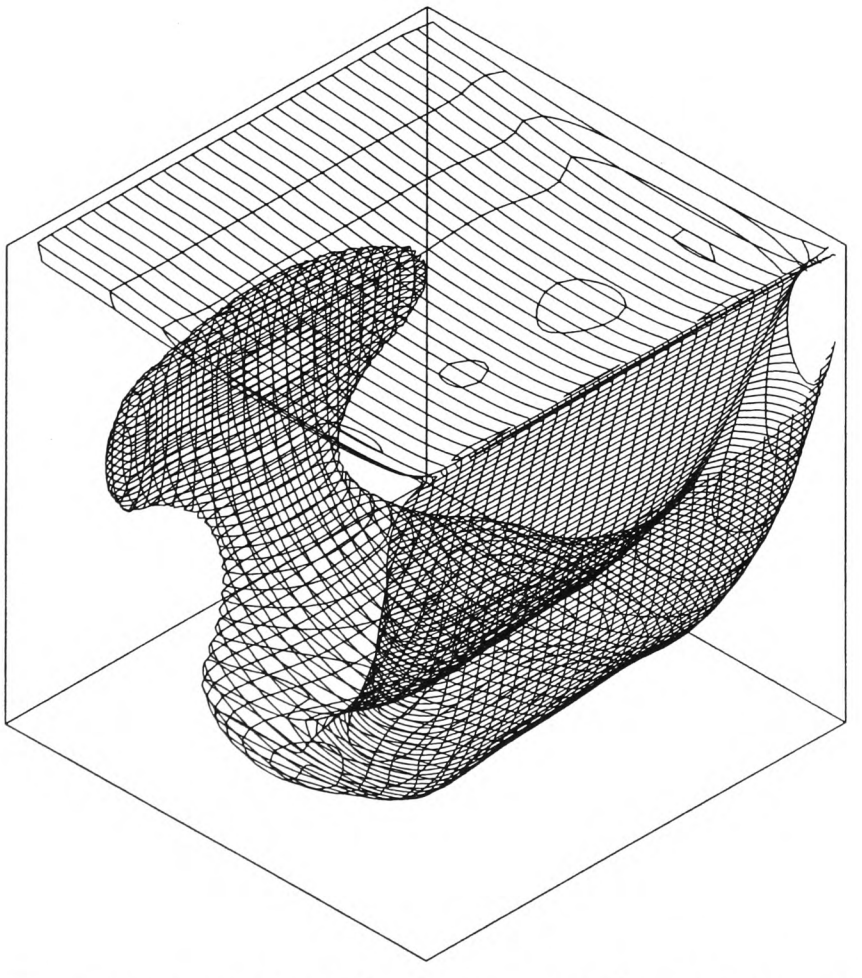


Figure 6.28 Flow in original grid - contour surface of velocity at 0.3ms^{-1}

6.4 Closure

This chapter demonstrates that the LPE algorithm is fully three dimensional. It does not however fully prove that the adapted results are superior to the unadapted, though experience from the two dimensional cases suggest that they should be. One aspect of the adaption algorithm that works well is the surface fitting method (3.2.3) which is used to maintain the boundaries of the skewed wedge (6.3.1).

There are problems in comparing unadapted and adapted results in that the position of the final adapted grid cells may be so far from the original that any direct comparison through a slice of the domain will only be possible with a lot of interpolation.

CHAPTER 7: FLUID STRUCTURE INTERACTION

7.1 Introduction

The adaptive grid technique is not limited to deforming the grid to a solution. It can also be used to modify the grid sensibly to allow for deforming internal and external boundaries. An example of this is fluid structure interaction, where the pressure exerted by a fluid is allowed to deform an object in the flow. The adaptive grid tools can then be used to both adapt to the local solution and to maintain a good quality grid around the object. Such problems occur for instance in the study of aerofoil flutter (Allen 1995, Sengupta et al. 1991), nuclear reactor safety (Donea et al. 1982), and buildings in wind (Slone et al. 1997).

Most techniques currently used for this kind of problem combine finite element codes for both fluid and structural modelling. (Donea et al. 1982, Hamdan and Dowling 1995, SenGupta et al. 1991), though Farhat (1995) uses a finite volume unstructured grid solver for the fluid simulation. This enables both sides of the problem to be treated in the same way.

Traditionally fluid structure interaction has been modelled by passing data between a stress simulation code to a fluid simulation code and the current work is no exception, though there is ongoing work at the University of Greenwich concerned with developing a fully coupled finite element code (Slone 1997). Though there are disadvantages and time penalties involved in using separate codes these are outweighed at the current time by the opportunity to use a mature and capable code for the fluid simulation code, PHOENICS and an in house stress code FET (Slone 1997).

FET stands for Finite Element Transient. It is designed to solve the system

$$M\underline{\ddot{d}} + C\underline{\dot{d}} + K\underline{d} = F \qquad \textbf{(7.1)}$$

Where $\underline{d}$ is displacement, M is the mass matrix, C is the damping matrix, K is the stiffness matrix and F is the transient load vector. The three terms on the left hand side relate to inertia, numerical damping and stiffness respectively.

In previous work FET has been used both to modify the structure and the surrounding grid. The surrounding grid is moved by using the spring analogy to connect grid nodes with springs with a low Young's modulus. In the current work it is only used on the structure under load.

For the purposes of the current work FET is treated solely as a black box. The only information needed is how to import and export data. FET reads data files in the neutral .ANL format used by the commercial code FEMGEN. The data files take the form of node locations, adjacency information and boundary conditions.

7.2 Algorithm

The algorithm is transient. The adaption module is called at the start of each time step before the PHOENICS sweeps begin. The stress code is called from within the adaptive module and run independently with data taken from PHOENICS, transient, but over one time step only. The time steps for the stress code and the flow code are matched. The restart facilities available within FET are used to allow the case to be restarted at all except the first time step.

Running the fluids code transient allows the effect of the velocity of the structure to be added into the flow simulation by the use of convection terms due to the change in volume of the cells adjacent to the deforming structure. These terms are calculated automatically within PHOENICS when the appropriate flags are set. Additionally the adaptive grid technique may behave poorly when coping with very large grid deformation over one move. The length of the time step can be used as a lever to control how much deformation takes place at each call to the structural solver.

The deformation of the structure is calculated by using the pressure on the surface of the obstruction calculated by the flow solver. In the finite element approximation the pressure over the face of each element is assumed to be constant and the load on each node in the boundary is calculated as a function of the pressure on the neighbouring elements. This function is internal to the structural code FET.

The stress code is called from within the adaption module and the resulting grid deformation passed through to the adaptive grid solver. The stress modified nodes are moved in the adaption domain and then the grid is moved around them, keeping them fixed. The adaptive grid solver can either be used to just carry out Laplace smoothing of the grid, or to adapt to other flow features. One option may be to develop a function to produce an artificial variable to drive the grid to better maintain its form about the obstruction.

7.3 Implementation

The details of the implementation cover how the two codes are coupled together and in particular how the data files for the structural solver FET are prepared from data available in PHOENICS. This section also covers the necessary modifications to the adaption pre processor S_ADAPT and processor G_ADAPT.

By hiding the stress code inside the adaption module no further modifications are made to PHOENICS EARTH beyond those required by the main adaption routines.

This version is limited to two dimensional problems. This greatly simplifies the input files to the stress code as the obstruction can be looked at as four sided elements with no depth. The third, inactive dimension in PHOENICS is ignored.

7.3.1 S_ADAPT Modifications

The only modification to S_ADAPT, the satellite adaption coding, is the addition of an extra adaptive patch type, STRESS, which activates the stress coding in ground and

sets up a blank, or no movement region up inside the adaptive domain. The additional routine WRITANL, used to determine the input files for the stress code could be incorporated into S_ADAPT at a future date.

7.3.2 WRITANL

The role of the program WRITANL is to write out an input file for the stress code FET in the neutral .ANL file format used by the commercial finite element software FEMGEN, using a PHOENICS grid file and a user controlled data file ANLIN.DAT. The file includes the specification of all the boundary conditions for the stress code, though with all loading set to zero. In addition WRITANL produces an inform file, .INF, which contains the number of nodes and elements in the stress model and a restart flag, and a file FE.INP which contains the i,j,k locations of cells in the PHOENICS grid from which the value of pressure needs to be extracted to complete the definition of the loaded boundary conditions for the stress model.

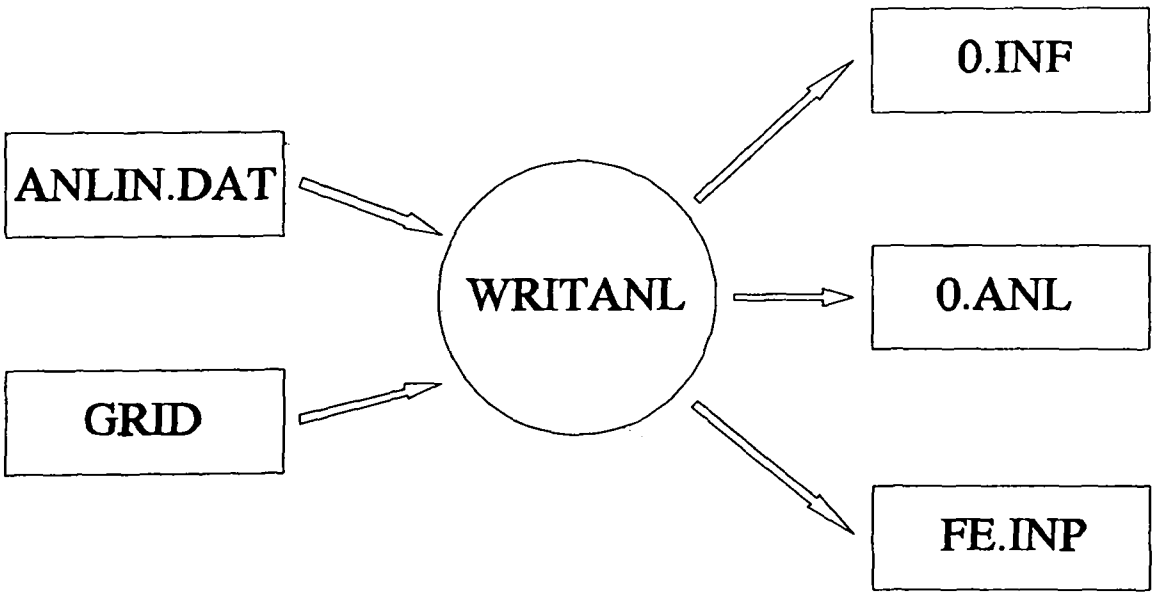


Figure 7.1 Preparation of Data Files for FET

The format of ANLIN.DAT is local to WRITANL, though it has the same patch style layout as the other adaption files. It incorporates patches defining the whole domain of the stress model, material properties, fixed boundaries and pressure loaded boundaries.

7.3.3 G_ADAPT Modifications

The stress module is called during the adaption cycle before the adaptive solver is used. The only modification to the main adaption coding is this call.

Between the main adaption coding and the stress code is an interface which takes the original stress input file 0.ANL produced by WRITANL and the data file FE.INP and writes out the final stress input file 1.ANL, including the pressure values in the specification of the boundary conditions taken from PHOENICS at the point in the solution cycle where the adaption module is called.

The stress patch is tied to an individual adaption domain. The modified grid is passed back to PHOENICS and through the rest of the adaption coding through the domain. The stress module is invisible to the rest of the code.

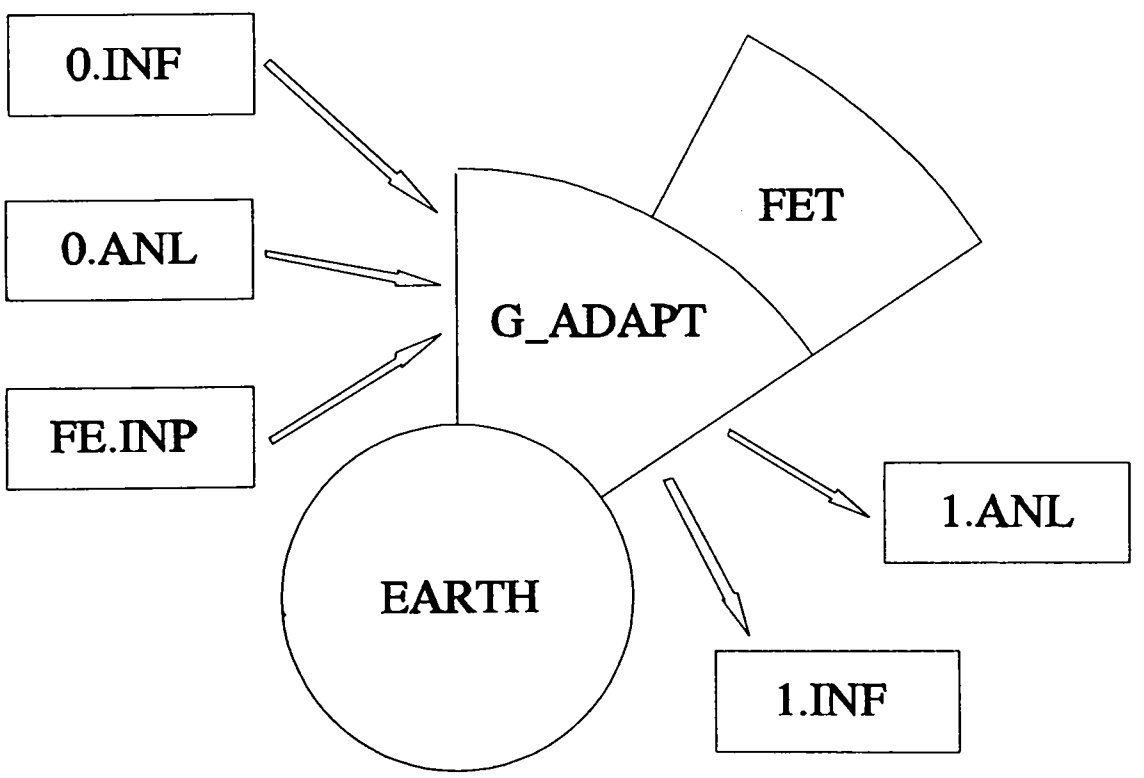


Figure 7.2 Interaction between G_ADAPT and FET

7.3.4 FET Operation

FET reads in the data files 1.INF, 1.ANL, and SCRIPT, and outputs the result files 1.STR, 1.DIS and 1.NEU.

SCRIPT contains data controlling the operation of the solver, including the length of

the time step.

1.STR is a restart file, which holds the modified nodal positions, strains and stresses, 1.DIS holds the nodal displacement, and 1.NEU is a neutral file used by the package FEMVIEW to visualise the results produced by the stress code.

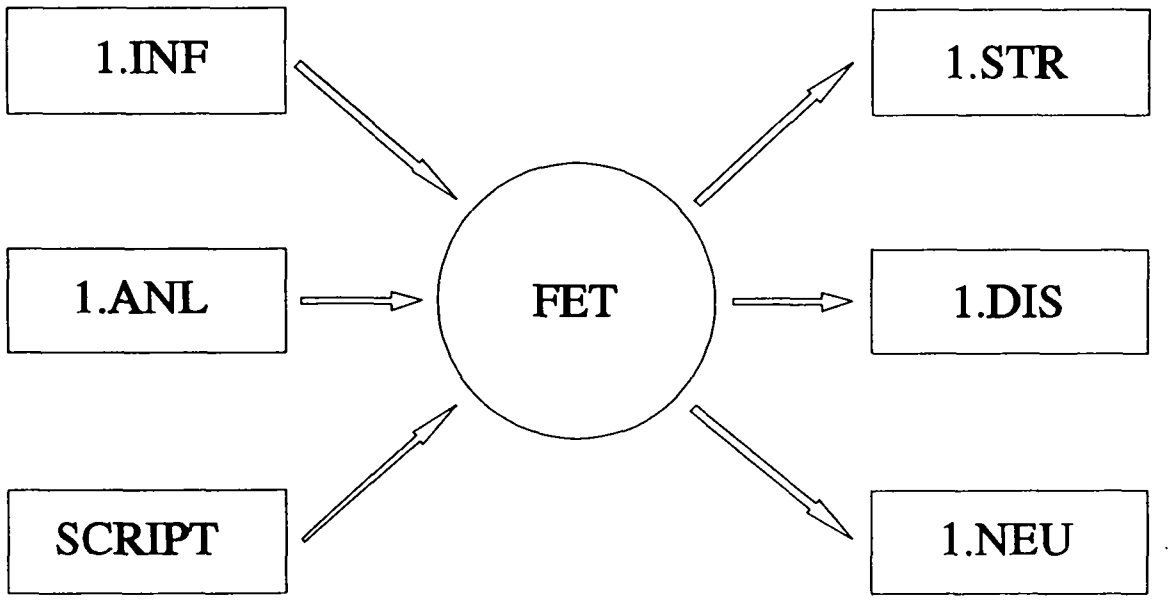


Figure 7.3 Input and Output Files for FET

7.4 Example Case

A simple case of a flexible block structure in an air flow has been run to demonstrate the technique.

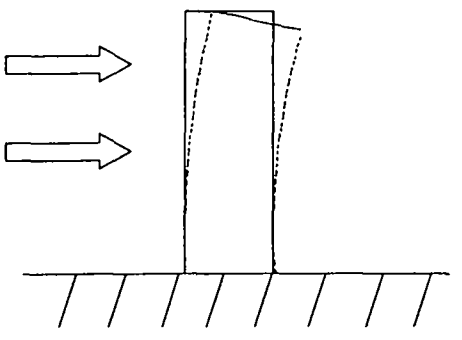


Figure 7.4 Example stress case

The block has a fixed base and deflects under the pressure load from the flow.

7.4.1 Boundary conditions

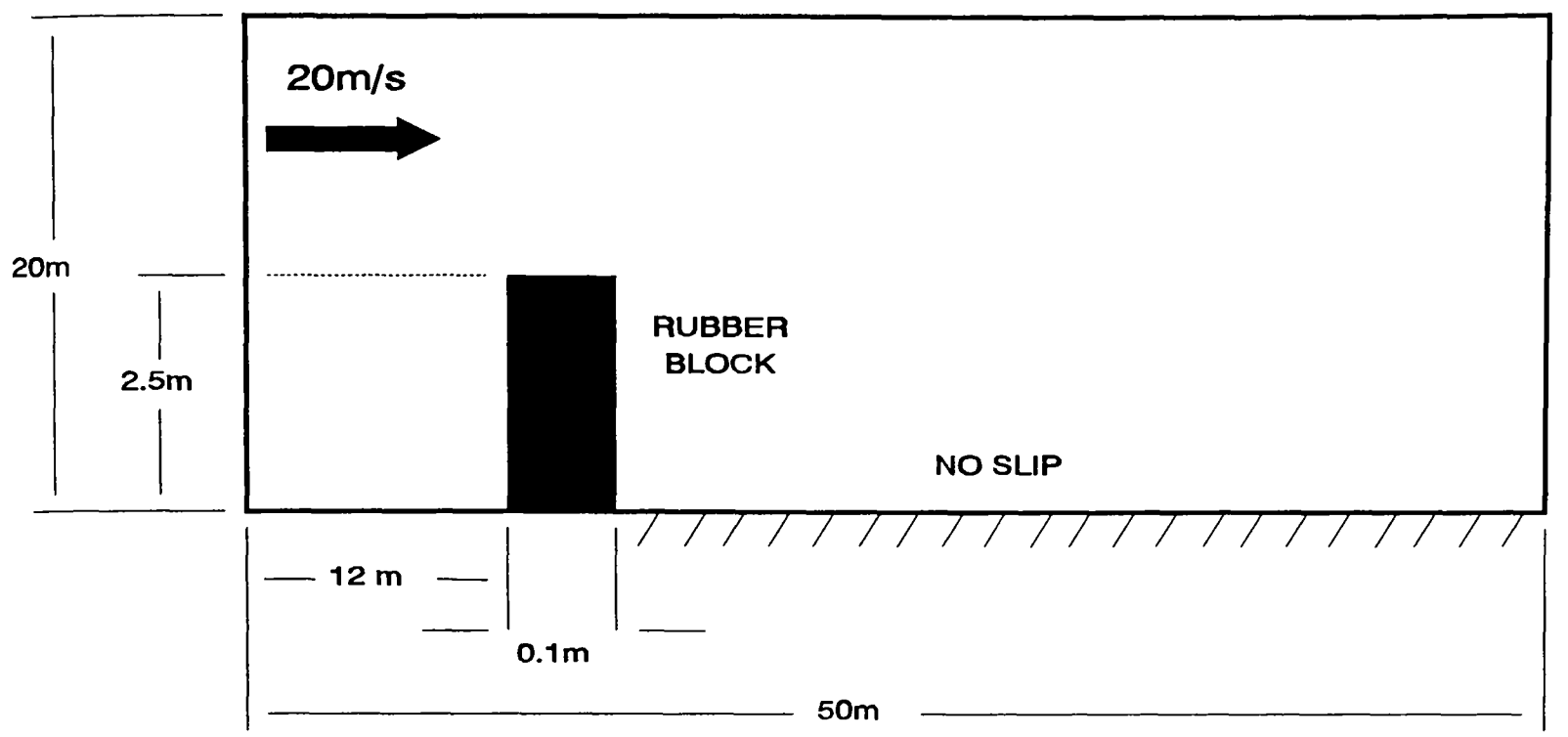


Figure 7.5 Grid Dimensions

There is a constant inlet velocity of 20 m/s, and the fluid is assumed to be air at 300K and 1 atmosphere pressure. The simulation is run over twenty time steps of five seconds each. The flow solver is run for fifty outer iterations every time step. A no slip boundary condition is placed after the obstruction, helping to create a large recirculation behind it.

The block has the material properties of rubber. It has a Young's modulus of $4x10^6$ Nm^{-2}, a Poisson's ratio of 0.475, and a density of 1130 kgm^{-3}. Rubber was chosen because its properties give enough flexibility to the block to give a large deflection in a low speed flow without collapsing. It measures 0.1x2.5m.

A large domain is used to allow the full eddy to develop after the obstruction, and to prevent the free boundary at the top of the domain from influencing the results.

The main grid is rigged to provide a fine, regular mesh directly over and around the obstruction. Away from the obstruction the grid becomes increasingly coarse to minimise the computational resources required for the run.

The problem was run over 40 time steps of 10 seconds each. 100 sweeps of the PHOENICS solver were carried out within each time step.

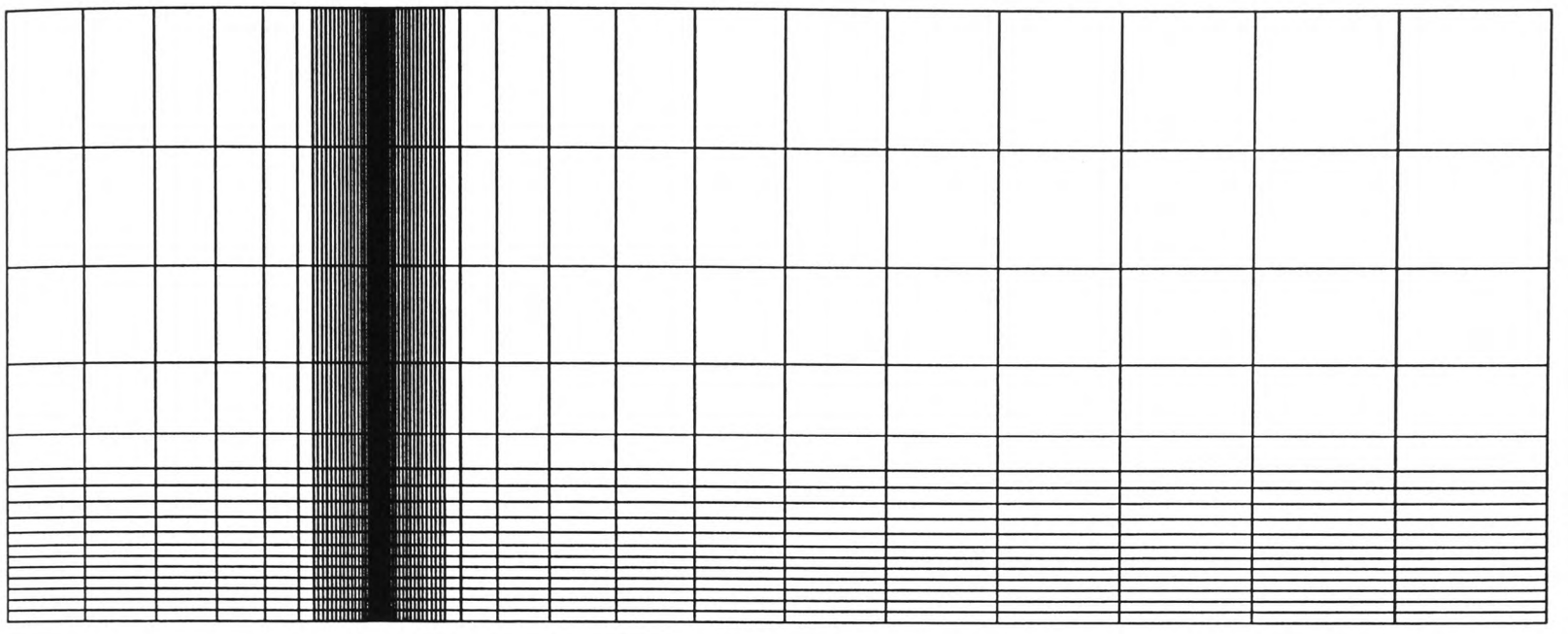


Figure 7.6 Example stress case grid

7.4.2 Adaption Parameters

The adaption domain is restricted to the regular grid around the obstruction. The LPE equation is weighted for Laplace solving only. No account is made for the local flow conditions in the grid movement. This means that the only movement that will occur will be due to the deflection of the obstruction.

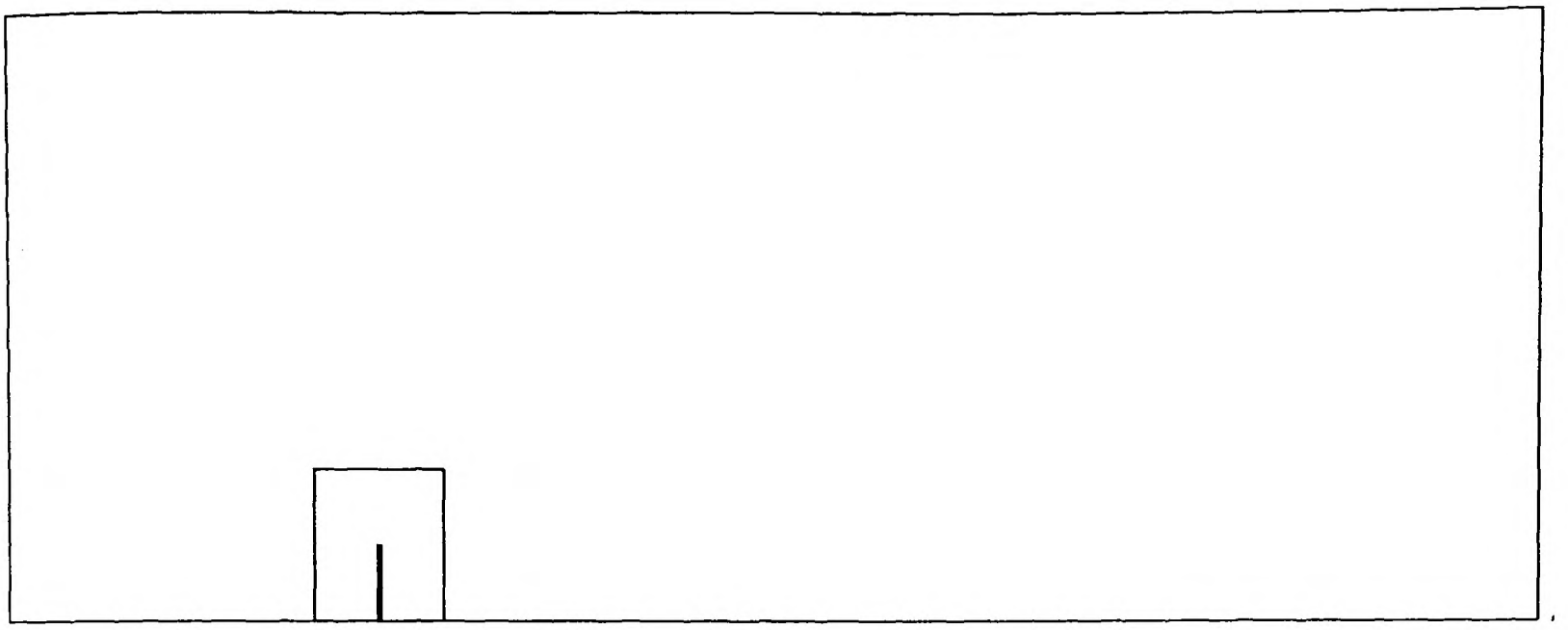


Figure 7.7 Adaptive domain and blockage in whole domain

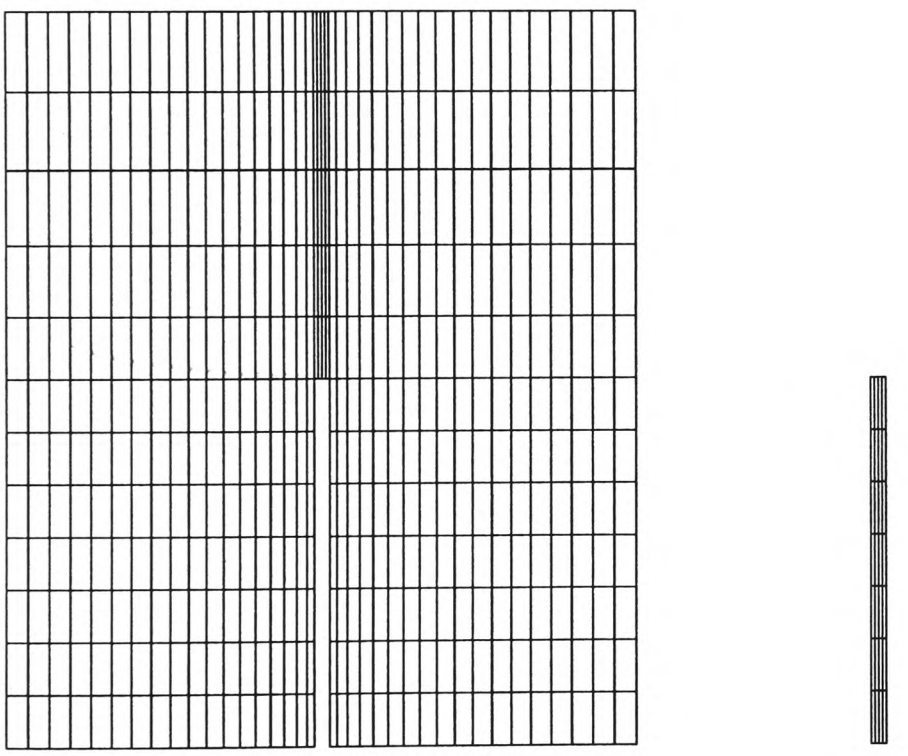


Figure 7.8 Adaptive domain with block blanked and block grid

In each call the LPE solver goes through thirty iterations. No interpolation of data is used or needed.

7.4.3 Results

Using the adaption estimation based on the spacing in the initial grid the adapted region moved a total of 157.8% in the x direction and 9.3% in the y direction.

Figure 7.9 shows the velocity field for the whole domain without any deformation. Figures 7.10 through 7.17 show the grid in the adapted region and its local velocity field at different stages during the simulation.

Figure 7.19 shows the displacement of the top left corner of the block against time.

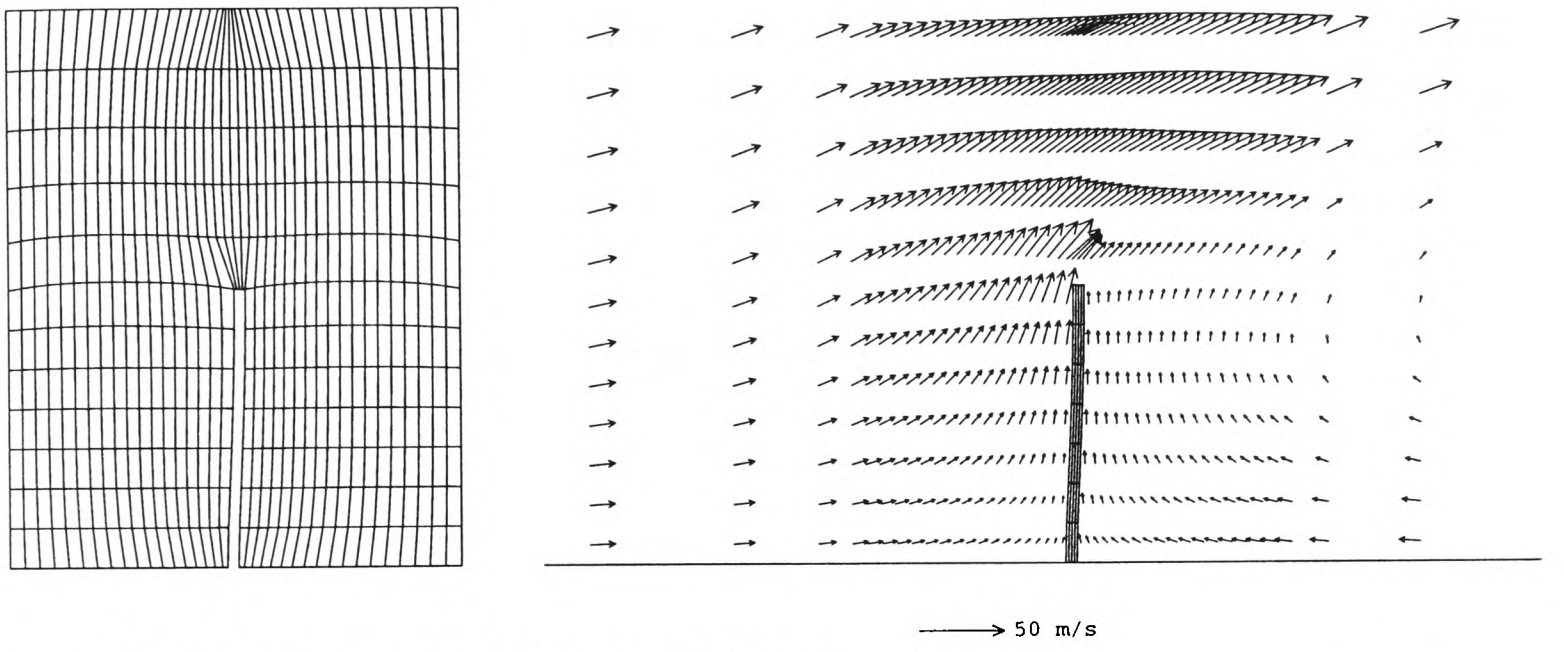


Figure 7.9 Grid and velocity fields at 30 seconds

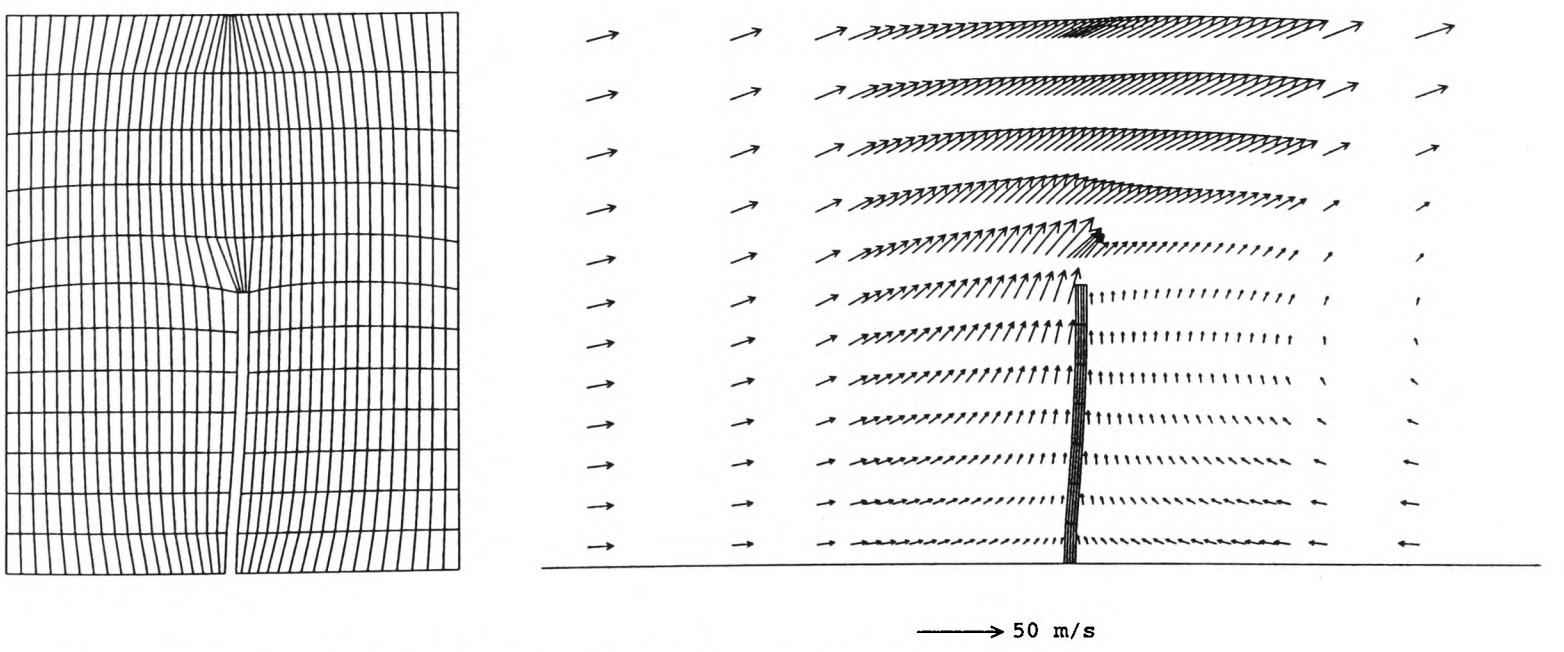


Figure 7.10 Grid and velocity fields at 50 seconds

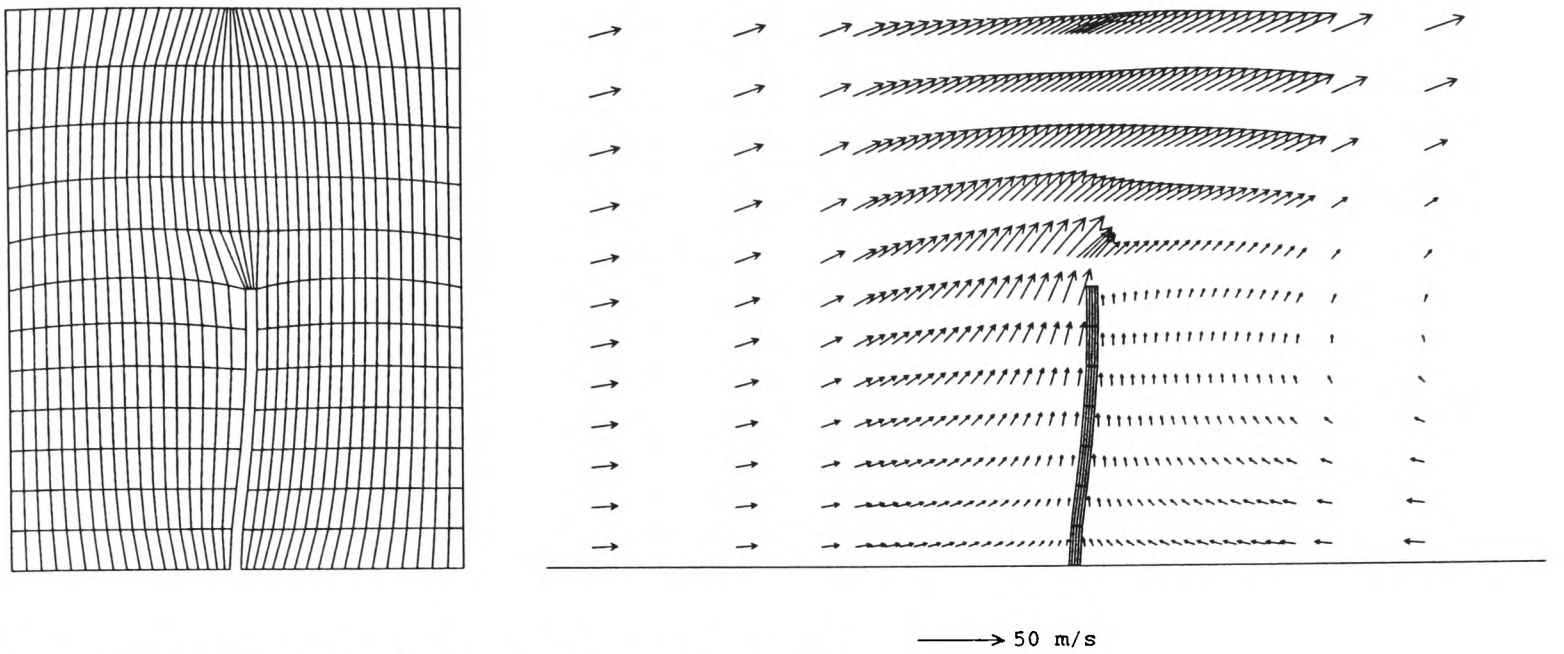


Figure 7.11 Grid and velocity fields at 70 seconds

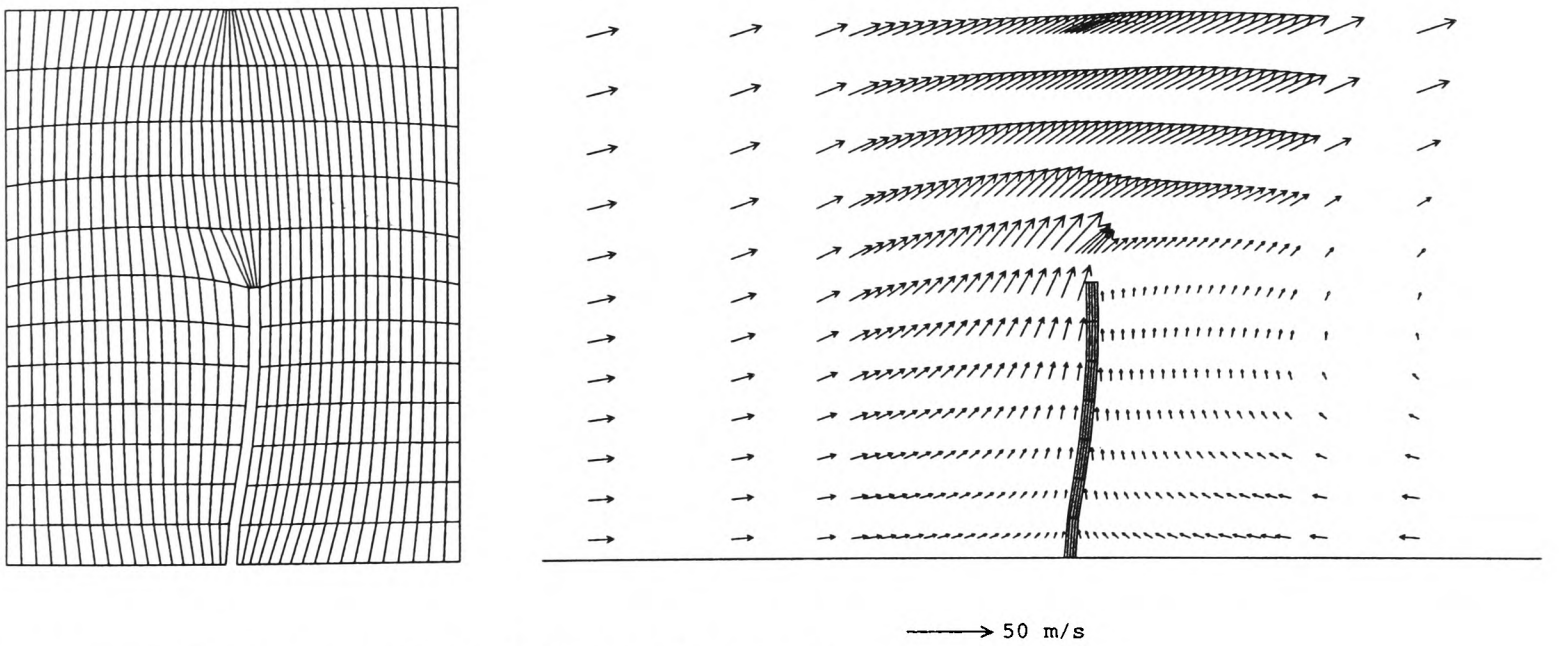


Figure 7.12 Grid and velocity fields at 90 seconds

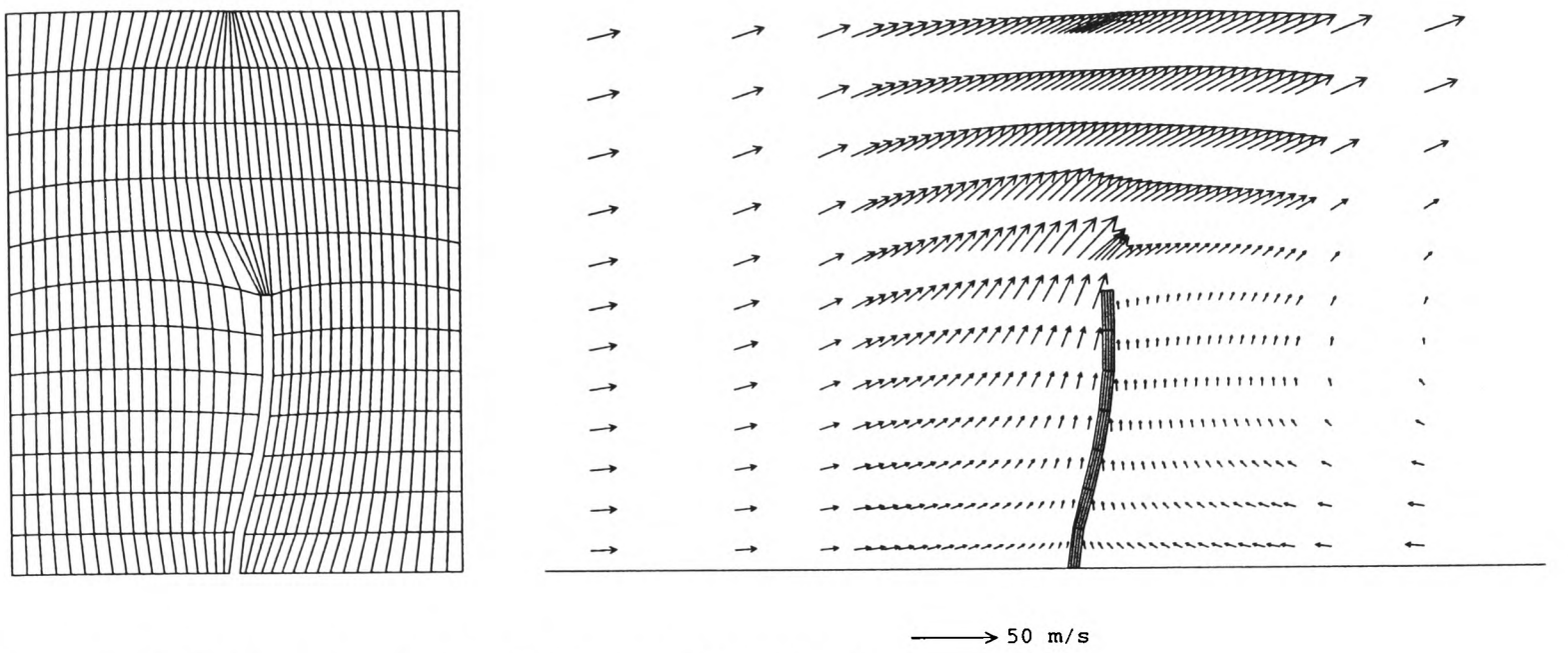


Figure 7.13 Grid and velocity fields at 150 seconds

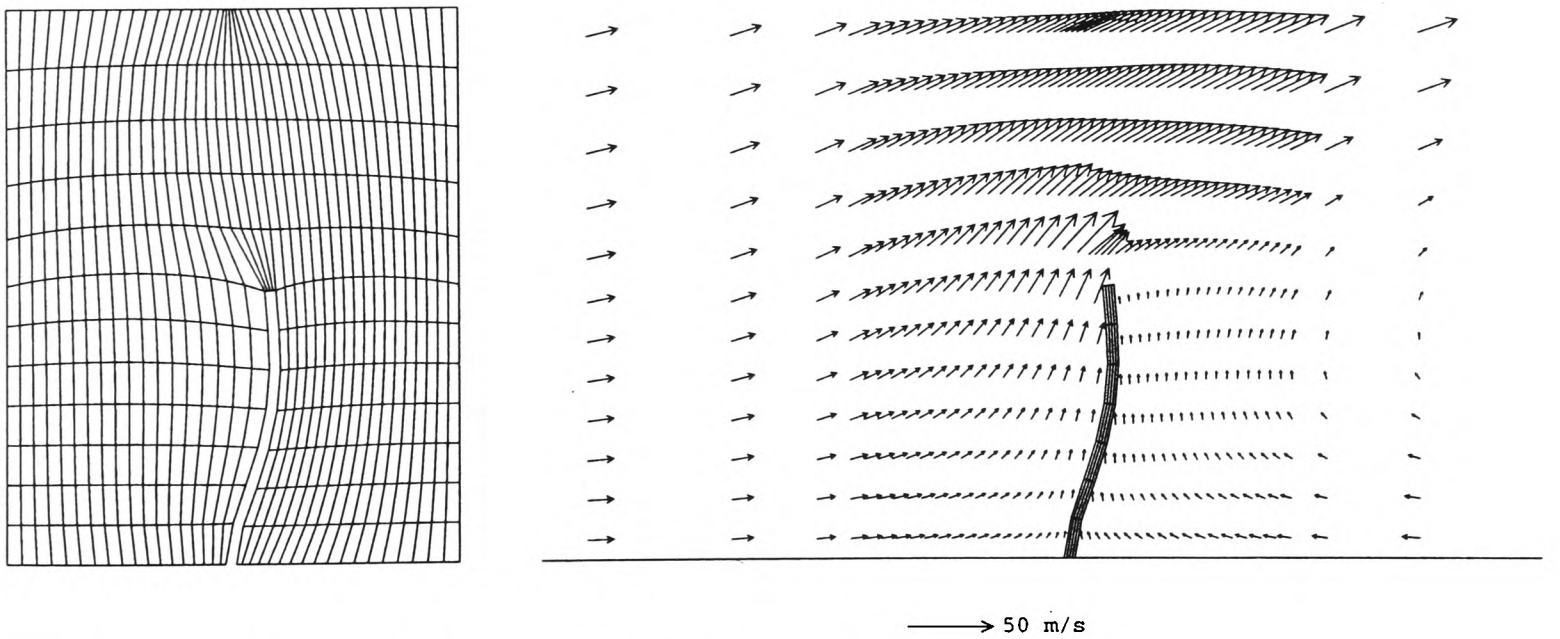


Figure 7.14 Grid and velocity fields at 210 seconds

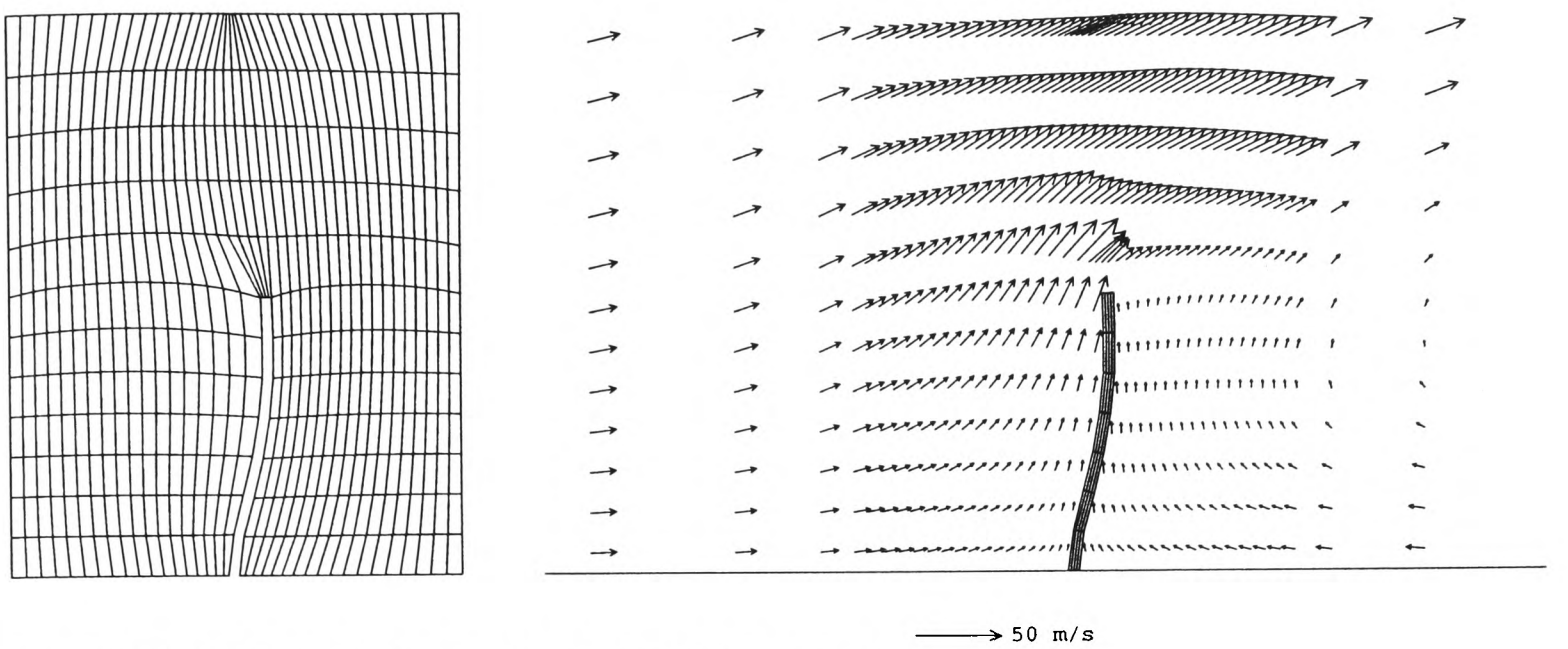


Figure 7.15 Grid and velocity fields at 300 seconds

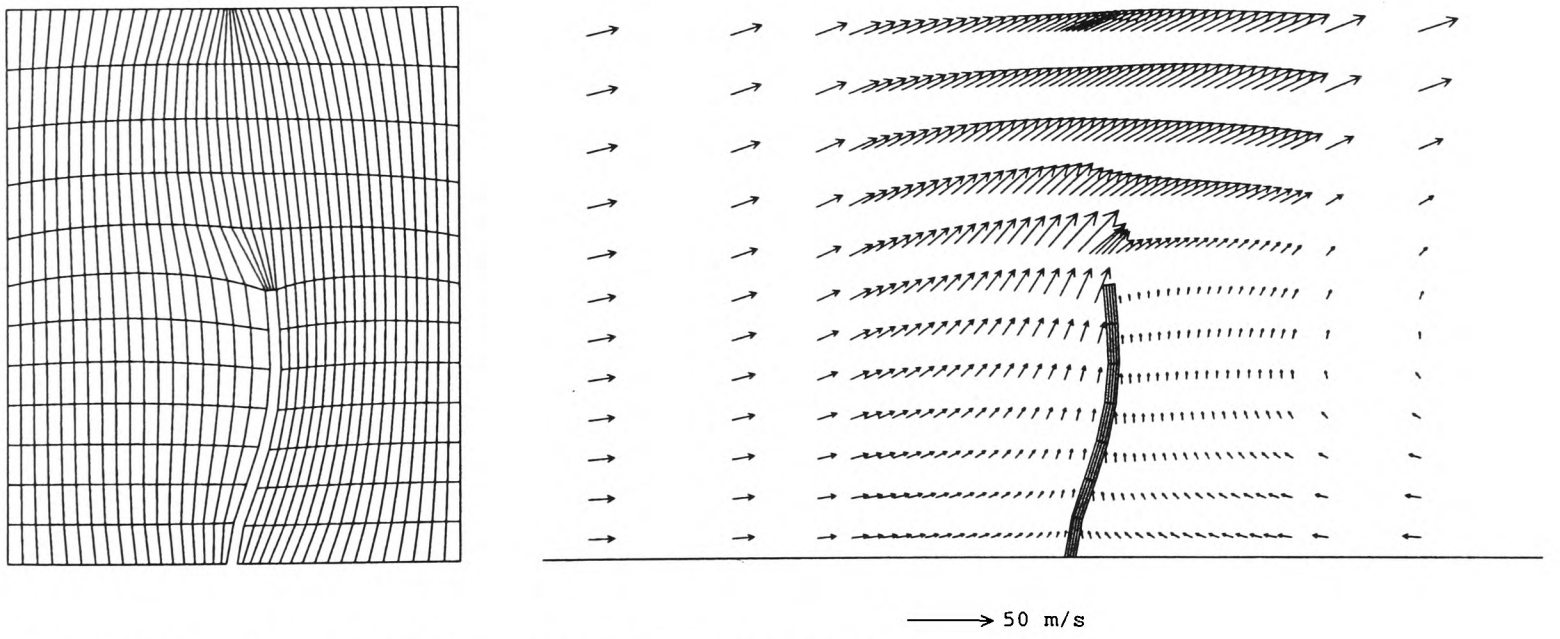


Figure 7.16 Grid and velocity fields at 400 seconds

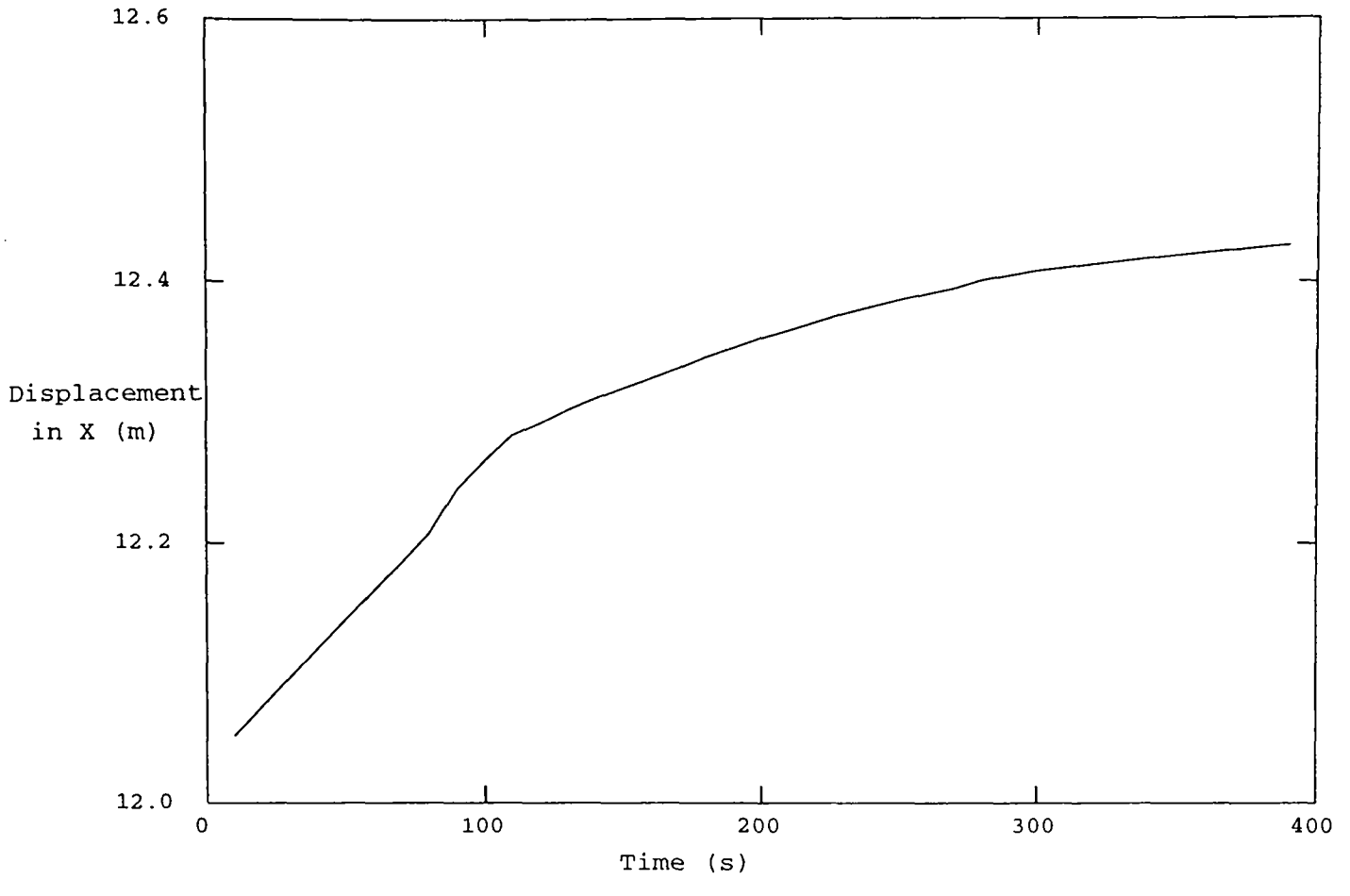


Figure 7.17 Displacement of leading corner of obstruction against time

The grid becomes increasingly skewed at the trailing edge as time passed but no lines overlap. The adaptive grid solver does move points to fit the grid better to the obstruction, but the well documented behaviour of the Laplace term in attracting points to convex surfaces and away from concave surfaces here helps to reduce the concentration of cells on the leading edge of the obstruction.

The cells above the obstruction become skewed though they remain aligned with the local flow.

The behaviour of the grid could possibly be improved by using an artificial geometrical or stress based variable instead of or in addition to that used for adaption. The artificial variable could involve attraction to the boundary of the obstruction, or something based on giving a slope that will push cells in the direction of the major movement of the obstruction.

7.5 Closure

The aim of the work presented in this chapter is to demonstrate that the adaptive grid technique can be used for fluid structure interaction problems, not necessarily to prove that the method is superior to other methods, though the ability to use a powerful fluid dynamics code such as PHOENICS is an advantage.

The use of the adaptive grid technique could also be extended to other problems requiring moving grids, including structures or boundaries with prescribed movement, such as vibrating aerofoils (Allen 1995), or pulsing blood vessels and other organs (Xu et al. 1993).

The work is currently restricted to 2D problems with only one structure being deformed. As PHOENICS, FET and the adaptive grid technique are all 3D capable codes and there is no reason why the work cannot be stretched to 3D with multiple structures. It would also be straightforward to allow other forms of loading such as temperature or fixed loads to be added into the simulation of the deformation of the structure, as these modes are already allowed for in FET. The codes are on the whole kept very separate. Greater integration would involve removing the input and output files used by FET and incorporating the information passed to and from it into routine calls PHOENICS and adaption data files.

CHAPTER 8: NEURAL NETS

8.1 Introduction

The aim of this chapter is to briefly introduce a new technique for grid adaption. This technique involves the use of a self organising neural net, or SONN. This chapter is based on work published in Patel, Bennett, Zhao and Knight (1995) and Zhao et al (1996).

Neural nets commonly consist of one or more layers of artificial neurones or processing elements (PE's) that can be trained to resolve certain types of problems by being given answers to known problems. In the training phase the links or synapses between different PE's are weighted to affect how they respond to given stimulus. These weights determine if a given stimulus will cause a PE to activate and represent the knowledge held in the net.

Neural nets have been a popular tool for some time and a range of different neural net techniques have been used in areas such as pattern recognition and financial modelling. A few papers have begun to appear dealing with the use of neural nets for grid generation (Hartle 1995, Khan et al 1993), but the level of success has been low. These papers revolve around the use of a back propagation network to recognise and fit grids to a range of known geometries.

A big advantage in the use of neural nets is that it is possible to state problems with a limited understanding of the underlying mathematics. The learning process, involved in working with known answers, allows the neural net to develop and refine an internal model to allow it to then be used against similar but unsolved problems. This capability may be of great importance with adaptive grids in that the user may be able to deal with simple terms to control the grid movement without having to be an expert.

The SONN is a new technique that has been developed to deal with grid adaption in

particular, though it has roots in the Kohonen self-organising map (Kohonen 1990).

8.2 SONN Algorithm

The structure of the SONN exactly mimics the structure of the grid that it is modelling, with each grid node represented by a single PE and the connectivity between the PE's defined by the grid lines. The technique is naturally unstructured. There are no limits to the number of connections between individual PE's, though the current implementation is restricted to two dimensions.

Each PE has a single input, which relates to the function that is used to move the grid, a number of state variables which relate to its physical position and a number of outputs equal to the number of state variables which consist of its new physical position after movement. The input can be modelled as being controlled by a separate layer of specialist PE's which determine the value of the function which is passed to the self organising PE's.

The PE's are organised into neuron groups which relate directly to grid cells. Each PE may belong to many groups. Two neuron groups relating to a six node two dimensional grid are shown in figure 8.1

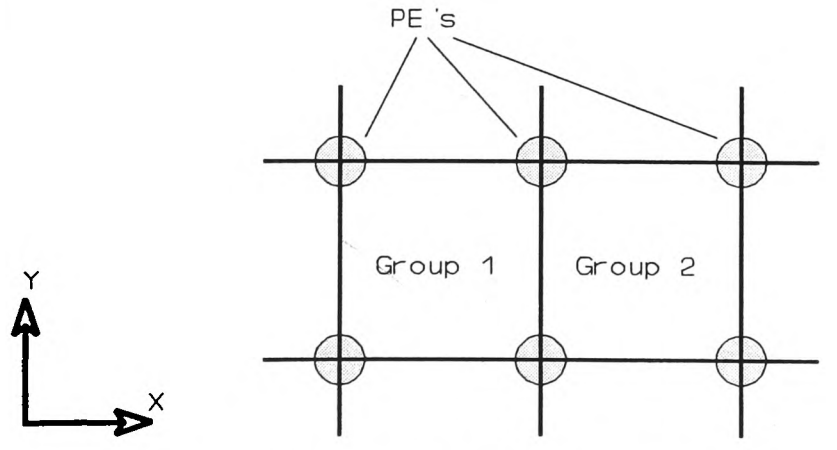


Figure 8.1 Two Dimensional Neuron Group

In this case each PE has two state variables, its X and Y location, and two output variables, its new X and Y locations.

The PE's in each group share a common system energy, known as the information potential (IP). The self organisation process involves the evening out of IP over the network by allowing information to flow between neighbouring groups with different information levels. The IP is defined as the density of the information in a group of PE's, and is a combination of state variables and inputs.

The IP is expressed as the product of a state control function F_S and an input control function F_I.

$$IP = F_I F_S \tag{8.1}$$

For each group the input control function F_I is

$$F_I = \frac{1}{N} \sum_{j=1}^{N} Q_j \tag{8.2}$$

Where the neural group has N PE's and Q_j is the value of the input function on each PE.

In one dimension the state control function F_S is

$$F_S^{1D} = |X_2 - X_1| \tag{8.3}$$

Figure 8.2 One Dimensional Neuron Group

For the one dimensional case the IP is analogous to the equidistribution equation (1.1), with the weight function w related to the input control function F_I, and the grid cell clearance ΔS effectively the same as the state control function F_S.

$$w \Delta x = constant \tag{8.4}$$

In two dimensions the state control function is linked to the area of the cell. Other grid properties, such as orthogonality, aspect ratio, or skewness, could be used instead.

$$F_s^{2D} = \sum_{j=1}^{N} \begin{vmatrix} X_j & X_{j+1} \\ Y_j & Y_{j+1} \end{vmatrix} + \begin{vmatrix} X_N & X_1 \\ Y_N & Y_1 \end{vmatrix} \tag{8.5}$$

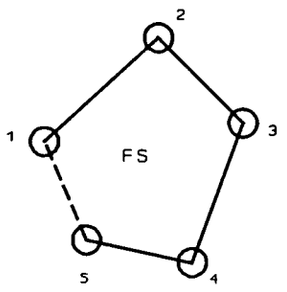


Figure 8.3 State control function F_S

The neural net is stable if the flow of information is small or non-existent. If there is a disturbance in one part of the net its effect is propagated and dissipated by the adjustment of state variables until a new equilibrium is reached.

The equilibrium of the system of N neuron groups is defined as

$$E = \sum_{i=1}^{N-1} (IP_{i+1} - IP_i)^2 \tag{8.6}$$

The learning process finishes when the equilibrium is less than some defined tolerance or when a given number of self organising steps or epochs have been exceeded.

The flow of IP, the 'learning process' takes place over a series of epochs and is controlled by the modification of the state variables of the PE's. The state variable S at iteration t is determined from

$$S_t = S_{t-1} + \Delta S_t \tag{8.7}$$

Where

$$\Delta S_t = -\eta \left.\frac{\partial E}{\partial S}\right|_t + \alpha \Delta S_{t-1} \tag{8.8}$$

E is the equilibrium, defined in equation 8.6, η is the learning rate, and α is the momentum term. Both the learning rate and the momentum term are small and positive. Their value is controlled by the user.

No known solutions or expected output data are used to train the SONN. It organises itself based on input data alone and the user is only interested in its final arrangement. There is no separate processing phase. The only part of the operation of the SONN that is of interest at this time is the training phase.

8.3 Implementation

The SONN is controlled by the user file CONFIG.SNN. It takes data from the file MESH.DAT and outputs a modified version of MESH.DAT call NEW_MESH.DAT.

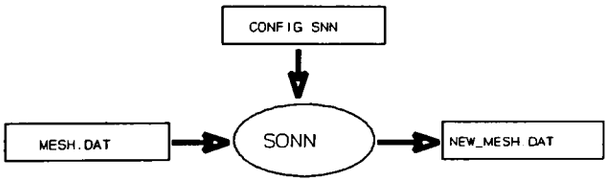


Figure 8.4 SONN call structure

CONFIG.SNN controls the learning rate, the momentum term, convergence criteria and a maximum size ratio between the smallest and largest cells.

MESH.DAT contains data on the location and degrees of freedom for all grid nodes, the definition of the grid cells, and the initial value of the input function at each node. NEW_MESH.DAT has the same format as MESH.DAT but does not contain input function values. The format of these two files is unstructured.

8.4 Neural Net Adaption inside PHOENICS

As a way of demonstrating the use of neural nets adaption with a flow solver the

SONN was used to adapt a grid taken from a problem for which PHOENICS calculated results are available.

In the current work the SONN is essentially a static adaption technique. The reasons for this are that the SONN is a third party code (Zhao et al. 1996) written in C++, whilst PHOENICS and the adaption routines are written in FORTRAN, and that the data structures used by the SONN, being inherently unstructured, are very different from PHOENICS. Given time both problems could be resolved, with the SONN software either rewritten or linked directly to the FORTRAN code, though the implementation would not be platform independent. There is no reason why it could not become dynamic in future.

To give the illusion of dynamic adaption the SONN is run in batch mode with the PHOENICS solver EARTH.

The SONN alone works on information input onto each PE, or grid node. To allow its use to be stretched to practical problems coupled with a flow solver it is necessary to recover useful data on the grid nodes to drive the grid movement. This is the same problem as for the LPE adaption method discussed in this thesis, and the same techniques have been used to solve it. For this reason a preprocessor, NEUPHI, was written to prepare data for the SONN which interpolates cell centred PHOENICS data onto the grid nodes and then applies a weight function. It is worth noting that the raw interpolated data alone would only be useful for a minority of flow problems. Though arguably different neural net techniques could be used to determine the best treatment of the solution data for adaption, at this stage the choice and treatment of the weight function is in the hands of the user.

NEUPHI generates the SONN input file MESH.DAT using the input file NEU.IN. As well as generating the input function it also determines the degrees of freedom of all grid nodes and the nodes which form each grid cell.

A post processor, NEUXYZ, was written to recover a new PHOENICS grid from data

output from the SONN in the file NEW_MESH.DAT. EARTH always picks up the grid file XYZ at the start of each run. NEUXYZ renames the old XYZ file to give a history of the adaption, then writes a new file XYZ.

The final structure of the adaption routine is

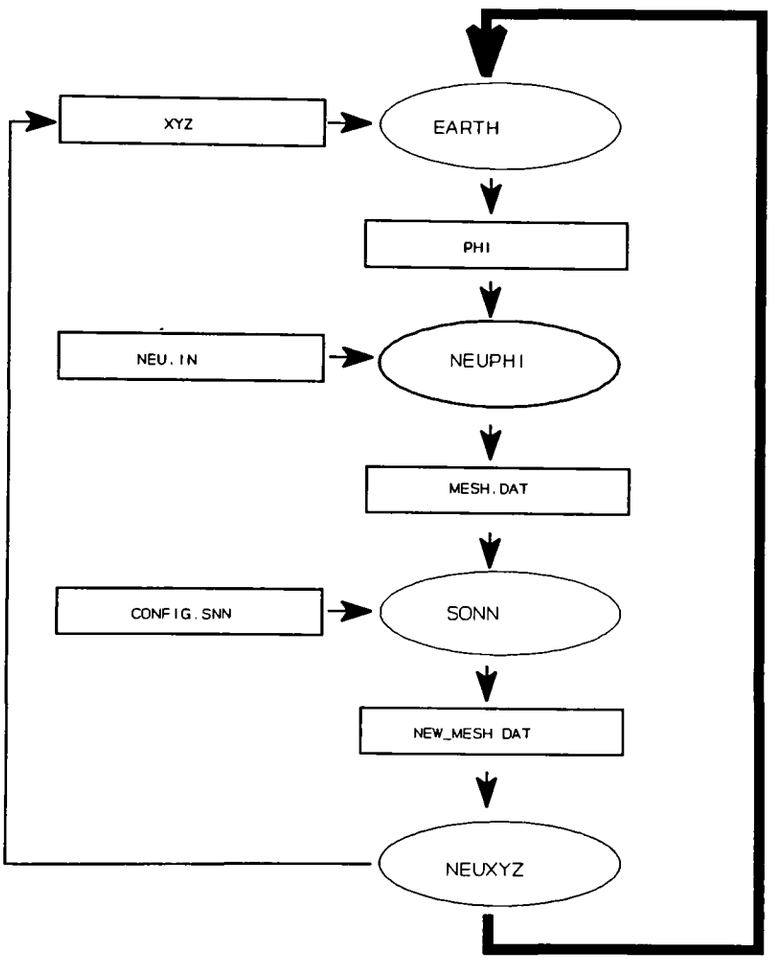


Figure 8.5 Structure of SONN adaption

8.5 Example - X-Y Convection in a Skewed Flow

The choice of example case is limited by the current inability of the neural net to deal with awkward geometry, there being no method to maintain geometry except for fixing the movement of specific grid nodes in one or more directions.

The example case chosen here is a simple diffusion problem solved on a uniform

mesh with an angled flow and different concentrations of a contaminant C which is used to drive the adaption. Four different variations of the case are run, three with the flow entering the domain at different angles and one with the addition of a wall as a boundary condition.

For the purposes of comparison the same problem was also adapted using the LPE method.

8.5.1 Boundary Conditions

The problems are solved over a 30x30 cell square uniform grid of unit dimensions. The problems are set up with four inlets placed at regular intervals over the west and south faces of the domain and outlets on the north and east faces. The inlet velocity is uniform over all the inlets. The value of the contaminant C at the inlets alternates between 0 and 1 and is shown in figure 8.6.

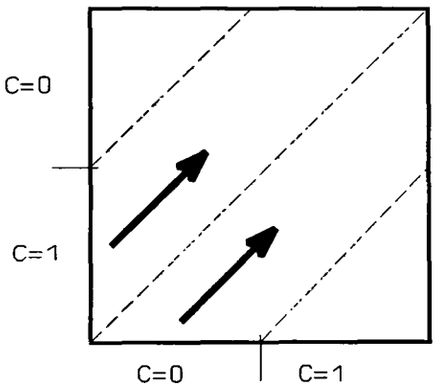


Figure 8.6 Boundary conditions

The problems were run for 500 sweeps.

The four cases are run with the inlet flow at 11°, 22° and 45°. A fourth case is also run with a wall instead of an outlet on the north face and the inlet flow at 45°, forcing the flow to bend over. These changes to the inlet conditions and the outlet were the only differences between the four cases.

8.5.2 Adaption Parameters

All four cases were treated with the same adaption parameters

The whole problem domain was adapted by both methods, with points on all boundaries allowed to slide.

The SONN was run with 200 epochs, a learning rate of 0.1 and a momentum term of 0.8. The SONN moved the grid using the gradient of the contaminant C as the input control function. The gradient was formed by taking the sums of the gradients along the different grid lines smoothed using an x^2 power law.

For the LPE method the parameters were

First call (sweep number)	Last call (sweep number)	Frequency (sweeps)	Internal Iterations	Relaxation
10	100	10	1	1.0

LPE Equation			Variable	Weight Function
λ_L	λ_P	λ_E	Contaminant C	$1+\phi_x+0.01k$ Exponential smoothing
1	0	2		

8.5.3 Results

Due to the batch mode operation of the SONN it is not possible to directly compare the time spent in adaption by the different techniques, but the SONN is not prohibitively expensive.

More user control was required than was first expected to deal with the weight function used to adapt the grid with the SONN. Together with the additional control needed for geometrical control the initial hopes of the SONN being very easy to use may not be justified, though it has not been developed far enough at the current time.

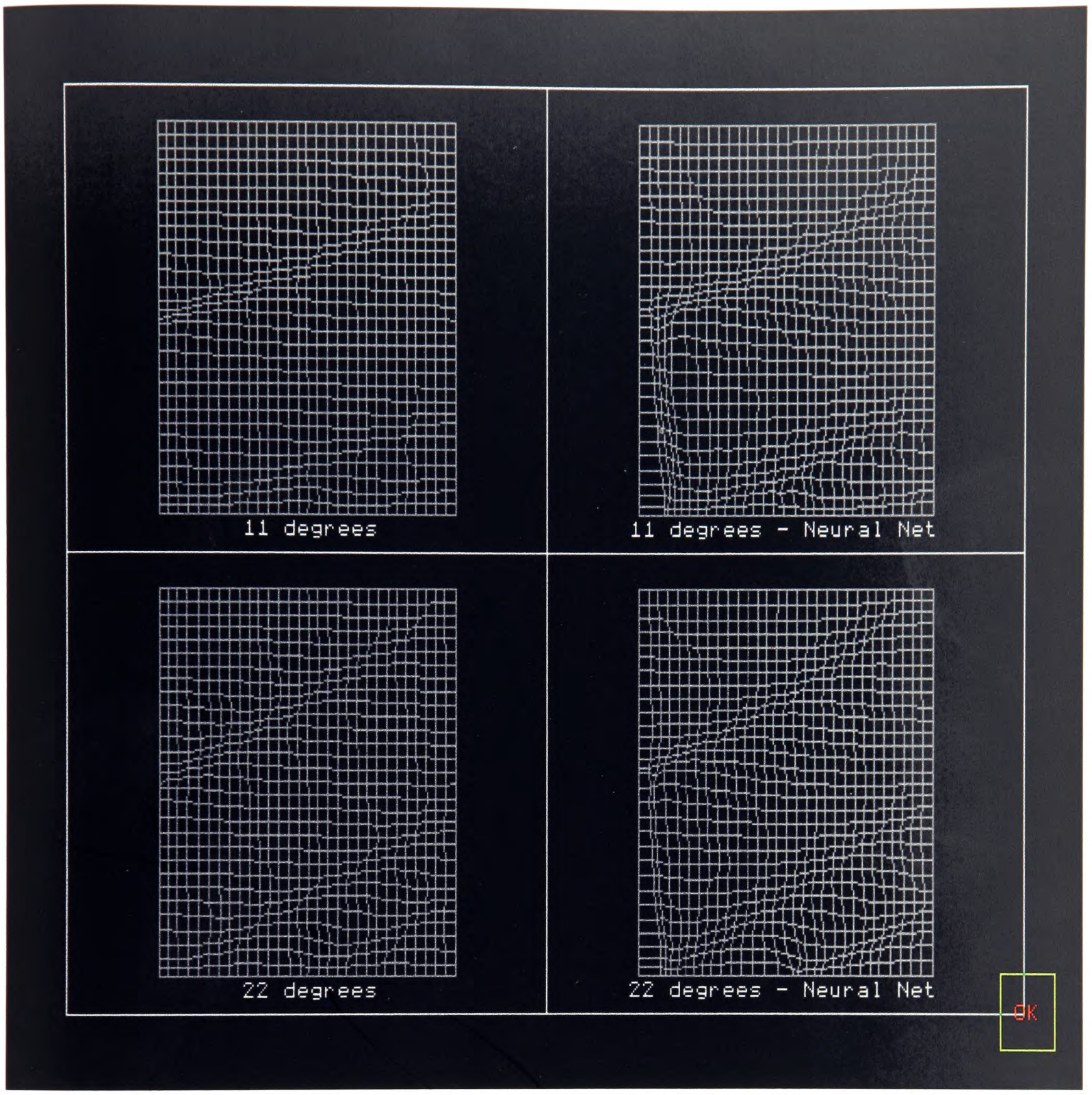


Figure 8.1 Grids adapted by the LPE method and by the SONN

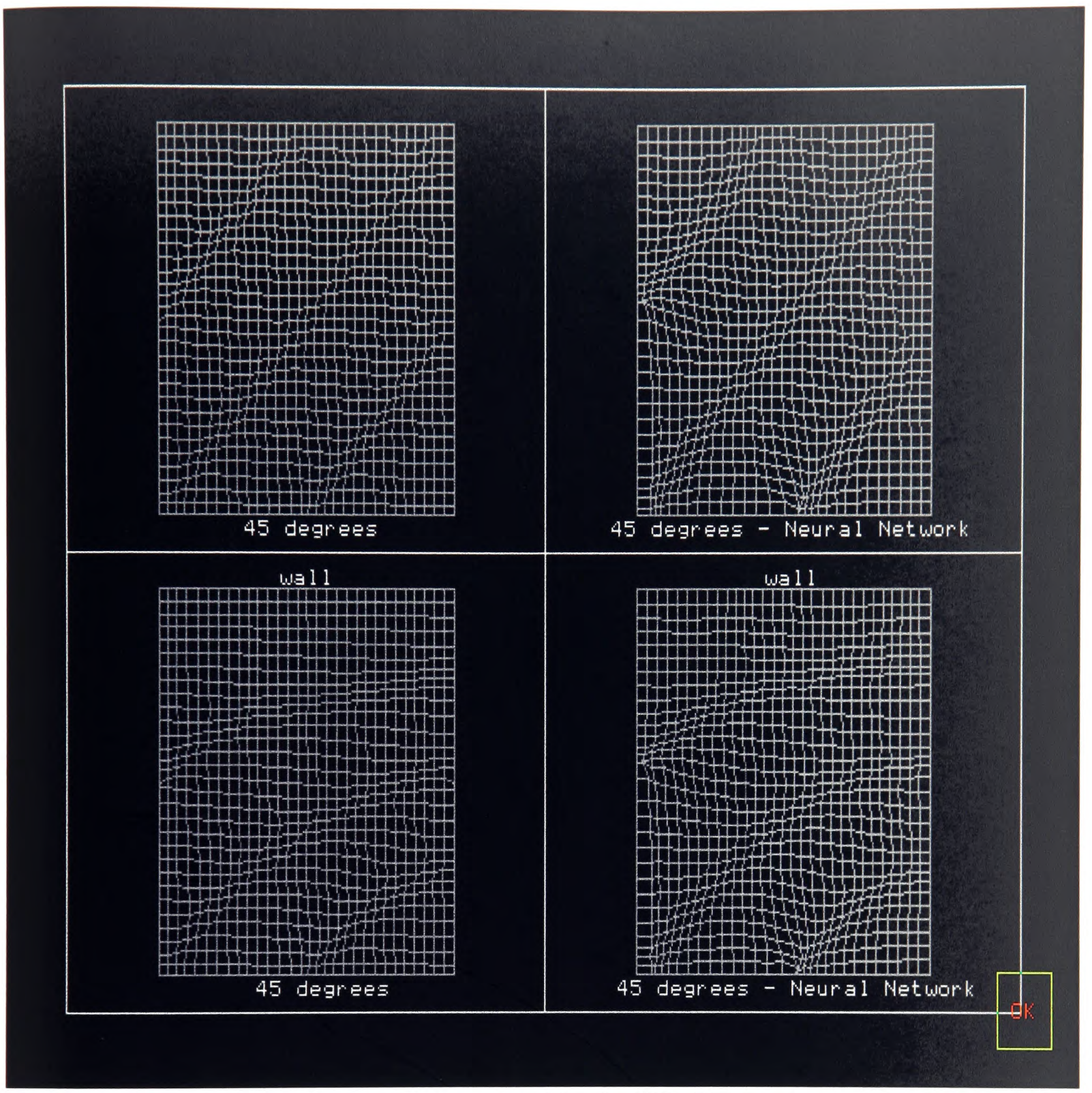


Figure 8.2 Grids adapted by the LPE method and by the SONN

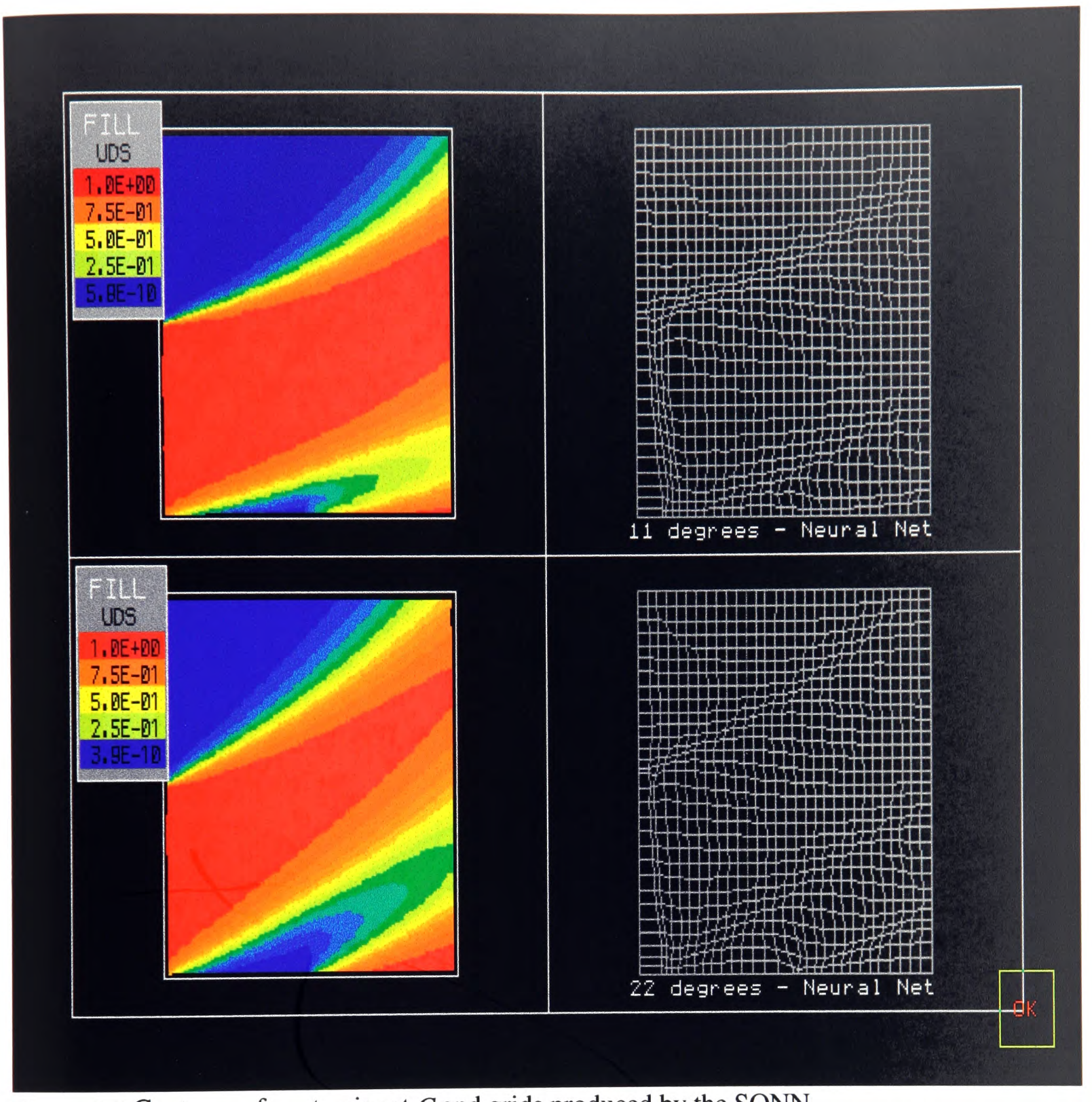


Figure 8.3 Contours of contaminant *C* and grids produced by the SONN

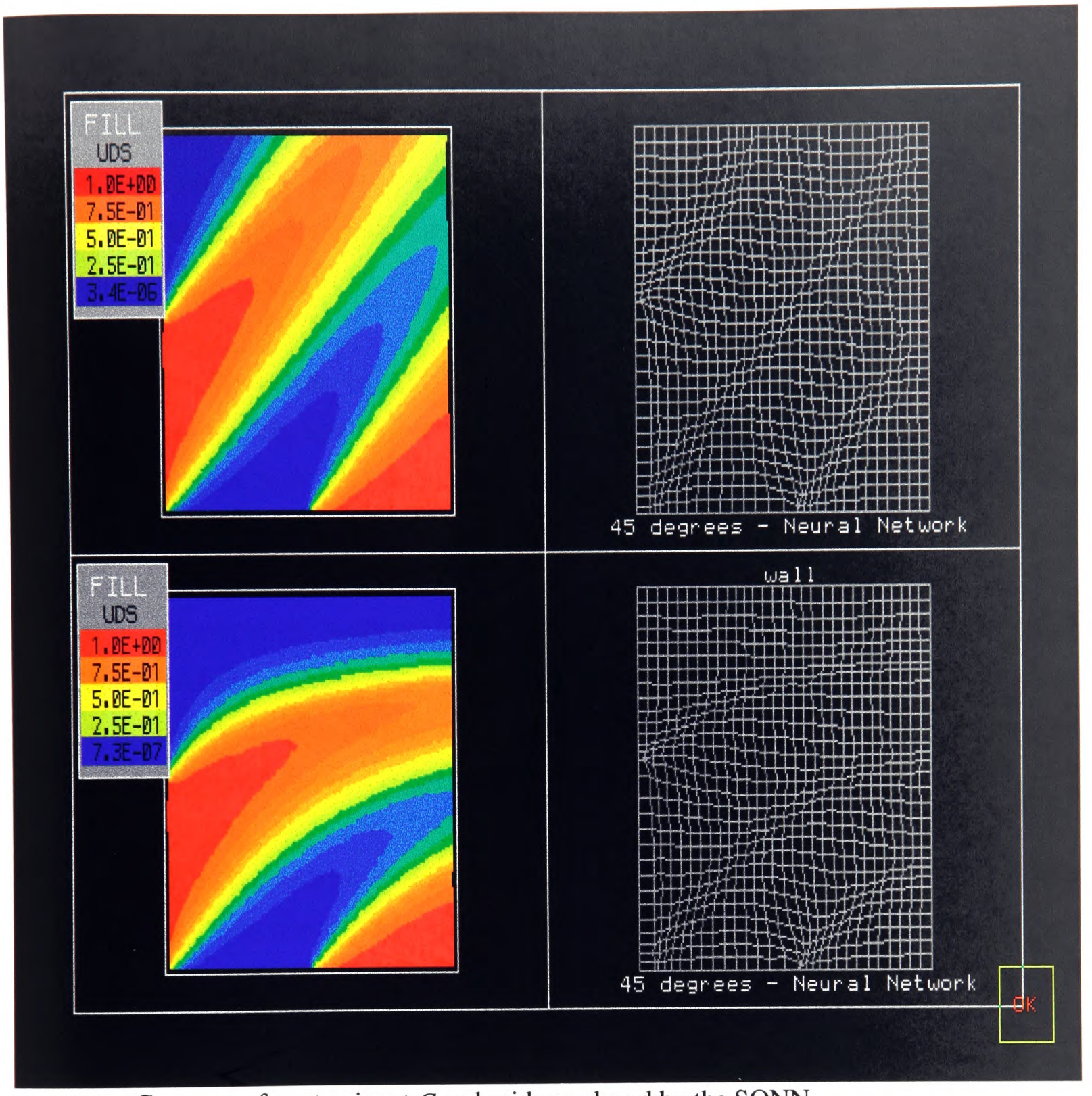


Figure 8.4 Contours of contaminant *C* and grids produced by the SONN

8.6 Closure

The greater experience gained with the use of the LPE adaption algorithm enables a better final grid to be produced, but the SONN has potential. With suitable modifications to the state control functions any chosen grid properties could be improved.

A current limitation of the SONN is that there are no terms controlling grid orthogonality. Though it is very hard for grid cross over to occur it is quite possible that the grid becomes so skewed that the underlying flow code, in this case PHOENICS, will diverge.

A useful part of the work on the SONN has been the recurrence of problems thrown up by the LPE adaption method with the response to solution phenomena. Oscillations, exaggerated response to minor perturbations and lack of response can be countered with the careful use of weight function smoothing, as for the LPE method.

Between each epoch interpolation of the weight function occurs and over many epochs important flow perturbations can be smoothed out and lessen the impact of adaption. The maximum number of epochs carried out each time the SONN is called must be carefully chosen.

Development of a neural net based grid adaption method is continuing at the University of Greenwich as part of a new PhD program (Hu 1998).

CHAPTER 9: CONCLUSIONS AND FUTURE WORK

9.1 General Conclusions

The aim of producing a general, code independent, three dimensional dynamic grid adaption algorithm has been successful. Validation tests presented in chapters five and six show that the LPE technique can improve the presentation and accuracy of a range of flow problems and compares well with other adaptive techniques in literature. Chapter seven shows how the technique can be used as a general tool for controlling all kinds of grid movement.

The work has thrown up a range of points that relate to all adaptive grid methods, as has been found with the work on neural net grid adaption using the SONN method.

- Some knowledge of how the adaptive algorithm works is necessary to get most out of it. Though a lot of work has gone into making the algorithm safe, including monitoring the solution residuals to check for divergence, the time spent choosing and testing the right adaption parameters may make adaption uneconomic.

- There must be some property of the flow that provides a strong gradient or easily identifiable phenomena to drive grid adaption.

- Grid adaption is not guaranteed to move phenomena that occur out of place in a coarse grid to their correct positions.

- The initial resolution of the grid covering the problem domain is critical. It must be fine enough to pick up the phenomena that the adaptive algorithm is designed to enhance. In contrast too fine a grid and adaption will not make any difference.

- Tolerance to grid quality in the underlying CFD code is crucial. It is quite

possible to get grids that appear reasonable but cause the code to fail.

• The grid movement needed to improve the solution accuracy may be less than anticipated (e.g. the Laval nozzle case 5.3.3). Excessive movement may outweigh the benefits of grid alignment and density on accuracy because of the associated loss of grid quality. Because of this the visual impact on the grid of adaption, which is a very useful and important side effect, may lead to unrealistic hopes about the quality of the final results.

• The grid at each stage of adaption can only ever be as good as the data available at the time when adaption takes place. It is not possible to get the best grid for the final solution at the first pass of the adaptive grid solver. Indeed, if a grid moves too fast to a temporary solution that has a solution feature in the wrong position it may never be possible to recover its correct position as the grid in that area may be too coarse. As several passes of the adaptive solver should be used in the running of a particular problem so that the grid evolves with the solution, the main aim of the adaption algorithm should be to make the grid a little better at each call, not necessarily the best possible.

The ideal adaption algorithm would involve very small movements at every iteration of the flow solver after a certain level of convergence has already been achieved. In practice this must be balanced against the requirements of time and disruption to the flow solver due to solution interpolation and the need to recompute geometrical data.

9.1.1 Additional Tools

An important part of the work has been the development of a range of tools to model and maintain grid geometry and to manipulate the behaviour of the LPE algorithm. These tools are modular and are not tied to the LPE algorithm in particular.

Curve and surface modelling using cubic splines and bicubic patches and breaking at

important geometrical features works well and are useful where complex geometry must be maintained, as in the 10 degree supersonic wedge intake case (5.3.5), or where important features of the grid need to be maintained, such as the boundary layer cells in the RAE2822 aerofoil case (5.3.1).

There are weaknesses in the technique. The grid overlapping algorithm is not perfect but will behave on a reasonable grid with reasonable movement. Shepard's interpolation to transfer data between grids is safe, easy to implement and can be fast but may not be particularly accurate.

9.1.2 The Weight Function

A strength of the current work is the treatment of the weight function which drives grid point movement in the adaption algorithm. To stretch the technique from heat conduction problems and convective flow to transonic and supersonic cases with strong shock waves demands a range of tools to manipulate the variables used to generate the weight function and to manipulate the function itself, the most important being power law smoothing.

The aim of the weight function is to indicate where there should be a higher concentration of grid cells. Though its calculation depends upon some solution behaviour it need not be tied too closely to that behaviour to generate the best possible grid. Properties such as smoothness and orthogonality may be lost if the grid is moved exactly to the variation of some solution data. With this in mind the novel technique of weight function smoothing using power laws to influence the range of effect of phenomena and grid smoothness has been developed. The technique is fast and instantly produces a smoothly varying function to adapt the grid to, avoiding regions adjacent to high weight values with a low grid density. The closest techniques presented in literature may be the monitor surface used by Eiseman (1987) and adaption in parametric space discussed in section 1.2.1.5.

9.2 Future Work

The main aim of this thesis is to show that the LPE algorithm, with a range of extra tools, can be used to improve the performance of CFD codes over a wide range of CFD problems. This aim has been achieved, but there is always room for improvement. A lot of work has gone into making the technique safe and usable, but this is still an area where more work is needed.

9.2.1 Transient Cases

The only transient case presented in the current work is the example case used to demonstrate the coupling of stress and fluid flow (7.4). Transient cases are currently wholly dependent on the underlying flow code, in this case PHOENICS, and the response of the code to grid movement has not always been predictable. More work is needed to show that the LPE algorithm is truly useful for transient flow problems.

There may also be problems dealing with a very rapidly varying solution where high gradient regions may move to areas of the grid already coarsened in previous time steps which may now be unable to accurately resolve the solution enough to attract grid nodes back. This will limit the maximum timestep available for the problem. This is an area where mesh refinement may work better, as the underlying mesh will always be the same. There are some problems, such as the modelling of detonation waves, where mesh movement schemes alone may be unable to achieve the length scales needed to accurately resolve the problem and some sort of grid refinement scheme would have to be used as well, or instead of grid redistribution.

9.2.2 Three Dimensional Cases

The LPE algorithm is fully three dimensional, and its use for three dimensional problems has been demonstrated in chapter 6. However, the problems in chapter 6 only show that it works. No validation has been presented since it is much more difficult to compare the adapted and non adapted solutions for three dimensional problems than for two dimensional problems. More work is needed to stretch the limits of the algorithms performance and prove its value. Suitable three dimensional

cases are also needed for validation.

9.2.3 New Solvers

The LPE method as implemented here uses the simplest and most reliable of solvers, a relaxed point by point iteration scheme. It would however be possible to use some form of matrix algorithm such as the TDMA scheme to increase the speed of operation and convergence of the adapted grid. Such schemes may help to improve the coupling of the response of grid points at different extremes of the adaptive domain, and in problems such as the coupling of stress and fluid flow (7.4) will allow the grid to sort itself out quickly.

Such schemes are beyond the scope of the thesis as they cannot show that the algorithm works any more than the current scheme does, but maybe more important in developing the usefulness of the algorithm as a genuine CFD tool. An example of where they may be useful is in improving the computation time of three dimensional problems.

9.2.4 Linking with Other Codes

The LPE algorithm presented in this thesis is designed to be code independent. Though it has been developed as an attachment to PHOENICS, only a handful of routines depend directly upon it. It would be straightforward to transfer the algorithm to any other flow code using a structured grid or even an unstructured code using rectangular or cuboid elements.

It may also be possible to make the algorithm unstructured though this would mean that a different scheme would have to be used to discretise the LPE equation, and the powerful weight smoothing techniques used in structured grids would have to be heavily modified or discarded. A possible way of using function based weight smoothing would be to generate some kind of smooth weight surface for two dimensional problems or hypercube for three.

9.2.5 Refinement and Multiblock Implementation

Hybrid schemes coupling refinement and movement are very powerful. The problem lies in finding a fluid code that is capable of dealing with hybrid grids. One option would be to use the multiblock version of PHOENICS.

Multiblock facilities in PHOENICS have been available from version 2.1 onwards. Adaption within individual blocks in the grid is straightforward and requires no modification to the current implementation of the LPE method as it is based on individual domains within the problem area. The extension to the whole problem domain and in particular the crossover region between coarse and embedded fine meshes is not straightforward. Alternative options for dealing with embedded fine grids include:

- Moving all points over the coarsest grid covering the whole domain, including the region covered by fine grid. This is possible as there will be a direct correspondence between points in the fine mesh and the coarse. Points in the parts of the coarse grid covered by the fine grid would have a function value taken from the fine grid. After the adaption of the coarse grid the unmoved grid points in the fine grid could either be modified by interpolation or by adaption with the previously moved points fixed.

- defining a new method to deal with boundary regions between separate adaption domains. Fine regions could be blocked out of coarse grids.

Use of the knowledge gained through manipulation of the weight function could be used also to determine regions in an initial single domain that could benefit from refinement based on some initial solution. As dynamic allocation of memory is very limited in FORTRAN based PHOENICS it may only be possible to implement this method as a static option. It would only be possible to modify the grid between runs.

Unfortunately it was not possible to have full access to the multiblock version of

PHOENICS during the course of this thesis.

9.2.6 Automation of Parameter Choice

In the current method the user controls not only the choice of the adapted regions but also up to 15 adaption parameters. Knowledge of the technique is needed to get the most out of it. Any techniques that can be used to reduce this number of parameters will greatly increase the usability of the algorithm.

It would be possible to automatically break the problem domain down into areas that need adaption by using some measure of solution error or the first and second derivatives of a range of solution variables. This would be similar to the measures used to flag areas to be refined in grid refinement techniques (1.4.2). Such data could also be used to determine which variable or variables are used to drive grid adaption and their treatment.

Such a technique would be very powerful in that it would make the use of adaption safer and lessen the need for the user to be an expert in adaptive grids. It may not be easy to implement, though the pattern matching abilities of some kinds of neural nets may be a good starting point.

REFERENCES

Acharya, S. Moukalled, F.H. (1990)
An Adaptive Grid Solution Procedure for Convection-Diffusion Problems
Journal of Computational Physics Vol.91 No.1 32-54

Acharya, S. Patankar, S.V. (1985)
Use of an Adaptive Grid Procedure for Parabolic Flow Problems
International Journal of Heat and Mass Transfer Vol.28 No.6 1057-66

Adams, E.C. Conlisk, A.T. (1995)
Adaptive Grid Scheme for Vortex-Induced Boundary Layers
AIAA Journal Vol.33 No.5 864-70

Agbormbai, A. Patel, M Pericleous, K. (1991)
Second Progress Report on Adaptive Grids for Shock Capture in PHOENICS
Internal report Thames Polytechnic

Albone, C.M. (1988)
An Approach to Geometric and Flow Complexity using Feature-Associated Mesh Embedding (FAME): Strategy and First Results
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 215-35

Allen, C.B. (1995)
An Implicit Upwind Scheme with Grid Adaption for Unsteady Euler Aerofoil Flows
9th International Conference on Numerical Methods in Laminar and Turbulent Flow 445-56

Allwright, S.E. (1988)
Multiblock Techniques for Transonic Flow Computation about Complex Aircraft Configurations
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 367-74

Anderson, D.A. (1990)
Grid Cell Volume Control with an Adaptive Grid Generator
Applied Mathematics and Computation Vol.35 209-17

Anderson, D.A. (1987a)
Equidistribution Schemes, Poisson Generators, and Adaptive Grids
Applied Mathematics and Computation Vol.24 211-27

Anderson, D.A. (1987b)
An Adaptive Grid Scheme Controlling Cell Area/Volume
AIAA Paper 87-0202

Arina, R. (1989)
Three Dimensional Adaptive Grids with Limited Skewness
AGARD cp464 Paper 12

Arina, R. (1988)
Adaptive Orthogonal Curvilinear Co-ordinates
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 353-9

Arney, D.C. Flaherty, J.E. (1990)
An Adaptive Mesh Moving and Local Refinement Method for Time Dependant Partial Differential Equations
ACM Transactions on Mathematical Software vol.16 No.1 48-71

Arney, D.C. Flaherty, J.E. (1986)
A Two Dimensional Mesh Moving Technique for Time-Dependent Partial Differential Equations
Journal of Computational Physics Vol.67 124-44
Bennett, C.R. (January 1994)
User Manual for Earth Part of Adaptive Code
Communication to BAe Farnborough
Bennett, C.R. (August 1993)
User Manual for Satellite Part of Adaptive Code
Communication to BAe Farnborough
Bennett, C.R. (October 1992)
Grid Adaption using the Laplace-Poisson-Equidistribution Equation
MSc Thesis, University of Greenwich
Benson, R.A. McRae, D.S. (1991)
A Solution-Adaptive Mesh Algorithm for Dynamic/Static Refinement of Two and Three Dimensional Grids
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 185-99
Berger, M.J. Colella, P. (1989)
Local Adaptive Mesh Refinement for Shock Hydrodynamics
Journal of Computational Physics Vol.82 64-84
Berger, M.J. (1986)
Data Structures for Adaptive Grid Generation
SIAM Journal of Scientific and Statistical Computation Vol.7 No.3 904-16
Biswas, R. Flaherty, J.E. Arney, D.C. (1993)
An Adaptive Mesh-Moving and Refinement Procedure for One-Dimensional Conservation Laws
Applied Numerical Mathematics Vol.11 259-282
Blom, J.G. Verwer, J.G. (1994a)
VLUGR2: A Vectorized Local Uniform Grid Refinement Code for PDE's in 2D
Report NM-R9403, CWI, Amsterdam
Blom, J.G. Verwer, J.G. (1994b)
VLUGR3: a vectorizable adaptive grid solver for PDEs in 3D, part I: algorithmic aspects and applications
Applied Numerical Mathematics, Vol.16, No.1-2 129
Blom, J.G. Verwer, J.G. (1994c)
VLUGR3: A Vectorizable Adaptive Grid Solver for PDE's in 3D II. Code Description
Report NM-R9405, CWI, Amsterdam
Bockelie, M.J. Eiseman, P.R. Smith, R.E. (1990)
A General Purpose Time Accurate Adaptive Grid Method
Lecture Notes in Physics, No.371, 524-8
Bowyer, A. Woodwark, J. (1983)
A Programmer's Geometry
London, Butterworths
Brackbill, J.U. (1993)
An Adaptive Grid with Directional Control
Journal of Computational Physics Vol.108 38-50

Brackbill, J.U. Saltzman, J.S. (1982)
Adaptive Zoning for Singular Problems in Two Dimensions
Journal of Computational Physics Vol.46 342-68
Carey, C. Scanlon, T. J. Fraser, S.M. (1993)
SUCCA - an alternative scheme to reduce the effects of multidimensional false diffusion
Applied Mathematical Modelling Vol.17 263-70
Carey, G.F. Sharma, M. Wang, K.C. Pardhanani, A. (1988)
Some Aspects of Adaptive Grid Computations
Computers and Structures Vol.30 No.1/2 297-302
Carey, G.F. (1987)
Mesh Refinement and Redistribution
Numerical Methods for Transient and Coupled Problems (Eds. R. W. Lewis, E. Hinton, P.Bettess, and B.A. Schrefler) John Wiley 1-12
Castillo, J.E. (1991)
An Adaptive Direct Variational Grid Generation Method
Computers and Mathematics with Applications Vol.21 No.5 57-64
Catherall, D. (1991)
The Adaption of Structured Grids to Numerical Solutions for Transonic Flow
International Journal for Numerical Methods in Engineering Vol.32 921-37
Chang, P.Y. Shyy, W. (1991)
Adaptive Grid Computation of Three-Dimensional Natural Convection in Horizontal High-Pressure Mercury Lamps
International Journal for Numerical methods in Fluids Vol.12 143-60
Chao, Y.C. Liu, S.S. (1991)
Streamline Adaptive Grid Method for Complex Flow Computation
Numerical Heat Transfer, Part B Vol.20 145-68
Chawner, J.R. Anderson, D.A. (1991)
Development of an Algebraic Grid Generation Method with Orthogonality and Clustering Control
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 107-17
Coyle, J.C. Flaherty, J.E. Ludwig, R. (1986)
On the Stability of Mesh Equidistribution Strategies for Time-Dependent Partial Differential Equations
Journal of Computational Physics Vol.62 26-39
Cross, M. Chow, P. Bailey, C. Croft, N. Ewer, J. Leggett, P. McManus, K. Pericleous, K.A. Patel, M.K. (1995)
PHYSICA - A Software Environment for the Modelling of Multi-Physics Phenomena
Proc. ICIAM 95, July 1995
Dandekar, H.W. Hlavacek, V. (1993)
An Explicit 3D Finite-Volume Method for Simulation of Reactive Flows using a Hybrid Moving Adaptive Grid
Numerical Heat Transfer, Part B Vol.24 1-29
Dannenhoffer III, J.F. (1991)
A Comparison of Adaptive-Grid Redistribution and Embedding for Steady Transonic Flows
International Journal for Numerical methods in Engineering Vol.32 653-63

Dannenhoffer III, J.F. (1987)
Grid Adaption for Complex Two Dimensional Transonic Flows
ScD Thesis, Massachusetts Institute of Technology, August 1987

Dannenhoffer III, J.F. (1985)
Grid Adaption for the 2D Euler Equations
AIAA Paper 85-0484

Davies, C.B. Venkatapathy, E. (1992)
Application of a Solution Adaptive Grid Scheme to Complex Three-Dimensional Flows
AIAA Journal Vol.30 No.9 2227-33

Davies, R.L. Dannenhoffer, J.F. (1994)
Three Dimensional Adaptive Grid-Embedding Euler Technique
AIAA Journal Vol.32 No.6 1167-74

Desbois, F. Jacquotte, O-P. (1991)
Surface mesh Generation and Optimization
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 131-42

De Zeeuw, D. Powell, K.G. (1993)
An Adaptively Refined Cartesian Mesh Solver for the Euler Equations
Journal of Computational Physics Vol.104 56-68

Donea, J. Giuliani, S. Halleux, J.P. (1982)
An Arbitrary Lagrangian-Eulerian Finite Element Method for Transient Dynamic Fluid-Structure Interactions
Comp. Meth. in Applied Mechanics and Engineering Vol.33 689-723

Dvinsky, A.S. (1991)
Adaptive Grid Generation from Harmonic Maps on Riemannian Manifolds
Journal of Computational Physics Vol.95 450-76

Dwyer, H.A. (1985)
Generation of Fully Adaptive and/or Orthogonal Grids
Proceeding of the 9th International Conference on Numerical Methods in Fluid Dynamics (Eds. Soubbaramayer and Broujot) 203-7 Springer-Verlag

Dwyer, H.A. (1984)
Grid Adaption for Problems in Fluid Dynamics
AIAA Journal Vol.22 No.12 1705-12

Edwards, M.G. Oden, J.T. Demkowicz, L. (1993)
An h-r-Adaptive Approximate Riemann Solver for the Euler Equations in Two Dimensions
SIAM Journal of Scientific Computing Vol.14 No.1 185-217

Eiseman, P.R. (1987)
Adaptive Grid Generation
Computer Methods in Applied Mechanics and Engineering Vol.64 321-76

Eiseman, P.R. Bockelie, M.J. (1987)
Adaptive Grid Solution for Shock Vortex Interaction
Proceeding of the 11th International Conference on Numerical Methods in Fluid Dynamics 240-3 Springer-Verlag

Eiseman, P.R. (1986)
Alternating Direction Adaptive Grid Generation for Three Dimensional Regions
Proceeding of the 10th International Conference on Numerical Methods in Fluid Dynamics (Lecture Notes in Physics 264) 258-63 Springer-Verlag

Eiseman, P.R. (1985a)
Grid Generation for Fluid Mechanics Computations
Annual Review of Fluid Mechanics Vol.17 487-522
Eiseman, P.R. (1985b)
Adaptive Grid Generation by Mean Value Relaxation
Journal of Fluids Engineering, Transactions of the ASME Vol.107 No.4 477-83
Ewing, R.E. Espedal, M.S. Puckett, J.A. Schmidt, R.J. (1988)
Simulation Techniques for Multiphase and Multicomponent Flows
Communications in Applied Numerical Methods Vol.4 335-42
Farhat, C. Lin, T.Y. (1990)
Transient Aeroelastic Computations using Multiple Moving Frames of Reference
AIAA Paper 90-3053
Farhat, C. (1995)
High Performance Simulation of Coupled Nonlinear Transient Aeroelastic Problems
Private Communication
FEMGEN, FEMVIEW
Femsys Ltd, 1 St Albans Rd., Leicester LE2 1GF
Forsey, C.R. Billing, C.M. (1988)
Some Experience with Grid Generation on Curved Surfaces using Variational and Optimisation techniques
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 341-52
Fraga, E.S. Morris, J. Ll (1992)
An Adaptive Mesh Refinement Method for Nonlinear Dispersive Wave Equations
Journal of Computational Physics Vol.101 94-103
Franca, A.S. Haghighi, K. (1994)
Adaptive Finite Element Analysis of Transient Thermal Problems
Numerical Heat Transfer, Part B Vol.26 273-92
Franke, R. Nielson, G. (1980)
Smooth Interpolation of Large Sets of Scattered Data
International Journal for Numerical methods in Engineering Vol.15 1691-704
Ghia, U. Ghia, K.N. Shin, C.T. (1982)
High-RE Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method
Journal of Computational Physics Vol.48 387-411
Giannakopoulos, A.E. Engel, A.J. (1988)
Directional Control in Grid Generation
Journal of Computational Physics Vol.74 422-39
Gnoffo, P.A. (1983)
A Finite-Volume, Adaptive Grid Algorithm applied to Planetary Entry Flowfields
AIAA Journal Vol.21 No.9 1249-54
Gordon, W.J. Wixom, J.A. (1978)
Shepard's Method of Metric Interpolation to Bivariate and Multivariate Interpolation
Mathematics of Computation Vol.32 No.141 253-64
Greenburg, J.B. (1983)
A New Self-Adaptive Grid Method
AIAA Paper 83-1934

Guj, G. Stella, F. (1993)
A Vorticity-Velocity Method for the Numerical Solution of 3D Incompressible Flows
Journal of Computational Physics Vol.106 286-98
Haase, W. Misegades, K. Naar, M. (1985)
Adaptive Grids in Numerical Fluid Dynamics
International Journal for Numerical methods in Fluids Vol.5 515-28
Hagmeijer, R. (1994)
Grid Adaption Based on Modified Anisotropic Diffusion Equations Formulated in the Parametric Domain
Journal of Computational Physics Vol.115 169-83
Hall, D.J. Zingg, D.W. (1995)
Viscous Airfoil Computations Using Adaptive Grid Redistribution
AIAA Journal Vol.33 No.7 1205-10
Hamdan, F.H Dowling, P.J. (1995)
Fluid-Structure Interaction: Application to Structures in an Acoustic Fluid Medium, Part 1: An Introduction to Numerical Treatment
Engineering Computations Vol.12, 749-58
Hartle, S.L. (1995)
Neural Network Grid Generation
Paper presented at the CFDCC Seminar, University of Bristol 12 September 1995
Harvey, A.D. Acharya, S. Lawrence, S.L. (1993)
Space Marching Calculations About Hypersonic Configurations Using a Solution Adaptive Mesh Alorithm
AIAA Journal Vol.31 No.10 1809-18
Harvey III, A.D. Acharya, S. Lawrence, S.L. Cheung, S. (1991)
Solution-Adaptive Grid Procedure for High-Speed Parabolic Flow Solvers
AIAA Journal Vol.29 No.8 1232-40
Hawken, D.F. Gottlieb, J.J. Hansen, J.S. (1991)
Review of Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations
Journal of Computational Physics Vol.95 254-302
Holcomb, J.E. Hindman, R.G. (1984)
Development of a Dynamically Adaptive Grid method for Multidimensional Problems
AIAA Paper 84-1668
Hsu, C.C. Tu, C.G. (1987)
Self-Adaptive Gridding for Inviscid Transonic Projectile Aerodynamics Computation
International Journal for Numerical Methods in Fluids Vol.7 No.6 567-79
Hsu, K. Lee, S.L. (1991)
A Numerical Technique for Two-Dimensional Grid Generation with Grid Control at All of the Boundaries
Journal of Computational Physics Vol.96 451-69
Huang, H. Prosperetti, A. (1994)
Effect of Grid Orthogonality on the Solution Accuracy of the Two-Dimensional Convection-Diffusion Equation
Numerical Heat Transfer, Part B Vol.26 1-20
Hu, L. (1998)
Adaptive Meshing within Smartfire
Internal report at the University of Greenwich

Jacquotte, O-P. (1991)
Recent Progress on Mesh Optimization
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 581-96

Jacquotte, O-P. Coussement, G. (1992)
Structured Mesh Adaption: Space Accuracy and Interpolation Methods
Computer Methods in Applied Mechanics and Engineering Vol.101 397-432

Jacquotte, O-P. (1992)
Generation, Optimisation, and Adaption of Multiblock Grids around Complex Configurations in Computational Fluid Dynamics
International Journal for Numerical methods in Engineering Vol.34 443-54

Jeng, Y.N. Chen, J.L. (1992)
Geometric Conservation Law of the Finite-Volume Method for the SIMPLER Algorithm and a Proposed Upwind Scheme
Numerical Heat Transfer, Part B Vol.22 No.2 211-34

Jeng, Y.N. Liou, S.C. (1993)
Adaptive Grid Generation by Elliptic Equations with Grid Control at All of the Boundaries
Numerical Heat Transfer, Part B Vol.23 135-51

Jeng, Y.N. Liou, S.C. (1992a)
Two Modified Versions of Hsu-Lee's Elliptic Solver of Grid Generation
Numerical Heat Transfer, Part B Vol.22 125-40

Jeng, Y.N. Liou, S.C. (1992b)
A New Adaptive Grid Generation by Elliptic Equations with Orthogonality at All of the Boundaries
Journal of Scientific Computing Vol.7 No.1

Jeng, Y.N. Liou, S.C. (1989)
Modified Multiple One-Dimensional Adaptive Grid Method
Numerical Heat Transfer, Part B Vol.15 241-7

Kallinderis, Y. (1992)
Finite Volume Navier-Stokes Algorithm for Adaptive Grids
International Journal for Numerical Methods in Fluids Vol.15 No.2 193-217

Khan, A.I. Topping, B.H.V. Bahreininejad, A. (1993)
Parallel training of neural networks for finite element mesh generation
Civil-Comp93, Part 3: Neural Networks and Combinatorial Optimization in Civil and Structural Engineering, 1993, p.81

Kim, H.J. Thompson, J.F. (1990)
Three Dimensional Adaptive Grid Generation on a Composite Block Grid
AIAA Journal Vol.28 no.3 470-7

Kim, H.J. (1987)
Three Dimensional Adaptive Grid Generation on a Composite Structure
PhD Thesis, Mississippi State University

Knupp, P.M. (1992)
A Robust Elliptic Grid Generator
Journal of Computational Physics Vol.100 409-18

Kohonen, T. (1990)
The Self-Organizing Map
Proceedings of the IEEE Vol.78 No.9 1464-80

Kwon, J.H. Jeong, H.K. (1995)
Solution Adaptive Grid Generation for Compressible Flow
Computers and Fluids Vol.25 No.6 551-60

Lawal, A. (1990)
Adaptive Grid Method for Convection-Diffusion Equations
International Journal of Heat and Mass Transfer Vol.33 No.8 1633-41

Lee, D. Yeh, C.L. (1994)
Computation of Turbulent Recirculating Flows Using a Hybrid Adaptive Grid
Numerical Heat Transfer, Part A Vol.26 415-30

Lee, D. Tsuei, Y.M. (1993)
A Hybrid Adaptive Gridding Procedure for Recirculating Fluid Flow Problems
Journal of Computational Physics Vol.108 122-41

Lee, D. Tsuei, Y.M. (1992a)
A Modified Adaptive Grid Method for Recirculating Flows
International Journal for Numerical Methods in Fluids Vol.14 775-91

Lee, D. Tsuei, Y.M. (1992b)
A Formula for Estimation of Truncation Errors of Convection Terms in a Curvilinear Co-ordinate System
Journal of Computational Physics Vol.98 90-100

Lee, K.D. Henderson, T.L. Choo, Y.K. (1991)
Grid Quality Improvement by a Grid Adaption Technique
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 597-606

Lee, K.D. Loellbach, J.M. Kim, M.S. (1990)
Adaption of Structured Grids for Improved Navier Stokes Solution
AIAA Paper 90-0125

Lee, K.D. Loellbach, J.M. (1989a)
A Mapping Technique for Solution Adaptive Grid Control
AIAA paper 85-0486 Proceedings of AIAA 7th Applied Aerodynamics Conference p129-39 July 1989

Lee, K.D. Loellbach, J.M. (1989b)
Geometry-Adaptive Surface Grid Generation Using a Parametric Projection
Journal of Aircraft Vol.26 No.2 162-7

Lehtimäki, R. (1995)
Grid Optimization based on Grid Quality Measures
9th International Conference on Numerical Methods in Laminar and Turbulent Flow 445-56

Lesoinne, M. Farhat, C. (1995)
Geometrical Conservation Laws for Aeroelastic Computations using Unstructured Dynamic Meshes
AIAA paper 95-1709

Lin, S-Y. Wu, T-M. (1993)
An Adaptive Multigrid Finite-Volume Scheme for Incompressible Navier-Stokes Equations
International Journal for Numerical methods in Fluids Vol.17 687-710

Lu, N. Eiseman, P.R. (1991)
A Grid Quality Manipulation System
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 607-16

Matsuno, K. Dwyer, H.A. (1988)
Adaptive Methods for Elliptic Grid Generation
Journal of Computational Physics Vol.77 40-52

Mattheij, R.M.M. Smooke, M.D. (1989)
Stability and Convergence of Linear Parabolic Mixed Initial-Boundary Value Problems on Nonuniform Grids
Applied Numerical Mathematics Vol.6 471-85

Mavriplis, C. (1992)
Adaptive Mesh Strategies for the Spectral Element Method
ICASE Report No.92-36

Molenaar, J. (1995)
Adaptive Multigrid applied to a Bipolar Transistor Problem
Applied Numerical Mathematics Vol.17 61-83

Montgomery, M. Fleeter, S. (1995)
The Effect of Grid Irregularity on Truncation Error for Discretizations of Laplace's Equation
International Journal for Numerical Methods in Engineering Vol.38 3243-57

Moukalled, F. Acharya, S. (1991)
Local adaptive grid Procedure for Incompressible Flows with Multigridding and Equidistribution Concepts
International Journal for Numerical Methods in Fluids Vol.13 No.9 1085-111

Nakahashi, K. Diewert, G.S. (1987)
Self Adaptive Grid Method with Application to Airfoil Flow
AIAA Journal Vol.25 No.4 513-20

Nakahashi, K. Diewert, G.S. (1986)
Three Dimensional Adaptive Grid Method
AIAA Journal Vol.24 No.6 938-54

Nakahashi, K. Diewert, G.S. (1985)
A Practical Adaptive Grid Method for Complex Fluid Flow Problems
Proceeding of the 9th International Conference on Numerical Methods in Fluid Dynamics (Eds. Soubbaramayer and Broujot) 422-6 Springer-Verlag

Niederdrenk, P. (1991)
Solution-Adaptive Grid Generation by Hyperbolic/Parabolized Hyperbolic P.D.E.'s
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 173-84

Noack, R.W. Anderson, D.A. (1990)
Solution Adaptive Grid Generation using Parabolic Partial Differential Equations
AIAA Journal Vol.28 No.6 1016-23

Oden, J.T. Strouboulis, T. Devloo, P. (1986)
Adaptive Finite Element Methods for the Analysis of Inviscid Compressible Flow: Part I. Fast Refinement/Unrefinement and Moving Mesh Methods for Unstructured Meshes
Computer Methods in Applied Mechanics and Engineering Vol.59 327-62

Palacio, A. Malin, M.R. Proumen, N. (1990)
Numerical Computations of Steady Transonic and Supersonic Flow Fields
International Journal of Heat and Mass Transfer Vol.33 No.6 1193-204
Palmerio, B. (1992)
Mesh Adaption for Compressible Flows
Rapports de Recherche #1683 INRIA May 1992
Pao, S.P. Abdol-Hamid, K.S. (1991)
Grid Adaption to Multiple Functions for Applied Aerodynamic Analysis
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 119-29
Pascau, A. Gaspar, F. (1995)
A Multigrid Method for Domains containing Regions with Different Levels of Refinement
9th International Conference on Numerical Methods in Laminar and Turbulent Flow 1657-66
Patankar, S.V. (1980)
Numerical Heat Transfer and Fluid Flow
Hemisphere, New York
Patel, M.K. Bennett, C. Zhao, J. Knight, B. (1995)
Application of Neural Networks for Adaptive Grids
Internal Report
Patel, M.K. Pericleous, K.A. Baldwin, S. (1995)
The Development of a Structured Mesh Grid Adaption Technique for Resolving Shock Discontinuities in Upwind Navier-Stokes Codes
International Journal for Numerical Methods in Fluids Vol.20 1179-97
Patel, M.K. Cross, M. Markatos, N.C. (1988)
An Assessment of Flow Oriented Schemes for reducing False Diffusion
International Journal for Numerical Methods in Engineering Vol.26 2279-2304
Pember, R.B. Bell, J.B. Colella, P. Crutchfield, W.Y. Welcome, M.L. (1995)
An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions
Journal of Computational Physics Vol.120, 278-304
Peraire, J. Peiró, J. Morgan, K. (1992)
Adaptive Remeshing for Three-Dimensional Compressible Flow Computations
Journal of Computational Physics Vol.103 269-85
Petzold, L.R. (1987)
Observations on an Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations
Applied Numerical Mathematics Vol.3 347-60
Quirk, J.J. (1996)
Parallel Adaptive Grid Algorithm for Computational Shock Hydrodynamics
Applied Numerical Mathematics Vol.20 No.4 427-53
Quirk, J.J. (1991)
An Adaptive Grid Algorithm for Computational Shock Hydrodynamics
PhD Thesis, Cranfield Institute of Technology, January 1991
Rai, M.M. Anderson, D.A. (1982)
Application of Adaptive Grids to Fluid-Flow Problems with Asymptotic Solutions
AIAA Journal Vol.20 No.4 496-502

Ramakrishnan, R. Singh, D.H. (1994)
Modeling Scramjet Combustor Flowfields with a Grid Adaption Scheme
AIAA Journal Vol.32 No.5 930-5
Ramakrishnan, R. (1994)
Structured and Unstructured Grid Adaption Schemes for Numerical Modeling of Field Problems
Applied Numerical Mathematics Vol.14 285-310
Rhie, C.M. Chow, W.L. (1983)
Numerical Study of the Turbulent Flow Past and Airfoil with Trailing Edge Separation
AIAA Journal Vol.21 No.11 1525-32
Roache, P.J. Salari, K. Steinberg, S. (1991)
Hybrid Adaptive Poisson Grid Generation and Grid Smoothness
Communications in Applied Numerical Methods Vol.7 345-54
Sallam, S. El-Tarazi, M.N. (1993)
Quadratic Spline Interpolation on Uniform Meshes
Applied Numerical Mathematics Vol.11 419-27
Sanz-Serna, J.M. Christie, I. (1986)
A Simple Adaptive Technique for Nonlinear Wave Problems
Journal of Computational Physics Vol.67 348-60
Saouab, S. Vandromme, D. (1991)
Application of a Variational Method in Compressible Flow Computations
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 557-67
Seibert, W. Fritz, W. Letcher, S. (1989)
On the Way to an Integrated Mesh Generation System for Industrial Applications
AGARD cp464 Paper 14
Seldner, D. Westermann, T. (1988)
Algorithms fo Interpolation and Localisation in Irregular 2D Meshes
Journal of Computational Physics Vol.79 1-11
SenGupta, G. Borland, C.J. Johnson, F.T. Bussoletti, J.E. Melvin, R.G. Young, D.P. Bieterman, M.B. Palotas, P.A. (1991)
Analysis of Unsteady Aerodynamic and Flutter Characteristics of an Aeroelastic Model in Transonic Flow
Transonic Unsteady Aerodynamics and Aeroelasticity AGARD 1991
Shen, C-Y. Reed H.L. (1993)
Shepard's Interpolation for Solution-Adaptive Methods
Journal of Computational Physics Vol.106 52-61
Shirayama, S. (1991)
Simple Diagnosis for the Quality of Generated Grid Systems
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 547-55
Shyy, W. Gingrich, W.K. Gebhart, B. (1992)
Adaptive Grid Solution for Buoyancy-Induced Flow in Vertical Slots
Numerical Heat Transfer, Part A Vol.22 51-70
Shyy, W. (1991a)
Structure of an Adaptive Grid Computational Method from the Viewpoint of Dynamic Chaos
Applied Numerical Mathematics Vol.7 263-85

Shyy, W. (1991b)
Structure of an Adaptive Grid Computational Method from the Viewpoint of Dynamic Chaos, Part II: Grid Addition and Probability Distribution
Applied Numerical Mathematics Vol.7 523-45
Shyy, W. (1988)
Computation of Complex Fluid Flows using an Adaptive Grid Method
International Journal for Numerical methods in Fluids Vol.8 475-89
Shyy, W. (1986)
An Adaptive Grid Method for Navier-Stokes Flow Computation II: Grid Addition
Applied Numerical Mathematics Vol.2 9-19
Shyy, W. Chen, M.H. (1990)
Steady-State Natural Convection with Phase Change
International Journal of Heat and Mass Transfer Vol.33 No.11 2545-63
Singh, D.J. Kumar, A. Tiwari, S.N. (1991)
Numerical Simulation of Shock Impingement on Blunt Cowl Lip in Viscous Hypersonic Flows
Numerical Heat Transfer, Part A Vol.20 No.3 329-44
Slone, A.K. Bailey, C. Pericleous, K. Cross, M. (1997)
Dynamic Fluid-Structure Interactions using Finite Volume Unstructured Mesh Procedures
CEAS International Forum on Aeroelasticity and Structural Dynamics, June 1997 417-24
Soni, B.K. (1991)
Grid Optimization: A Mixed Approach
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. Thompson, J.F.) 617-28
Spalding, D.B. (1989)
The PHOENICS Beginner's Guide TR/100
Cham Ltd, 40 High st, Wimbledon Village, London SW19 5AU
Sweby, P.K. (1988)
An Approximate Equidistribution Technique for Unstructured Grids
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 360-6
Szmelter, J. Marchant, M.J. Evans, A. Weatherill, N.P. (1992)
Two-Dimensional Navier-Stokes Equations with Adaptivity on Structured Meshes
Computer Methods in Applied Mechanics and Engineering Vol.101 355-68
Thompson, C.P. Leaf, G.K. Van Rosendale, J. (1992)
A Dynamically Adaptive Multigrid Algorithm for the Incompressible Navier-Stokes Equations, Validation and Model Problems
Applied Numerical Mathematics Vol.9 511-32
Thompson, J.F. Warsi, Z.U.A. Mastin, C.W. (1985)
Numerical Grid Generation - Foundations and Applications
North-Holland, Amsterdam
Thompson, J.F. (1987)
A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure
Computer Methods in Applied Mechanics and Engineering Vol.64 377-411

Thompson, J.F. (1985)
A Survey of Dynamically-Adaptive Grids in the Numerical Solution of Partial Differential Equations
Applied Numerical Mathematics I 3-27
Thompson, J.F. (1983)
A Survey of Grid Generation Techniques in Computational Fluid Dynamics
AIAA Paper 83-0447
Thompson, M.C. Ferziger, J.H. (1989)
An Adaptive Multigrid Technique for the Incompressible Navier-Stokes Equations
Journal of Computational Physics Vol.82 94-121
Trompert, R.A. Verwer, J.G. (1991)
A Static-Regridding Method for Two-Dimensional Parabolic Partial Differential Equations
Applied Numerical Mathematics Vol.8 65-90
Tu, Y. Thompson, J.F. (1990)
Three-Dimensional Solution-Adaptive Grid Generation on Composite Configurations
AIAA Paper 90-0329
Tzanos, C.P. (1989)
A Method of Adaptive Nodes for Convective Heat Transfer Problems
Numerical Heat Transfer, Part B Vol.15 153-69
Van Der Maarel, H.T.M. (1992)
Adaptive Multigrid for the Steady Euler Equations
Communications in Applied Numerical Methods Vol.8 749-60
Verwer, J.G. Trompert, R.A. (1993)
Analysyis of local uniform grid refinement
Applied Numerical Mathematics Vol.13 251-70
Warren, G.P. (1992)
Application of Multigrid and Adaptive Grid Embedding to the Two-Dimensional Flux-Split Euler Equations
Communications in Applied Numerical Methods Vol.8 771-84
Warsi, Z.U.A. Thompson, J.F. (1990)
Application of Variational Methods in the Fixed and Adaptive Grid Generation
Computers and Mathematics with Applications Vol.19 No.8/9 31-41
Weatherill, N.P. Marchant, M.J. (1994)
Grid Adaption using a Distribution of Sources applied to Inviscid Compressible Flow Simulations
International Journal for Numerical methods in Fluids Vol.19 739-64
Weatherill, N.P. Soni, B.K. (1991)
Grid Adaption and Refinement in Structured and Unstructured Algorithms
Numerical Grid Generation in Computational Fluid Dynamics and Related Fields (Eds Arcilla A.S. Hauser, J. Eiseman, P.R. THompson, J.F.) 143-57
Weatherill, N.P. (1988)
A Strategy for the use of Hybrid Structured-Unstructured Meshes in Computational Fluid Dynamics
Conference on Numerical Methods for Fluid Dynamics (Eds Morton, Baines) 101-16

Wu, J. (1995)
Local Refinement Strategy and Criteria for Solving Incompressible Navier-Stokes Equations on Parallel Computers
9th International Conference on Numerical Methods in Laminar and Turbulent Flow 445-56

Xu, X.Y. Griffith, T.M. Collins, M.W. Jones, C.J.H. Tardy, Y. (1993)
Coupled Modelling of Blood Flow and Arterial Wall Interactions by the Finite Elememt Method
Computers in Cardiology 1993 687-90

Yasar, O. Moses, G.A. (1992)
Explicit Adaptive-Grid Radiation-Magnetohydrodynamics
Journal of Computational Physics Vol.100 38-49

Zegeling, P.A. Blom, J.G. (1992)
A Note on the Grid Movement induced by MFE
International Journal for Numerical methods in Engineering Vol.35 623-36

Zhang, H. Moallemi, M.K. (1995)
MAGG - A Multizone Adaptive Grid-Generation Technique for Simulation of Moving and Free Boundary Problems
Numerical Heat Transfer, Part B Vol.27 255-276

Zhao, J. Patel, M.K., Knight, B. Bennett, C.R. (1996)
Self-organizing Neural Network for Optimal Grid Distribution
University of Greenwich Paper no.96/IM/12

APPENDIX A: THE LPE EQUATION IN ONE AND THREE DIMENSIONS

A.1 Introduction

The purpose of this appendix is to expand the discretisation of the LPE equation from two dimensions, as presented in chapter 2, into one and three dimensions. This appendix also shows part of the derivation of the metric tensors that define the transformation from physical to curvilinear space.

A.2 Calculation of Metric Tensors

The contravariant metric tensors are defined as the dot products of the contravariant base vectors $\nabla\xi_{i,i=1,3}$.

$$g^{ij} = \nabla\xi_i \cdot \nabla\xi_j \qquad \textbf{(A.1)}$$

The contravariant base vector $\nabla\xi$ is defined as being the normal to the surface where ξ is constant.

The contravariant base vectors can be calculated using the relationship

$$\nabla\xi_i = \frac{1}{\sqrt{g}} \underline{r}_{\xi_j} \times \underline{r}_{\xi_k} \qquad i,j,k \text{ cyclic} \qquad \textbf{(A.2)}$$

where $\underline{r}_{\xi_i}$ is the gradient of the line ξ_i = constant.

$$\underline{r}_{\xi_i} = \frac{\partial x}{\partial \xi_i}\underline{i} + \frac{\partial y}{\partial \xi_i}\underline{j} + \frac{\partial z}{\partial \xi_i}\underline{k} \qquad \textbf{(A.3)}$$

Using the vector identity

$$(\underline{A}\times\underline{B})\cdot(\underline{C}\times\underline{D}) = (\underline{A}\cdot\underline{C})(\underline{B}\cdot\underline{D}) - (\underline{A}\cdot\underline{D})(\underline{B}\cdot\underline{C}) \qquad \textbf{(A.4)}$$

The metric tensors g^{ij} are (noting $g^{ij} = g^{ji}$).

$$g^{11} = \nabla\xi_1 \cdot \nabla\xi_1$$

$$= \frac{1}{g}(\underline{r}_{\xi_2} \times \underline{r}_{\xi_3})^2 \qquad \textbf{(A.5)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3})^2\right)$$

$$g^{22} = \nabla\xi_2 \cdot \nabla\xi_2$$

$$= \frac{1}{g}(\underline{r}_{\xi_1} \times \underline{r}_{\xi_3})^2 \qquad \textbf{(A.6)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})^2\right)$$

$$g^{33} = \nabla\xi_3 \cdot \nabla\xi_3$$

$$= \frac{1}{g}(\underline{r}_{\xi_1} \times \underline{r}_{\xi_2})^2 \qquad \textbf{(A.7)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})^2\right)$$

$$g^{12} = \nabla\xi_1 \cdot \nabla\xi_2$$

$$= \frac{1}{g}(\underline{r}_{\xi_2} \times \underline{r}_{\xi_3})(\underline{r}_{\xi_3} \times \underline{r}_{\xi_1}) \qquad \textbf{(A.8)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3})\right)$$

$$g^{13} = \nabla\xi_1 \cdot \nabla\xi_3$$

$$= \frac{1}{g}(\underline{r}_{\xi_2} \times \underline{r}_{\xi_3})(\underline{r}_{\xi_1} \times \underline{r}_{\xi_2}) \qquad \textbf{(A.9)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})\right)$$

$$g^{23} = \nabla\xi_2 \cdot \nabla\xi_3$$

$$= \frac{1}{g}(\underline{r}_{\xi_3} \times \underline{r}_{\xi_1})(\underline{r}_{\xi_1} \times \underline{r}_{\xi_2}) \qquad \textbf{(A.10)}$$

$$= \frac{1}{g}\left((\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3})\right)$$

The jacobian g cancels out in the full equations.

A full derivation of the metric tensors is given in Thompson et al. (1985).

The six vector dot products are evaluated using central difference formula as follows

$$\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1} = \left(\underline{i}\frac{\partial x}{\partial \xi_1} + \underline{j}\frac{\partial y}{\partial \xi_1} + \underline{k}\frac{\partial z}{\partial \xi_1}\right) \cdot \left(\underline{i}\frac{\partial x}{\partial \xi_1} + \underline{j}\frac{\partial y}{\partial \xi_1} + \underline{k}\frac{\partial z}{\partial \xi_1}\right)$$

$$= \left(\frac{\partial x}{\partial \xi_1}\right)^2 + \left(\frac{\partial y}{\partial \xi_1}\right)^2 + \left(\frac{\partial z}{\partial \xi_1}\right)^2 \qquad \textbf{(A.11)}$$

$$= \frac{1}{4}\left((x_{i+1,j,k} - x_{i-1,j,k})^2 + (y_{i+1,j,k} - y_{i-1,j,k})^2 + (z_{i+1,j,k} - z_{i-1,j,k})^2\right)$$

$$\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2} = \left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2} + \underline{k}\frac{\partial z}{\partial \xi_2}\right) \cdot \left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2} + \underline{k}\frac{\partial z}{\partial \xi_2}\right)$$

$$= \left(\frac{\partial x}{\partial \xi_2}\right)^2 + \left(\frac{\partial y}{\partial \xi_2}\right)^2 + \left(\frac{\partial z}{\partial \xi_2}\right)^2 \qquad \textbf{(A.12)}$$

$$= \frac{1}{4}\left((x_{i,j+1,k} - x_{i,j-1,k})^2 + (y_{i,j+1,k} - y_{i,j-1,k})^2 + (z_{i,j+1,k} - z_{i,j-1,k})^2\right)$$

$$\underline{r}_{\xi_3}\cdot\underline{r}_{\xi_3}=\left(\underline{i}\frac{\partial x}{\partial \xi_3}+\underline{j}\frac{\partial y}{\partial \xi_3}+\underline{k}\frac{\partial z}{\partial \xi_3}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_3}+\underline{j}\frac{\partial y}{\partial \xi_3}+\underline{k}\frac{\partial z}{\partial \xi_3}\right)$$

$$=\left(\frac{\partial x}{\partial \xi_3}\right)^2+\left(\frac{\partial y}{\partial \xi_3}\right)^2+\left(\frac{\partial z}{\partial \xi_3}\right)^2 \qquad \textbf{(A.13)}$$

$$=\frac{1}{4}\left((x_{i,j,k+1}-x_{i,j,k-1})^2+(y_{i,j,k+1}-y_{i,j,k-1})^2+(z_{i,j,k+1}-z_{i,j,k-1})^2\right)$$

$$\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_2}=\left(\underline{i}\frac{\partial x}{\partial \xi_1}+\underline{j}\frac{\partial y}{\partial \xi_1}+\underline{k}\frac{\partial z}{\partial \xi_1}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_2}+\underline{j}\frac{\partial y}{\partial \xi_2}+\underline{k}\frac{\partial z}{\partial \xi_2}\right)$$

$$=\frac{\partial x}{\partial \xi_1}\frac{\partial x}{\partial \xi_2}+\frac{\partial y}{\partial \xi_1}\frac{\partial y}{\partial \xi_2}+\frac{\partial z}{\partial \xi_1}\frac{\partial z}{\partial \xi_2} \qquad \textbf{(A.14)}$$

$$=\frac{1}{4}\Big((x_{i+1,j,k}-x_{i-1,j,k})(x_{i,j+1,k}-x_{i,j-1,k})+(y_{i+1,j,k}-y_{i-1,j,k})(y_{i,j+1,k}-y_{i,j-1,k})$$

$$+(z_{i+1,j,k}-z_{i-1,j,k})(z_{i,j+1,k}-z_{i,j-1,k})\Big)$$

$$\underline{r}_{\xi_1}\cdot\underline{r}_{\xi_3}=\left(\underline{i}\frac{\partial x}{\partial \xi_1}+\underline{j}\frac{\partial y}{\partial \xi_1}+\underline{k}\frac{\partial z}{\partial \xi_1}\right)\cdot\left(\underline{i}\frac{\partial x}{\partial \xi_3}+\underline{j}\frac{\partial y}{\partial \xi_3}+\underline{k}\frac{\partial z}{\partial \xi_3}\right)$$

$$=\frac{\partial x}{\partial \xi_1}\frac{\partial x}{\partial \xi_3}+\frac{\partial y}{\partial \xi_1}\frac{\partial y}{\partial \xi_3}+\frac{\partial z}{\partial \xi_1}\frac{\partial z}{\partial \xi_3} \qquad \textbf{(A.15)}$$

$$=\frac{1}{4}\Big((x_{i+1,j,k}-x_{i-1,j,k})(x_{i,j,k+1}-x_{i,j,k-1})+(y_{i+1,j,k}-y_{i-1,j,k})(y_{i,j,k+1}-y_{i,j,k-1})$$

$$+(z_{i+1,j,k}-z_{i-1,j,k})(z_{i,j,k+1}-z_{i,j,k-1})\Big)$$

$$\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3} = \left(\underline{i}\frac{\partial x}{\partial \xi_2} + \underline{j}\frac{\partial y}{\partial \xi_2} + \underline{k}\frac{\partial z}{\partial \xi_2} \right) \cdot \left(\underline{i}\frac{\partial x}{\partial \xi_3} + \underline{j}\frac{\partial y}{\partial \xi_3} + \underline{k}\frac{\partial z}{\partial \xi_3} \right)$$

$$= \frac{\partial x}{\partial \xi_2}\frac{\partial x}{\partial \xi_3} + \frac{\partial y}{\partial \xi_2}\frac{\partial y}{\partial \xi_3} + \frac{\partial z}{\partial \xi_2}\frac{\partial z}{\partial \xi_3} \qquad \textbf{(A.16)}$$

$$= \frac{1}{4}\Big((x_{i,j+1,k} - x_{i,j-1,k})(x_{i,j,k+1} - x_{i,j,k-1}) + (y_{i,j+1,k} - y_{i,j-1,k})(y_{i,j,k+1} - y_{i,j,k-1}) + (z_{i,j+1,k} - z_{i,j-1,k})(z_{i,j,k+1} - z_{i,j,k-1}) \Big)$$

Using the terminology **E,W** for **i+1,i-1**; **N,S** for **j+1,j-1**; and **H,L** for **k+1,k-1** gives

$$\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1} = \frac{1}{4}\Big((x_E - x_W)^2 + (y_E - y_W)^2 + (z_E - z_W)^2 \Big) \qquad \textbf{(A.17)}$$

$$\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2} = \frac{1}{4}\Big((x_N - x_S)^2 + (y_N - y_S)^2 + (z_N - z_S)^2 \Big) \qquad \textbf{(A.18)}$$

$$\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3} = \frac{1}{4}\Big((x_H - x_L)^2 + (y_H - y_L)^2 + (z_H - z_L)^2 \Big) \qquad \textbf{(A.19)}$$

$$\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2} = \frac{1}{4}\Big((x_E - x_W)(x_N - x_S) + (y_E - y_W)(y_N - y_S) + (z_E - z_W)(z_N - z_S) \Big) \qquad \textbf{(A.20)}$$

$$\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3} = \frac{1}{4}\Big((x_E - x_W)(x_H - x_L) + (y_E - y_W)(y_H - y_L) + (z_E - z_W)(z_H - z_L) \Big) \qquad \textbf{(A.21)}$$

$$\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3} = \frac{1}{4}\Big((x_N - x_S)(x_H - x_L) + (y_N - y_S)(y_H - y_L) + (z_N - z_S)(z_H - z_L) \Big) \qquad \textbf{(A.22)}$$

The tensors as used in the discretisation of the main equations then become

$$g^{11} = (\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3})^2 \qquad \textbf{(A.23)}$$

$$g^{22} = (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})^2 \qquad \textbf{(A.24)}$$

$$g^{33} = (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})^2 \qquad \textbf{(A.25)}$$

$$g^{12} = (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3}) \qquad \textbf{(A.26)}$$

$$g^{13} = (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2}) \qquad \textbf{(A.27)}$$

$$g^{23} = (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_2})(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_3}) - (\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_3}) \qquad \textbf{(A.28)}$$

A.3 Derivation of Laplace Term

The Laplace system in three dimensions is (equation 2.15)

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{\xi_i \xi_j} = 0 \qquad \textbf{(A.29)}$$

Discretised, using central difference formula (equations 2.1, 2.2 and 2.3), this becomes

$$\begin{aligned} & g^{11}(r_E - 2r_P + r_W) + g^{22}(r_N - 2r_P + r_S) + g^{33}(r_H - 2r_P + r_L) \\ & + \frac{g^{12}}{2}\left[(r_{EN} - r_{WN}) - (r_{ES} - r_{WS})\right] + \frac{g^{13}}{2}\left[(r_{EH} - r_{EL}) - (r_{WH} - r_{WL})\right] \\ & + \frac{g^{23}}{2}\left[(r_{NH} - r_{SH}) - (r_{NL} - r_{SL})\right] = 0 \end{aligned} \qquad \textbf{(A.30)}$$

Separating the terms in equation A.30 leads to the algebraic equation

$$a_P^L r_P = a_E^L r_E + a_W^L r_W + a_N^L r_N + a_S^L r_S + a_H^L r_H + a_L^L r_L + s^L \qquad \textbf{(A.31)}$$

The source term s^L is

$$s^L = a_{NH}^L r_{NH} + a_{WH}^L r_{WH} + a_{EH}^L r_{EH} + a_{SH}^L r_{SH} + a_{NW}^L r_{NW} + a_{NE}^L r_{NE} + a_{SW}^L r_{SW} + a_{SE}^L r_{SE} + a_{NL}^L r_{NL} + a_{WL}^L r_{WL} + a_{EL}^L r_{EL} + a_{SL}^L r_{SL} \quad \textbf{(A.32)}$$

where

$$a_P^L = a_E^L + a_W^L + a_N^L + a_S^L + a_H^L + a_L^L \quad \textbf{(A.33)}$$

$$a_E^L = a_W^L = g^{11} \quad \textbf{(A.34)}$$

$$a_N^L = a_S^L = g^{22} \quad \textbf{(A.35)}$$

$$a_H^L = a_L^L = g^{33} \quad \textbf{(A.36)}$$

$$a_{NE}^L = a_{SW}^L = \frac{g^{12}}{2} \quad \textbf{(A.37)}$$

$$a_{NW}^L = a_{SE}^L = -\frac{g^{12}}{2} \quad \textbf{(A.38)}$$

$$a_{EH}^L = a_{WL}^L = \frac{g^{13}}{2} \quad \textbf{(A.39)}$$

$$a_{EL}^L = a_{WH}^L = -\frac{g^{13}}{2} \quad \textbf{(A.40)}$$

$$a_{NH}^L = a_{SL}^L = \frac{g^{23}}{2} \quad \textbf{(A.41)}$$

$$a_{NL}^{L} = a_{SH}^{L} = -\frac{g^{23}}{2} \tag{A.42}$$

A.4 Derivation of Poisson Term

The Poisson equation in three dimensions is (equation 2.27)

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{\xi_i \xi_j} + \sum_{k=1}^{3} g^{kk} P_k r_{\xi_k} = 0 \tag{A.43}$$

The first term in the equation is identical to the Laplace equation and breaks down in the same way.

The finite difference approximation to the second term of the Poisson equation is

$$\sum_{k=1}^{3} g^{kk} P_k r_{\xi_k} \rightarrow g^{11}\frac{(r_E - r_W)}{2} P_1 + g^{22}\frac{(r_N - r_S)}{2} P_2 + g^{33}\frac{(r_H - r_L)}{2} P_3 \tag{A.44}$$

Separating the terms in equations A.30 and A.44 leads to the algebraic equation

$$a_P^P r_P = a_E^P r_E + a_W^P r_W + a_N^P r_N + a_S^P r_S + a_H^P r_H + a_L^P r_L + s^P \tag{A.45}$$

where

$$a^P \equiv a^L \tag{A.46}$$

$$s^P = s^L + g^{11} P_1 \frac{(r_E - r_W)}{2} + g^{22} P_2 \frac{(r_N - r_S)}{2} + g^{33} P_3 \frac{(r_H - r_L)}{2} \tag{A.47}$$

A.5 Derivation of Equidistribution Term

The equidistribution equation as used in the LPE system is

$$\sum_{i=1}^{3} g^{ii} r_{\xi_i} \left(\frac{W^i{}_{\xi_i}}{W^i} + \frac{x_{\xi_i} x_{\xi_i\xi_i} + y_{\xi_i} y_{\xi_i\xi_i} + z_{\xi_i} z_{\xi_i\xi_i}}{x_{\xi_i}{}^2 + y_{\xi_i}{}^2 + z_{\xi_i}{}^2} \right) = 0 \qquad \textbf{(A.48)}$$

The discretisation of the equidistribution equation is simplified by first seeing that the denominator of the second term in the brackets is identical to vector dot products used to evaluate the metric tensors, which is calculated elsewhere and can be substituted in A.48.

$$\underline{r}_{\xi_i} \cdot \underline{r}_{\xi_i} \equiv x_{\xi_i}^2 + y_{\xi_i}^2 + z_{\xi_i}^2 \qquad \textbf{(A.49)}$$

Discretised, equation A.48 gives

$$g^{11} \frac{(r_E - r_W)}{4} \left[\left(\frac{WI_E - WI_W}{WI_P} \right) + \frac{1}{(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})} \left((x_E - x_W)(x_E - 2x_P + x_W) \right. \right. \qquad \textbf{(A.50)}$$

$$\left. \left. + (y_E - y_W)(y_E - 2y_P + y_W) + (z_E - z_W)(z_E - 2z_P + z_W) \right) \right]$$

$$+ g^{22} \frac{(r_N - r_S)}{4} \left[\left(\frac{WJ_N - WJ_S}{WJ_P} \right) + \frac{1}{(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})} \left((x_N - x_S)(x_N - 2x_P + x_S) \right. \right.$$

$$\left. \left. + (y_N - y_S)(y_N - 2y_P + y_S) + (z_N - z_S)(z_N - 2z_P + z_S) \right) \right]$$

$$+ g^{33} \frac{(r_H - r_L)}{4} \left[\left(\frac{WK_H - WK_L}{WK_P} \right) + \frac{1}{(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3})} \left((x_H - x_L)(x_H - 2x_P + x_L) \right. \right.$$

$$\left. \left. + (y_H - y_L)(y_H - 2y_P + y_L) + (z_H - z_L)(z_H - 2z_P + z_L) \right) \right] = 0$$

Where r is the variable being solved for.

Separating the terms in equation A.50 leads to the algebraic equation

$$a_P^E r_P = a_E^E r_E + a_W^E r_W + a_N^E r_N + a_S^E r_S + a_H^E r_H + a_L^E r_L + s^E \tag{A.51}$$

where

$$a_P^E = a_E^E + a_W^E + a_N^E + a_S^E + a_H^E + a_L^E \tag{A.52}$$

$$a_E^E = a_W^E = \frac{g^{11}}{4} \frac{(r_E - r_W)^2}{(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})} \tag{A.53}$$

$$a_N^E = a_S^E = \frac{g^{22}}{4} \frac{(r_N - r_S)^2}{(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})} \tag{A.54}$$

$$a_H^E = a_L^E = \frac{g^{33}}{4} \frac{(r_H - r_L)^2}{(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3})} \tag{A.55}$$

$$\begin{aligned} s^E = g^{11} \frac{(r_E - r_W)}{4} \Bigg[\Bigg(\frac{WI_E - WI_W}{WI_P} \Bigg) \\ + \frac{1}{(\underline{r}_{\xi_1} \cdot \underline{r}_{\xi_1})} \big((p_E - p_W)(p_E - 2p_P + p_W) + (q_E - q_W)(q_E - 2q_P + q_W) \big) \Bigg] \\ + g^{22} \frac{(r_N - r_S)}{4} \Bigg[\Bigg(\frac{WJ_N - WJ_S}{WJ_P} \Bigg) \\ + \frac{1}{(\underline{r}_{\xi_2} \cdot \underline{r}_{\xi_2})} \big((p_N - p_S)(p_N - 2p_P + p_S) + (q_N - q_S)(q_N - 2q_P + q_S) \big) \Bigg] \\ + g^{33} \frac{(r_H - r_L)}{4} \Bigg[\Bigg(\frac{WK_H - WK_L}{WK_P} \Bigg) \\ + \frac{1}{(\underline{r}_{\xi_3} \cdot \underline{r}_{\xi_3})} \big((p_H - p_L)(p_H - 2p_P + p_L) + (q_H - q_L)(q_H - 2q_P + q_L) \big) \Bigg] \end{aligned} \tag{A.56}$$

A.6 The LPE Equation

The discretised LPE equation is represented by the combination of the algebriac systems given for each of the three terms in equations A.31, A.45 and A.51

$$a_P^{LPE} r_P = a_E^{LPE} r_E + a_W^{LPE} r_W + a_N^{LPE} r_N + a_S^{LPE} r_S + a_H^{LPE} r_H + a_L^{LPE} r_L + s^{LPE} \quad \textbf{(A.57)}$$

Where, with the terms modified by a user controlled weight λ^L, λ^P, λ^E for the Laplace, Poisson and Equidistribution terms respectively,

$$a_P^{LPE} = a_E^{LPE} + a_W^{LPE} + a_N^{LPE} + a_S^{LPE} + a_H^{LPE} + a_L^{LPE} \quad \textbf{(A.58)}$$

$$a_E^{LPE} = a_W^{LPE} = (\lambda^L + \lambda^P) a_E^L + \lambda^E a_E^E \quad \textbf{(A.59)}$$

$$a_N^{LPE} = a_S^{LPE} = (\lambda^L + \lambda^P) a_N^L + \lambda^E a_N^E \quad \textbf{(A.60)}$$

$$a_H^{LPE} = a_L^{LPE} = (\lambda^L + \lambda^P) a_H^L + \lambda^E a_H^E \quad \textbf{(A.61)}$$

$$s^{LPE} = \lambda^L s^L + \lambda^P s^P + \lambda^E s^E \quad \textbf{(A.62)}$$

A.7 The LPE Equation in One Dimension

For the one dimensional system ξ_1 the metric tensor disappears as here is a one to one correspondence between physical and curvilinear space.

All other components are set or forced to zero. There are no significant dot products, and the metric tensor goes to 1, as the jacobian g cancels out.

The full form of the LPE equation in one dimension is

$$a_P^{LPE} r_P = a_E^{LPE} r_E + a_W^{LPE} r_W + s^{LPE} \quad \textbf{(A.63)}$$

where

$$a_P^{LPE} = a_E^{LPE} + a_W^{LPE} \tag{A.64}$$

$$a_E^{LPE} = a_W^{LPE} = a_E^L + a_E^P + a_E^E \tag{A.65}$$

$$s^{LPE} = s^L + s^P + s^E \tag{A.66}$$

where the Laplace terms are

$$a_E^L = a_W^L = \lambda^L \tag{A.67}$$

$$s^L = 0 \tag{A.68}$$

The Poisson terms are

$$a_E^P = a_W^P = \lambda^P \tag{A.69}$$

$$S^P = \lambda^P P_1 \frac{(x_E - x_W)}{2} \tag{A.70}$$

And the Equidistribution terms are

$$a_E^E = a_W^E = \lambda^E \tag{A.71}$$

$$s^E = \lambda^E \frac{(x_E - x_W)}{4} \left(\frac{WI_E - WI_W}{WI_P} \right) \tag{A.72}$$

Giving

$$a_E^{LPE} = a_W^{LPE} = \lambda^L + \lambda^P + \lambda^E \tag{A.73}$$

$$s^{LPE} = \lambda^{P} P_1 \frac{(x_E - x_W)}{2} + \lambda^{E} \frac{(x_E - x_W)}{4} \left(\frac{WI_E - WI_W}{WI_P} \right) \qquad \textbf{(A.74)}$$

APPENDIX B: SOFTWARE STRUCTURE

B.1 Introduction

This Appendix is divided up into sections covering adaption files, the adaption initialisation module S_ADAPT, the adaption calculation module G_ADAPT, and the PHOENICS file structure which influences their development. The following adaption components are all discussed in relation to their implementation within PHOENICS.

B.2 PHOENICS Components

The structure of PHOENICS is shown in figure B.1.

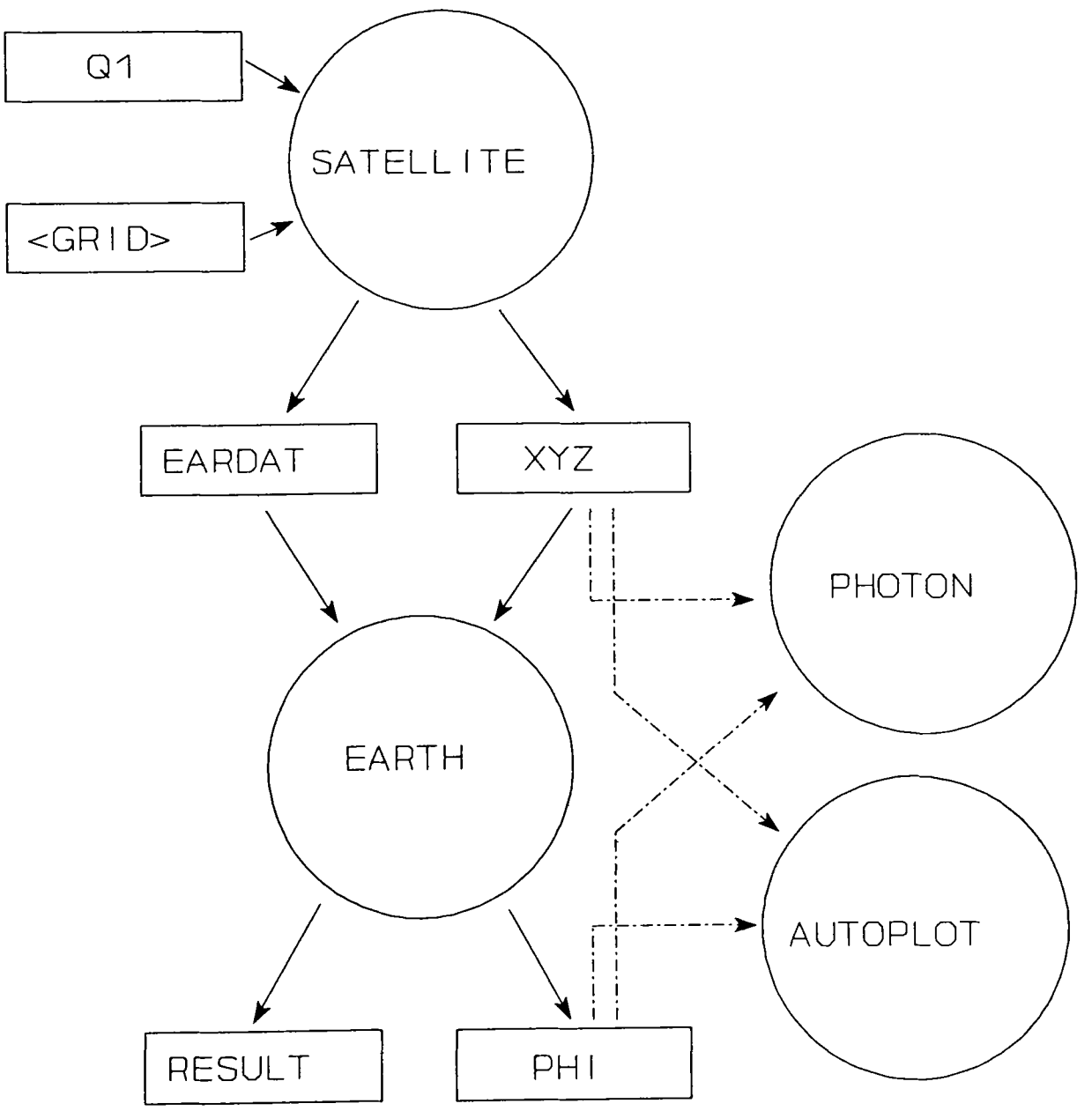


Figure B.1 Main Programs and Data Files within Phoenics

The main programs of PHOENICS include:-

SATELLITE which is the pre processor that interprets the user controlled file Q1 and sets up the data files EARDAT and XYZ which define the problem within EARTH.

EARTH is the solution processor. It contains a range of iterative solvers to resolve the problem defined in the files EARDAT and XYZ and produces the results files RESULT and PHI

PHOTON is an interactive post processor which uses the data within PHI and XYZ to produce a graphical representation of the solution.

AUTOPLOT is an interactive post processor which uses data from the PHOENICS solution as well as tables to produce and compare line plots.

The main files designed to alow user control and pass information between the different programs of PHOENICS include:-

Q1 contains the definition of the case to be solved. The problem is defined using the PHOENICS input language PIL. It is possible to define the grid within the Q1 file but an outside grid may be read in instead by specifying a grid file within Q1 (**<GRID>**).

XYZ contains the location of the grid points that define the problem in a defined format. This file is only used for BFC problems. In other cases the grid data are passed to EARTH with EARDAT. EARTH only reads the file named XYZ but the same file with different names can be produced.

EARDAT contains all the data that define the problem in a form readable by the processor EARTH.

PHI is a result file containing the values within the grid cells of all variables solved for. Its two main uses are firstly to allow the solution to be visualised using the auxiliary PHOENICS programs PHOTON and AUTOPLOT and secondly to allow the problem to be restarted from previous results. Simulations run in EARTH can also be restarted using a PHI file.

RESULT is a line printer type results file that contains a range of information including all initial settings, selected cell values of variables, information on convergence, timing and simple ascii plots.

B.3 Adaption Files

A range of adaption data files exist alongside the two adaption modules S_ADAPT and G_ADAPT. These files are used to control the behaviour of the algorithm, pass data between the two modules, and provide the user with information on its performance.

Summaries of the file interaction within the modified SATELLITE and EARTH are given in figures B.2 and B.3 respectively.

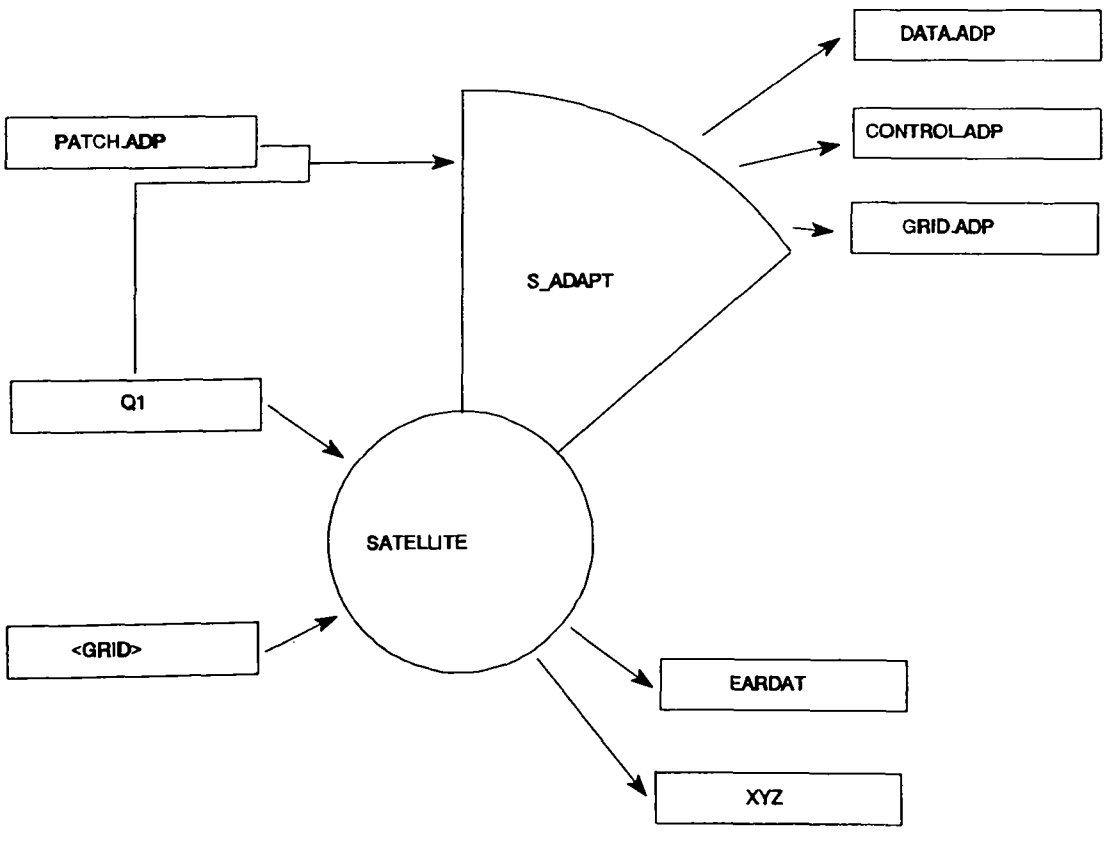


Figure B.2 Main Adaption Files within PHOENICS Satellite

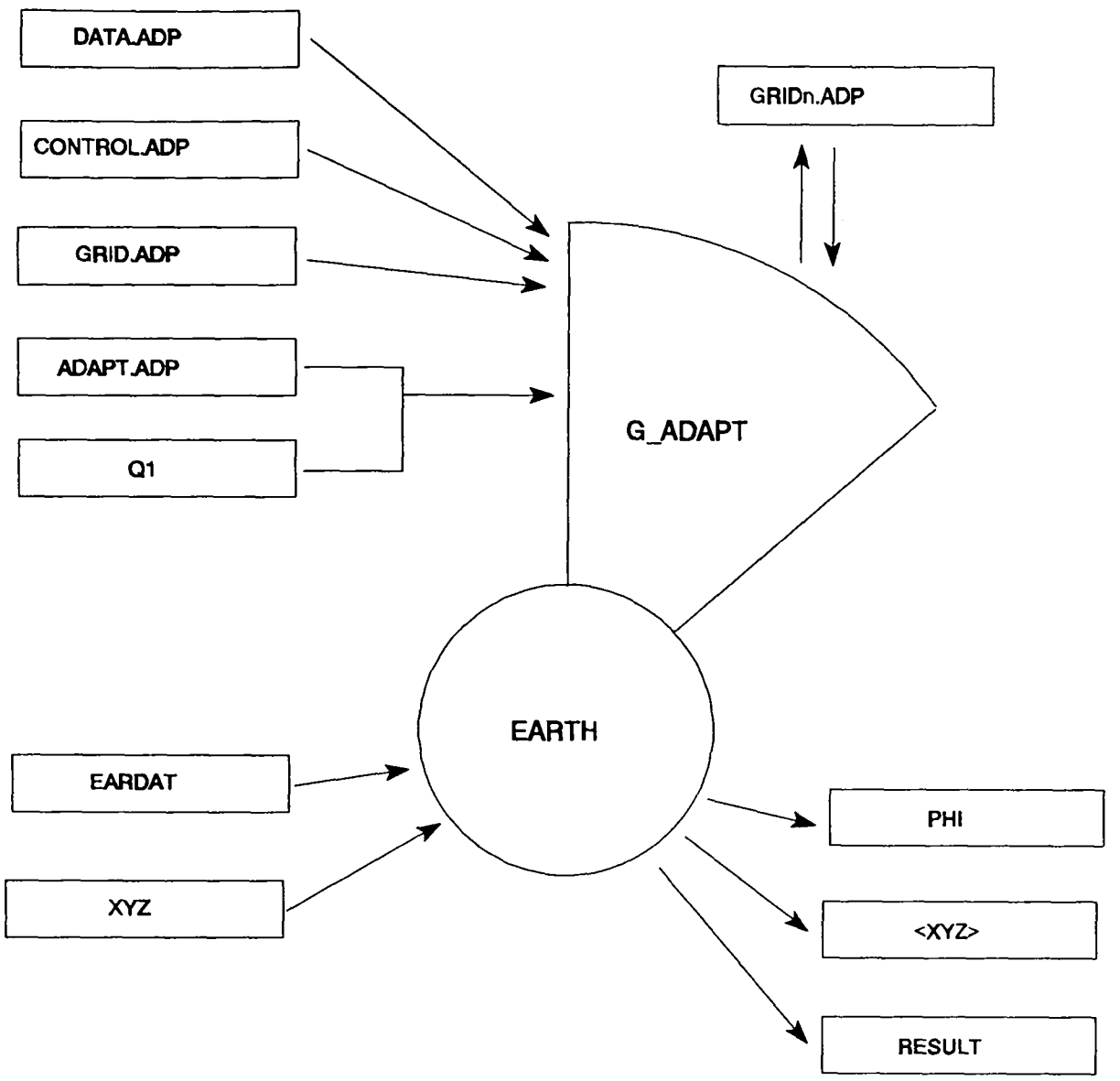


Figure B.3 Main Adaption Files within PHOENICS Earth

The principal user input files are:-

PATCH.ADP contains the physical definition of the adaption regions, surfaces, curves and blanks in the form of named patches.

ADAPT.ADP contains the weights for the LPE equation, the choice of the weight function and the frequency of adaption within the defined adaption regions.

The files transferring data between the two adaption modules are:-

DATA.ADP contains and links the adaption regions, surfaces, curves and blanks and all the data required to fit those surfaces and curves for use in the adaption ground.

CONTROL.ADP contains the control functions required by a poisson grid generator to recreate the grids covered by the adaption regions.

GRIDn.ADP contains the latest grids covered by the adaption regions. A new, numbered, grid file is dumped at each call from PHOENICS EARTH to the adaption module G_ADAPT. The grid data used for adaption are always read from the latest grid file GRIDn.ADP. The first grid file is GRID.ADP.

B.4 S_ADAPT

A single argument in the Q1 file allows access to S_ADAPT. the Q1 file is also used to pass an argument through to EARTH to allow access to G_ADAPT in EARDAT.

The only other link within S_ADAPT to PHOENICS is to the grid data, which is loaded into local arrays within S_ADAPT for processing each adaption region. Though this duplicates information held within the adaptive satellite it allows simpler and faster access within the adaptive routines and reduces dependence on PHOENICS.

The Q1 file can also be used to hold the adaption input files. In PHOENICS version 2.0 and later the Q1 file is used to hold macro files to display the results of the problem within the visualiser PHOTON by including the text as comments in a section marked at its beginning and end by key words. Any text in the Q1 file that begins in the third column or later is treated as a comment. In the same manner the adaption input files PATCH.ADP and ADAPT.ADP can be included in the Q1 file. During the execution of SATELLITE the Q1 file is read and all of its commands are held in a stack within memory. Though the Q1 file may remain open it can be accessed by the adaptive routines without disrupting SATELLITE. Within S_ADAPT it is scanned for a comment beginning with an exclamation mark followed by the key word PATCH.

The adaption set up information follows. If the keyword is not found then the external file PATCH.ADP will be opened instead.

B.4.1 Tasks in S_ADAPT

The main tasks carried out within the adaption code, S_ADAPT, lying within the modified satellite are:-

- Input of major physical features of adaption, though not equation choice and constants as these may be changed more frequently. The physical features are the regions within the grid that will be adapted, curves and surfaces within those regions that must be maintained, and blocks within which the grid is not allowed to move to account for complex geometry, described here as 'blanks'. All features are named, and the first three letters of each name define the feature.

- Checking of regions defined within the adaption input file PATCH.ADP against each other and the main boundaries of the problem for overlaps. Overlaps with other regions are flagged but otherwise ignored. Overlaps with main boundaries are flagged and the regions deleted.

- Linking of surfaces, curves, and blanks to specific regions and checking of overlaps with other patches of the same type. Note that patches which extend outside regions are ignored.

- Division of curves and surfaces into sub curves and sub surfaces around important geometrical features (see section 3.2.3) using user defined criteria. These criteria are more fully explained in the previous chapter but include the options of whether to cut the surfaces or curves at discontinuities and/or turning points and at what tolerances of discontinuities or turning points they are cut.

• Division of sub curves and sub surfaces into patches, which are given implicitly by the cell dimensions which they cover, and the modelling of those patches by the use of cubic splines in the case of curves and bicubic patches in the case of surfaces to allow their recovery after the grid points which initially define them are allowed to move (see sections 3.2.1.1 and 3.2.2.1)

• Output of the adaption data file DATA.ADP.

• Calculation of the control functions required to generate the original grid using a Poisson grid generator to enable the Poisson term in the LPE equation to be used. This information is output to the file CONTROL.ADP.

• Output to the file GRID.ADP of the parts of the grid which define the adaptive regions. This is done to firstly reduce the dependence of the adaption routines on the underlying code, PHOENICS, so that the main adaption routine is always reading the grid from independent files then putting the results into PHOENICS, and secondly to keep a record of the grids in the regions after each time that they are called. Having the grid files in a format private to the adaption routines allows fast and easy access to the grid coordinates.

• Output of the adaption information files ADAPT.USE and SATLOG.ADP.

• Output of the adaption control file PLOT.ADP.

B.5 G_ADAPT

The role of the adaption module G_ADAPT is to modify selected adaption regions in the main grid to improve the accuracy of the solution by using the solution at controlled points during the run of the solver.

G_ADAPT is split into three parts, a preliminary phase, involving the processing of adaption input files, a processing phase, in which the main algorithm is run, and a

bookkeeping phase, where extra adaption files are deleted and the total movement of the grid calculated.

The adaption routines are first activated at sweep number two to run the preliminary phase. Sweep two is chosen instead of sweep one because if the case is being restarted from results files dumped by a previous run then the default first sweep number is two. Other activations are controlled using parameters in the adaption input file ADAPT.ADP. The bookkeeping phase is activated when the last sweep of PHOENICS takes place.

The adaption module runs over each defined adaption region in turn. Each region is treated as an individual problem with no link to the remainder of the regions. This allows the storage requirements of the adaption module to be limited to the size of the largest defined region rather than the whole simulation. Many internal iterations of the adaptive algorithm are possible for each region and can be used to increase the movement of the grid at each call to the adaptive routines. This also allows the use of relaxation to improve the performance of the solver. A check of the percentage movement at each iteration allows movement to be switched off for that region if the grid converges or begins to diverge. The number of iterations is limited by the need to interpolate the variable needed to move the grid between iterations (see 3.3). The interpolation procedure involves some smoothing of the variable so at each call the impact of the major flow features which drive the grid will be dissipated.

Towards the aim of making the adaption module independent of PHOENICS all working is done in memory independent of PHOENICS.

One call is made to the adaptive code at the end of each sweep, whether for the preparatory phase, the main phase or the bookkeeping phase. This is done to keep the adaptive code as separate as possible from PHOENICS without seriously affecting its performance.

B.5.1 Communication between Adaption Module and EARTH

The majority of the data on the problem within EARTH is held in a single, large one dimensional real array called the F array. A number of functions and common blocks are provided within EARTH to access the required parts of the array. The adaption module accesses the F array using functions written as part of the module, though based on known PHOENICS functions.

Communication is deliberately limited to enable the adaption module to be as independent as possible from PHOENICS so that it may easily be bolted onto an alternative structured CFD code.

Communication between G_ADAPT and EARTH occurs through the argument list passed to it when called from EARTH, the reading of the adaption variable from the F array, the writing of sections of modified grid to the F array, the writing of information on adaption to prepared locations in the F array for the purposes its graphical display, and the access to other variables and the grid stored in the F array for the purposes of interpolation between the grids pre and post adaption. The grid sections are read into the adaption module from the adaption grid file GRIDn.ADP.

The arguments passed to the adaption module when it is called incorporate the sweep and step number and the last sweep and step number, the limits of the problem domain and the current iz slab. The limits of the domain and the iz slab number are used in the access of the F array.

The adaption module also uses the PHOENICS routines BGEOM, DUMP and DUMPXYZ. BGEOM is used internally by EARTH to calculate geometric properties for the grid, such as volume, and by the adaption module to reset those properties following grid movement. DUMP and DUMPXYZ are used to output PHOENICS data files PHI and XYZ respectively.

B.5.2 Tasks in G_ADAPT

This section covers the tasks carried out within the adaption module G_ADAPT within PHOENICS EARTH. The tasks are between the preliminary phase, the main phase and the bookkeeping, or final phase, and into the main adaption algorithm and a number of secondary functions that mainly involve communication between the module and PHOENICS, and between the module and the user.

B.5.2.1 Preparatory Phase in G_ADAPT

The tasks in the preparatory phase include the initialisation of adaption data at the start of run time using data in ADAPT.ADP. The code is also checked by carrying out one pass without the calculated grid movement passed through to PHOENICS EARTH. This phase could be incorporated into the satellite adaption module S_ADAPT, but it is computationally cheap compared to the total run time and to the run time of PHOENICS SATELLITE, and allows more flexibility in using adaption.

B.5.2.2 Processing Phase in G_ADAPT

The main tasks in the main phase of the adaption routine include:-

- Extraction of the adaption variable. This involves the interpolation and manipulation of a chosen variable from cell centres to give a normalised value on grid nodes within the region being adapted. This variable is used to generate the weight components.

- Calculation of different weight components for each coordinate direction by using an arbitrary function on the adaption variable.

- Calculation of movement vectors over adaption region using a point by point Jacobi solver on the LPE equation, calculating all movement on the original grid before updating it.

• Checking of movement vectors to maintain surfaces and lines, and prevent grid crossover.

• Interpolation of the adaption variable between the old and new grids to allow multiple internal iterations of the adaption solver.

The secondary tasks include:-

• Placing modified grid back into PHOENICS after adaption. and the reactivation of the routines within PHOENICS that calculate grid constants.

• Calculation and output of temporary grid movement information.

• Extraction and interpolation of variables from the old to the modified grid and the writing of these variables back into PHOENICS.

• Placing of information on adaption into PHOENICS arrays to allow the data to be displayed graphically using PHOTON and AUTOPLOT. The data may optionally include the total grid movement in each coordinate direction, the calculated weight function for each direction and the normalised and interpolated variable used for adaption. This variable may be a composite of a number of PHOENICS variables and there may not be a directly corresponding PHOENICS variable stored (see section 4.8).

• Output of PHOENICS result file PHI and grid file XYZ at the beginning of each call to the adaption module and at the end of the run. These dumps are optional. Each file has a name which is controlled from within the adaption module and incorporates the sweep number when it is dumped. The name also incorporates the step number if the problem is transient. The files are dumped using the PHOENICS routines DUMP for the PHI file and DUMPXYZ for the XYZ file.

B.5.2.3 Bookkeeping Phase in G_ADAPT

The tasks in the bookkeeping phase include :-

- Calculation of final grid movement by comparing the initial and final grids held in the files GRIDn.ADP.

- Output of a PHOENICS XYZ grid file. Other file dumps are optional, the grid file dump at the end of the run is forced.

APPENDIX C: PSEUDO CODE FOR ADAPTIVE MODULES

C.1 Introduction

This appendix contains pseudo code to show the structure of the adaption initialisation module S_ADAPT and the adaption calculation module G_ADAPT as implemented in the pre processor SATELLITE and solver EARTH of the CFD code PHOENICS.

C.2 SATELLITE Adaption S_ADAPT

One call is made to S_ADAPT from PHOENICS when adaption is activated. The main role of S_ADAPT is to recover geometrical data on chosen adaption regions within the problem domain. This data consists of control functions for the Poisson equation (section 2.5) which can be used to generate the local mesh, and coefficients of parametric equations which are used to model user defined curves and surfaces.

C.2.1 MAIN

```
read region definition patches
sort region definition patches  PATCHSORT
loop through regions
      output regions limits to plot file
      get region grid
      calculate scale factors for error output and minimum cell
clearance
      output grid to grid file
      determine number of and sort blanks in region
      determine number of and sort surfaces and curves in region
      loop through surfaces and curves
            cut surfaces and determine coefficients SURFSORT
            extract surface grid
      end loop
      determine control functions to regenerate grid (Poisson
term)
end loop
end
```

C.2.1.1 PATCHSORT sort region patches
check regions do not overlap and are within grid
link surfaces and curves to regions
link blanks to regions
end

C.2.1.2 SURFSORT Cutting Surfaces and Determining Coefficients
determine direction of surface
break surface into subsurfaces **SURFCUT**
calculate patch coefficients
end

C.2.1.2.1 SURFCUT Breaking Surfaces
determine maximum gradients and second derivatives
find turning points
if turning points are within 1 patch distance then
    multiply turning point tolerance by two
    repeat find turning points step
end if
find discontinuities
determine subsurface limits
end

C.3 EARTH Adaption G_ADAPT

G_ADAPT is split into three main parts, STARTADAPT, ADAPT and BOOKADAPT, which are concerned with setting up adaption data within PHOENICS EARTH, adapting the grid and bookkeeping at the end of the run respectively. The adaptive module can be called at the end of each sweep for steady state problems and additionally at the start of each time step for transient problems. The first and last parts, STARTADAPT and BOOKADAPT are only called once each at the second sweep and final sweep of the run respectively for steady state problems. The second sweep is used because the numbering conventions used within PHOENICS may mean that the run starts at sweep number two. For transient problems STARTADAPT is called at the start of the first time step. ADAPT can be called at any sweep between

the calls to STARTADAPT and BOOKADAPT.

STARTADAPT reads in adaption data from external adaption files, sets up temporary run time files and tests the adaption code.

BOOKADAPT deletes the temporary run time files and determines a measure for the total movement.

ADAPT modifies selected regions of the problem domain to given parameters and constraints.

C.3.1. MAIN

```
if sweep equals 2 then
      initialise adaption routines   STARTADAPT
else if sweep equals last
      dump results files
      book keeping                   BOOKADAPT
else
      dump results files
      run adaption routine           ADAPT
end if
end
```

C.3.1.1 STARTADAPT Initialise Adaption Routines

```
delete old adaption grid files
read input file and write direct access adaption parameter file
read screen plot file and write direct access plot file
open log file
run adaption routine with no movement allowed ADAPT
end
```

C.3.1.2 BOOKADAPT Book Keeping

```
delete adaption parameter and screen output files
determine total movement measure
end
```

C.3.1.3 ADAPT Adaption Routine

```
open adaption files
loop over regions
      read region parameters
      get region grid from most recent grid file
      get control functions for region
      get adaption variable normalised at cell vertices GETVAR
      loop over internal iterations
            calculate adaption weights                    GETWEIGHT
            calculate grid movement                       GETMOVE
            if polar type grid then
                  modify to get correct shape
            end if
            interpolate adaption variable from old to new grid
      end loop
      determine cell angles, if required
      interpolate solution from old to new grid
      write grid to CFD code
      write grid to file
end loop
reset geometry information in underlying code
close adaption files
end
```

C.3.1.3.1 GETVAR Get Adaption Variable

```
get variable at cell centres from PHOENICS
interpolate variable to cell vertices
normalise variable
end
```

C.3.1.3.2 GETWEIGHT Calculate Adaption Weights

```
use chosen weight routine
apply local averaging smoothing to weights, if required
apply power law smoothing to weights, if required
apply exponential smoothing to weights, if required
end
```

C.3.1.3.3 GETMOVE Calculate Grid Movement

calculate raw movement from LPE equation

use blank patch data to set movement in chosen regions to zero

cut movement to chosen fraction of minimum cell clearance

set movement at edges to movement one cell in, if required

fit surfaces and curves

apply movement to grid

calculate percentage change in grid

end