NASA Technical Memorandum 106754

An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations

William John Coirier
Lewis Research Center
Cleveland, Ohio

October 1994

National Aeronautics and
Space Administration

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my thesis advisor, Professor Ken Powell. His freshness of thought, his manner of simplifying complicated processes and most of all his enthusiasm and optimism have been crucial to this work. By sheer necessity, this work has shifted my views to explore less often traveled paths. I must thank Ken for guiding me (and in some cases, literally pushing me) down these paths. This experience will leave my mind forever open to the application of new ideas and processes. Special thanks go to my Branch Chief, Dr. D.R. Reddy, for providing the advocacy for this important research and to NASA in general for providing the opportunity for me to pursue my degree. This has been a very important endeavour, both professionally and personally.

Thanks go to my office mates, Dr. P.C.E. Jorgenson and Mr. C.J. Steffen, Jr., for putting up with both my foul and my happy moods, but most of all for providing me with the very needed ears which to sound off ideas to. Special thanks go to the entire crew manning the "fish cluster", whose well intended criticisms and complements always provided encouragement. Special thanks go to Dr. Erlendur Steinthorson for providing me the geometry for the branched duct and the coolant passage. Mountains of thanks go to Mr. David Fricker for keeping my workstation, marmite, in running order.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF APPENDICES

APPENDIX

CHAPTER I
INTRODUCTION

In this thesis a numerical algorithm is presented which solves the conservation laws of gas-dynamics on a computational domain that is created using a Cartesian, cell-based grid generation procedure. This grid generation process creates non-overlapping volumes which fill the domain whereupon the steady, compressible Navier-Stokes equations are solved in discrete conservation law form using an upwinded, finite-volume approach. A non-traditional means of storing the data associated with the grids is used that is based upon a tree data structure, easily allowing solution-adaptive mesh refinement. The motivation behind this particular solution strategy and the inherent strengths and weaknesses associated with it is outlined below.

1.1 Compressible vs. Incompressible Formulation

Since it is desired to compute single-phase flows of fluids in the continuum range the Navier-Stokes equations are chosen to be solved. The choice of the compressible over the incompressible form of the equations is made since by the introduction of a suitable preconditioning strategy, incompressible flows can be solved with the same, compressible based flow solver. Solving the compressible equations for extremely low Mach number flows causes the equations to become ill-conditioned, and has spurred a great deal of interest in conditioning the resulting viscous and inviscid compressible equations through the use of matrix preconditioning: see, for instance [81][46][85][78][45][18] and many others. For application of preconditioning in an unstructured formulation, see [39] and [40]. This ill-conditioning is due to the increasing disparity of the eigenvalues of the inviscid flux Jacobian with vanishing Mach number. If formulated in terms of primitive variables the ill-conditioning can be traced to a vanishing of the pressure from the Jacobian's diagonal.

Regardless of the formulation, successful progress is being made in developing suitable pre-conditioning strategies, providing a means of retro-fitting existing, compressible based solvers into solvers using a matrix pre-conditioning approach. In this way, a compressibility based, Cartesian-cell code could be used to compute low Mach number flows using a preconditioning strategy.

1.2 Conservative vs. Non-Conservative Formulation

The physical laws governing the flows of fluids are found by treating the fluid as a continuum and applying the concept of conservation to the mass, momentum and energy contained within a control volume, yielding a set of conservation laws. When applied to a differential control volume, which is shrunk to a vanishingly small size, this results in a set of partial differential equations which are typically referred to, as a set, as the Navier-Stokes equations. Since the conservation concept really applies to a volume, it is best, instead, to view the governing fluid equations in the integral format.

The concept of conservation is at the cornerstone of many physical phenomena, and it is no surprise that it can be used to describe fluid flows successfully. From a heuristic standpoint, it only makes sense then to solve the governing equations in a discrete way which mimics this important framework; to satisfy conservation laws discretely upon control volumes which tile the domain upon which the flow solution is desired. Importantly, as pointed out in the landmark work by Lax ([42], [43] and [44]) this type of formulation allows the admittance of weak solutions of the equations. When a proper entropy producing mechanism is chosen, the physically correct weak solution can be found, which will have the correct jumps across discontinuities and the proper speed of the discontinuities. These properties are essential when computing compressible flows since shock waves and contact discontinuities are present in all but the simplest cases. For an internal flow, conservation in even a shock- or discontinuity-free flow is extremely important and is guaran-

teed by the use of a conservative differencing scheme. The finite-volume formulation locally satisfies conservation on a cell by cell basis, which when all the cells are assembled to tile the domain, naturally conserves the quantities globally. It is due to these very important properties that the finite-volume formulation is used here to solve the compressible Navier-Stokes equations.

1.3 Unstructured vs. Structured Grid Formulation

Before the rationale behind the choice of an unstructured type of grid is explained, it is first necessary to discuss the benefits and drawbacks of what is traditionally referred to as a structured grid approach. A structured grid approach stores the data associated with the grid geometry and flow solution in logically ordered one-, two- or three-dimensional arrays (corresponding to solving a problem in one-, two- or three spatial dimensions). This has direct restrictions upon the topology of the grids that may be described and limits the connectivity of cells to their neighbors. For a structured grid approach, each cell has only one cell lying on each of its faces. These local neighbor cells are always needed in a computation, for, say, reconstructing the local solution gradients in a cell, and for a structured approach are found simply by incrementing or decrementing indices in the arrays. This simplification in the data structure has a large impact upon the simplicity of the resulting flow solver and due to this simplicity, can result in computationally efficient solution strategies. By locating all of the data in contiguous locations in memory, which is naturally done by storing the data in arrays, compilers can create very efficient code, and if care is taken on vector class supercomputers, extremely fast computations can result.

Another benefit that can be realized for simple domains is the smoothness of the grid. This grid smoothness is a direct consequence from the solution strategy used to form the grid. There are currently two classes of methods used to generate body-fitted, or structured grids; algebraic based and partial differential equation based. (A compilation of many

methods may be found in [77].) Algebraic based methods are more easily applied to complicated geometries, but typically require a smoothing operation that is locally applied to the algebraically generated grid. This smoothing operation can cause problems, since it does not preclude grid line crossing [75] and does not always yield the desired smoothness that can be obtained by the other class of grid generation procedure, PDE based methods. The most widely used PDE based approach solves elliptic equations where the spatial coordinates of the grid lines are the dependent variables. Clustering is achieved by adding user-specified source terms to the equations. The benefits of this approach come by the smoothness guaranteed by the elliptic equations. Since solutions to elliptic equations satisfy certain differentiability properties and guarantee satisfaction of a local maximum principle, smooth grids can be obtained and grid line crossing can be eliminated. Another benefit of structured grid generation is the ability to cluster high aspect ratio cells in regions where there are obvious viscous layers, such as near walls and in the wakes of bodies. This requires special care in the stretching for viscous computations, which has consequences shown analytically in Chapter III. All of these properties: speed; simplicity; smoothness and anisotropy; have made structured grids a very valuable tool. To highlight the differences amongst the structured, unstructured and Cartesian approaches, representative grids near the leading edge of an airfoil are shown. Figure 1.1 shows a typical grid generated with a structured grid approach for an inviscid computation.

The big drawback of using structured grids is directly tied to the trait which makes them so simple; the grid data structure. By limiting cells to have a set number of sides, the domain about complicated geometries must be decomposed into logical zones or patches. Each patch needs to have the same number of sides as the cells which make up the domain. In two dimensions, this means four-sided patches, since the base elements in the domain are quadrilaterals, and in three-dimensions zones of six sides, since the base elements are hexahedra. Then, in each zone, a grid generation approach is applied and the resulting iso-coordinate lines joined, yielding a grid network and the computational vol-

umes. Depending on the flow solver to be used, the grid lines may need to be contiguous across the patches, increasing the difficulty of the problem. This patching or zoning approach becomes increasingly complicated as the complexity of the geometry increases. For realistic geometries, it can literally take man-months or man-years to generate a single grid. So, a calculation that may only take tens of hours on a supercomputer might have taken months of grid generation time. This downside of structured grids has led to the recent favor given to unstructured grids to compute the flows about complicated geometries.

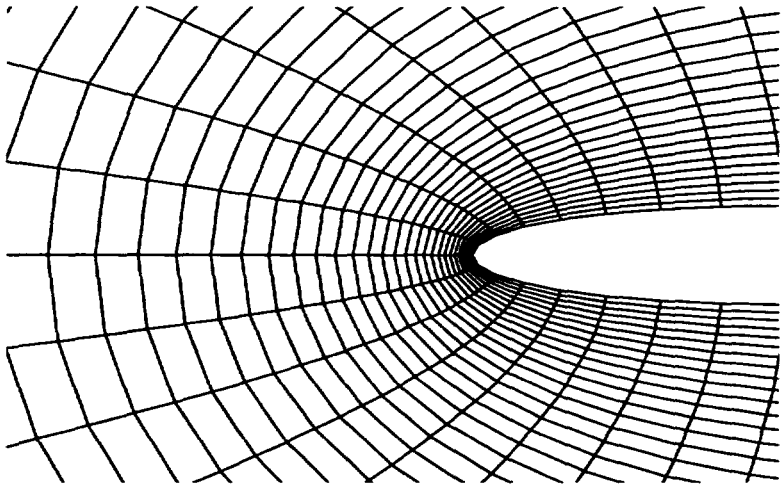


Figure 1.1 Example Structured Grid

1.4 Cartesian vs. Unstructured Grids

Traditional unstructured grids that are in use by the aerodynamic community today are typically based upon triangular (in two-dimensions) or tetrahedral (in three-dimensions) elements. This means that the grid generation process now entails specifying the bounding surfaces and filling the domain with these elements. The volume grid is generated so that the cells on the boundaries of the domain have faces coincident with the boundaries. In this sense, the volume grid is constrained by the boundary discretization. There are two

different methods that are currently being used to generate the volume grids for unstructured meshes. One, the advancing front method, is based upon starting at the boundaries and advancing a "front" of vertices into the computational domain. The new vertices are connected with existing nodes (triangulated) and the front is advanced until the entire domain is tessellated. A good description of this approach and other unstructured approaches may be found in [89] and [5]. Grid size and smoothness can be controlled by source terms which vary in space according to a user specified criteria. Since these source terms rely upon some sort of background mesh, the process is not as straightforward as it might first appear.

A different approach based upon triangulating a cloud of points is also used quite frequently. The basic idea is to consider a point for triangulation, and connect it to its neighbors in a way that satisfies certain geometric constraints. If the existing local points do not satisfy the geometric criteria, a point or points are locally added to satisfy the criteria. In [5], an excellent summary of the successful implementations of the Delaunay criterion to perform the triangulations is made. The Delaunay criterion is shown to have many important implications regarding locality of neighbors, interpolation properties [88], smoothness of the grid and is shown to satisfy certain geometric properties that are inherent in a multitude of problems. In [5], important positivity properties are shown to result from grids that satisfy the Delaunay criterion in two-dimensions, and Delaunay-like criteria in three-dimensions for a linear Galerkin finite-element formulation of Laplace's equation. Similar geometric constraints are found here for a cell-centered, finite-volume formulation, but not in the context of triangular grids (Chapter III). An example of a traditional unstructured grid is shown in Figure 1.2. This grid was generated using a Delaunay-based point insertion algorithm [39].

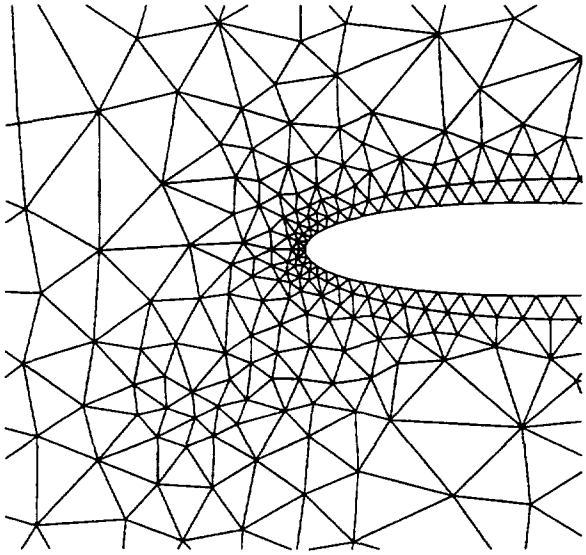


Figure 1.2 Example Unstructured Grid

A sibling approach to the advancing front method is the prismatic grid approach. In this case, layers of cells are advanced from the body surfaces in a hyperbolic type of manner. This generates prism-shaped cells near bodies that may be more suitable for viscous computations. The control of the mesh smoothness can be achieved by constraints applied to the resulting front shape, as in [57], or by using a more hyperbolic grid generation method coupled with a Cartesian-cell based approach, as in [54], or by a more geometric manner, as in [86].

Regardless of the approach, traditionally unstructured grids are gaining popularity, and many useful results are being obtained. The sheer amount of researchers working in this area has led to many impressive advances. There are a large number of very talented and well-funded people working on this approach. This fact, coupled with the strong mathematical background that is present on the finite-element side of the fence for unstructured grids make this a contender that will be very difficult to beat.

All of these methods require the discretization of a surface mesh, and require the resulting volume grid to match the surface mesh where the volume cells are adjacent to the

boundaries. This makes the surface discretization the controlling factor for the grid resolution and quality near the body. For complicated geometries and flows, this brings the user back into the grid generation process, discretizing the surface. If truly automated grid generation is desired, the user should only have to specify functional descriptions of the body and allow the grid generation to proceed automatically.

1.5 The Cartesian, Cell-Based Approach

The Cartesian, cell-based approach presented here performs this important task and generates a volume grid automatically when given the functional or discrete description of multiple bodies and domains. In addition, due to the special data structure used to store the grid and flow data, adaptive mesh refinement is a natural extension of the approach. This enables the possibility of achieving grid converged solutions upon automatically generated grids, bringing the user as far out of the loop as possible.The use of Cartesian cells to discretize a domain is not a new idea. Indeed, it is the simplest possible discretization for domains which are square or rectangular. The novelty of the Cartesian-cell approach arises from the application of Cartesian-cells to non-square domains. For non-simple geometries, all Cartesian-based approaches must perform some type of a specialized procedure on the boundaries to account for the non-alignment of the boundary with the cell geometry. The particular strategies of these special boundary procedures are dependent upon the type of algorithm used on the interior of the domain, and are naturally dependent upon the equation set(s) that are being solved. Typically, the simple geometry of the Cartesian cell is used to help define the usually non-simple intersection tests with the body surfaces to provide the geometric data needed by the boundary procedure. The approach taken here is to create N-sided polygons where the Cartesian-cells straddle boundaries of the grid, and perform flux balances on these conservation volumes. Figure 1.3 illustrates a simple, non-adaptively refined, coarse Cartesian cell based grid on the same geometries as Figure 1.1 and Figure 1.2.

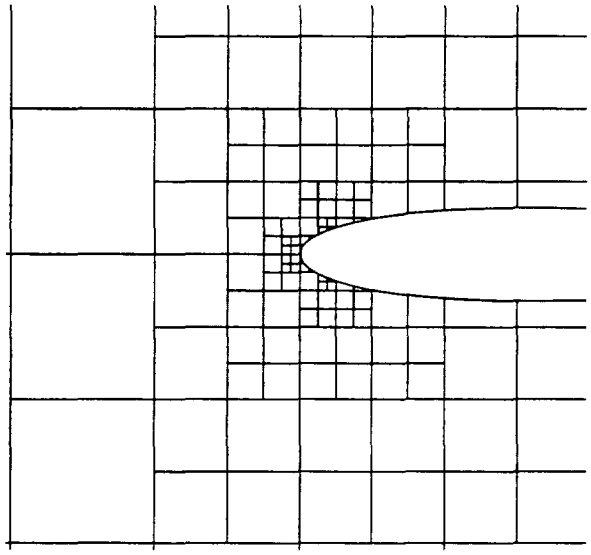


Figure 1.3 Example Un-refined Cartesian grid.

In addition to the geometric flexibility afforded with unstructured meshes in general, and the Cartesian approach in particular, is the ability to perform solution adaptive mesh refinement. Solution-adaptive mesh refinement adds cells locally to regions where an increased resolution is desired. Due to the particular data structure implemented in the Cartesian-cell approach here, the local subdivision of cells is straightforward. Figure 1.4 shows an example of a solution-adapted grid about the same geometry as the preceding figures.

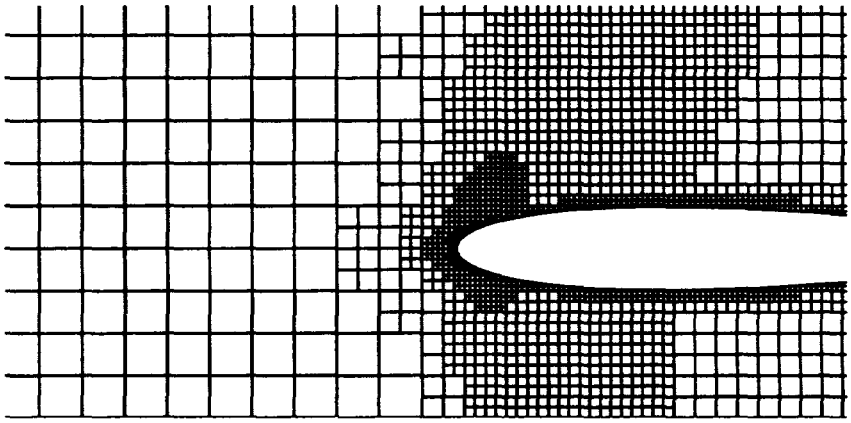


Figure 1.4 Example adaptively-refined Cartesian-cell grid.

1.6 Review of Cartesian Approaches

There are a wide number of fluid-dynamic-related applications to which Cartesian-based algorithms have been applied, and depending upon the simplicity or complexity of the governing solution procedure, different approaches have resulted. To categorize the past work in the Cartesian area, a strategy might be to group the approaches on a data-structure format. That is, categorize the works as to whether the governing data structures follow a structured-grid-like or an unstructured-, finite-element-like approach. The drawback to this data-structure based categorization is that it places too much emphasis on the ingenuity of the programmer, and not on the complexity or lack of complexity inherent in the approach. Here, the past work in this area is categorized by the governing equations being solved. The simplifications or complexities of the Cartesian approach are due to the particular information needed in the boundary procedures, which is naturally dependent upon the equations being solved, hence the rationale behind this categorization.

The Cartesian-based approach has been used quite successfully for the solution of the full potential equation by a number of researchers. One of the first applications was by Purvis and Burkhalter [62], where they used a finite-volume procedure to solve the full

potential equation, and took advantage of the flux formulation at cut-cell boundaries to simplify the boundary procedures. The background mesh was regular, and stored in a single array. Since the flux through the body face is zero by construction, the only contributions to the cell update came through the exposed faces of the cut cells. A more efficient relaxation strategy was used by Wedan and South in [90], who used a line Gauss-Seidel relaxation scheme instead of the point Gauss-Seidel as used in [62]. Neither of these approaches used local refinement. A very successful application of the Cartesian-based approach for the full potential equation is from the landmark work by Young et.al. [91], resulting in the TRANAIR code. There, a finite-element-based procedure is used to solve the full potential equation. A hierarchical representation of the locally-refined, Cartesian cells is stored in an octree data structure, and a GMRES preconditioning strategy is used to accelerate convergence of the scheme. Since no body-conforming meshes need to be defined, and the solution strategy is robust, it is claimed to be easy to use for the novice user. This has resulted in a very useful engineering tool to analyze geometrically complex configurations. As pointed out in [91], this gridding strategy, coupled with the finite-element solution procedure, results in a common framework which can be used to solve a variety of computational physics problems, ranging from acoustics and aerodynamics to electro-magnetic radiation.

For transonic, weakly-shocked flows and unshocked flows, solution of the full potential equation can result in excellent predictions of the flow field. But, when the shock strengths are increased, or the flow is rotational, it is necessary to solve the complete Euler equations. Unsteady, adaptively-refined solutions of the Euler equations were made by Berger and Oliger [15], Berger and Colella [11] and Quirk [63] for Cartesian geometries. These calculations highlighted the excellent results that can be obtained for highly complicated shock hydrodynamic problems when care is take in performing the mesh refinement and flux computations. The flexibility and utility offered for non-Cartesian bodies that was demonstrated for the full potential equation was first extended to the Euler equations by

Clarke, Salas and Hassan [19], where non-refined, finite-volume solutions about multi-element airfoils were calculated using the approach. Later, Morinishi [56] also applied a similar approach, but did not perform adaptive mesh refinement. For adaptively-refined flows about non-Cartesian geometries using the Cartesian-based approach, there is the work by Berger and LeVeque [12][13][14], Pember et. al. [60], Epstein, et.al. [25]. Recently, the work by Quirk [64] and Chiang et.al. [16] for unsteady, adaptively refined flows about non-Cartesian geometries has shown the excellent flow-feature-capturing capability of the Cartesian approach using upwind-based, finite-volume strategies. For the extremely difficult problem of unsteady flows about deforming and moving bodies, the upwind, finite-volume variant of the Cartesian based approach is being investigated by Bayyuuk [9]. For steady inviscid flows, adaptively-refined solutions using an upwinded finite-volume approach are shown by DeZeeuw et.al. in [24][22] and [23] and by Coirier in [20]. The use of state vector splitting for the Euler equations in [58] and [59] and the extension of the state vector splitting concept to the Navier-Stokes equations in [30] and [31] make use of Cartesian grids. The state-vector splitting ideas presented in [58] and [31] present multi-dimensional flux functions based on gas kinetics ideas, but do not show the improved resolution of non-grid aligned discontuities or the improvement when compared to the standard, face-aligned upwind flux formula which is inherent with the premise of using the multi-dimensional flux. The recent use of the Cartesian-based approach in three dimensions by Melton et. al. in [53] and [52], using ideas for the cut cell generation borrowed from computer aided design and computer graphics, shows promise by providing a common way of describing the geometry of the problem.

The only extensions of the Cartesian-cell based approach to the Navier-Stokes equations may be found in the Ph.D. thesis works by Quirk [63] with a finite-volume approach, and by Gooch [31] using a state-vector splitting approach. The state-vector splitting approach is not a conservative method, and as shown in [31], has serious monotonicity problems near discontuities and on boundaries. The work in the finite-volume area by

Quirk in [63] had its major emphasis upon unsteady, upwinded numerics, and concentrated upon obtaining high quality unsteady shock hydrodynamic solutions. There, the extension to the Navier-Stokes equations was brief, and demonstrated for only a few simple test cases. The work performed here carefully extends the finite-volume formulation of the Cartesian cell based approach to solving the Navier-Stokes equations, giving a more complete analysis of the approach and demonstrations of its capabilities.

1.7 Scope of the Thesis

This thesis first addresses implementing the Cartesian-cell approach to solve the Euler equations. Since the proper treatment of the convective terms is essential before the viscous terms can be addressed, Chapter II illustrates the Cartesian-cell approach for solving the Euler equations. In this chapter the approach is first validated against some well-known transonic airfoil flows. Then, the accuracy of the approach is carefully assessed by using an exact solution of the Euler equations to perform a grid convergence study. The results from uniform and adaptive mesh refinement are compared to uniform refinement of a structured grid solver using a similar treatment of the convective terms. The benefits of adaptive mesh refinement are highlighted by performing a grid-convergence study on a very non-smooth flow; the supersonic flow through a mixed-compression inlet.

Next, in Chapter III, a collection of viscous numerical flux functions are analyzed to see which, if any, are good candidates to use with the Cartesian-cell approach. The flux formulae are analyzed upon representative grids by using them to construct finite-volume solutions to Laplace's equation, which is a model equation for the viscous terms of the full Navier-Stokes equations. Local Taylor-series expansions are then found, and the accuracy and positivity of the resulting stencils are examined. Six schemes are critically analyzed, of which two appear to represent the best choices available.

In Chapter IV these two flux formulae are used to compute adaptively-refined solutions of the Navier-Stokes equations for a series of well-known flows. The results from the two different flux formulations are compared to each other and to either accepted computational results, experiment or theory. The most robust of these flux functions is used to compute the flow through a geometrically complicated internal flow.

For flows at high Reynolds numbers, both an altered grid-generation procedure and a different formulation of the viscous terms are shown to be necessary. A grid-generation procedure based on body-aligned cell cutting coupled with a viscous stencil-construction procedure based on quadratic programming is presented in Chapter V.

CHAPTER II
Solution of the Euler Equations Using a Cartesian, Cell-Based Approach

2.1 Preliminaries

The primary effort of this thesis is the development of a Cartesian, cell-based algorithm to solve the compressible Navier-Stokes equations. Since the convective terms contain many important non-linearities, and since a proper discretization of them is essential, effort is first spent creating an accurate solver for the Euler equations.

Finite-volume algorithms are well developed for structured grids, but with the increased geometric flexibility afforded by unstructured grids, emphasis has recently been refocused on improving the state of the art of flow solvers for unstructured grids. For structured grids, the conservation volumes are described by hexahedra/quadrilaterals while traditional, unstructured grids use tetrahedral/triangular volumes. In both of these cases, obtaining cell to cell connectivity and cell geometry is simplified since the number of faces/edges for all cells is the same. This simplification in connectivity and geometry also simplifies the flow solver, since there are always a fixed number of geometric entities to be operated upon. For instance, a three-dimensional structured grid always will have six face neighbors to a cell, accessed by incrementing and decrementing local array indices. A traditional, two-dimensional unstructured grid will always have three neighbors on its faces, and always have the cell geometry be described by three vertices. This geometric/ connective simplification comes at a cost, though. The complexity and level of effort is now switched to the grid generation. The task of grid generation about arbitrarily complicated geometries is a formidable and highly manpower intensive task. It is not uncommon that an order of magnitude more time is spent gridding up the volume grid for a calculation about a complicated geometry than is actually spent computing the flow field. The

approach presented here tries to overcome this difficulty by taking the user out of the grid generation loop as much as possible, and letting the complexity be switched back to the flow solver. This switch of emphasis now means that more innovative solution algorithms are needed, since many of the niceties afforded by the simpler grid structures are no longer available.

For complicated geometries, unstructured grids are much easier to generate than structured grids. Typically, a surface mesh is specified and the volume grid is generated in an unstructured manner by generating a cloud of points in the domain, which are then triangulated, or by, say, an advancing front method, where points are added along a "front" which is advanced from the boundaries to the interior of the domain. Both approaches are based upon a specified surface discretization which is then closely coupled to the volume-grid generation by requiring the specified faces on the boundary surfaces to be faces of cells in the volume grid. In the approach considered here, the volume grid and surface description are not strongly coupled in this manner. The computational boundaries are described functionally, and are "cut" out of the automatically generated, Cartesian-cell based volume grid, yielding N-sided polygons near the boundaries.The ability to create the volume grid automatically and cut cells gives the Cartesian-cell approach its utility, but adds some complexity to the resulting computer code. Since the cell geometry and hence the cell-to-cell connectivity for all cells is *not* known apriori, a unique data structure is needed to describe the conservation volumes. Indeed, it is this complication that sets the approach apart from most of the traditional unstructured grid approaches that are prevalent today.

In this work, the grid is generated recursively from a single cell, and during the creation of the grid, the hierarchical relation between newly created cells and their parents is stored in a binary-tree data structure. The cut cells, which are the background Cartesian cells *cut* into N-sided polygons, are created automatically using many concepts borrowed from computer graphics applications, and since they are hierarchically related to their Carte-

sian-cell parents, they are also stored in the tree. This procedure of cell cutting is a subject unto itself, and its robustness is absolutely crucial for the utility of this approach. For clarity, the cell-cutting procedures are described in detail in Appendix A: although not presented here, their importance can not be overstated.

This chapter first outlines the data structures implemented and the resulting algorithms used to obtain information from them. Then, an upwind, cell-centered, finite-volume approach is described in detail and demonstrated and benchmarked for a set of well known inviscid test cases. Finally, the accuracy of the Cartesian-cell approach is assessed by performing a grid-refinement study for an analytical solution to the Euler equations, and is compared to the results obtained from a standard, structured-grid approach.

2.2 Some Comments on Grids/Data Structures

Unlike structured flow solvers, where grid connectivity is implicit to the algorithm, unstructured solvers require connectivity and geometric data to be supplied in a more explicit manner. The amount of data explicitly stored and the amount that can be found implicitly by interrogating the stored data structures is very important from a usability and efficiency standpoint. Specifically, it determines the way actions are performed on the data stored, and ultimately limits the type of algorithms that can be employed. This issue is not as important for structured grids, where cell connectivity and grid data are easily obtained because the cell and flow data are stored logically in two and three-dimensional arrays: connectivity is implicitly found by incrementing/decrementing indices.

For traditional unstructured grids, connectivity is needed that allows cells to know cell-to-cell, cell-to-edge and cell-to-vertex connectivity. A common way that this is done is by storing cells and vertices in long, one-dimensional arrays (vectors) and storing the cell-to-cell, cell-to-vertex, vertex-to-edge connectivity in each cell or at each vertex as references to the indices of these long arrays[61]. This indirection is something that all unstructured

solvers need, and how this indirection is handled delineates the resulting data structures. Since these lists are data structures that are *global* entities to the code, a *local* change in the grid, by say, cell refinement, is felt *globally* by the data structure. A division of a cell located in the beginning part of the list will change the indexed locations of all cells following it in the list. Maintaining this global structure through these local changes can be difficult. Perhaps more importantly, domain decomposition is not readily achieved, since the grid is stored in an essentially uni-dimensional structure. Domain decomposition can prove to be crucial for applying and developing algorithms for parallel architectures, and is not readily obtained from traditional unstructured grid data structures.

2.3 A Binary-Tree Data Structure

In the approach presented here, the grid and flow data are stored and accessed through the use of a hierarchical data structure: a binary data tree. The binary tree is really a logical geometric abstraction that has computational analogs and, as a result, has a natural domain decomposition. As is shown in Figure 2.5, the tree is grown through the development of the grid by cell division. The grid typically begins with a single mother, or toot cell, and grows by a recursive subdivision of each cell into its children. Each child is geometrically contained within the boundaries of the parent cell, and is located logically below the parent cell in the tree. Figure 2.5 illustrates this concept for the subdivision of a single cell, cell A, isotropically into 4 cells, cells D, E, F and G. The cell division occurs by first splitting the parent cell in x, and then the two children in y. In this figure, cells B and C are the children of cell A, and cells D and E and cells F and G are the children of, respectively, cells B and C. Arbitrary subdivisions of the cells are allowed during the process, only requiring that the newly created cells be non-overlapping polygons that fill the space occupied by the parent cell. To take advantage of the smooth grid that can be achieved, the root cell is taken to be a square Cartesian cell of unit aspect ratio, and cell division is obtained by isotropically splitting each cell into four, equal area children. Since N-sided cut cells

are obtained by "cutting" them out of their background, Cartesian cell, they are always logically locatable in the tree.

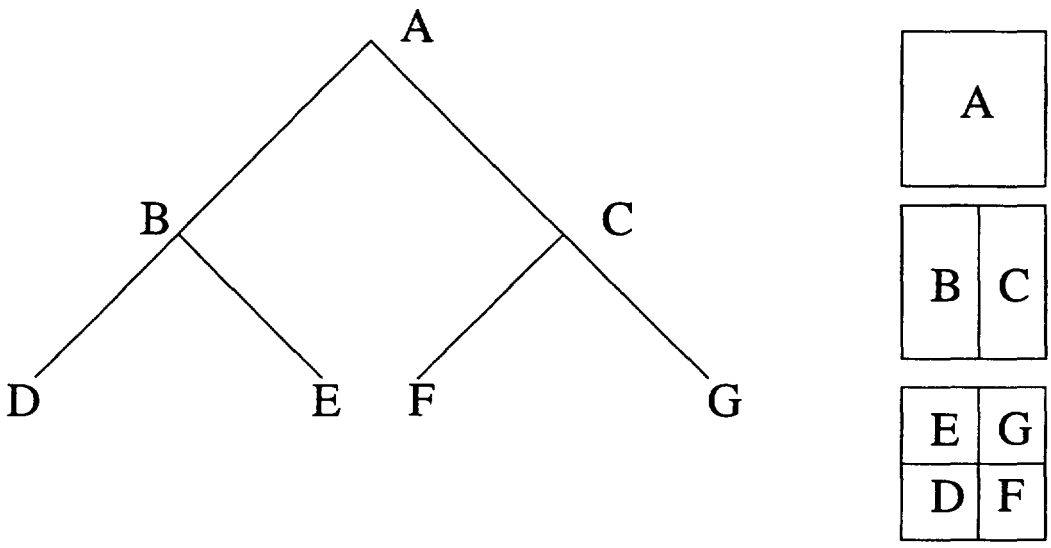


Figure 2.1 Illustration of Binary Tree

Before anything more can be said about using this type of data structure in a flow solver or grid generator, it is necessary to bring out some important nomenclature that is used to refer to the tree. A node is a location in the tree where a logical branching occurs. A leaf is a node of the tree located at the bottom of the tree, and therefore has no children, while the root of the tree is the location where the tree begins. More specifically, a root is a node that has no parent, and has only children, while a leaf is a node that has no children, and only has a parent. Figure 2.6 shows this for a simple tree. Since the grid generation results in the desired grid by spawning the whole tree, flow calculations are carried out on the cells represented by the tree leaves. Important information can be stored at the interior nodes of the tree. An example, not used here though, would be the storing of the information needed to perform the restriction and prolongation operators for multi-grid, as in [24].

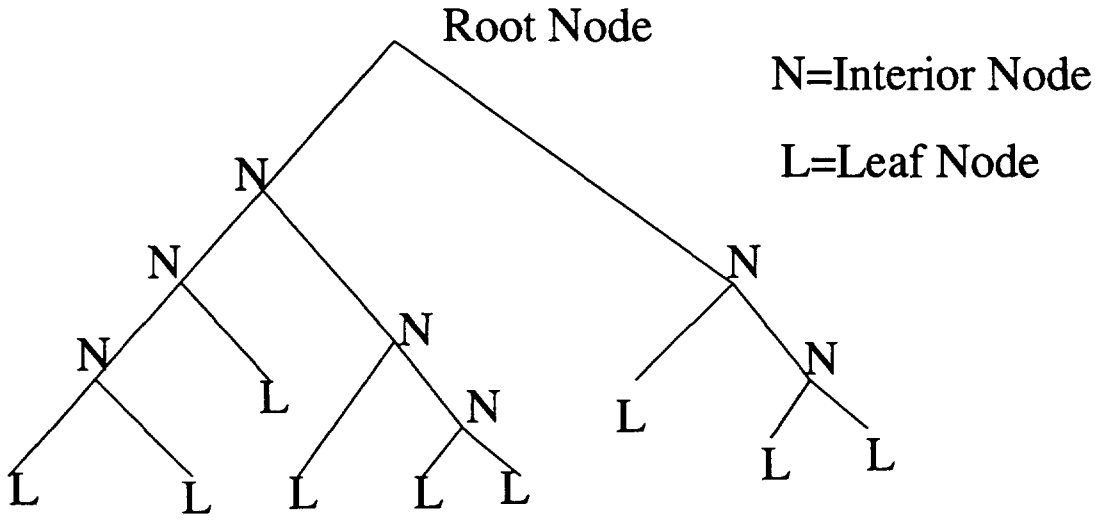


Figure 2.2 Illustration of Root, Nodes and Leaves of a Tree

Regardless of the type of cell division (by isotropic division of a logically square cell or an anisotropic splitting of an N-sided polygon), the tree data structure contains important logical information that can be used to obtain cell connectivity and to visit all cells in a logical manner. The tree is replicated in code by each cell having a pointer to its parent and a pointer to each of its two children. Each node of the tree has a separate location in memory which contains data in itself and pointers to other data structures as well. It should be noted here that the entire grid generator and flow solver are written in ANSI standard C: All pointers actually refer to a location in memory reserved for a particular data structure and data type and do not refer to a type of indicial notation, as in FORTRAN. The tree is an assemblage of nodal data structures where each node has a pointer to its parent and its two children. In addition, each tree node has pointers to flow data structures and, for cut cells, pointers to a structure that contains the geometric data needed to describe the cell. Flow data structures are allocated only for leaves of the tree. Cut cells are described by a local linked list of pointers to a globally-maintained list of vertices, while Cartesian cell geometry is described completely by centroid location and cell size. Since the number of cut cells will be small, storing this list of vertices does not create a significant amount of

memory overhead. Carrying around pointers to auxiliary data structures is inexpensive, since if the auxiliary data structure is not needed, memory is not allocated for it, and the only extra space taken is a pointer in the calling data structure.

For the binary-tree data structure considered here, there are a few simple operations that are performed upon the tree: visiting the tree; inferring face-to-cell connectivity; inferring vertex-to-cell connectivity; tree spawning and tree pruning. The following sub-sections illustrate these basic operations.

2.3.1 Visiting the Tree

Storing the tree in the manner described above provides a logically recursive means to visit the tree resulting in a modular approach to programming the solver. In addition, since important geometric data are inherently stored in the hierarchy of the tree, the tree also provides a logical means of obtaining cell-to-cell connectivity. There are only two types of cells that exist in the grid: flow cells and no-flow cells. Flow cells are Cartesian cells and cut cells that are located logically in the computational domain and are cells upon which flux balances are performed. No-flow cells are cells that are exterior to the domain, and are not considered for computation. Regardless, all cells are located at the leaves of the tree (they have no children). By using a recursive technique, all cells can be visited using a depth-first recursion as is shown by the following pseudo code:

```
function Visit_Tree_to_Leaf(TREE NODE Node)
if(this is a leaf of the tree)
    if(this is a flow_cell) perform action upon tree leaf
    return
end_if
Visit_Tree_to_Leaf(Node->LEFT_CHILD)
Visit_Tree_to_Leaf(Node->RIGHT_CHILD)
return
```

As can be seen, invocation of this routine visits all nodes located below Node in the

tree. All leaves of the tree are visited by invoking this routine at the root node of the tree, and when invoked at a branch of the tree, all leaves on that branch are visited. Since all cells below Node are geometrically contained within its boundaries, this allows a simple means of decomposing the flow domain and visiting cells within each decomposition. Although no special use is made of this trait, it should have direct implications on a parallel implementation of the solver. Figure 2.7 shows the order of visitation using this depth first recursion for a small sub-branch of a tree. The visitation is performed by invoking the function represented in the pseudo-code shown in Figure 2.3, where the action at the leaves is simply to print out the "name" of the leaf.

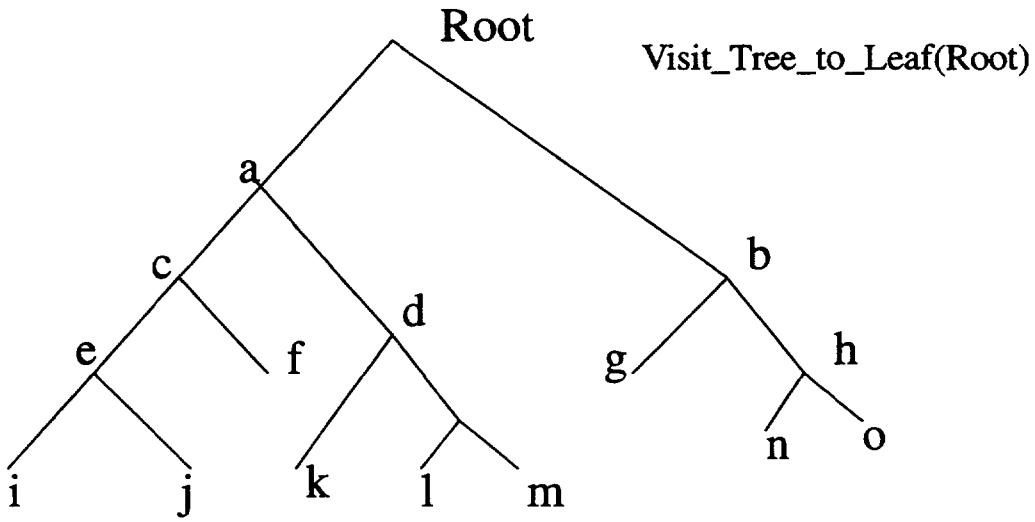


Figure 2.3 Illustration of Depth-First Recursion upon a Sample Tree by invoking Visit_Tree_to_Leaf(Root)

2.3.2 Inferring Connectivity From the Tree: Order-One Neighbors

Cell-face and vertex neighbors are needed to perform flux computations, gradient reconstruction and a plethora of other important functions. Cell-face neighbors are defined as cells that share a face with the cell in question. Since the tree inherently contains the geometric hierarchy of the grid, it can be used to obtain cell-face neighbors. Cell-face neighbors are found by logically traversing the tree in a search direction determined by the outward pointing normal to the face. The tree is traversed upwards until one or both of the children below a candidate node are in the search direction. Then it is searched downward through all branches of the tree that are in the search direction *and* have a face that is coincident with the cell face to which face neighbors are needed. It is useful to note that this search procedure applies to any shape of cells, with arbitrary numbers of neighbors along an edge. If the cell creation satisfies the requirement of all new cells being geometric subsets of the parent cell, all neighbors can be found with this search procedure. If during the downward phase of the search, a cell is a leaf, *and* it is a flow cell, *and* it has a face match to the candidate face, it is added to the list of cell-face neighbors.

The concept of this particular tree-searching algorithm is best described by a simple example. Figure 2.8 shows the sub-tree and the upward and downward search phases for a sample grid. The East-face neighbors to cell C are desired, so that the search direction is as indicated in Figure 2.8. The upward search phase begins at node C and terminates at the tree sub-node S. The downward search phase begins here, and recursively searches down the tree, adding the matched neighbors to the list. The upward searching logic dictates that the search procede up the tree until one or both of the children below the candidate cell are in the search direction. The downward searching logic dictates recursion down the tree, through branches whose children have faces in the search direction.When the downward search reaches a leaf, if the leaf has a face that is geometrically contained within, is coincident to, or spans the search face, it is a cell-face neighbor. Although in the worst case one

might need to search the entire tree, this is a rare happenstance, and for the entire mesh the expected number of searches is much smaller.

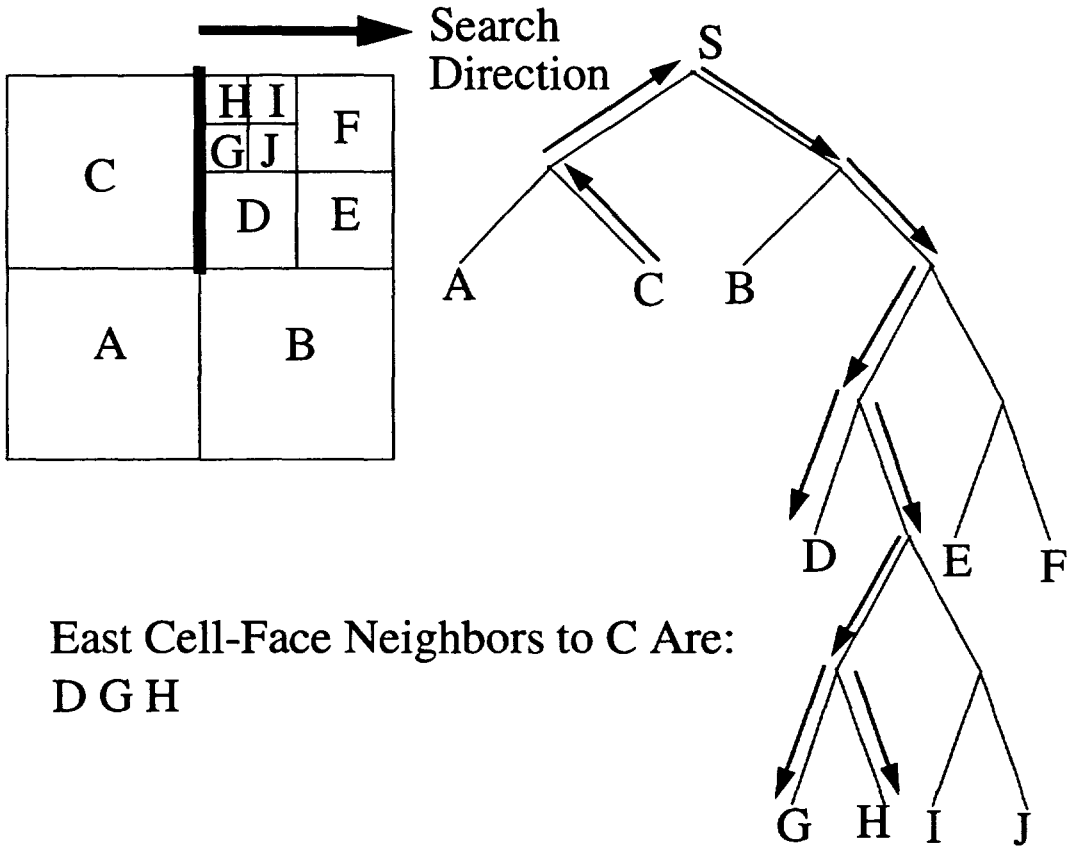


Figure 2.4 Example cell-face neighbor search path in tree.

The search direction of the tree is found from the geometry of the face where the neighbor is desired. Geometrically, the face is described by its outward pointing unit normal, $(\hat{n}_x, \hat{n}_y)$, the location of its midpoint, (x_{mid}, y_{mid}) and the face endpoints. A search direction can be formed and candidate cells can be examined to see if they are in the desired search direction. A vector, V_c, is formed emanating from the face midpoint to the candidate cell centroid. The sign of the dot product of this vector with the search direction is examined, which indicates whether the candidate cell is in the search direction or not. An illustration of the search direction is shown in Figure 2.9, where it is desired to find

the neighbors of the East face of Cell C.

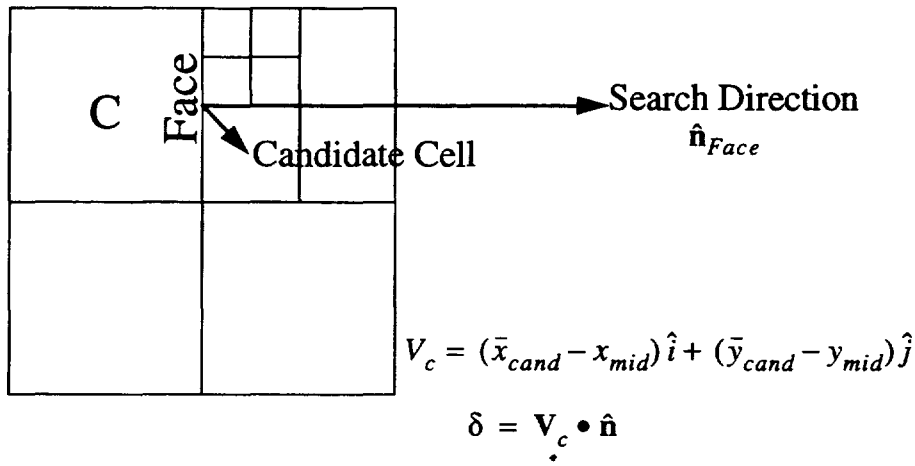


Figure 2.5 Search Direction for Neighbor Finding

Vertex neighbors are found from a list-directed, recursive search of the already found cell-face neighbors, since the vertex neighbors of a cell are face neighbors of the original cell's face neighbors. Vertex neighbors are neighbors of a cell that *only* share a vertex with the cell in question. The search procedure recursively visits the neighbors of the two faces that share the candidate vertex, adding matched neighbors to a local list, finally terminating the search after the originating cell is visited. After the search is terminated, the local list is examined, and cells that are already face neighbors of the originating cell are deleted. This procedure is necessary since vertices of degrees greater than four can occur when using the distance cutting grids to be investigated in Chapter IV.

2.3.3 Cell Refinement: Growing Tree Sub-Branches

One of the strong points of unstructured grids is the ability to perform adaptive mesh refinement by local mesh enrichment. The ease in which this refinement is performed for the Cartesian-mesh approach is afforded by the tree data structure. The use of the binary

tree allows refined cells to be added to the domain by simply creating new sub-branches logically below the refined cell. Figure 2.10 illustrates this where a single cell in a sub-branch is isotropically refined into 4 cells. Use of the tree allows this local mesh refinement to be maintained locally in the data structure; before refinement, cell b had no children, and was considered to be a leaf. After refinement, it has a whole sub-branch of the tree below it. Special routines to alter the connectivity of neighbors to the refined cell are not needed: the connectivity is logically maintained by the tree.

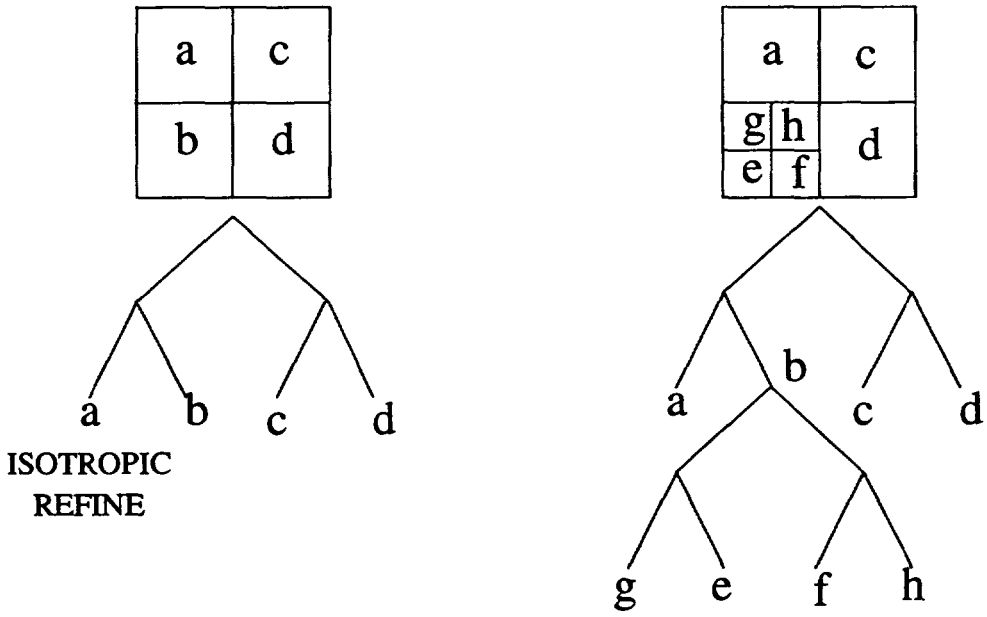


Figure 2.6 Cell Refinement via Tree Sub-Branch Growth

2.3.4 Cell Coarsening: Pruning the Tree

Cell coarsening may be needed in regions of the flow where there is an excess of resolution. Cells are coarsened by merging cells that share a face, creating a larger cell from their union. If this merging of cells is restricted always to lie in a sub-branch, this process can be viewed as simply pruning the tree up a set level of branches. Figure 2.11 shows this process where the cells, g and e and h and f are asked to coarsen, and form two larger

cells from their union. As in the cell refinement, this change in the mesh topology is only local in the grid and the data structure. The tree still maintains all connectivity in a logical manner.

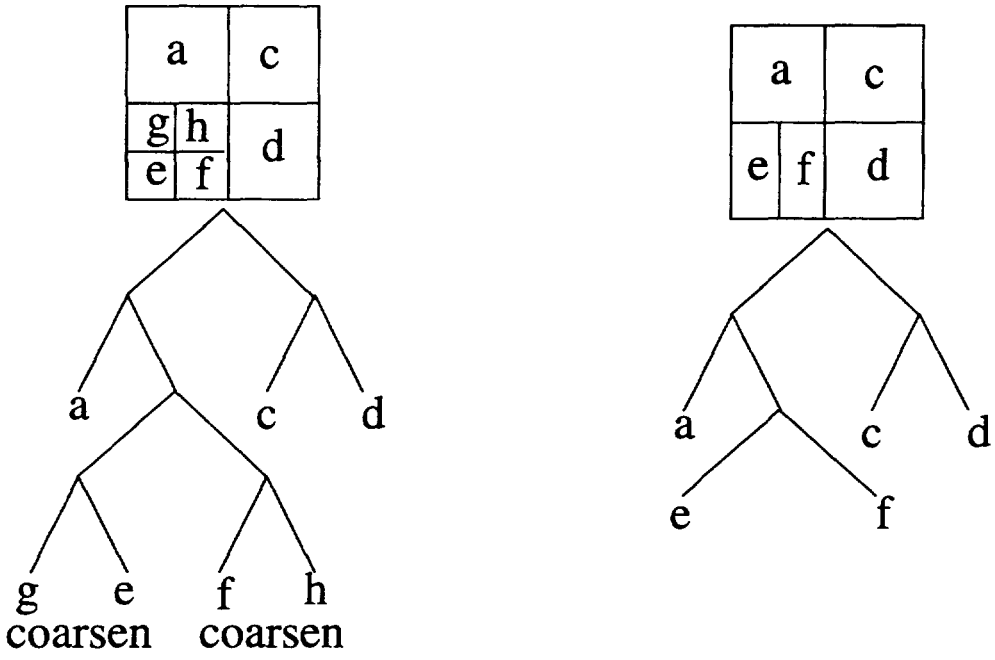


Figure 2.7 Illustration of Tree Pruning for Cell Coarsening

2.4 Solution of the Euler Equations Using a Cartesian, Cell-Based Approach

The Euler equations are solved using a cell-centered, finite-volume, upwinding approach. A limited, linear reconstruction of cell averaged data is used to provide input to a numerical flux function, yielding the flux through cell-to-cell interfaces. The numerical fluxes are computed in an upwind fashion using an appropriate approximate Riemann solver. These fluxes are then used to perform a flux balance upon the conservation volume, which is then used to advance the conserved variables in time. The procedure follows standard practice for a finite-volume scheme. The solution procedure can be broken up

into three stages; reconstruction; flux construction; and evolution. Each of the three stages is described in detail below.

2.4.1 Minimum-Energy Reconstruction

The cell primitive variables in each cell are reconstructed, in the spirit of MUSCL interpolation [79], using a linear reconstruction procedure. The reconstruction used is based upon the Minimum-Energy reconstruction presented by Barth and Frederickson in [4]. The process of reconstruction to arbitrary degrees of accuracy using this approach is presented in [4][5][6] and [20]; A similar interpretation is presented below.

Reconstruction can be viewed as being the discrete inverse of the cell-averaging process. Define the cell average of an arbitrary function, u, to be

$$A_n(u) = \frac{1}{A}\int_A u dA = \bar{u} \tag{2.1}$$

The reconstruction solves the inverse of the cell-averaging process: find the expansion about the cell centroid to K-th order, $u^k(x, y)$, using the cell-averaged data of the cell to be reconstructed and the cell-averaged data of a set of support cells surrounding the cell. The support cells are typically taken to be nearest neighbor cells (cells that are face and vertex neighbors of the cell), but this restriction is not necessary. Indeed, to perform higher-order reconstructions, it is may be necessary to include a larger support set.

By expanding $u^K(x, y)$ in terms of zero-mean polynomials, conservation of the mean of the object cell is ensured, resulting in the general expansion

$$u^K(x, y) = \bar{u} + \sum_j \alpha_j \Psi_j(x, y) \tag{2.2}$$

The Ψ_j are constructed so that their cell averages vanish. These zero-mean polynomials are constructed from the set of polynomials

$$\Omega_j = x^n y^m \tag{2.3}$$

$\forall (n+m) \leq K$. This means that the zero-mean polynomials are found from the cell averages of these base polynomials as

$$\Psi_j = \Omega_j - A_n(\Omega_j) \tag{2.4}$$

For a linear reconstruction, the quadrature for the Ψ are already computed as a matter of course; they are simply the cell centroid, $(\bar{x}, \bar{y})$. For a higher order reconstruction, they can be obtained during a preprocessing step, using a numerical quadrature. For a linear expansion, this results in the expression

$$u^1(x,y) = \bar{u} + u_x(x-\bar{x}) + u_y(y-\bar{y}) \tag{2.5}$$

where the α_j have been replaced with a more meaningful representation.

The Minimum-Energy reconstruction minimizes the Frobenius norm of the differences between the cell averages of the reconstructed polynomial and the cell averages of the support set. This, in essence, examines the quality of the reconstruction by taking its cell average in the neighboring cells and minimizing the difference between this average and the true cell average in the support cells. That is, minimize with respect to the α_j

$$S = \sum_n \omega_n \left[A_n(u^K - u)\right]^2 \tag{2.6}$$

Taking $\frac{\partial S}{\partial \alpha_j} = 0$ and solving for the α_j results in the linear system

$$L_{ij}\alpha_j = b_i \tag{2.7}$$

where

$$L_{ij} = \sum_n \omega_n A_n(\Psi_i) A_n(\Psi_j) \tag{2.8}$$

$$b_i = \sum_n \omega_n (\bar{u}_n - \bar{u}) A_n(\Psi_i) \tag{2.9}$$

For a given mesh, L_{ij} is dependent only upon the geometry, and not the solution. Hence, it can be formed, inverted and pre-multiplied with the b_i, yielding a summation only over the support cells. This preprocessing step makes the reconstruction procedure efficient; once a grid has been generated, it only requires a simple sum over the support cells to

compute the reconstruction. It should also be noted that this type of reconstruction does not require any special ordering of points, and is K-exact in the sense that it reconstructs polynomials of order K exactly. The ω_n are introduced in (2.6) to allow a data-dependent reconstruction, and, as is mentioned by Barth in [5], can be chosen to be functions of the geometry and/or the solution, but for simplicity $\omega_n = 1$.

Application of the Minimum-Energy reconstruction technique for a linear reconstruction yields the following form for the L_{ij} and b_i

$$L = \begin{bmatrix} \sum_n (\bar{x}_n - \bar{x})^2 & \sum_n (\bar{x}_n - \bar{x})(\bar{y}_n - \bar{y}) \\ \sum_n (\bar{x}_n - \bar{x})(\bar{y}_n - \bar{y}) & \sum_n (\bar{y}_n - \bar{y})^2 \end{bmatrix} \tag{2.10}$$

$$b_i = \begin{bmatrix} \sum_n (\bar{u}_n - \bar{u})(\bar{x}_n - \bar{x}) \\ \sum_n (\bar{u}_n - \bar{u})(\bar{y}_n - \bar{y}) \end{bmatrix} \tag{2.11}$$

The inverse of L is found as

$$\|L\| = \left(\sum_n (\bar{x}_n - \bar{x})^2\right)\left(\sum_n (\bar{y}_n - \bar{y})^2\right) - \left(\sum_n (\bar{x}_n - \bar{x})(\bar{y}_n - \bar{y})\right)^2 \tag{2.12}$$

$$L^{-1} = \frac{1}{\|L\|}\begin{bmatrix} \sum_n (\bar{y}_n - \bar{y})^2 & -\sum_n (\bar{x}_n - \bar{x})(\bar{y}_n - \bar{y}) \\ -\sum_n (\bar{x}_n - \bar{x})(\bar{y}_n - \bar{y}) & \sum_n (\bar{x}_n - \bar{x})^2 \end{bmatrix} \tag{2.13}$$

The inverse, computed in a preprocessing step, is used during the reconstruction to find the α_j as

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = L^{-1} b \tag{2.14}$$

which is computationally efficient, since the b_i involve only a simple sum over the support set.

2.4.2 Gradient-Limiting Procedure

The reconstruction of cell-averaged data shown above does not preclude the introduction of new extrema: there is no means to ensure that the reconstructed solution is bounded by the data used to perform the reconstruction. To enforce this, the reconstruction is limited by evaluating the cell-averaged data of the support cells used and reducing the reconstructed gradient to achieve monotonicity of the data. This will in turn guarantee monotonicity of the solution if the numerical flux function is a positivity-preserving function (which for an upwind scheme is sufficiently implied by positivity of the dissipation matrix), and provided that a proper choice is made for the time step. The concept of restricting the local solution to be bounded by its immediate neighbors is based upon a discrete interpretation of a local maximum principle, and is used to evaluate the stencils obtained for a model equation of the Navier-Stokes equations, as will be shown in a subsequent chapter.

To ensure monotonicity of the reconstruction at cell interfaces, the reconstructtion is required to be bounded by the data used to perform the reconstruction. Since the flux quadrature will be performed at the cell interfaces, a less diffusive limiter could be formed by evaluating the reconstructed function at these quadrature points. But, here the function is evaluated at the extremities of the cells, which, for a linear reconstruction, will yield the minimum and maximum variations of the reconstruction.The limiter presented here follows that by Barth and Jespersen [8]. Since a robust scheme is desired, at the slight expense of a more diffusive operator, the entire gradient is limited with a single limiter ϕ, as

$$u^1 = \bar{u} + \phi\,[u_x(x-\bar{x}) + u_y(y-\bar{y})\,] \tag{2.15}$$

In addition, since a set of reconstructions for the four primitive variables is limited, $\vec{U} = (\rho, u, v, P)$, the minimum value of the limiter over all of the individual primitives is found, and applied to all the equations. That is, let ϕ_j be the limiter found for the j-th primitives reconstruction. A single limiter, $\Phi = min(\phi_j)$ is found and applied to all reconstructions as

$$\mathbf{U}^1 = \bar{\mathbf{U}} + \Phi\,[\mathbf{U}_x(x-\bar{x}) + \mathbf{U}_y(y-\bar{y})\,] \tag{2.16}$$

The ϕ_j are found by examining the cell averaged values of the support set, and the unlimited values of the reconstruction at the vertices of the conservation volume. The minimum and maximum of the data used in the reconstruction for the j-th cell from the set of N support cells are

$$u_j^{min} = min(\bar{u}_j, \bar{u}_n) \qquad n = 1, N \tag{2.17}$$

$$u_j^{max} = max(\bar{u}_j, \bar{u}_n) \qquad n = 1, N \tag{2.18}$$

The limiting procedure requires the reconstruction at each vertex to be bounded by the min and max values shown above. If $u_i^K = u^K(x_i, y_i)$ is the unlimited reconstruction evaluated at the i-th vertex, then the limiter is formed by examining the sign of the difference between this value and the cell-centered value. If this sign is positive, the reconstruction is limited to be less than the maximum value, (2.17), while if the sign is negative, it is limited to be greater than the minimum value, (2.18). If there is no variation, or it is numerically negligible, then there is no need to limit. That is,

$$\phi_j = \begin{cases} min\left(1, \dfrac{u_j^{max} - \bar{u}_j}{u_i - \bar{u}_j}\right) & if \quad u_i - \bar{u}_j > 0 \\ min\left(1, \dfrac{u_j^{min} - \bar{u}_j}{u_i - \bar{u}_j}\right) & if \quad u_i - \bar{u}_j < 0 \\ 1 & if \quad u_i - \bar{u}_j = 0 \end{cases} \tag{2.19}$$

An exasperating consequence of limiting can be a phenomenon loosely referred to as "limiter cycling". By necessity, the limiting must be performed in a non-linear fashion. Limiter cycling is caused by this non-linearity, coupled with the non-smoothness of this particular limiter, and its tendency to turn on due to random noise in smooth regions of the flow. The effect of this phenomenon may be reduced by the use of limiter freezing. That is, after the residual has dropped a set amount, or if the iteration count has reached a set fraction of the total number of iterations, the limiter is "frozen"; it is not recomputed, instead using the value computed at the stage when it was frozen. This procedure, although ad-hoc, can help in situations where a limiter cycle has developed. This does imply that the solution can no longer be guaranteed to be non-positive. In practice, though, this restriction can give reasonable results. Although it is not guaranteed to eliminate the problem, introduction of a smoother limiter, as in [84] or [1], could help with this phenomenon.

2.4.3 Numerical Flux Construction

When solving any integral conservation law using a finite-volume technique, it is necessary to approximate the flux through the boundaries of the conservation volumes. By first principles, a conservation law relates the volumetrically-averaged rate of change of a conserved quantity in a conservation volume to the fluxes of this quantity through its faces and the rate of their production in the volume. For the inviscid flow of a non-conducting, compressible fluid, application of the concepts of conservation of mass, momentum and

energy results in a set of integral relations; the Euler equations. Written using tensor notation the conservation law is stated

$$\frac{\partial}{\partial t}\int_{Vol} q_i dV + \oint_S E_{ij} n_j dS = 0 \tag{2.20}$$

where $n_j dS$ is the differential surface area vector, and q_i the vector of conserved quantities. In two dimensions, the conserved quantities are

$$q_i = (\rho, \rho u, \rho v, \rho E) \tag{2.21}$$

and the Cartesian components of the flux tensor are written as

$$E_{i1} = (\rho u, \rho u^2 + P, \rho uv, \rho uH) \tag{2.22}$$

$$E_{i2} = (\rho v, \rho uv, \rho v^2 + P, \rho vH) \tag{2.23}$$

Following standard notation, the Cartesian components of velocity along the x and y axes are u and v, respectively. The definition of the fluid total enthalpy, H relates the conserved quantities to the hydrostatic pressure, P as

$$H = E + \frac{P}{\rho}, \qquad H = h(P,\rho) + \frac{(u^2+v^2)}{2}, \qquad E = e(P,\rho) + \frac{(u^2+v^2)}{2} \tag{2.24}$$

from which a calorically perfect equation of state closes the relation between (h, e) and (P, ρ). Letting γ be the ratio of the specific heat at constant pressure to that at constant volume, the pressure is directly related to the conserved quantities as

$$P = (\gamma - 1)\left(q_4 - \frac{1}{2q_1}(q_2^2 + q_3^2)\right) \tag{2.25}$$

The finite-volume method approximates the conservation laws in (2.20) over the domain by evaluating individual conservation laws on non-overlapping control volumes

that span the entire domain. By constructing the fluxes through the interfaces of these control volumes, and letting these fluxes contribute equally to the cells that share the interfaces, the conservation laws are discretely satisfied *locally* by construction and *globally* by the mutual cancellation of fluxes through the faces of the volumes when they are assembled into the domain. It is because of this basic and very important property that the finite-volume technique is as powerful as it is.

The conservation laws, specialized to individual control volumes in two dimensions whose shapes are invariant in time and are described by arbitrary polygons, can be written as

$$A\frac{\partial \bar{\mathbf{q}}}{\partial t} = -\sum_{edges} \mathbf{F}\Delta S \tag{2.26}$$

where F is the flux through the i-th face of the polygon describing the conservation volume. For the Euler equations, with a unit (outward pointing) normal, $\hat{n} = (\hat{n}_x, \hat{n}_y)$ the flux can be written as

$$\mathbf{F} = \begin{bmatrix} \rho u_c \\ \rho u_c u + \hat{n}_x P \\ \rho u_c v + \hat{n}_y P \\ \rho u_c H \end{bmatrix} \tag{2.27}$$

where $u_c = u\hat{n}_x + v\hat{n}_y$ is the inner product of the velocity vector with the (outward) pointing unit normal to the faces (contravariant velocity) and $v_c = -\hat{n}_y u + \hat{n}_x v$ is the covariant velocity.

Regardless of the form of the numerical flux, F in (2.26), the surface integral must be performed numerically using some sort of quadrature. Since a linear reconstruction procedure is used, which has a truncation error of second order, it is only expected to get a sec-

ond-order scheme. Therefore, it is intuitively obvious, and defended numerically in [4], that using a quadrature of higher order for the flux integral is unwarranted. So, here a second-order Gaussian quadrature along the edges is used, which translates simply to evaluating the kernel of the integral (2.26) at the edge midpoints. From this, the semi-discrete form of the Euler equations can be written as

$$\frac{\partial \mathbf{q}}{\partial t} = R \tag{2.28}$$

where the residual, R is

$$\mathbf{R} = -\frac{1}{A} \sum_{edges} \mathbf{F}(\hat{x}_{mid}) \Delta S \tag{2.29}$$

It is a bit understated to say that the modeling of the true flux, (2.27), by a discrete numerical flux has been the subject of much research. As a result of this concentration of work, there are a multitude of ways to form the fluxes. The two most used numerical fluxes are central differencing with explicitly-added (scalar) dissipation and uni-dimensional upwind schemes. The choice that results in a numerical flux that closely mimics the physics of the true flux would always appear to be the best. The choice made here is to use a flux based upon the solution of a local Riemann problem, oriented normal to the face, with initial states determined by the reconstructed data on the two sides of the cell interface: an upwinded formulation based on Godunov's scheme. The cost of solving the Riemann problem exactly, as in Godunov's scheme, might not be warranted for the dynamically mild flows here, so an approximate Riemann solver is used instead.

For a given means of reconstructing the left and right states at the conservation volume interfaces, it is the choice of the particular approximate Riemann solver that delineates the upwind schemes in use today. Effort is currently underway to develop new upwind flux formulations based on uni- and multi-dimensional flux functions, of which the uni-dimen-

sional schemes appear to be the most robust. In this thesis, three flux formulations are used; Roe's flux difference splitting[68], van Leer's flux vector splitting[80] and Liou's Advection Upstream Splitting Method [48]. Computationally, it is a simple matter to change flux functions by replacing a single routine that computes the flux given the left and right interface states.

2.4.3.1 Roe's Flux-Difference Splitting: FDS

Roe's flux-difference splitting solves an approximate Riemann problem at the cell interface exactly. The left and right states at the interface are connected by a path in phase space that is composed of a set of discrete waves, with uniform states between. This then provides a means of forming the flux at the intermediate state yielding the upwinded flux. That is, if the flux is first linearized about the left state

$$\tilde{F}(U_L, U_R) = F(U_L) + \Delta F^- \tag{2.30}$$

and then the right

$$\tilde{F}(U_L, U_R) = F(U_R) - \Delta F^+ \tag{2.31}$$

then sum these and divide by two, the numerical flux is obtained as

$$\tilde{F} = \frac{1}{2}(F(U_L) + F(U_R)) - \frac{1}{2}(\Delta F^+ - \Delta F^-) \tag{2.32}$$

The flux differences are written in terms of the upwinded flux Jacobians, formed at an undetermined intermediate state, $\hat{U}$

$$\Delta F^+ = \hat{A}^+ \Delta U, \qquad \Delta F^- = \hat{A}^- \Delta U \tag{2.33}$$

Combining this into (2.32) yields the upwinded flux as

$$\tilde{F} = \frac{1}{2}(F_L + F_R) - \frac{1}{2}|\hat{A}|(U_R - U_L) \tag{2.34}$$

The dissipation matrix is formed at a state that is found by noting that the flux is a homogeneously quadratic function of a parameter vector which itself is linearly related to the conserved variables. This allows one to find the intermediate state, and form the dissipation matrix. In practice, the dissipation matrix is computed in terms of the eigenvalues and eigenvectors of the flux Jacobian evaluated at the intermediate state, and the change in the Riemann invariants about the intermediate state. That is, the flux is written

$$\tilde{F} = \frac{1}{2}(F_L + F_R) - \frac{1}{2}\sum_{i=1}^{4} |\hat{a}_i| \Delta \hat{V}_i \hat{R}_i \tag{2.35}$$

The intermediate state is formed as

$$\begin{aligned} \hat{\rho} &= \sqrt{\rho_L \rho_R} \\ \hat{u} &= u_L \omega + u_R (1-\omega) \\ \hat{v} &= v_L \omega + v_R (1-\omega) \\ \hat{H} &= H_L \omega + H_R (1-\omega) \end{aligned} \tag{2.36}$$

where

$$\omega = \frac{\sqrt{\rho_L}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{2.37}$$

The eigenvalues, $\hat{\lambda}_i$, at the intermediate state are

$$\hat{\lambda} = \begin{bmatrix} \hat{u}_c - \hat{a} \\ \hat{u}_c \\ \hat{u}_c \\ \hat{u}_c + \hat{a} \end{bmatrix} \tag{2.38}$$

while the change in the characteristic variables about the intermediate state, $\Delta \hat{V}_i$, are

$$\Delta\hat{V}_i = \begin{bmatrix} \dfrac{(\Delta\rho - \hat{\rho}\hat{a}\Delta u_c)}{2\hat{a}^2} \\ \dfrac{\hat{\rho}\Delta v_c}{\hat{a}} \\ \Delta\rho - \dfrac{\Delta P}{\hat{a}^2} \\ \dfrac{(\Delta\rho + \hat{\rho}\hat{a}\Delta u_c)}{2\hat{a}^2} \end{bmatrix} \tag{2.39}$$

with $\Delta(\ ) = (\ )_R - (\ )_L$. The acoustic eigenvalues, $\hat{\lambda}_{1,4}$ are modified to prevent expansion shocks, yielding the $\hat{a}_i$ as

$$\hat{a}_{2,3} = \hat{\lambda}_{2,3}$$

$$\hat{a}_{1,4} = \begin{cases} \left|\hat{\lambda}_{1,4}\right| & \left|\hat{\lambda}_{1,4}\right| \geq \frac{1}{2}\delta\hat{\lambda}_{1,4} \\ \dfrac{\hat{\lambda}^2_{1,4}}{\delta\hat{\lambda}_{1,4}} + \frac{1}{4}\delta\hat{\lambda}_{1,4} & \left|\hat{\lambda}_{1,4}\right| \leq \frac{1}{2}\delta\hat{\lambda}_{1,} \end{cases} \tag{2.40}$$

where

$$\delta_i = max\left(4\left(\lambda_{i,R} - \lambda_{i,L}\right), 0\right) \tag{2.41}$$

The columns of $\hat{\mathbf{R}}$ are the right eigenvectors of $\mathbf{A}$ and are formed at the intermediate state.

$$\hat{\boldsymbol{R}} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ \hat{u} - n_x\hat{a} & -n_y\hat{a} & \hat{u} & \hat{u} + n_x\hat{a} \\ \hat{v} - n_y\hat{a} & n_x\hat{a} & \hat{v} & \hat{v} + n_y\hat{a} \\ \hat{H} - \hat{u}_c\hat{a} & \hat{v}_c\hat{a} & \dfrac{(\hat{u}^2 + \hat{v}^2)}{2} & \hat{H} + \hat{u}_c\hat{a} \end{bmatrix} \tag{2.42}$$

This numerical flux has been proven to be robust, can capture strong, cell-aligned shocks

in one cell and, importantly, treats contact surfaces properly, yielding low dissipation across shear and boundary layers. The implementation of this flux for non-reacting, single-component gases is relatively straightforward, but, for reacting or non-reacting flows of multi-component gases the intermediate state is not unique, and is costly to compute [47]. Despite its few shortcomings, this numerical flux has proven to be quite useful, and is probably the most used upwind flux formula today.

2.4.3.2 Van Leer's Flux-Vector Splitting: FVS

The flux-vector splitting approach, pioneered by Van Leer, is based upon decomposing the numerical flux into upwind- and downwind-propagating components which are formed by the respective upwind states at the interface. That is

$$\tilde{F}(U_L, U_R) = F^+(U_L) + F^-(U_R) \tag{2.43}$$

The upwinded fluxes are formed by splitting the mass flux into forward and backward propagating components, giving rise to the name flux-vector splitting. As is shown in [80], there are various symmetry properties required of the split fluxes in subsonic regions and the requirement that the split fluxes smoothly join their unsplit fluxes at the sonic points. To ensure stability and positivity of the flux, the eigenvalues of the split flux Jacobians must have the proper signs, so that the eigenvalues of $\partial F^+ / \partial q$ are strictly positive, while the eigenvalues of $\partial F^- / \partial q$ are strictly negative. It is chosen to split the fluxes in subsonic regions in terms of a polynomial in Mach number, of which the lowest order gives rise to the well known splittings

$$\tilde{F}^{\pm} = \begin{bmatrix} F_m^{\pm} \\ F_m^{\pm} u \\ F_m^{\pm} v \\ F_m^{\pm} \left[H - m_e (u_c \mp a)^2\right] \end{bmatrix} + P^{\pm} \begin{bmatrix} 0 \\ n_x \\ n_y \\ 0 \end{bmatrix} \tag{2.44}$$

The split mass flux and pressure are formed as polynomials in Mach number of the contravariant velocity, $M_c = u_c / a$, as

$$F_m^{\pm} = \pm \frac{\rho a}{4} (M_c \pm 1)^2 \tag{2.45}$$

$$P^{\pm} = \frac{P}{4} (M_c \pm 1)^2 (2 \mp M_c) \tag{2.46}$$

Different forms of the splittings can be found, by choosing different m_e. The original formulation requires the bracketed terms in the energy flux be a perfect square in Mach number, but here $m_e = 0$ is chosen.

The advantages of flux-vector splitting are many, of which simplicity and robustness are its best virtues. The FVS approach is simple to implement, yields a smooth linearization (which is useful for implicit formulations of upwind schemes), and provides a straightforward means to extend the splitting to multi-component flows of real gases.

The source of robustness of the scheme is found by examining the difference in the split fluxes, which, when linearized, yields the dissipation matrix. It can be seen that as the Mach number approaches zero, the difference in split fluxes does not vanish, as in Roe's scheme. This property gives FVS its robustness, but also gives excess diffusion across grid-aligned shear and contact surfaces, where the transverse Mach numbers are low. It is precisely this excess diffusion which has caused the scheme to fall out of favor, as it adds too much diffusion in shear and boundary layers in a Navier-Stokes computation [83].

This is unfortunate, since the flux-vector splitting approach has been shown to be robust, efficient and easy to implement. Attempts have been made to improve the original

flux formula, yet stay within the philosophy of flux vector splitting, but it was shown that even for a first order flux, a low-diffusion modification of the original flux was non-positive [21]. Work in the area of improving the original FVS formulation by some hybrid scheme is important, and has led to the following new, and unproven method.

2.4.3.3 Liou's Advection Upstream Splitting Method: AUSM

This novel flux function combines the efficiency of flux-vector splitting with the accuracy of flux-difference splitting. As opposed to the original flux-vector splitting proposed by van Leer, where the mass flux is split into upwinded contributions at cell interfaces, the AUSM scheme constructs a velocity at the interface. Dependent upon the sign of this velocity, the convected components of the flux vector are taken from the proper side of the interface, hence the term Advection Upstream Splitting Method. The momentum flux through the interface due to pressure is treated separately. The inviscid flux is split into two contributions: convective and pressure

$$\tilde{\mathbf{F}} = u_{c_{1/2}} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho H \end{bmatrix}_{L/R} + P_{1/2} \begin{bmatrix} 0 \\ n_x \\ n_y \\ 0 \end{bmatrix} \tag{2.47}$$

where L/R means that the quantities are chosen from the left or right, depending upon the sign of the interface convection speed, $u_{c_{1/2}}$. The interface convection speed is found by splitting the Mach number as

$$u_{c_{1/2}} = a_{L/R} M_{1/2} \tag{2.48}$$

which leaves the split flux to be defined as

$$\tilde{\mathbf{F}} = M_{1/2}\begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a H \end{bmatrix}_{L/R} + P_{1/2}\begin{bmatrix} 0 \\ n_x \\ n_y \\ 0 \end{bmatrix} \tag{2.49}$$

where

$$M_{1/2} = M^{+}{}_{L} + M^{-}{}_{R}, \qquad P_{1/2} = P^{+}_{L} + P^{-}{}_{R} \tag{2.50}$$

There are various ways to split the Mach number and pressure: the simplest are the splittings proposed by Van Leer, which are the lowest order polynomials that satisfy certain continuity and positivity constraints.

$$M^{\mp} = \pm\frac{1}{4}(M \pm 1)^2, \qquad P^{\pm} = \frac{P}{4}(M \pm 1)^2(2 \mp M) \tag{2.51}$$

The final form of the flux formula may be written in a more standard way as

$$\tilde{\mathbf{F}} = M_{1/2}\frac{1}{2}\left(\begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a H \end{bmatrix}_{L} + \begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a H \end{bmatrix}_{R}\right) - \frac{1}{2}|M_{1/2}|\Delta\begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a H \end{bmatrix} + (P^{+} + P^{-})\begin{bmatrix} 0 \\ n_x \\ n_y \\ 0 \end{bmatrix} \tag{2.52}$$

This flux formula has been shown to have accuracy rivaling that of Roe's flux difference splitting, has a lower operation count, is straightforward to apply to gases of multicomponent species, and is not susceptible to the slow shock problem and carbuncle phenomenon as exhibited by Roe's scheme. Although it has many desirable qualities, it is only now beginning to be used extensively: Some anomalies are showing up in unsteady and steady calculations [10], prompting the schemes' author to improve it. This has only very recently resulted in a new and improved scheme, AUSM+ [49]. Due to the recent creation of the AUSM+ scheme, it is not included in the calculations to follow.

2.4.4 Evolution

For simplicity, the semi-discrete form of the equations, (2.29), are advanced in time using a multi-stage scheme. A spatially-varying time step is used, and is indeed quite necessary, since there is typically a many-order variation in cell size across the mesh due to cell refinement and cutting. A generic multi-stage scheme is used to advance the solution from the n-th to the (n+1)-th time level

$$\begin{aligned} \mathbf{q}^{(0)} &= \mathbf{q}^{n} \\ \mathbf{q}^{(m)} &= \mathbf{q}^{(0)} + \lambda_m \Delta t \mathbf{R}(\mathbf{q}^{(m-1)}), \qquad m = 1, mstages \\ \mathbf{q}^{n+1} &= \mathbf{q}^{(mstages)} \end{aligned} \tag{2.53}$$

Unless otherwise noted, a three stage scheme is used with coefficients of $\lambda_i = (0.18, 0.50, 1.0)$ and a Courant number of $\nu = 1.0$. This particular choice of number of stages and coefficients comes from [8], and in practice has proven to be best from an efficiency standpoint. The specific values of the multi-stage coefficients are nearly the same as those in [82], which were chosen to damp optimally a second-order upwind scheme, and are used in [8] for upwinded, unstructured-grid computations.

2.4.5 Boundary Procedures

Boundary conditions are applied by specifying the fluxes at boundary faces according to the type of boundary condition to be enforced. Slip boundary conditions are typically applied by extrapolating the pressure to the face Gauss point using either a first- or second-order reconstruction. Since the mass flux through the face is zero, the flux is then

$$F_{slip} = (0, n_x P_g, n_y P_g, 0)^T \tag{2.54}$$

where P_g is the pressure evaluated at the Gauss point. Supersonic inflow fluxes are completely specified from the imposed flow conditions, while supersonic outflow fluxes are

found by extrapolation of the primitives from the interior, and then computing the flux from them. For flows where the conditions are subsonic at the boundary faces, different types of procedures are used, dependent upon the particular flow process.

For an airfoil calculation, the far-field boundaries are typically located over 250 chord lengths away, so that the standard lifting far-field boundary procedure of the imposition of a point vortex, located at the quarter chord of the airfoil, is unnecessary. There, it is sufficient to use the approximate Riemann solver to compute the flux, given an extrapolated Left state from the interior and a Right state set from the free stream conditions.

For internal-flow computations, at inflow boundaries it has been found best to use a procedure where the flux at the interface is found by specifying total temperature, total pressure and flow angle from the inflow conditions, and extrapolate a backwards propagating Riemann invariant, R^-. Following the procedure outlined by Chima [17], where the Riemann invariant based on the total velocity is

$$R^- = U - \frac{2a}{(\gamma - 1)} \tag{2.55}$$

Writing this invariant in terms of the total speed of sound, and solving for U, the speed is found from the positive root of the resulting quadratic

$$U = \frac{R^-(\gamma - 1) + \sqrt{4a_0^2\left(\frac{\gamma + 1}{\gamma - 1}\right) - 2R^{-2}(\gamma - 1)}}{(\gamma - 1)} \tag{2.56}$$

From this, the sound speed is then found, yielding the pressure and temperature, while the Cartesian velocity components are found from the specified flow angle. For internal flows, the outflow-boundary fluxes are found by extrapolating the density and velocity components to the Gauss point, and specifying the pressure, yielding the flux.

2.5 Solution-Adaptive Mesh Refinement

The Cartesian, cell-based approach gains its strength primarily from two features; the ability to compute flows about complicated geometries where the initial grid is obtained automatically, and by the inherent ease in which adaptive mesh refinement can be performed. Adaptive mesh refinement is an attempt to improve the quality of a solution on a given grid by adding cells locally where an increased resolution is desired, and by possibly removing cells where the current resolution is unnecessarily too high. This feature, coupled with the automated means of mesh generation, attempts to yield grid-converged solutions about geometrically-complicated domains with minimal user intervention. Adaptive mesh refinement can also be viewed as a means of obtaining the most resolving power for a given amount of resources. This matter of minimizing the amount of resources used is crucial when performing three-dimensional simulations, but is not as important in two-dimensions. Flows with large localized variations in geometric and flow-induced length scales are amenable to adaptive mesh refinement, where resolution of all length scales with a uniform mesh size would be prohibitive.

Each level of adaptive mesh refinement is comprised of two stages. In the first stage, refinement criteria are constructed for all cells on the mesh, and then in the second stage, cells are tagged for refinement or coarsening based on these criteria. After the mesh is enriched, a new calculation is made, converging the solution to a steady, and hopefully more accurate, solution. This process of refining the grid and converging the solution on the new grid is repeated in an automated fashion, a set number of times, until a given level of refinement is achieved. The choice of refinement criteria, and the means in which to analyze these criteria over the mesh to determine which cells to refine, delineate the schemes in use today.

The refinement criteria and grading procedure used here are based upon those presented in [23][24] and [22]. In [87] it is shown that any refinement criterion should be

weighted with a local cell size, so that as cells get refined to smaller and smaller sizes, they are weighted less and less. If the criterion were not weighted in this way, strong flow features such as shock waves, whose locations can be dictated by weaker features in the flow, could dominate, and adaptive refinement might not improve the solution quality. The criteria used here are based wholly upon those presented by DeZeeuw and Powell in [22]; unless otherwise noted, no changes to the form of the refinement criteria or the selection levels are made.

The refinement and coarsening of the cells are based upon a statistical description of the cell-size-weighted velocity divergence and curl. The local velocity divergence is used to detect compressive phenomena, while the velocity curl is used to detect shear. Each of these is weighted by the local cell size so that smaller cells contribute less to the overall weighting.

$$\tau_C = |\nabla \bullet u| l^{3/2} \tag{2.57}$$

$$\tau_R = |\nabla \times u| l^{3/2} \tag{2.58}$$

The local cell size is taken to be the square root of the cell area and the 3/2 power is chosen to ensure that the criteria's importance diminishes with increasing refinement. These adaptive criterion are examined over the whole mesh, and cells are determined to be refined or coarsened according to the variance of each of these criteria about zero. That is, let

$$\sigma_C = \sqrt{\frac{\sum_{n=1}^{N} \tau_C^2}{N}} \tag{2.59}$$

$$\sigma_R = \sqrt{\frac{\sum_{n=1}^{N} \tau_R^2}{N}} \tag{2.60}$$

Cells are tagged for refinement or coarsening according to these criteria. Cells are refined if

$$(\tau_C > \sigma_C \quad or \quad \tau_R > \sigma_R) \quad and \quad l > l_{min} \tag{2.61}$$

and cells are coarsened if

$$\tau_C < \frac{\sigma_C}{10} \quad and \quad \tau_R < \frac{\sigma_R}{10} \tag{2.62}$$

The constants can be adjusted in (2.61) and (2.62), as is stated in [22], to tune the criteria for different flow fields. In the work here, unless otherwise noted, the refinement criteria used are exactly as presented here.

2.6 Validation of the Approach

The Cartesian, cell-based approach for solving the Euler equations is first validated against some well known inviscid flow fields. Two test cases are chosen from a collection of test cases for inviscid flows [2]; transonic flow over the ubiquitous NACA 0012 and transonic flow over the RAE 2822 airfoil. Both cases illustrate the automated grid generation and adaptive mesh-refinement of the Cartesian, cell-based approach, and provide confidence in the overall method.

2.6.1 AGARD Test Case 2: NACA 0012, $M_\infty = 0.85$, $\alpha = 1°$

This corresponds to the same geometry and free stream conditions as Test Case 02 in the collection of inviscid flow test cases, AR-211 [2]. This flow contains a strong shock on the upper and lower surfaces, and is a good test of the shock capturing ability of the flow

solver, and by examining the mesh, whether the adaptive mesh-refinement procedure refines about the important features of the flow.Three levels of adaptive mesh refinement are made beyond a coarse, base grid. A close-up of the final, adapted grid and flow field are shown in Figure 2.12 and Figure 2.13.

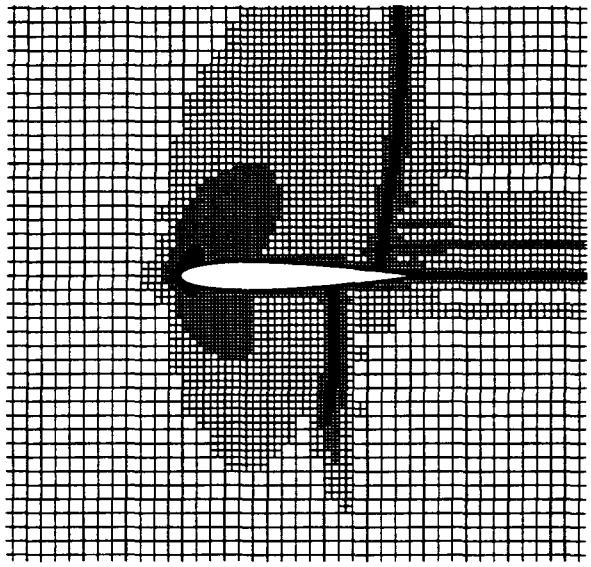


Figure 2.8 Refinement Level 3 Adapted Grid

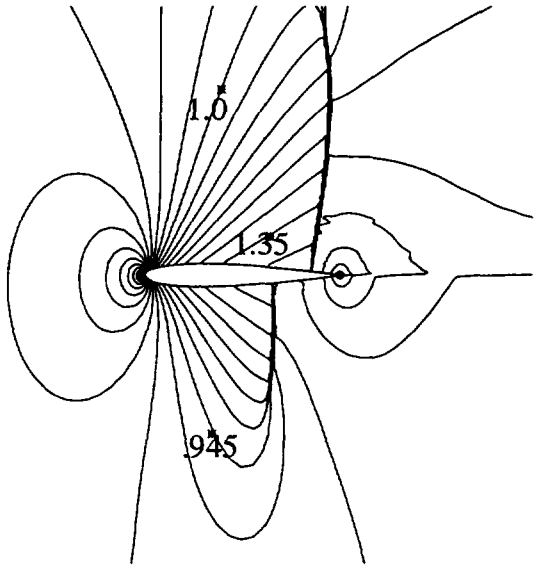


Figure 2.9 Refinement Level 3 Mach Number Contours: Min=0.003, Max=1.43, Increment=0.05.

The computed surface pressure coefficients and Mach numbers are compared with the results tabulated in [2]. There, the tabulated results were computed on a 320 by 64 structured mesh, using a central differencing scheme with explicitly added dissipation, with 320 points on the airfoil surface. Figure 2.14 and Figure 2.15 compare the surface Mach number and pressure coefficients for the adaptively-refined Cartesian, cell-based approach and the structured grid results in the AGARD test case. The final, adapted grid has 13251 cells, which is approximately 64% as many as the structured grid. Although there is little difference between the two cases, one can say that the adaptively-refined results show a sharper shock wave, although both do a good job for this flow.

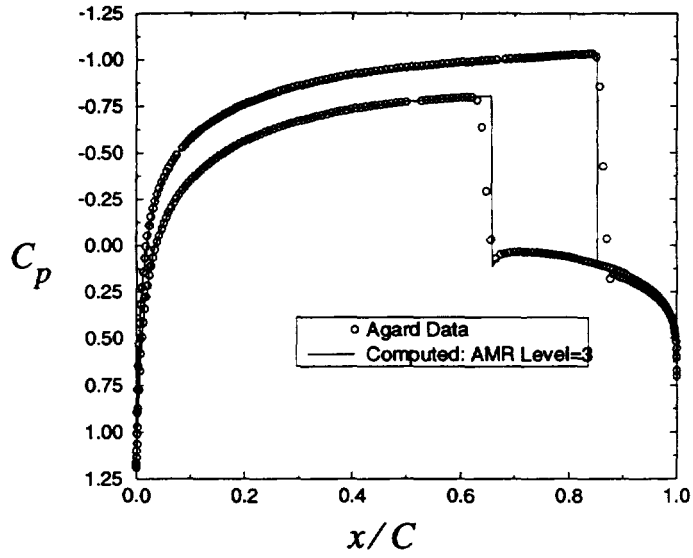


Figure 2.10 Comparison of Surface Pressure Coefficient to AGARD Data

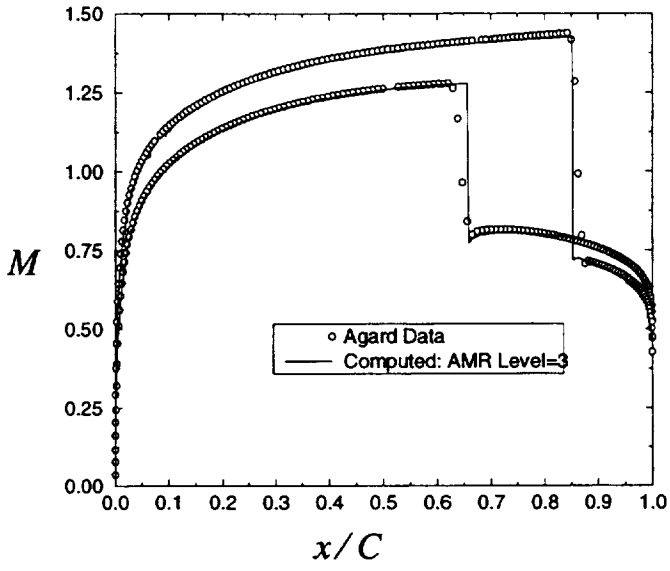


Figure 2.11 Comparison of Surface Mach Number to AGARD Data

2.6.2 AGARD Test Case 6: RAE 2822, $M_\infty = 0.75$, $\alpha = 3°$

This case corresponds to the same geometry and free-stream conditions as Test Case 06 in the collection of inviscid flow test cases [2]. Adaptive mesh refinement is performed for four levels beyond the base grid level. Figure 2.16 shows the final, adapted grid and Figure 2.17 shows the Mach number contours at the final refinement level.

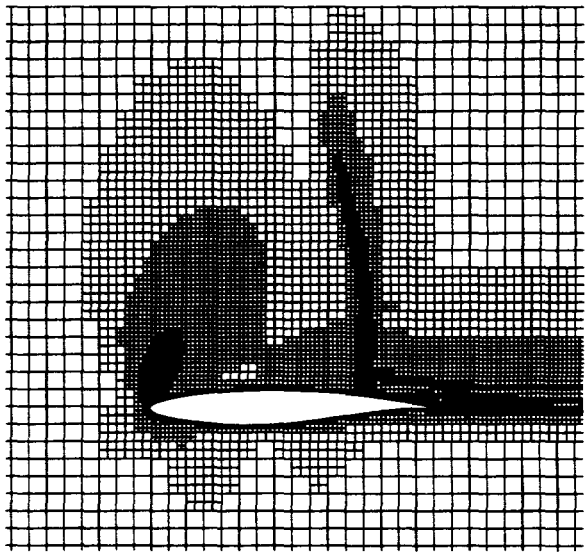


Figure 2.12 RAE 2822 Level 4 Adapted Grid

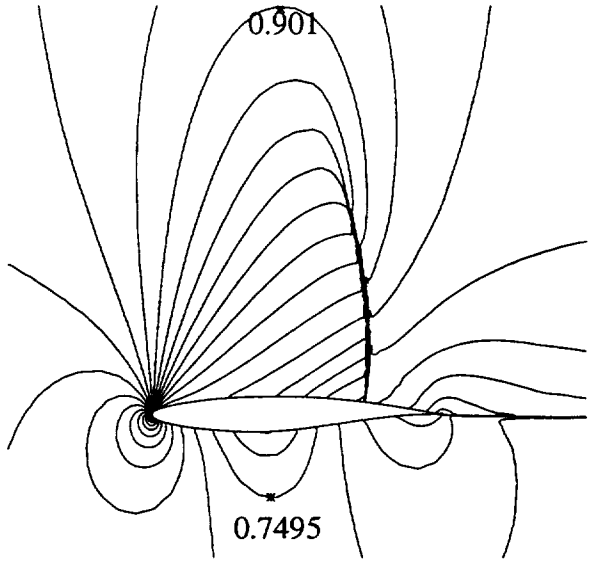


Figure 2.13 RAE 2822 Level 4 Adapted Mach Number Contours: Min=0.004, Max=1.759, Increment=0.050

The solution compares well to the results tabulated in [2], where the same contributors for the AGARD Case 02 computed the solution on a structured grid of 320 by 64 cells, using a central differencing scheme with explicitly added dissipation. The adaptively-refined Cartesian-mesh approach captures the upper surface shock crisply, and gives overall excellent results.

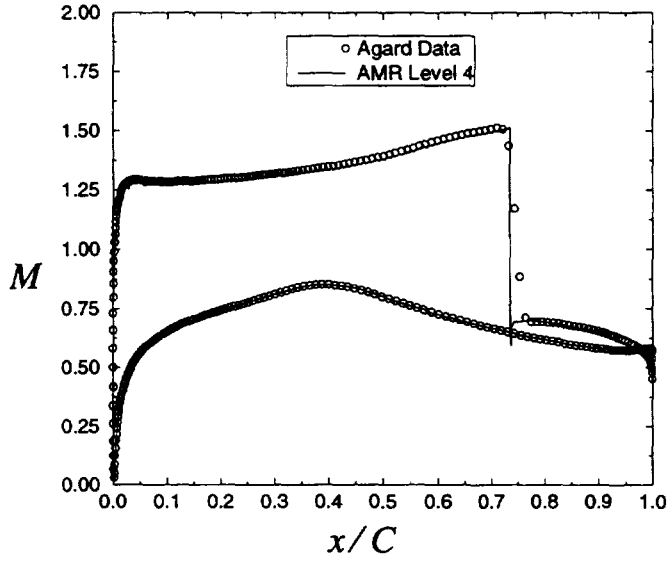


Figure 2.14 Comparison to Computed Surface Mach Numbers from AGARD 211

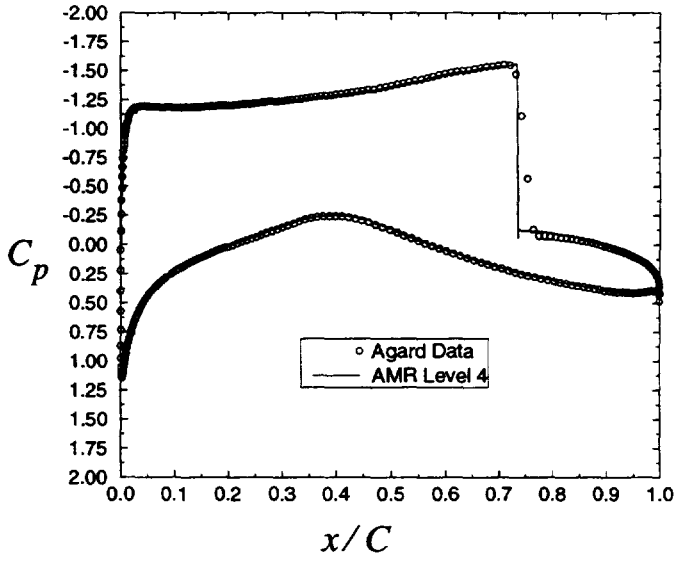


Figure 2.15 Comparison to Computed Surface Pressure Coefficients from AGARD 211

The effect of adaptive mesh refinement upon the computed lift and drag of the airfoil is shown in Figure 2.20. Here, the computed lift and drag coefficients are plotted against the number of cells in the grid from the base grid level to the final, fourth level of adapted mesh refinement.

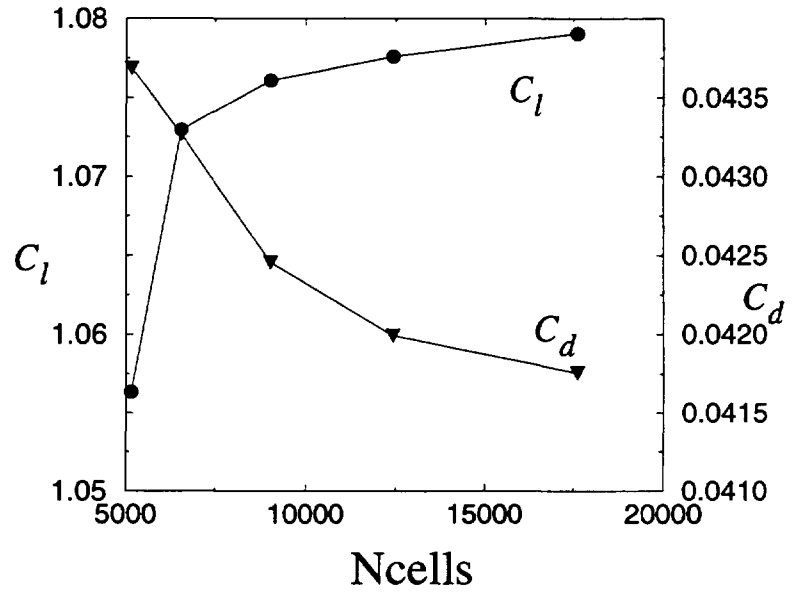


Figure 2.16 Grid Refinement of Lift and Drag Coefficients

As is seen in the figures, the computed lift and drag are approaching grid convergence, yet have not appeared to be completely grid converged, even though the final grid level has nearly 18000 cells. This smooth behavior of approaching a grid-converged solution comes about due to the care taken in performing the calculation. The body is represented by a series of cubic splines, so that by construction, the geometry is C_1 at all of the control points, except the trailing edge. It is necessary to use this geometric description, as opposed to a linearly-faceted body, where a linear function would be used to provide geometry between the geometric control points. If a linear description is used, grid refinement will often resolve the discontinuous breaks on the body surface, especially in regions of higher Mach number, such as near the suction peak on the airfoil. These geometric breaks may be un-resolved at the lower grid refinement levels, only to appear as refinement proceeds. Until all of the facets are uncovered, grid refinement can not be achieved.

In addition to care in defining the body, it is necessary to run a large number of iterations, to converge the flow in the trailing edge region of the airfoil. The trailing edge region is where the circulation about the body is set, and until the flow is resolved well and converged well there, grid convergence will not be achieved.

2.7 Accuracy Assessment of the Approach

Now that the Cartesian-cell-based approach has been demonstrated for a few airfoils, and shown to give reasonable results, it is assessed more carefully to examine the accuracy of the scheme. An exact solution of the Euler equations (Ringleb's flow) is used not only to infer the order of error of the Cartesian-mesh approach but also to compare the magnitude of the error directly to that obtained with a structured mesh approach. Care is taken in the formulation and implementation of both schemes, so that a meaningful comparison results.

2.7.1 A Hodograph Solution to the Euler Equations: Ringleb's Flow

Ringleb's flow is a hodograph solution to the Euler equations [2], and has been used to assess the accuracy of other structured- and unstructured-mesh approaches [4][29][33]. A variety of flows can be attained, depending upon the choice of parameters used. The solution is parameterized in terms of the total velocity, q, and streamline constant, k as

$$x(q,k) = \frac{1}{2\rho}\left(\frac{2}{k^2} - \frac{1}{q^2}\right) - \frac{J}{2} \tag{2.63}$$

$$y(q,k) = \pm\frac{1}{k\rho q}\sqrt{1-\left(\frac{q}{k}\right)^2} \tag{2.64}$$

$$c = \sqrt{1 - \frac{(\gamma-1)}{2}q^2} \tag{2.65}$$

$$\rho = c^{\frac{2}{\gamma - 1}} \tag{2.66}$$

$$J = \frac{1}{c} + \frac{1}{3c^3} + \frac{1}{5c^5} - \frac{1}{2}\log\left(\frac{1+c}{1-c}\right) \tag{2.67}$$

where the density, ρ, is made non-dimensional by its stagnation value and all speeds are made non-dimensional by the stagnation sound speed. The flow angle θ is related to the streamline constant and total velocity by

$$\theta = 2\pi - \text{asin}\left(\frac{q}{k}\right) \tag{2.68}$$

The flow in the first quadrant is computed that is bounded by the streamlines $k = 0.75$ and $k = 1.5$. The outflow boundary is situated along the $y = 0$ line of symmetry and the inflow boundary is along the iso-velocity line of $q = 0.5$. The resulting flow has a sub-sonic inflow and a mixed supersonic/subsonic outflow. Figure 2.21 shows contours of the Mach number of the flow field obtained with these parameters. As can be seen from the figure, the flow can be visualized as a transonic, accelerating flow contained between two curved streamlines.

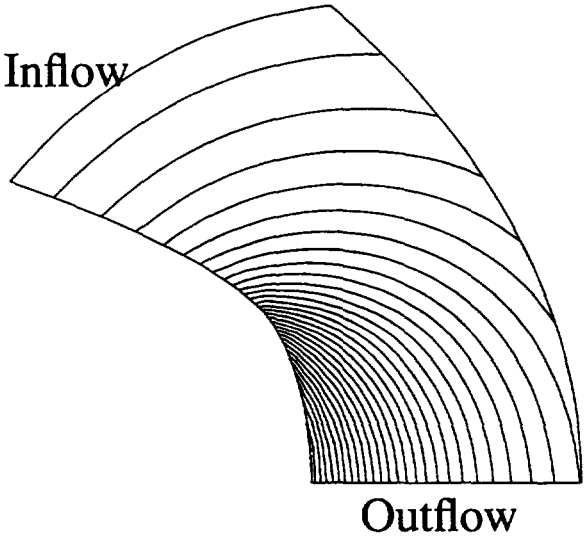


Figure 2.17 Mach Contours, Ringleb's Flow

If one defines the discrete error for the i-th cell as

$$e_i = |\bar{\rho}_i - \rho_{exact}(\bar{x}_i, \bar{y}_i)| \tag{2.69}$$

then the L_p norms of this error are

$$L_p = \left(\frac{\sum e_i^P}{N} \right)^{\frac{1}{P}} \tag{2.70}$$

First, an assessment of the order of accuracy and the magnitude of the error using the Cartesian-mesh approach is made. The order is inferred by evaluating the behavior of the error norms with increasing mesh refinement, while the magnitude of the error is assessed by comparing the error norms directly with those obtained from a structured mesh solver.

2.7.2 Structured Solver Formulation and Results

The structured-grid flow solver uses Fromm's differencing of the primitive variables on a coordinate-by-coordinate basis. Roe's linearized Riemann solver is used to compute the fluxes through the cell interfaces. Care is taken in the formulation of the boundary proce-

dures so that a direct comparison of the two codes yields meaningful results. Slip boundary conditions are applied by extrapolating the pressure to the Gauss points in a manner consistent with the interior scheme. At the subsonic inflow, a boundary procedure based upon constant total conditions and an extrapolated Riemann invariant (as in [17]) is used. Roe's approximate Riemann solver is used at the mixed supersonic/subsonic outflow boundary. The left and right states are supplied to the flux function from extrapolated and exact conditions evaluated at the Gauss points.

The meshes used for the structured grid calculations have a family of coordinate lines lying along the exact solution streamlines. The other coordinate-line family was generated using a sinusoidal blending of the streamline and iso-velocity constants. A sample structured mesh is shown in Figure 2.22 with 400 cells. A sequence of successively finer meshes of 10x10, 20x20, 40x40 and 80x80 cells were used to compute Ringleb's flow, upon which the solution error norms were computed. The norms are tabulated in Table I.

Table I Structured Grid Error Norms for Ringleb's Flow

Ncells	L_1	L_2	L_∞
100	2.368e-03	2.796e-03	1.066e-02
400	4.517e-04	5.352e-04	2.700e-03
1600	1.157e-04	1.337e-04	6.918e-04
6400	3.050e-05	3.514e-05	1.745e-04

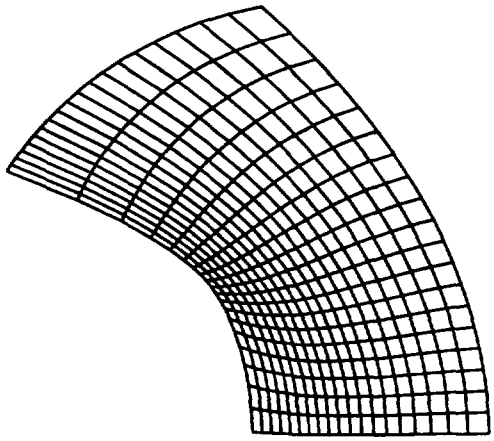


Figure 2.18 Sample Structured Grid for Ringleb's Flow: 20 x 20 Grid

By plotting the logarithm of the norms against the logarithm of the characteristic cell size, $1/\sqrt{N}$, one can infer the order of the truncation error form the slope of the plot. A least-squares curve fit of the data gives slopes of the L_1, L_2 and L_∞ norms of 2.08, 2.09 and 1.97 respectively, indicating that the structured scheme is uniformly 2nd order accurate. It should be noted that the mesh used here is quite beneficial: with a closer examination, one can see that by virtue of the mesh generation, not only is one family of mesh lines aligned exactly with the flow streamlines, but clustering is achieved near the place of minimum radius of curvature of the left wall. Indeed, in a later section, it is shown that the adaptively- refined mesh is clustered in this region.

2.7.3 Cartesian-Mesh Results for Uniform Mesh Refinement

Next, the Cartesian-mesh approach is used to compute Ringleb's flow on a sequence of successively finer uniform Cartesian meshes. The uniform meshes are generated by recursively refining a set number of levels below the root of the tree, and then cutting the geometry out of the mesh. The number of levels below the root cell characterizes the fineness of the uniform meshes, which is referred to as the mesh base level, L_0. Figure 2.23 shows

the coarsest mesh, which is 4 levels below the root cell, hence at a mesh base level of $L_0 = 4$. Uniformly refined calculations were made for base grid levels of 4, 5, 6 and 7, of which the error norms are tabulated in Table II.

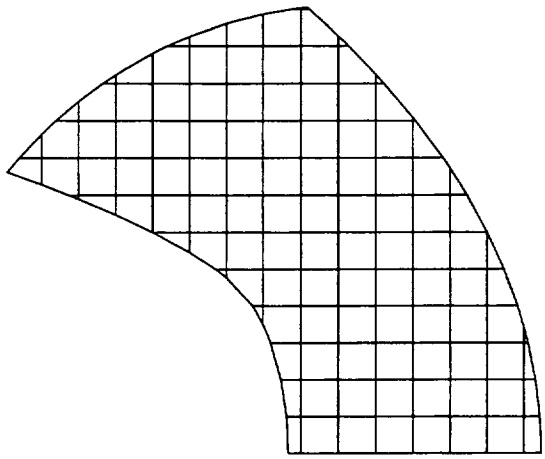


Figure 2.19 Uniform Cartesian Mesh, $L_0 = 4$

Table II Uniformly Refined Error Norms for Cartesian-Mesh Approach, Ringleb's Flow

N	L_1	L_2	L_∞
118	5.345e-03	8.658e-03	4.497e-02
417	1.571e-03	2.554e-03	1.458e-02
1578	4.236e-04	7.394e-04	6.928e-03
6134	9.793e-05	1.983e-04	2.674e-03

A least-squares curve fit of the uniformly refined norm data yields slopes for the L_1, L_2 and L_∞ norms of 2.02, 1.91 and 1.40, respectively. Using the two finest meshes one obtains slopes for the L_1, L_2 and L_∞ norms of 2.16, 1.94 and 1.40, respectively. These slopes indicate that the Cartesian-cell based scheme is globally 2nd order accurate and that the local error is between first- and second-order. An analysis of the effect of the boundary

cells upon the solution accuracy is estimated by computing error norms separately for the boundary cells, and then computing the slopes as above. The computed slopes of the boundary cell L_1, L_2 and L_∞ norms were 1.68, 1.49 and 1.40. Although the local error is degraded by the irregularity in the mesh due to the cut cells/boundaries, the scheme remains globally second-order accurate. It is precisely this behavior which is the saving grace of the Cartesian approach.

2.7.4 Cartesian-Mesh Results for Adaptive Mesh Refinement

The effect of adaptive refinement is assessed next. Beginning at a base uniform mesh of level $L_0 = 4$, adaptation proceeds through 4 levels of refinement. The refinement is made according to the rotationality and compressibility parameters described above, although for this irrotational flow, the rotationality parameter is nearly zero and does not effect the refinement topology. Figure 2.24 shows the adapted mesh that corresponds to a mesh refinement of 2 levels below the base, uniform mesh.

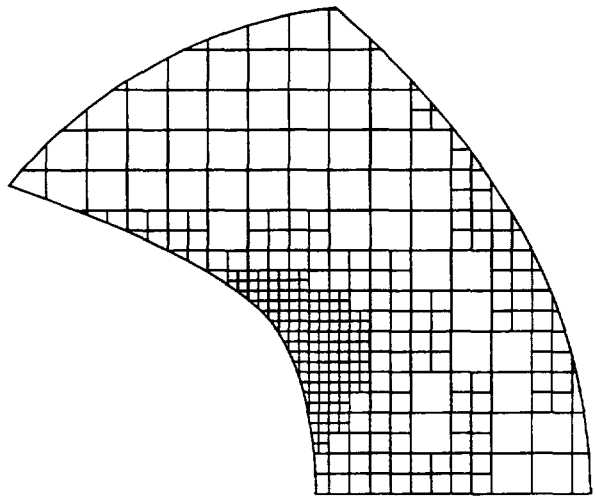


Figure 2.20 Adaptively Refined Cartesian Mesh, AMR Level 2

The adaptively refined norms are tabulated in Table III.

Table III Adaptively Refined Error Norms for the Cartesian-Mesh Approach, Ringleb's Flow

N	L_1	L_2	L_∞
118	5.345e-03	8.658e-03	4.497e-02
165	2.691e-03	3.995e-03	2.146e-02
337	1.310e-03	1.693e-03	6.658e-03
754	5.570e-04	7.093e-04	2.846e-03
1846	2.263e-04	3.056e-04	1.599e-03

2.7.5 Accuracy Assessment: Comparison and Discussion of Results

The error norms are compared with the characteristic cell size in Figure 2.20, Figure 2.26 and Figure 2.25 for the structured, uniformly and adaptively refined Cartesian calculations. As is shown in the figures, the L_1 and L_2 norms continue to behave in a 2nd order accurate fashion throughout the refinement, and the L_∞ norm is appreciably reduced in the beginning stages of the refinement process.

These figures also indicate that the adaptive refinement would require approximately twice the number of cells than the structured solver for a given error magnitude. What is also indicated is that the adaptive mesh refinement requires approximately 2/3 the number of cells for a given error norm than the uniformly refined (un-adapted) procedure.

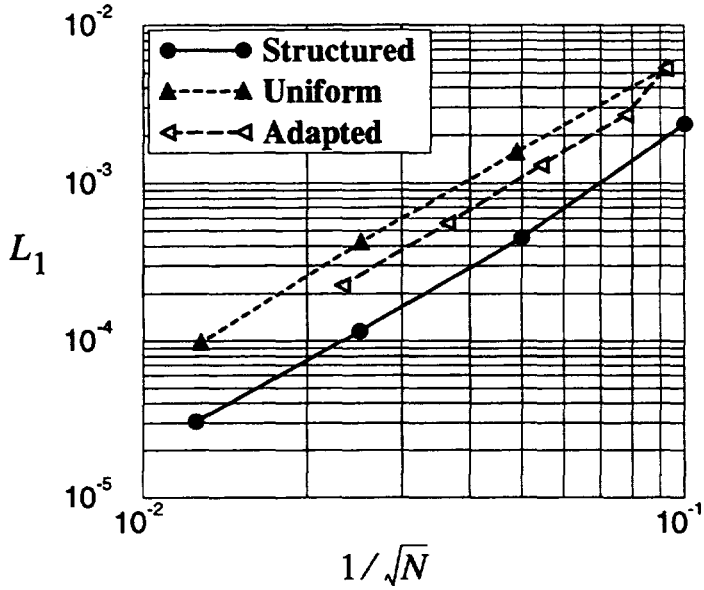


Figure 2.21 L_1 Norms of the Error

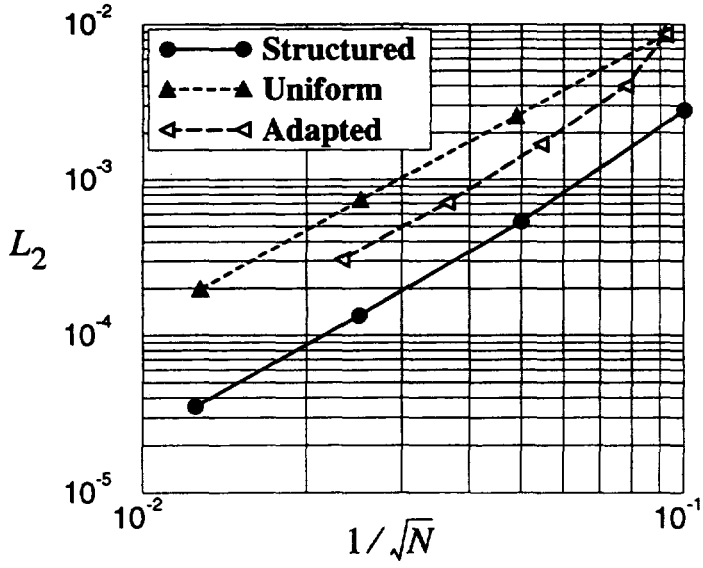


Figure 2.22 L_2 Norms of the Error

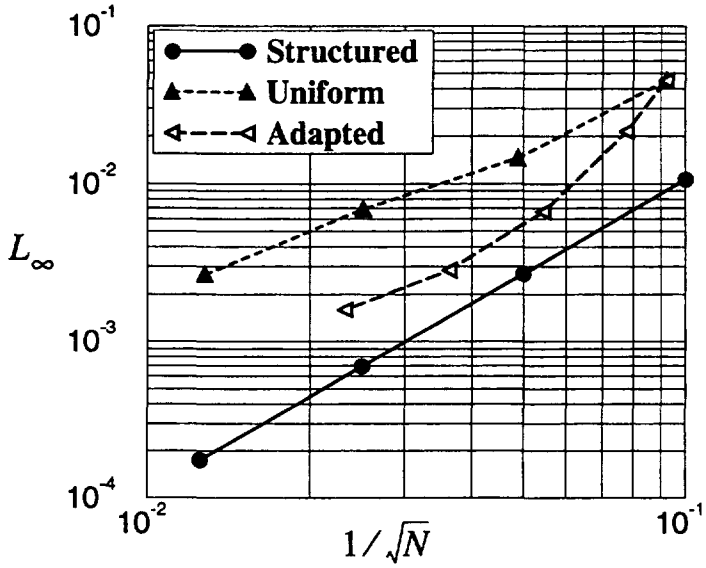


Figure 2.23 L_∞ Norms of the Error

To determine if the refinement strategy could be improved, the parameters that determine cell refinement and coarsening in (2.61) and (2.62) were adjusted. First, to see if the length-scale weighting of the refinement criterion was not tuned properly, the length-scale weight powers were changed to 1 and to 2, but the effect was negligible. In addition, the cutoff parameters for coarsening and refining were adjusted. Cells were refined for refinement parameters greater than 1/2 and 3/2 times the standard deviation about zero, with no appreciable effect. Cells were coarsened for refinement parameters less than 1/4 and 1/2 the standard deviation about zero (the default level is 1/10), also to no effect. These results indicate that the refinement procedure is tuned properly for this smooth flow, and that no appreciable gains could be made with respect to the structured results.

The relatively poor performance of the Cartesian solver with respect to the structured solver is best explained by the smoothness of the flow and by the flow alignment of the structured grid. Ringleb's flow is a very smooth flow and has essentially a single length scale; it is surmised that on the structured grid, even a very coarse grid captures enough of

the flow field so that refinement beyond this saturation will not yield much improvement over uniform refinement. In addition, the structured mesh used can be viewed as being in some sense "optimal"; not only is one family of the coordinate lines exactly aligned with the streamlines of the flow, but by virtue of the mesh generation, the mesh is denser in the region where the gradients are higher. Indeed, the grid alignment with the solution streamlines can be very beneficial since the Riemann solver used is only one-dimensional.

2.7.6 Accuracy Assessment: Non-Smooth Flow

One of the great promises of adaptive mesh refinement is to achieve a high level of resolution and accuracy using a minimum of resources. But, as is shown in the preceding example, for a flow with a single length scale, or one that is fairly uniform, it is hard to beat uniform mesh refinement. The following example shows that for the proper type of flow field, adaptive mesh refinement can give an appreciable gain in performance over uniform mesh refinement. For this study, the supersonic flow through a stylized axi-symmetric inlet is computed using the Cartesian-cell approach on a sequence of uniformly- and adaptively-refined meshes. The inlet studied is based upon the mixed compression inlet investigated in [34]. This inlet is designed to decelerate a $M_\infty = 2.5$ flow through a series of oblique shock waves which terminate at a normal shock wave in the diffuser. For the study here, the free-stream Mach number is reduced to $M_\infty = 2.0$ and the cowl is displaced radially outward one-half cowl radius from the centerline. Extrapolation-type procedures are applied at the exit boundaries, yielding a supersonic flow throughout the inlet. The centerbody and inner cowl surface shapes are made using the curve fits supplied in [34], while the outer cowl surface is stylized for this study. Uniformly-refined computations were made for base grid levels $L_0 = 6$ through $L_0 = 9$. Adaptively-refined calculations were made starting at a grid base level $L_0 = 6$ and refining 5 levels. Figure 2.28 and Figure 2.29 show computed pressure contours and the resulting adapted grid corresponding to three refinement levels below the base mesh. Grid convergence is assessed by comparing

the drag coefficients of the uniformly and adaptively refined calculations, shown in Figure 2.30.

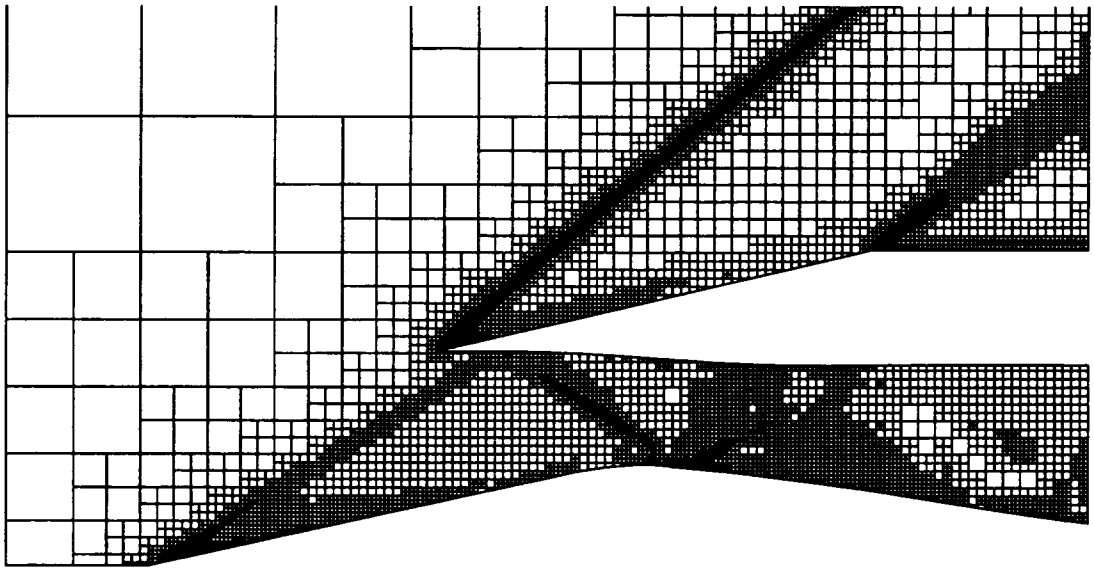


Figure 2.24 Adapted Grid: Axi-Symmetric Inlet, AMR Level 3

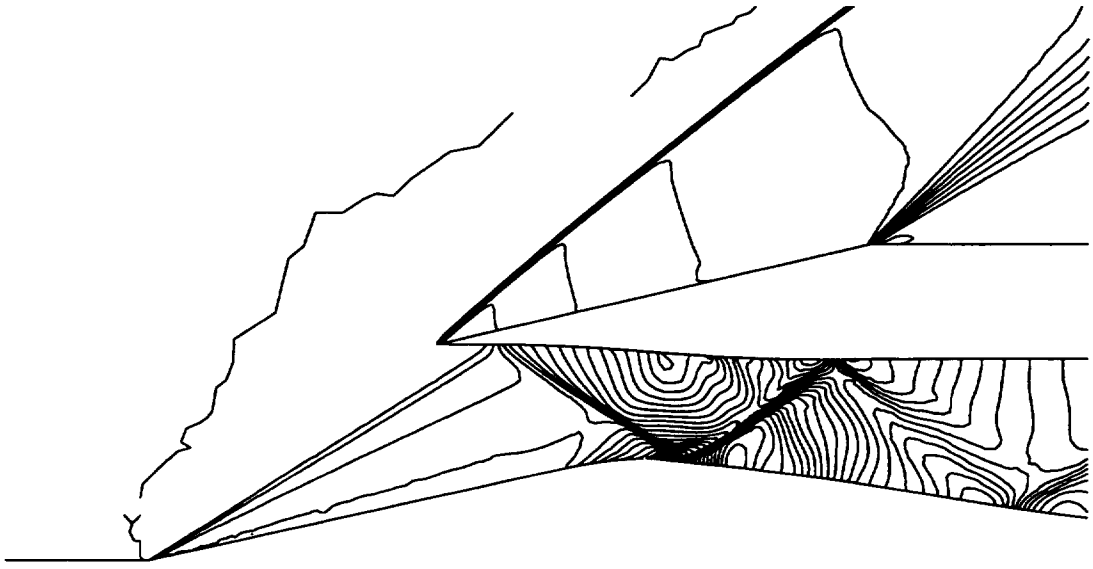


Figure 2.25 Adapted Grid: Pressure Contours, AMR Level 3

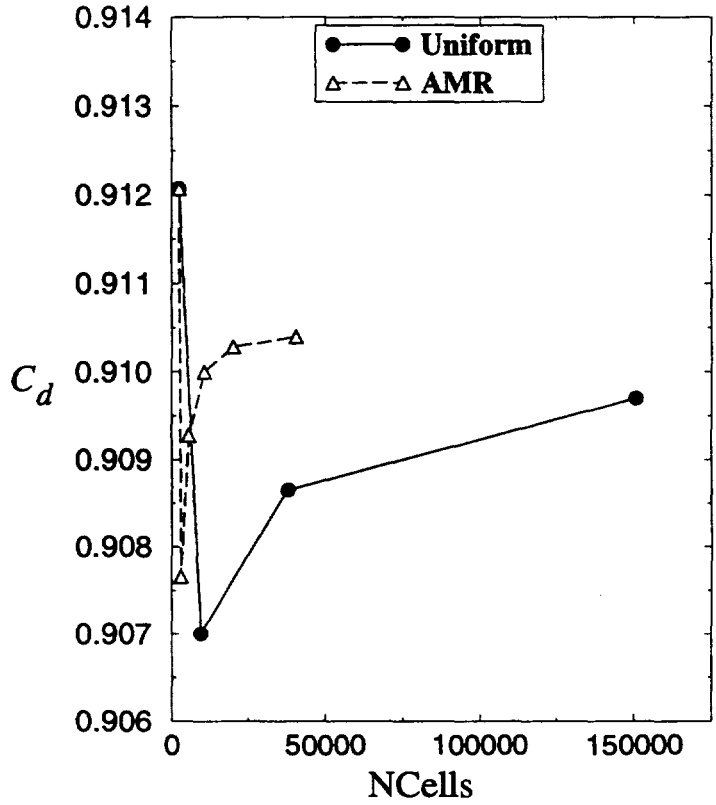


Figure 2.26 Mesh Convergence of Drag Coefficients: Uniformly and Adaptively Refined Grids

At this point, an accurate Euler solver based upon the Cartesian-mesh approach has been developed and proven; effort now turns to investigating the efficacy of the Cartesian-mesh approach for solving viscous flows.

CHAPTER III
An Accuracy and Positivity Analysis of Existing Cell-Centered Viscous Flux Formulae

The compressible Navier-Stokes equations are directly obtained in conservation-law form from first principles. Applying the concepts of conservation to an arbitrary conservation volume gives the Navier-Stokes equations

$$\frac{\partial}{\partial t}\int_{Vol} q_i dV + \oint_S E_{ij}\hat{n}_j dS = \oint_S G_{ij}\hat{n}_j dS + \int_{Vol} H_i dV \tag{3.1}$$

The inviscid flux tensor, E_{ij}, is as defined in (2.20), and volumetric forces that may be acting upon the fluid, such as gravitational or buoyancy forces, are accounted for in H_i. The viscous flux tensor G_{ij}, contains the contribution of the forces acting upon the control volume boundaries to the momentum and energy state in the volume. The force per unit area (stress) in the i-th direction on a face of the control volume whose normal is $\hat{n}_j$, is described by the Cartesian stress tensor

$$\sigma_{ij} = -P\delta_{ij} + \tau_{ij} \tag{3.2}$$

The flux of momentum into the conservation volume from the random, kinetic motion of the gas is the hydrostatic pressure, and is treated separately in (3.2), while the flux of momentum due to the shear and dilitational deformation of the fluid is represented by the shear stress tensor, τ_{ij}. Flows of air are considered here, where the assumption of a Newtonian fluid is valid, so that the stress on the fluid elements is linearly related to the rate of strain of the fluid. By assuming isotropy in the shear stress tensor and in the linear relation between the stress and rate of strain, the shear stress tensor may be written as

$$\tau_{ij} = \mu (u_{i,j} + u_{j,i}) + \lambda \delta_{ij} u_{k,k} \tag{3.3}$$

The constants, μ and λ, are related by requiring the mean normal stress (1/3 of the trace of (3.2)), to be only due to the hydrostatic pressure (Stokes' hypothesis), so that $\lambda = -\frac{2}{3}\mu$. The shear stress tensor is then written as

$$\tau_{ij} = \mu (u_{i,j} + u_{j,i}) - \frac{2}{3}\mu \delta_{ij} u_{k,k} \tag{3.4}$$

In addition to the convection of energy into the control volume, shear-deformation work on the control volume and heat conduction within the fluid is occurring. By assuming the heat conduction to be modelled by Fourier's law, $q_i = -kT_{,i}$, the viscous flux tensor may be written in two dimensions as

$$G_{1j} = [0, \quad \tau_{xx}, \quad \tau_{xy}, \quad u\tau_{xx} + v\tau_{xy} - q_x]^T \tag{3.5}$$

$$G_{2j} = [0, \quad \tau_{yx}, \quad \tau_{yy}, \quad u\tau_{yx} + v\tau_{yy} - q_y]^T \tag{3.6}$$

The shear stress tensor and heat flux vector are then

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \tag{3.7}$$

$$\tau_{xy} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{3.8}$$

$$\tau_{yx} = \tau_{xy} \tag{3.9}$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \tag{3.10}$$

$$q_x = -k \frac{\partial T}{\partial x} \tag{3.11}$$

$$q_y = -k \frac{\partial T}{\partial y} \tag{3.12}$$

In practice, the equations are made non-dimensional by a suitable reference state. Here,

the reference conditions are taken as:

Length: l_∞

Density: ρ_∞

Speed: a_∞

Pressure: $\rho_\infty a_\infty^2$

Time: l_∞ / a_∞

The constitutive relations assume a calorically perfect equation of state, and a laminar viscosity based on Sutherland's law. The definition of the fluid Prandtl number is used to find the thermal conductivity from the laminar viscosity and specific heat at constant pressure.

$$\mu = \mu_0 \frac{T^{\frac{3}{2}}}{T+S}, \qquad S = 110\text{K}^\circ, \qquad \mu_0 = 1.45\times10^{-6}\left[\frac{kg}{m-\sec}\right] \tag{3.13}$$

$$Pr = \frac{\mu C_p}{k}, \qquad Pr = 0.72 \tag{3.14}$$

Substitution of these reference conditions into (3.1) specialized to having no volumetric forces, and a conservation volume whose shape is invariant in time, gives the form of the compressible Navier-Stokes equations used here:

$$A\frac{\partial \bar{q}_i}{\partial t} = -\oint_A E_{ij}\hat{n}_j dA + \frac{1}{Re_\infty}\oint_A G_{ij}\hat{n}_j dA \tag{3.15}$$

The functional form of the shear stress tensor (3.7) is unchanged and the heat fluxes use the Prandtl number definition in (3.14), as

$$q_x = -\frac{\mu}{Pr(\gamma-1)}\frac{\partial T}{\partial x} \tag{3.16}$$

$$q_y = -\frac{\mu}{Pr(\gamma-1)}\frac{\partial T}{\partial y} \tag{3.17}$$

The Reynolds number in (3.15) is based upon the reference conditions which make the equations non-dimensional as

$$Re_\infty = \frac{\rho_\infty a_\infty l_\infty}{\mu_\infty} \tag{3.18}$$

The solution of the conservation law (3.15) follows that of all finite-volume approaches: the surface integrals on the right hand side are replaced by a numerical quadrature. Since the convective terms are expected to be second-order accurate, the viscous flux terms are integrated to the same order using the same type of quadrature. This entails evaluating the kernel of the integral at the midpoints of the cell to cell interfaces. The shear stresses and heat fluxes are functions of the local thermodynamic state (to find the laminar viscosity and thermal conductivity from the constitutive relations) and the Cartesian gradient of the velocity vector and temperature field. Since the local state can be found from the reconstructed solutions at the cell centroids, the essence of computing the viscous fluxes is finding the gradients of the velocity vector and the temperature at the midpoints of the cell edges.

First, an assessment will be made of existing viscous flux formulae that have been used in a selection of structured and unstructured finite-volume schemes. This assessment will be made using a model equation that represents the viscous terms of the Navier-Stokes equations, and will investigate the accuracy and positivity of the existing schemes upon mesh topologies that will regularly occur in a Cartesian-mesh calculation. From this assessment, two schemes emerge as best choices for solving the Navier-Stokes equations on unstructured meshes. The best scheme should maintain high accuracy as well as posi-

tivity of the operator, but for the present state of affairs, these two properties appear to be in direct competition. The performance of these two schemes is demonstrated for a selection of test problems in chapter IV, while the analytical assessment of the candidate schemes begins here.

3.1 An Analysis of Existing Viscous Flux Formulae for the Navier-Stokes Equations

Here an assessment is made of some existing viscous flux formulae that are in use today in structured and unstructured, cell-centered finite-volume Navier-Stokes solvers. The assessment is made by analyzing the solution of Laplace's equation, since for an incompressible viscous flow with constant laminar viscosity, the viscous terms in the momentum equations are identically Laplace operators acting upon the velocity field. For each of the viscous flux formulae, the accuracy is examined in the form of a local Taylor series expansion about the cell centroid for three different grid types: uniform Cartesian, East Face refined Cartesian and uni-directionally stretched Cartesian (see Figure 3.1, Figure 3.2 and Figure 3.3). Positivity properties of the operator are then found by examining the coefficients of the cells used in the local Laplacian.

3.1.1 Model Grid Topologies and Formulae

The finite-volume formulae for the discrete Laplacians on the model grid topologies are shown here. For the uniform Cartesian grid shown in Figure 3.1, the Laplace operator reduces to

$$L(u_0) = \frac{1}{h}(u_{x,E} - u_{x,W} + u_{y,N} - u_{y,S}) \tag{3.19}$$

where $u_{x,E}$, $u_{x,W}$, $u_{y,N}$ and $u_{y,S}$ are the reconstructed gradients at the East, West, North

and South faces respectively.

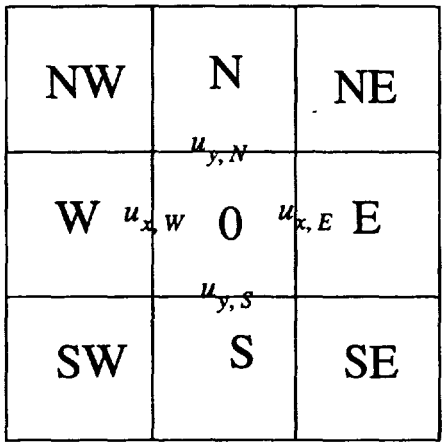


Figure 3.1 Uniform Cartesian Grid: Nomenclature

For the East-face refined Cartesian grid, the operator is

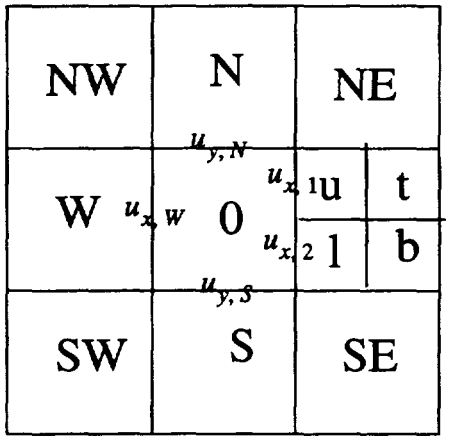


Figure 3.2 East-Face Refined Cartesian Grid: Nomenclature

$$L(u_0) = \frac{1}{h}\left[\frac{1}{2}(u_{x,1} + u_{x,2}) - u_{x,W} + u_{y,N} - u_{y,S}\right] \tag{3.20}$$

Figure 3.2 shows the topology, nomenclature for cell labeling and locations of the $u_{x,1}$ and $u_{x,2}$.

For the uni-directionally refined grid, the Laplace operator is given by

$$L(u_0) = \frac{1}{h}\left[u_{x,E} - u_{x,W} + AR_0\left(u_{y,N} - u_{y,S}\right)\right] \tag{3.21}$$

The cells are uniformly stretched in the positive y-direction only, and non-unit aspect ratio cells are allowed. The geometric stretching is specified by the ratio of successive cell heights in the stretching direction, so that

$$\frac{\Delta y_N}{\Delta y_0} = \frac{\Delta y_0}{\Delta y_S} = \beta \tag{3.22}$$

The difference in centroid heights is then

$$\begin{aligned} \bar{y}_N - \bar{y}_0 &= \frac{h(1+\beta)}{2AR_0} \\ \bar{y}_0 - \bar{y}_S &= \frac{h(1+\beta)}{2\beta AR_0} \end{aligned} \tag{3.23}$$

where the aspect ratio is defined as $AR_0 = h/\Delta y_0$.

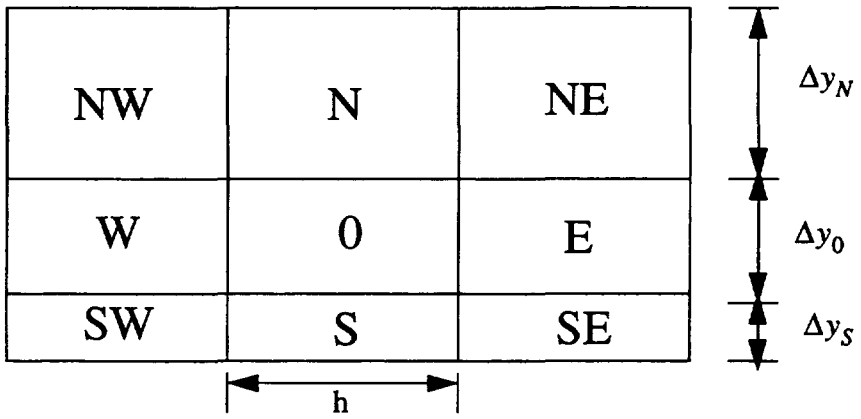


Figure 3.3 Grid and Nomenclature for Uni-directionally Stretched Grid

The uni-directionally stretched grid is included in the analysis since it is a common topology encountered in structured grid calculations, and its inclusion sheds light into the

performance of the current viscous flux formulae in use today. Although the Cartesian-mesh approach does not preclude the use of non-unit aspect ratio cells, none of the calculations performed here use them.

3.1.2 Tools Needed to Assess the Viscous Flux Formulae

To perform the analysis of the viscous flux formulae the schemes are assessed for solving a generic Laplacian of a scalar variable, u. The choice of this model equation is based upon the form of the viscous terms of the Navier-Stokes equations. If it is assumed that the fluid is incompressible with a constant laminar viscosity, the momentum equations may be written in non-conservation form as

$$\rho \frac{\partial u_i}{\partial t} + u_j u_{i,j} + P_{,i} = \frac{\mu}{Re_\infty} u_{i,kk} \tag{3.24}$$

From this it is seen that the Laplacian is representative of the viscous stress terms of the Navier-Stokes equations.

There are many useful physical principles that are described by Laplace's equation, and as a result of centuries of effort at solving it analytically, there are a variety of mathematical tools at hand. Indeed, the Cauchy-Riemann equations provide a framework for the entire field of complex variables and conformal mappings, which by their very nature provide solutions to Laplace's equation for analytic functions in the complex plane. An important principle obtained from complex analysis and applied here is found from the Cauchy integral formula. This states that the value of an analytic function at a point can be found from a contour integral around a closed path around that point. It is important to stress the analyticity of the function, since the real and imaginary parts of an analytic function in the complex plane directly satisfy the Cauchy-Riemann equations. Specifically, this leads to a maximum principle that is held by solutions to Laplace's equation, which states that the maxima of the function are on the boundaries of the domain. Locally, this implies

that the value at a point is bounded by the solution in the neighborhood of that point. If this is interpreted on a discrete level, as in applying a finite-difference approximation of Laplace's equation, it provides conditions upon the coefficients used in the stencil so that the approximation will satisfy a discrete, local maximum principle. If an N-point stencil is considered, that is arrived at by some means to solve Laplace's equation, it can be written as

$$\nabla^2 u \approx L(u) = \sum_{n=0}^{N} \alpha_n u_n = 0 \tag{3.25}$$

where the support set used contains at least the nearest neighbors to u_0. This can be rewritten to solve for u_0 as

$$u_0 = \sum_{n=1}^{N} \omega_n u_n \tag{3.26}$$

so that $\omega_n = -\frac{\alpha_n}{\alpha_0}$. A discrete maximum principle then states that u_0 is bounded by its neighbors

$$min(u_1, u_2, \ldots) < u_0 < max(u_1, u_2, \ldots) \tag{3.27}$$

A sufficient condition to satisfy (3.27) is to require all the $\omega_n \geq 0$. Therefore, by whatever means a stencil is found as a Laplace operator, if all of the coefficients of the stencil are positive, then the scheme discretely mimics an important physical property of the continuous Laplacian: it satisfies a maximum principle.

In addition to the positivity of the stencil, the accuracy of the approximation is readily obtained by applying a local Taylor series approximation to (3.25). Expanding the series to K-th order about $(x, y)_0$, results in

$$L(u) = \sum_{k=0}^{K} \sum_{m=0}^{k} \sum_{n=0}^{N} \alpha_n \xi_n^{k-m} \eta_n^m \frac{1}{(k-m)!\,m!} \frac{\partial^K u}{\partial x^{k-m} \partial y^m} \tag{3.28}$$

where $\xi_n = x_n - x_0$ and $\eta_n = y_n - y_0$. Collecting like partial derivatives the expansion may be written as

$$L(u) = (\sum_n \alpha_n) + (\sum_n \alpha_n \xi_n)\frac{\partial u}{\partial x} + (\sum_n \alpha_n \eta_n)\frac{\partial u}{\partial y} + \frac{(\sum_n \alpha_n \xi^2{}_n)}{2}\frac{\partial^2 u}{\partial x^2} + \ldots \qquad (3.29)$$

A general set of conditions for the α_n can then be arrived at that must be met for the discrete approximation of any linear partial differential equation. For a discrete Laplacian to have a truncation error of second order, all of the following conditions must be met:

$$\sum_n \alpha_n = 0 \qquad (3.30)$$

$$\sum_n \alpha_n \xi_n = 0 \qquad (3.31)$$

$$\sum_n \alpha_n \eta_n = 0 \qquad (3.32)$$

$$\sum_n \alpha_n \xi^2{}_n = 2 \qquad (3.33)$$

$$\sum_n \alpha_n \xi_n \eta_n = 0 \qquad (3.34)$$

$$\sum_n \alpha_n \eta^2{}_n = 2 \qquad (3.35)$$

$$\sum_n \alpha_n \xi^3{}_n = 0 \qquad (3.36)$$

$$\sum_n \alpha_n \xi^2{}_n \eta_n = 0 \qquad (3.37)$$

$$\sum_n \alpha_n \xi_n \eta^2{}_n = 0 \qquad (3.38)$$

$$\sum_n \alpha_n \eta^3{}_n = 0 \qquad (3.39)$$

The conditions set by equation (3.30) to (3.35) guarantee a consistent, first-order approximation, while the addition of the conditions set by (3.36) to (3.39) give a consistent sec-

ond- order accurate scheme. That is, for a second-order accurate scheme, the Laplace operator can be written as

$$L(u) = \nabla^2 u + O(h^2) \tag{3.40}$$

and a first-order accurate stencil is then

$$L(u) = \nabla^2 u + O(h) \tag{3.41}$$

A stencil is termed inconsistent if the Laplace operator is found to be

$$L(u) = k_1 u_{xx} + k_2 u_{xy} + k_3 u_{yy} + O(h) \tag{3.42}$$

and either $k_1 \neq 1$ and/or $k_2 \neq 0$ and/or $k_3 \neq 1$. This can also be termed a zero-th order approximation, since in the limit of zero mesh size, the modified equation does not converge to the desired partial differential equation.

It is possible, from a poor means of creating the stencil, to obtain a stencil whose truncation error that varies inversely with h. That is

$$L(u) = (\alpha_1 u_x + \alpha_2 u_y)\frac{1}{h} + (k_1 u_{xx} + k_2 u_{xy} + k_3 u_{yy}) + O(h) \tag{3.43}$$

where $\alpha_1 \neq 0$ and/or $\alpha_2 \neq 0$. In this case, even grid convergence to the wrong equation can never be achieved. This type of truncation error will be termed as dangerously inconsistent and should be avoided.

Positivity of the operator is found by requiring

$$\alpha_n \geq 0 \qquad \forall \; n > 0 \tag{3.44}$$

As can be seen, once the weights of the stencil are found the accuracy and positivity can be directly obtained. The positivity of the stencil is found by comparing the smallest weight to the root mean square of all of the stencil weights. This is termed the positivity parameter, $\tilde{\alpha}_{min}$ and is defined as

$$\tilde{\alpha}_{min} = \frac{min(\alpha_n, 0)}{\sqrt{\sum_{n=1}^{N} \frac{\alpha^2_n}{N}}} \tag{3.45}$$

for a stencil with N members in its support set.

Here, the assessment is made of the candidate flux functions by using them to construct finite-volume solutions to Laplaces's equation on the model grids. Following the standard finite-volume approach, the solution to Laplace's equation is written as

$$\int_A \nabla^2 u dA = \oint_S \nabla u \bullet \hat{n} dS \tag{3.46}$$

For an arbitrary polygon, a second-order quadrature may be written as

$$\nabla^2 u = \frac{1}{A} \sum_{edges} \nabla u \bullet \hat{n} \Delta S \tag{3.47}$$

As stated earlier, the essence of the problem is obtaining the gradient of a scalar quantity at the midpoint of the cell edges. The schemes analyzed here use two different approaches to find the gradients at the cell faces. The first, and most prevalent method is based upon an application of the divergence theorem to a co-volume or polygon surrounding the face, where the resulting surface integral is found using a second-order quadrature. The schemes are differentiated by the particular co-volumes used and the way the data are obtained at the vertices of the co-volume. Another approach is based upon reconstructing a polynomial at the face, and differentiating it to obtain the gradients. The analysis of these schemes begins here.

3.1.3 Green-Gauss Reconstruction: Centroidal Path

This procedure finds the gradients at the cell-to-cell interfaces by applying the divergence theorem to a polygon formed by joining the centroids of cells in a path about the

face. For a uniform Cartesian grid, the path about the East face is shown in Figure 3.4.

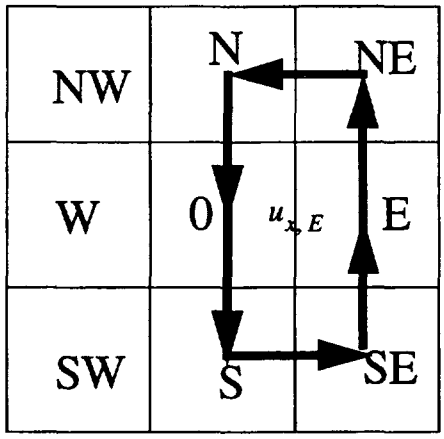


Figure 3.4 Centroidal Path Reconstruction: Path for East Face Reconstruction

For more arbitrary polygons and connectivity, the cells used in the reconstruction are found by creating a positively oriented list of cells found from the face and vertex neighbor connectivity. After this list is generated, the gradient is then found by a simple second-order accurate quadrature

$$\nabla u = \frac{1}{\Omega}\sum_{j}\frac{(u_j + u_{j+1})}{2}\boldsymbol{n}_j \tag{3.48}$$

where $\boldsymbol{n}_j$ is the outward pointing normal of the j-th face of the positively oriented polygon whose area is Ω. This reconstruction procedure is based upon that outlined in [38].

Application of this reconstruction procedure results in a decoupled, or rotated Laplacian, as is shown in Figure 3.5.

This stencil is marginally positive, consistent and second-order accurate and is classically known as a "rotated" Laplacian. Because it has vanishing weights for cells that are face

$$L(u) = \frac{1}{2h^2} * \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & -4 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} = \nabla^2 u + \frac{h^2}{12}(u_{xxxx} + 6u_{xxyy} + u_{yyyy}) + .$$

Figure 3.5 Centroidal Path Stencil: Uniform Cartesian Grid

neighbors of the cell, it can lead to a checkerboard type of instability. This decoupling arises since cells oriented along the front i+j=k are decoupled from those along fronts described by i+j=k+1 and i+j=k-1. This decoupling can lead to instability, and if somehow stabilized by increased truncation error at boundaries, can yield a non-smooth solution.

For the East-refined model grid, the gradients at the interfaces are found following the centroidal-path reconstruction procedure, which, after insertion into the flux balance formulation, (3.20) gives the weights shown in Figure 3.1. As can be seen, this stencil is inconsistent and non-positive, with a $\tilde{\alpha}_{min} = -0.2184\ldots$

$$L(u) = \frac{1}{420h^2} * \begin{array}{|c|c|c|c|} \hline 225 & 23 & \multicolumn{2}{c|}{176} \\ \hline -30 & -910 & 48 & 0 \\ \cline{3-4} & & 48 & 0 \\ \hline 225 & 23 & \multicolumn{2}{c|}{176} \\ \hline \end{array} = \frac{1}{420}(411u_{xx} + 425u_{yy}) - \frac{h}{560}(7u_{xxx} + 65u_{xyy}) + \ldots$$

Figure 3.6 Centroidal Path Stencil: East Face Refined Grid

For the stretched grid, the situation gets even worse. The results of the truncation error analysis are shown in Figure 3.7. For brevity, the functional forms of the stencil weights are not presented.

$$L(u) = \frac{1}{h^2} \begin{array}{|c|c|c|} \hline \alpha_{NW} & \alpha_N & \alpha_{NE} \\ \hline \alpha_W & \alpha_0 & \alpha_E \\ \hline \alpha_{SW} & \alpha_S & \alpha_{SE} \\ \hline \end{array} = u_{xx} + \frac{(1+\beta)^2}{4\beta}u_{yy} + h\left(\frac{(\beta-1)(1+\beta)^3}{24AR_0\beta^2}u_{yyy} + \frac{(\beta^2-1)}{4AR_0\beta}u_{xxy}\right) + \ldots$$

Figure 3.7 Centroidal Path Stencil: Uni-directionally Stretched Cartesian

The inconsistency is independent of cell aspect ratio, but the truncation error is not: it varies inversely with increasing aspect ratio. As is shown in the subsequent sections, this particular form of the inconsistency error is the same for an entire class of schemes: schemes

that are termed linearity preserving. For low stretchings, the inconsistency error is low, and, as shown in [65], if the stretching varies linearly with the grid size, the scheme approaches consistency in the limit of zero mesh size. This stresses the importance of mesh smoothness for structured and unstructured schemes. Figure 3.8 shows the variation of the inconsistency error, ε_{yy} with $\beta - 1$ where the inconsistency error is that obtained for the stencil whose modified equation results in the form

$$L(u) = u_{xx} + \varepsilon_{yy} u_{yy} + \ldots \tag{3.49}$$

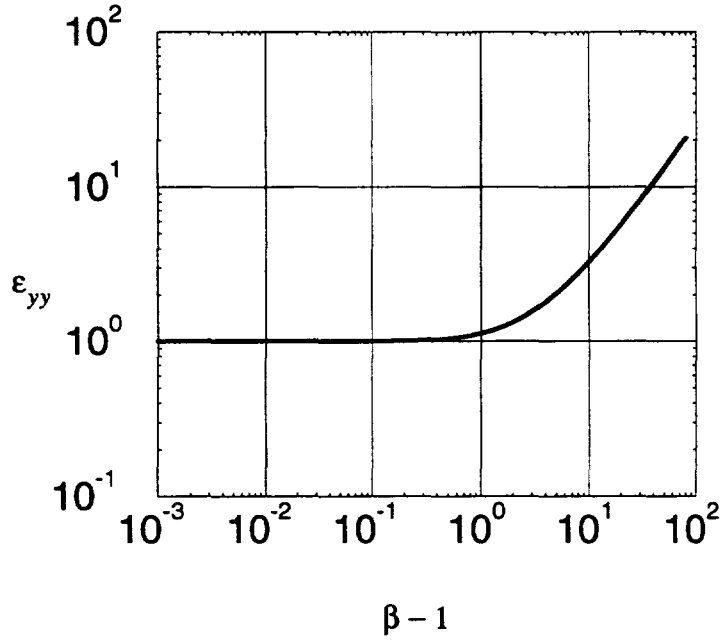


Figure 3.8 Inconsistency Error for Linearity Preserving Schemes

It is illuminating to show the stencil on a grid one might encounter with a typical structured grid calculation. To do this, the aspect ratio and stretching are set to be $AR_0 = 100$ and $\beta = 1.2$ and the weights in (3.47) are evaluated. The resulting stencil is shown in Figure 3.9. The resulting stencil is very non-positive, with a positivity parameter of $\tilde{\alpha}_{min} = -1.262\ldots$.

$$L(u) = \frac{1}{h^2}$$

2273	4545	2273
-5000	-10001	-5000
2728	5454	2728

$$= u_{xx} + 1.0083u_{yy} + h(4\times10^{-3}u_{yyy} + 9\times10^{-4}u_{xxy}) + \ldots$$

Figure 3.9 Centroidal Path Stencil: Sample Stretching/Aspect Ratio

As is seen, the scheme is very non-positive on stretched meshes with a reasonable aspect ratio, but is also non-positive on unstretched meshes with a non-unit aspect ratio, where the positivity measure is

$$\tilde{\alpha}_{min} = \frac{1 - AR_0^{\,2}}{\sqrt{5AR_0^{\,4} - 6AR_0^{\,2} + 5}} \tag{3.50}$$

On a mesh that has unit aspect ratio, but is stretched, the scheme decouples the East and West neighbors, and can have a non-positive North neighbor, which gives the positivity measure

$$\tilde{\alpha}_{min} = \frac{1 - \beta}{\sqrt{3\beta^2 - 2\beta + 3}} \tag{3.51}$$

It is worth noting here that this scheme for the viscous flux functions was the first flux formula implemented into the Cartesian solver, before any analysis was made, and from the very outset it was difficult to get converged solutions. Typically, if a solution was converged, it was "noisy" and non-monotone. This behavior prompted an analytical investigation of the scheme, and revealed the above decoupled behavior. This decoupling and non-

positivity of neighbors in the Laplace stencil is a recurrent theme that shows up in many of the schemes soon to be analyzed.

3.1.4 Green-Gauss Reconstruction: Existing Faces Co-Volume

This reconstruction procedure has been used successfully in [39] and [36] on triangular unstructured grids, although the analysis here indicates a tendency to non-positive and decoupled stencils. The basic idea behind this type of reconstruction is to form a co-volume about the face where the reconstructed gradient is needed by using the faces already present in the mesh. To perform the path integration, the values at the face midpoints are taken to be the mean of the two centroid values of the cells that share the face. Figure 3.10 shows the covolume created for reconstructing the gradient about the East face on the uniform Cartesian mesh.

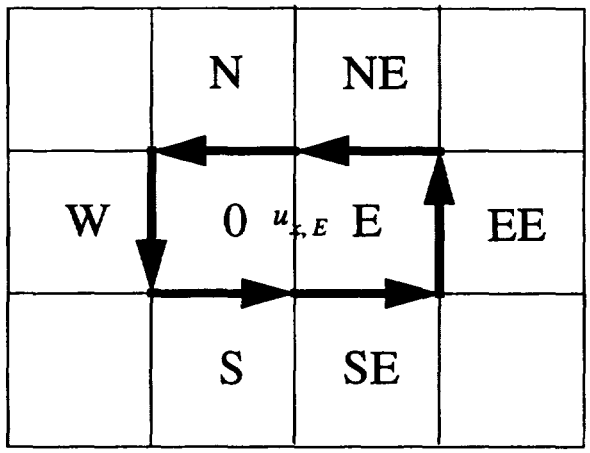


Figure 3.10 Existing Faces Reconstruction Path: Uniform Cartesian Grid

The gradient is then calculated following the line quadrature shown in (3.48), where for an example, the gradient $u_{x,E}$ in Figure 3.10 would be

$$u_{x,E} = \frac{1}{2h^2}\left[\frac{(u_E + u_{EE})}{2}h + \frac{(u_0 + u_W)}{2}(-h)\right] \tag{3.52}$$

For the uniform Cartesian grid, this reconstruction procedure *completely decouples all order-one neighbors* although it is second order accurate and consistent with weights shown in Figure 3.11.

$$L(u) = \frac{1}{4h^2}$$

		1		
	0	0	0	
1	0	-4	0	1
	0	0	0	
		1		

$$\nabla^2 u + \frac{h^2}{3}(u_{xxxx} + u_{yyyy}) + \ldots$$

Figure 3.11 Existing Face Path Stencil: East Face Refined Cartesian Grid

On the East face refined Cartesian grid, the solution decouples in the y-direction, and results in a non-positive stencil shown in Figure 3.12. The scheme is dangerously inconsistent, with the leading truncation error term $u_x/(80h)$. With a truncation error of this form, grid convergence could never be achieved, as successive refinement of the mesh will never result in convergence of the modified equation resulting from the discrete Laplacian. It is also very non-positive, with a positivity parameter of $\tilde{\alpha}_{min} = -0.863\ldots$.

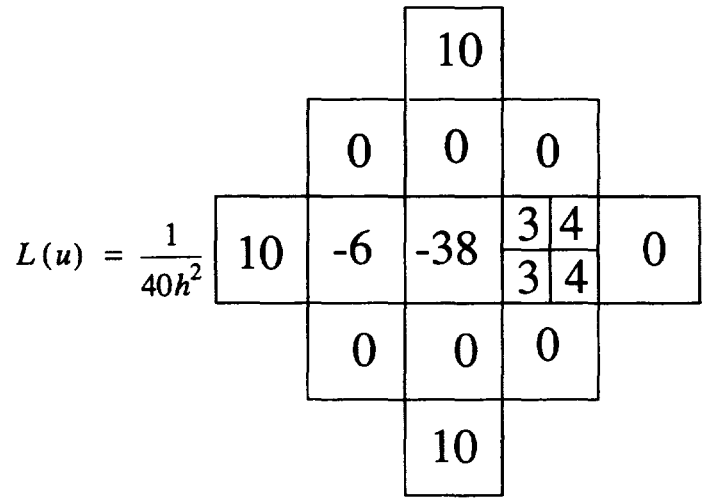


Figure 3.12 Existing Face Path Stencil: East-Face Refined Grid

The poor behavior of this scheme is also evident on the stretched mesh. Figure 3.13 shows the stencil and modified equation that results from application of this reconstruction procedure.

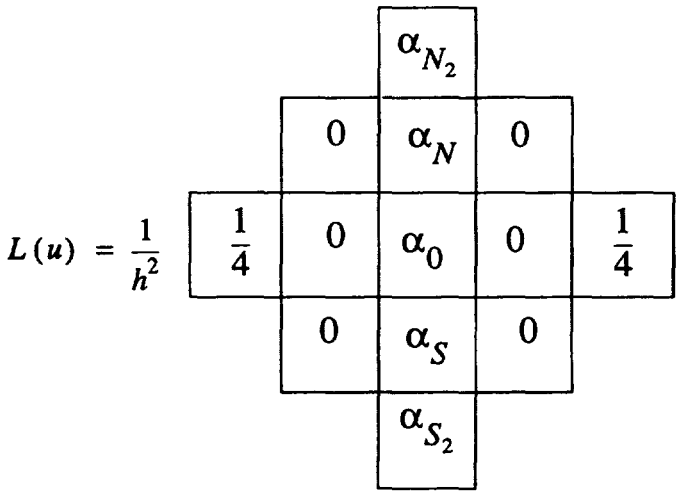


Figure 3.13 Existing Face Path Stencil: Uni-directionally Stretched Cartesian Grid

The stencil coefficients are shown below, as well as the important components of the modified equation.

$$\alpha_{N_2} = \frac{AR_0^2}{(3+\beta)} \tag{3.53}$$

$$\alpha_N = \frac{AR_0^2(1-\beta)}{(3+\beta)(1+3\beta)} = -\alpha_S \tag{3.54}$$

$$\alpha_{S_2} = \frac{AR_0^2\beta}{(1+3\beta)} \tag{3.55}$$

$$\alpha_0 = -\frac{1}{2} - \frac{AR_0^2}{(3+\beta)} - \frac{AR_0^2\beta}{(1+3\beta)} \tag{3.56}$$

The inconsistency error is large, where the term in Figure 3.13 is

$$\varepsilon_{yy} = \frac{(1+\beta)^4(3+2\beta+3\beta^2)}{8\beta^3(3+10\beta+3\beta^2)} \tag{3.57}$$

and the first-order truncation error term is a large polynomial in the stretching parameter. As can be seen, the stencil is decoupled completely in the x-direction, and gives non-positive coefficients in the North cell for non-unity stretchings.

Due to the tendency of the scheme to decouple order-one neighbors or to give a non-positive scheme, it is not considered a good candidate scheme for solving the Navier-Stokes equations and will not be analyzed further.

3.1.5 Green-Gauss Reconstruction: Diamond Path with Simple Vertex Weighting

This reconstruction procedure is used in a variety of cell-centered, finite-volume compressible flow solvers that have been used for a wide variety of calculations ([27][32][50][72][65] and many others). The reconstruction of the gradients at each face is found by applying the divergence theorem to a four-sided polygon, or diamond path,

whose vertices are defined by the two centroids of the cells sharing the face and the two vertices subtending the face. Figure 3.14 shows the path for the interface at the refinement boundary of the East Face refined grid.

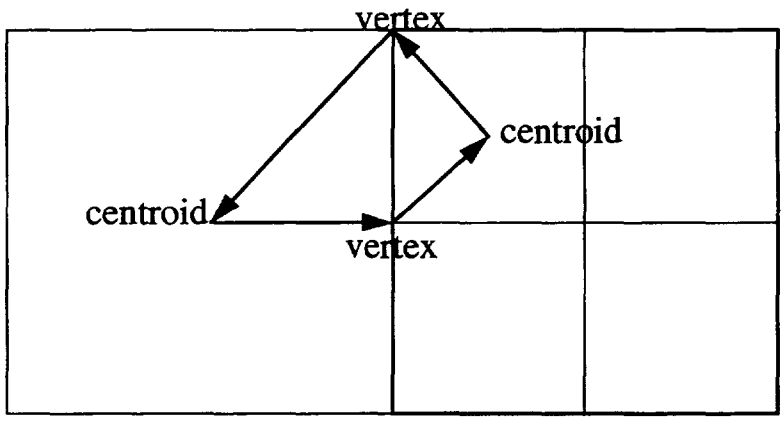
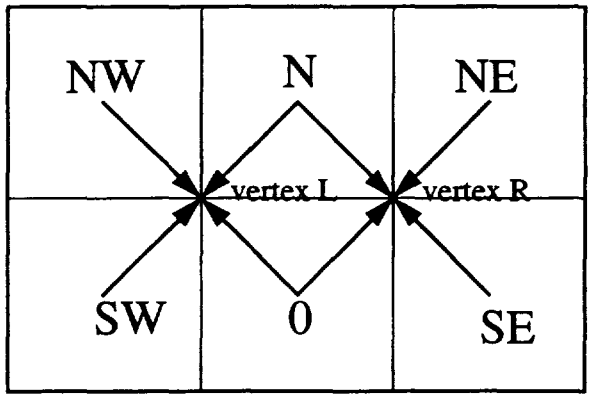


Figure 3.14 Diamond Path Reconstruction: Sample Path

The path is local, but does require some sort of an interpolation or averaging procedure to provide data at the vertices of the face. The simple approach, taken in [50] and in [72] is to average the data from the cells that share the vertex in question. Figure 3.15 shows this for an un-refined Cartesian grid, where the gradient is desired at the North face.



$$u_R = \frac{1}{4}(u_N + u_0 + u_{SE} + u_{NE})$$

$$u_L = \frac{1}{4}(u_N + u_0 + u_{NW} + u_{SW})$$

Figure 3.15 Simple Averaging Procedure at Subtended Vertex

This approach naturally brings in the nearest neighbors of a cell, and attempts to keep the stencil local by keeping the path local. In terms of weights, the cell averaging procedure

can be written as

$$u_{vertex} = \frac{\sum_{n=1}^{N} \omega_n u_n}{\sum_{n=1}^{N} \omega_n} \tag{3.58}$$

where N is the number of cells that share the vertex and $\omega_n = 1$ for all n for this simple averaging procedure.

Applying this reconstruction procedure to the uniform Cartesian grid results in a desirable Laplacian, namely the (-4,1,1,1,1) stencil shown in Figure 3.21. This comes about due to a *geometric cancellation* of terms in the reconstruction procedure, and in fact, does not bring into play the averaged data at the vertex. This can be realized by inspecting the quadrature used in the reconstruction for, say, the East face. The integral around the diamond path is broken up into four segments, shown below.

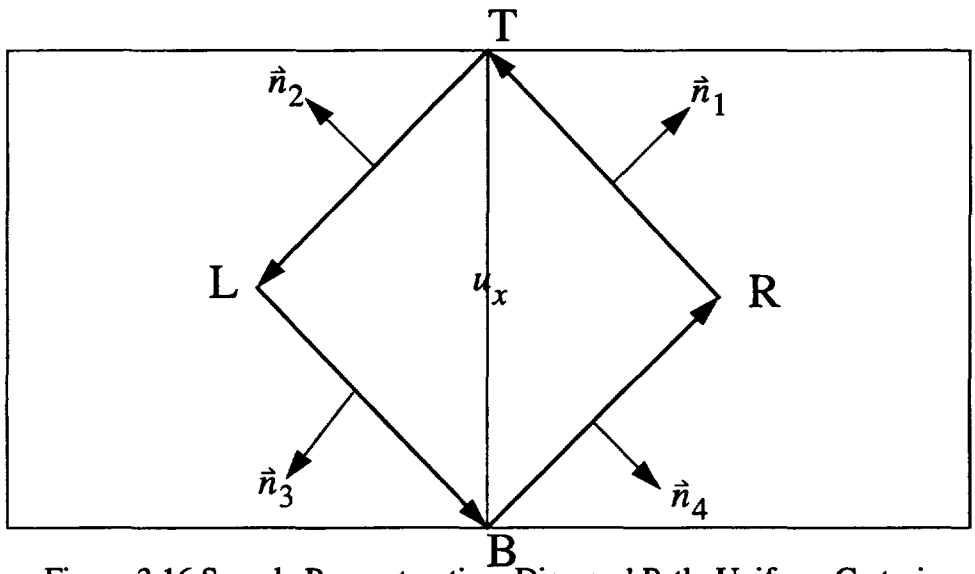


Figure 3.16 Sample Reconstruction, Diamond Path, Uniform Cartesian

Then, the reconstructed gradient, u_x is simply

$$u_x = \frac{1}{\Omega}\left(\frac{(n_{1,x}+n_{2,x})}{2}u_T + \frac{(n_{2,x}+n_{3,x})}{2}u_L + \frac{(n_{3,x}+n_{4,x})}{2}u_B + \frac{(n_{4,x}+n_{1,x})}{2}u_R\right) \tag{3.59}$$

The area of the co-volume is Ω. By symmetry, $n_{1,x}+n_{2,x} = n_{3,x}+n_{4,x} = 0$, so that there is no contribution of u_T and u_B to the gradient. The resulting gradient is simply the face difference, $u_x = (u_R - u_L)/h$.

This geometric cancellation does not always occur, and when it does not, the method of obtaining the data at the vertices of the face is more important. The stencil obtained using the simple weighting procedure upon the East face refined grid is shown in Figure 3.17.

$$L(u) = \frac{1}{36h^2} * \begin{array}{|c|c|c|c|} \hline 0 & 33 & -3 & \\ \hline 36 & -154 & 29 & 0 \\ \hline & & 29 & 0 \\ \hline 0 & 33 & -3 & \\ \hline \end{array} = \frac{u_x}{24h} + \frac{1}{576}(501u_{xx} + 509u_{yy}) + \ldots$$

Figure 3.17 Diamond Path Reconstruction, Simple Weighting: Stencil for Uniform Grid

As is seen, the resulting stencil is not positive with $\tilde{\alpha}_{min} = -0.110\ldots$ and is dangerously inconsistent, and therefore can never be grid converged.

Application of this reconstruction procedure on the uni-directionally stretched grid results in gradients that are simply the face differences divided by the distance over which the difference is made. The resulting stencil is

$$L(u) = \frac{1}{h^2} * \begin{array}{|c|c|c|} \hline 0 & \alpha_N & 0 \\ \hline 1 & \alpha_0 & 1 \\ \hline 0 & \alpha_S & 0 \\ \hline \end{array} = u_{xx} + \frac{(1+\beta)^2}{4\beta} u_{yy} + \frac{h(1+\beta)^3(\beta-1)}{24AR_0\beta} u_{yyy} + \ldots$$

Figure 3.18 Diamond Path Reconstruction with Simple Weighting: Stretched Grid

$$\begin{aligned} \alpha_0 &= -2(1+AR_0^2) \\ \alpha_N &= \frac{2AR_0^2}{(1+\beta)} \\ \alpha_S &= \beta\alpha_N \end{aligned} \tag{3.60}$$

It is important to note that this stencil comes about more from fortunate geometric cancellations than from a proper reconstruction procedure. This will be shown in more detail in a later section.

3.1.6 Green-Gauss Reconstruction: Diamond Path Using a Linearity Preserving Weighting Function

An obvious improvement to the previous scheme is to provide a more accurate means of finding the data at the vertices of the faces.The reconstruction procedure here is identical to that shown in Section 3.1.5 but uses a linearity-preserving weighting to find the values at the vertices. This ensures that the reconstruction procedure using the path integration will reconstruct linear functions exactly since the Green-Gauss reconstruction is simply a weighted sum of two independently linear reconstruction procedures. The sin-

gle, diamond path can be broken up into the path about two triangles, which share a common face. Since the gradient in each triangle is known exactly, then, by application of the divergence theorem, the line integral about the diamond path will simply be the area-weighted sum of the individual triangle gradients divided by the total area. Since each sub-triangle will reconstruct a linear function exactly given linear data, the reconstructed gradient over the diamond path inherits this same property. The improved weighting used here ensures that the data provided to the reconstruction procedure is linearly obtained from the centroid data. This property of reconstructing linear gradients exactly is termed linearity preservation, and is shown later to imply an important property of the constructed Laplacian.

The linearity-preserving weighting function used here is based upon the linearity-preserving Laplacian derived by Holmes and Connell in [35] and used by Rausch et. al. in [66] and by Knight in [41]. There, the similarity between a local Laplace operator and an averaging procedure is used to find the weights used in the weighting function (3.58). The procedure shown there is based upon perturbing the weights of the cells from one, and minimizing the sum of the squares of the perturbation. The perturbations are then found by applying the method of Lagrange multipliers subject to the constraints that the constructed Laplacian is zero for all linear data. Consider the vertex 0, surrounded by a set of vertices, as in Figure 3.19.

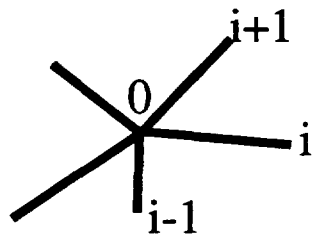


Figure 3.19 Schematic of vertices surrounding object vertex for linearity-preserving reconstruction.

If the weights are

$$\omega_i = 1 + \Delta\omega_i \tag{3.61}$$

minimize the cost function

$$C = \sum_i \Delta\omega_i^2 \tag{3.62}$$

subject to the linearity constraints

$$\begin{aligned} L(x_0) &= \sum_i \omega_i (x_i - x_0) = 0 \\ L(y_0) &= \sum_i \omega_i (y_i - y_0) = 0 \end{aligned} \tag{3.63}$$

This results in the weights as

$$\omega_i = 1 + \lambda_x (x_i - x_0) + \lambda_y (y_i - y_0) \tag{3.64}$$

$$\lambda_x = \frac{I_{xy} R_y - I_{yy} R_x}{I_{xx} I_{yy} - I^2_{xy}} \tag{3.65}$$

$$\lambda_x = \frac{I_{xy} R_x - I_{xx} R_y}{I_{xx} I_{yy} - I^2_{xy}} \tag{3.66}$$

where the various moments are

$$I_{xx} = \sum_i (x_i - x_0)^2 \tag{3.67}$$

$$I_{yy} = \sum_i (y_i - y_0)^2 \tag{3.68}$$

$$I_{xy} = \sum_i (x_i - x_0)\,(y_i - y_0) \tag{3.69}$$

$$R_x = \sum_i (x_i - x_0) \tag{3.70}$$

$$R_y = \sum_i (y_i - y_0) \tag{3.71}$$

The data at the vertex $(x, y)_0$ are then found by requiring the Laplacian evaluated there

be zero. In other words,

$$L(u_0) = \sum_i \omega_i (u_i - u_0) = 0 \tag{3.72}$$

from which the general form, (3.58), is found.

The same weights can be obtained in a different way, as is shown in Chapter V, where the formulation gives rise to weights which preserve higher-order functions, and is termed a consistency-preserving Laplacian. But, applying a higher-order weighting in the context of supplying data to a reconstruction procedure that can at best preserve only linear data is unnecessary.

Due to fortunate geometric cancellations, the stencils obtained on the uniform Cartesian and uni-directionally stretched Cartesian meshes are identical to those obtained using the simple, unity weighting in Section 3.1.5. The stencil obtained on the East-Face refined mesh is improved, although it is still inconsistent and not positive, with a comparatively low $\tilde{\alpha}_{min} = -0.115\ldots$. The resulting stencil is

$$L(u) = \frac{1}{171h^2} * \begin{array}{|c|c|c|} \hline 0 & 162 & -15 \\ \hline 171 & -733 & \begin{array}{cc} 134 & 0 \\ 134 & 0 \end{array} \\ \hline 0 & 162 & -15 \\ \hline \end{array} = \frac{1}{1368}(1167u_{xx} + 1243u_{yy}) - \frac{h}{5742}(469u_{xxx} + 279u_{xyy}) + \ldots$$

Figure 3.20 Diamond Path Reconstruction Stencil using Linearity Preserving Weighting Function

3.1.7 Polynomial Reconstruction: $K_v = 1$

This reconstruction procedure is similar to that presented by Mitchell and Walters in [55], and reconstructs a linear function about the face using a set of support cells and a Lagrange polynomial type of reconstruction upon this support set. This function is then differentiated to obtain the gradients needed in the flux calculations. The set of support cells is taken to be the minimum number needed to solve for the unknown coefficients in the expansion. Since there is a large number of possible combinations of cells to make up the support set, some criterion is needed to select them. The two cells that share the face are always included in the support set, which simplifies matters for a linear reconstruction since only three cells are needed. As suggested in [55], the third cell can be found so that the centroid of the co-volume (triangle) is closest to the face midpoint in addition to the implied requirement that the resulting three points are not co-linear. For the linear expansion, the reconstruction can be realized for *any* non-co-linear points, but *which* point is the *best* choice in terms of quality of the reconstruction and its contribution to the cell stencil does not motivate the selection criterion.

A general linear function is expanded about the face as

$$u(x, y) = C_0 + C_1 x + C_2 y \tag{3.73}$$

where the function is evaluated at the centroids of the cells in the support set, so that the gradients are found from the linear relation, $A_{ij} C_j = u_i$

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{3.74}$$

The gradients are

$$u_x = \frac{1}{2det(A)}\left[u_1(y_2 - y_3) + u_2(y_3 - y_1) + u_3(y_1 - y_2)\right]$$
$$u_y = \frac{-1}{2det(A)}\left[u_1(x_2 - x_3) + u_2(x_3 - x_1) + u_3(x_1 - x_2)\right] \tag{3.75}$$
$$2det(A) = x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3 + x_1y_2 - x_2y_1$$

It should be noted that $det(A)$ is also the signed area of the triangle formed by connecting the 3 points and that this reconstruction yields the exact gradients for linear functions, hence is termed linearity preserving.

For the uniform Cartesian grid, the reconstruction is evaluated at each face, for all of the support sets that allow an invertible system. For each face, there are a number of possible support sets that may be chosen, and of these support sets the ones that are invertible yield the same gradient. This gradient is simply the divided face difference across the face, $(u_R - u_L)/h$. This is a step in the right direction, yielding the desired, standard Laplacian weights shown in Figure 3.21.

$$L(u) = \frac{1}{h^2} * \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} = \nabla^2 u + \frac{h^2}{12}(u_{xxxx} + u_{yyyy}) + \ldots$$

Figure 3.21 Linear Reconstruction: Uniform Cartesian Mesh

For the East refined Cartesian grid, the divided face-difference gradients for the North, South and West faces are found, regardless of the choice of the invertible support sets. For the $u_{x,1}$ and $u_{x,2}$ gradients, choosing the third point so that the triangle's centroid is clos-

est to the face midpoint requires that the NorthEast and SouthEast points be used in the gradients $u_{x,1}$ and $u_{x,2}$, respectively, which results in the stencil shown in Figure 3.22. This stencil is non-positive with a positivity parameter of $\tilde{\alpha}_{min} = -0.292\ldots$, and is inconsistent.

$$L(u) = \frac{1}{4h^2} *$$

0	4	-1	
4	-18	4	0
		4	0
0	4	-1	

$$= \frac{13}{16}\nabla^2 u + \frac{h}{64}(-7u_{xxx} - 13u_{xyy}) + \ldots$$

Figure 3.22 Linear Reconstruction: East-Face Refined Cartesian Mesh

This choice is not the best and the stencil is not unique; positive and non-positive stencils can be realized if different choices are made for the third point when reconstructing the gradients $u_{x,1}$ and $u_{x,2}$. By taking advantage of the symmetry of the first-order neighbors of the 0, l and u cells (see Figure 3.2), one can see that there are a total of seven possible stencils that can be selected. Since the cells are symmetric about y=0, the choices of the third point for the $u_{x,1}$ reconstruction are cells t, b, l, NW, N, SW and W (see Figure 3.2). The $u_{x,2}$ stencil is then found by invoking symmetry in the data. Each of these choices is analyzed; a summary of the stencils is shown in Table III.

Table III K_v = 1 Stencils

Cell Choice	$\tilde{\alpha}_{min}$	Decoupled?	Modified Equation
NE	-0.29	No	$\frac{13}{16}\nabla^2 u - \frac{h}{64}(7u_{xxx} + 13u_{xyy}) + \ldots$
t	-1.0	No	$\frac{3}{2}u_{xx} + u_{yy} + \frac{h}{32}(11u_{xxx} + u_{xyy}) + \ldots$
b	0	No	$\frac{33}{32}\nabla^2 u + \frac{h}{32}(u_{xxx} + u_{xyy}) + \ldots$
l	0	No	$\frac{1}{24}(21u_{xx} + 25u_{yy}) + \frac{h}{96}(-7u_{xxx} + 3u_{xyy}) + \ldots$
NW	-0.176	No	$\frac{1}{32}(21u_{xx} + 29u_{yy}) + \frac{h}{128}(-7u_{xxx} + 19u_{xyy}) + \ldots$
N	0	No	$\frac{7}{8}\nabla^2 u + \frac{h}{96}(-7u_{xxx} + 3u_{xyyy}) + \ldots$
SW	0	No	$\frac{21}{16}\nabla^2 u - \frac{h}{64}(7u_{xxx} + 13u_{xyy}) + \ldots$
W	0	Yes	$u_{yy} + \frac{h^2}{12}u_{yyyy} + \ldots$

As can be seen from examination of the table, none of the choices yields a consistent Laplacian, and the stencil found from the geometrically-chosen support set is not the optimum. The least inconsistent scheme is found from the choice of cell b to close the stencil, as it has terms closer to unity pre-multiplying the Laplacian. The ambiguity of support set choice can yield a very dangerous stencil, as is shown in the last entry in the table. For the choice of the West cell to close the support set, a stencil is found that is completely decoupled in x, and results in a modified equation that has no x component in the Laplacian: a very dangerous choice, indeed. It is also important to note that none of the support sets

yielded a dangerously inconsistent stencil.

The results for the uni-directionally stretched grid are the same as in the previous section: the gradients obtained are simply the face differences divided by the length over which the difference is taken. On the sample mesh, the results are identical to those in Figure 3.8.

3.1.8 Polynomial Reconstruction: $K_V = 2$

This reconstruction procedure is similar to that presented above, but uses a quadratic polynomial instead of a linear function. As before, once a suitable set of support cells is found, the reconstructed polynomial is differentiated, giving the gradients needed in the flux formulae. The choice of cells to make up the support set is not obvious, and an improper choice can yield an improper stencil, or a set where the gradients are not realizable. The system to solve for the coefficients is found by requiring the expansion to be equal to the values at the support set data points for

$$u(x, y) = C_0 + C_1 x + C_2 y + C_3 x^2 + C_4 xy + C_5 y^2 \tag{3.76}$$

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ 1 & x_3 & y_3 & x_3^2 & x_3 y_3 & y_3^2 \\ 1 & x_4 & y_4 & x_4^2 & x_4 y_4 & y_4^2 \\ 1 & x_5 & y_5 & x_5^2 & x_5 y_5 & y_5^2 \\ 1 & x_6 & y_6 & x_6^2 & x_6 y_6 & y_6^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} \tag{3.77}$$

The gradients are found by differentiating (3.76) and then integrating over the interfaces. A second-order quadrature is used, which for these linear gradients is exact, and entails

evaluating the gradients at the face midpoints. The gradients are then

$$\begin{aligned} u_x &= C_1 + 2C_3 x + C_4 y \\ u_y &= C_2 + C_4 x + 2C_5 y \end{aligned} \tag{3.78}$$

Things can be simplified somewhat, so that in practice a local coordinate system situated at the face midpoint is used that is scaled by a local length scale, δ , where δ is taken to be the mean of the length scales of the two cells that share the face. That is,

$$\xi = \frac{x - x_{mid}}{\delta} \qquad \eta = \frac{y - y_{mid}}{\delta} \qquad \delta = \frac{1}{2}(\sqrt{A_L} + \sqrt{A_R}) \tag{3.79}$$

The form of the Vandermonde type matrix, (3.77), is unchanged, but the entries are now in terms of ξ and η. This results in a simplified gradient, which when evaluated at the face midpoint gives

$$\begin{aligned} u_x &= \frac{C_1}{\delta} \\ u_y &= \frac{C_2}{\delta} \end{aligned} \tag{3.80}$$

For the uniform mesh, care is needed in forming the support sets about the faces: To allow inversion of the matrix, a quadratic function in both x and y must be able to be formed from the support set. For the uniform grid, this precludes the use of the face's natural neighbors. As an example, consider reconstructing the East face gradient on the uniform grid, using the face natural neighbors, as in Figure 3.23. This configuration of the support set yields a singular matrix, (3.77), and requires the use of a modified support set. This singular behavior can be seen to be caused by the lack of sufficient data in the x-direction: only a linear function can be formed from two unique points. To overcome this non-invertibility, a modified support set is used for the uniform mesh, shown in Figure 3.24. For the un-refined mesh, application of the modified support set gives the non-face decoupled, standard second-order accurate Laplacian shown in Figure 3.21.

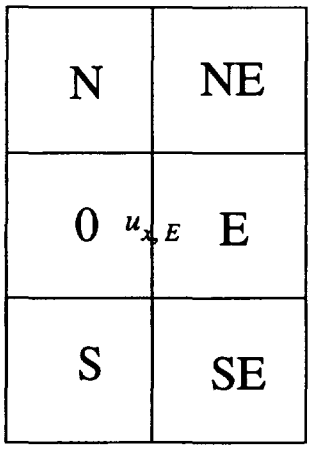


Figure 3.23 Singular Support Set About East Face.

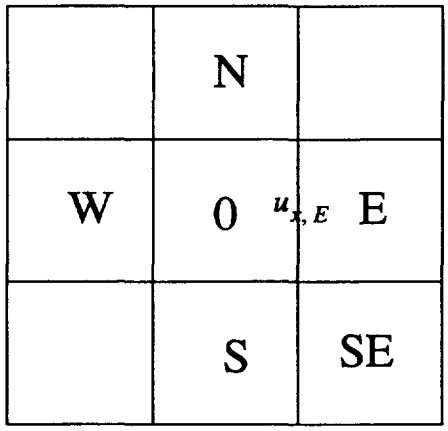


Figure 3.24 Modified Support Set for East Face Reconstruction.

For the East-Face refined Cartesian grid, the North, South and West gradients are found by applying a similar modified support set shown in Figure 3.24. The resulting gradients are the standard gradients, as shown in (3.77). For the gradients at the refinement boundaries, a very large number of possible support sets can be formed, and, as in the linear reconstruction, some good and some bad stencils can be constructed. There are a total of

$$\frac{10 \bullet 9 \bullet 8 \bullet 7}{4!} = 210 \tag{3.81}$$

combinations that may be made if the set of cells to be taken from the union of the order one neighbors of the two cells sharing the face, and always start the list with the two cells that share the face. This large number of stencil choices can be reduced, if the connected face neighbors of the object face are used, and the set closed with the remaining un-chosen cells. For the $u_{x,1}$ reconstruction five cells are found: cells 0, N, NE, u and l. A single cell must be chosen to close the set. Looking at the order-one neighbors of the two cells gives a choice of one cell from a set of seven. Some common-sense rules can be applied, as in [55], by applying a locality principle: require the "geometric center" (interpreted to be the centroid of the polygon formed by the support set) to be closest to the face midpoint. Ordering in terms of this distance, the choices of the sixth cell are found to be SE, NW, S, W, t, SW and b. The $u_{x,1}$ are then found by computing the gradient (3.78), and evaluating it at the face midpoint. $u_{x,2}$ is found with the same procedure, choosing a set of cells for its reconstruction in a symmetric fashion. Each choice gives a different stencil for the Laplacian, and as in the linear reconstruction procedure, the geometrically chosen set does not yield the best stencil. In fact, this stencil is decoupled from the East neighbors, and is shown in Figure 3.25. Each of the choices to close the support set is analyzed, and shown in Table III.

$$L(u) = \frac{1}{2h^2} *$$

0	1	1	
2	-6	0	0
		0	0
0	1	1	

$$= \nabla^2 u + h\frac{u_{xyy}}{2} + \ldots$$

Figure 3.25 Quadratic Reconstruction: SE Cell Choice

Table IV $K_v = 2$ Stencils

Cell Choice	$\tilde{\alpha}_{min}$	Decoupled?	Modified Equation
SE	0	Yes	$\nabla^2 u + h\frac{u_{xyy}}{2} + \ldots$
NW	0	No	$\nabla^2 u + \frac{h}{120}(-7u_{xxx} + 15u_{xyy}) + \ldots$
S	0	Yes	$\nabla^2 u + h\frac{u_{xyy}}{2} + \ldots$
W	0	No	$\nabla^2 u + \frac{h}{104}(-7u_{xxx} + 13u_{xyy}) + \ldots$
t	0	No	$\nabla^2 u + \frac{h}{576}(5u_{xxx} + 63u_{xyy}) + \ldots$
SW	-.0694	No	$\nabla^2 u - \frac{h}{64}(7u_{xxx} + 13u_{xyy}) + \ldots$
b	0	No	$\nabla^2 u + \frac{h}{128}(u_{xxx} + 19u_{xyy}) + \ldots$

The optimally-chosen stencil decouples and, of the seven stencils, two decouple and one is non-positive. All of the stencils obtained are consistent and first-order accurate; the best of all of the stencils found in this work. Attempts are made to improve the stencil selection of this scheme, as is shown in a later section, but the tendency of the quadratic reconstruction scheme to yield non-positive Laplacians does not result in a robust scheme for solving the Navier-Stokes equations. This is unfortunate, since this scheme is shown next to be better behaved on uni-directionally stretched meshes.

Application of the quadratic reconstruction procedure on the stretched mesh runs into the same difficulties as encountered on the uniform Cartesian mesh. By choosing the obvious cells in the support set to reconstruct the gradient at a face, a singular system can result. As before, by choosing a modified support set, as in Figure 3.24, unique gradients can be found for a set of choices for the support set. Applying this modified support set, the East and West gradients are the simple face difference values, while the North and South gradients are gradients whose truncation errors follow those of a quadratic function. Substitution of these into the flux formula gives the Laplacian

$$L(u) = \frac{1}{h^2} * \begin{array}{|c|c|c|} \hline 0 & \alpha_N & 0 \\ \hline 1 & \alpha_0 & 1 \\ \hline 0 & \alpha_S & 0 \\ \hline \end{array} \quad \nabla^2 u + h\frac{(-1+\beta^2)}{6AR_0\beta} u_{yyy} + \ldots$$

Figure 3.26 Uni-directionally Stretched Grid: Stencil for Quadratic Reconstruction

The coefficients in the stencil are

$$\alpha_0 = -\frac{2(1+2\beta+4AR_0{}^2\beta+\beta^2)}{(1+\beta)^2 h^2}$$
$$\alpha_N = \frac{8AR_0^2\beta}{(1+\beta)^3 h^2} \tag{3.82}$$
$$\alpha_S = \beta\alpha_N$$

This stencil is positive, consistent and first-order accurate. It is especially noteworthy that all of the stencils created using this quadratic reconstruction scheme are consistent and first order accurate. As will be shown in the next sections, this behavior will be evident on arbitrary meshes. But, as is also shown in practice in the next chapter, a positive stencil on realistic cut, Cartesian grids is very difficult to maintain.

3.2 Summary and Choice of Viscous Flux Functions

At this point, six different schemes have been analyzed to reconstruct the gradients needed to form the viscous flux function. The first two schemes, which use a Green-Gauss type reconstruction on a path formed by cell centroids or by existing faces in the mesh, were shown to have a tendency to produce decoupled solutions. This decoupling can inhibit convergence, and if converged, can yield noisy, non-smooth solutions. This was in fact observed when using the first scheme to compute a low Reynolds number flow, and prompted the analysis shown here. Due to this behavior, these two schemes are not wise choices for the viscous flux functions in a general, Navier-Stokes solver.

Of the four remaining schemes, the Green-Gauss reconstruction upon the diamond path using the simple vertex weighting has been shown to give a dangerously inconsistent Laplacian upon one of the model meshes. This behavior is traceable to the accuracy in which the gradient is obtained: The reconstructed gradient truncation error is zeroth order.

This property of a linearity-preserving weighting, and the benefits of using higher than linearity preserving reconstructions can be realized by analyzing the construction of the discrete Laplacian in a little more detail. If the Laplacian is to be constructed upon an arbitrary control volume and a second-order quadrature of the reconstructed gradients at the control volume interfaces is performed, the general formula for the Laplacian is

$$L(u) = \frac{1}{A}\sum_{faces} (D(u) \bullet \vec{\mathbf{n}}) \tag{3.83}$$

The reconstructed gradient $D(u)$ is expanded in a Taylor series about the cell centroid as $D(u) = D_x\hat{i} + D_y\hat{j}$ where

$$\begin{aligned} D_x &= b_0 u_x + b_1 u_y + (b_2 u_{xx} + b_3 u_{xy} + b_4 u_{yy})\, h + O(h^2) \\ D_y &= c_0 u_x + c_1 u_y + (c_2 u_{xx} + c_3 u_{xy} + c_4 u_{yy})\, h + O(h^2) \end{aligned} \tag{3.84}$$

The terms b_1 and c_0 are included to account for non-linearity preserving expansions of the gradients. Inserting into (3.83) and collecting like terms, the following is obtained

$$\begin{aligned} AL(u) = {} & u_x \sum (b_0 n_x + c_0 n_y) + u_y \sum (b_1 n_x + c_1 n_y) + \\ & u_{xx} \sum (b_2 n_x + c_2 n_y)\, h + u_{xy} \sum (b_3 n_x + c_3 n_y)\, h \\ & u_{yy} \sum (b_4 n_x + c_4 n_y)\, h + \ldots \end{aligned} \tag{3.85}$$

The sums are taken over the faces, where for general meshes, each expansion will have different b_n and c_n. From here, it can be seen where fortunate geometric cancellations actually occur, and how importantly the quality of the reconstructed gradients comes into play.

A perfectly-reconstructed gradient will have terms in the expansions (3.84) corresponding to the exact Taylor series expansion. In that case, the b_n and c_n shown in (3.84) should be

$$b_0 = 1, b_1 = b_4 = 0, b_2 = \frac{x_g}{h}, b_3 = \frac{y_g}{h}$$
$$c_1 = 1, c_0 = c_2 = 0, c_3 = \frac{x_g}{h}, c_4 = \frac{y_g}{h} \tag{3.86}$$

where the $(x, y)_g$ is taken to be the midpoint of each of the faces. By application of discrete forms of the divergence theorem, one can see that this will always guarantee a second-order accurate Laplacian upon an arbitrary mesh. What is also apparent, is that to obtain a higher-order-accurate Laplacian, a quadrature of higher order on the control volume faces is also needed.

The behavior of the non-linearity-, linearity- and quadratic-preserving schemes is readily apparent by examination of the reconstructed gradients in terms of the discrete Laplacian formula. If a non-linearity preserving gradient is formed, then $b_0 \neq 1$, $c_1 \neq 1$, $b_1 \neq 0$ and $c_0 \neq 0$, which, unless a very fortunate geometric cancellation occurs, will not even yield an inconsistent Laplacian. This implies that a Laplacian can be obtained that will never yield a grid converged solution, even to the wrong equation, since the expanded Laplacian will contain weights to order 1/h. If linearity-preserving gradients are formed, then at least local grid convergence can be achieved, but consistency is not guaranteed. In this case, a linearity preserving reconstruction will give $b_0 = c_1 = 1$ and $b_1 = c_0 = 0$. A quadratic-preserving reconstruction is then seen to be the only type of reconstruction to guarantee a first-order accurate Laplacian on arbitrarily distorted meshes. From this viewpoint, the choice of flux functions appears to be a simple one; namely, choose the quadratic reconstruction procedure.

But, this analysis of the accuracy of the Laplacian leaves out a very important and practical aspect: positivity of the stencil. Positivity is a difficult thing to prove on general meshes, since as can be seen in (3.85), the discrete Laplacian involves not only the geometry of the cell faces, but also the spatial locations of the surrounding cells, and their weights in the reconstructed gradients. If simplifications are made regarding the connec-

tivity and shape of the mesh, statements about positivity of the stencil may be proven. This is shown in [7] for finite-volume solutions of Laplace's equation upon triangular/tetrahedral meshes (in two and three dimensions) using a vertex based scheme with a special co-volume upon which conservation is maintained. This procedure is shown to be equivalent to a Galerkin formulation using linear finite-elements with a lumped mass matrix. In two dimensions, a Delaunay mesh is sufficient to ensure positivity, while in three-dimensions, slightly more restrictive criteria are needed. Although positivity can be gained by a particular quality of the mesh, the accuracy of the resulting stencils is not addressed and is suspect.

This whole argument seems to indicate opposing forces at work: accuracy versus positivity. Can strictly positive and accurate stencils be obtained on generally distorted meshes? Or, can one relax the requirement upon one, say accuracy, at the expense of the other, positivity? A positive, yet highly inaccurate scheme can be obtained by using only face data of the cells. But, as is pointed out above, this can yield a dangerously inconsistent stencil since the reconstructed gradients will not be ensured to be at least linearity preserving. Perhaps then, a linearity-preserving scheme is the lowest form which one can use. But, the question is then, how bad will it be to try a higher-order reconstruction, as in the quadratic reconstruction scheme?

It is noteworthy to point out that when solving the Navier-Stokes equations, the positivity of the update to the cell contains contributions from both the inviscid and viscous operators. For high Reynolds numbers, the issue of positivity of the viscous operator might not be as dramatically evident as for lower Reynolds numbers. Since the viscous terms are scaled inversely with the Reynolds number, the positivity of the inviscid operator can mask the non-positivity of the viscous stencil. This is a dangerously deceptive scenario: Although stability might be obtained, the violation of the local maximum principle is still evident in the viscous terms.

The two best schemes from the preceding analysis are used in the next chapter to compute low/moderate Reynolds number flows corresponding to some standard viscous test problems. The two schemes are chosen based upon the qualities of either positivity or accuracy. The Green-Gauss reconstruction scheme with the linearity-preserving vertex weighting is chosen to represent the less accurate, yet more positive scheme. The quadratic reconstruction scheme is then chosen to represent the more accurate, yet less positive scheme.

The analysis shown in this chapter sheds light onto the behavior of the schemes, but only on simplified grid topologies. Since many different topologies are possible, a more general method is needed that can discretely compare the two reconstruction procedures, but on arbitrary meshes. In Appendix B, methods to form the discrete Laplace operators on arbitrary meshes for the candidate reconstruction schemes are derived. Once these operators are formed, a local Taylor series expansion yields an estimate of the local error while positivity of the operator is assessed by examination of the coefficients in the stencil. For the diamond-path reconstruction, general conditions for positivity in terms of mesh geometry are derived and presented in Appendix B. The discrete accuracy analysis can explain much of the behavior of the two schemes, but the best proof, as always, lies in the actual use of the procedures to compute actual flows: The next chapter performs a detailed comparison of the results of computations using both of the candidate schemes.

CHAPTER IV
Adaptively-Refined Solutions of the Navier-Stokes Equations Using a Cartesian, Cell-Based Approach

This chapter compares results obtained using the two candidate viscous flux functions for some well-known flow fields. Where solutions are available, and differences are noticeable, the diamond-path reconstruction scheme and the quadratic reconstruction scheme results are directly compared to each other and to theory, experimental data, or accepted computational results. But first, some important procedures in obtaining the results are highlighted.

4.1 Implementation Specifics

Development of a working flow solver following the methods described requires many ancillary procedures and operations to work properly. As in most complicated algorithms, it is the specifics of the details that make or break the assembled components. This section will outline a few things that have been used here that otherwise would be left out of the description. Although this list is not complete, the important pieces are shown.

4.1.1 Time-Step Calculation

Some important information can be found by examining the positivity of the assembled inviscid and viscous operators. Consider solving a convection-diffusion equation, representative of the x-momentum equation, as

$$\rho \frac{\partial u_0}{\partial t} + \nabla \bullet F = \frac{\mu}{Re} \nabla^2 u_0 \tag{4.1}$$

Integrating over the control volume, and dividing by the area yields

$$\frac{\partial u_0}{\partial t} = -\frac{1}{\rho A}\oint_S F \bullet n dS + \frac{\mu}{\rho Re} L(u_0) \tag{4.2}$$

where L(u) is the discrete Laplacian. If the inviscid flux balance is approximated using a first-order upwind flux, further simplified to be based on the maximum wave speed only, as

$$F = \sigma_f^+ u_0 + \sigma_f^- u_f \tag{4.3}$$

where the modified wave speeds are

$$\sigma_f^{\pm} = \frac{(|\sigma| \pm \sigma)}{2} \tag{4.4}$$

the semi-discrete form of the model equation becomes

$$\frac{\partial u_0}{\partial t} = -\frac{1}{A}\sum_{f=1}^{faces} (\sigma_f^+ u_0 + \sigma_f^- u_f)\Delta S_f + \frac{\mu}{\rho Re}\left(\alpha_0 u_0 + \sum_{n=1}^{N} \alpha_n u_n\right) \tag{4.5}$$

Discretizing in time with a simple Euler approximation and grouping terms

$$u_0^{n+1} = u_0^n\left[1 - \Delta t\left(\sum_{f=1}^{faces}\frac{\sigma_f^+}{A} - \frac{\alpha_0 \mu}{\rho Re}\right)\right] - \sum_{f=1}^{faces}\frac{\Delta t \sigma_f^-}{A} u_f + \frac{\mu \Delta t}{\rho Re}\sum_{n=1}^{N}\alpha_n u_n \tag{4.6}$$

Requiring the new value of u_0 to be bounded by the data used to compute it, a positivity constraint upon each of the terms pre-multiplying u results. By construction, all of the $\sigma_f^+ \geq 0$ and $\sigma_f^- \leq 0$. Positivity of the Laplacian ensures that all the α_n are positive and consistency then requires $\alpha_0 < 0$. Requiring positivity of the bracketed terms results in the time step

$$\frac{\Delta t}{A} = \frac{1}{\sum_{f=1}^{faces} \sigma_f^+ - \frac{\mu \alpha_0 A}{\rho Re}} \tag{4.7}$$

This shows a clear differentiation between the inviscid and viscous terms limiting the time step. In practice, as is shown in [51], it is best to combine the two in the following way

$$\frac{\Delta t}{A} = C_n \left(\frac{\Delta t_i \Delta t_v}{\Delta t_i + \Delta t_v} \right) \tag{4.8}$$

where Δt_i and Δt_v represent the inviscid and viscous time steps.

$$\frac{\Delta t_i}{A} = \frac{1}{\sum_{f=1}^{Faces} \left(\left| u \hat{n}_x + v \hat{n}_y \right| + a \right) \Delta S_f} \tag{4.9}$$

$$\frac{\Delta t_v}{A} = \frac{K_v}{-\frac{A\mu}{\rho Re} \alpha_0} \tag{4.10}$$

For safety's sake, K_v in (4.10), as inspired by [51], is taken to be $K_v = 0.25$. The importance of performing the discrete Laplacian analysis is now apparent, from the inclusion of the true stencil weight, α_0 in (4.10).

4.1.2 CFL Cut-Back Procedure

Typically, most calculations are begun with initial conditions corresponding to uniform flow at the reference state. This can cause severe start-up problems for flows around realistic geometries, where a large transient in the residuals can cause negative pressures and temperatures, which can sometimes kill the calculation. To overcome this problem, a CFL cutback procedure is used, which limits the maximum relative change in density and pressure per time step. At the beginning of each time step, the maximum relative change in

pressure and density over the grid is found for a Courant number of one. That is, if the residuals represent the flux balances divided by the cell area, then the change in the conserved variables is

$$\Delta \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \Delta \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} = \Delta t|_{C_n = 1} \begin{bmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{bmatrix} \tag{4.11}$$

The relative changes in density and pressure are then

$$\varepsilon_\rho = \frac{\Delta \rho}{\rho} = \frac{\Delta q_0}{q_0} \tag{4.12}$$

$$\begin{aligned} \varepsilon_P &= \frac{\Delta P}{P} \\ \Delta P &= (\gamma - 1)\left[\Delta q_3 - (u \Delta q_1 + v \Delta q_2) + \Delta \rho \frac{(u^2 + v^2)}{2}\right] \end{aligned} \tag{4.13}$$

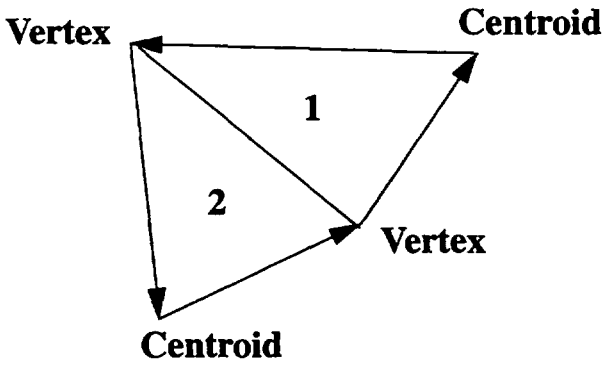
Requiring the maximum change per time step in either the normalized density or pressure to be less than some specified tolerance, ε_{cut}, the Courant number may be cut back by

$$\begin{aligned} C_n &= min\,(\tilde{C}_n, C_{n,\,max}) \\ \tilde{C}_n &= \frac{\varepsilon_{cut}}{\varepsilon} \\ \varepsilon &= max\,(\varepsilon_\rho, \varepsilon_P) \end{aligned} \tag{4.14}$$

where $C_{n,\,max}$ is the maximum allowable Courant number for the time stepping scheme and ε is taken to be the maximum relative change in density or pressure over the entire grid. Typically, $\varepsilon_{cut} = 0.1$ seems to work well. Usually, if a CFL cut back is needed, it only appears in the early stages of the calculation, and the Courant number quickly increases back to the maximum allowed.

4.1.3 Viscous Gradients Reconstruction Procedure

The heart of the reconstruction of the viscous gradients for the diamond path reconstruction is an application of the divergence theorem to a four-sided polygon. Since the reconstruction involves the line integral of quantities around this polygon, this single path can be broken into two paths around two triangles which share a common face. The common face is taken to be the face separating the two cells. Since the gradient in each of these triangles is easily found from (3.75), they are combined to give the gradient in the entire co-volume.



$$\nabla u = \frac{\nabla u_1 A_1 + \nabla u_2 A_2}{A_1 + A_2}$$

Figure 4.1 Path Integration for Gradient Reconstruction

4.1.4 No-Slip Boundary Conditions

No-slip boundary conditions are applied to the cut cells on the no-slip boundaries by specifying the two velocity components to be zero and the wall temperature to be a specified value. The components of the inviscid flux at the boundary is then simply

$$F = P\begin{bmatrix} 0 \\ \hat{n}_x \\ \hat{n}_y \\ 0 \end{bmatrix} \tag{4.15}$$

Evaluation of the viscous flux at the boundary requires the gradients to be evaluated at the boundary face. The gradients at the face can be reconstructed in a few different manners. The simplest is accomplished by performing a line integral around the path formed by the two vertices on the boundary and the cell centroid. Since this path is a triangle, the gradients are easily found from (3.75), and is termed a local triangular reconstruction.

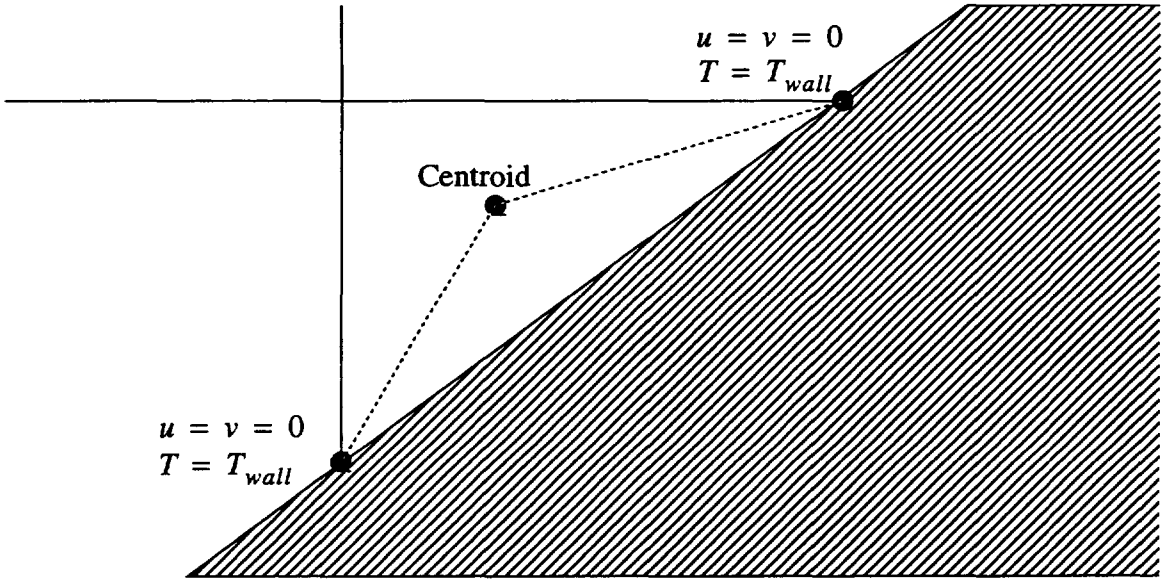


Figure 4.2 No Slip Boundary Conditions: Local Triangular Reconstruction

Unless otherwise noted, the wall temperature is always specified, so the gradients of temperature for the heat fluxes into the wall are found using the same procedures.

4.2 A Practical Comparison of the Two Candidate Reconstruction Schemes

The two candidate reconstruction schemes are used to compute some low and moderate Reynolds number flows; two low Reynolds number flows in a driven cavity and two low Reynolds number flows over backward facing steps. A higher Reynolds number flow over a flat plate is computed, and the orientation of the flat plate with respect to the base coordinate axes is changed to bring out the effects of cell cutting. For each case, the Laplacian stencils are examined on each grid for accuracy and positivity, and the computed results from the two schemes are directly compared to each other and to theory or experimental data. Finally, to demonstrate the geometric complexities that may be gridded and computed with the Cartesian cell scheme, the flow through a branched duct with cooling fins is computed.

4.2.1 Driven Cavity Flow

The laminar flow in a square, driven cavity is computed and compared to the computational data of Ghia et. al. [28]. A schematic of the geometry and boundary conditions is shown in Figure 4.3.

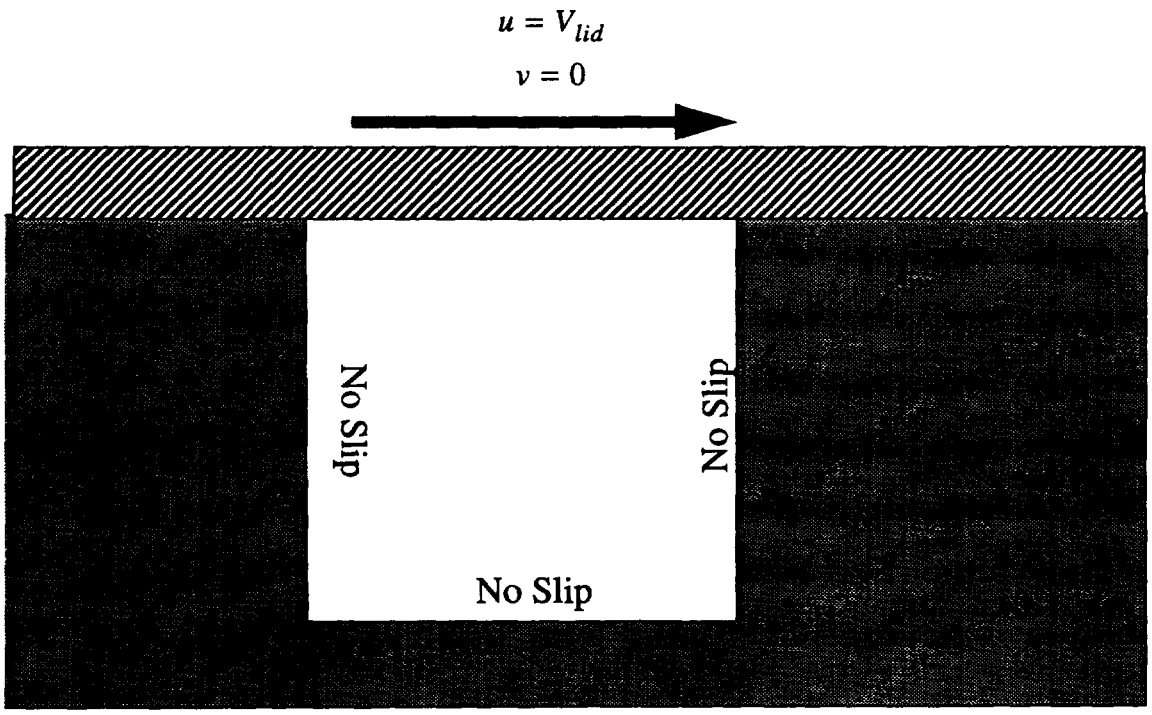


Figure 4.3 Schematic of Driven Cavity Flow

In [28] an incompressible formulation of the Navier-Stokes equations was solved using an implicit multi-grid method, where tabulated u- and v-velocity data is supplied along the lines through the geometric center of the cavity. The computed data were obtained on a 129 by 129 unstretched grid. To compare with these incompressible results, the Mach number used here is taken to be $M_{lid} = 0.1$ so that the non-dimensionalizing Reynolds number is related to the lid Reynolds number as

$$Re_{\infty} = \frac{Re_{lid}}{M_{\infty}} \tag{4.16}$$

The moving cavity lid sets up a strong vortex in the cavity, which induces smaller, secondary (and for the high Reynolds numbers, tertiary) vortices in the corner regions. Data are supplied for Reynolds numbers of $Re_{lid} = 100, 400, 1000, 3200, 5000, 7500$ and 10,000 [28]. Computations are made here using adaptive mesh refinement for the two

reconstruction schemes for the Reynolds numbers of 100 and 400. The results are compared to each other and to the computational results of [28]. The quality of the grids and the accuracy of the two reconstruction procedures are assessed by tabulating the discrete accuracy and positivity parameters outlined in Appendix B.

4.2.1.1 Re=100

A uniform base grid of 1024 cells (32 by 32) is generated, and three levels of adaptive mesh refinement beyond the base grid are obtained for both schemes. Both schemes do an excellent job, even on the coarse base grid: Adaptive mesh refinement improves the solution slightly, but the initial solution is quite good. Figure 4.4 and Figure 4.5 show the computed u- and v-velocity profiles along vertical and horizontal lines through the geometric center of the cavity for the diamond path scheme (the number of cells at each refinement level is shown in parenthesis). Figure 4.6 shows the final adapted grid and Figure 4.7 shows contours of u-velocity.

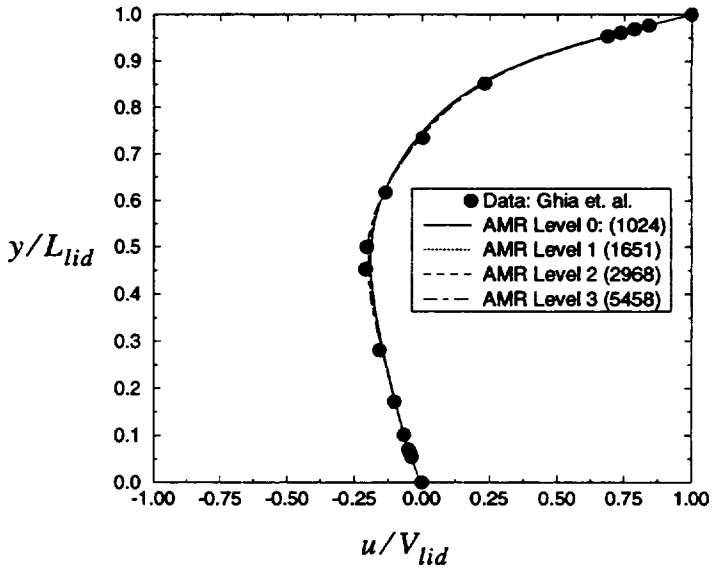


Figure 4.4 u-velocity Along Vertical Line Through Geometric Center

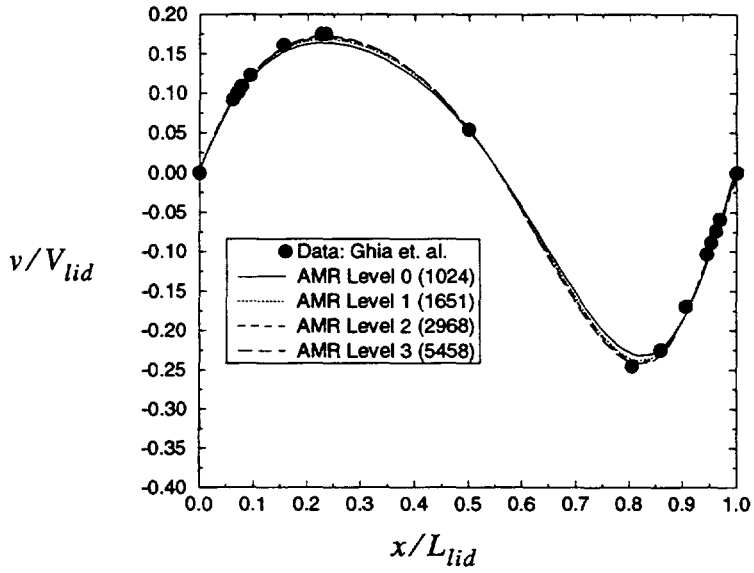


Figure 4.5 v-velocity Along Horizontal Line Through Geometric Center

As is seen in the adapted grid, the mesh-adaptation strategy resolves the singularity in the u-velocity at the corners at the expense of not resolving other important regions of the flow: There are two secondary vortices induced in the corners of the cavity, as shown in Figure 4.8.

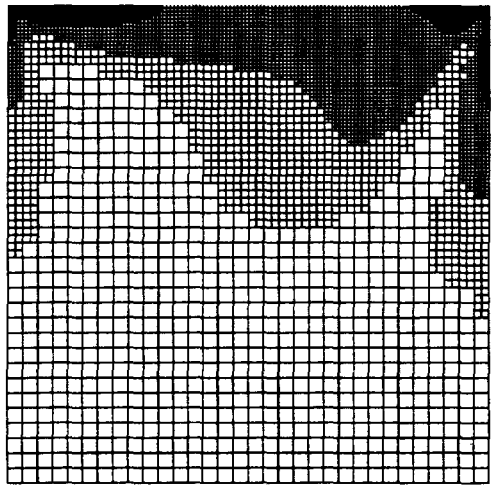


Figure 4.6 Refinement Level 3 Adapted Grid, Re=100 Case

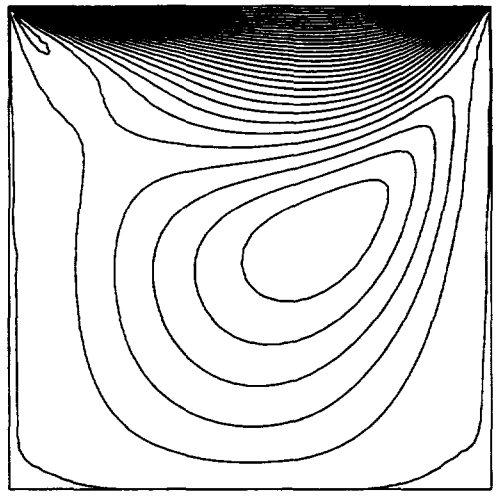


Figure 4.7 Refinement Level 3, u-velocity Contours

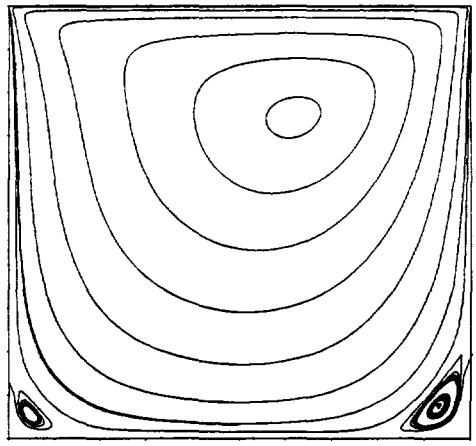


Figure 4.8 Particle Paths: Level 3 Grid

There is a negligible difference between the results computed by the two viscous-flux schemes, so there are no comparison plots shown here. It is encouraging to see that both schemes performed well, and it is interesting to see that there is little difference between the results from the two reconstruction procedures. The lack of difference between the two schemes can be attributed to the good quality of stencils that both schemes create on the grids. Table VI and Table VII characterize the global non-positivity of the mesh and the

non-positivity of the worst stencils on the mesh by showing the L_1 norm of the positivity parameter, (3.45),over all cells analyzed and the minimum of all the positivity measures.

Table VI Discrete Positivity Measures, Quadratic Reconstruction Scheme, Re=100 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	0.000e+00	0.000e+00
1	0.000e+00	0.000e+00
2	-1.584e-03	-5.423e-01
3	-4.215e-05	-1.069e-01

Table VII Discrete Positivity Measures, Diamond Path Reconstruction Scheme, Re=100 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	0.000e+00	0.000e+00
1	-6.627e-03	-3.580e-01
2	-1.107e-02	-3.580e-01
3	-1.108e-02	-3.580e-01

The quadratic scheme generates fewer stencils with negative weights, but has the most non-positive stencil of the two schemes. Both schemes yield adequate stencils for the grids, which they should as the grids are regular due to no cell cutting, and the only non-smoothness is incurred across refinement boundaries.

The accuracy is evaluated next by comparing the sums in the discrete Taylor series expansion to the values that should be obtained for a consistent Laplacian. For each cell, the stencil weights are used to compute the sums in (3.30) to (3.35), and the L_1 and L_∞

norms of the differences between the sums and the values required for a consistent Laplacian are made. As an example, the sum of $\sum \alpha_n x_n^2$ should be equal to two if the Laplacian is consistent in the u_{xx} term. So, to measure the deviation of the stencil from this, the norms of $\sum \alpha_n x_n^2 - 2$ over the grid are found. If all of the stencils were consistent, these norms would be identically zero. Since both schemes are at least linearity preserving, all of the terms to order 0 and 1 in the sums, $\sum \alpha_n$, $\sum \alpha_n x_n$ and $\sum \alpha_n y_n$ sum to zero. The quadratic-reconstruction scheme, by construction, yields stencils that are consistent: This behavior is replicated in the discrete accuracy analysis as the sums described above are zero to machine precision. The results for the diamond-path scheme, which cannot guarantee consistency, are shown in Table VIII.

Table VIII Discrete Accuracy Analysis: Diamond-Path Reconstruction, Re=100 Grids

Mesh Level	L_1 of $\sum \alpha_n x_n^2 - 2$	L_1 of $\sum \alpha_n x_n y_n$	L_1 of $\sum \alpha_n y_n^2 - 2$	L_∞ of $\sum \alpha_n x_n^2 - 2$	L_∞ of $\sum \alpha_n x_n y_n$	L_∞ of $\sum \alpha_n y_n^2 - 2$
0	0.00e+00	3.33e-25	0.00e+00	0.00e+00	1.78e-15	0.00e+00
1	2.43e-02	-1.41e-04	2.54e-02	9.53e-01	8.60e-01	9.53e-01
2	3.83e-02	8.96e-06	4.08e-02	9.53e-01	8.60e-01	9.53e-01
3	3.95e-02	-7.89e-05	4.06e-02	9.53e-01	9.17e-01	9.53e-01

The results indicate that in a global sense the stencils obtained by the diamond path reconstruction are good, and the error in consistency is low and would be difficult to notice. What is also indicated is that there are a few bad cells in the grid, but globally the error is quite low.

4.2.1.2 Re=400

The conditions for this case are identical to the first, except that the Reynolds number based on the lid speed and cavity depth is 400. A base grid is generated and 3 levels of adaptive mesh refinement are performed for both schemes. As in the previous case, both schemes are compared to each other and to the data in [28]. For this case, the base grid solution is poor, but the adaptive mesh refinement improves the solution quality progressively with each refinement level. Figure 4.9 and Figure 4.10 show the u- and v-velocities compared to the computational data in [28] (the number of cells at each refinement level is shown in parenthesis). As is seen in a plot of the grid at the final refinement level, the adaptive-mesh refinement strategy has focused the refinement upon the singularities in velocity at the upper corners, but does resolve the shear well interior to the domain. Although there is room for improvement in the adaptation criteria, the criteria that were developed for inviscid flows performs adequately.

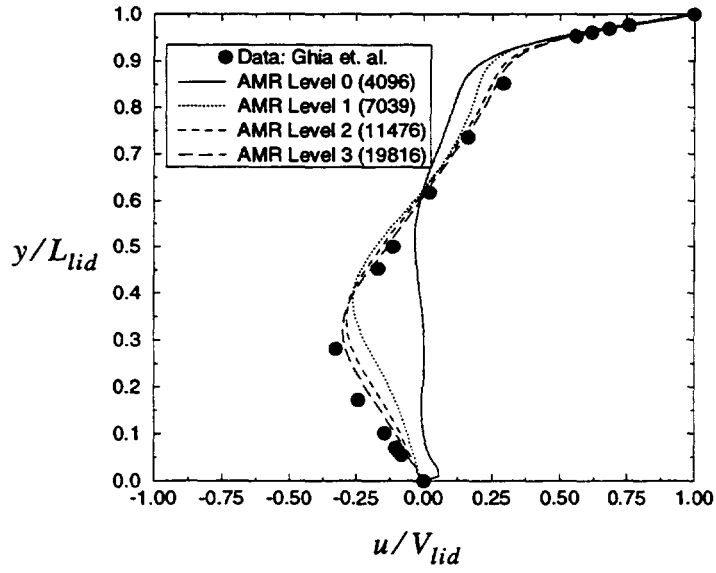


Figure 4.9 u-velocities on Vertical Line Through Geometric Center, Re=400

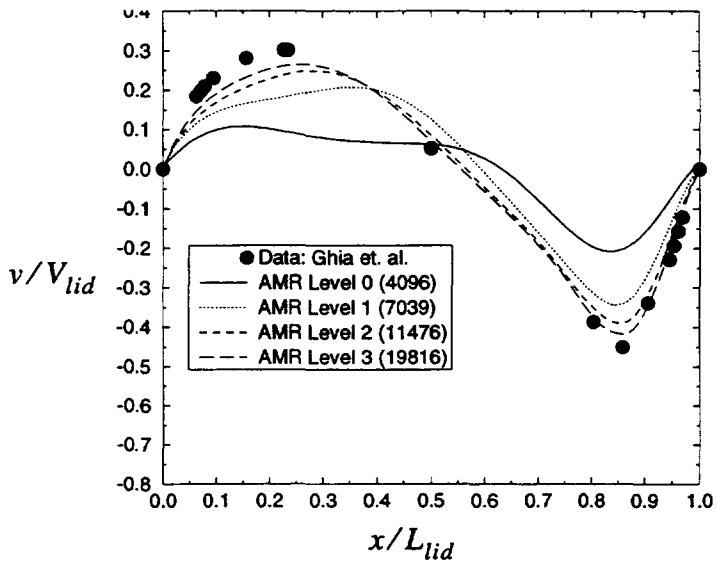


Figure 4.10 v-velocities on Horizontal Line Through Geometric Center, Re=400

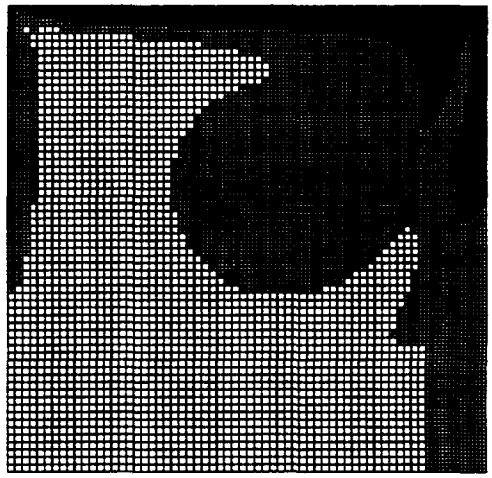


Figure 4.11 Final Adapted Grid, Re=400 Driven Cavity

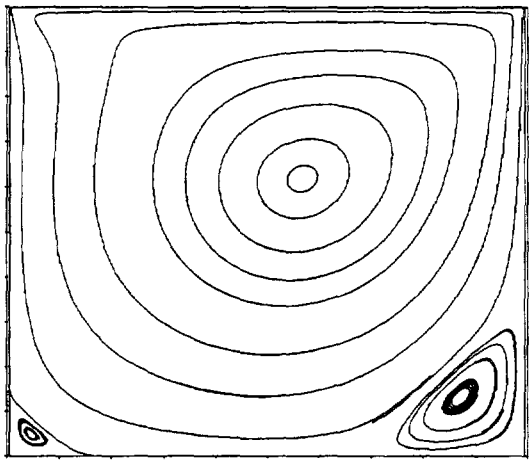


Figure 4.12 Particle Paths, Re=400 Driven Cavity

There is negligible difference between the results of the two viscous flux construction schemes. This is directly attributable to the quality of the stencils that both schemes create on the grids. Table IX and Table X show the discrete positivity measures for the schemes.

Table IX Discrete Positivity Measures, Quadratic Reconstruction Scheme, Re=400 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	0.000e+00	0.000e+00
1	0.000e+00	0.000e+00
2	-9.962e-05	-5.425e-01
3	-6.566e-05	-3.962e-01

Table X Discrete Positivity Measures, Diamond-Path Reconstruction Scheme, Re=400 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	0.000e+00	0.000e+00
1	-3.548e-03	-3.576e-01
2	-6.477e-03	-3.580e-01
3	-7.003e-03	-3.682e-01

For these grids, the diamond path scheme produces many more non-positive stencils than the quadratic scheme, but the scheme with the most non-positive stencil is the quadratic reconstruction. The diamond-path reconstruction scheme can not guarantee consistency, but the quadratic scheme does: The accuracy norms for the quadratic reconstruction are zero to machine precision. As before, the diamond path scheme is strictly inconsistent, but in a global sense, this inconsistency is low.

Table XI Discrete Accuracy Measures, Diamond Path Reconstruction, Re=400 Grids

Mesh Level	L_1 of $\sum\alpha_n x_n^2 - 2$	L_1 of $\sum\alpha_n x_n y_n$	L_1 of $\sum\alpha_n y_n^2 - 2$	L_∞ of $\sum\alpha_n x_n^2 - 2$	L_∞ of $\sum\alpha_n x_n y_n$	L_∞ of $\sum\alpha_n y_n^2 - 2$
0	0.00e+00	-1.64e-33	0.00e+00	0.00e+00	3.55e-15	0.00e+00
1	1.23e-02	9.80e-05	1.22e-02	9.53e-01	8.60e-01	9.53e-01
2	2.18e-02	9.65e-05	2.34e-02	9.53e-01	8.60e-01	9.53e-01
3	2.45e-02	5.81e-05	2.53e-02	9.53e-01	9.17e-01	9.53e-01

4.2.2 Backward-Facing Step Flows

The low Reynolds number flow over a backward-facing step (sudden expansion) is computed using the Cartesian cell-based approach. The results are compared to the experimental results of [3] at Reynolds numbers of 100 and 389, based on the pre-step hydraulic diameter and mass flow rate. That is,

$$Re = \frac{V(2h)}{\nu_\infty} \tag{4.17}$$

where V is the mass averaged flow rate, which for the fully developed profile entering the channel is $V = \frac{2}{3}u_{max}$. For both Reynolds number calculations, the boundary conditions are applied as indicated in Figure 4.13.

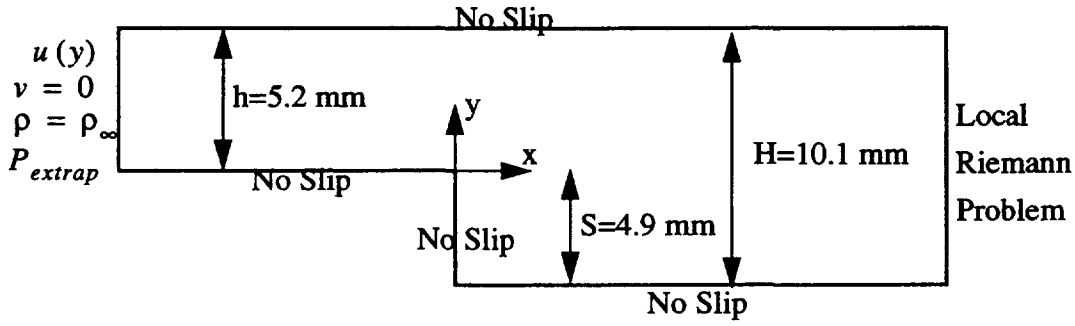


Figure 4.13 Backward Facing Steps Schematic

The inflow velocity profile is fully developed

$$\begin{aligned} u(y) &= u_{max}(1-\eta^2) \\ \eta &= 1+\frac{2}{h}(y-h) \end{aligned} \tag{4.18}$$

The pressure at the inflow is extrapolated to first-order from the interior, from which the flux is directly found. A local Riemann problem is solved at the outflow boundary, using the inviscid numerical flux function, where the left state is taken as the cell average of the

cell on the boundary, equivalent to a first-order flux construction, while the right state is obtained from a parabolic velocity profile, (constant) freestream density, and a specified (constant) back pressure. Since the inflow pressure is allowed to adjust and the mass-flow rate is fixed, the proper streamwise pressure gradient on the flow is provided. No-slip boundary conditions are applied on all walls, with a specified temperature equal to that at reference conditions. The reference conditions are chosen so that the Mach number of the maximum velocity of the inflow profile is 0.2.

4.2.2.1 Re=100

A coarse base grid is generated, and both schemes converge the solution through the requested three levels of adaptive mesh refinement. Figure 4.14 shows the convergence histories of the two schemes. Adaptive mesh refinement is made according to the convective criteria presented in Chapter II, and no attempts are made to modify the criteria. The mesh refinement improves the solution through each level of refinement, and both schemes perform well and yield nearly identical results. Figure 4.15 shows the improvement of the solution due to mesh refinement for the diamond-path reconstruction scheme, at a given location in the backstep, where there is a significant reversed flow region. A close-up of the adapted grid near the backstep at the final level of mesh refinement is shown in Figure 4.16. Both schemes are compared to each other and to the experimental data at a selected series of locations in Figure 4.17 to Figure 4.21.

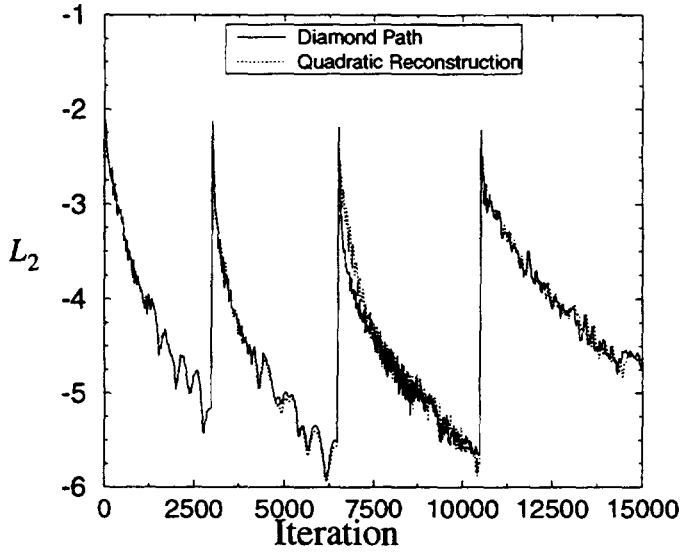


Figure 4.14 Convergence History: Re=100 Backstep

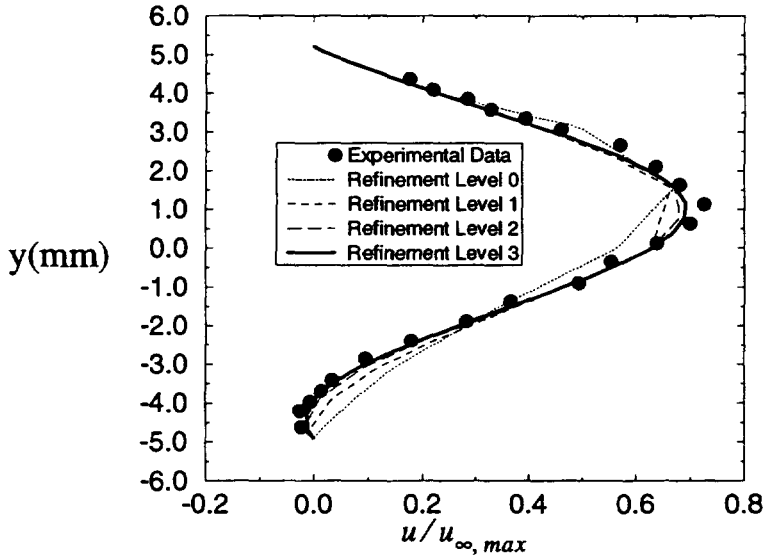


Figure 4.15 Re=100 Backstep: Comparison of Adapted Solutions to Data at x/S=2.55

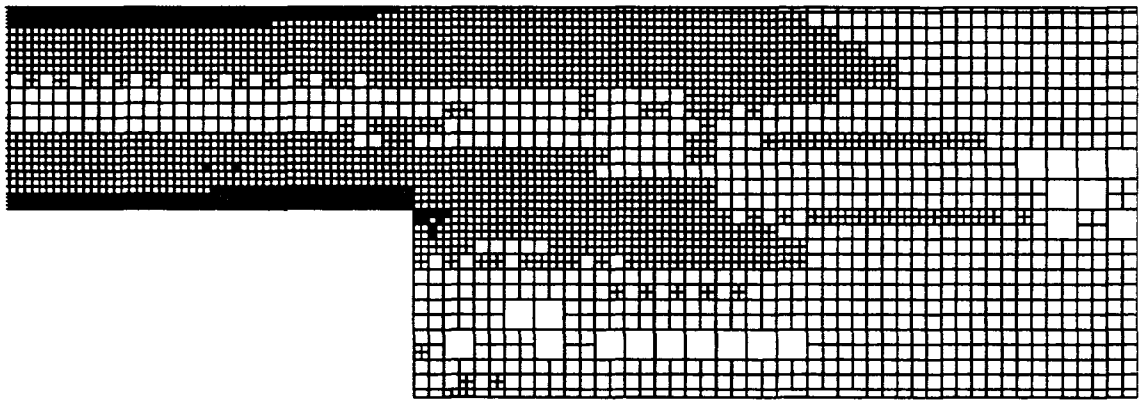


Figure 4.16 Adapted Grid at Refinement Level 3: Close-up Near Step

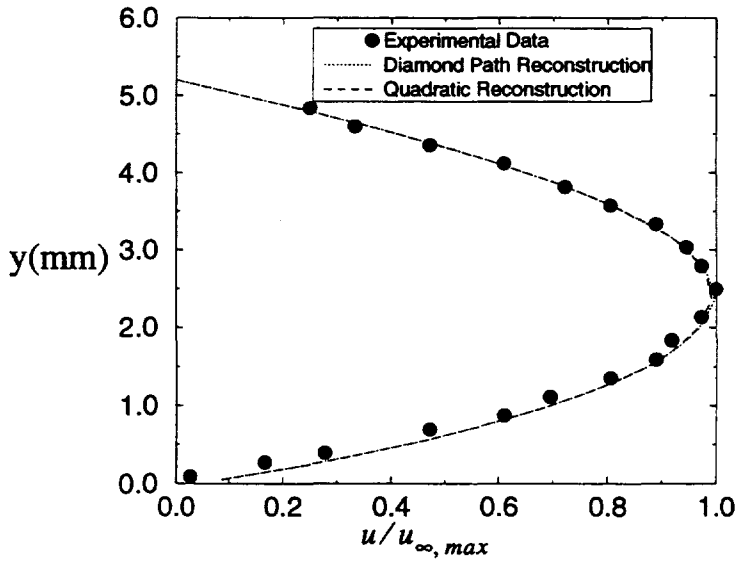


Figure 4.17 Comparison of the Diamond Path Reconstruction and Quadratic Reconstruction Computed Results to Experimental Data at x/S=0.

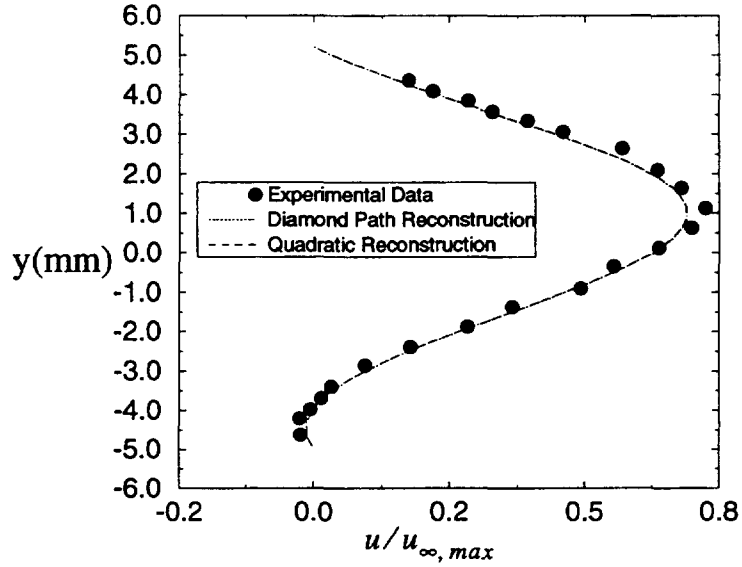


Figure 4.18 Comparison of the Diamond Path Reconstruction and Quadratic Reconstruction Computed Results to Experimental Data at x/S=2.55.

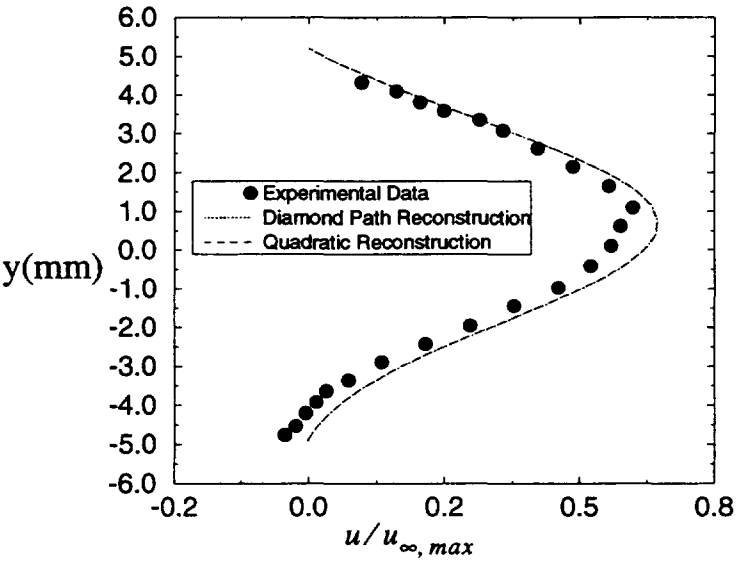


Figure 4.19 Comparison of the Diamond Path Reconstruction and Quadratic Reconstruction Computed Results to Experimental Data at x/S=3.06.

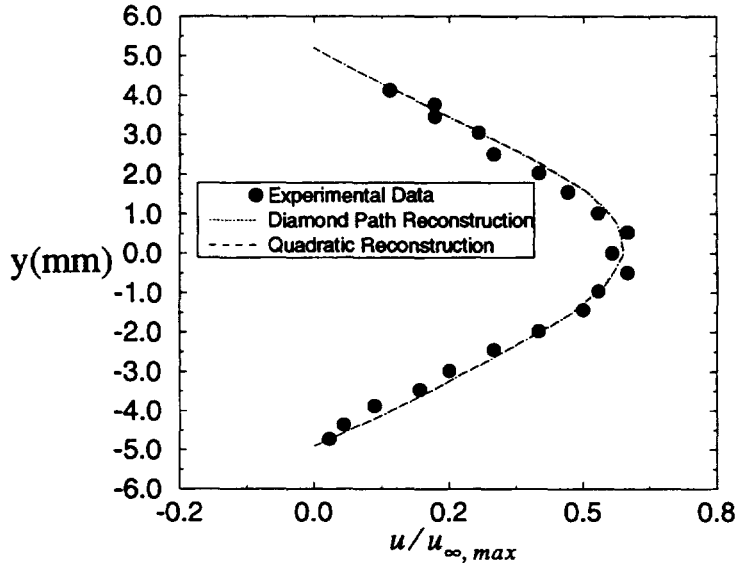


Figure 4.20 Comparison of the Diamond Path Reconstruction and Quadratic Reconstruction Computed Results to Experimental Data at x/S=4.18.

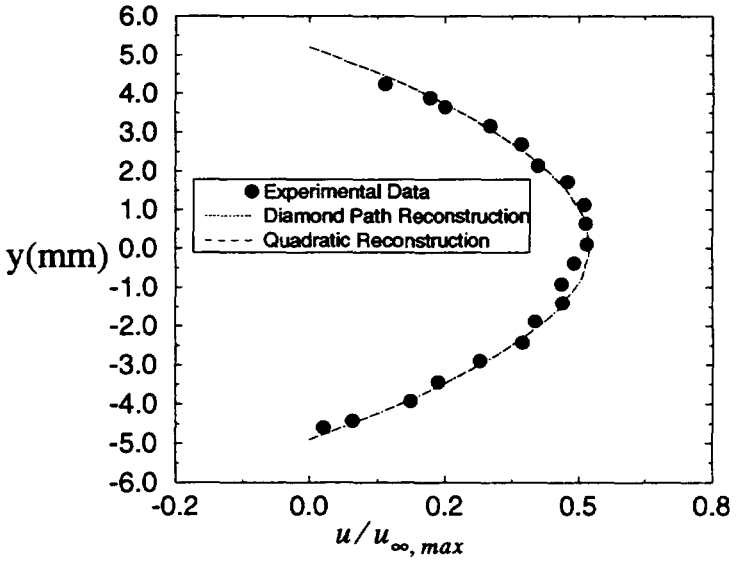


Figure 4.21 Comparison of the Diamond Path Reconstruction and Quadratic Reconstruction Computed Results to Experimental Data at x/S=0

The two viscous reconstruction schemes are analyzed as before, by comparing the dis-

crete accuracy and positivity measures on the grids obtained throughout the refinement process.

Table XII Discrete Positivity Measures for Diamond-Path Reconstruction, Re=100 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	-2.953e-02	-3.580e-01
1	-2.045e-02	-3.580e-01
2	-2.544e-02	-3.580e-01
3	-1.723e-02	-3.682e-01

Table XIII Discrete Positivity Measures for Quadratic Reconstruction, Re=100 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	-4.675e-04	-3.212e-01
1	-5.415e-03	-1.956e+00
2	-4.898e-03	-2.330e+00
3	-1.295e-02	-2.818e+00

As can be seen from these two tables, the diamond-path reconstruction scheme creates more non-positive stencils than the quadratic reconstruction scheme, but these non-positive stencils are less non-positive than the quadratic stencils. What is also shown is that the quadratic reconstruction scheme generates some very non-positive stencils, since the minimum $\tilde{\alpha}_{min}$ over the grid are approaching over 280 percent of the root mean square of the coefficients in the local stencil. Considering the magnitude of this non-positivity, it is surprising that a solution was obtained at all. Table XIV shows the discrete accuracy norms

for the diamond-path reconstruction scheme. The quadratic-reconstruction scheme guarantees consistency by construction: These norms are all at machine zero for all refinement levels for the quadratic reconstructions, and are not shown here.

Table XIV Discrete Accuracy Measure, Diamond Path Reconstruction Scheme, Re=100 Grids

Mesh Level	L_1 of $\sum\alpha_n x_n^2 - 2$	L_1 of $\sum\alpha_n x_n y_n$	L_1 of $\sum\alpha_n y_n^2 - 2$	L_∞ of $\sum\alpha_n x_n^2 - 2$	L_∞ of $\sum\alpha_n x_n y_n$	L_∞ of $\sum\alpha_n y_n^2 - 2$
0	8.49e-02	-2.64e-03	5.97e-02	9.53e-01	8.60e-01	9.53e-01
1	6.46e-02	1.88e-05	4.19e-02	1.10e+00	9.17e-01	9.53e-01
2	9.33e-02	-5.79e-08	8.80e-02	1.44e+00	9.17e-01	9.53e-01
3	6.11e-02	1.30e-05	5.52e-02	1.39e+00	9.17e-01	9.53e-01

Since the diamond-path scheme does not guarantee consistency, the discrete sums are not identically equal to the required values, but in an L_1 sense, the entire grid is nearly consistent.

4.2.2.2 Re=389

For both reconstruction schemes, the same, coarse base grid is generated, after which adaptive mesh refinement is performed. Adaptive mesh refinement is desired for three levels of refinement beyond the base grid. The diamond-path reconstruction scheme performs well, and successfully converges the solution at all refinement levels, while the quadratic reconstruction scheme is only able to converge the base and next two levels of refinement. The final, third level of refinement diverges, as will be seen, due to a set of very non-positive stencils. The following plot shows the "convergence" history of the two schemes.

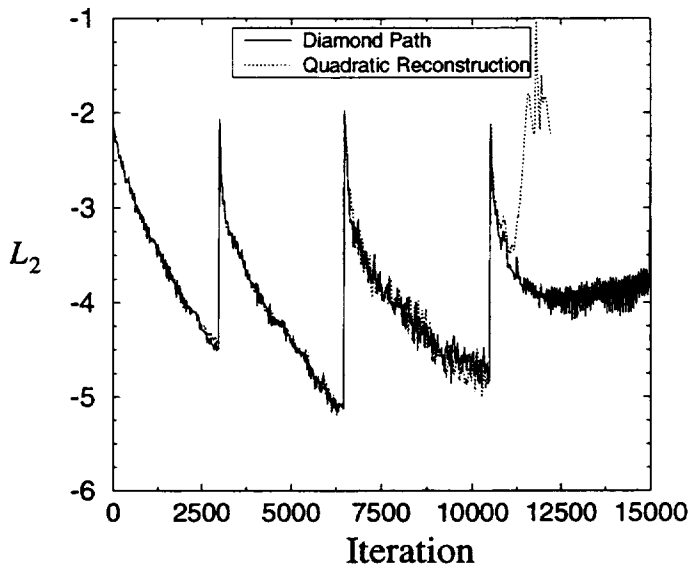


Figure 4.22 Convergence History, Re=389: Diamond Path and Quadratic Reconstructions

As in the previous case, direct comparisons between the two reconstruction procedures and between experiment are shown, in Figure 4.24 to Figure 4.28, but here the comparisons are made at the second refinement level. As before, both schemes obtain stencils that are of comparable accuracy, so that it is difficult to see any appreciable differences on the velocity plots. Figure 4.23 illustrates the improvement in the solution quality automatically obtained with the solution-adaptive mesh refinement.

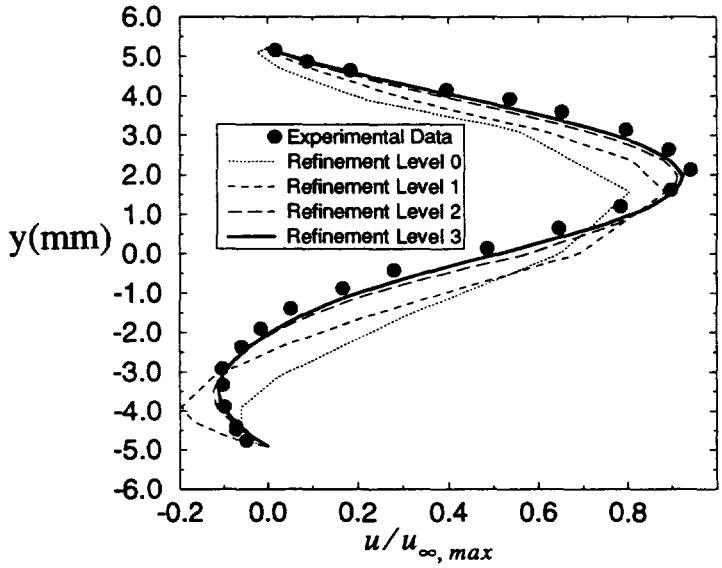


Figure 4.23 Effect of Adaptive Mesh Refinement Upon Solution at x/S=2.55, diamond-path scheme.

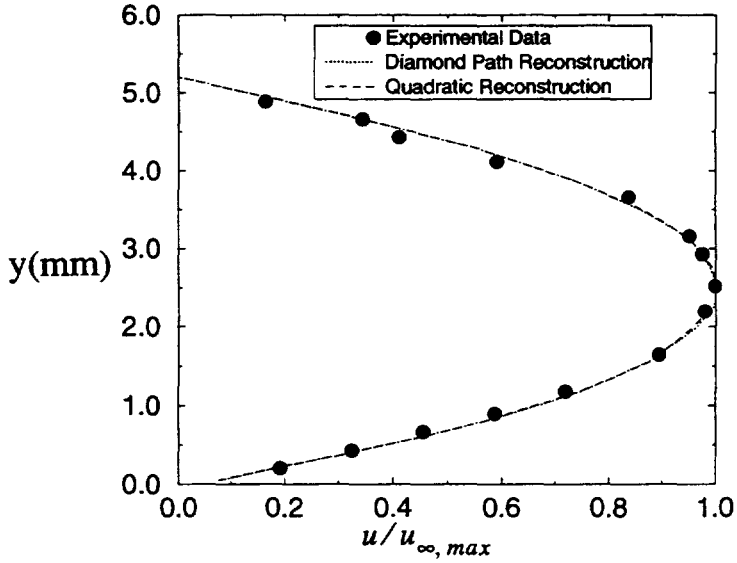


Figure 4.24 Comparison of the Diamond Path and Quadratic Reconstructions computed results to Experimental data as x/S=0, AMR Level 2.

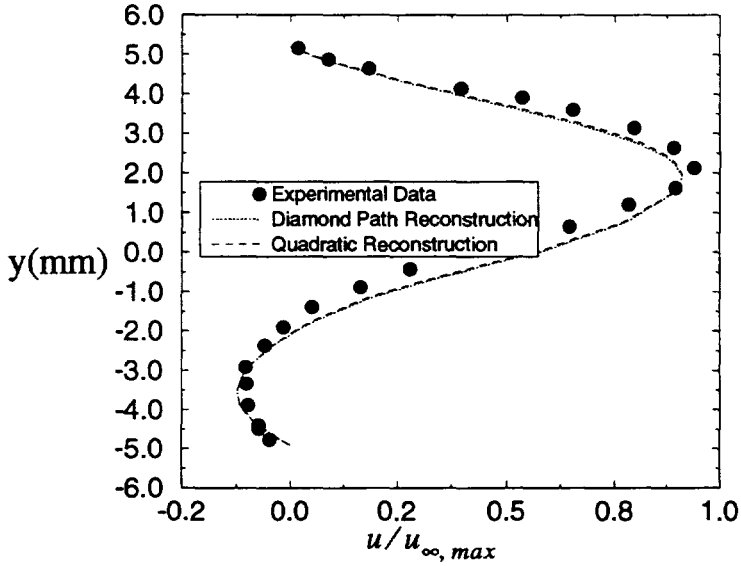


Figure 4.25 Comparison of the Diamond Path and Quadratic Reconstructions computed results to Experimental data as x/S=2.55, AMR Level 2.

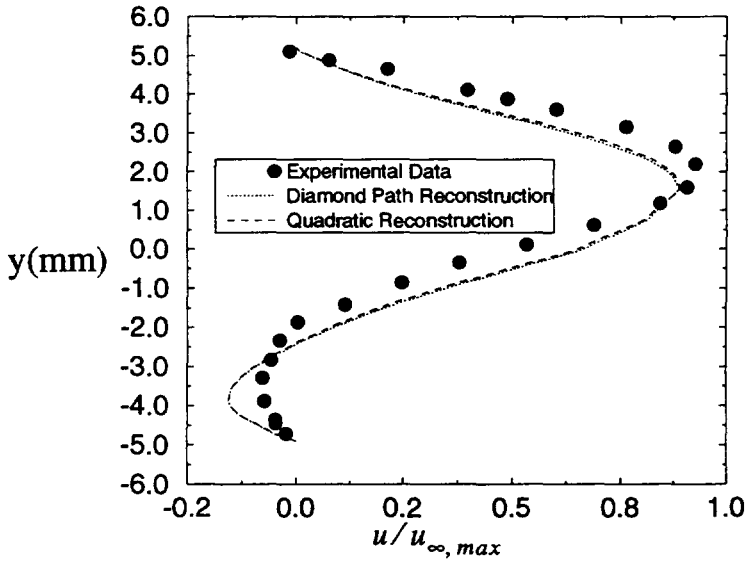


Figure 4.26 Comparison of the Diamond Path and Quadratic Reconstructions computed results to Experimental data as x/S=3.06, AMR Level 2

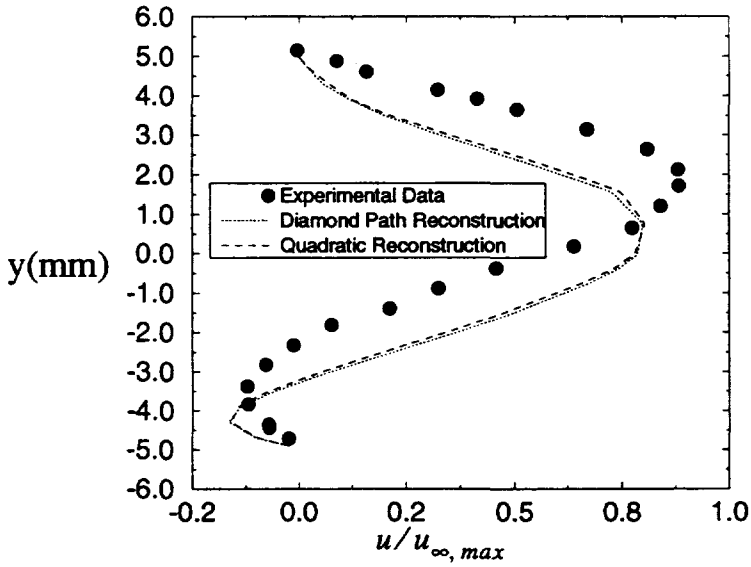


Figure 4.27 Comparison of the Diamond Path and Quadratic Reconstructions computed results to Experimental data as x/S=4.18, AMR Level 2.

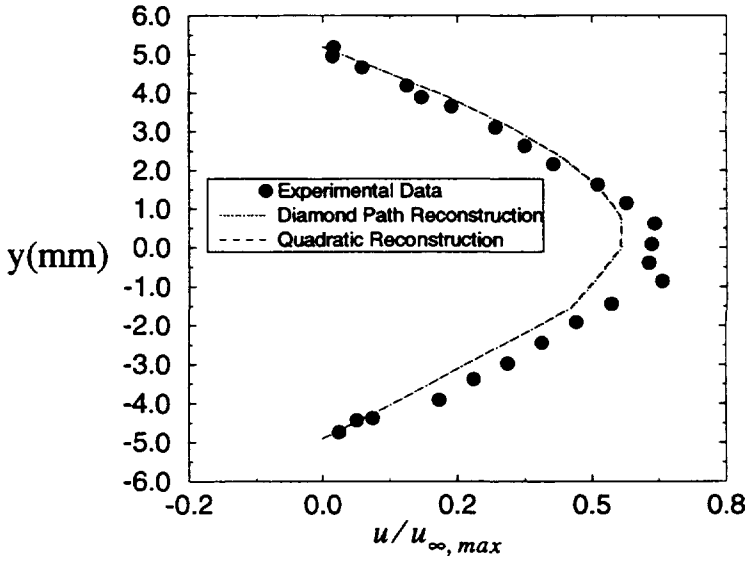


Figure 4.28 Comparison of the Diamond Path and Quadratic Reconstructions computed results to Experimental data as x/S=13.57, AMR Level 2.

There is little difference between the computed profiles from the two reconstruction schemes. What is very noticeable, though, is that the quadratic reconstruction scheme diverges on the final grid-refinement level. An examination of the norms of the positivity

measure shows similar behavior as in the lower Reynolds number case. The diamond-path reconstruction yielded more non-positive stencils than the quadratic scheme, but the non-positive stencils created by the quadratic scheme were more non-positive than the diamond path's. Table XV and Table XVI show the discrete positivity measures for the grids obtained through adaptive mesh refinement for the diamond-path and quadratic reconstruction schemes.

Table XV Discrete Positivity Measure, Diamond Path Reconstruction,Re=389Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	-2.953e-02	-3.580e-01
1	-2.040e-02	-3.580e-01
2	-2.572e-02	-3.671e-01
3	-2.216e-02	-3.682e-01

The discrete accuracy analysis shows similar trends as before, and are shown in Table

Table XVI Discrete Positivity Measure for Quadratic Reconstruction, Re=389 Grids

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min\,(\tilde{\alpha}_{min})$
0	-4.675e-04	-3.212e-01
1	-9.501e-03	-2.341e+00
2	-1.167e-02	-2.820e+00
3	-9.796e-03	-2.491e+00

XVIII. The quadratic reconstruction procedure guarantees consistency, and the diamond-path reconstruction does not. The inconsistency due to the diamond-path reconstruction is low, and the stencils are nearly consistent. Since consistency is guaranteed by the quadratic reconstruction, the accuracy norms are all at the level of machine zero, and are not

shown here.

Table XVII Discrete Accuracy Analysis, Diamond Path Reconstruction, Re=389

Mesh Level	L_1 of $\sum\alpha_n x_n^2 - 2$	L_1 of $\sum\alpha_n x_n y_n$	L_1 of $\sum\alpha_n y_n^2 - 2$	L_∞ of $\sum\alpha_n x_n^2 - 2$	L_∞ of $\sum\alpha_n x_n y_n$	L_∞ of $\sum\alpha_n y_n^2 - 2$
0	8.49e-02	-2.64e-03	5.97e-02	9.53e-01	8.60e-01	9.53e-01
1	6.78e-02	-3.25e-04	5.37e-02	1.10e+00	9.17e-01	9.53e-01
2	9.95e-02	1.08e-04	9.81e-02	1.39e+00	9.17e-01	9.53e-01
3	8.28e-02	-8.45e-05	8.12e-02	1.39e+00	9.17e-01	9.53e-01

The non-convergence of the final refinement level for the quadratic reconstruction scheme is directly attributed to a region containing some very non-positive stencils. Figure 4.29 and Figure 4.30 shows a close-up of u-velocity contours in the region, and Figure 4.31 shows the positivity measures of the stencils in the region.

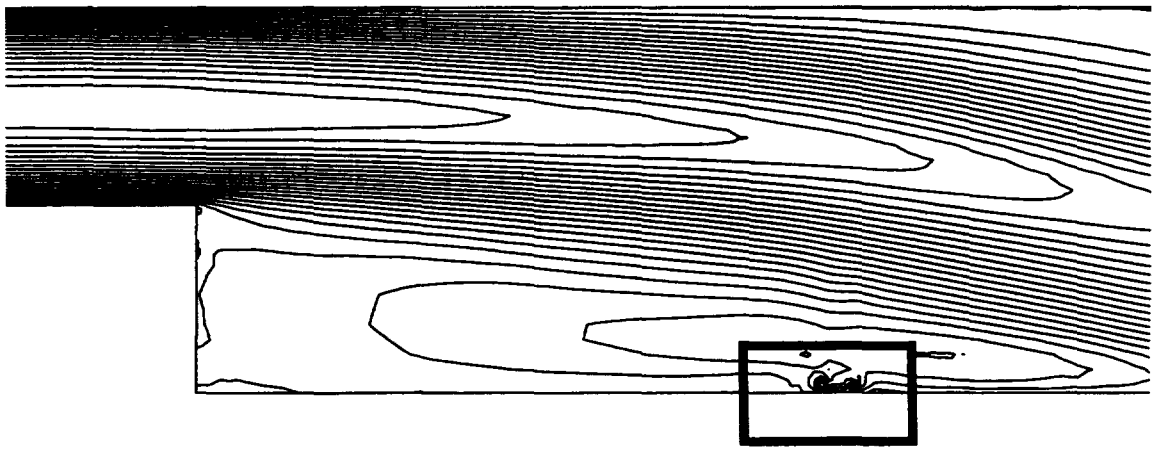
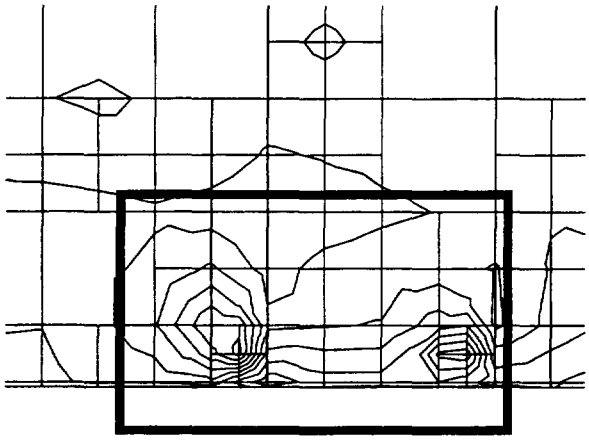


Figure 4.29 Contours of u-velocity from Quadratic Reconstruction, Level 3 AMR



See $\tilde{\alpha}_{min}$ in Figure 4.31

Figure 4.30 Close-up of Contours of u-velocity from Quadratic Reconstruction, Level 3 AMR

<table>
<tr><td>0</td><td colspan="2">0</td><td>0</td><td>0</td><td>0</td><td colspan="2">0</td><td>0</td></tr>
<tr><td>-.65</td><td colspan="2">0</td><td>0</td><td>0</td><td>0</td><td colspan="2">0</td><td>0</td></tr>
<tr><td rowspan="2">-1.94</td><td>0</td><td>0</td><td rowspan="2">-1.94</td><td rowspan="2">0</td><td rowspan="2">-1.94</td><td>0</td><td>0</td><td rowspan="2">-1.94</td></tr>
<tr><td>-.94</td><td>-.87</td><td>-.94</td><td>-.87</td></tr>
</table>

Figure 4.31 Discrete Positivity Measures, $\tilde{\alpha}_{min}$, in Problem Region of Figure 4.30

As can be seen from these figures, a set of extremely non-positive stencils are made in the recirculation region of the flow, near the reattachment point. This region of the domain is close to the reattachment point, where both the u- and v-velocities are low, yet the shear is not: positivity of the viscous operators is very important. The stencils obtained from the quadratic reconstruction procedure in this region have negative weights whose values are

nearly two times the root mean square of the weights in the stencil, which is a dangerously high value. The diamond path reconstruction procedure is not guaranteed to be positive either, but as Table XV and Table XVI show, is not nearly as non-positive as the quadratic reconstruction scheme and is able to obtain a solution through all levels of refinement.

4.2.3 Laminar, Developing Flow Over a Flat Plate: Coordinate Axes Aligned

The laminar, developing flow over a flat plate which is aligned with the freestream is next used to validate the solver. Boundary-layer theory provides a similarity solution which is used to judge the quality of the computed results. Uniform flow is imposed ahead of the plate, and the flow is allowed to develop from the leading edge. This is a more stringent test than imposing a velocity profile at some location downstream of the leading edge, and allowing the flow to develop. Since resolution near the singularity at the leading edge is directly responsible for the quality of the flow downstream, this flow solution brings to light the effects of adaptive mesh refinement.

The similarity solution is well known and is found in many introductory fluid mechanics books. The ordinary differential equation which results can not be solved in closed form, so in practice, one uses its tabulated numerical solution (see [71]). The streamfunction is formulated in terms of the similarity variable

$$\eta = \frac{y}{\sqrt{\frac{\upsilon x}{u_\infty}}} \tag{4.19}$$

which relates the u and v components to the similarity solution as

$$\frac{u(\eta)}{u_\infty} = \frac{\partial f}{\partial \eta} \tag{4.20}$$

$$\frac{v}{\sqrt{\frac{\upsilon u_\infty}{x}}} = \phi_v(\eta) = \frac{1}{2}\left(\eta\frac{\partial f}{\partial \eta} - f\right) \tag{4.21}$$

The computation is made with a free-stream Mach number taken to be $M_\infty = 0.2$, which eliminates any need for a compressibility transformation of the similarity solution. For simplicity, the reference length is taken so that at $x/l_\infty = 1$ the Reynolds number based on distance from the leading edge is 10,000. The computational domain and boundary conditions are illustrated in Figure 4.32.

\

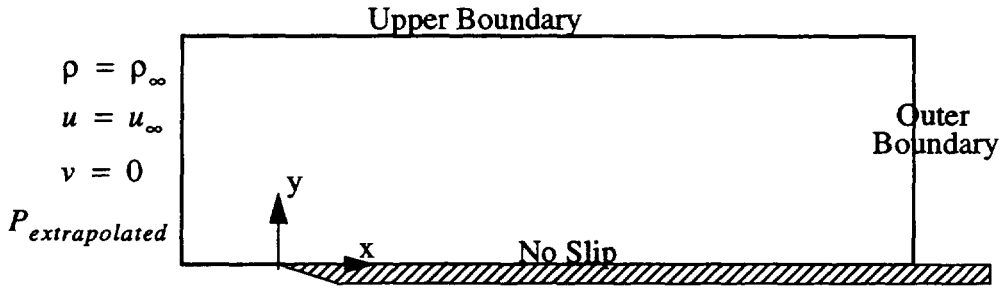


Figure 4.32 Schematic for Flat Plate Flow

At the upper, outflow and stagnation streamline boundaries the density and velocity components are extrapolated from the interior and the pressure is specified at its reference value. The root cell of the grid is located so that its south face lies along the plate surface. When care is taken in providing an acceptable base grid resolution, the results of the Cartesian-cell approach are excellent through all refinement levels. The exact solution provides a means of defining a suitable resolution of the flow for the base grid by allowing a means to specify the cut cell face lengths and an acceptable resolution normal to the plate. The allowable cut-cell face size is defined as a function of distance along the plate, guided by the exact solution. The acceptable face-size criterion for plate bounded cut cells is specified for the base grid as

$$\Delta S_{cut} = \eta_0 \sqrt{\frac{x/l_\infty}{Re_\infty M_\infty}} \tag{4.22}$$

The constant $\eta_0 = 0.2$ ensures that the first cell will be sufficiently near the wall to predict the velocity to less than approximately 10% of the free-stream value. This face size criterion is used to determine when the base grid is sufficiently resolved *at* the wall, but does not specify a desirable resolution *away* from the wall.

The exact solution of the boundary layer flow also suggests a natural means of determining a local length scale normal to the plate. Each point in the flow can be located in a plate-aligned coordinate system, from which its similarity coordinate is then found. If the plate-aligned and normal coordinates are taken to be (s,n), the value of the similarity coordinate is found as

$$\eta = \frac{n}{\delta(s)} \tag{4.23}$$

where for the flat plate boundary layer flow,

$$\delta(s) = \sqrt{\frac{s}{M_\infty Re_\infty}} \tag{4.24}$$

But, this only determines the viscous layer height, and not the variation of the length scale across it. If the desired grid should be spaced so that the maximum change in velocity normal to the plate is relatively constant, the exact solution suggests a length scale in the similarity variable as

$$\Delta\eta = \frac{\Delta u}{f''(\eta)} \tag{4.25}$$

where Δu is taken to be a constant. The physical length scale, l, is then found as

$$l = \Delta\eta\delta(s)\eta_\delta \tag{4.26}$$

where the boundary layer edge is located at $\eta_\delta \cong 8$. If a cell has a size greater than this length scale, it is tagged for refinement. This grid-smoothing procedure is implemented recursively until convergence of the grid is obtained. Figure 4.33 and Figure 4.34 compare the grids obtained at the base refinement levels with and without the smoothing procedure by showing a close-up of the grid near the plate leading edge.

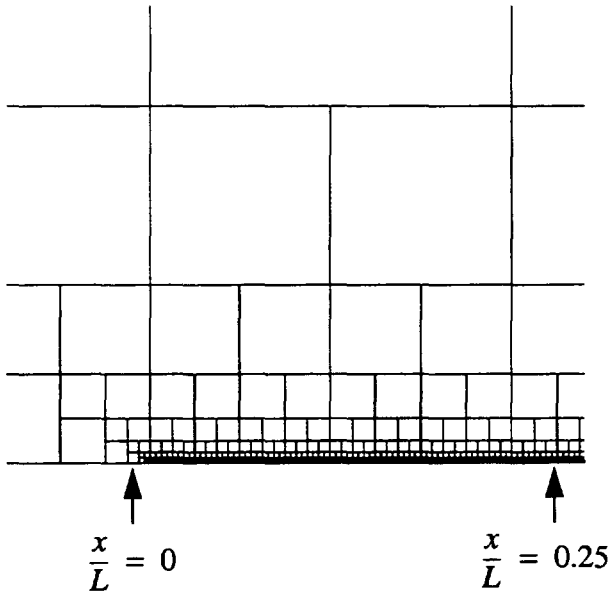


Figure 4.33 Close-up of Base Level Grid: No Length-Scale-Based Geometric Refinement

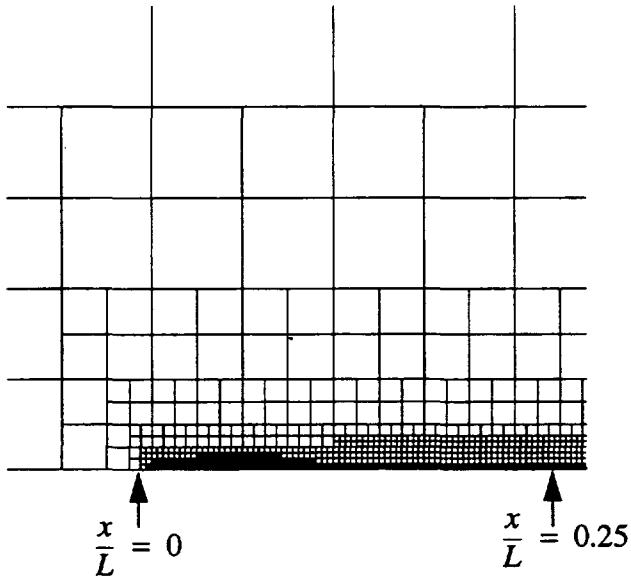


Figure 4.34 Close-up of Base Level Grid: Length-Scale Smoothing: $\Delta u = 0.2$

Both viscous reconstruction procedures are used and are compared as before, by examining the discrete positivity and accuracy measures, and by comparing the computed results to each other and to theory. The base grids for both schemes have 9837 cells, which were generated using the length-scale smoothing parameter of $\Delta u = 0.1$. Both schemes are asked to perform two levels of mesh refinement beyond the base grid. Due to memory limitations, the quadratic scheme only performs one level of mesh refinement beyond the base grid, and is slowed considerably by cpu paging for the desired final grid of over 51,000 cells. This is a consequence of storing pointers to the 6 neighbor cells needed for each face gradient reconstruction, which is quite redundant, as all the information could be obtained from the tree. For this work, speed has always been the primary goal, and due to this, memory usage is not as efficient as could be. The effect of adaptive mesh refinement is shown in Figure 4.35 and Figure 4.36 for the diamond path scheme for all levels of refinement at a location corresponding to $Re_x = 8000$.

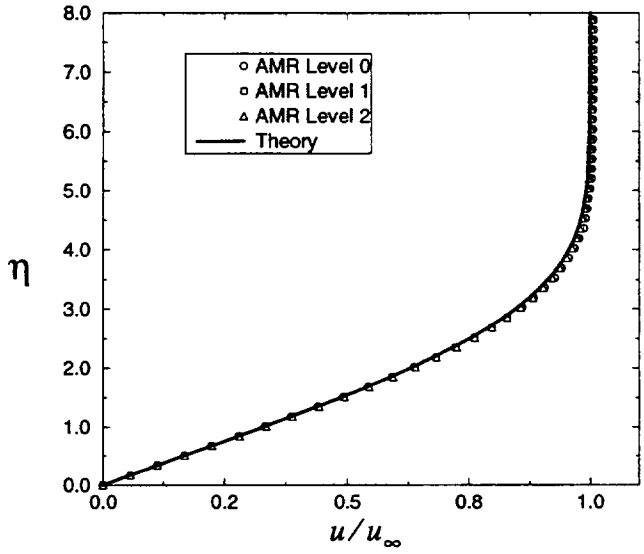


Figure 4.35 u-velocity Profiles: Effect of Adaptive Mesh Refinement at $Re_x = 8000$, Diamond Path Reconstruction.

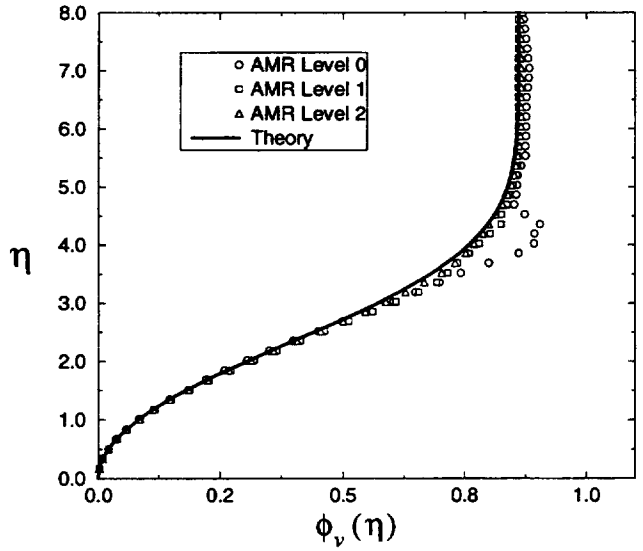


Figure 4.36 v-velocity Profiles: Effect of Adaptive Mesh Refinement at $Re_x = 8000$, Diamond Path Reconstruction.

The flow is well resolved so that there is little effect from the adaptive mesh refinement in the u-velocity profiles, but there is a slight improvement in v, as the mesh refinement

moves a refinement boundary farther away from the wall, to a region of less gradient. On a large scale, the skin friction is predicted well, as is indicated by the u-velocity profiles, and is shown in Figure 4.37. There is a slight overprediction in the first 10 percent of the plate, but overall the agreement is quite good.

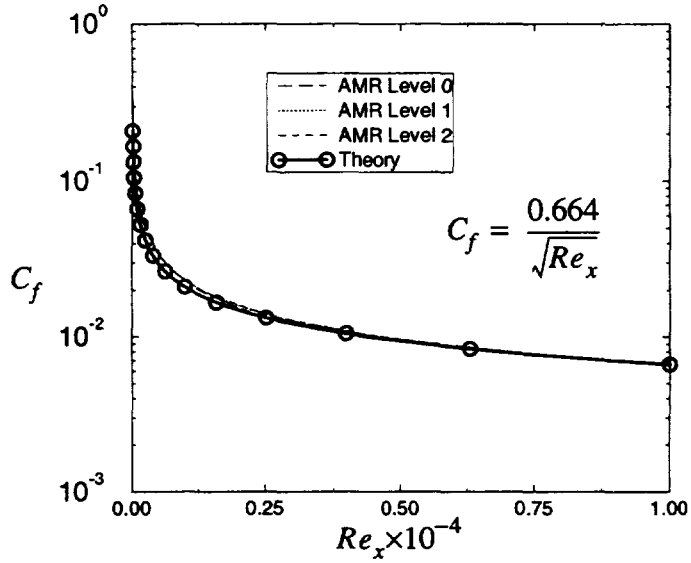


Figure 4.37 Skin Friction Through Adaptive Mesh Refinement, Diamond-Path Scheme.

There is little difference between the results of the two reconstruction schemes at the final grid refinement level. Examination of the u-velocity profiles at the final refinement levels yields results from the two schemes that are indistinguishable, and the v-velocities are nearly identical. Figure 4.38 shows the v-profiles from the two schemes at a location of $Re_x = 8000$. The lack of difference between the two schemes is directly attributable, as in the previous cases, to the quality of the grids obtained on all the meshes. The inconsistencies incurred by the diamond-path reconstruction scheme are low, and yields results that are little different from the quadratic scheme.

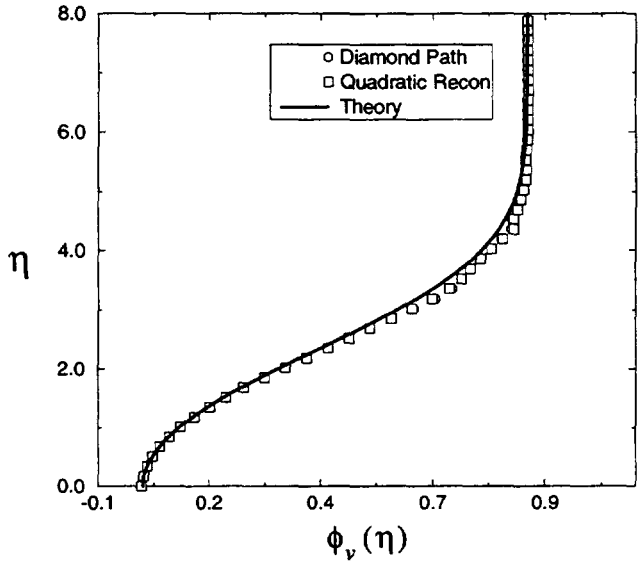


Figure 4.38 Comparison of v-velocity profiles at $Re_x = 8000$.

Table XVIII and Table XIX compare the positivity measures of the two schemes on the adaptively refined grids and Table XX shows the norms of the accuracy measures for the diamond-path reconstruction. As stated earlier the diamond-path reconstruction does not guarantee consistency, while the quadratic reconstruction does. This is directly shown by the accuracy norms: the quadratic reconstruction gives the correct accuracy norms to machine zero, and they are therefore not shown here. Also as before, neither scheme guarantees positivity of the stencil: The diamond-path scheme gives more non-positive stencils than the quadratic scheme, but the quadratic scheme gives stencils which are more non-positive.

Table XVIII Discrete Positivity Measure for Diamond Path Reconstruction

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min(\tilde{\alpha}_{min})$
0	-2.649e-02	-3.456e-01
1	-1.187e-02	-3.679e-01
2	-1.025e-02	-3.682e-01

Table XIX Discrete Positivity Measure for Quadratic Reconstruction

Mesh Refinement Level	L_1 of $\tilde{\alpha}_{min}$	$min(\tilde{\alpha}_{min})$
0	-1.460e-02	-7.651e-01
1	-2.556e-03	-7.628e-01

Table XX Discrete Accuracy Analysis, Diamond Path Reconstruction

Mesh Level	L_1 of $\sum\alpha_n x_n^2 - 2$	L_1 of $\sum\alpha_n x_n y_n$	L_1 of $\sum\alpha_n y_n^2 - 2$	L_∞ of $\sum\alpha_n x_n^2 - 2$	L_∞ of $\sum\alpha_n x_n y_n$	L_∞ of $\sum\alpha_n y_n^2 - 2$
0	6.67e-02	6.35e-04	1.14e-01	9.53e-01	8.60e-01	9.53e-01
1	3.33e-02	2.13e-04	5.26e-02	9.53e-01	9.17e-01	9.53e-01
2	3.21e-02	1.08e-04	4.81e-02	9.53e-01	9.17e-01	9.53e-01

The geometric properties arising from the alignment of the coordinate axes with the plate are fortunate. The root cell is located so that its southern face is exactly coincident with the plate surface, which means that all of the cells located on the plate are uncut and the only non-smoothness incurred by the grid is that across refinement boundaries.

The care taken in providing a suitable base grid comes about from experience. If there is insufficient resolution of the cells in the boundary layer, the results obtained reflect the under-resolution, as any viscous flow solver will. More importantly, if the flow is under-resolved so that refinement boundaries are located deep inside the boundary layer, the non-

smoothness of the grid normal to the wall can cause severe oscillations in the skin friction. This finding is not too surprising if one considers the results of the analysis: The current viscous flux functions are highly sensitive to grid non-smoothness and produce non-positive operators at refinement boundaries.

Use of the viscous length-scale geometric scaling helps to alleviate this problem, provided a suitable choice is made for the velocity scale, Δu. Figure 4.39 to Figure 4.42 show the grids obtained at the base (zero refinement) level for different values of the velocity scale.

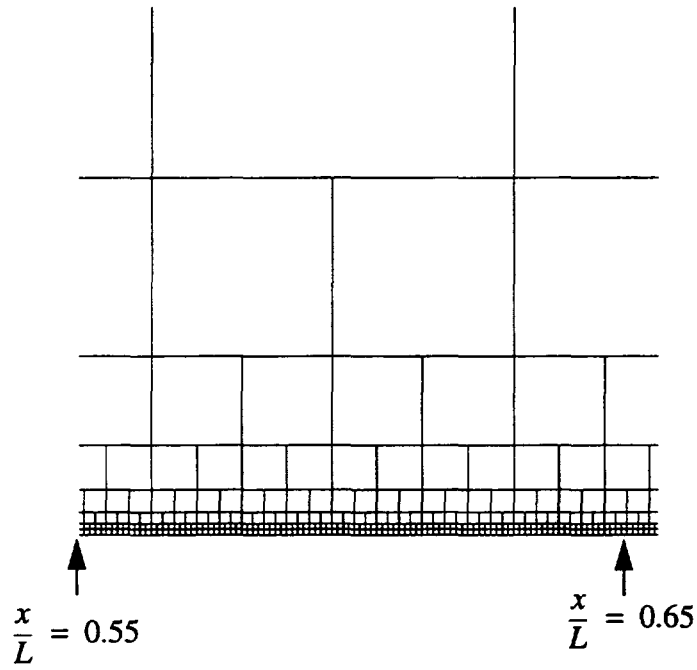


Figure 4.39 Base Grid Close-up: No Viscous Length Scale Based Geometric Refinement

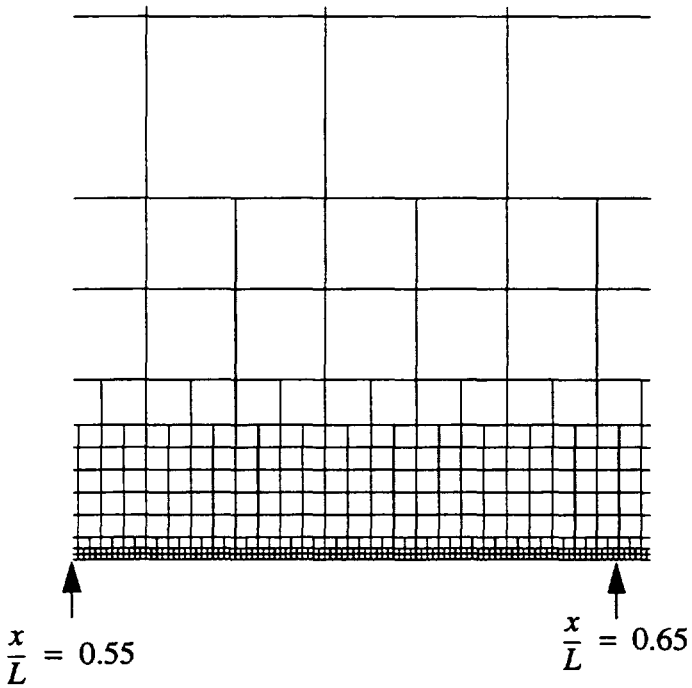


Figure 4.40 Base Grid Close-up: Viscous Length Scale Geometric Refinement: $\Delta u = 0.2$

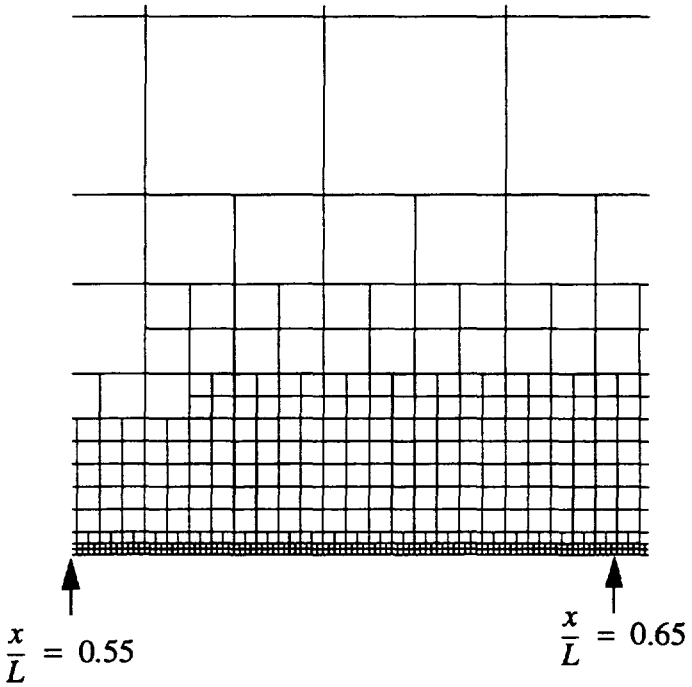


Figure 4.41 Base Grid Close-up: Viscous Length Scale Geometric Refinement: $\Delta u = 0.15$

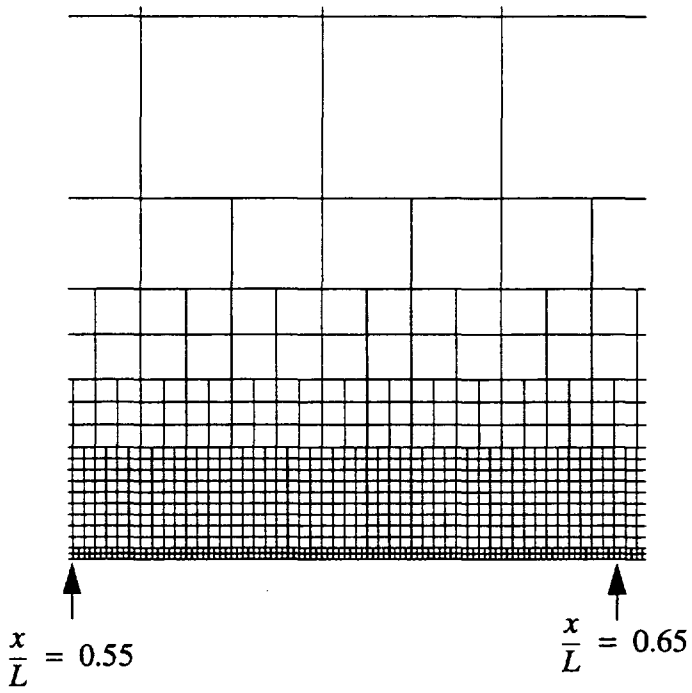


Figure 4.42 Base Grid Close-up: Viscous Length Scale Geometric Refinement: $\Delta u = 0.1$

Close-ups of the skin friction at the same plate locations indicated in Figure 4.39 to Figure 4.42 are shown in Figure 4.43 and Figure 4.44. When viewed on a larger scale, the skin friction is predicted well, but when an under-resolved grid is used, or there is a refinement boundary located sufficiently close to the wall (in a region of high velocity gradient), the skin friction is oscillatory. Although the mean quantities are smooth, their derivatives are not. It is crucial to note that this oscillation in the skin friction comes about from refinement boundaries being located in the viscous layer, and are not due to cut cells. The location of the root cell on the plate boundary ensures that all of the boundary cells on the plate are uncut. As can be seen from these skin friction plots, the reduction in the magnitude and period of the skin friction oscillations is directly attributed to the locations of refinement boundaries in high gradient regions. Indeed, even at the finest viscous-scaling level, there is a refinement boundary located 2 cells from the wall, showing up as a periodic oscillation with a very low magnitude in the skin friction.

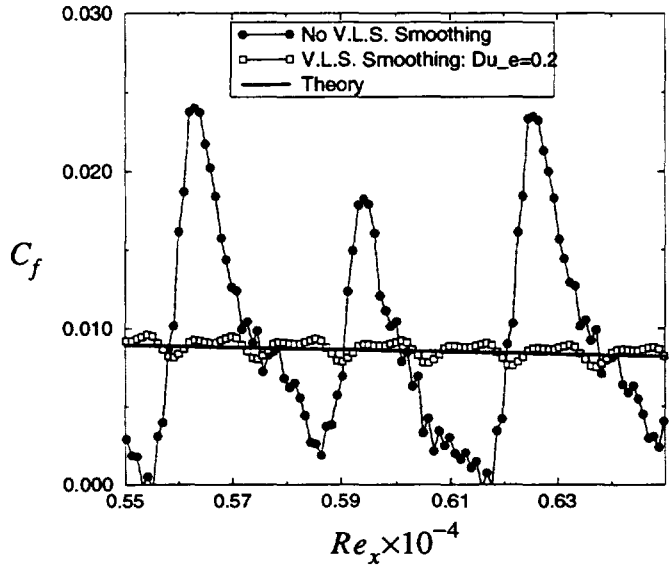


Figure 4.43 Close-up of Skin Friction for Different levels of Viscous Layer Smoothings

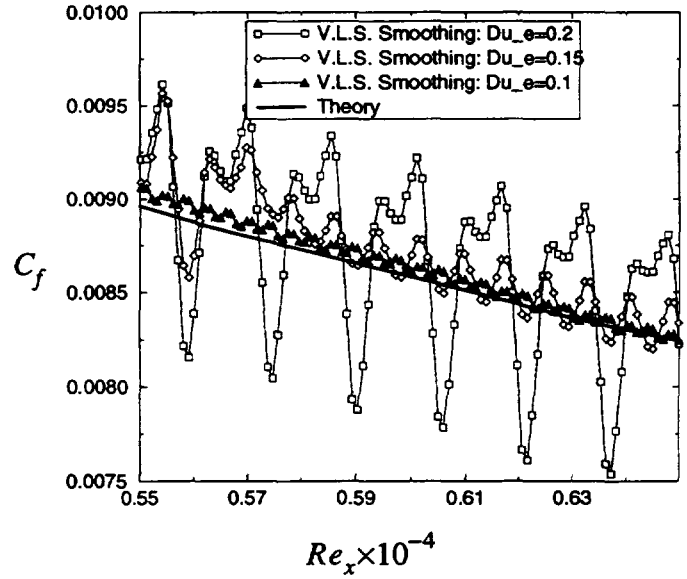


Figure 4.44 Close-up of Skin Friction for Different levels of Viscous-Layer-Smoothings: Note the Change in Axis Scale

Adaptive mesh refinement can alleviate the oscillations in skin friction, but can also introduce refinement boundaries due to the axial variation in length scale. This refinement is desired, and is the real reason to perform adaptive mesh refinement. Figure 4.45 to Figure 4.46 show close-ups in the same region as before of the grid for the next two levels of adaptive mesh refinement. In this sequence of grids the base grid is the same grid as shown in Figure 4.42. The first refinement eliminates the refinement boundary close to the wall, which eliminates the fine scale oscillation in the skin friction. The next level of refinement has added cells to resolve the plate normal length scale, which by the growth of the boundary layer height, decreases with increasing distance from the plate leading edge. The adaptive mesh refinement at this level has introduced a refinement boundary near the plate, at approximate $x/L = 0.6$. This refinement boundary introduces a hump in the skin friction, shown in Figure 4.44.

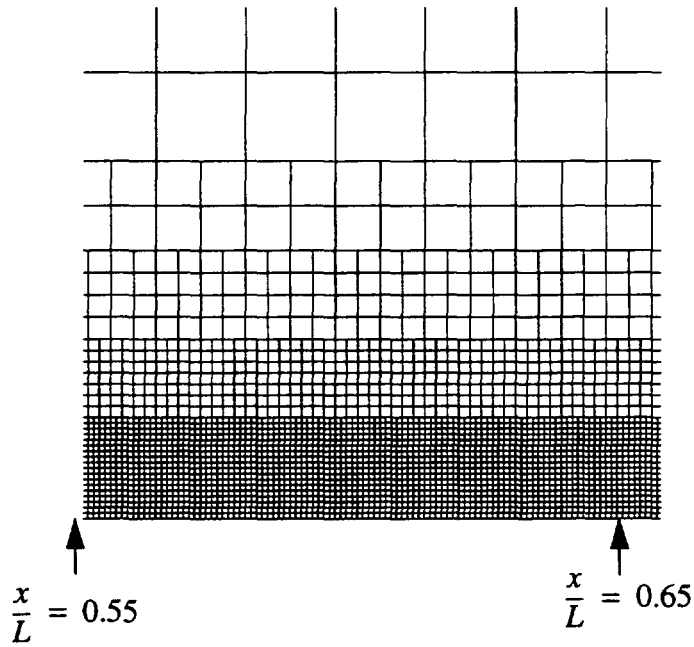


Figure 4.45 AMR Level 1 Grid for Viscous-Length-Scale Base grid $\Delta u_e = 0.1$

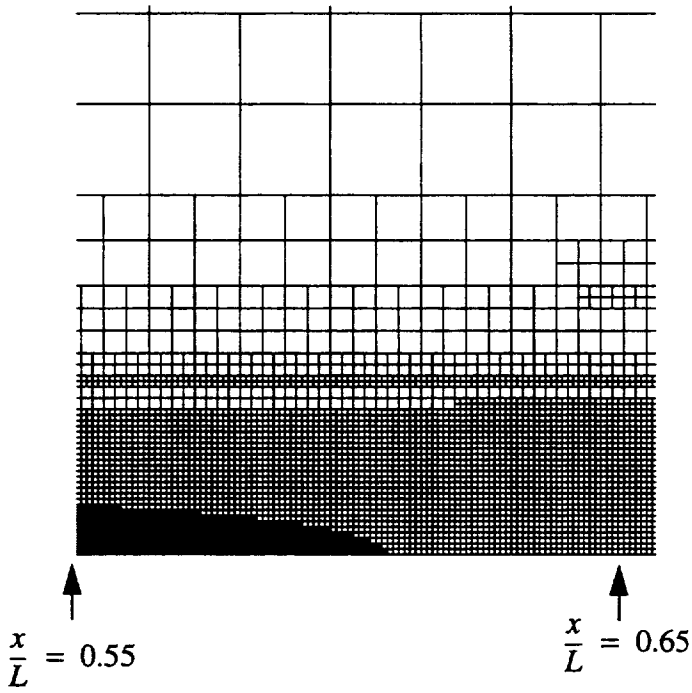


Figure 4.46 AMR Level 2 Grid for Viscous-Length-Scale Base grid $\Delta u_e = 0.1$

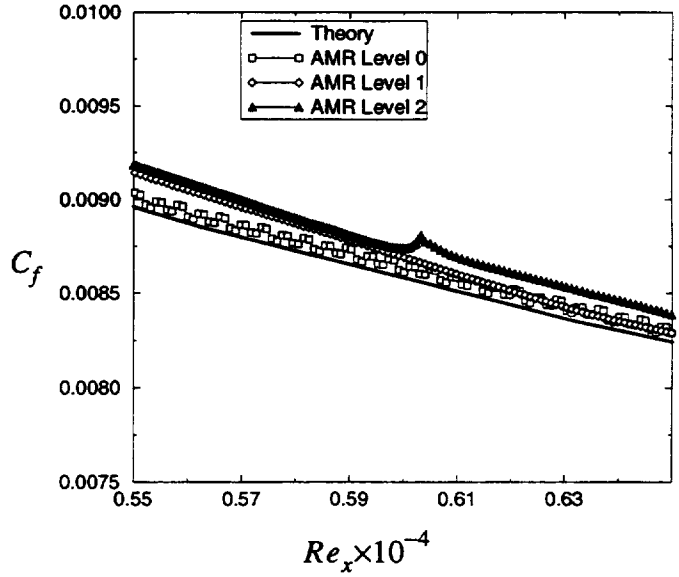


Figure 4.47 Close-up of Skin Friction through adaptive mesh refinement.

The oscillations in skin friction in this case were all caused by the grid non-smoothness introduced by refinement boundaries. The effect of grid non-smoothness induced by cell-cutting is investigated next.

4.2.4 Laminar, Developing Flow Over a Flat Plate: Non-Coordinate Axes Aligned

The same Reynolds number and free-stream conditions are used to compute the flow over the same free-stream aligned flat plate as in the previous section, but here, the plate is no longer aligned with the base coordinate axes. The computational domain is of the same extent as in the previous case, but is rotated 30 degrees about the plate leading edge. Figure 4.48 shows the grid at the base refinement level.

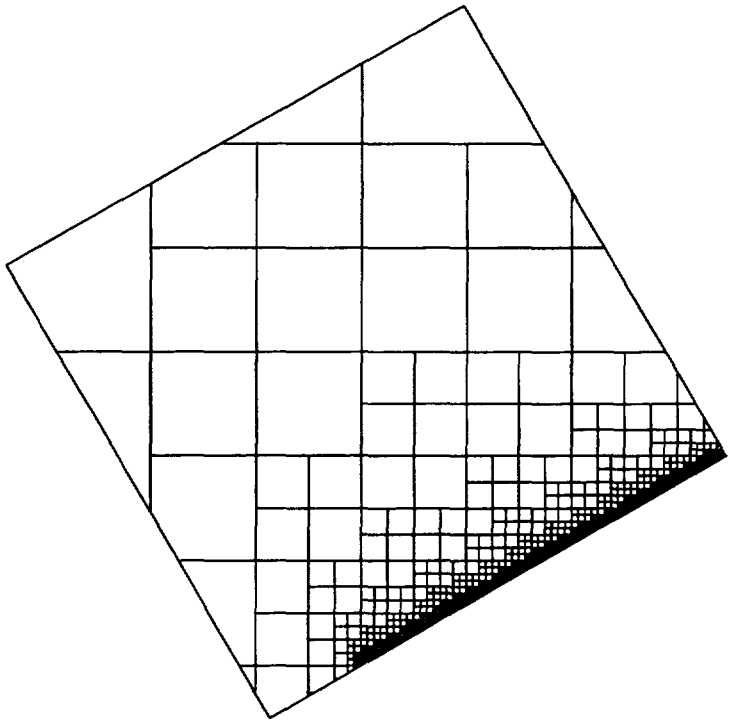


Figure 4.48 Rotated Plate, Base Refinement Level Grid

This orientation causes a very non-smooth grid near the surface of the plate due to cell cutting, and also causes a mis-alignment of the un-cut cell faces with the flow. Both of

these factors act to decrease the quality of the solution, but through different mechanisms. Cell cutting greatly increases the non-smoothness of the grid, which as is shown earlier, can decrease the accuracy and increase the non-positivity of the viscous operators. In addition, the mis-alignment of the cell faces with the dominant flow direction causes the inviscid flux function to add excessive dissipation, as is shown in [69]. These factors all combine to make the non-axis aligned results less accurate than the axis aligned results.

In practice, the positivity of the stencils turns out to play a major role. As in the previous case, two levels of adaptive mesh refinement beyond the base grid are attempted. Neither reconstruction procedure (without modification) is able to converge the flow through all the refinement levels. The diamond-path reconstruction scheme diverges midway through the second refinement level, while the quadratic-reconstruction procedure incurs excessive time-step reductions at the base level, until the Courant number is cut to below an acceptable level ($C_n = 1\times10^{-4}$), signalling the computation to stop.

But all is not lost. The analysis in 3.2.1.1 indicates that a positive stencil for a Laplacian using the diamond-path reconstruction can be obtained by setting the weights used to find the vertex data to zero. What is also indicated by the analysis is that this scheme will not exhibit the same linearity-preservation property as before, and this will result in local inaccuracy. The following results are obtained by setting the cut cells' vertex weights and the weights of their neighbors to zero for the rotated plate using the diamond-path reconstruction scheme. It is important to note that the only modification is made to the cut cells and their neighbors: all of the interior-cell flux computations are unchanged. The viscous-layer scaling was used for the base grid, where experience dictates setting the velocity scale to $\Delta u = 0.1$, and the solution is obtained at the base grid level and two levels of adaptive mesh refinement. Figure 4.49 and Figure 4.50 show the velocities along and normal to the plate at $Re_x = 8000$ through the mesh refinement while Figure 4.51 and Figure 4.52 show the results along the plate at the final refinement level.

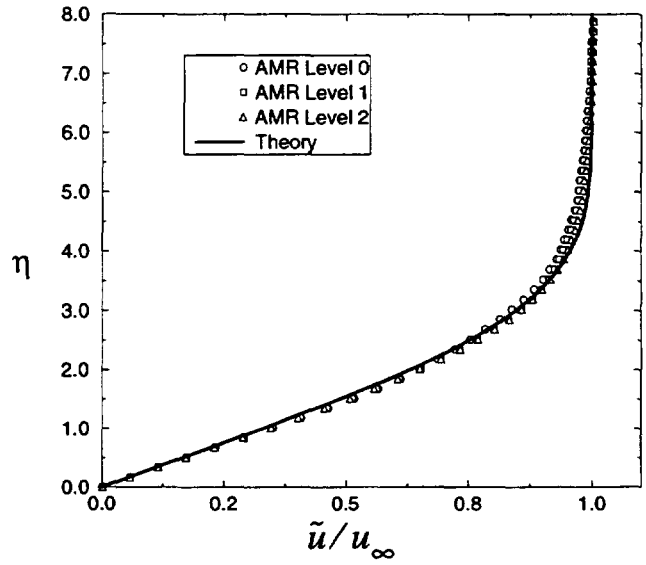


Figure 4.49 Effect of Adaptive Mesh Refinement for the Rotated Plate: u-velocity

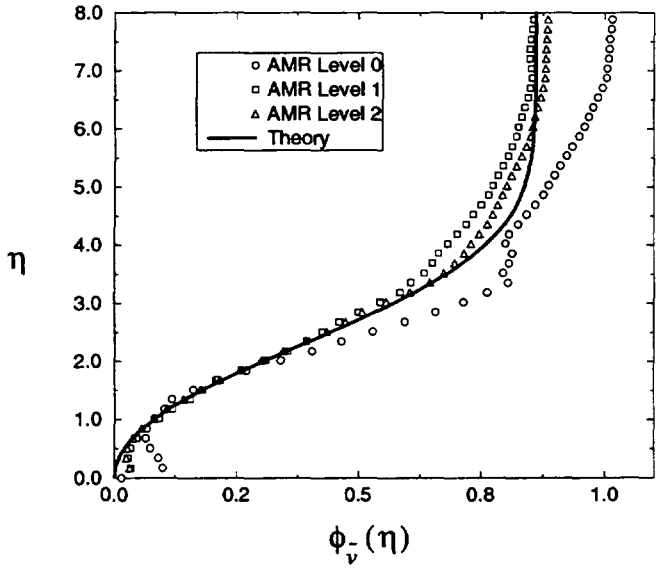


Figure 4.50 Effect of Adaptive Mesh Refinement for the Rotated Plate: v-velocity

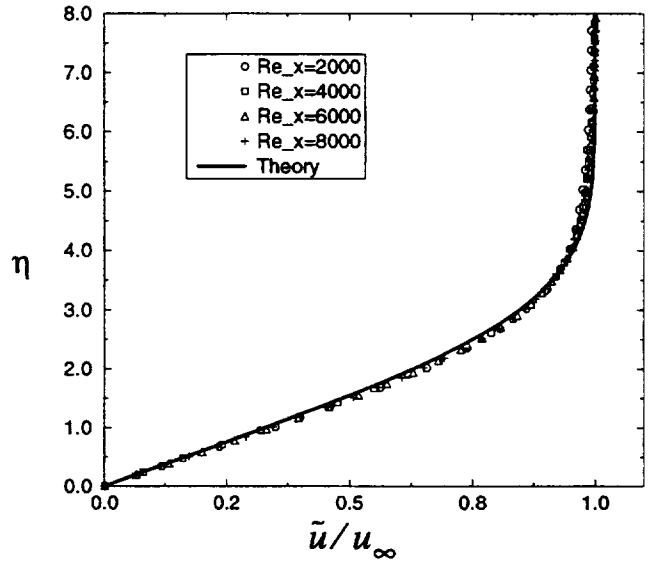


Figure 4.51 Plate-Aligned Velocity Comparison at Final Refinement Level

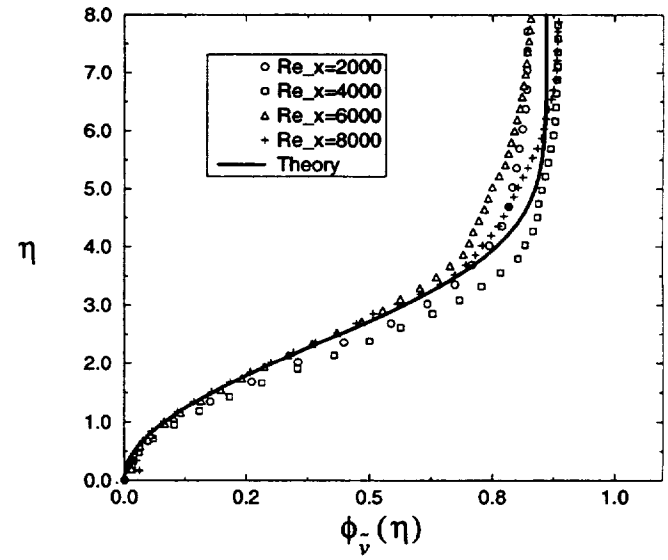


Figure 4.52 Plate-Normal Velocity Comparisons at refinement level 2

As in the axis-aligned case, after enough mesh refinement is performed, the solution agrees well with theory, although the normal velocity component is not predicted as well as in the axis aligned plate case. The predicted skin friction is very oscillatory, as is shown in Figure 4.53, where the skin friction is shown at the final refinement level. Even though

the viscous-layer scaling is used to generate the grid, and the grid is smooth away from the body, the non-smoothness caused by grid irregularity of the cut cells causes problems with the skin friction. Figure 4.55 shows a close-up of the grid near the wall for the final level of adaptive mesh refinement, showing the smoothness of the grid away from the wall and the grid irregularities caused by the cell cutting. It should be noted that this final grid contains over 49,000 cells, which compared to most structured grid codes, is quite a large number of cells for this particularly simple case. The inefficiency incurred by the Cartesian approach through its use of unit aspect ratio cells is evident: For even moderate Reynolds number flows, it is quite necessary to use stretched elements.

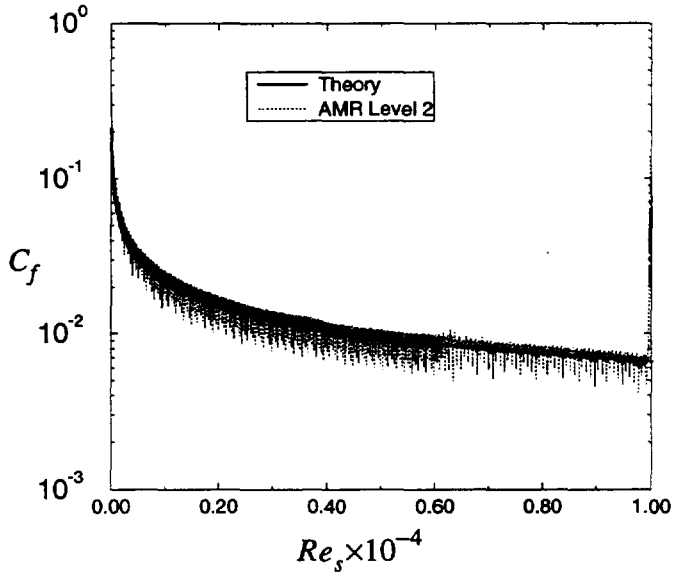


Figure 4.53 Skin Friction for the Rotated Plate

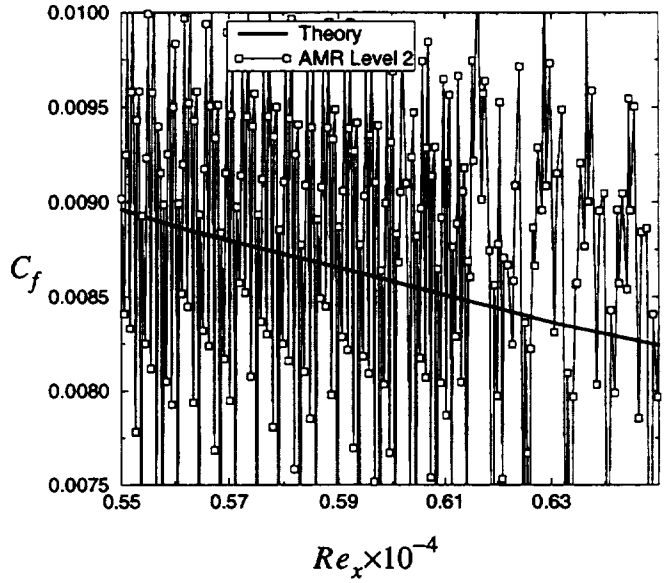


Figure 4.54 Skin Friction for the Rotated Plate, Close-up View

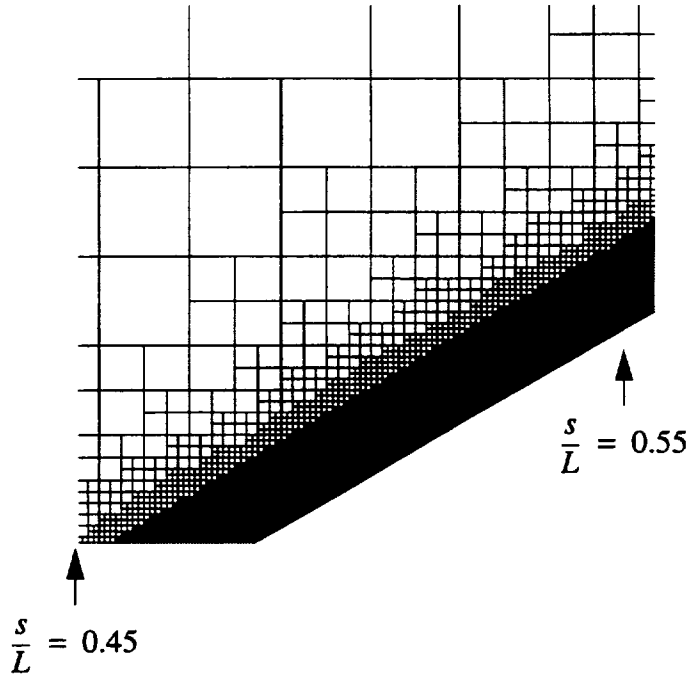


Figure 4.55 Close-up of grid near wall

For these results, the triangular-based reconstruction shown in section 4.1.3 is used to compute the gradients at the walls for input to the viscous flux function and for computation of the skin friction. A more complicated procedure can be used which is less sensitive to grid non-smoothness, but is also less robust. A function in u and v is expanded about the body face Gauss point using basis functions that are identically zero at the Gauss points, and a least-squares minimization procedure is used to find the unknown coefficients. In practice, a linear expansion using only order-one neighbors to the boundary cell is found to work best. The expansion is formulated as

$$u = u_x \phi_1 + u_y \phi_2 \tag{4.27}$$

where the basis functions are linear, and vanish at the Gauss points

$$\phi_1 = x - x_g \qquad \phi_2 = y - y_g \tag{4.28}$$

Application of this procedure results in an improvement of the skin friction shown in Figure 4.56, but, a closer examination shown in Figure 4.57 indicates that there are still severe oscillations due to the non-smoothness of the grid. Use of higher-order reconstructions and/or higher-order neighbors does not improve the results: The best are obtained using, as expected, the lower-order expansion using the natural neighbors. Although there are still severe oscillations, the change in basis function for the wall gradient reconstruction improved the behavior.

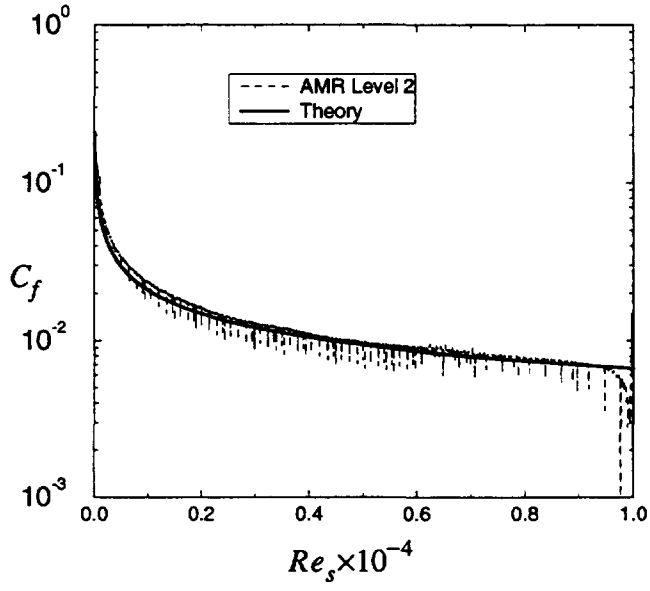


Figure 4.56 Skin Friction Obtained using Linear Expansion, Final AMR Level

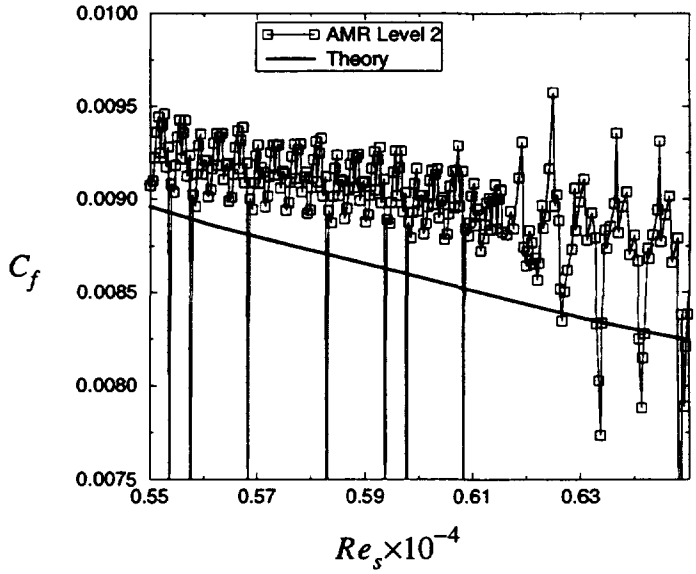


Figure 4.57 Skin Friction Obtained using Linear Expansion, Close-up

As seen by inspecting the grid, there is more than adequate grid resolution near the wall, and this resolution extends uniformly out into the boundary layer, where the refinement boundaries are located sufficiently far away from the wall. The plots of the plate-nor-

mal and plate-tangential velocities show that with sufficient resolution, the mean flow quantities can be predicted accurately, but the irregularities of the grids inhibit smooth derivatives of these quantities. All of the preceding analysis indicates that the current viscous flux functions are highly sensitive to grid smoothness and orthogonality. This is shown in practice to be a very important property, as in the axis-aligned flat plate where there was no cell cutting but there were many cells with refinement boundaries located deep inside the viscous layer. With cell cutting, the situation is more serious: Not only must care be taken to ensure adequate and uniform resolution in high-gradient regions, but the cell cutting introduces more irregularity into the grid, to which the viscous flux function is shown to be very sensitive. On a positive note, the Cartesian approach is shown to give reasonable mean flow quantities, as is shown in the quality of the driven cavity, back step, and flat plate results. The next case illustrates the utility of the Cartesian-cell approach for the viscous flow through a complicated geometry.

4.2.5 Flow in a Branched Duct with Cooling Fins

A major advantage of using the Cartesian-cell approach is its ability to generate grids about complicated geometries with minimal user intervention. The grid generation and the flow computations with the adaptive mesh refinement of this case highlight this capability.

The geometry of this case corresponds to an experiment conducted in [70] which was designed to simulate, in a simplified manner, the flow in the cooling passages of a turbine blade. The flow passages inside turbine blades are extremely complex, with many protuberances and bends designed to decelerate the flow and enhance mixing with secondary coolant flows. The flow is highly mixing dominated and usually at a low, yet turbulent, Reynolds number. Due to the complexity of the domain and flow, it is extremely manpower intensive to create a computational grid in the domain. As a further complication, the flow is very difficult to compute, due to the turbulence levels and ambiguity of the inflow and outflow conditions. The purpose of the branched-duct experiment is to elimi-

nate some of these ambiguities by providing simplified geometry and flow conditions, but still be representative of the complex flow processes that occur in the real turbine coolant passages. Figure 4.58 shows a diagram of the branched duct geometry with arrows indicating the flow directions.

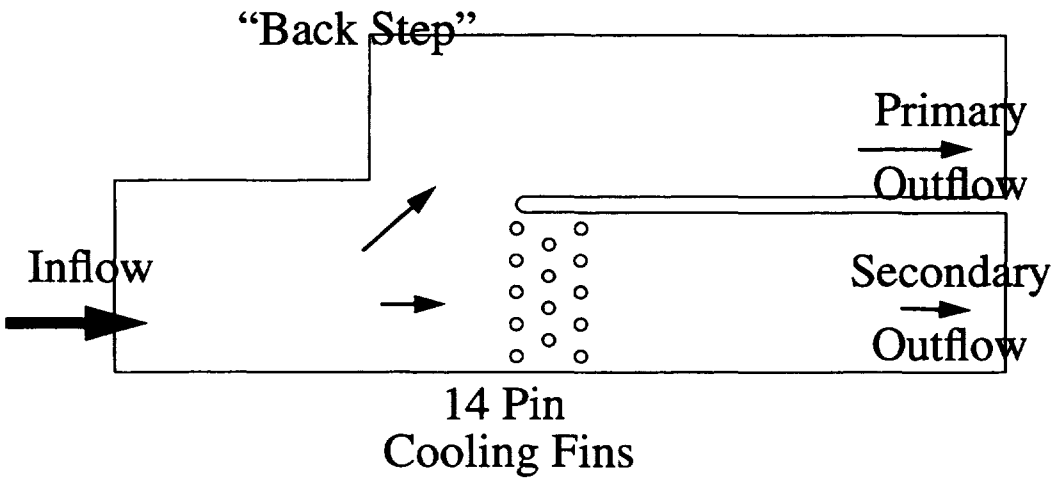


Figure 4.58 Schematic of Branched Duct Flow and Geometry

In the actual turbine coolant passages, it is desired to slow the flow down to allow increased heat transfer from the hot blade to the coolant flow. The pin fins accomplish this by a deceleration of the flow downstream of the pins by providing an increased blockage and mixing of the flow in the secondary passage. Since the flow is blocked through this passage by the presence of the pin fins and their wakes, the primary flow is diverted upwards, around the diverter and out the primary outflow exit. This geometry and flow field has a few similarities with the backward facing step flow: There is a large recirculation zone aft of the sudden expansion, and additionally (as shown here and in the experimental results [70]), a large separation zone just aft of the diverter plate, in the primary flow passage.

The following calculation demonstrates the Cartesian-cell approach for this geometry, but

in no way attempts to compute the flow at the test conditions. The test conditions are turbulent while the conditions shown here are laminar, and only are used to demonstrate the flow solver's current capability. Although the inclusion of a turbulence model is certainly possible with the Cartesian-cell approach, it is not investigated here.

A fully-developed, laminar profile is provided at inflow while the pressure is set at the outflow boundaries. No-slip boundary conditions are applied on the duct, splitter and pin surfaces. Two different Reynolds numbers are computed which are chosen to provide a low Reynolds number based on pin diameter and a low enough Mach number to preclude compressibility effects. Both conditions are characterized by the maximum Mach number in the inflow profile and the Reynolds number based on the pin diameter and maximum inflow velocity. In terms of the pin Reynolds number, radius and inflow maximum Mach number, the non-dimensionalizing Reynolds number is

$$Re_{\infty} = \frac{Re_{pin}}{2M_{\infty}\frac{r_{pin}}{l_{\infty}}} \tag{4.29}$$

The lower Reynolds number calculation corresponds to $Re_{pin} = 25$ and $M_{\infty} = 0.1$ while the higher Reynolds number corresponds to $Re_{pin} = 100$ and $M_{\infty} = 0.25$. The dimensions of the geometry, in centimeters, are shown in Figure 4.59 while the location and radii of the pin fins are shown in Table XXI.

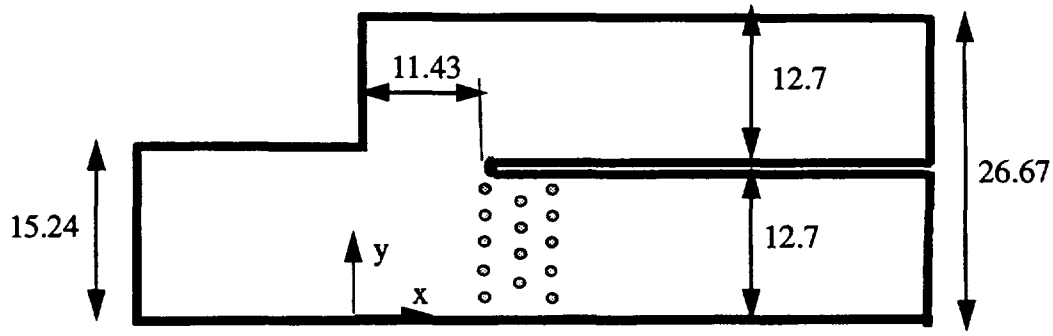


Figure 4.59 Geometry of Branched Duct

Table XXI Pin Fin Locations and Radii

x (m)	y(m)	r(m)
0.114300	0.012700	0.004763
0.114300	0.038100	0.004763
0.114300	0.063500	0.004763
0.114300	0.088900	0.004763
0.114300	0.114300	0.004763
0.139700	0.025400	0.004763
0.139700	0.050800	0.004763
0.139700	0.076200	0.004763
0.139700	0.101600	0.004763
0.165100	0.012700	0.004763
0.165100	0.038100	0.004763
0.165100	0.063500	0.004763
0.165100	0.088900	0.004763
0.165100	0.114300	0.004763

The base grid without viscous length scale geometric refinement, shown in Figure 4.60, was generated in 52 seconds on an IBM RS6000, model 560. This grid contains 3150 cells with one continuously represented outer body and 14 pin fins. A close-up of the pin fin region is shown in Figure 4.61.

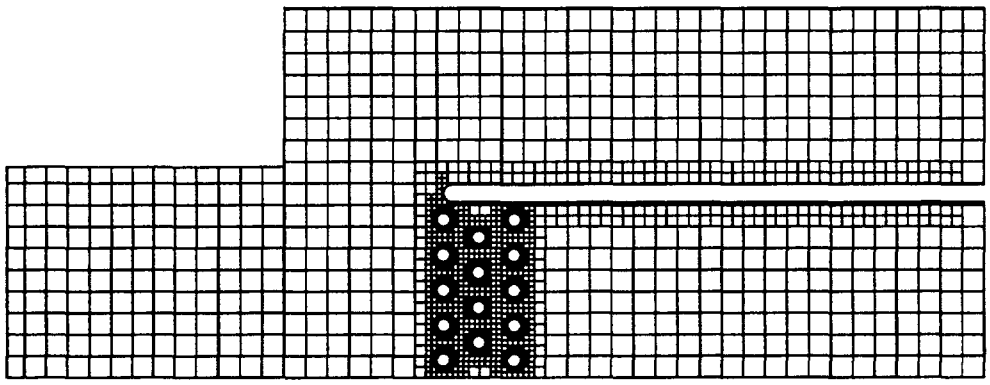


Figure 4.60 Branched Duct: Base Grid

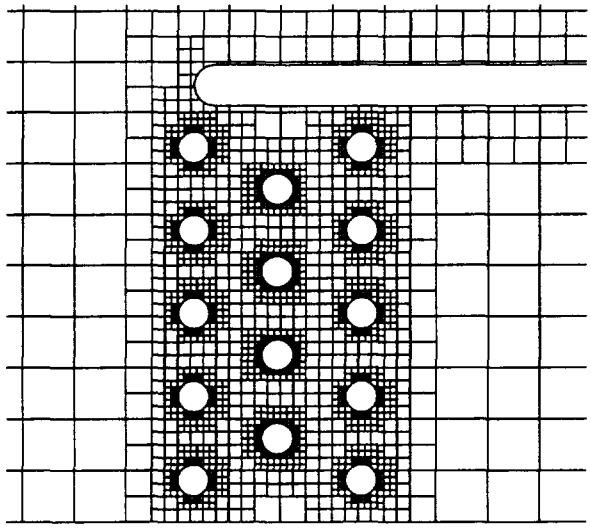


Figure 4.61 Branched Duct: Base Grid: Close-up

Viscous-length-scale based geometric refinement is performed, where for the fully developed inflow, the shear is specified from the parabolic velocity profile and is used in

both the primary and secondary passages to provide the geometric scaling. The viscous scaling is performed for the pin fins also by using a scaling which is based on the Blasius profile with an arc-length-based coordinate whose origin is at the $\theta = \pi$ position on each fin. The base grid, shown in Figure 4.62 with a close-up of the pin fin region in Figure 4.63, has 13,675 cells and was generated in 1445 seconds.

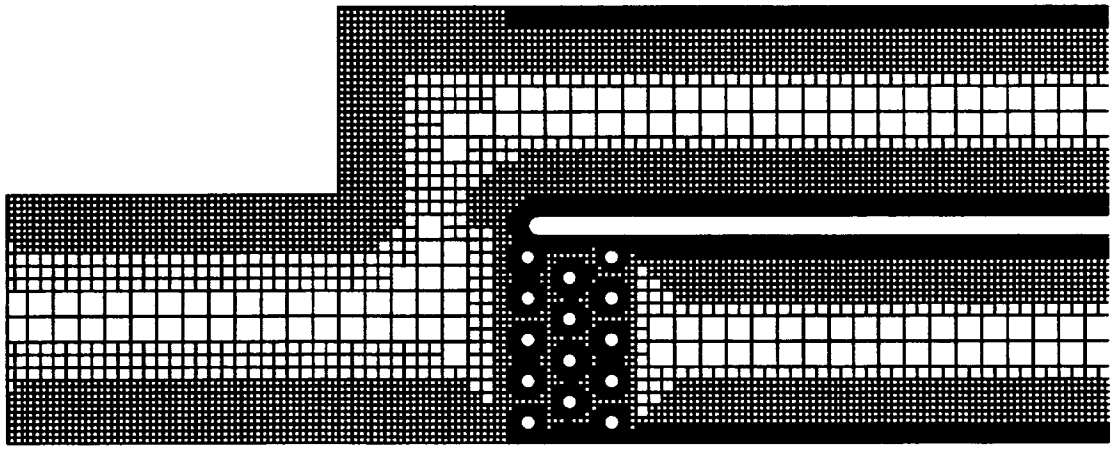


Figure 4.62 Viscous Length Scale Based Grid, Base Refinement Level.

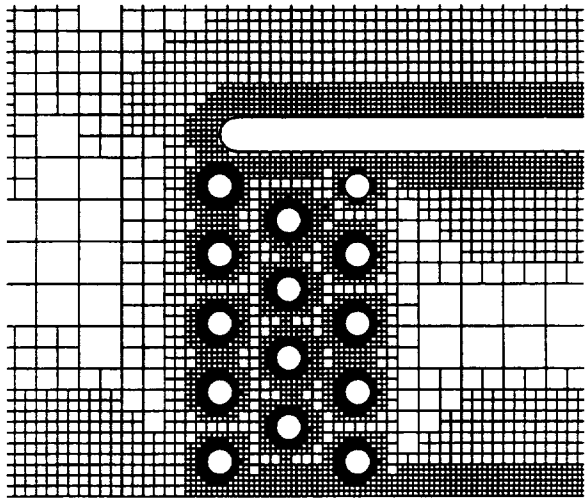


Figure 4.63 Viscous Length Scale Based Grid, Base Refinement Level: Close-up of Pin Fins.

Due to the non-robustness of the quadratic reconstruction procedures, the diamond-path

reconstruction using the linearity-preserving weightings is used. In addition, experience from the preceding cases indicates that setting the vertex weights in the diamond path reconstruction to zero for the cut cells and their neighbors gives a more robust solver. For both Reynolds numbers, the solutions were converged on the base and first refinement levels. Both Reynolds numbers did not converge on the second refinement level, incurring excessive Courant number cutbacks until the Courant number was below the cut-off threshold. Even though many levels of refinement were not achieved for these cases, the results shed light onto the characteristics of the flow fields.

First, the lower Reynolds-number case is shown. Figure 4.64 and Figure 4.65 show total-velocity contours on the base grid level. The flow contains many recirculation regions in addition to the obvious, large recirculation zone aft of the back step. The deceleration of the flow upstream of the pin fins causes a large thickening of the boundary layer ahead of the pins on the lower wall, although the flow remains attached. There is a mild separation on the primary passage side of the diverter plate.

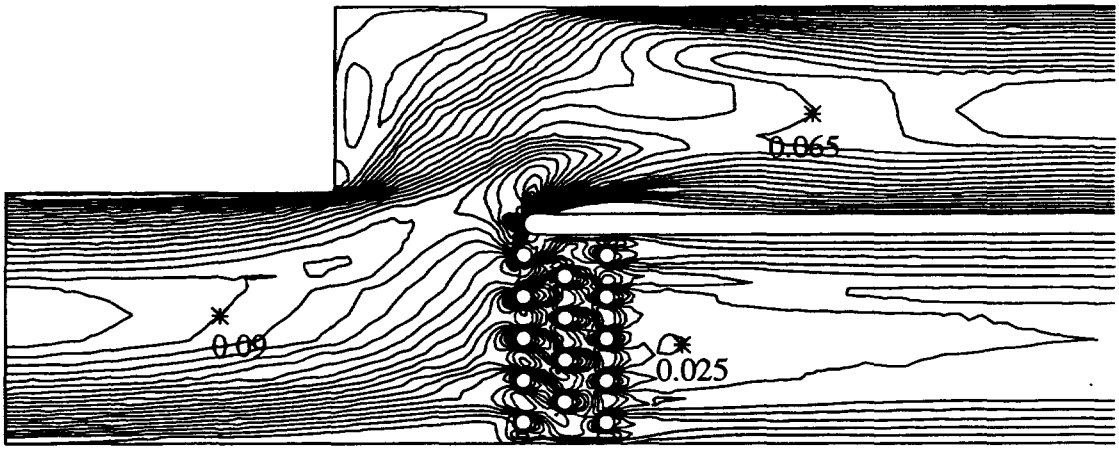


Figure 4.64 Low Reynolds Number Branched Duct Case: Base Refinement Level, Total Velocity Contours: Min=0, Max=0.1945, Increment=0.005

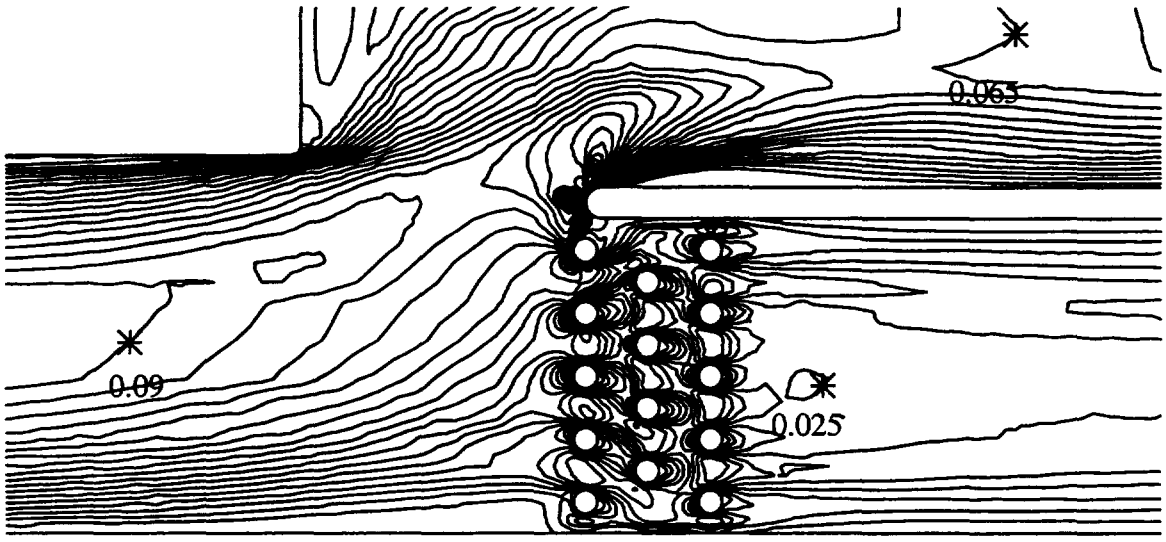


Figure 4.65 Low Reynolds Number Branched Duct Case: Base Refinement Level, Close-up of Pin Region, Total Velocity Contours

Solution adaptive mesh refinement improves the quality of the solution. Figure 4.66 shows contours of the total velocity, and Figure 4.67 shows the adapted grid. Line plots of the u velocity at different streamwise locations are shown in Figure 4.69 to Figure 4.75.

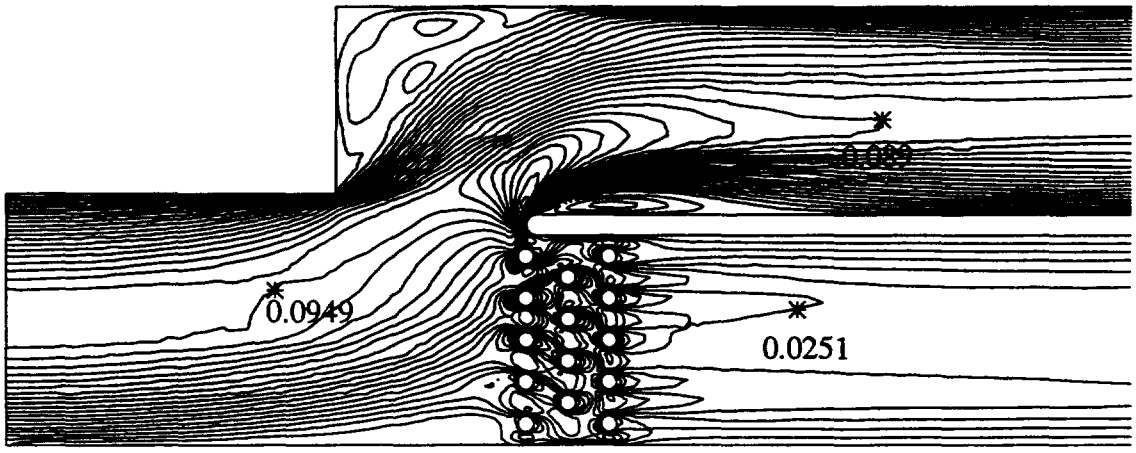


Figure 4.66 Total velocity Contours, Low Reynolds Number Case, Final Refinement Level: Min=0, Max=0.1368, Increment=0.005.

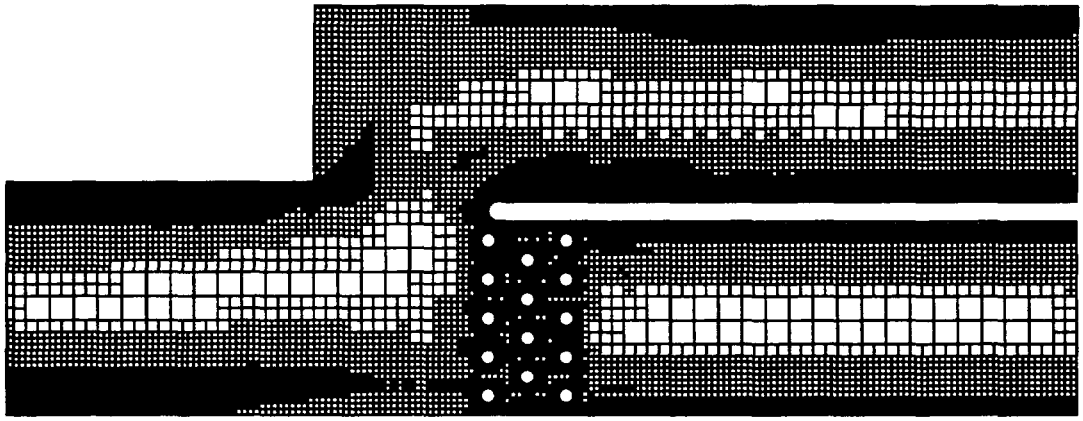


Figure 4.67 Final Adapted Grid, Low Reynolds Number Case

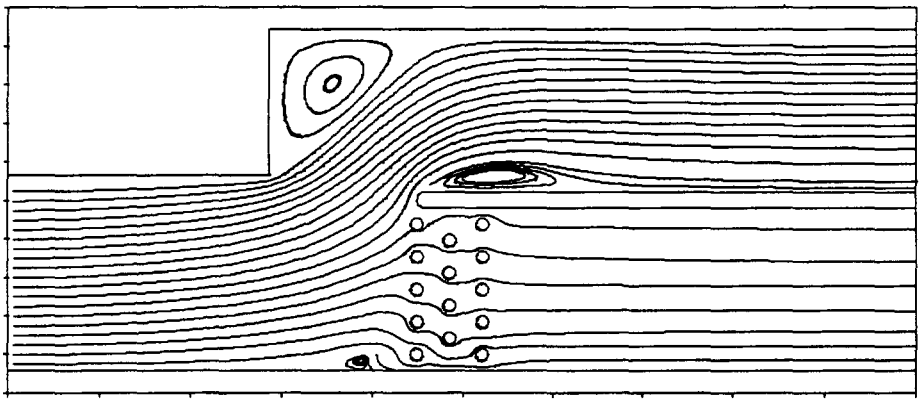


Figure 4.68 Particle Traces: Low Reynolds Number Case, Final Refinement Level

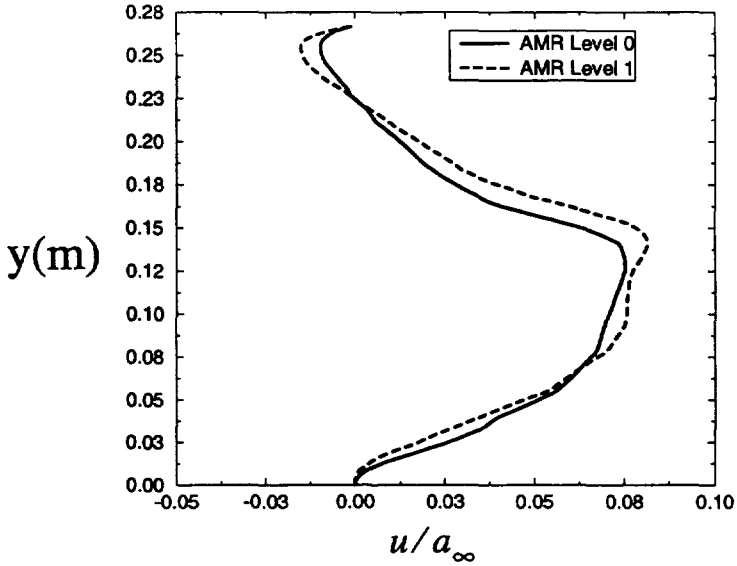


Figure 4.69 u-velocity at x=.05 meters (Ahead of Pins)

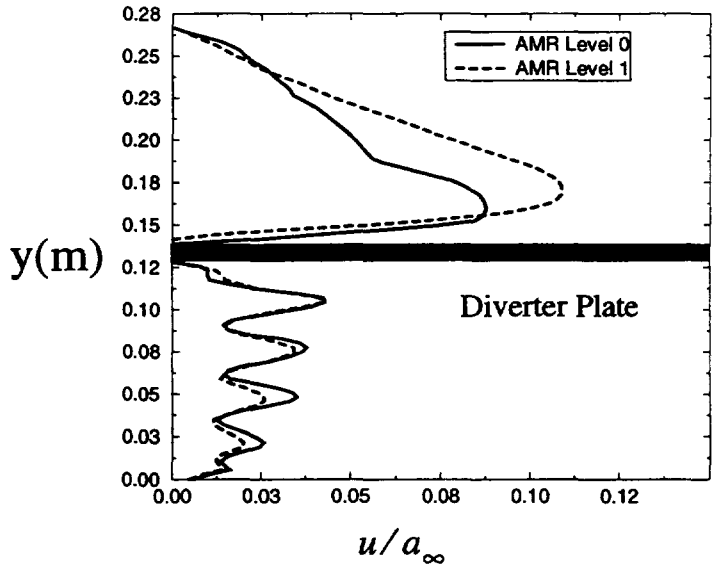


Figure 4.70 u-velocity at x=0.127 meters (Behind First Pin Row)

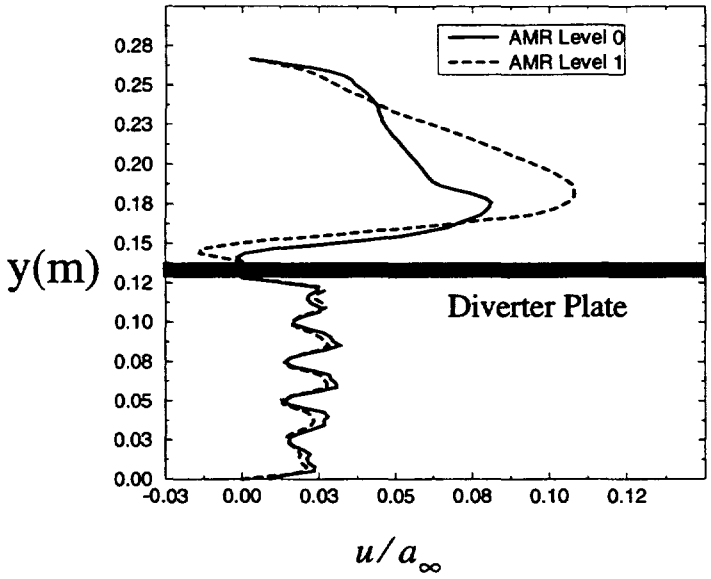


Figure 4.71 u-velocity at x=0.1524 meters (Behind Second Pin Row)

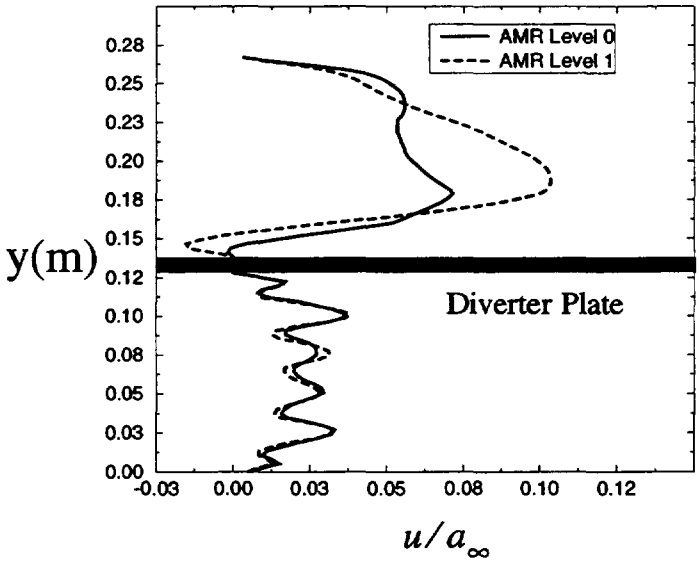


Figure 4.72 u-velocity at x=0.1794 meters (Behind Final Pin Row)

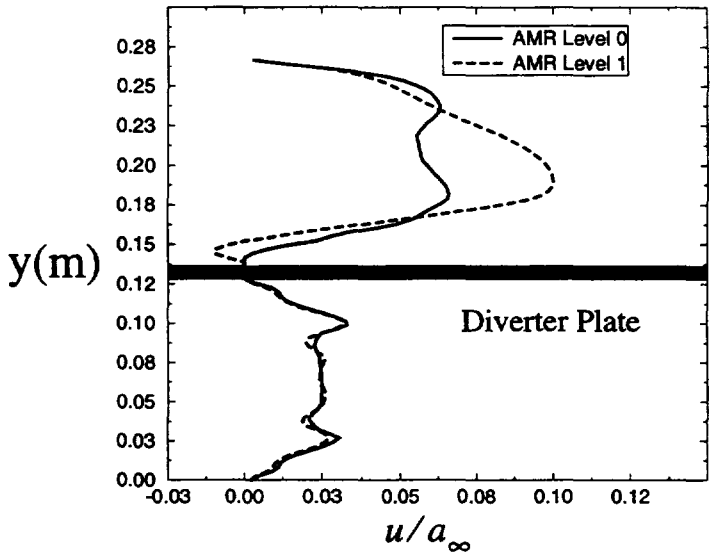


Figure 4.73 u-velocity at x=0.2 meters (Downstream of Pins)

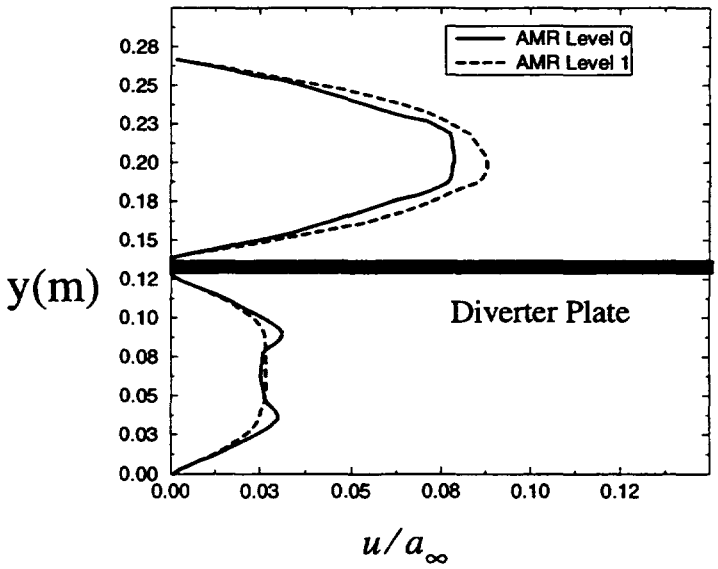


Figure 4.74 u-velocity at x=0.5 meters (Near Exit Plane)

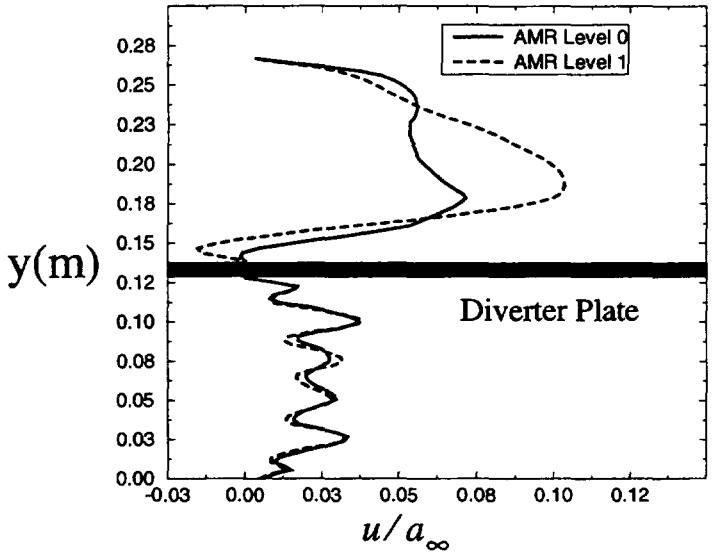


Figure 4.75 u-velocity at x=0.1794 meters (Behind Final Pin Row)

As seen in the velocity upstream of the pins, there is a substantial upstream influence of the pin fins which diverts much of the flow into the upper passage. The velocity-deficit region upstream of the pins is quite large, as shown in Figure 4.69 and is approaching separation at x=0.05 meters. The wakes of the individual pins are discerned in the plots located inside of the pin regions, but are seen to dissipate quickly due to the coarse grids in the wake regions. The flow above the diverter plate shows the recirculation zone aft of the diverter leading edge for the final refinement level, which does not show up at the base grid level. In addition, far downstream of the pins, the primary and secondary flows are approaching a fully-developed profile, where the primary flow has approximately three times the mass flow as the secondary flow. The adaptive mesh refinement has increased the quality of the solution.

The higher Reynolds number case exhibits many of the same phenomena, but has a few particulars of the flow that are different from the lower Reynolds number case. The upstream influence from the pins causes a mild separation zone on the lower wall, ahead of the pins, which did not occur in the lower Reynolds number case. The wakes behind the

pins are thinner and extend farther downstream before being dissipated. Importantly, due to the thinner viscous layers in the pin region, the pins block less of the flow, and because of this, the flow far downstream shows that the mass flow in the upper channel is approximately twice that in the lower flow. The separation zone aft of the sudden expansion is larger, and the recirculation zone on the upper side of the diverter plate, just aft of the leading edge, extends farther downstream and is larger in the transverse direction also, due to the increase mean velocity of the inflow. The upper wall in the upper channel exhibits a separation zone caused by the adverse pressure gradient from the reattachment of the lower recirculation zone. Again, the adaptive mesh refinement improved the quality of the solution automatically. Contours of the u-velocity are shown in Figure 4.76 and Figure 4.77. Particle traces are shown in Figure 4.78 while line plots of the u-velocity at the same selected axial locations as the previous case are shown in Figure 4.79 to Figure 4.84.

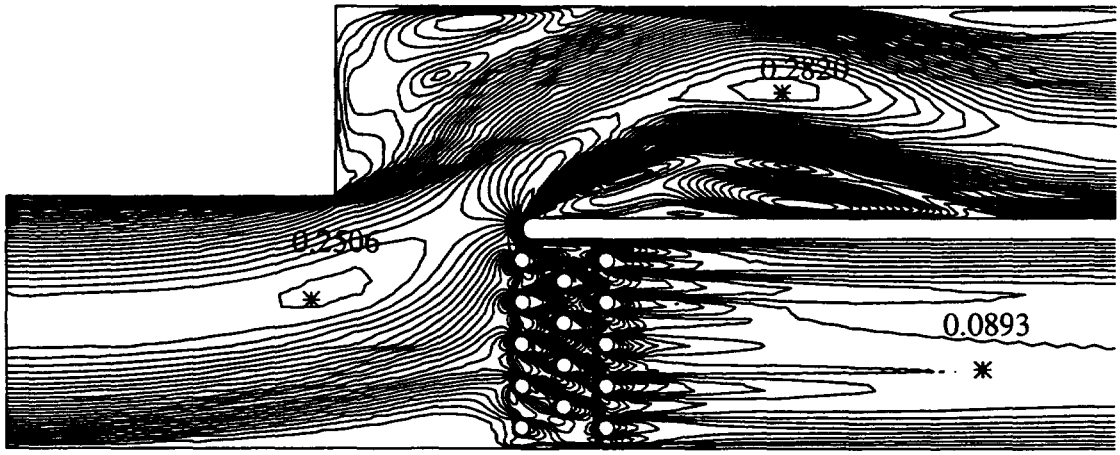


Figure 4.76 Total-velocity Contours, Final Refinement Level, High Re Case. Min=0.0, Max=0.316, Increment=0.010.

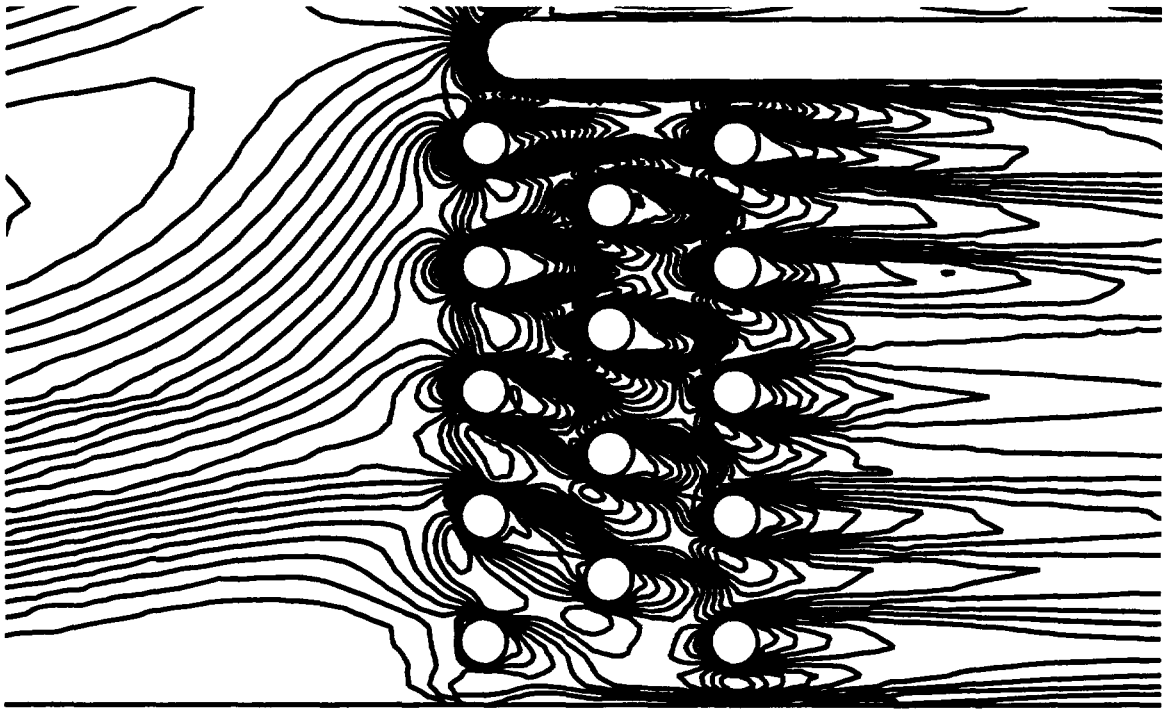


Figure 4.77 u-velocity Contours, Final Refinement Level, High Re Case

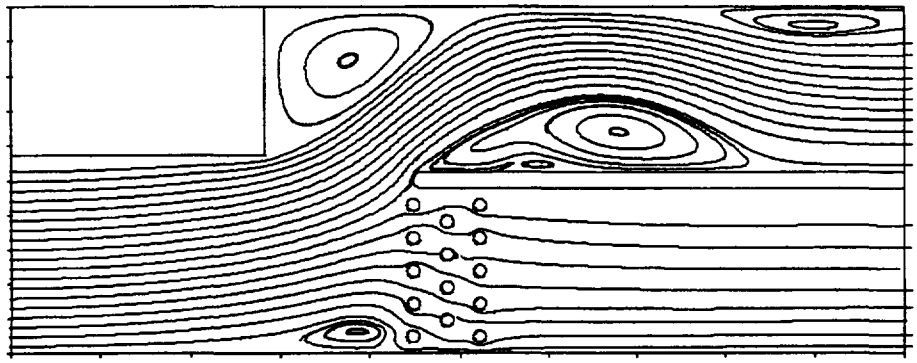


Figure 4.78 Particle Traces, High Reynolds Number Case, Final Refinement Level

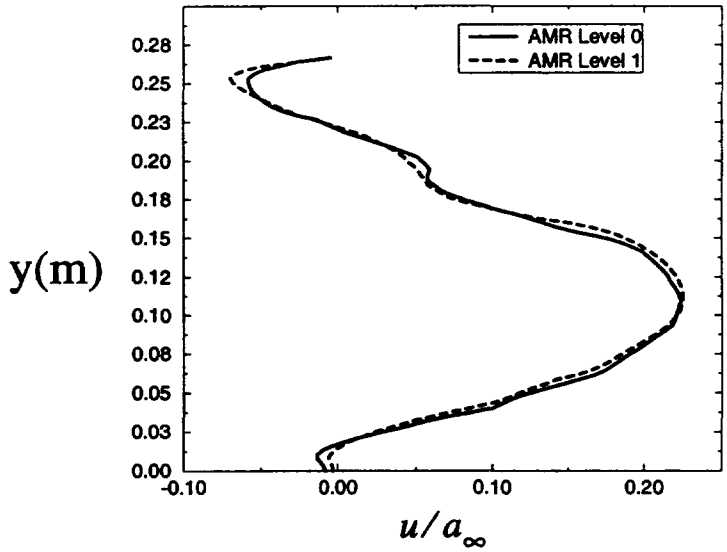


Figure 4.79 u-velocity at x=0.05 meters, High Reynolds Number Case

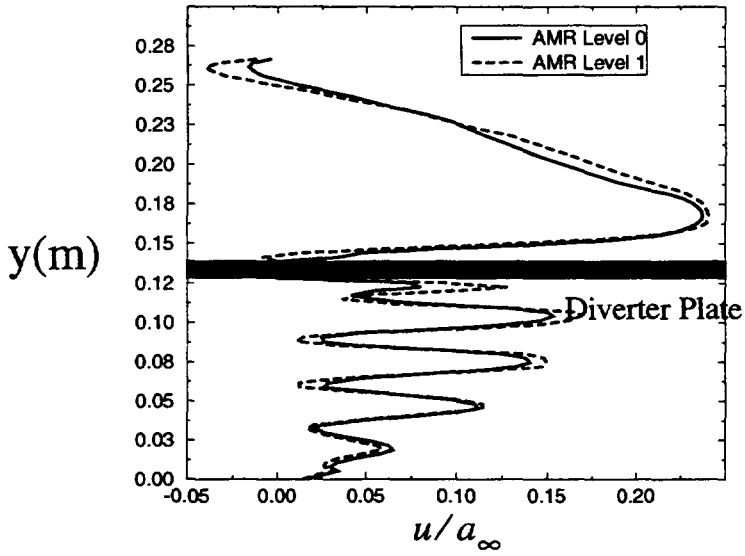


Figure 4.80 u-velocity at x=0.127 meters, High Reynolds Number Case

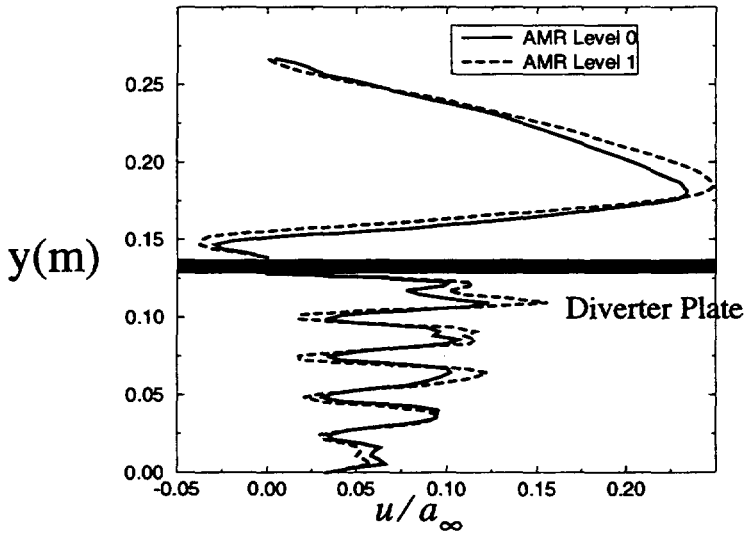


Figure 4.81 u-velocity at x=0.1524 meters, High Reynolds Number Case

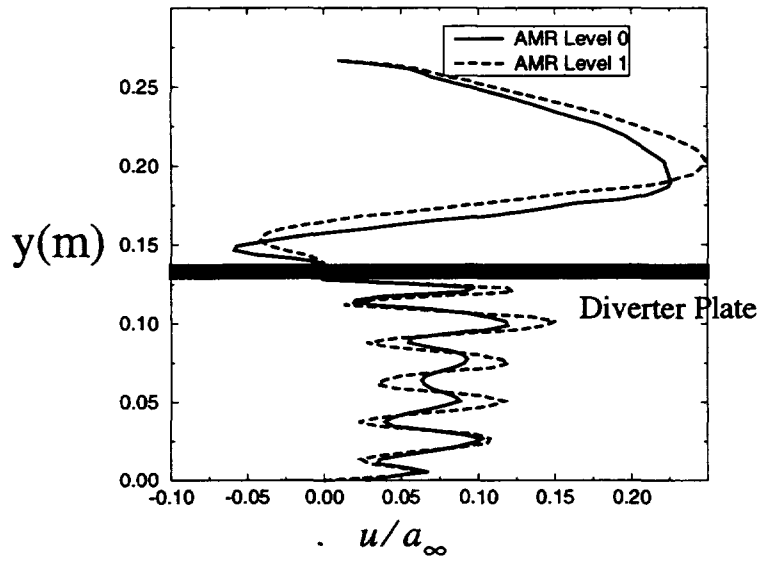


Figure 4.82 u-velocity at x=0.1794 meters, High Reynolds Number Case

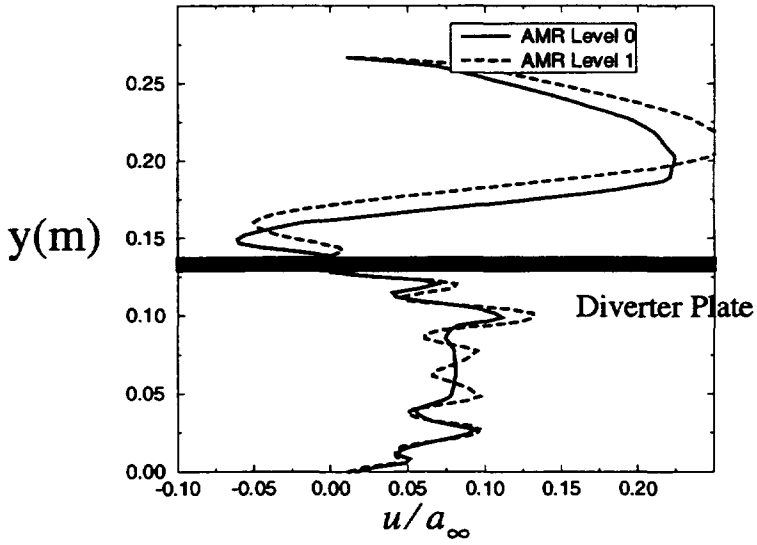


Figure 4.83 u-velocity at x=0.2 meters, High Reynolds Number Case

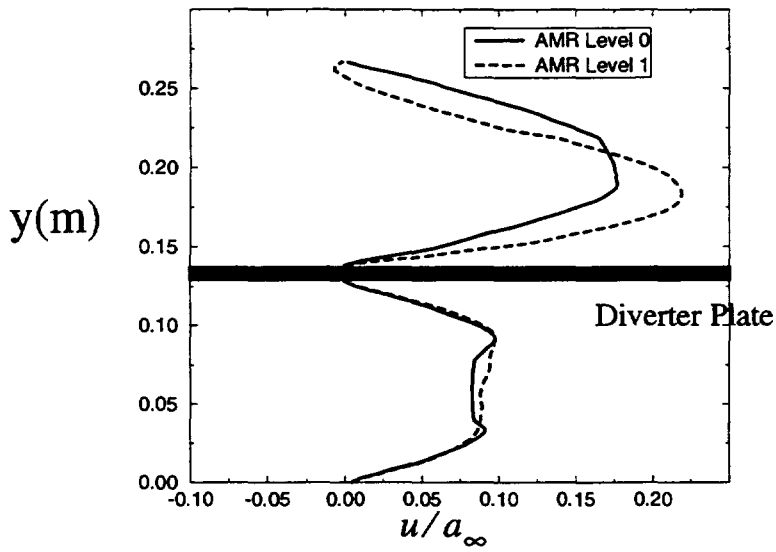


Figure 4.84 u-velocity at x=0.5 meters, High Reynolds Number Case

CHAPTER V
Level Distance Line Cutting and Stencil Creation with the Cartesian-Cell Approach

The very nature of the isotropic grids that the Cartesian-cell approach generates makes it unsuitable for computing high Reynolds number flows. The anisotropy of the length scales in a viscous layer requires that for reasonable resolution stretched cells must be used. This comes about from both an accuracy and an efficiency standpoint. From the analysis shown in Chapter III, the linearity-preserving schemes have a truncation error which varies inversely with the cell aspect ratio. This indicates that for grids which have the same sufficiently smooth stretching, the high aspect-ratio cells will have a lower truncation error. More importantly, isotropic refinement of the Cartesian cells to resolve gradients in the primary shear direction will unnecessarily refine cells in the streamwise direction. For a higher Reynolds number laminar flow this is a tremendous waste of cells, since the two length scales will differ by a factor of $\sqrt{Re}$. The situation gets even worse for turbulent flows, where typical turbulence models require resolution of the laminar sublayer. For the Cartesian approach, the use of a stretched, or non-unit aspect ratio, root cell can help alleviate the problem, but only for cases where the viscous layers are aligned with one of the coordinate axes. Since this in general can not be true, some means of creating body-aligned meshes must be found for the Cartesian, cell-based approach to be used for high Reynolds number applications.

One possible avenue to pursue would be to be resigned to the fact that the current viscous flux functions can't handle the grid non-smoothness and non-orthogonality well, and use body-fitted or prismatic meshes near no-slip boundaries. This is certainly a valid approach, and has led to what is currently being termed the hybrid-grid approach. The

term hybrid implicitly suggests that two types of grids are used, body-fitted-like grids near bodies and unstructured grids to fill the voids created between the body-fitted grids. Since the Cartesian approach can be viewed as a unstructured grid approach also, it too can be used to create a volume grid outside of the local body-fitted grids. This has been pursued by Melton et. al. in [54] where the Cartesian-grid approach was mated with a prismatic grid generator. In addition, a Cartesian-based approach was also investigated in the same light, for a hybrid grid approach, by Ward et. al in [86]. For completeness, a hybrid grid generated using the Cartesian, cell-based scheme presented in this thesis is shown in Figure 5.1 and Figure 5.2 for the fourteen pin fins of the branched duct geometry. The pin body-fitted grids were generated using an elliptic O-mesh generator that uses a mesh clustering algorithm in the radial direction. The outer boundaries of the pin meshes were input as bodies to the Cartesian grid generator, which automatically created the volume grid in the void between the body-fitted meshes. Although the grid is non-smooth near the outer boundaries of the body-fitted grids, this could be alleviated by better smoothness criterion for the Cartesian generator.

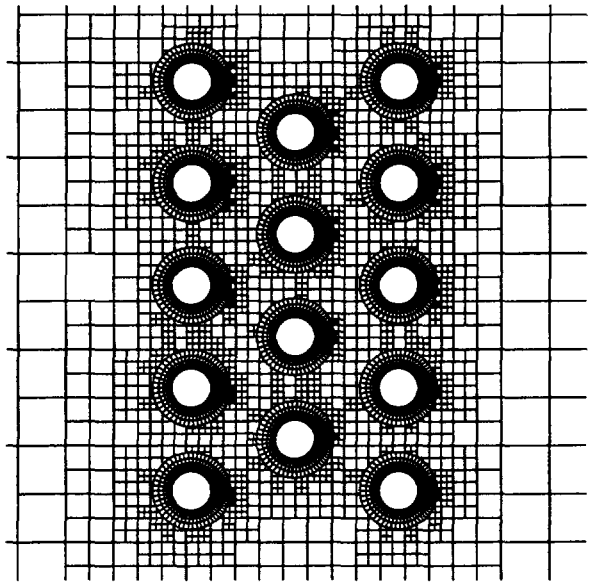


Figure 5.1 Example hybrid Cartesian/Body-Fitted mesh for the pin fin geometry.

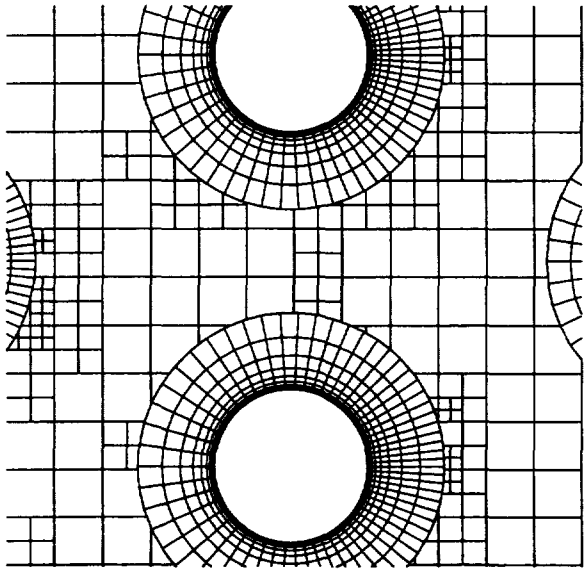


Figure 5.2 Close-up of Pin Region

This chapter investigates a new grid-generation procedure that is based upon the Cartesian-cell approach. Locally body-aligned cells are generated through a procedure called level-distance line cutting. This approach creates high aspect ratio cells near bodies by cutting faces out of the background mesh which correspond to iso-distance lines from the nearest no-slip surfaces. Near the body, the grids look much like a body fitted or collar grid, since, by construction, a stretching in the distance lines is easily specified. A consequence of this distance-cutting procedure is a well resolved, but extremely non-smooth, grid. As shown analytically in Chapter III and in practice in chapter IV, non-smooth grids can wreak havoc upon the current viscous flux functions in use today. In an attempt to alleviate this problem, a non-conservative procedure for the viscous terms in the Navier-Stokes equations is proposed. It is envisaged to solve the Navier-Stokes equations upon the distance-cut grids using this procedure, but the approach is only outlined here and demonstrated for a model equation. The heart of the procedure is the creation of a stencil from a given set of support cells. Two methods are used, if necessary, to create the stencils: one is based upon a linear solution of equations which yields a desired level of accuracy, but cannot guarantee positivity; the other is based upon using a quadratic

programming approach, which finds the stencil by satisfying a set of equality constraints (to create the stencil to a desired accuracy) and a set of inequality constraints (to ensure positivity of the stencil) by minimizing a quadratic objective function. This chapter outlines and demonstrates the grid-generation and describes the stencil-creation approaches.

5.1 Level Distance Line Cutting

This procedure is automated, once suitable stretching parameters are specified, and comprises two stages. The first stage of the procedure generates a Cartesian mesh suitable for an Euler calculation, based upon the grid-generation procedures outlined in appendix A and demonstrated in the preceding Chapters. The next stage finds, for each vertex in the mesh, the minimum distance to all no-slip surfaces. The minimum distance to all no-slip surfaces is found using a recursive bisection method for each surface definition segment on the no-slip surfaces. By defining a suitable stretching function in this distance coordinate, and a maximum distance to stretch to, stretched lines of constant distance from the body are identified, and are used to "cut" any cells which have faces that span this particular distance line. Figure 5.3 illustrates the cutting by showing a close-up of a coarse distance-function grid near the surface of a circular cylinder. In this Figure, two distance lines have been cut. Figure 5.4 shows the same view after more distance lines have been cut. This iso-distance cutting proceeds over all distance lines, which are determined by the stretching function, resulting in a mesh in which one can discern the background Cartesian grid, but is actually composed of many N-sided, non-Cartesian cells.

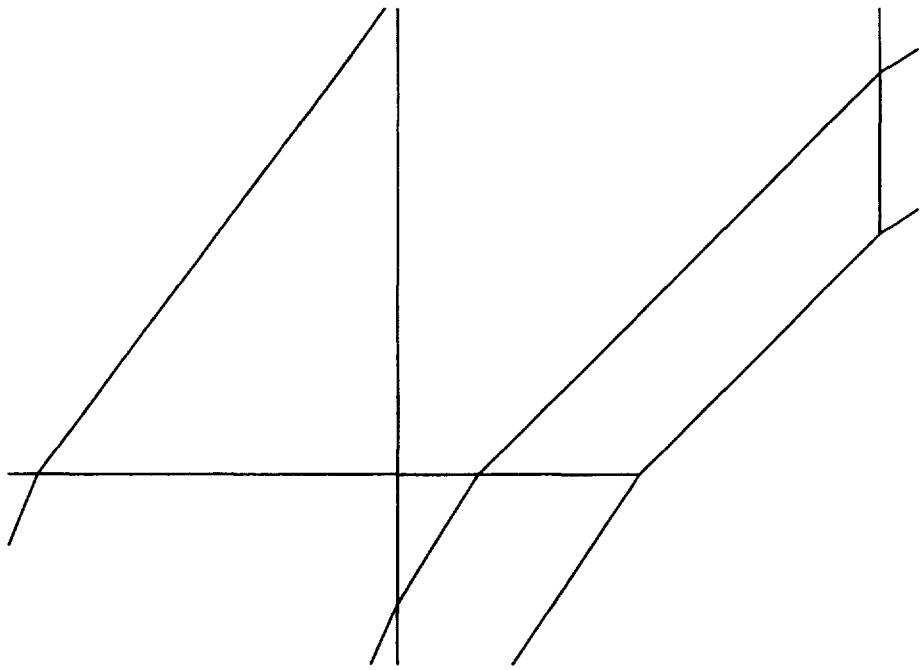


Figure 5.3 Two Level Distance Line Cut Grid Near Circular Cylinder

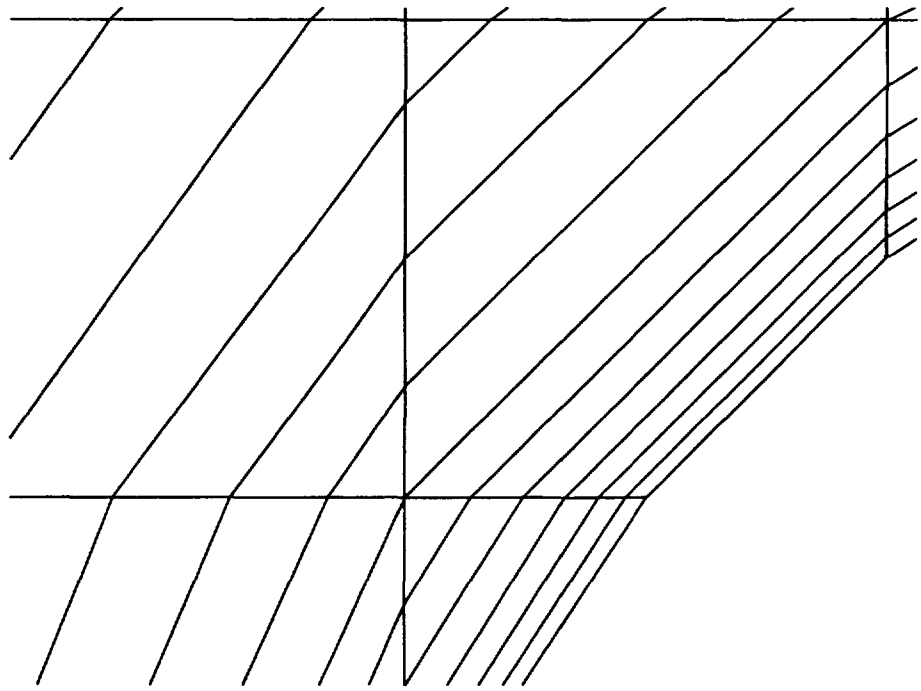


Figure 5.4 Distance Cut Mesh For Circular Cylinder with 11 Stretched, Distance Lines Cut

Each cell is split into two cells, where both of the new cells share a face which is aligned with the iso-distance level. The creation of two cells by splitting a single cell is easily handled by the binary tree.

The cutting procedure is straightforward to implement with the binary tree data structure. For a given iso-line to cut, each cell is visited, and each face is examined to see if this distance line is contained within the face. If a face contains this distance line, the location of the line in the face is found by linear interpolation between the two vertices. Once the two intersections with the cell have been found, each vertex of the cell is visited, and by comparison to the distance line, are found to lie on either side of the iso-distance face. This is used to create a list of nodes that comprises the vertices describing each cell. The tree is split below the parent cell, and both cut cells are now situated below the branch, as illustrated in Figure 5.5, where cell A, originally comprised of four vertices v0, v1, v2 and v3, is split into two cells, B and C. Cell B is described by the vertices v1, v2, v3 v5 and v4, while cell C is described by v4, v5 and v0. The new vertices created by the distance cutting, v4 and v5, are found by linear interpolation between the two existing vertices subtending the face where the distance line is to be cut.

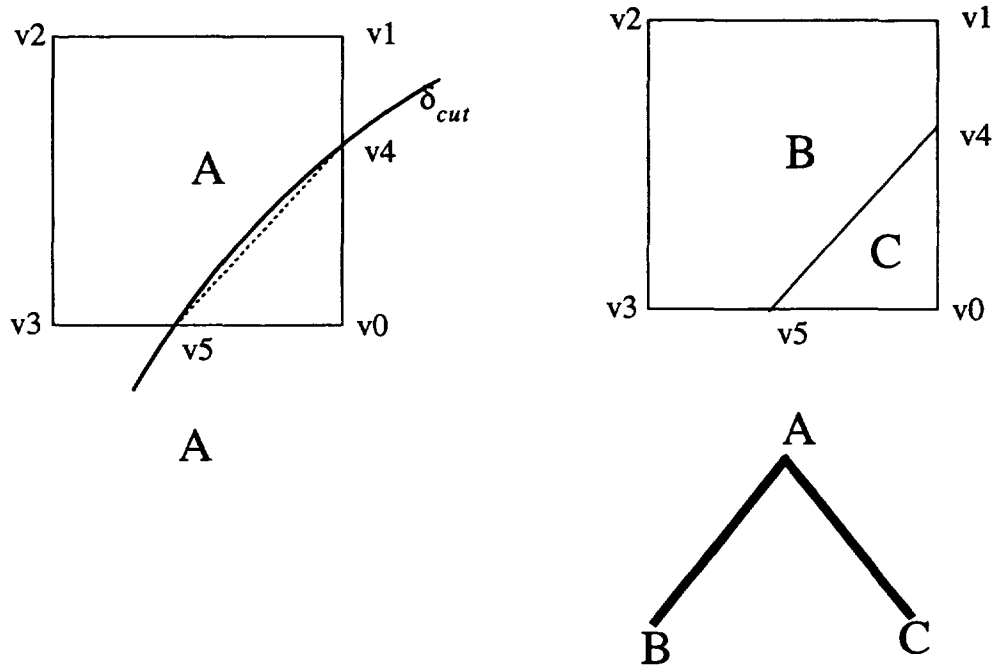


Figure 5.5 Illustration of Distance Cell Cutting Procedure

The iso-distance cutting procedure routinely generates cells interior to the mesh that have a many magnitude variation of cell area across cell faces. In many cases, the worst area variations come about when the iso-distance line comes extremely close to a vertex of the background Cartesian mesh. When this type of grid non-smoothness occurs, it can be eliminated by moving the distance line to the vertex of the background mesh, and eliminating the small cell. This is illustrated in Figure 5.6, where the small triangular cell that is created with a vertex of the background mesh, v0, and the distance line δ_{lift} is eliminated by moving the distance line to the vertex, and pruning the triangular cell from the tree. This procedure is termed iso-line lifting, since it "lifts" the iso-distance line to the vertex of the background, Cartesian mesh.

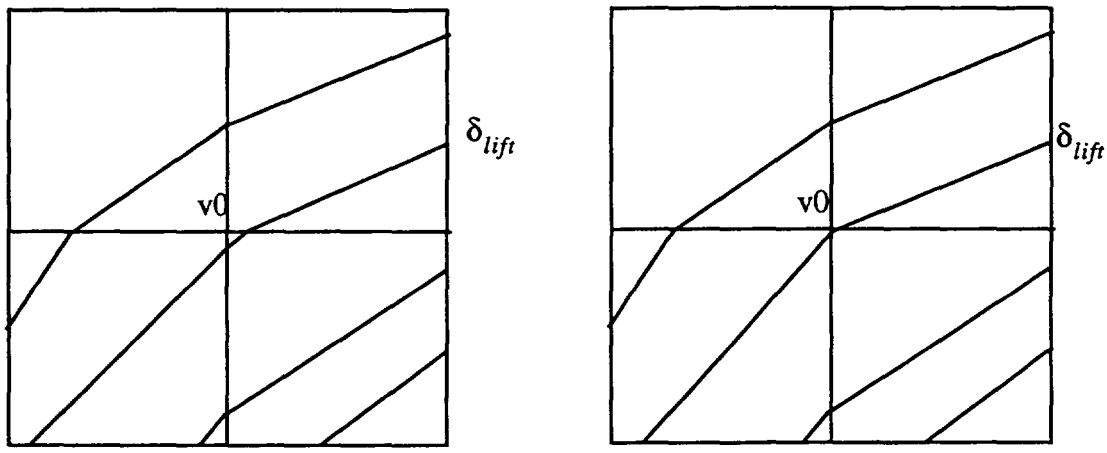


Figure 5.6 Iso-line Lifting to Eliminate Small Triangular Cells

Although this helps with the mesh smoothness, there are still many instances where extremely large variations in cell sizes across the faces can occur, which as shown in previous sections, can lead to extremely inaccurate and non-positive stencils. Indeed, as shown analytically and in practice, even a two-to-one variation in cell size can lead to serious inconsistencies in the viscous terms. In addition, the geometry of the cell-to cell interactions becomes extremely important, as it is best to keep the centroid-to-centroid lines nearly perpendicular to the faces. For many of the cells created by the distance cutting, all of these important properties are clearly violated.

There are only a few alternatives available, of which none are too attractive. One alternative is to stick with the current state of affairs of the viscous flux construction, and to try to smooth or modify the grid so that accurate stencils can be created. This avenue then implies a geometric way out of the problem, by moving mesh points and cells until some suitable criteria of grid smoothness based upon the accuracy of the stencil is achieved. The other approach would be to try to live with the grid non-smoothness, and create stencils which can handle the irregularities. The geometric approach is greatly hindered by the

inaccuracies inherent in the current viscous flux functions. Indeed, the current flux functions have been shown to give extremely poor results for model equations on distorted grids [37][67]and [26]. As shown here and backed up by the results in [37] and [67], the current classes of viscous flux functions are extremely sensitive to grid smoothness and are highly dependent upon geometric cancellations to obtain accuracy and consistency. More importantly, positivity has been shown to be extremely difficult to maintain on even the mildly non-smooth grids obtained through the isotropic cell refinement. When considering the non-isotropic nature of the grids obtained with the distance cutting procedures, it is highly unlikely that a robust and conservative procedure will be found. This is true in practice, as will be shown in a subsequent section. The alternative approach is to create a stencil which is both positive and accurate. The next section outlines a procedure that attempts to satisfy these two competing requirements.

5.2 Stencil Creation

The stencils created using a conservative flux formulation are found by first reconstructing gradients at the cell interfaces, and then performing a line integral about the cell to obtain the desired terms in the governing equation. A given reconstruction technique is analyzed by using the procedure to solve a model equation, in this case Laplace's, and then performing a Taylor series analysis using the created stencil. So, it is seen that the way a particular scheme is measured for quality is disconnected from the way the stencil is created: by reconstructing the gradients at the faces and then performing the line integral, the contributions of cells to the stencil are greatly complicated. Accuracy and positivity are a desired outcome of the stencil, but are not used explicitly to create the stencil.

The procedures outlined here discard the conservation property and attempt to create stencils which by construction satisfy positivity and/or accuracy. The accuracy and positivity conditions are well defined, although difficult to obtain. Accuracy is gauged by a local

Taylor series expansion. Reiterating the accuracy and positivity constraints explained in Chapter III, consider a discrete approximation to Laplace's equation with a set of N support cells as

$$\nabla^2 u \approx L(u) = \sum_{n=0}^{N} \alpha_n u_n = 0 \tag{5.1}$$

Positivity is guaranteed if $\alpha_0 < 0$ and $\alpha_n > 0$ for all $n = 1, N$. Accuracy is measured from a Taylor series expansion about the object cell as

$$L(u) = (\sum_n \alpha_n) + (\sum_n \alpha_n \xi_n) \frac{\partial u}{\partial x} + (\sum_n \alpha_n \eta_n) \frac{\partial u}{\partial y} + \frac{(\sum_n \alpha_n \xi^2{}_n)}{2} \frac{\partial^2 u}{\partial x^2} + \ldots \tag{5.2}$$

where $\xi_n = x_n - x_0$ and $\eta_n = y_n - y_0$. For the discrete approximation to be accurate, the grouped terms in the above equations comprise a set of linear relations

$$\sum_n \alpha_n = 0 \tag{5.3}$$

$$\sum_n \alpha_n \xi_n = 0 \tag{5.4}$$

$$\sum_n \alpha_n \eta_n = 0 \tag{5.5}$$

$$\sum_n \alpha_n \xi^2{}_n = 2 \tag{5.6}$$

$$\sum_n \alpha_n \xi_n \eta_n = 0 \tag{5.7}$$

$$\sum_n \alpha_n \eta^2{}_n = 2 \tag{5.8}$$

$$\sum_n \alpha_n \xi^3{}_n = 0 \tag{5.9}$$

$$\sum_n \alpha_n \xi^2{}_n \eta_n = 0 \tag{5.10}$$

$$\sum_n \alpha_n \xi_n \eta^2{}_n = 0 \tag{5.11}$$

$$\sum_n \alpha_n \eta^3{}_n = 0 \tag{5.12}$$

A first-order accurate stencil will satisfy six equality constraints, (5.3) to (5.8), while a second-order accurate stencil must satisfy ten, (5.3)to (5.12).

5.2.1 An Accuracy-Preserving Laplacian

This technique expands upon the ideas presented for the linearity-preserving Laplacian developed by Holmes and Connell in [35]. Their approach is shown here to be a special case of this technique. A discrete approximation to Laplace's equation is formulated as

$$L(u) = \sum_{n=1}^{N} \omega_n (u_n - u_0) \tag{5.13}$$

The Taylor series approximation is expanded out to be

$$\begin{aligned} L(u) = & [\sum \omega_n (u_0 - u_0)] + [\sum \omega_n (x_n - x_0)] u_x \\ & [\sum \omega_n (y_n - y_0)] u_y + [\sum \omega_n (x_n - x_0)^2] \frac{u_{xx}}{2} + \\ & [\sum \omega_n (x_n - x_0)\ (y_n - y_0)] u_{xy} + [\sum \omega_n (y_n - y_0)^2] \frac{u_{yy}}{2} + \ldots \end{aligned} \tag{5.14}$$

Each of the square bracketed terms can be considered to be the discrete Laplacian applied to a different function. That is

$$L(1) = 0 \tag{5.15}$$

$$L(x) = 0 \tag{5.16}$$

$$L(y) = 0 \tag{5.17}$$

$$L(x^2) = 2 \tag{5.18}$$

$$L(xy) = 0 \tag{5.19}$$

$$L(y^2) = 2 \tag{5.20}$$

$$L(x^3) = 0 \tag{5.21}$$

$$L(x^2y) = 0 \tag{5.22}$$

$$L(xy^2) = 0 \tag{5.23}$$

$$L(y^3) = 0 \tag{5.24}$$

If the Laplacian weights are expanded about unity as

$$\begin{aligned}\omega_n = 1 &+ \lambda_x(x_n - x_0) + \lambda_y(y_n - y_0) + \gamma_x(x_n - x_0)^2 + \\ &\alpha(x_n - x_0)(y_n - y_0) + \gamma_y(y_n - y_0)^2 + \\ &\delta_1(x_n - x_0)^3 + \delta_2(x_n - x_0)^2(y_n - y_0) + \\ &\delta_3(x_n - x_0)(y_n - y_0)^2 + \delta_4(y_n - y_0)^3\end{aligned} \tag{5.25}$$

then a linear system results for the coefficients in (5.25). If the expansion of ω_n is formulated as

$$\omega_n = 1 + \lambda_x\phi_1 + \lambda_y\phi_2 + \gamma_x\phi_3 + \alpha\phi_4 + \gamma_y\phi_5 + \ldots \tag{5.26}$$

then the linear system is Ax=s-b, where

$$A_{ij} = \sum \phi_i\phi_j \tag{5.27}$$

$$x = (\lambda_x, \lambda_y, \gamma_x, \alpha, \gamma_y, \delta_1, \delta_2, \delta_3, \delta_4)^T \tag{5.28}$$

$$b_i = \sum \phi_i \tag{5.29}$$

For Laplace's equation the vector s is

$$s = (0, 0, 2, 0, 2, 0, 0, 0, 0) \tag{5.30}$$

Second-order accuracy can be obtained by solving the full system, or lesser accuracy by reducing the number of unknowns. A first-order accurate Laplacian can be obtained by

solving for the first five unknowns, and a zeroth-order Laplacian, corresponding to the linearity preserving Laplacian developed by [35], is found by only solving for the first two unknowns in the expansion. This technique can be applied, in principle, for solving arbitrary PDE's where the constant vector s is replaced by the correct coefficients corresponding to the desired pde.

Application of this approach to the model grids analyzed in chapter III results in some very interesting stencils. First, applying the zeroth-, first-, and second-order accuracy preserving Laplacians to the uniform grid results in the following stencils.

$$L(u) = \frac{1}{h^2}$$

1	1	1
1	-8	1
1	1	1

$$= 3\nabla^2 u + \frac{h^2}{4}(u_{xxxx} + 4u_{xxyy} + u_{yyyy})$$

Figure 5.7 Stencil for Zeroth-order ("Linearity Preserving") Laplacian: Uniform Grid

$$L(u) = \frac{1}{5h^2}$$

1	3	1
3	-16	3
1	3	1

$$= \nabla^2 u + \frac{h^2}{60}(5u_{xxxx} + 12u_{xxyy} + 5u_{yyyy})$$

Figure 5.8 Stencil for First-order and Second-order preserving Laplacian: Uniform Grid

The stencil obtained for the linearity preserving Laplacian is horribly inconsistent, as expected, while the first- and second-order Laplacians are both second-order accurate, and can be viewed as a linear combination of two second-order accurate stencils. If the standard Laplacian is termed L_{st} and the rotated Laplacian L_{rot}, then the stencil created using this procedure is the linear combination

$$L = \frac{3}{5}L_{st} + \frac{2}{5}L_{rot} \tag{5.31}$$

For the East face refined grid, application of the zeroth-, first- and second-order accuracy preserving Laplacian procedure results in the following stencils.

$$L(u) = \frac{1}{49h^2} *$$

53	49	45	
53	-439	46	0
		46	0
53	49	45	

$$= \frac{1}{392}(1199u_{xx} + 1203u_{yy}) + h(\frac{-23}{224}u_{xxx} + -\frac{187}{1568}u_{xyy}) + ..$$

Figure 5.9 Zeroth-order Accurate Stencil for East Refined Grid

$$L(u) = \frac{1}{6751h^2} *$$

1697	4272	509	
4176	-25868	4368	0
		4368	0
1697	4272	509	

$$= \nabla^2 u + \frac{h}{27004}(-1911u_{xxx} - 3933u_{xyy}) + .$$

Figure 5.10 First-order Accurate Stencil for East Refined Grid

$$L(u) = \frac{1}{h^2} *$$

1/2	0	1/2	
0	-2	0	0
		0	0
1/2	0	1/2	

$$= \nabla^2 u + O(h^2) + \ldots$$

Figure 5.11 Second-order Accurate Stencil for East Refined Grid

The truncation error of the stencils behaves as expected, but the weights are not ones which are so obvious. Quite unfortunately, the rotated Laplacian appears for the second-order stencil. This is unfortunate since this configuration is one to avoid as it allows the development of decoupled solutions. Although none of the stencils shown here are non-positive, there is no mechanism to preclude non-positive stencils from being created. This turns out to be true in practice, but on many grid topologies, this procedure alone is sufficient. For those where positivity is not achievable, the following approach is a valid, though costly, alternative.

5.2.2 A Quadratic-Programming Approach to Stencil Creation

The approach here is to use a quadratic-programming method to solve the linear equality conditions (5.3) to (5.12) subject to a set of inequality conditions, namely positivity. Since the development of an efficient quadratic-programming solver is beyond the scope of this thesis, a packaged solver is used instead. The particular solver was obtained from the Internet, and is called DONLP for Do Non-Linear Programming[74]. The objective function is quite arbitrary, and here it is chosen to minimize the sum of the squares of all non-face neighbors to the cell in question. This choice of objective function returns the

desired standard Laplacian on a uniform grid and gives reasonable results in practice. For the East face refined grid, the stencil weights obtained are shown in Figure 5.12. The resulting modified equation is

$$L(u) = \nabla^2 u + (-0.0155 u_{xxx} + 0.0118 u_{xyy})\, h^2 + \ldots \tag{5.32}$$

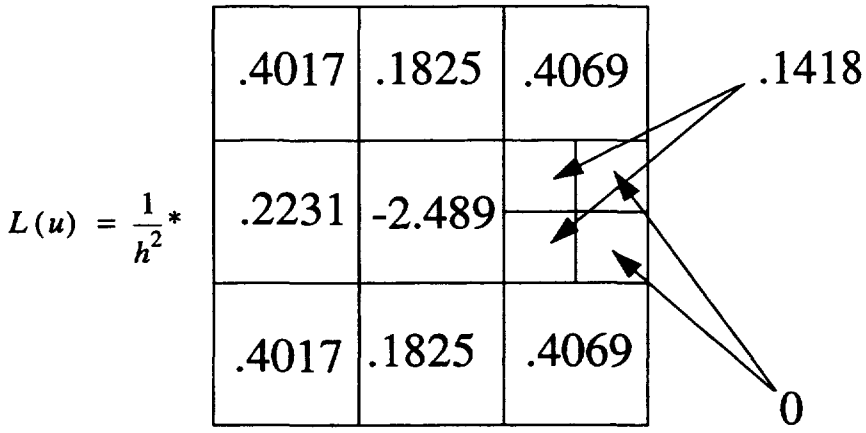


Figure 5.12 Second-order Accurate Stencil for East Refined Grid: Quadratic Programming Approach

This approach is in no ways optimal. There appear to be many objective functions that one may choose to minimize that will give different stencil weights. Indeed, the quadratic programming approach might not be the most efficient means to solve the constraints. The approach shown here is only a demonstration of the concept and is only a small step towards a more complete methodology.

CHAPTER VI
Concluding Remarks

6.1 Summary

In this thesis a Cartesian, cell-based scheme has been presented which solves the compressible Euler and Navier-Stokes equations using a cell-centered, upwind, finite-volume approach. The equations have been solved in conservation-law form upon an unstructured network of cells which were created using a Cartesian, cell-based grid-generation procedure. This method used a recursive subdivision of unit aspect ratio (Cartesian) cells creating arbitrarily shaped polygons when the Cartesian cells straddle a boundary, using a modified polygon clipping algorithm. The grid has been stored in a hierarchical data structure; a binary data tree. Storage of the grid in this data structure allows solution-adaptive mesh refinement by performing pruning and growth operations upon the tree branches and allows cell-to-cell connectivity to be inferred by a logical traversal of the tree. The grid-generation process is automatic, once suitable representations of the boundary surfaces are defined, and is able to produce volume grids about complicated geometries with minimal user intervention.

The Euler and Navier-Stokes equations have been solved in conservation law form using a finite-volume formulation. The convective terms were treated in an upwinded manner: A gradient-limited, linear reconstruction of the primitive variables in each cell was performed which provides input states to an approximate Riemann solver at each cell-to-cell interface. The semi-discrete form of the equations were solved using an explicit, multi-stage scheme with a spatially varying time step. The Euler solver was validated and its accuracy critically assessed by comparison to accepted computational results and to an exact solution of the Euler equations.

An analytical assessment of six conservative, viscous flux functions suitable for use

with the Cartesian, cell-based solver has been made. All of the schemes reconstructed gradients of the primitive variables at the cell interfaces. Four of the viscous flux schemes were based upon an application of the divergence theorem to a co-volume surrounding the face. These schemes were termed as Green-Gauss type reconstructions and were delineated by the co-volumes used and the means of obtaining the data at the co-volume vertices. The two remaining schemes were based upon reconstructing either a linear or quadratic polynomial at the interfaces, and differentiating the polynomial to obtain the gradients. This assessment compared the candidate schemes by solving a model equation of the viscous terms, Laplace's equation, using the candidate flux functions upon some grid topologies representative of those that may be obtained using the Cartesian approach. The stencils created by these flux functions were assessed for accuracy (by local Taylor series expansion), and for positivity (by examination of the stencil coefficients).

Two of the Green-Gauss type schemes were shown to yield decoupled stencils upon uniform and non-uniform Cartesian grids. The two remaining Green-Gauss schemes used identical co-volumes, but differed by the means of obtaining the data at the vertices of the co-volume. The simplest approach, using a unity weighting, was shown to be highly inaccurate, giving stencils with a leading truncation error term that varied inversely with the cell size, precluding grid convergence. The Green-Gauss approach using a linearity-preserving weighting was shown to be inconsistent upon arbitrary meshes. The linear, polynomial-based reconstruction was shown to exhibit the same general behavior as the linearity-preserving Green-Gauss reconstruction. The quadratic, polynomial-based reconstruction approach was shown to be the only scheme able to give consistent, first-order accurate stencils upon arbitrarily distorted meshes. A complete Taylor series analysis using a conservative type of reconstruction of gradients at the cell interfaces also showed that a quadratic reconstruction is the only means of obtaining a first-order accurate stencil upon arbitrarily distorted meshes. From the discrete analysis, all of the schemes were shown to have a tendency to yield non-positive stencils.

The two best reconstruction schemes were used to compute adaptively-refined solutions to the Navier-stokes equations for a series of low and moderate Reynolds number flows. The results computed using the two reconstruction schemes were compared to each other, to theory, to accepted computational results and to experiment. Both schemes were shown to give excellent results when sufficiently resolved and smoothed grids were used. A means of obtaining discrete Taylor-series expansions for the stencils created by each of the schemes was presented and used to compare the quality of the stencils created by the two approaches. For the Cartesian-generated grids, both methods yielded nearly identical results, as was indicated by the discrete analysis, and was directly attributed to the geometric quality of the grids obtained using the Cartesian-based cells. Neither scheme, without modification, could yield positive stencils upon all of the grids, and for arbitrary cut grids this non-positivity inhibited convergence. The quadratic scheme was shown to give fewer non-positive stencils than the Green-Gauss type scheme, but was also shown to give cells with the largest magnitude of non-positivity. Although more non-positive stencils were created with the Green-Gauss scheme, these stencils were less non-positive than the quadratic approach, making the linearity-preserving Green-Gauss type of reconstruction the most robust.

It was shown that with sufficiently resolved grids, excellent quality results of the mean flow quantities could be obtained, but derivative quantities of the computed results were extremely non-smooth, and very sensitive to grid quality. This made the skin-friction obtained from most calculations extremely oscillatory and for the most part, unusable. Although smooth mean flow results could be obtained, a better means of treating cut cells and of computing the derivative quantities upon them is still lacking. This is directly attributed to the state of the art of the computation of the viscous flux terms in the Navier-Stokes equations, and is a pacing item for other unstructured grid based approaches, in addition to the Cartesian approach.

An obvious drawback to the Cartesian approach is its inadequacy in treating high Rey-

nolds number flows. For a high Reynolds number flow, stretched grids must be used from both an accuracy and an efficiency standpoint. Since the Cartesian approach uses unit aspect-ratio cells, application of the Cartesian approach to a high Reynolds number flow is extremely computationally inefficient. Two alternatives, both based on the Cartesian approach, were presented. One was based upon using a hybrid Cartesian/body-fitted mesh approach. This essentially tries to accommodate the smoothness and orthogonality requirements needed by the current viscous flux formulae, creating locally body-fitted meshes near the bodies. This would be a valid short term solution, in that more emphasis is placed upon the grid generation, and uses the currently available viscous flux formulae. A drawback of this approach is that the grid generation process is no longer automatic, requiring the generation of the local body-fitted meshes. An alternative method was presented which still retained the autonomy of the grid generation procedure, but places an extreme strain on the viscous flux construction. This method was based upon the construction of locally- aligned cell faces by cutting iso-distance lines from the bodies out of the mesh. This approach created extremely non-smooth grids, which have been shown to be very poor from both an accuracy and positivity standpoint using current viscous flux formulae. Due to this, a non-conservative method was presented and only minimally tested, based upon a stencil creation procedure using both a linear method and quadratic-optimization approach. The linearity-preserving Laplacian weighting of [35] was shown to be a subset of this approach.

In summary, the Cartesian, cell-based approach has been shown here to provide adaptively-refined solutions to the Navier-Stokes equations upon automatically generated grids. The mesh refinement was shown to improve the solution quality, also in an automated fashion. This demonstrated a new method, where adaptively-refined solutions to the Navier-Stokes equations can be obtained upon complex domains with minimal user intervention. A series of cell-centered, viscous flux formulae have been investigated for use in the Cartesian, cell-based scheme. The assessments were made analytically, with specific

focus paid upon the accuracy and positivity of the schemes when used to construct discrete Laplacian operators. Two flux functions, representative of linearity- and quadratic-preserving reconstructions schemes, were used to compute adaptively-refined solutions of the Navier-Stokes equations. The linearity preserving scheme was shown in practice, and by analysis, to be the more robust of the schemes. Neither scheme could guarantee positivity of the viscous operators, which inhibited convergence and decreased the overall robustness of the solver. Regardless of this comparatively negative finding, the approach can yield excellent solutions, and has been demonstrated to provide automatically generated solutions to the Navier-Stokes equations for complex domains at low and moderate Reynolds numbers.

6.2 Concluding Remarks

The bulk of this thesis, that is Chapters III, IV and V, sheds light upon a recurrent struggle apparent in all modern numerical methods for conservations laws: the close coupling of the numerical scheme to the geometric qualities of the conservation volumes. This close coupling is tested to an extreme by the very nature of the Cartesian-based grid generation and the adaptive-mesh refinement. This thesis has shown how difficult it is to attain both accurate and positive viscous flux functions on arbitrary grids, and has illustrated how difficult this dichotomy, grid smoothness opposing flux accuracy, is to overcome. This dichotomy is one that is inherent in all modern methods, both structured and unstructured. The success of the structured grid based approach is directly tied to the quality of the grids that can be attained coupled with the simplicity of the flux functions used upon them. Traditional unstructured mesh schemes suffer from similar constraints, also testing the coupling between the numerics and grid smoothness. Both approaches have placed positivity of the operators at a higher priority than accuracy. This sacrifice of accuracy can still yield useful trends, as long as the grids used are not too non-smooth, which, unfortuneately, can be very problem dependent.

The Cartesian approach presented here severely tests the limits of these constraints. The automation of the grid generation has come at the cost of grid smoothness. For the inviscid portion of the scheme, this non-smoothness is not as crucial, due to the decoupling of the reconstruction process from the grid structure, and the use of a spatially varying time step. But, here it is shown that it is very difficult to attain both an accurate and positive viscous operator on the comparatively smooth irregularities induced across refinement boundaries, let alone the extreme non-smoothness induced by cell cutting. For a high Reynolds number flow, the non-positivity of the viscous operator, induced by the grid non-smoothness, might not be as apparent. This is a deceptive, and possibly dangerous, scenario. Although an overall positivity preserving operator might be attained, the non-positivity of the viscous operator can be masked by the inviscid operator, clearly violating the important, physically based, positivity property of the viscous terms that must be maintained. Unfortunately, application of the Cartesian approach to high Reynolds number flows is an extremely inefficient proposition, due to the isotropic nature of the grids, so even this somewhat tainted saving grace can't come to its rescue. The best hope for a robust and accurate Cartesian, cell-based approach for computing viscous flows is a radically new treatment of the viscous terms, perhaps following that outlined in Chapter V.

6.3 Recommended Future Efforts

Firstly, the inadequacy of the viscous flux formulae for generally distorted grids must be addressed. The current flux formulae require a smooth grid to get reasonable solutions. If this smoothness constraint could be lifted, the grid-generation process could be greatly simplified. Even though the resulting flux formulae might be more expensive to evaluate and store, the gains in simplicity of mesh generation could far outweigh the extra costs incurred. Work in this area could have a wide reaching impact, as improvement in the flux formulae could be used by many already-developed solvers, both structured and unstructured.

For the Cartesian, cell-based approach, probably the most pressing need is the development of a three-dimensional upwind-based inviscid flow solver. This is crucially dependent upon development of a common three-dimensional grid generator. As in the work completed for this thesis, the development of a robust grid-generation procedure for arbitrary geometries is most certainly not a trivial task, but is one that can have a great payoff. Current work is underway developing a NURBS-based Cartesian grid generator, and should provide a common, publicly available Cartesian-cell grid-generation capability. Due to the relatively large (two-to-one) change in cell size across refinement boundaries and the intractably large changes near cut-cell boundaries, a more complex and robust viscous flux function will be needed. If a new flux function is not created, a hybrid Cartesian/ body-fitted approach is a viable alternative, provided a suitable smoothness is maintained across the body-fitted to Cartesian interfaces.

Extension of the Cartesian based approach to solving other important problems in computational physics, such as acoustics and electromagnetics and radiation should prove useful. As with any flow solver, this approach can readily be extended to reacting and turbulent flows. Regardless of the equation set being solved, if the isotropic resolution of disparate length scales for solutions in and about complicated domains is desired, the Cartesian-cell based approach is a valuable and viable tool that can be applied. It is shown here to be able to provide automatically-generated and grid-refined solutions for the Navier-Stokes equations for both complex flows and geometries.

APPENDICES

Appendix A
Cartesian, Cell-Based Grid Generation Using a Polygon Clipping Algorithm

The basic idea behind the Cartesian, cell-based grid generation approach is to create arbitrarily shaped cells wherever Cartesian cells from the background mesh are intersected by a boundary of the domain. The domain may consist of an arbitrary number of closed surfaces which may be described functionally and whose orientation of control points implicitly determines the location of the computational domain. The non-Cartesian cells which are created by the grid generation procedure are termed cut cells since they can be viewed as being Cartesian cells which are "cut" out of the background mesh by the boundary. The robustness of the procedure used to create the cut cells rises to extreme importance: If a single boundary/cell topology fails to create the proper cut cell, the entire mesh is unusable. This is further complicated by the fact that many of the boundaries may contain discontinuities in slope and in many cases, non-convex cut cells must be created. Since the flow solution approach solves conservation laws in all cells in the domain by performing a second- or higher-order reconstruction followed by a flux quadrature over the cell edges, the outcome of the cell cutting procedure must be a list of vertices or edges which describe the cut cells, yielding the quadrature points and edge lengths.

The cell cutting procedures implemented for this work have consisted of two separate approaches. The first approach used was based upon creating a list of vertices describing the cut cell from the vertices of the Cartesian cell and the intersections of this cell with the boundary. Each Cartesian cell vertex was examined for locality by determining whether the vertex is located inside, on, or outside of the body, and based on this locality deciding

if it should be added to the list of vertices which describe the cut cell. This list of vertices is then ordered in a counterclockwise manner about a mean point of the cell. From this list, the cell centroid and cell edge quadrature points are easily found. This approach works well when convex cells need to be created and when it is unnecessary to preserve discontinuous breaks in the geometry. But, situations where there are discontinuities in slope on the bodies occur quite regularly, and it is necessary to correctly preserve these breaks in the surfaces to properly model the flow. A more robust and applicable cell cutting procedure is implemented here which is based upon the concept of polygon clipping, taking advantage of the convexity of the Cartesian cell and using many procedures developed in the computational graphics field. This procedure preserves all breaks in the bodies and in practice proves to be a more robust procedure than the former, locality based approach. Indeed, as can be seen from the following analysis, the former method can be viewed as a simpler polygon clipping algorithm.

The concepts behind polygon clipping are simple: Polygon clipping performs a Boolean operation on a set of polygons, returning a list of vertices describing polygon(s) that result from the desired operator. The more complicated clipping algorithms are able to return multiple polygons from the Boolean operations upon convex and non-convex polygons that may contain holes and coincident edges.

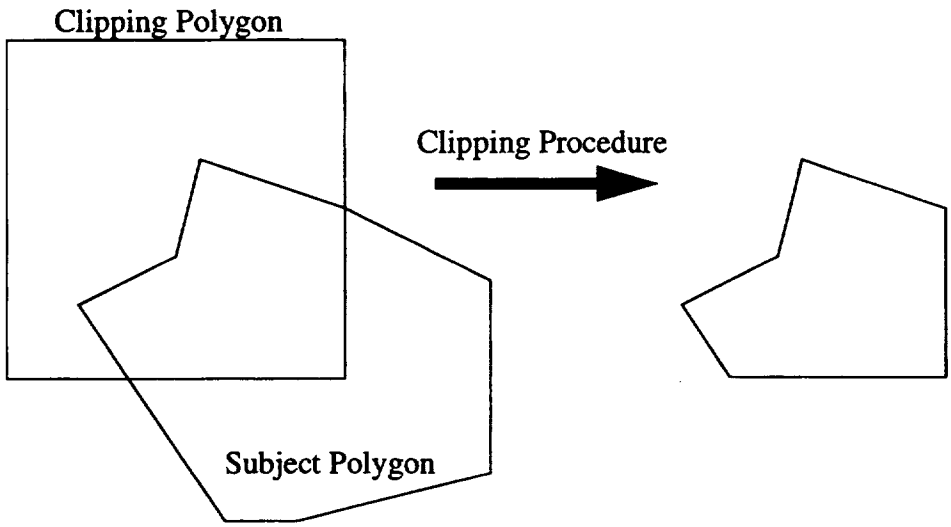


Figure A.1 Boolean and Operation using a Polygon Clipping Algorithm

The polygon clipping algorithm used here is based upon the clipper proposed by Sutherland and Hodgman [76], and is one of the more simple clippers in that it requires the clipping polygon to be convex and will only return one polygon from the operation. In addition, without modification, this algorithm only returns the logical and of the clipping.

The procedure takes a subject polygon and "clips" it against a convex clipping polygon, which returns the logical and of the two regions (Figure A.1). The clipping is performed on an edge by edge basis of the clipping polygon, and determines if vertices of the subject polygon lie logically to the left or right of the clipping edge, creating a new polygon from the points on the left and the intersections of the current subject edge with the clipping edge. The subject polygon resulting from clipping against the previous edge is used in the clipping against the next edge, until there are no edges left which must be clipped, yielding the desired polygon. The subject and clipping polygons are positively ordered in a counter-clockwise manner, which determines the handedness of the clipping test and the logical "in/out" location of points on the clipping polygon relative to the subject polygon. The input to the operation is the clipping and subject polygons, while the output is the clipped polygon. If the output polygon is empty, there is no intersection. For

each edge to clip against, the handedness of an arbitrary point is easily found by taking the cross product of the vector along the clipping edge with the vector originating from the beginning of the edge to the point in question. For a clipping edge described by the vector along point A to point B, the handedness of the test point, P is found from Figure A.2.

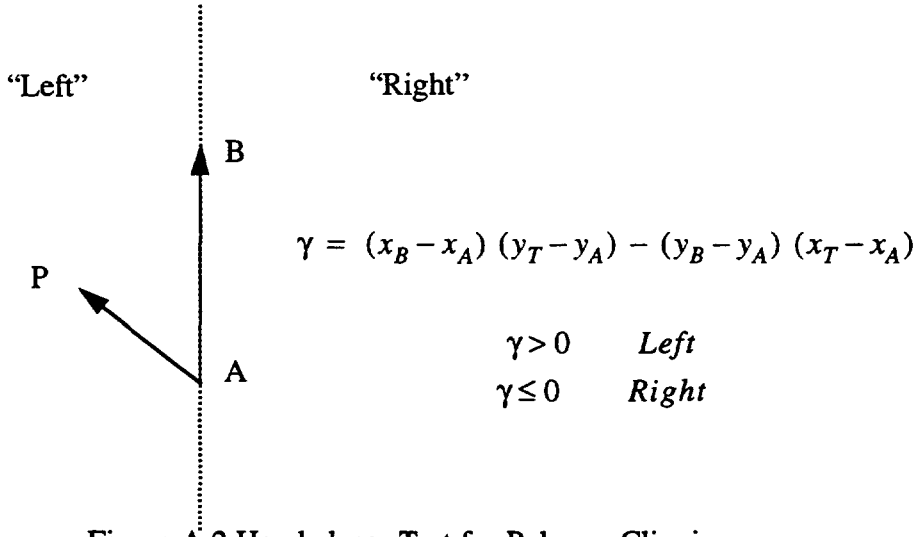


Figure A.2 Handedness Test for Polygon Clipping

For each edge of the current subject polygon, connecting point 1 to point 2, the handedness of the points in relation to the current clipping edge is used to determine whether or not to add point 1 and/or point 2 and/or the intersection of the subject edge with the clipping edge to the output list. The logic behind the Sutherland-Hodgman clipping algorithm is based upon the four possible combinations of handedness of the points 1 and 2, which are shown in Figure A.3 to Figure A.6.

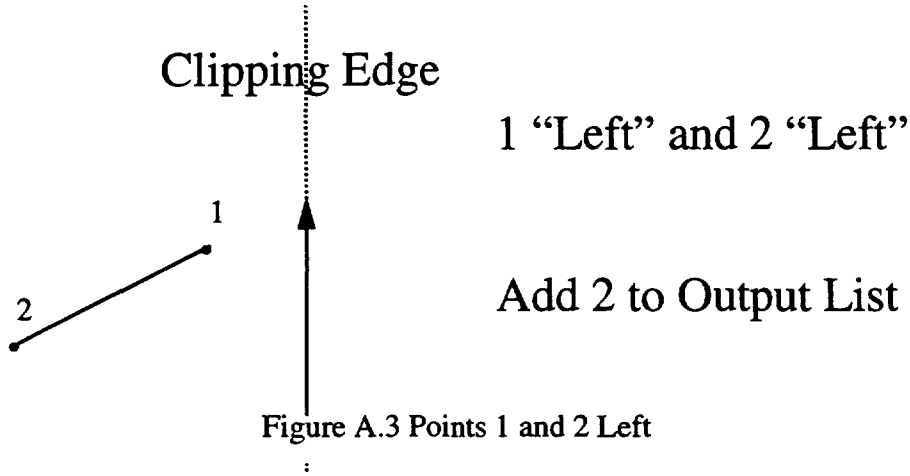


Figure A.3 Points 1 and 2 Left

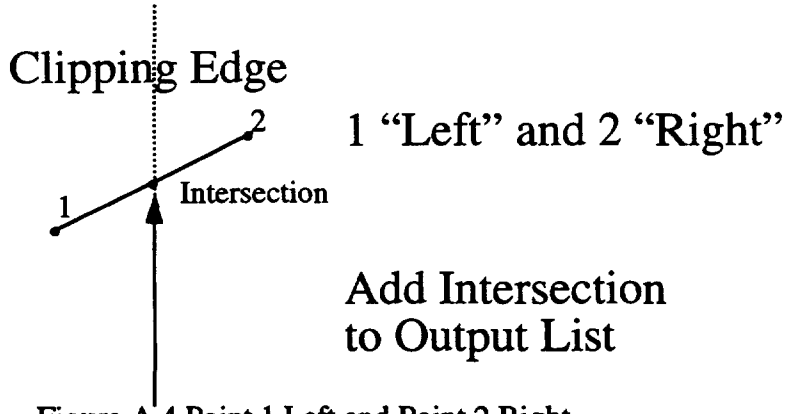


Figure A.4 Point 1 Left and Point 2 Right

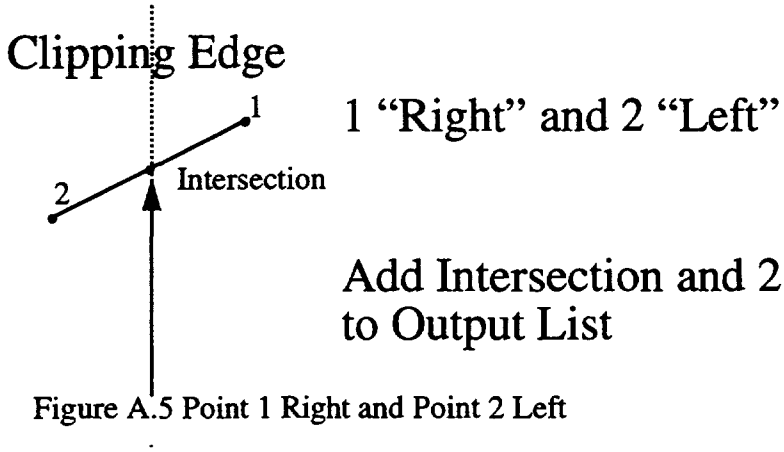


Figure A.5 Point 1 Right and Point 2 Left

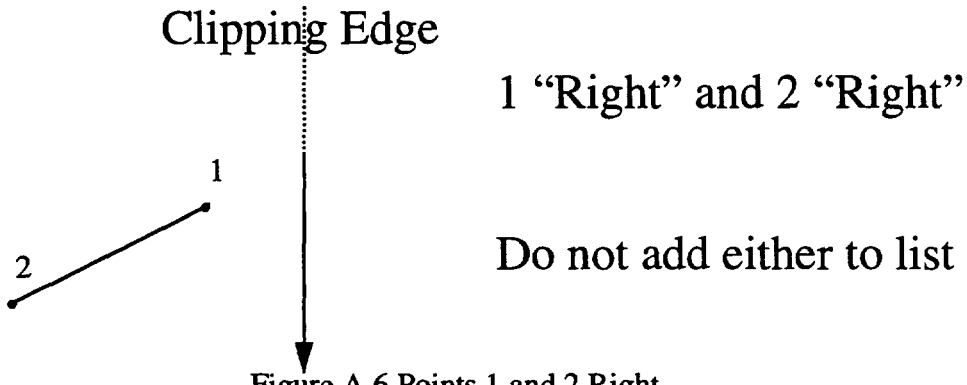


Figure A.6 Points 1 and 2 Right

For the polygon shown in Figure A.1, the clipping procedure applied against the Cartesian clipping cell proceeds by clipping against the East then North then West and then South faces, shown, respectively, in steps 1 to 4 in Figure A.7.

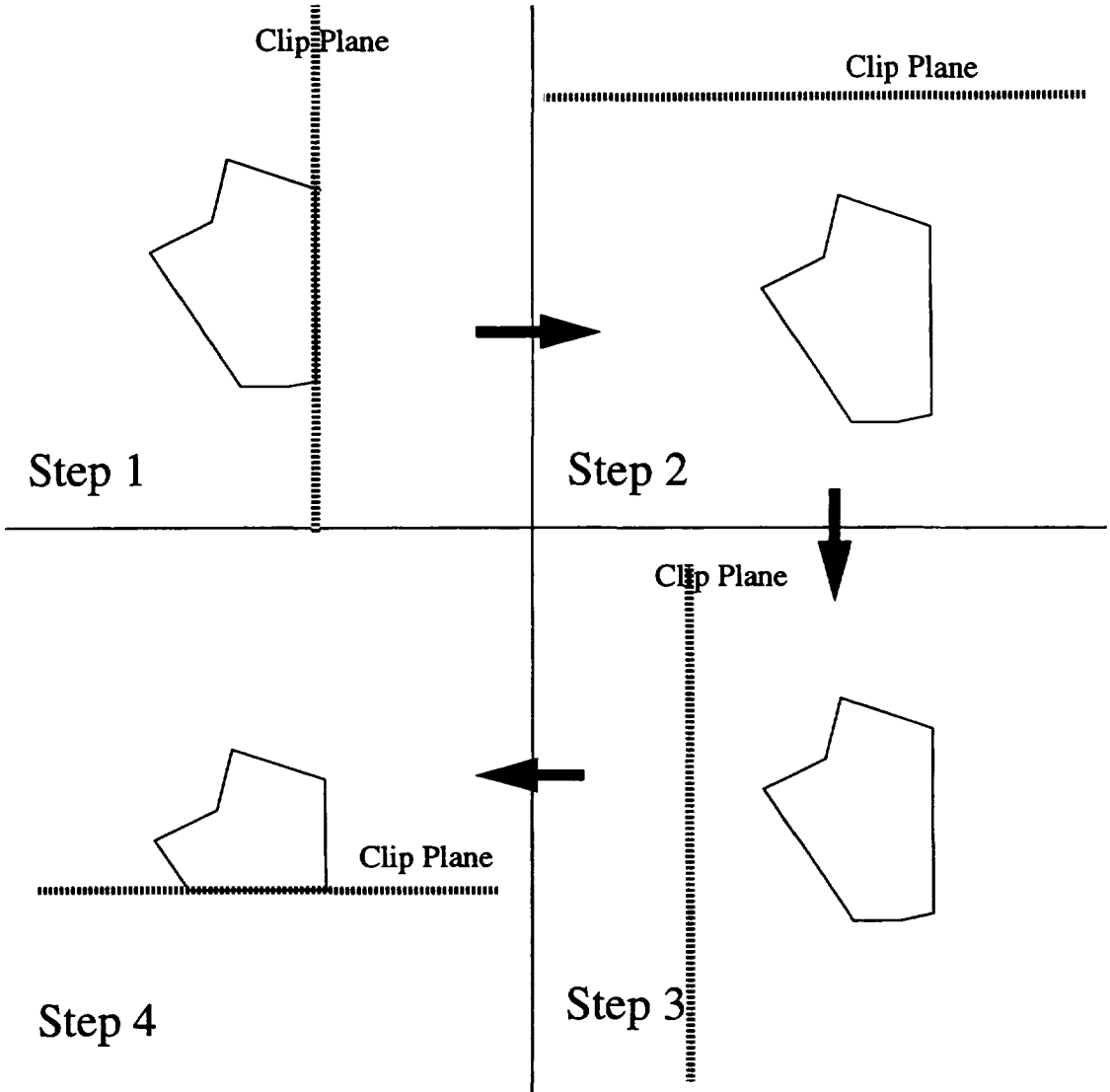


Figure A.7 Application of Sutherland-Hodgman Clipping to Polygon Shown in Figure A.1

As seen here, the clipping operation returns the Boolean and of the clipper and clipped polygon, but this is not always what is desired from the cell cutting operation. Consider the Cartesian clipping polygon, C, the Subject polygon S, and the result of the polygon clipping, polygon P_1. Often, what is desired is P_2, where $C = P_1 + P_2$ as is shown in Figure A.8. As an example, if S were the surface of an airfoil, and P_1 is interior to the airfoil, the computational cell needed would actually be P_2. This entails determining whether the clipped polygon lies within or exterior to the computational domain, and if it

lies within, recovering the polygon $P_2 = C - P_1$.

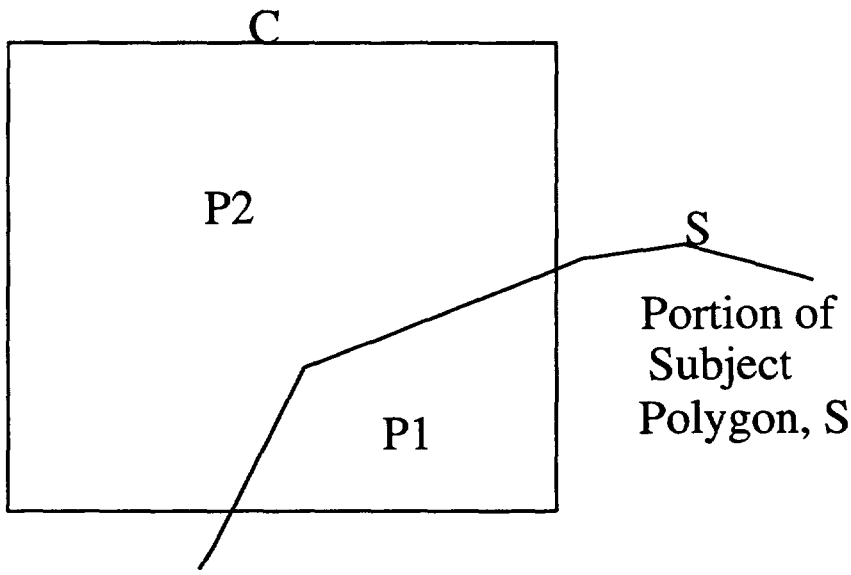


Figure A.8 Relationships Between Clipping, Subject and Clipped Polygons

For body surfaces that are described by functions other than linear basis functions, the clipping algorithm works much the same, but the intersections of the bodies with the clipping, Cartesian cells, is found using a root finding procedure.

The grid generation procedure is applied in a recursive manner using the binary tree data structure by recurring down the tree, staying within a sub-branch, until all leaves below the sub-branch have either been cut, are determined to not need to be cut or have been blanked. If a leaf cell is deemed inadequate for cutting, by either the cell size, boundary face size or number of intersections with the body, a new sub-branch is created below it, and the grid generation procedure recurs down through the new sub-branch. After all cells have been cut, the mesh is examined for smoothness across refinement boundaries, and is recursively smoothed by refining where the difference between refinement levels is greater than one. Since the grid generation procedure is applied locally by recursion, it is efficient and can generate base grids very quickly. Importantly, this procedure lends itself well to a parallel implementation, since the grid generation below a given, coarse tree, will

be contained locally within each processor, and the only interprocessor communication will be after the base grid is generated, and is checked for refinement boundary smoothness.

To further demonstrate the capability of the Cartesian grid generator using the polygon clipping, a base grid in a flow passage representative of the cooling passage within a turbine blade is shown. The geometry corresponds to that in [73], and is a projection onto the xy plane of the geometry in the curved surface that is located along the turbine blade mean chord line. The grid contains 2640 cells and was generated in 218 seconds on an IBM RS6000 Model 560 workstation. The geometry definition is made using a linearly-represented, continuous outer boundary and 14 cooling fins.

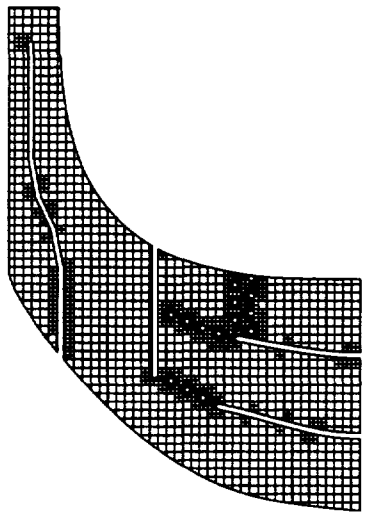


Figure A.9 Turbine Coolant Passages: Base Grid

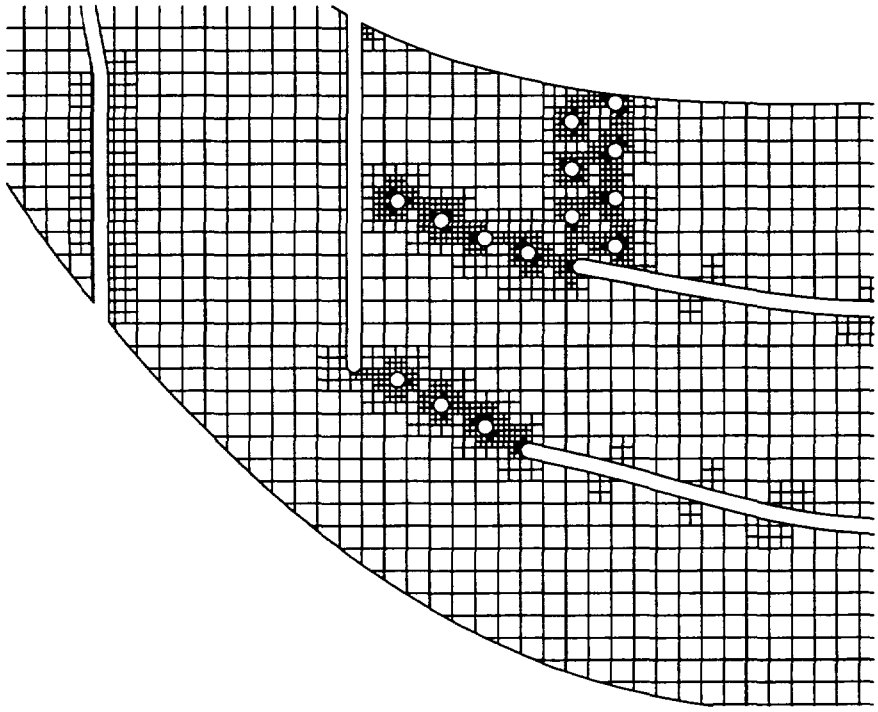


Figure A.10 Close-up of Pin Region

Appendix B
A Discrete Accuracy Analysis of two Cell-centered Viscous Flux Formulae

If care is taken in the formulation, the discrete cell-centered, finite-volume formulation of the Laplacian upon arbitrary meshes can be obtained given the gradient reconstruction procedure. The result of this is, for each cell, the discrete weights of all the support cells used in the Laplacian; the α_n in (3.25). From these weights, the positivity and accuracy of the stencil for each cell can be found by examining the sums in (3.29) and the weights in (3.44). The primary drawback of this is that the resulting formulae do not readily yield useful information, unless they are evaluated on given meshes so that different reconstructions can be directly compared. So, the following shows the formulae obtained and how to evaluate them on arbitrary meshes for the two candidate schemes. These are then used to explain the behavior of the schemes when they are used to compute some low and moderate Reynolds number flows.

B.3 General Laplacian: Diamond Path Reconstruction Using the Linearity-Preserving Weighting

Applying the diamond-path type reconstruction procedure on an arbitrary N-sided polygonal control volume, and expanding the resulting formula including all the weights used to provide data at the subtended vertices of the face, the following general formula for the Laplacian is obtained

$$L(u_0) = \alpha_0 u_0 + \sum_{f=1}^{F} \beta_f u_f + \sum_{v=1}^{V} \gamma_v u_v \tag{B.1}$$

The stencil contains all the first order neighbors of the cell, which are delineated by being

either a face or vertex neighbor. The contributions from the face neighbors are from the first summation for the F face neighbors, while the second summation is over all of the V vertex neighbors. The coefficients in (B.1) contain important information about the geometry of the grid, cell faces and cell centroids, as well as the weights of the cells used to find the data at the vertices subtending the faces. The coefficients are

$$\alpha_0 = \sum_{f=1}^{F} \left[\frac{N_{0,f}}{2AA_f} + \frac{\omega_{0,f}}{A} \left(\frac{N_{B,f}}{2A_f} + \frac{N_{T,f-1}}{2A_{f-1}} \right) \right] \tag{B.2}$$

$$\beta_f = \frac{N_{R,f}}{2AA_f} + \sum_{n=1}^{F} \omega_{f,n} \left(\frac{N_{B,n}}{2AA_n} + \frac{N_{T,n-1}}{2AA_{n-1}} \right) \tag{B.3}$$

$$\gamma_v = \sum_{f=1}^{F} \omega_{v,f} \left(\frac{N_{B,f}}{2AA_f} + \frac{N_{T,f-1}}{2AA_{f-1}} \right) \tag{B.4}$$

The $\omega_{i,j}$ are the weights of the i-th cell used in the cell weighting of the j-th vertex. For a simple averaging procedure, these weights are the inverse of the number of cells contributing to the average, while if a linearity-preserving weighting is used, the weights are found from the more complicated procedure, (3.61) to (3.72). If the faces and vertices are ordered in a positive (counter-clockwise) direction, then the vertex and face orderings are meaningful, where the j-th face is subtended by the j-th and j+1-th vertices. The cell area is A, and the area of the co-volume about the f-th face is A_f. The N_j are related to the cell geometry and are the dot products of face normals and the normals to the vector that joins the two centroids of the cells that share a face.

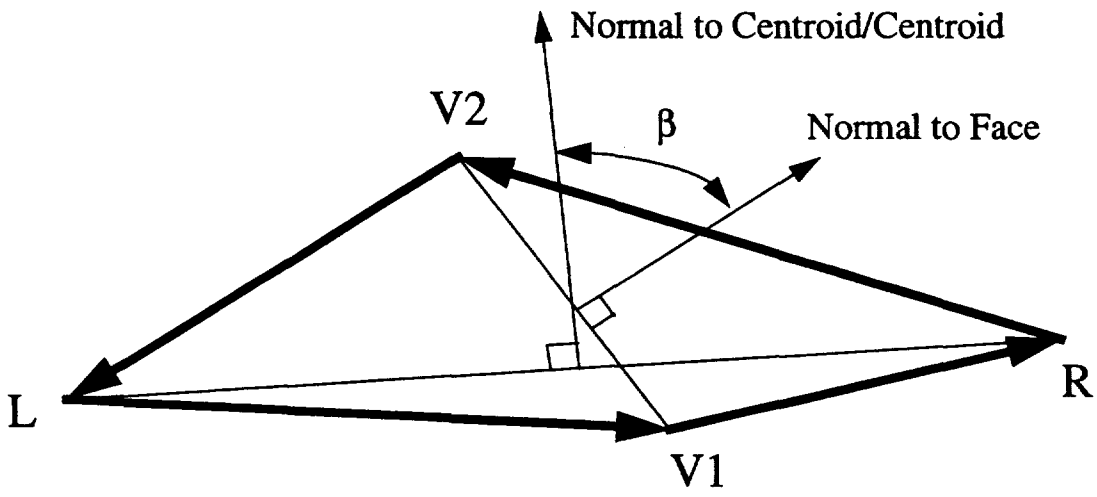


Figure B.1 Important Geometric Terms in Diamond Path Reconstruction

After some manipulation, the coefficients in (B.1) are found to be

$$\alpha_0 = \sum_{f=1}^{F}\left[-\frac{(\Delta S_f)^2}{2AA_f} + \frac{\omega_{0,f}}{A}(\cot\beta_{f-1} - \cot\beta_f)\right] \tag{B.5}$$

$$\beta_f = \frac{(\Delta S_f)^2}{2AA_f} + \sum_{n=1}^{F}\frac{\omega_{f,n}}{A}(\cot\beta_{n-1} - \cot\beta_n) \tag{B.6}$$

$$\gamma_v = \sum_{f=1}^{F}\frac{\omega_{v,f}}{A}(\cot\beta_{f-1} - \cot\beta_f) \tag{B.7}$$

where the ΔS_f is the length of the f-th face. As can be seen, a positive scheme can be guaranteed if the following three conditions are all met

$$\sum_{f=1}^{F}\omega_{0.,f}(\cot\beta_{f-1} - \cot\beta_f) \le \sum_{f=1}^{F}\frac{(\Delta S_f)^2}{2A_f} \tag{B.8}$$

$$\sum_{n=1}^{F}\omega_{f,n}(\cot\beta_n - \cot\beta_{n-1}) \le \frac{(\Delta S_f)^2}{2A_f} \tag{B.9}$$

$$\sum_{n=1}^{F} \omega_{v,n} (\cot\beta_{n-1} - \cot\beta_n) \geq 0 \qquad \text{(B.10)}$$

(B.8) guarantees that the weight of the object cell is negative, while (B.9) ensures the weights of the face neighbors are positive. (B.10) guarantees positivity of the vertex neighbors contributions to the Laplacian.

A positive scheme can be guaranteed by a few different ways. If the cotangents are all equal, the two cotangents in each sequence of the summations will cancel. This can be true if the mesh is orthogonal, in that each face is perpendicular to the vector joining the centroids that share the face, so that the cotangent is everywhere zero. Example meshes would be a triangular mesh of all equilateral triangles or a quadrilateral mesh that is either unstretched, or stretched along the coordinate axes only. In [7] it is shown that in two-dimensions, a Delaunay mesh guarantees positivity when using a linear Galerkin, finite element formulation, although for the reconstruction scheme here, a Delaunay mesh does not guarantee it. A deeper analysis of the positivity criteria for the scheme here, upon a general, triangular mesh is called for; on a general mesh, the strict inclusion of the weights will undoubtedly greatly increase the complexity of the analysis.

For simpler meshes, a few points can be made, though. For the unique type of mesh formed by equilateral triangles or by unstretched quadrilaterals, it can seen that the Laplacian can be interpreted as a simple average of the surrounding cells. This simple average results, on a uniform Cartesian mesh, with the desirable (-4,1,1,1,1) weighting of the cell and its four face neighbors. Since in general, one can not guarantee that the cotangents are equal, a more general way would be if the weights in the sums are all equal. Then, the summation of the difference of the adjacent cotangents would vanish. But, this is very restrictive, since many of the $\omega_{f,n}$ are already zero (not every neighbor of the cell contributes to the vertex weighting of a particular face where the flux is found), and would require that all the weights are zero. Then again, the simple face weighted Laplacian

would result. On general meshes, this can result in a very inaccurate scheme, but can ensure positivity. Little can be said a priori as far as sensitivity of the scheme to non-smoothness of the mesh is concerned. Only experience with the scheme on various meshes can give some insight as to how it will behave.

B.3 General Laplacian: $K_v = 2$ Reconstruction

The general Laplacian using the $K_v = 2$ is a bit more complicated than the diamond path scheme. The reconstructed gradient involves the inverse of a general thirty-six element Vandermonde type matrix, of which only two rows of the inverse are actually needed. For a given face, consider the contribution of the j-th support cell to the reconstructed gradient at a given face dotted with the face normal, $D(u)$,

$$\frac{\partial D(u) \bullet \vec{\mathbf{n}}}{\partial u_j} = \frac{(A_{1j}^{-1} n_x + A_{2j}^{-1} n_y)}{\delta} \tag{B.11}$$

The matrix A is formed in the face centered coordinate system that is scaled by the local length scale, δ. The contribution of each cell to the assembled, discrete Laplacian is then found by summing the contribution from each component face since

$$\alpha_j = \frac{1}{Area} \sum_{faces} \frac{\partial D(u) \bullet \vec{\mathbf{n}}}{\partial u_j} \tag{B.12}$$

From this, the assembled Laplacian can be analyzed as before on different grids. Since much of the inherent behavior of the Laplacian is hidden in a rather complicated manner by the matrix A and its inverse, it is not obvious how different topologically similar grids will behave, as opposed to, say, triangular grids, where things can be said about meshes that satisfy a Delaunay-like adjacency. Although this may be true, insight into particular grids as far as accuracy and positivity can be found by locally examining the Laplacian

stencil coefficients on general meshes created by the Cartesian-mesh grid generator.

As is shown in the preceding analysis of the positivity of this scheme upon the model grids, there is no immediately obvious connection between support-set selection and stencil positivity. Some obvious choices of support sets are not invertible, and some invertible sets give positive stencils, while some do not. The positivity constraints come about only after the stencil is assembled in each cell, which is greatly complicated by the fact that the gradients computed at each face contributes to the stencil of the two cells that share the face. From this, it can be seen that the positivity constraint is actually an implicit inequality constraint that couples all cells together through the conservation law formulation.

In practice, a carefully organized procedure is needed to find the support sets about each face. The procedure used to find the support set to perform the reconstruction at each face is based upon using the directed face neighbors to the face. If there is not enough data to invert the system, the next available cells in a prioritized list are chosen to close the system. This prioritized list is formed by ordering the extra cells in increasing distance from the face midpoint, in the spirit of the stencil selection criterion presented in [55]. The matrix is then assembled and examined to see if it is singular or ill-conditioned. Since an ill-conditioned matrix could indicate a poor stencil, this criterion is more stringent than just checking for singularity, and should yield better stencils. In this case, the stencil is considered to represent an ill-conditioned stencil if the determinant is less than some cut-off value, taken here to be 1×10^{-5} . If the chosen support set is found to yield a singular or ill-conditioned stencil, the last added cell to the support list is deleted, and the next available cell in the list of extra cells is added in its place, and a new matrix assembled and examined. This process recurs until a well-conditioned matrix is found or the cells in the auxiliary support list are exhausted. If this is the case, the procedure is re-initialized, but now a linear reconstruction is recursively searched for. In this way, the stencil search pro-

cess will always yield a reconstruction that in the worst case is a linear reconstruction. This procedure typically yields grids where over 98 percent of the interior faces use a quadratic reconstruction.

BIBLIOGRAPHY

[1] M. Aftosmis, D. Gaitonde, and T. Sean Tavares. On the Accuracy, Stability and Monotonicity of Various Reconstruction Algorithms for Unstructured Meshes. AIAA Paper AIAA-94-0415, 1994.

[2] AGARD Subcommittee C. Test Cases for Inviscid Flow Field Methods. AGARD Advisory Report 211, 1986.

[3] B.F. Armaly, F. Durst, J.C.F. Pereira, and B. Schonung. Experimental and Theoretical Investigation of Backward-Facing Step Flow. *Journal of Fluid Mechanics*, 127:473–496, 1983.

[4] T. J. Barth and P. O. Frederickson. Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction. AIAA Paper 90-0013, 1990.

[5] T.J. Barth. On Unstructured Grids and Solvers. In *Computational Fluid Dynamics*. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[6] T.J. Barth. A 3-D Upwind Euler Solver for Unstructured Meshes. AIAA 10th Computational Fluid Dynamics Conference Procedings, 1991.

[7] T.J. Barth. Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes. AIAA Paper AIAA-91-0721, 1991.

[8] T.J. Barth and D.C. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA Paper 89-0366, 1989.

[9] S.A. Bayyuk, K.G. Powell, and B. van Leer. An Algorithm For the Simulation of 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry. AIAA 11th Computational Fluid Dynamics Conference Procedings, 1993.

[10] L. Bergamini and P. Cinnella. Using the Liou-Steffen Algorithm for the Euler and Navier-Stokes Equations. *AIAA Journal*, 32(3):657–658, 1993.

[11] M. J. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

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

[13] M.J. Berger and R.J. LeVeque. A Rotated Difference Scheme for Cartesian Grids in Complex Geometries. *Computing Systems in Engineering*, 1:305–311, 1990.

[14] M.J. Berger and R.J. LeVeque. Stable Boundary Conditions for Cartesian Grid Calculations. *Computing Systems in Engineering*, 1(2-4):305–311, 1990.

[15] M.J. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484–512, 1984.

[16] Y.L. Chiang. *Simulation of Unsteady Inviscid Flow On An Adaptively Refined Cartesian Grid*. PhD thesis, The University of Michigan, Department of Aerospace

Engineering, 1992.

[17] R.V. Chima. Viscous Three-Dimensional Calculations of Transonic Fan Performance. NASA TM 103800.

[18] Y. Choi and C.L. Merkle. Time-Derivative Preconditioning for Viscous Flows. AIAA paper AIAA-91-1652, 1991.

[19] D. K. Clarke, M. D. Salas, and H. A. Hassan. Euler Calculations for Mutielement Airfoils Using Cartesian Grids. *AIAA Journal*, 24, 1986.

[20] W.J. Coirier and K.G. Powell. An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations. AIAA Paper 93-3335-CP, 1993.

[21] W.J. Coirier and B. van Leer. Numerical Flux Formulas for the Euler and Navier-Stokes Equations: II. Progress in Flux-Vector Splitting. In AIAA 10th Computational Fluid Dynamics Proceedings, AIAA paper 91-1566-CP, 1991.

[22] D. DeZeeuw and K.G. Powell. Euler Calculations of Axisymmetric Under-Expanded Jets by an Adaptive-Refinement Method. AIAA Paper 92-0321, 1992.

[23] D. DeZeeuw and K.G. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *Journal of Computational Physics*, 104(1):56–58, 1993.

[24] D.L. DeZeeuw. *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*. PhD thesis, The University of Michigan, Department of Aerospace Engineering, 1993.

[25] B. Epstein, A. Luntz, and A. Nachshon. Multigrid Euler Solver about Arbitrary Aircraft Configurations with Cartesian Grids and Local Refinement. AIAA-89-1960-CP, 1989.

[26] J.A. Essers and E. Renard. An Implicit Flux-Vector Splitting Finite-Element Technique for and Improved Solution of Compressible Euler Equations on Distorted Grids. In *11-th International Conference on Numerical Methods in Fluid Dynamics: Lecture Notes in Mathematics*, number 323. Springer Verlag, 1988.

[27] G. Grasso and M. Marini and M Passalacqua. Viscous High-Speed Flow Computations by Adaptive Mesh Embedding Techniques. *AIAA Journal*, 30(7):1780–1788, 1992.

[28] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[29] A.G. Godfrey, C.R. Mitchell, and R.W. Walters. Practical Aspects of Spatially High Accurate Methods. AIAA Paper AIAA-92-0054, 1992.

[30] C. Gooch and H. Oksuzoglu. Extension of State-Vector Splitting to the Navier-Stokes Equations. AIAA Paper 93-3374-CP, 1993.

[31] C.F. Gooch. *Solution of the Navier-Stokes Equations on Locally-Refined Cartesian Meshes using State-Vector Splitting*. PhD thesis, Stanford University, 1993.

[32] G. Grasso, M. Marini, and M Passalacqua. LU Implicit TVD Scheme for the Solution

of Viscous Two-Dimensional High Speed Flows. AIAA Paper AIAA-91-1573-CP, 1991.

[33] D.W. Halt and R.W. Agarwal. Compact Higher Order Characteristic-Based Euler Solver for Unstructured Grids. *AIAA Journal*, 30(8):1993–1999, 1992.

[34] W.R. Hingst and D.F. Johnson. Experimental Investigation of BOundary Layers in an Axi-Symmetric, Mach 2.5 Inlet. NASA TM X-2902.

[35] D.G. Holmes and S.D. Connell. Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids. AIAA Paper 89-1932-CP, 1989.

[36] C.J. Hwang and J.L. Liu. Locally Implicit Hybrid ALgorithm for Steady and Unsteady Viscous Flows. *AIAA Journal*, 30(5):1228–1236, 1992.

[37] J.M. Hyman, R.J. Knapp, and J.C. Scovel. High Order Finite Volume Approximations of Differential Operators on Nonuniform Grids. *Physica D, North-Holland Press*, 60:112–138, 1992.

[38] P.A. Jacobs. Single-Block Navier-Stokes Integrator. ICASE Contractor Report 187613, 1991.

[39] P.C.E. Jorgenson. *An Implicit Numerical Scheme for the Simulation of Internal Viscous Flows on Unstructured Grids*. PhD thesis, Iowa State University, Department of Mechanical Engineering, 1993.

[40] P.C.E. Jorgenson and R.H. Pletcher. An Implicit Numerical Scheme for the Simulation of Internal Viscous Flows on Unstructured Grids. AIAA paper AIAA-94-0306, 1994.

[41] D. Knight. A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting. AIAA Paper 93-0875, 1993.

[42] P. D. Lax. Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves. *Society for Indistrial and Applied Mathematics.*

[43] P. D. Lax. Weak Solutions of Nonlinear Hyperbolic Equations and Their Numerical Computation. *Communications on Pure and Applied Mathematics*, VII, 1954.

[44] P. D. Lax. Systems of Conservation Laws. *Communications on Pure and Applied Mathematics*, XIII, 1960.

[45] D. Lee and Bram van Leer. Progress in Local Preconditioning of the Euler and Navier-Stokes Equations. AIAA paper AIAA-93-3328-CP, in the AIAA 11th CFD Conference, 1993.

[46] W.T. Lee. *Local Preconditioning of the Euler Equations*. PhD thesis, The University of Michigan, 1991.

[47] M. S. Liou, B. van Leer, and J. S. Shuen. Splitting of Inviscid Fluxes for Real Gases. *Journal of Computational Physics*, 87, 1990.

[48] M.S. Liou and C.J.Steffen, Jr. A New Flux Splitting Scheme. *Journal of Computational Physics*, 107:23–39, 1993.

[49] M.-S. Liou. A Continuing Search for a Near-Perfect Numerical Flux Scheme: Part I:

AUSM+, 1994.

[50] M.-S. Liou and A.T. Hsu. A Time Accurate Finite Volume High Resolution Scheme for Three-Dimensional Navier-Stokes Equations. AIAA Paper AIAA-89-1994-CP, 1989.

[51] D.J. Mavripilis and A. Jameson. Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes. *AIAA Journal*, 28(8):1415–1425, 1990.

[52] J.E. Melton, M.J. Berger, M.A. Aftosmis, and M.D. Wong. 3D Applications of a Cartesian Grid Euler Method. proposed for publication at the 33rd Aerospace Sciences Meeting, 1995.

[53] J.E. Melton, F.Y. Enomoto, and M.J. Berger. 3D Automatic Cartesian Grid Generation for Euler Flows. AIAA Paper AIAA-93-3386-CP, in AIAA 11th Computational Fluid Dynamics Conference Proceedings, 1993.

[54] J.E. Melton, S.A. Pandya, and J.L Steger. 3D Euler Flow Solutions using Unstructured Cartesian and Prismatic Grids. AIAA paper AIAA-93-0331, 1993.

[55] C.R. Mitchell and R.W. Walters. K-Exact Reconstruction for the Navier-Stokes Equations on Arbitrary Grids. AIAA Paper 93-0536, 1993.

[56] K. Morinishi. A Finite Difference Solution of the Euler Equations on Non-Body-Fitted Cartesian Grids. *Computers and Fluids*, 21(3):331–334, 1992.

[57] K. Nakahashi. Optimum Spacing Control of the Marching Grid Generation. AIAA paper AIAA-91-21368, 1991.

[58] H. Oksuzoglu. *State Vector Splitting: A Numerical Scheme for the Euler Equations.* PhD thesis, Stanford University, 1992.

[59] H. Oksuzoglu. State Vector Splitting for the Euler Equations of Gasdynamics. AIAA paper AIAA-92-0326, 1992.

[60] R. Pember, J. Bell, P. Colella, W. Crutchfield, and M. Welcome. Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow. AIAA Paper 93-3385-CP, 1993.

[61] K.G. Powell, P.L. Roe, and J. Quirk. Adaptive-Mesh Algorithms for Computational Fluid Dynamics. ICASE Report, 1992.

[62] J.W. Purvis and J.E. Burkhalter. Prediction of Critical Mach Number for Store Configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[63] J.J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* PhD thesis, Cranfield Institute of Technology, College of Aeronautics, 1991.

[64] J.J. Quirk. An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two-Dimensional Bodies. ICASE Report 92-7, 1992.

[65] R. Radaspiel and R.C. Swanson. An Investigation of Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations. AIAA Paper AIAA-89-0548, 1989.

[66] R.D. Rausch, J.T. Batina, and H.T.Y. Yang. Spatial Adaption Procedures on

Unstructured Meshes for Accurate Unsteady Aerodynamic Flow Computation. NASA TM 104039, 1991.

[67] E. Renard and J.A. Essers. An Analysis of Severe Grid Distortion Effects on the Accuracy of Some Discretization Schemes for Convection-Diffusion Equations. In *Numerical Grid Generation in Computational Fluid Dynamics*, 1988.

[68] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *Journal of Computational Physics*, 43, 1981.

[69] C. L. Rumsey, B. van Leer, and P. L. Roe. Effect of a Multi-Dimensional Flux Function on the Monotonicity of Euler and Navier-Stokes Computations. AIAA 91-1530, 1991.

[70] L.M. Russell, D.R. Thurman, P.S. Simonyi, S.A. Hippensteele, and P.E. Poinsatte. Measurements and Computational Analysis of Heat Transfer and Flow in a Simulated Turbine Blade Internal Cooling Passage. AIAA Paper AIAA-93-1797, 1993.

[71] H. Schlichting. Boundary-layer theory, 1979. McGraw-Hill, Inc., Seventh Edition.

[72] J.-S. Shuen. Upwind Differencing and LU Factorization for Chemical Non-Equilibrium Navier-Stokes Equations. *Journal of Computational Physics*, 99(2):233–250, 1992.

[73] P.H. Snynder and R.J.Roelke. The Design of an Air Cooled Metallic High Temperature Radial Turbine. AIAA Paper 88-2872, 1988.

[74] P. Spellucci. DONLP: Do Nonlinear Programming, 1993. Obtained via Xnetlib server.

[75] C.J. Steffen, Jr. Private Communication, 1994.

[76] I.E. Sutherland and G.W. Hodgman. Reentrant Polygon Clipping. *Communications of the ACM, Graphics and Image Processing*, 17(1):32–42, 1974.

[77] J. F. Thompson. Numerical Grid Generation. Elsevier Science Publishing Company, Inc., 1982.

[78] E. Turkel. Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations. ICASE Report 86-14, 1986.

[79] B. van Leer. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32, 1979.

[80] B. van Leer. Flux-vector Splitting for the Euler Equations. *Lecture Notes in Physics*, 170, 1982.

[81] B. van Leer, W. T. Lee, and P. L. Roe. Characteristic Time-Stepping or Local Preconditioning for the Euler Equations. in AIAA 10th Computational Fluid Dynamics Conference Proceedings, 1991.

[82] B. van Leer, C. H. Tai, and K. G. Powell. Design of Optimally-Smoothing Multi-Stage Schemes for the Euler Equations. In *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[83] B. van Leer, J.L. Thomas, P.L. Roe, and R.W. Newsome. A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations. In AIAA 8th Computational Fluid Dynamics Proceedings, 1987.

[84] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. AIAA Paper AIAA-93-0880, 1993.

[85] S. Venkateswaran, J.M. Weiss, C.L. Merkle, and Y.H. Choi. Preconditioning and Time-Step Definition in Reacting Navier-Stokes Computations. *Journal of Computational Physics*, 1993.

[86] S. Ward and Y. Kallinderis. Hybrid Prismatic/Tetrahedral Grid Generation for Complex 3-D Geometries. AIAA paper AIAA-93-0669, 1993.

[87] G. Warren, W. K. Anderson, J. Thomas, and S. Krist. Grid Convergence for Adaptive Methods. AIAA 10th Computational Fluid Dynamics Conference Procedings, 1991.

[88] D.F. Watson. Contouring: A Guide to the Analysis and Display of Spatial Data. Pergamon Press, 1992.

[89] N.P. Weatherill. Numerical Grid Generation. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-06, 1990.

[90] B. Wedan and J.C. South Jr. A Method for Solving the Transonic Full-Potential Equation for General Configurations. AIAA paper AIAA-83-1869, 1983.

[91] D. P. Young, R. G. Melvin, M. B. Bieterman, and J. E. Bussoletti. A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computatational Physics. *Journal of Computational Physics*, 92:1–66, 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1994	Technical Memorandum

4. TITLE AND SUBTITLE

An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations

5. FUNDING NUMBERS

WU-505-62-52

6. AUTHOR(S)

William John Coirier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-9174

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-106754

11. SUPPLEMENTARY NOTES

This report was submitted as a dissertation in partial fulfillment of the requirements for the degree Doctor of Philosophy to The University of Michigan, Ann Arbor, Michigan, 1994. Responsible person, William John Coirier, organization code 2610, (216) 433-5764.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A Cartesian, cell-based scheme for solving the Euler and Navier-Stokes equations in two dimensions is developed and tested. Grids about geometrically complicated bodies are generated automatically, by recursive subdivision of a single Cartesian cell encompassing the entire flow domain. Where the resulting cells intersect bodies, polygonal "cut" cells are created. The geometry of the cut cells is computed using polygon-clipping algorithms. The grid is stored in a binary-tree data structure which provides a natural means of obtaining cell-to-cell connectivity and of carrying out solution-adaptive refinement. The Euler and Navier-Stokes equations are solved on the resulting grids using a finite-volume formulation. The convective terms are upwinded, with a limited linear reconstruction of the primitive variables used to provide input states to an approximate Riemann solver for computing the fluxes between neighboring cells. A multi-stage time-stepping scheme is used to reach a steady-state solution. Validation of the Euler solver with benchmark numerical and exact solutions is presented. An assessment of the accuracy of the approach is made by uniform and adaptive grid refinements for a steady, transonic, exact solution to the Euler equations. The error of the approach is directly compared to a structured solver formulation. A non-smooth flow is also assessed for grid convergence, comparing uniform and adaptively refined results. Several formulations of the viscous terms are assessed analytically, both for accuracy and positivity. The two best formulations are used to compute adaptively refined solutions of the Navier-Stokes equations. These solutions are compared to each other, to experimental results and/or theory for a series of low and moderate Reynolds numbers flow fields. The most suitable viscous discretization is demonstrated for geometrically-complicated internal flows. For flows at high Reynolds numbers, both an altered grid-generation procedure and a different formulation of the viscous terms are shown to be necessary. A hybrid Cartesian/body-fitted grid generation approach is demonstrated. In addition, a grid-generation procedure based on body-aligned cell cutting coupled with a viscous stencil-construction procedure based on quadratic programming is presented.

14. SUBJECT TERMS

Cartesian-based; Adaptive mesh refinement; Euler; Navier-Stokes; Dissertation

15. NUMBER OF PAGES

249

16. PRICE CODE

A11

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	