INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
An adaptively-refined quadtree grid method for incompressible flows

by

Stuart Scott Ochs

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Major Professor: R Ganesh Rajagopalan

Iowa State University
Ames, Iowa
1998

Copyright © Stuart Scott Ochs. 1998. All rights reserved.
This is to certify that the Doctoral dissertation of
Stuart Scott Ochs
has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Signature was redacted for privacy.

Major Professor
Signature was redacted for privacy.
For the Major Program
Signature was redacted for privacy.
For the Graduate College
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>THE QUADTREE GRID</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>GRID ADAPTION</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>UNI-DIMENSIONAL POWER-LAW METHOD</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Theoretical Formulation</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Governing Equations</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Discretization of the Scalar Transport Equation</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Discretized Momentum Equations</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Velocity-Pressure Coupling</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Pressure-Correction Equation</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Pressure Equation</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Boundary Conditions</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Solution Procedure</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Results</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Lid-driven Cavity Flow</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Duct Flow with a Sudden Contraction</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Channel Flow with Two Right-Angle Bends</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Flow over a Circular Cylinder</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>UPWIND METHOD</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Theoretical Formulation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Governing Equations</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Discretization of the Scalar Transport Equation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Discretized Momentum Equations</td>
<td>43</td>
</tr>
</tbody>
</table>
### Velocity-Pressure Coupling

*Page 44*

### Pressure-Correction Equation

*Page 44*

### Boundary Conditions

*Page 45*

### Solution Procedure

*Page 47*

### Results

*Page 48*

- Lid-driven Cavity Flow .................................................. 18
- Backward-Facing Step Flow ............................................... 50
- Flow over a Circular Cylinder ......................................... 57
- Hybrid Grid .................................................................... 59

#### 6 THREE-DIMENSIONAL FLOWS

*Page 63*

- The Octree Grid ............................................................... 63
- Flow Solution Method .......................................................... 63
- Results .................................................................. 65
  - 3-D Driven Cavity Flow ................................................. 65
  - 90 Degree Curved Duct Flow ............................................ 68

#### 7 CONCLUSIONS AND RECOMMENDATIONS

*Page 75*

### BIBLIOGRAPHY

*Page 79*
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Quadtree cell division</td>
</tr>
<tr>
<td>2.1</td>
<td>Quadtree grid</td>
</tr>
<tr>
<td>2.2</td>
<td>Quadtree data structure</td>
</tr>
<tr>
<td>4.1</td>
<td>A typical quadtree grid section</td>
</tr>
<tr>
<td>4.2</td>
<td>Quadtree grid boundary control volume</td>
</tr>
<tr>
<td>4.3</td>
<td>Driven cavity. Re = 100: quadtree grid (4,288 cells)</td>
</tr>
<tr>
<td>4.4</td>
<td>Driven cavity. Re = 100: u velocity profiles. x = 1/2</td>
</tr>
<tr>
<td>4.5</td>
<td>Driven cavity. Re = 100: u velocity profiles. x = 1/32</td>
</tr>
<tr>
<td>4.6</td>
<td>Driven cavity. Re = 100: u velocity profiles. x = 1/64</td>
</tr>
<tr>
<td>4.7</td>
<td>Driven cavity. Re = 100: total velocity RMS values</td>
</tr>
<tr>
<td>4.8</td>
<td>Driven cavity. Re = 400: quadtree grid (9,472 cells)</td>
</tr>
<tr>
<td>4.9</td>
<td>Driven cavity. Re = 400: u velocity profiles. x = 1/2</td>
</tr>
<tr>
<td>4.10</td>
<td>Driven cavity. Re = 400: u velocity profiles. x = 1/64</td>
</tr>
<tr>
<td>4.11</td>
<td>Driven cavity. Re = 400: total velocity RMS values</td>
</tr>
<tr>
<td>4.12</td>
<td>Duct with sudden contraction: four-level quadtree grid</td>
</tr>
<tr>
<td>4.13</td>
<td>Duct with sudden contraction: u velocity profiles. x = -2.5mm</td>
</tr>
<tr>
<td>4.14</td>
<td>Duct with sudden contraction: u velocity profiles. x = -1.0mm</td>
</tr>
<tr>
<td>4.15</td>
<td>Duct with sudden contraction: u velocity profiles. x = 1.0mm</td>
</tr>
<tr>
<td>4.16</td>
<td>Z channel: schematic diagram</td>
</tr>
<tr>
<td>4.17</td>
<td>Z channel: initial grid</td>
</tr>
<tr>
<td>4.18</td>
<td>Z channel: final grid</td>
</tr>
<tr>
<td>4.19</td>
<td>Z channel: streamline contours</td>
</tr>
<tr>
<td>4.20</td>
<td>Z channel: normalized velocity profiles</td>
</tr>
<tr>
<td>4.21</td>
<td>Cut cells</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6.4</td>
<td>3-D driven cavity, grid 1: velocity vectors. a) z = 0.5 plane. b) x = 0.5 plane.</td>
</tr>
<tr>
<td>6.5</td>
<td>3-D driven cavity, grid 2: velocity vectors. a) z = 0.5 plane. b) x = 0.5 plane.</td>
</tr>
<tr>
<td>6.6</td>
<td>3-D driven cavity: u velocity profiles. x = 0.5, z = 0.5.</td>
</tr>
<tr>
<td>6.7</td>
<td>90 degree curved duct: schematic diagram.</td>
</tr>
<tr>
<td>6.8</td>
<td>90 degree curved duct: adapted grid.</td>
</tr>
<tr>
<td>6.9</td>
<td>90 degree curved duct: velocity vectors. z = 20mm plane.</td>
</tr>
<tr>
<td>6.10</td>
<td>90 degree curved duct: velocity vectors. y = 112mm plane.</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

The author would like to express his deepest gratitude to his major professor, Dr. R. Ganesh Rajagopalan, for his invaluable guidance and support throughout the author's graduate studies. The author also extends his appreciation to the members of his Program of Study Committee, Dr. John Gustafson, Dr. T.J. McDaniel, Dr. Bruce Munson, Dr. Richard Pletcher, and Dr. Jerry Vogel.

Thanks are also due to the faculty and graduate students of the Department of Aerospace Engineering and Engineering Mechanics, and in particular, Dr. Richard Hindman, Dr. Laith Zori, Dr. Sutikno Wirogo, Ashvin Mahajan, Dr. James Miller, and Dr. Bijoyendra Nath, for their theoretical discussions and technical advice; to the tireless secretaries of the department, Sue Ferguson, Laurie Hoifeldt, Colleen Johnson, and Dee Pfeiffer; to Jim Wellman, department computer systems support specialist; and to Joey Bramhall, for his technical editing and flawless proofreading.

The author also wishes to acknowledge the partial financial support for this research provided by Sandia National Laboratories under Sandia National Grant number AF-8861.
ABSTRACT

This study presents an adaptively-refined quadtree grid method used in conjunction with a pressure-based flow solution algorithm for the incompressible Navier-Stokes equations. The quadtree grid, which is composed of quadrilateral cells that can be successively subdivided into four quadrants, is examined, and the quadtree data structure, and its advantages when used in the numerical solution of the Navier-Stokes equations, is discussed. Several strategies for solution adaptive grid refinement, which is used to resolve high-gradient flow regions, are then presented.

Two different flow solution methods, the uni-dimensional power-law method and the upwind method, both based on a cell-centered, finite-volume technique, are studied. These methods solve the governing equations for the primitive variables on a collocated grid using the SIMPLE algorithm. The uni-dimensional power-law method approximates the total flux across the face between two cells by using flow variable values at points perpendicular to the face. The values at these points are determined through a linear reconstruction from cell-centered values, which results in additional source terms in the governing equations. The upwind method employs a second-order upwind approximation for the convective flux across a cell face. The diffusive flux, which contains a primary component and a cross-diffusion component, is determined through a coordinate transformation based on the geometry of the two adjoining cells.

The results of this study show that an adaptively-refined quadtree grid can yield a better grid distribution over the flow, and therefore give more accurate solutions as well as improved convergence rates than can a structured grid with a similar number of grid points.
1 INTRODUCTION

The numerical solution of a system of partial differential equations, by the use of finite difference, finite volume, or finite element methods, requires the generation of a grid, or mesh, of the computational domain over the physical domain [1]. The Navier-Stokes equations, which govern the behavior of fluid flow, are one such set of differential equations. The numerical solution of these equations is known as computational fluid dynamics (CFD). A grid used in the practice of CFD divides the flowfield's physical domain into discrete points or elements, or discretizes the physical and computational domains [2]. The grid must be sufficiently resolved to reduce local truncation errors [3] and must accurately model the geometry of the flowfield and its boundaries [2]. It is desirable that the grid allows for the efficient "communication" of data between neighboring grid points [2], and that the grid generation process be made as easy and automated as possible.

A structured grid is defined as the set of points formed by the intersections of the lines of a boundary-conforming curvilinear coordinate system, in which some line (in two dimensions) or surface (in three dimensions) is coincident with each boundary segment [4]. Geometries and flowfield data are stored in two- or three-dimensional arrays in which each position corresponds to a point in the structured grid. With this data structure, neighbors of each grid point, and the data associated with these neighbors, can be found by increasing or decreasing the index of the array. A body-fitted structured grid is well-ordered, allows higher-order difference representations, permits simple and accurate boundary difference representations, and can be clustered in high-gradient regions [5]. Structured grids also allow for the high aspect-ratio cells required in viscous regions. However, the generation of a structured grid for complex geometries is usually a difficult and time-consuming process. Also, adding a point to a structured grid implies adding a line (or a surface in three dimensions) and the entire set of points that accompany it. Therefore, a structured grid that is sufficiently refined in high gradient regions of a flow may contain unnecessary refinement in other flow regions, increasing storage requirements as well as the computational time necessary to solve the numerical equations.

In order to extend structured grid methods to complex geometries, composite grid schemes may be
used. These schemes use a set of interconnected simple grids that make up a larger composite grid [5]. Composite grids and the methods that employ them are also referred to as multi-block methods [6, 7], multi-zone methods [8], patched (non-overlapping) grids [4, 5, 9], overset grids [5, 10], chimera grids [10, 11], overlapping grids [11], domain-splitting methods [12], multi-domain methods [9], and subregion methods [4]. While these methods allow the use of structured grids with complex geometries, they also possess several drawbacks. In patched grid methods, grid generation is difficult because grid interfaces must be defined. Overset grid methods require interpolation of data points along boundaries and the transfer of information between grids, which can be difficult if more than two grids overlap at a point [5]. Other problems with composite grids include maintaining conservation, both across zonal boundaries and globally, and the difficulty of automating the grid generation procedure. Also, since each zone is independent (or uncoupled), the only interaction between subdomains occurs through the transfer of boundary data, which degrades the efficiency and perhaps the stability of the numerical method [13].

Another technique, known as grid embedding, represents the entire flow domain by a single coarse grid and locally refines the grid in regions of high gradients without changing the basic grid structure [3]. This method has been used to predict transonic flows using the Euler equations [14, 15] and incompressible flows using the Navier-Stokes equations [16]. While grid embedding typically suffers from the same convergence and stability problems as composite grid methods if grids of different levels are uncoupled [13]. Coelho et al. [3] employ a grid-embedding method which solves the entire flow domain simultaneously, without regard to the level of grid refinement, enhancing the coupling between regions of different grid levels. However, grid embedding has other drawbacks: a certain knowledge of the flowfield must be known a priori to embed the refined grid in the most advantageous position, and there is no obvious method by which a generalized data structure relating grids of differing levels can be determined.

A third technique, the use of unstructured grids composed of triangles (in two dimensions) or tetrahedra (in three dimensions), has received extensive study in CFD. For an excellent review of unstructured grid flow solvers, the reader is directed to the paper by Venkatakrishnan [7]. These types of unstructured grids have several advantages, namely their ability to be composed over complex geometric shapes, their flexibility in adapting to flow features, and their ease of implementation within automated grid generation procedures [7]. However, in viscous flow regions, triangular or tetrahedral unstructured grids contain either high aspect-ratio (highly skewed) cells or an inordinately large number of cells; high aspect-ratio cells lead to higher truncation error and reduced accuracy, while a large number of cells requires a prohibitively large amount of computational memory and time [17, 18, 19, 20]. Also, these
types of unstructured grids contain no repeating topology; that is, there is no natural or mathematical relationship between cells. This makes it difficult to compute the connectivity between cells and requires a large amount of computational memory to store the connectivity arrays [21].

Hybrid grids may alleviate the problems of triangular or tetrahedral unstructured grids in viscous regions. Hybrid grids are composed of either hybrid structured/unstructured grids or mixed-element unstructured grids [22]. Hybrid structured/unstructured grids use structured grids in viscous regions and unstructured grids in inviscid regions, but these grids require different data structures and flow solution methods for the different regions of the flow. Mixed-element unstructured grids use prismatic or hexahedral elements in viscous regions, however these grids require the same large connectivity memory as in triangular or tetrahedral unstructured grids.

Recently, a great deal of research has focused on the use of quadtree grids for flow solutions. A quadtree grid consists of quadrilateral elements that are successively subdivided into four smaller quadrilaterals, as shown in Figure 1.1. The quadtree grid's three-dimensional counterpart is the octree grid, which consists of hexahedral elements which are subdivided into eight smaller hexahedra. This method typically solves the flow on a coarse Cartesian grid and refines individual cells based on a predetermined grid adaption criteria. Complex boundaries can be resolved through the use of "cut cells," which are irregularly shaped cells formed by subtracting the boundary geometry from the intersecting Cartesian cell.

The evolution of the quadtree grid method has seen increasing sophistication in flow solution methods, boundary geometries, and grid adaption strategies. The full potential equation was solved by Purvis and Burkhalter [23] using a structured Cartesian mesh with cut cells at complex boundaries, but with no grid refinement. Boeing's TRANAIR code, as described in Young et al. [24], also solved

![Figure 1.1 Quadtree cell division](image-url)
the full potential equation (but in three dimensions) using a finite element method in conjunction with a locally refined Cartesian mesh stored in an octree data structure, wherein complex geometries were resolved with cut cells.

The unsteady Euler equations were solved by Berger and Osher [25], Berger and Collela [26], and Quirk [27] using an adaptive quadtree grid method on Cartesian geometries. Clarke et al. [28] and Morinishi [29] solved the Euler equations on a Cartesian mesh with cut cells at complex boundaries, but performed no grid adaptation. By combining quadtree grids with cut cells, Berger and LeVeque [30, 31, 32]. Epstein et al. [33], and Pember et al. [34] were able to solve the Euler equations on adaptive grids with complex geometries. Quirk [35] and Chiang [36] solved the unsteady Euler equations on a quadtree grid with complex boundaries using an upwind, finite-volume method. Bayyak et al. [37] also solved the unsteady Euler equations on a quadtree grid with complex boundaries using an upwind, finite-volume method, but with moving and deforming bodies. Lastly, De Zeeuw and Powell [38, 39]. De Zeeuw [40], and Coirier and Powell [41] solved the steady Euler equations on a quadtree grid with cut cells at complex boundaries using an upwind, finite volume method.

The extension of these methods to viscous flow solutions began with Quirk [27]. This work concentrated on inviscid flow solutions and was extended to the Navier-Stokes equations for only a relatively few simple test cases. Gooch [42] solved the Navier-Stokes equations using a state-vector splitting scheme, however this approach is non-conservative and suffers from monotonicity problems. A full examination of the quadtree grid method for viscous flow solutions was first performed by Coirier [43] and Coirier and Powell [44, 45], who solved the Navier-Stokes equations using a cell-centered, upwind, finite-volume method which was applied to a variety of complex geometrical configurations. Aftosmis [46] used a node-based, upwind scheme to simulate viscous flows on adaptively-refined meshes: an initial, body-fitted structured grid was used, eliminating the need for cut-cells, but it re-introduced the drawbacks associated with structured meshes. For three-dimensional analyses, Lockheed's SPLITFLOW code, as described by Karman [17], utilized a mixed-element hybrid grid approach, using prismatic cells in viscous regions and an unstructured Cartesian grid stored in an octree data structure in inviscid regions.

Each of the previously mentioned applications of quadtree grid methods to viscous flow problems employ density-based flow solution methods. Density-based methods, which were developed to solve the compressible Navier-Stokes equations, retain density as a dependent variable in the solution of the continuity equation [47]. These methods have been successfully applied to a wide range of compressible flow problems. Density-based flow solvers, however, show poor convergence properties when applied to
low-speed flows. This is due to an increase in the ratio of acoustic speeds to convective particle speeds in inviscid regions and a large difference between inviscid and viscous time scales in viscous regions. Therefore, at low Mach numbers there exists a wide disparity in the magnitudes of the eigenvalues, which creates a stiff system and slows convergence [48]. These problems may be remedied through the use of preconditioning methods, as described in References [48, 49, 50].

Low speed, viscous flows may also be successfully resolved through the solution of the incompressible Navier-Stokes equations, which may be accomplished through several existing methods. The artificial compressibility method introduced by Chorin [51] adds a pseudo-time derivative of pressure to the continuity equation. The solution is advanced to steady-state by marching in pseudo-time using the standard time-marching algorithms used for compressible flows. However, since the method marches in pseudo-time, it cannot be directly applied to unsteady flows. The vorticity-streamfunction approach replaces the velocity components with vorticity and streamfunction as the dependent variables, resulting in two uncoupled equations. This approach, while convenient to use in two dimensions, is not popular for three dimensions due to inherent complexities. Pressure-based methods, such as the SIMPLE family of algorithms described by Patankar [52], extract a Poisson equation for pressure from the continuity and momentum equations. These methods solve the momentum and pressure equations for the primitive variables in an iterative manner.

The present research involves a new technique in which an adaptively-refined quadtree grid is used in conjunction with a pressure-based algorithm to solve the incompressible Navier-Stokes equations. A cell-centered, finite volume procedure is used as the flow solution method. This procedure solves the governing equations for the primitive variables on a colocated grid and is based on the SIMPLE-type algorithms of Patankar [52]. Solution adaptive grid refinement is then used to resolve high-gradient flow regions.

In this work, the quadtree grid is first described, along with its refinement procedure and data structure. Next, solution adaptive grid refinement techniques are discussed. Two different flow solution methods are then described. For each method, the discretization of the governing equations and the flow solution algorithm are discussed, and the results obtained for a variety of test cases are presented. The extension of these methods to three-dimensional flows is then described. Finally, conclusions of the present research and recommendations for further study are presented.
2 THE QUADTREE GRID

The quadtree data structure is composed of a hierarchical relationship based on a recursive subdivision of space [53]. A quadtree grid, based on this data structure, consists of a number of quadrilateral cells that may be successively subdivided into four quadrants. In certain CFD applications, the quadtree grid is also referred to as an adaptively-refined Cartesian grid [38, 43], a locally refined rectangular grid [24], an h-refinement adaptive grid [18, 54], an unstructured Cartesian grid [17], an unstructured quadrilateral grid [55, 56], or some combination of the preceding.

Examples of a quadtree grid and its corresponding tree data structure are shown in Figures 2.1 and 2.2. Figure 2.1 shows an original "root" cell 0, which represents the entire physical domain. This cell is defined to have a level of one. Cell 0 is divided into four cells labeled 1 through 4, which have a grid level of two. Cells 2 through 4 are each subdivided into four, level-three cells, while cell 1 is further subdivided into four, level-four cells. A cell that is divided is defined as a "parent" cell, while cells that are formed by the division of a parent cell are known as "children" cells. Cells can be both a parent and a child. Figure 2.2 shows the data structure of this grid, with connections between parent and children cells.

The quadtree grid is particularly useful in the numerical solution of a system of partial differential equations, such as those that govern fluid flow. This is due to the inherent efficiency of the quadtree's data structure, as well as the fact that the grid can be locally refined where necessary. The efficiency of the quadtree data structure can be seen in Figure 2.1. For example, consider cell 9. If this cell's "south" neighbor was to be determined, it could be found in the following manner: Cell 9's parent is cell 3, and cell 3's "southwest" child is cell 9. It is known from the data structure that cell 3 has a south neighbor of cell 2. Therefore, the south neighbor of cell 9 must be the northwest child of cell 2, which is cell 8. This connection between neighboring cells can also be seen in Figure 2.2. Since a cell's position in the tree is known, and its neighbors must occupy certain positions in the tree relative to the cell, a cell's neighbors can easily be determined.

In the present study, the connectivity between cells is stored as an array of pointers from each
Figure 2.1 Quadtree grid

Figure 2.2 Quadtree data structure
cell to its parent cell and children cells (where defined). Also stored are pointers from cells to their surrounding nodes and faces (or edges), pointers from edges to their end nodes and adjacent cells, and the coordinates of each node. These "pointers" are actually implemented as indexed arrays of integers in the FORTRAN 77 code that was written to solve the system of equations. These arrays are defined in the following list:

**NCELL** - Total number of cells (elements)
**NNODE** - Total number of nodes (points)
**NEDGE** - Total number of edges (faces)

**X(N)** - x-coordinate of node N
**Y(N)** - y-coordinate of node N

**LNODE(1,NEL)** - Relates local node number I (I = 1,2,3,4) of element NEL to a global node number
- **LNODE(1,NEL)** = global node number, southwest corner
- **LNODE(2,NEL)** = global node number, southeast corner
- **LNODE(3,NEL)** = global node number, northeast corner
- **LNODE(4,NEL)** = global node number, northwest corner

**NELCON(NC,NEL)** - NCth connection of element NEL (NC = 1,2,3,4)
- **NELCON(1,NEL)** = south element
- **NELCON(2,NEL)** = east element
- **NELCON(3,NEL)** = north element
- **NELCON(4,NEL)** = west element
- **NELCON(5,NEL)** = south edge
- **NELCON(6,NEL)** = east edge
- **NELCON(7,NEL)** = north edge
- **NELCON(8,NEL)** = west edge

**NEDGECON(NC,NED)** - NCth connection of edge NED (NC = 1,2,3,4)
- **NEDGECON(1,NED)** = first node
- **NEDGECON(2,NED)** = second node
- **NEDGECON(3,NED)** = left element
- **NEDGECON(4,NED)** = right element

**CLEVEL(NEL)** - Level of element NEL

**ELEVEL(NED)** - Level of edge NED

**CPARENT(NEL)** - Parent element of element NEL
CCHILD(I,NEL) - Child elements of element NEL.

CCHILD(1,NEL) = southwest element
CCHILD(2,NEL) = southeast element
CCHILD(3,NEL) = northeast element
CCHILD(4,NEL) = northwest element

EPARENT(NED) - Parent edge of edge NED
ECHILD(I,NED) - Child edges of edge NED
ECHILD(1,NED) = first edge
ECHILD(2,NED) = second edge

The cells of a quadtree grid that have no children cells, sometimes referred to as the "leaves" of a tree data structure, are the cells on which the solution of the governing equations takes place and are defined as "top-level cells." These top-level cells are indicated by the cell numbers inside boxes in Figure 2.2. In the present research, no two neighboring top-level cells are allowed to differ in cell level by more than one. Also, "holes" in the grid, defined as top-level cells in which all neighboring top-level cells on two opposite sides of the cell have a level greater than that of the cell, are not allowed. These restrictions result in a "smooth" mesh, which lessens both the complexity of the data structure and the degradation of the solution.

The solution procedure involves first generating an initial coarse grid over the flow domain. The governing equations are then solved on the coarse grid until a predetermined level of convergence or maximum number of iterations has been reached. The grid is then refined in high-gradient regions of the flow according to a predefined grid adaption criteria, and a new solution is found on the refined grid. This procedure of grid refinement and flow solution is repeated until a predetermined maximum grid level has been reached. The flow solution is then found on the final adapted grid.
There exist several distinctly different grid adaption strategies for CFD that have emerged over the past decade [57]. These different methods can be classified as

- r-methods These methods "relocate" grid points through translation towards high-gradient regions of the flow. The number of grid points and elements remains fixed, and the data structure remains unchanged.

- h-methods The h methods refine mesh elements of size $h$ by adding (or deleting) grid points and elements.

- p-methods These methods, used in finite element solutions, involve modifying the spectral order of the approximations over subregions of the grid.

- Combined methods A combination of two or more of the above methods is also possible.

The present analysis involves only h-refinement, in which quadrilateral cells are divided into four quadrants.

A variety of grid adaption criteria may be used to determine which quadtree cells to refine after an intermediate flow solution is found. If knowledge of the flow solution is known beforehand, regions of the grid to be refined may be specified according to their coordinate values.

De Zeeuw and Powell [38] suggest subdividing cells that have a total velocity difference across the cell that is greater than some percentage (5%, typically) of the maximum total velocity difference. This strategy can be applied using any number of flow variables or their gradients.

Kallinderis and Nakajima [20] define a generic “detection parameter." $S$, and divide all cells in which the value of $S$ is above a threshold level, $S_{th}$, given by

$$S_{th} = S_{ave} + \alpha S_{sd}$$

(3.1)

where $S_{ave}$ is the average of $S$. $S_{sd}$ is the standard deviation, and $\alpha$ is an empirically determined value, typically 0.3. Suggested detection parameters are velocity gradients for viscous regions and velocity differences for shocks.
Coiier [43] and Charlton and Powell [21] refine cells based on the cell-size-weighted velocity divergence and curl. Such that

\[
\tau_c = \nabla \cdot (\nu \nabla^{3/2})
\]

\[
\tau_R = \nabla \cdot (c \nabla^{3/2})
\]

where \( l \) is a cell's characteristic length. Cells are tagged for refinement if

\[
(\tau_c > \sigma_c \text{ or } \tau_R > \sigma_R) \text{ and } l > l_{\text{min}}
\]

and for derefinement if

\[
(\tau_c < \sigma_c \text{ and } \tau_R < \sigma_R) \text{ and } l < l_{\text{max}}
\]

where

\[
\sigma_c = \sqrt{\frac{\overline{\tau_c^2}}{N}}
\]

and

\[
\sigma_R = \sqrt{\frac{\overline{\tau_R^2}}{N}}
\]

The Lockheed SPLITFLOW code [17] has a variety of adaption functions available to users, generically referred to as a function \( f \). For each cell, the quantities

\[
a = \frac{\partial f}{\partial x} \Delta x^{1+\nu}
\]

\[
b = \frac{\partial f}{\partial y} \Delta y^{1+\nu}
\]

\[
c = \frac{\partial f}{\partial z} \Delta z^{1+\nu}
\]

and

\[
A = \sqrt{a^2 + b^2 + c^2}
\]

are computed, where \( \nu \) is a user-defined constant. The magnitude of the adaption parameter \( A \) is statistically averaged to find its mean, \( m \), and standard deviation, \( \sigma \). An upper threshold value of the adaption function is defined as

\[
A_u = m + \epsilon \sigma
\]

and a lower threshold value is defined as

\[
A_l = m - d \sigma
\]

where \( d \) and \( \epsilon \) are user-defined constants. Cells with values of the adaption function greater than \( A_u \) are refined, while cells with adaption function values lower than \( A_l \) are derefined. There are also limits placed on the minimum and maximum cell size.
Warsi and Kania [18] determine an error criterion for the global mesh as

\[ \Psi = \left( \omega \sum_i \frac{\nabla \phi_i \cdot \mathbf{v}_i}{\sum_i \mathbf{v}_i} \right) \nu \]

where \( \omega \) is a constant, \( \phi \) is a flow variable, \( \mathbf{v}_i \) is a cell's volume, and \( \nu \) is the mean domain volume, and refine those cells with a local solution error \( \nabla \phi_i \cdot \mathbf{v}_i \geq \Psi \).

This study primarily involves showing the proof-of-concept of combining a quadtree grid with a pressure-based flow solution algorithm, and is less concerned with the comparison of different adaption criteria. Therefore, different adaption criteria are used throughout the study, depending on the test case. These criteria are chosen to yield a desired grid, produce a solution of increased accuracy, or match the adaption criteria of previous research. Adaption criteria are described along with the test cases with which they are used. Also, since the present method involves steady flow solutions on initially coarse grids, it is believed that any adaption criteria will yield only a small amount of grid derefinement, and any savings in computational time or memory will be minimal. Therefore, derefinement of cells is not employed.
4 UNI-DIMENSIONAL POWER-LAW METHOD

The flow solution method described in this chapter is based on previous methods used to calculate fluxes across regions of different levels of grid refinement in embedded grid [3] and multi-block methods [5]. These methods, both previous and present, calculate fluxes between cells of different sizes through the use of the values of flow variables at points that are perpendicular to cell faces. The flow values are not stored at these points, but are found through interpolation of the stored cell-center flow values. Since the points used in the flux calculation are perpendicular to cell faces, they lie on lines that are parallel to some coordinate direction, thus the term “uni-dimensional.”

Theoretical Formulation

This section describes the process of transforming the governing partial differential equations into a discretized form which can be solved numerically. A scalar transport equation, which has the same form as the momentum equations, is discretized first. A velocity-pressure coupling mechanism is then described. The system of equations is then closed by a pressure-correction equation and a pressure equation, which are derived from the continuity equation. Finally, boundary conditions and their implementation in the numerical solution are discussed.

Governing Equations

The examples presented in this study involve the solution of the steady, laminar, two-dimensional, incompressible Navier-Stokes equations, which can be expressed as

continuity equation:

\[
\frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0
\]  

+4.1

x momentum equation:

\[
\frac{\partial}{\partial x} (\rho u u) + \frac{\partial}{\partial y} (\rho v u) = \frac{\partial}{\partial x} \left( \mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial u}{\partial y} \right) - \frac{\partial p}{\partial x}
\]  

+4.2
y momentum equation:

\[
\frac{\partial}{\partial x} (\rho u y) - \frac{\partial}{\partial y} (\rho v y) = \frac{\partial}{\partial x} \left( \mu \frac{\partial y}{\partial x} \right) - \frac{\partial}{\partial y} \left( \mu \frac{\partial y}{\partial y} \right) - \frac{\partial p}{\partial y}
\]

Discretization of the Scalar Transport Equation

A scalar transport equation, which has the same form as each of the momentum equations, is discretized and examined in this section. Once the scalar transport equation is discretized, the discretized momentum equations can be found in a similar manner. Also, by studying the properties of the discretized scalar transport equation, the behavior of the discretized momentum equations can be inferred in a straight-forward manner.

The steady, two-dimensional, generalized transport equation for a scalar variable \( \phi \) can be written as

\[
\frac{\partial}{\partial x} (\rho u \phi) - \frac{\partial}{\partial y} (\rho v \phi) = \frac{\partial}{\partial x} \left( \Gamma \frac{\partial \phi}{\partial x} \right) - \frac{\partial}{\partial y} \left( \Gamma \frac{\partial \phi}{\partial y} \right) + S
\]

or

\[
\frac{\partial J_x}{\partial x} - \frac{\partial J_y}{\partial y} = S
\]

where

\[
J_x = \rho u \phi - \Gamma \frac{\partial \phi}{\partial x}
\]

\[
J_y = \rho v \phi - \Gamma \frac{\partial \phi}{\partial y}
\]

and \( S \) represents a source term. The terms \( J_x \) and \( J_y \) represent the total fluxes in the \( x \) and \( y \) directions, respectively. The first terms on the right hand side of the above equations represent the convective fluxes while the second terms represent the diffusive fluxes.

Figure 4.1 shows a generalized section of a quadtree grid that will be used in the discretization of the governing equations. Integrating the continuity and transport equations over the control volume \( V_p \), the quadrilateral element surrounding point \( P \), gives

\[
F_x = (F_{u_1} - F_{u_2}) + F_n - F_s = 0
\]

and

\[
J_x = (J_{u_1} + J_{u_2}) + J_n - J_s = (S_C + S_{\text{po}}) V_p
\]

where

\[
F_x = \int_{P} (\rho u dy) = (\rho u A)_x
\]
represents the mass flux across edge $e$.

$$J_e = \int_2^3 J_e \, dy = \int_2^3 J_e \, A_e$$

This gives the integrated total flux of scalar variable $\phi$ across edge $e$, and the source term has been linearized in an appropriate manner. Fluxes across the other faces of control volume $V_P$ are defined similarly.

Multiplying the integrated continuity equation by the scalar variable $\phi$ at point $P$ and subtracting the result from the integrated transport equation gives

$$(J_e - F_{e,0P}) - (J_{e_1} - F_{e_1,0P}) - (J_{e_2} - F_{e_2,0P})$$

$$+ (J_n - F_{n,0P}) - (J_s - F_{s,0P}) = (S_C - S_{POP})V_P$$

Following the discretization procedure of Patankar [52], each term in the above equation can be represented as follows:

$$(J_e - F_{e,0P}) = a_g (\phi_P - \phi_{E_1})$$

$$(J_{e_1} - F_{e_1,0P}) = a_W (\phi_{W_1} - \phi_{P_1})$$

$$(J_{e_2} - F_{e_2,0P}) = a_W (\phi_{W_2} - \phi_{P_2})$$

$$(J_n - F_{n,0P}) = a_N (\phi_P - \phi_N)$$

$$(J_s - F_{s,0P}) = a_S (\phi_S - \phi_P)$$
where

\[ \eta_E = D_w \eta(P_i) = \left[ -F_w, 0 \right] \]
\[ \eta_{w1} = D_w \eta(P_{w1}) = \left[ -F_{w1}, 0 \right] \]
\[ \eta_{w2} = D_w \eta(P_{w2}) = \left[ -F_{w2}, 0 \right] \]
\[ \eta_v = D_v \eta(P_{v}) = \left[ -F_v, 0 \right] \]
\[ \eta_z = D_z \eta(P_z) = \left[ -F_z, 0 \right] \]

and

\[ D_w = \frac{f_w \Delta y_i}{\delta x_i} \]
\[ P_i = \frac{F_i}{D_w} \]

Terms at the other control volume faces are determined similarly. The operator \([a, b] \) denotes the maximum of the included quantities. The function \( \eta(|P|) \) is defined by the power-law scheme [52], for which

\[ \eta(|P|) = \left[ 0, (1 - 0.1|P|)^2 \right] \]

This function represents a curve fit to the exact solution of the steady, one-dimensional convection - diffusion equation with no source terms.

Substituting the discretized terms, Equation 4.12 becomes

\[ a\eta_{\Omega} + a_{w1}\eta_{\Omega} + a_{w2}\eta_{\Omega} + a_{\eta}\eta_{\Omega} + a_{s}\eta_{\Omega} = a_{e}\eta_E + a_{w1}\eta_{w1} + a_{w2}\eta_{w2} + a_{\eta}\eta_{\eta} + a_{s}\eta_{s} - (S_{\iota} - S_{p}\eta) \psi \]

The values of \( \eta \) at locations not represented by grid points, such as \( E_1, P_1, \) and \( P_2 \) in Figure 4.1, can be determined by linear reconstruction as

\[ \eta_{E_1} = \eta_E + (\nabla \eta_E \cdot \tilde{EE}_1) \]
\[ \eta_{P_1} = \eta_P + (\nabla \eta_P \cdot \tilde{PP}_1) \]
\[ \eta_{P_2} = \eta_P + (\nabla \eta_P \cdot \tilde{PP}_2) \]

The gradients of flow quantities are determined by a linear least-squares procedure [58]. Substituting these values in the governing equation yields

\[ a\eta_{\Omega} + a_{w1}\eta_{\Omega} + a_{w2}\eta_{\Omega} + a_{\eta}\eta_{\Omega} + a_{s}\eta_{\Omega} + a_{w1}(\nabla \eta_{\Omega} \cdot \tilde{PP}_1) + a_{w2}(\nabla \eta_{\Omega} \cdot \tilde{PP}_2) \]
\[ = a\eta_{\Omega} + a_{w1}\eta_{w1} + a_{w2}\eta_{w2} + a_{\eta}\eta_{\eta} + a_{s}\eta_{s} + a_{e}(\nabla \eta_{\Omega} \cdot \tilde{EE}_1) - (S_{\iota} - S_{p}\eta) \psi \]
Notice that terms involving the gradient at point $P$ are contained on the left hand side of the governing equation. These terms involve neighboring control volumes that are one level higher in the quadtree grid formulation than the level of control volume $V_P$. Similarly, terms involving the gradient at a neighboring grid point fall on the right hand side of the governing equation. These terms involve neighboring control volumes one level lower than the level of control volume $V_P$. Therefore, the discretized scalar transport equation, written for a general quadtree grid control volume, can be represented as

$$a_P \phi_P = \sum a_{nb} u_{nb} - \sum a_{nb}(\nabla \phi_P \cdot \vec{r}_{P_i}) + \sum a_{nb}(\nabla \phi_{nb} \cdot \vec{r}_{nb_i}) + S_c \cdot V_P$$

where

$$a_F = \sum a_{nb} - S_P \cdot V_F$$

and the subscript $nb$ represents the neighboring grid points. Again, if the control volume about point $P$ has a quadtree grid level of $l$, then the second term on the right hand side of the above equation contains terms arising from the $l - 1$ neighbors of $V_P$, while the third term contains terms arising from the $l - 1$ neighbors of $V_P$.

**Discretized Momentum Equations**

The momentum equations, written in the preceding form, can now be expressed as

$$a_P u_P = \sum a_{nb} u_{nb} - \sum a_{nb}(\nabla u_P \cdot \vec{r}_{P_i}) + \sum a_{nb}(\nabla u_{nb} \cdot \vec{r}_{nb_i}) + S_c \cdot V_P - \left( \frac{\partial p}{\partial x} \right)_P V_P$$

$$u_P v_P = \sum a_{no} v_{no} - \sum a_{no}(\nabla v_P \cdot \vec{r}_{P_i}) + \sum a_{no}(\nabla v_{no} \cdot \vec{r}_{no_i}) + S_c \cdot V_P - \left( \frac{\partial p}{\partial y} \right)_P V_P$$

where the pressure gradient terms have been written separately from the discretized source terms.

**Velocity-Pressure Coupling**

The present analysis uses the colocated (or non-staggered) grid approach, in which values of velocity and pressure are stored only at cell centers. This choice was motivated by the difficulty of defining a grid that is staggered with respect to a quadtree grid. The colocated grid formulation requires that fluxes at cell faces be interpolated from cell-centered values. A linear interpolation of these fluxes decouples the velocity and pressure fields, which results in a “checkerboard” instability. This problem is alleviated by the procedure developed by Rhie and Chow [59] and refined by Peric et al. [60], as outlined below.

Referring again to Figure 4.1, the $u$ velocity at grid point $P$, written in terms of Equation 4.33, can be expressed as

$$u_P = \frac{1}{a_P} \left\{ \sum a_{nb} u_{nb} - \sum a_{nb}(\nabla u_P \cdot \vec{r}_{P_i}) + \sum a_{nb}(\nabla u_{nb} \cdot \vec{r}_{nb_i}) + S_c \cdot V_P \right\} - \frac{1}{a_P} \left( \frac{\partial p}{\partial x} \right)_P$$
where
\[ \dot{d}_p = \frac{V_p}{\Delta p} \]  
(4.36)

Representing the first term in the above equation as a pseudovelocity \( u \) gives
\[ u_p = u_p - \dot{d}_p \left( \frac{\partial p}{\partial x} \right)_p \]  
(4.37)

The \( u \) velocities at grid point \( E \) and the midpoint of edge \( e \) can be similarly written as
\[ u_E = u_E - \dot{d}_E \left( \frac{\partial p}{\partial x} \right)_E \]  
(4.38)
\[ u_e = u_e - \dot{d}_e \left( \frac{\partial p}{\partial x} \right)_e \]  
(4.39)

The terms \( u_e \) and \( \dot{d}_e \) are approximated by a linear interpolation, resulting in an equation for an edge velocity
\[ u_e = \text{lin} \left[ u_p + \dot{d}_p \left( \frac{\partial p}{\partial x} \right)_p, u_E + \dot{d}_E \left( \frac{\partial p}{\partial x} \right)_E \right] - \text{lin} \left[ \dot{d}_p, \dot{d}_E \right] \left( \frac{\partial p}{\partial x} \right)_e \]  
(4.40)

where \( \text{lin}[a,b] \) represents a suitable linear interpolation of quantities \( a \) and \( b \). Note that the term \( i(\partial p/\partial x)_e \) can be directly evaluated from cell center values. Other face velocities can be determined similarly. The above equation representing a cell interface velocity equals a linear interpolation of the cell center velocities plus a correction involving a third-order pressure gradient term \([9]\). This formulation serves as a mechanism to couple the velocity and pressure fields.

**Pressure-Correction Equation**

To close the system of equations, a correction equation for pressure is formulated from the continuity and momentum equations. The discretized momentum equations given in Equations 4.33 and 4.34 can only be satisfied if a correct pressure field is known. If an incorrect, or guessed, pressure field, denoted \( p^* \), is used to solve the momentum equations, the resulting velocity field, denoted \( u^* \) and \( v^* \), will not satisfy the continuity equation. The pressure and velocities can be written in terms of their guessed values and a correction as
\[ p = p^* + p' \]  
(4.41)
\[ u = u^* + u' \]  
(4.42)
\[ v = v^* + v' \]  
(4.43)

The \( u \)-momentum equation, written in terms of a guessed pressure field and an intermediate, "starred" velocity field, can be expressed as
\[ a_p u_p^* = \sum a_n b_n a_n^* - \sum a_n b_n (\nabla u_n^* \cdot \vec{r}_n) + \sum a_n b_n (\nabla u_n^* \cdot \vec{r}_{nb}) + \nabla \cdot (\vec{V}^* - A_p \Delta p^*) \]  
(4.44)
where \( A_p \) is the cross-sectional area of the cell surrounding point \( P \) in the direction perpendicular to the velocity component. In the case of a two-dimensional flow, \( A_p = \Delta y \) for the \( u \)-momentum equation.

Equation 4.41 is subtracted from Equation 4.33, resulting in

\[
\sum u_{n-b} u_{n-b}' = \sum \Delta y \left( \sum u'_{n-b} \cdot \vec{r}_{P} \right) - \sum \Delta y \left( \sum u'_{n-b} \cdot \vec{r}_{P} \right) - \Delta y \Delta \rho'_{P} \tag{4.45}
\]

Since the terms in the above equation within the summation signs go to zero at convergence, they can be neglected, which reduces the equation to

\[
u_{p} u_{p}' = -\Delta y \Delta \rho'_{P} \tag{4.46}
\]

or

\[
u_{p}' = -\Delta y \Delta \rho'_{P} \tag{4.47}
\]

where

\[
\Delta y = \frac{\Delta y}{\Delta p} = \frac{\Delta p}{\Delta y} \tag{4.48}
\]

Edge velocities such as \( u_{e} \), for example, can similarly be expressed as

\[
u_{e} = \nu_{e}' = u_{e}' \tag{4.49}
\]

where

\[
u_{e}' = -\Delta y \Delta \rho'_{e} \tag{4.50}
\]

and

\[
\Delta y = \text{lin}[\Delta p, \Delta E] \tag{4.51}
\]

In general, velocities can be expressed in terms of starred velocities and velocity-corrections as

\[
u = \nu' - \Delta \rho' \tag{4.52}
\]

\[
\nu = \nu' - \Delta \rho' \tag{4.53}
\]

These equations are known as the velocity-correction equations. Note that the pressure-correction difference term, \( \Delta \rho' \), in each of these equations is the difference in pressure-correction in the direction parallel to the velocity component under consideration.

Consider again the continuity equation integrated about the control volume \( \Gamma P \) in Figure 4.1.

\[
(puA)_e - (puA)_w_1 - (puA)_w_2 + (puA)_n - (puA)_s = 0 \tag{4.54}
\]
If the velocity terms in the above equation are replaced by their corresponding velocity-correction equations, the continuity equation becomes

\[ \sum A_{p} \left( \rho_{v} \frac{d}{dt} \frac{\partial p}{\partial x} \right) = \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} = 0 \]  

Upon rearrangement, this equation can be written as

\[ \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial p}{\partial x} = \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} = 0 \]  

where

\[ b_{p} = \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial p}{\partial x} \]  

Linear reconstruction is used to obtain pressure-correction values not located at control volume centers as

\[ \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial p}{\partial x} = \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} - \rho_{v} d_{A_{p}} \rho_{v} \frac{d}{dt} \frac{\partial (\rho v u)}{\partial x} = 0 \]  

Pressure Equation

Although the discretized momentum and pressure-correction equations are sufficient to solve the system of equations, the solution convergence can be enhanced by solving an additional equation for pressure. The discretized pressure equation is derived from the continuity and momentum equations in the following manner.

If the velocity components \( u \) and \( v \) at cell faces are written in terms of their pseudovelocities \( u_{c} \) and \( v_{c} \) as

\[ u_{c} = u_{c} - d_{x} \left( \frac{\delta p}{\delta x} \right) \frac{\delta x_{c}}{\delta x} \]  

(4.62)
and substituted into the discretized continuity equation, a discretized equation for pressure can be found in a manner similar to the derivation of the pressure-correction equation. The pressure equation is written as

\[
\begin{align*}
\sum a_{nb} p_{n_b} &- \sum a_{nb} (\nabla p \cdot \vec{r}_p) - \sum a_{nb} (\nabla p_{nb} \cdot \vec{r}_{nb}) = b_p
\end{align*}
\]  

where

\[
\begin{align*}
b_p &= \sum (\rho u)_{A_j} - \sum (\rho u)_{A_k} + \sum (\rho v)_{A_j} - \sum (\rho v)_{A_k}
\end{align*}
\]

The summations over the subscripts in the source term indicate the inclusion of each edge of the control volume. Note that the velocities that were used in the discretization of the pressure equation have not been approximated in any way. Therefore, if a correct velocity field is known, the solution of the pressure equation will yield the exact pressure field.

**Boundary Conditions**

The solution of any system of partial differential equations, by either an analytical or numerical method, is dependent on the system’s initial and boundary conditions. Therefore, it is critical that these conditions are represented accurately and implemented correctly in the solution procedure. Since the present analysis involves only steady fluid flow, initial conditions are less consequential, and only the system’s boundary conditions are discussed here.

A control volume adjacent to a boundary on its west edge is shown in Figure 4.2. The boundary edge, \(b\), has a midpoint \(B\). Consider again the discretized scalar transport equation.

\[
\begin{align*}
a_{popp} &= \sum a_{nb} x_{nb} - \sum a_{nb} (\nabla x \cdot \vec{r}_p) + \sum a_{nb} (\nabla x_{nb} \cdot \vec{r}_{nb}) + \dot{S} c V
\end{align*}
\]  

\[
\begin{align*}
Figure 4.2 Quadtree grid boundary control volume
\]
In the case of a boundary cell, the summations about neighbor cells, nb, are made only for non-boundary elements. The boundary conditions are incorporated into the linearized parts of the source term, \( S_c \) and \( S_p \).

At boundaries where the flow enters or leaves the domain, the source terms are updated as

\[
S_c = S^c_c + i D_h A(|P_n|) - \left\{ -F_{h,0} \right\} o_n
\]

and

\[
S_p = S^p_p - i D_h A(|P_n|) + \left\{ -F_{h,0} \right\}
\]

where \( S^c_c \) and \( S^p_p \) are parts of the source terms arising from other sources or boundaries. At inflow boundaries, the flow variable \( o_n \) and mass flow rate are specified. At outflow boundaries, flow variables and mass flow rates are extrapolated from the cell-center values, and updated to ensure global mass conservation.

At wall boundaries, the mass flux is zero, and the source terms reduce to

\[
S_c = S^w_c + D_h o_n
\]

and

\[
S_p = S^w_p - D_h
\]

Symmetry boundaries contain a zero gradient of flow variables with respect to the normal direction. Therefore, no additions to the source terms are required.

The pressure-correction equation and the pressure equation also require some modification at cells adjacent to a boundary. These equations are written as

\[
ap p'p = \sum a_{nb} p_{nb} - \sum a_{nb} (\nabla p' \cdot \nabla) + \sum a_{nb} (\nabla p'_{nb} \cdot \nabla) + b_p
\]

where

\[
b_p' = \sum u_{nb} o_{nb} + u_{nb} A_b
\]

and

\[
ap pp = \sum a_{nb} p_{nb} - \sum a_{nb} (\nabla p \cdot \nabla) + \sum a_{nb} (\nabla p_{nb} \cdot \nabla) + b_p
\]

where

\[
b_p = \sum u_{nb} o_{nb} + u_{nb} A_b
\]

Again, the summations are only over non-boundary neighbors. Mass flow rates are known at all boundaries, and are explicitly added to the source terms.
Solution Procedure

The solution procedure used in the present analysis is the SIMPLER method of Patankar [32] implemented within a grid adaption procedure. This algorithm can be summarized as

1. Generate an initial coarse grid.

2. Refine the grid, if necessary, in areas that are known a priori to require a high degree of grid clustering.

3. Apply the SIMPLER algorithm as follows:

   (a) Guess a velocity field. If a coarse grid solution is known, retain previously determined velocities at cells that have not been subdivided, and use a parent cell’s velocities as an initial guess for newly formed children cells.

   (b) Calculate the coefficients of the momentum equations and hence calculate the pseudovelocity \( u \) and \( v \).

   (c) Calculate the coefficients of the pressure equation, and solve it to obtain the pressure field.

   (d) Treating this pressure field as \( p^* \), solve the momentum equations to obtain new values of \( u^* \) and \( v^* \).

   (e) Solve the pressure-correction equation to find \( p' \).

   (f) Correct the velocity field with the velocity-correction equations, but do not correct the pressure field.

   (g) Return to Step (b) and repeat until convergence.

4. If the maximum grid level has not been reached, refine the grid according to the adaption criteria, and return to Step 3.

The discretized momentum, pressure, and pressure-correction equations are solved by a symmetric Gauss-Seidel point iterative method with under-relaxation.

Results

The uni-dimensional power-law method was applied to several test cases: a lid-driven cavity flow, a duct flow with a sudden contraction, a Z-channel flow with two bends, and the flow over a circular cylinder. Results and discussion of each case are described below.
Lid-driven Cavity Flow

The classical lid-driven cavity flow, in which the flow in a square cavity is driven solely by the motion of the cavity's lid, was the first test case considered. For comparison, the flow was first solved on structured grids of sizes $64 \times 64$ (4,096) cells and $128 \times 128$ (16,384) cells, using the staggered grid SIMPLER procedure of Patankar [52]. The results obtained on the $128 \times 128$ grid compare favorably to the grid independent solutions of Ghia et al. [61] and shall be considered here as the benchmark solution.

The present quadtree method was used to solve a lid-driven cavity flow first at a Reynolds number of 100 on the grid shown in Figure 4.3, which contains 4,288 top-level cells. The refinement locations of this grid were set explicitly, with an initial grid consisting of $16 \times 16$ cells. At each grid level, the flow solution was allowed to converge to a certain total velocity RMS value before refinement.

Figures 4.4-4.6 show $u$-velocity profiles at different $x$ cross-sections of the flow. It can be seen that the solutions are essentially the same for each grid system at the cavity's centerline. However, at locations near the $x = 0$ wall, the solution of the present method more closely matches the exact solution of the $128 \times 128$ grid than does the solution of the $64 \times 64$ grid.

The convergence histories, based on total velocity RMS values, of the solutions on each grid are shown in Figure 4.7. It can be seen that the quadtree solution converges at a much faster rate than solutions obtained on structured grids. This behavior is possibly due to multigrid-type effects, in which the lower-frequency errors are removed during the solutions on the coarser grids. The "spikes" in the convergence plot indicate refinement of the quadtree grid. Also, a comparison of the CPU times required for convergence of each solution show that these times closely correspond to the total number of iterations of each solution.

An analysis was also performed on a driven cavity flow at $Re = 400$. An initial grid of $16 \times 16$ cells was used, and grid refinement locations were set explicitly, to give a final grid containing 9,472 top-level cells, as shown in Figure 4.8.

The $u$-velocity profiles for the solution given by the present method, along with the structured grid solutions, are shown in Figures 4.9 and 4.10. These figures show that the quadtree method again yields a solution with accuracy comparable to that given by the $128 \times 128$ grid.

Figure 4.11 shows the total velocity RMS convergence for the solutions obtained on each grid. The quadtree grid method converges at a rate similar to that of the $64 \times 64$ grid and much faster than that of the $128 \times 128$ grid.
Figure 4.3 Driven cavity. $Re = 100$: quadtree grid (4,288 cells).

Figure 4.4 Driven cavity. $Re = 100$: $u$ velocity profiles. $x = 1/2$.
Figure 4.5 Driven cavity. $Re = 100$: $u$ velocity profiles. $x = 1/32$

Figure 4.6 Driven cavity. $Re = 100$: $u$ velocity profiles. $x = 1/64$
Figure 4.7 Driven cavity. $Re = 100$: total velocity RMS values

Figure 4.8 Driven cavity. $Re = 400$: quadtree grid (9,472 cells)
Figure 4.9 Driven cavity. \( Re = 400 \): \( u \) velocity profiles. \( x = 1/2 \)

Figure 4.10 Driven cavity. \( Re = 400 \): \( u \) velocity profiles. \( x = 1/64 \)
Duct Flow with a Sudden Contraction

The second test case was the two-dimensional flow through a duct with a sudden contraction. The results were compared to the experimental results of Durst et al. [62] and the numerical results of Coelho et al. [3]. Inlet conditions were taken from the experimental data, and outflow conditions were determined through global mass conservation. Because of the symmetry of the problem, computations were carried out on only one-half of the duct, with symmetrical boundary conditions at the duct's centerline. The duct was assumed to have a height of 10 mm and a contraction ratio of 4:1. The Reynolds number based on the average inflow velocity and duct height was 95.

The initial coarse grid contained 40 cells in the longitudinal direction, 24 cells across the inlet, and 6 cells across the outlet, giving 920 total cells. After an intermediate solution was found, the grid was adaptively refined by dividing the cells with a total velocity difference greater than 10% of the maximum total velocity difference. The final quadtree grid was obtained after four levels of adaption, resulting in the grid shown in Figure 4.12, containing 2,622 top-level cells.

Figures 4.13–4.15 show u velocity profiles predicted by the present method along with previously reported experimental and numerical results. As these figures show, the present results compare favor-
ably with the numerical results and are within the range of uncertainty observed in the experimental data.

Channel Flow with Two Right-Angle Bends

The flow through a channel with two bends that form a Z-like shape was the third test case analyzed. This geometry is taken from Perng and Street [12] and is chosen for its irregular shape, which requires either an unstructured, highly-skewed, or multi-block grid configuration to obtain a flow solution. A schematic diagram of the channel is shown in Figure 4.16. The channel consists of an inlet section of height \( h \) and a length of \( 4h \), which enters an expansion region at a distance of \( 2h \). The expansion region has a width of \( 2h \) and a height of \( 5.625h \). A contraction section of width \( h \) and length \( 12h \) completes the channel. The inlet boundary condition consists of a prescribed parabolic velocity profile with a maximum velocity of 1.0 and an average velocity of 1.0, while outflow conditions are determined through global mass conservation. The flow has a Reynolds number of 200.

The flow was initially solved on a grid consisting of 328 equally sized cells, which is shown in Figure 4.17, excluding the region of the contraction section past \( x = 8h \). The grid was refined to four levels, using an adaptation criteria of dividing cells with a total velocity difference greater than 15% of the maximum total velocity difference. To resolve the boundary layer, cells next to the walls of the channel were also subdivided at each level of refinement. The final grid, shown in Figure 4.18, contained 5,818 top-level cells.

Figure 4.19 shows a plot of the streamline contours of the channel flow. The flow contains three
Figure 4.13 Duct with sudden contraction: u velocity profiles, $x = -2.5\,mm$

Figure 4.14 Duct with sudden contraction: u velocity profiles, $x = -1.0\,mm$
Figure 4.15 Duct with sudden contraction: $u$ velocity profiles, $x = 1.0\,mm$

Figure 4.16 Z channel: schematic diagram
Figure 4.17  Z channel: initial grid

Figure 4.18  Z channel: final grid
large recirculating regions, one in the upper right corner of the expansion region, one extending down the left side of the expansion region, and one along the top of the contraction region. A comparison of the streamline contour plot with the adapted grid shows that the grid is more refined in areas of the flow with a high velocity gradient.

A plot of the normalized velocity profiles at two locations of the channel is shown in Figure 4.20. The results are compared to those of Perng and Street [12], who used a structured multi-block grid containing 6,924 cells, almost 20% more than the number of cells used by the present method. The results obtained by the present first-order method compare favorably to the previous results obtained using a higher-order scheme.

**Flow over a Circular Cylinder**

The flow around a two-dimensional circular cylinder is one whose properties have been well documented. These properties include the generation of vortices in the proximity of the cylinder, their convection downstream, and their influence on the pressure and velocity fields near the cylinder. In the present analysis, steady vortex generation is considered, for which a Reynolds number of forty, based on diameter, was chosen. The resulting flowfield lies within the "twin-vortex stage," as defined by
Morkovin in a survey of flow around circular cylinders [63].

The generation of a quadtree grid over a circular cylinder, or any curved or otherwise irregularly shaped boundary, requires the use of "cut cells." These are originally quadrilateral cells which, if intersected by a boundary, are cut to form general convex polygons. An example is shown in Figure 4.21, in which two cells, labeled $L$ and $R$, are cut by a boundary.

The use of cut cells requires an addition to the basic quadtree data structure, a separate data structure which contains the indices of all nodes, cells, and edges surrounding each cut cell. Pointers to this new data structure are stored in each cut cell, and new cell center coordinates and cell volumes are computed. Cells that are outside the flow domain are removed from the top-level solution queue.

A quadtree grid with cut cells is created in the following manner:

1. Define all irregularly-shaped boundaries, either functionally or geometrically.

2. Generate a coarse grid over the entire physical domain, including irregularly-shaped boundaries.

3. Refine all quadtree cells that are intersected by a boundary. Repeat this process until a specified maximum grid level has been reached. The maximum grid level should be chosen so that complex boundaries are sufficiently resolved.
4. Remove any cells from the top-level solution queue that are completely outside the flow domain.

5. Create cut cells from those elements that are intersected by a boundary.

The flow solution method remains essentially unchanged with the addition of cut cells. For example, the flux across the edge between cells $L$ and $R$ in Figure 4.21 is computed using the values of flow variables at points $L_1$ and $R_1$. Since the values at these points are not known, they are approximated by a linear reconstruction from the values at their respective cell centers. This reconstruction results in additions to the source terms in the discretized equations similar to those described in Equations 4.33, 4.34, 4.61, and 4.64.

In the present case, an initial grid of size $4 \times 2$ cells was generated over the flow domain, from $x = -20D$ to $x = 20D$ and from $y = 0$ to $y = 20D$, where $D$ is the diameter of the cylinder. The cylinder was centered at $(x = 0, y = 0)$. A symmetrical boundary was set at the $y = 0$ axis. The grid was then refined to a maximum level of twelve at cells intersected by the cylinder. These cells were then cut to form the boundary of the cylinder. The solution procedure was then carried out, using an adaption criteria of dividing cells with a total velocity difference greater than 80% of the maximum total velocity difference. Cells were not allowed to be refined past the maximum grid level of twelve. The final grid, shown in Figure 4.22, contained 3,230 top-level cells. Figure 4.23 shows the grid in the vicinity of the cylinder. It can be seen that the grid is refined to the greatest extent in the area closest to the cylinder, in the recirculating region, and downstream of the cylinder.
Figure 4.22  Circular cylinder: adapted grid

Figure 4.23  Circular cylinder: close-up of adapted grid
Streamline contours of the resulting flow are shown in Figure 4.24. As the figure shows, and as was determined through a close examination of the solution, flow separation from the cylinder occurs at an angle of 128° from the leading edge of the cylinder, and reattachment occurs at a distance of 2.4D downstream of the cylinder. These results compare favorably to the separation value of 123° and reattachment distance of 2.4D obtained by Kwak [64].

The pressure coefficient on the surface of the cylinder can be seen in Figure 4.25, which also includes the experimental results of Thom [65] and the numerical results of Apelt [66] and Kawaguti [67]. The pressure coefficient, $C_p$, is defined as

$$C_p = \frac{p - p_{\infty}}{\frac{1}{2} \rho_{\infty} V_\infty^2}$$

Although Figure 4.25 shows a good general agreement between the results of the present analysis and previous results, it is clear that the present method produces "wiggles" in the flow solution in the vicinity of the cylinder.

Several possible explanations for this non-smooth flow solution were considered. First, it was thought that the extrapolation of cell center values to the nodes was at fault, but the cell center values were also non-smooth. The possibility of a decoupling between velocity and pressure was considered. This explanation was "ejected" because of the fact that no "checkerboard" pressure distribution existed anywhere in the flow or in any of the other flow solutions.

It was finally concluded that the non-smooth flow solution was caused by inaccurate flux calculations. It was believed that the method of calculating fluxes through the use of values at points perpendicular to the face, and calculating these values through the use of linear reconstruction, was somehow insufficient. This led to a search for a different flow solution method which would account for a skewness between cell centers and faces. This flow solution method, the upwind method, and its application to a quadtree grid is described in the next chapter.

Figure 4.24 Circular cylinder: streamline contours
Figure 4.25  Circular cylinder: pressure coefficient vs. azimuth angle
This chapter describes the flow solution procedure of Mathur and Murthy [38], and its implementation within the framework of a quadtree grid and data structure. This solution procedure is a pressure-based method for unstructured meshes that calculates convective fluxes through a second-order upwind method. Similar methods are described in References [47, 69, 70, 71, 72, 73, 74].

Theoretical Formulation

The discretization of the governing equations is described in this section. A scalar transport equation, including its convection, diffusion, and gradient terms, is first discretized, and used as a model for the discretization of the momentum equations. Next, a method for velocity-pressure coupling is discussed. A pressure-correction equation, which closes the system of equations, is then derived from the continuity equation. Finally, boundary conditions and their implementation in the numerical solution are presented.

Governing Equations

As in the previous chapter, the present method involves the solution of the steady, laminar, two-dimensional, incompressible Navier-Stokes equations (Equations 4.1-4.3).

Discretization of the Scalar Transport Equation

Again, a scalar transport equation, which represents a generic form of the momentum equations, is discretized and examined. The steady, two-dimensional, generalized transport equation for a scalar variable $\phi$ can be written as

$$\frac{\partial}{\partial x} (\rho u \phi) + \frac{\partial}{\partial y} (\rho v \phi) = \frac{\partial}{\partial x} \left( \Gamma \frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial y} \left( \Gamma \frac{\partial \phi}{\partial y} \right) + S_\phi$$ (5.1)

or, in divergence form, as

$$\nabla \cdot \rho \nabla \phi = \nabla \cdot \Gamma \nabla \phi + S_\phi$$ (5.2)
Integrating this equation about a control volume \( V \) gives
\[
\int_V \nabla \cdot \rho \mathbf{V} \, dV = \int_V \nabla \cdot \Gamma \mathbf{V} \, dV + \int_{S_3} S_3 \, dV \tag{5.3}
\]
By applying the Gauss divergence theorem, this equation can be written as
\[
\int_A \rho \mathbf{V} \cdot \mathbf{n} \, dA = \int_A \Gamma \mathbf{V} \cdot \mathbf{n} \, dA + \int_{S_3} S_3 \, dV \tag{5.4}
\]
Discretizing about each face \( f \) of the control volume gives
\[
\sum_f \rho \mathbf{V} \cdot \mathbf{A}_f = \sum_f \Gamma \mathbf{V} \cdot \mathbf{A}_f - S_3 \, V \tag{5.5}
\]
or
\[
\sum_f F_f \mathbf{o}_f = \sum_f (G_f + S_3 \, V)
\]
where
\[
F_f = \rho \mathbf{V} \cdot \mathbf{A}_f
\]
is the mass flow rate out of face \( f \) of control volume \( V \), and
\[
G_f = \Gamma \mathbf{V} \cdot \mathbf{A}_f
\]
is the transport due to diffusion through face \( f \). To write the discretized scalar transport equation in terms of the cell-center values of the scalar variable \( \mathbf{o} \), terms for convection, diffusion, and gradient must first be found.

**Convection Term** The convective term in the discretized scalar transport equation is the product of the mass flow rate, \( F_f \), out of face \( f \) of the control volume and the scalar value at the face. \( \mathbf{o}_f \). The mass flow rate is known from the solution of the continuity and momentum equations, which are described in the following sections. The value of \( \mathbf{o}_f \) is determined through the values of \( \mathbf{o} \) and its gradient at the upwind cell as
\[
\mathbf{o}_f = \mathbf{o}_{\text{upwind}} + \nabla \mathbf{o}_{\text{upwind}} \cdot \mathbf{d}\mathbf{r}
\]
where \( \mathbf{o}_{\text{upwind}} \) is the value of \( \mathbf{o} \) at the upwind cell, \( \nabla \mathbf{o}_{\text{upwind}} \) is the reconstruction gradient at the upwind cell, and \( \mathbf{d}\mathbf{r} \) is the vector from the centroid of the upwind cell to the centroid of the face.

Figure 5.1 shows two cells, \( L \) and \( R \), and the face 1-2 between them. If the unit vector \( n \) represents the direction of the face area vector \( \mathbf{A} \) (out of control volume \( L \)), then the convection of the scalar variable \( \mathbf{o} \) out of face 1-2 of control volume \( L \) (and into control volume \( R \)) can be written as
\[
F_f \mathbf{o}_f = \max(F_f, 0) \left( \mathbf{o}_L + \nabla \mathbf{o}_L \cdot \mathbf{d}\mathbf{r}_L \right) + \min(F_f, 0) \left( \mathbf{o}_R + \nabla \mathbf{o}_R \cdot \mathbf{d}\mathbf{r}_R \right)
= \max(F_f, 0) \mathbf{o}_L + \min(F_f, 0) \mathbf{o}_R + \max(F_f, 0) \nabla \mathbf{o}_L \cdot \mathbf{d}\mathbf{r}_L + \min(F_f, 0) \nabla \mathbf{o}_R \cdot \mathbf{d}\mathbf{r}_R
= \max(F_f, 0) \mathbf{o}_L + \min(F_f, 0) \mathbf{o}_R + S_{\text{source}}
\]
The values $\sigma_L$ and $\sigma_R$ are treated implicitly in the solution procedure, while the source term is treated explicitly.

**Diffusion Term** The diffusion at a face $f$ is defined as

$$G_f = \Gamma \nabla \sigma \cdot \vec{A}_f$$

This equation can be transformed from the physical coordinates $(x, y)$ to the general coordinates $(\xi, \eta)$. If the $(\xi, \eta)$ coordinates are defined to be in the $\epsilon$, and $t$ directions, respectively, as shown in Figure 5.1, the term $\nabla \sigma \cdot \vec{A}$ can be written

$$\nabla \sigma \cdot \vec{A} = \sigma_\xi (\xi_x \vec{A}_x + \xi_y \vec{A}_y) + \sigma_\eta (\eta_x \vec{A}_x + \eta_y \vec{A}_y)$$

where $(\vec{A}_x, \vec{A}_y)$ are the Cartesian components of the area vector $\vec{A}$. This equation can be written for the face 1-2 as

$$\nabla \sigma \cdot \vec{A} = \frac{\partial \sigma - \sigma_L}{\vec{A} \cdot \epsilon} \frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \epsilon} - \frac{\partial \sigma - \sigma_L}{\vec{A} \cdot \epsilon} \frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \epsilon}$$

The first term on the right hand side of this equation is the primary component of diffusion, and the second term is the secondary or cross-diffusion component. In order to eliminate the necessity to compute face tangents and nodal values, the secondary diffusion term is re-written as the difference between the total diffusion and the primary diffusion component. Therefore

$$G_f = \Gamma_f \frac{\partial \sigma - \sigma_L}{\vec{A} \cdot \epsilon} + \Gamma_f \left( \frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \epsilon} - \frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \epsilon} \right)$$
\[
\begin{align*}
= D_r (\sigma_R - \sigma_L) + S_{\sigma},
\end{align*}
\]
(5.14)

where \( \overline{\sigma} \) is the average of the gradients at the two adjacent cell centers. Again, the values \( \sigma_L \) and \( \sigma_R \) are treated implicitly, while the source term is treated explicitly.

**Gradient Terms**  The reconstruction gradient is written from the Green-Gauss theorem as

\[
\nabla \sigma_r = \frac{\alpha}{l} \sum_{f} (\sigma_f A_f)
\]
where

\[
\frac{\sigma_f}{l} = \frac{\sigma_L - \sigma_R}{2}
\]

is the average of the values of \( \sigma \) at neighboring cells. The term \( \alpha \) is a limiter and is used to prevent the reconstruction from introducing local extrema. In the present study, the limiter of Venkatakrishnan [73] is used. The value of \( \sigma \) at a face \( f \) of cell \( c \) can then be written as

\[
\sigma_f = \sigma_c + \nabla \sigma_c \cdot \vec{r}
\]
(5.17)

The cell derivatives of the secondary diffusion terms are computed as

\[
\nabla \overline{\sigma} = \frac{1}{l} \sum_{f} (\overline{\sigma_f} A_f)
\]
(5.18)

where

\[
\overline{\sigma_f} = \frac{\sigma_{fL} - \sigma_{fR}}{2}
\]
(5.19)

is the average of the reconstructed values of \( \sigma \) at face \( f \) from both neighboring cells.

**Discretized Equation**  After collecting and regrouping terms, the discretized scalar transport equation for a control volume \( V_P \) surrounding point \( P \) can now be written as

\[
apu_P = \sum a_{nb} \sigma_{nb} + S_{\sigma} \nabla P
\]
(5.20)

where

\[
ap = \sum a_{nb} - S_P \nabla P
\]
(5.21)

and the subscript \( nb \) represents the neighboring control volume cell-centers.

**Discretized Momentum Equations**

The momentum equations, written in the preceding form, can now be expressed as

\[
apu_P = \sum_{nb} a_{nb} u_{nb} + S_u \nabla P - \sum_f p_f A_f
\]
(5.22)

\[
apv_P = \sum_{nb} a_{nb} v_{nb} + S_v \nabla P - \sum_f p_f A_{g_f}
\]
(5.23)

where the pressure gradient terms have been written separately from the discretized source terms.
Velocity-Pressure Coupling

The "checkerboard" pressure distribution problem is again alleviated by a Rhie and Chow \[39\] interpolation. The mass flow rate at a face \( f \) is written

\[
F_f' = \rho A \left( \frac{\bar{V}_f^+ - \bar{V}_f^-}{2} \right) - \rho \frac{V_f - V_f^-}{\Delta P + \Delta P^+} \left( \frac{P_f - P_f^-}{\Delta P} - \frac{\overline{P_f'} - \overline{P}}{\Delta P} \right) \frac{\Delta P}{\Delta P^+},
\]

where \( \bar{V}_f^+ \) is the velocity that results from the solution of the momentum equations, \( \Delta P \) is the average of the \( a_P \) coefficients of each of the momentum equations \( (a_P, a'_{pl}) \), and \( \overline{P_f'} \) is the averaged pressure gradient from each neighboring cell.

Pressure-Correction Equation

To close the system of equations, a pressure-correction equation is extracted from the continuity and momentum equations. The continuity equation, once integrated and discretized, can be written as

\[
\sum_j (F_j^* - F_j) = 0
\]

where \( F_j^* \) is the mass flow rate computed from Equation 5.24, and \( F_j \) is a mass flow rate correction defined as

\[
F_j' = -\rho \frac{V_f - V_f^-}{a_P + a_{P_n}} \frac{P_f' - P_f^-}{ds} \frac{\Delta P}{\Delta P^+}.
\]

Substituting Equation 5.26 into Equation 5.25 gives the following discretized pressure-correction equation

\[
a_P p' = \sum_n a_{n,s} p_{n,s} + b_p,
\]

where

\[
b_p = -\sum_j F_j^*
\]

is the total mass inflow into the cell. Once the discretized equation is solved for the pressure-corrections, the mass flow rate is corrected as

\[
F_f = F_f^* + F_f'
\]

This corrected mass flow rate is used to compute the convection term, given in Equation 5.10, of the coefficients of the momentum equations in the next iteration of the solution procedure. The pressure is corrected as

\[
p_P = p_P^* + a_P p'_P
\]
where \( \alpha_f \) is the pressure underrelaxation factor. Cell velocities are corrected as

\[
U^*_P = U_P^* - \frac{\sum_I P'_I \cdot \ell_I}{a_P^*}
\]

\[
V^*_P = V_P^* - \frac{\sum_I \ell'_I \cdot \ell_I}{a_P^*}
\]

where the pressure-correction at a face, \( P'_I \), is determined by averaging the pressure-correction values of the neighboring cells of face \( f \).

**Boundary Conditions**

The implementation of boundary conditions in the upwind method is similar to the process described for the power-law method, but is applicable to boundary control volumes composed of general convex polygons. Figure 5.2 shows a general boundary control volume, \( V_P \), with cell center point \( P \), boundary edge midpoint \( b \), unit normal \( \ell_b \), and unit normal \( n \) in the direction of area vector \( \ell \).

The diffusive flux at the boundary edge \( h \) can be written as the sum of a primary component and a secondary component, which is the difference between the total diffusion and the primary component.

\[
\begin{align*}
\eta_h &= \Gamma_b \frac{\partial \phi_b - \phi_P}{db} \frac{\ell}{\ell \cdot r_h} + \Gamma_b \left( \nabla \phi_P \cdot \ell - \nabla \phi_P \cdot r_h \frac{\ell}{\ell \cdot r_h} \right) \\
&= D_b (\phi_b - \phi_P) + S_{\text{diff}}
\end{align*}
\]

The discretized scalar transport equation is written as

\[
ap \partial \phi = \sum a_{nb} \partial_{nb} + S \nabla \phi
\]

where

\[
ap = \sum a_{nb} - S \nabla \phi
\]

![Figure 5.2 Boundary control volume](image)
All boundary conditions are incorporated into the linearized source terms as

\[ S_i = S_i^o + iD_i - F_{i,0}x_i - F_{i,0}p + S_i^b, \]

and

\[ S_p = S_p^o - D_p \]

where \( S_i^o \) and \( S_p^o \) are parts of the source terms arising from other sources or boundaries.

At inflow and wall boundaries, values of velocity can be substituted directly into the preceding source term equations. At symmetry boundaries, flow gradients are zero, and no additions to the source terms \( S_i^o \) and \( S_p^o \) are needed.

Since mass flow rates at each of the preceding boundary types are known, they can be substituted directly into the source term of the pressure-correction equation. Pressure-corrections and pressures at boundary edges, necessary for velocity corrections and gradient computations, can be computed as

\[ p_i' = p_i \]

and

\[ p_i = p_P - \nabla p_P \cdot \hat{r}_b \]

At outflow boundaries, flow velocities are not known. Therefore, the present method assumes an outflow pressure equal to a specified freestream pressure

\[ p_b = p_\infty \]

and employs the pressure boundary condition method of Mathur and Murthy [73]. An intermediate value of mass flow rate at an outflow boundary can be expressed as

\[ F_i^b = \rho_bA_b \cdot V_i^b = \frac{\rho_bV_P}{a_P} \left( \frac{p_b - p_P}{db} - \nabla p_P \cdot \hat{r}_b \right) \frac{A \cdot A}{A \cdot r_b} \]

The discretized continuity equation is written as

\[ \sum F_f + F_b = 0 \]

where the summation is over non-boundary faces. A boundary mass flow rate correction can be defined as

\[ F_b' = -\frac{\rho_bV_P}{a_P} \left( \frac{p_b - p_P}{db} \right) \frac{A \cdot A}{A \cdot r_b} \]

The boundary mass flow rate is then corrected by

\[ F_b = F_b^o + F_b' \]
The pressure-correction equation is then written as

\[ \alpha P P' = \sum \alpha_{nP} P'_{n} - \alpha P + b. \]  

(5.43)

Since the pressure at the outflow boundary is specified, the pressure-correction is zero

\[ p'_{f} = 0 \]  

(5.46)

The normal and tangential velocity components are computed from known values of mass flow rates by

\[ v_{n} = \frac{F_{k}}{\rho_{k} A_{b}} \]  

(5.47)

and

\[ v_{t} = v_{p} - \frac{v_{p} \cdot \frac{A_{1}}{A_{2}}}{A_{2}} \]  

(5.48)

These velocity components are then easily converted to Cartesian velocity components.

**Solution Procedure**

The solution procedure used with the upwind method is the SIMPLE algorithm of Patankar [52], used within a grid adaption procedure. Although the SIMPLE method generally converges at a slower rate than the SIMPLER method, it is chosen to avoid interpolation of the pseudovelocities to the cell faces. The solution procedure can be summarized as follows:

1. Generate an initial coarse grid.
2. Refine the grid, if necessary, in areas that are known a priori to require a high degree of grid clustering.
3. Apply the SIMPLE algorithm as follows:
   
   (a) Guess a pressure field \( p^* \). If a coarse grid solution is known, retain previously determined pressures at cells that have not been subdivided, and use a parent cell’s pressure as an initial guess for newly formed children cells. 
   (b) Calculate the coefficients of the momentum equations and solve them to obtain \( u^* \) and \( v^* \).
   (c) Solve the pressure-correction equation to find \( p' \).
   (d) Correct the mass flow rate at cell faces.
   (e) Calculate \( p \) by adding \( p' \) to \( p^* \).
(f) Correct the velocity field with the velocity-correction equations.

(g) Return to Step (b) and repeat until convergence.

4. If the maximum grid level has not been reached, refine the grid according to the adaption criteria, and return to Step 3.

The discretized momentum and pressure-correction equations are again solved by a symmetric Gauss-Seidel point iterative method with under-relaxation.

Results

The upwind method was used in a variety of test cases to analyze its utility. These tests include a lid-driven cavity flow, a backward-facing step flow, and the flow over a circular cylinder.

Lid-driven Cavity Flow

The upwind method was first applied to the lid-driven cavity flow problem of the previous chapter at a Reynolds number of 400 on the grid shown in Figure 4.8. The results were compared to the grid-independent solution of a 128 × 128 structured grid, as well as to those obtained by the previous power-law method. As in the previous chapter, the flow was first solved on an initial grid containing 16 × 16 cells, and grid refinement locations were set explicitly, to give a final grid containing 9,472 top-level cells.

Figures 5.3-5.4 show u-velocity profiles at different x cross-sections of the flow. It can be seen that the results obtained by the upwind method more closely match the benchmark solution of the 128 × 128 structured grid than do the results of the power-law method.

Figure 5.5 shows the total velocity RMS convergence for the solutions obtained with each method. As expected, the upwind solution, which used the SIMPLE flow solution algorithm, converges at a slower rate than the power-law solution, which used the SIMPLER algorithm. This slower convergence can also be explained by the fact that the second-order upwind method achieves its accuracy by determining the reconstruction gradient terms explicitly, which results in a formulation that is written as a first-order upwind method plus a correction. Nevertheless, both quadtree grid methods converge at a faster rate than does the 128 × 128 structured grid solution.
Figure 5.3 Driven cavity. $Re = 400$: $u$ velocity profiles, $x = 1/2$

Figure 5.4 Driven cavity. $Re = 400$: $u$ velocity profiles, $x = 1/64$
Backward-Facing Step Flow

The next test case considered was a backward-facing step flow, as shown in Figure 5.6. The inflow boundary was set to a parabolic flow condition, while the outflow boundary conditions were determined through a prescribed outlet pressure. In this situation, the outflow boundary length \( L \) is dependent on the Reynolds number to allow a fully developed flow at the outflow boundary and was therefore changed for each case studied; the flow was computed at Reynolds numbers of 100 and 389, which were based on the average inflow velocity and twice the inflow duct height \( 2h \). All other boundaries were set to no-slip wall conditions. Results were compared to the experimental results of Armaly et al. [76] and the numerical results of Coirier [43].

For the \( Re = 100 \) test case, the outflow boundary was set to a length of \( L = 80 \text{mm} \) or, in terms of step height, \( 16.35 \). The flow was then solved on an initial coarse grid containing 900 cells, and the solution was carried out through four levels of adaption. The adaption criteria that was used was identical to that of Coirier [43], which is given in Equations 3.2-3.7. The final grid, a section of which is shown in Figure 5.7, contained 10,032 top-level cells.

Figures 5.8-5.11 show \( u \) velocity profiles predicted by the present method as compared with previous
\[ u = u(y) \]
\[ v = 0 \]
\[ h = 5.2 \text{ mm} \]
\[ S = 4.9 \text{ mm} \]
\[ H = 10.1 \text{ mm} \]

Figure 5.6 Backward-facing step: schematic diagram

Figure 5.7 Backward-facing step. \( Re = 100 \) adapted grid
results. It can be seen that the present method agrees well with previously published results.

The outflow boundary for the $Re = 389$ test case was set to a length of $L = 300\text{mm}$ or $61.2S$. The initial grid consisted of 1,600 cells, and cells past $x = 100\text{mm}$ were of an increasing aspect ratio. The flow was again solved through four levels of adaption, using the adaption criteria of Coiner [43]. The final grid, part of which is shown in Figure 5.12, contained 20,251 top-level cells.

Velocity profiles of the present method were again compared to those of previous results, as shown in Figures 5.13-5.16. These results compare favorably to the experimental results of Armaly et al. [76], but because of convergence problems the numerical results of Coiner [43] were obtained after only two levels of adaption for the $Re = 389$ case, which may account for some of the discrepancy between the previous experimental and numerical results.

For each different Reynolds number flow, the reattachment length of the primary separation region was found through close inspection of the streamline contours and velocity vectors of the present results. Figure 5.17 shows the reattachment length versus Reynolds number for the present numerical results and the experimental results of Armaly [76]. It can be seen that the results of the present method match closely with the experimental results.

Figure 5.8  Backward-facing step. $Re = 100$: $u$ velocity profiles. $x/S = 0.0$.
Figure 5.9 Backward-facing step, $Re = 100$: $u$ velocity profiles. $x/S = 2.55$

Figure 5.10 Backward-facing step, $Re = 100$: $u$ velocity profiles. $x/S = 3.06$
Figure 5.11 Backward-facing step. $Re = 100$: $u$ velocity profiles. $x/S = 4.18$

Figure 5.12 Backward-facing step. $Re = 389$: adapted grid
Figure 5.13  Backward-facing step. $Re = 389$: $u$ velocity profiles. $x/S = 0.0$

Figure 5.14  Backward-facing step. $Re = 389$: $u$ velocity profiles. $x/S = 2.55$
Present results
• Armaly et al. (experimental)
□ Coirier (numerical)

Figure 5.15 Backward-facing step. $Re = 389$: $u$ velocity profiles. $x/S = 3.06$

Figure 5.16 Backward-facing step. $Re = 389$: $u$ velocity profiles. $x/S = 4.18$
Flow over a Circular Cylinder

Again, the usefulness of a quadtree grid as applied to a flow with geometrically complex boundaries was tested on the flow over a circular cylinder. As in the previous chapter, a flow with a Reynolds number of forty was chosen. The flow domain ranged from \( x = -20D \) to \( x = 20D \) and from \( y = 0 \) to \( y = 20D \), with the cylinder centered at \( (x = 0, y = 0) \) and a symmetry boundary at \( y = 0 \).

The initial grid containing \( 4 \times 2 \) cells was again refined to a maximum level of twelve at cells intersected by the cylinder. A grid adaption criteria of dividing cells with a total velocity difference greater than 80\% of the maximum total velocity difference was employed within the solution procedure. The final grid, shown in Figures 5.18 and 5.19, contained 3,962 top-level cells.

Figure 5.20 shows the streamline contours of the flow, which indicate the recirculating region behind the cylinder. It was determined through examination of the streamline contours and velocity vectors that the flow separates from the cylinder at an azimuth angle of 128°, and that reattachment occurs at 2.5D. These results are similar to those of Kwak [64], who predicted separation at 123° and reattachment at 2.4D.

The pressure coefficient on the surface of the cylinder can be seen in Figure 5.21. Like the power-
Figure 5.18  Circular cylinder: adapted grid

Figure 5.19  Circular cylinder: close-up of adapted grid
law method, these results show a general agreement with previously published results. However, the "wiggles" in the solution have not been eliminated through the use of the upwind method.

These results indicated that the conclusion of the previous chapter was incorrect, and that the oscillations in the flow solution were not caused by inaccurate flux computations in the uni-dimensional power-law method. Instead, it was hypothesized that the use of cut cells in viscous regions is inadequate in predicting accurate flow solutions, a conclusion verified by the results of Coirier [43] who also found solution oscillations in viscous regions near cut cells. It was determined that the geometry of a grid with cut cells, in which adjoining cells can differ greatly in size and shape, was the source of the errors in the flow solution. Therefore, it was decided that in order to predict accurate flow solutions in highly-viscous regions, either a higher-order flow solution method or a different type of grid must be used.

Hybrid Grid

To solve the problem of the flow solution oscillations in viscous regions, it was decided to make use of a mixed-element hybrid grid for the solution of the flow around the circular cylinder. In this method, the domain around the cylinder is divided into three regions, each containing a different cell type: a viscous region near the surface of the cylinder consisting of unstructured quadrilateral cells, the inviscid flow domain furthest from the body composed of a quadtree grid, and a transition region between these areas containing a triangular grid.

Since the upwind method can be applied to any grid composed of convex polygons, no change in the flow solution procedure was required. However, the use of different cell types necessitated a modification of the grid's data structure. Each cell's type was stored in an indexed array of integers. Grid adaption was carried out differently for each cell type, and care was taken to avoid creating hanging-nodes on the edges of triangular cells.

The viscous grid near the cylinder contained 1,200 quadrilateral cells; 60 cells around the surface of
the half-cylinder by 20 cells extending radially from the surface of the cylinder to a distance of 0.6D. The initial quadtree grid in the inviscid region contained $16 \times 8$ cells, which ranged from $x = -20D$ to $x = 20D$ and from $y = 0$ to $y = 20D$. This grid was refined to a level of seven at cells that were intersected by the viscous grid. Quadtree cells that were intersected by the viscous grid or that were inside the cylinder were removed from the top-level queue. The triangular grid was constructed with a Delaunay triangulation algorithm in the area between the viscous grid and the quadtree grid.

The flow solution and grid adaption algorithm was carried out through three cycles. A grid adaption criteria of dividing cells with a total velocity difference greater than 20% of the maximum total velocity difference was used. The quadtree grid was not allowed to be refined past a level of seven, while the viscous grid and triangular grid were not refined past a level of three. The final grid, a section of which is shown in Figure 5.22, contained 2,266 top-level cells.

Streamline contours of the flow can be seen in Figure 5.23. It was determined that the flow separates from the cylinder at an azimuth angle of $126^\circ$ and reattachment occurs at $2.5D$, results which compare favorably to those of Kwak [64].

The pressure coefficient on the surface of the cylinder is shown in Figure 5.24. It can be seen that the hybrid grid produces a smooth solution that agrees well with previous results.
Figure 5.22  Circular cylinder: hybrid grid

Figure 5.23  Hybrid grid circular cylinder: streamline contours
Figure 5.24 Hybrid grid circular cylinder: pressure coefficient vs. azimuth angle
In order to show the present method's usefulness in finding flow solutions for physically practical problems, this chapter describes the application of the upwind method to a three-dimensional octree grid. The octree grid and data structure are first described, followed by a discussion of the extension of the upwind method to three dimensions. Finally, the results of two test cases are presented.

The Octree Grid

An octree data structure, like the quadtree, is composed of a hierarchical relationship based on a recursive subdivision of space. An octree grid consists of hexahedral cells that may be successively subdivided into eight smaller hexahedra, as shown in Figure 6.1. The connectivity between cells is stored in arrays of pointers from each cell to its parent and children cells, pointers from cells to their surrounding nodes and faces, pointers from faces to their surrounding nodes and adjacent cells, and the coordinates of each node. Edges (one-dimensional lines connecting the nodes of a two-dimensional face) need not be computed or stored in the present formulation.

Flow Solution Method

The flow solution method for three dimensions remains essentially unchanged from that of two dimensions. The incompressible Navier-Stokes equations are written in three dimensions as:

continuity equation:
\[ \nabla \cdot (\rho \vec{V}) = 0 \] (6.1)

momentum equation:
\[ \nabla \cdot (\rho \vec{V} \vec{V}) = -\nabla p + \nabla \cdot (\mu \nabla \vec{V}) \] (6.2)
The scalar transport equation, once integrated and discretized, is represented in the same fashion as in two dimensions as

$$\sum F_f \phi_f = \sum G_f + S_c V$$  \hspace{1cm} (6.3)

where

$$F_f = (\rho \vec{V} \cdot \vec{A})_f$$  \hspace{1cm} (6.4)

and

$$G_f = (\Gamma \nabla \phi \cdot \vec{A})_f$$  \hspace{1cm} (6.5)

However, the summations are applied to each two-dimensional face $f$ of a control volume $V$. Again, the convective term is computed as the product of the mass flow rate out of a face $f$ and the value of $\phi$ at $f$ determined through a reconstruction from the upwind cell. The diffusive term is again written as the sum of a primary component and a secondary component determined as the difference between the total diffusion and the primary diffusion component.Collecting and regrouping terms gives the discretized scalar transport equation

$$a_p \phi_p = \sum a_{nb} \phi_{nb} + S_c V p$$  \hspace{1cm} (6.6)

where

$$a_p = \sum a_{nb} - S p V p$$  \hspace{1cm} (6.7)

The momentum equations, written in the preceding form, can now be expressed as

$$a_p u_p = \sum a_{nb} u_{nb} + S_c V p - \sum \rho_f A_{zf}$$  \hspace{1cm} (6.8)
Velocity-pressure coupling, mass flow rate correction, and pressure-correction are determined as in two dimensions. Cell velocities are corrected as

\[
\begin{align*}
up &= \frac{\sum f' r' A_{r'}}{a_p^2} \\
v_p &= \frac{\sum f' r' A_{r'}}{a_p^2} \\
w_p &= \frac{\sum f' r' A_{r'}}{a_p^2}
\end{align*}
\]

The SIMPLE algorithm is used as the flow solution procedure.

Results

The present method was applied to two test cases: a three-dimensional lid-driven cavity flow, and a 90 degree curved duct flow.

3-D Driven Cavity Flow

The first three-dimensional test case considered is a 3-D driven cavity at a Reynolds number of 100. The cavity had dimensions of unity in each coordinate direction, with a lid moving at unit velocity in the x direction at \( y = 1.0 \). Since the flow is symmetric about the \( z = 0.5 \) plane, the solution was carried out on only half of the cavity, with the appropriate symmetry boundary condition. The flow was solved on two different octree grids, each with two levels of adaption and an initial grid consisting of \( 16 \times 16 \times 8 \) cells. For grid 1, each cell was subdivided to give a final grid of 16,384 top-level cells, as shown in Figure 6.2. For grid 2, shown in Figure 6.3, only those cells within a distance of one-eighth of the cavity's width to a zero-velocity wall were divided, giving a final grid containing 9,328 top-level cells.

Velocity vectors for two cross-sections of the flow on each grid can be seen in Figures 6.4 and 6.5. Each figure shows velocity vectors in the \( z = 0.5 \) plane of symmetry (a) which indicate the primary flow movement in the cavity. The three-dimensional movement of the flow is shown by the velocity vectors in the \( x = 0.5 \) plane (b). It should be noted that while the vector lengths are scaled according to velocity magnitude, lengths are scaled differently between parts a) and b) of each figure.
Figure 6.3  3-D driven cavity: grid 2
Figure 6.4 3-D driven cavity, grid 1: velocity vectors. a) \( z = 0.5 \) plane, b) \( z = 0.5 \) plane

Figure 6.6 compares the \( u \)-velocity profiles at the \( x = 0.5, z = 0.5 \) centerline of each grid with the results of Mathur and Murthy [68], who used a grid composed of 19,800 hexahedral cells. It can be seen that the results show good agreement. It should also be noted that the results obtained on the second grid compare favorably with the results of the first, even though approximately 40% fewer cells were used.

90 Degree Curved Duct Flow

The second test case considered is the flow through a 90 degree curved duct with a square cross-section. A schematic diagram of a cross-section of the duct in the \( x - y \) plane is shown in Figure 6.7. It can be seen that the duct is composed of an inlet section 160mm long, a curved section with an inside radius of 72mm, and an outlet section 320mm long. The duct is 40mm wide and 40mm deep along its entire length. The inflow boundary contains a uniform flow in the \( x \) direction, while the outflow boundary contains a constant pressure. All other boundaries are composed of no-slip walls. The flow Reynolds number is 792, and is based on the average inflow velocity, \( u_m \), and the hydraulic diameter.
Figure 6.5 3-D driven cavity: grid 2, velocity vectors. a) \( z = 0.5 \) plane, b) \( x = 0.5 \) plane

Figure 6.6 3-D driven cavity: \( u \) velocity profiles, \( x = 0.5, z = 0.5 \)
D. Since the inflow velocity is uniform

$$u_{ir} = u_{\infty}$$  (6.14)

and

$$D = \sqrt{\frac{m}{\rho_{\infty} u_{ir}}}$$  (6.15)

where

$$m = \rho_{\infty} u_{ir} A_{ir}$$  (6.16)

The numerical simulation was carried out through two levels of adaption. The flow was first solved on a grid containing \((62 \times 10 \times 5)\) cells. Since the flow is symmetric about the \(z = 20\text{mm}\) plane, the appropriate symmetry boundary conditions were used, and the flow was computed for half the number of cells in the \(z\) direction than in the cross-flow direction. The grid was refined according to the adaption criteria of Courier [43] to give a final grid containing 11,136 top-level cells as shown in Figure 6.8. The flow was then computed to convergence.

Results of the present method are compared to the experimental results of Taylor et al. [77] and the numerical results of Tannamidis and Assanis [78]. Velocity vectors in the \(z = 20\text{mm}\) plane of symmetry are shown in Figure 6.9, while velocity vectors in the \(y = 112\text{mm}\) plane (the boundary between the curved and outlet sections) are shown in Figure 6.10. These results are qualitatively similar to those of References [77] and [78].
Figure 6.7 90 degree curved duct: schematic diagram
Figure 6.8  90 degree curved duct: adapted grid
Figure 6.9  90 degree curved duct: velocity vectors, $z = 20\text{mm}$ plane
Figure 6.10  90 degree curved duct: velocity vectors. $y = 112\, mm$ plane
7 CONCLUSIONS AND RECOMMENDATIONS

In this study, a new, adaptively-refined quadtree grid method for the incompressible Navier-Stokes equations has been successfully formulated and presented. The quadtree grid and its corresponding data structure were described, and their implementation within solution adaptive flow solution schemes discussed. Two pressure-based flow solution methods were presented, including each method's theoretical formulation, solution algorithm, and results obtained from several test cases.

The first flow solution method discussed was the uni-dimensional power-law method. This method computed the total flux across a face between two cells through a power-law approximation using flow values at points perpendicular to the face. The flow values at these points were approximated through a linear reconstruction from values at cell-centers, which were stored in memory. This reconstruction resulted in additional source terms in the discretized equations. Flow gradients were found through a linear least-squares procedure. Pressure-velocity coupling was achieved through the Rhie and Chow interpolation procedure. The system of equations was closed by extracting a pressure-correction equation and a pressure equation from the continuity and momentum equations. The SIMPLER algorithm was used as the flow solution procedure. Flow solution accuracy was increased through solution adaptive grid refinement.

The power-law method was applied to four test cases. The first three of these cases, a lid-driven cavity flow, a duct flow with sudden contraction, and a channel flow with two right-angle bends, used a Cartesian quadtree grid throughout the flow domain, including boundaries. These tests cases produced results that were as accurate as previous results, while using fewer grid points. The fourth test case, the flow over a circular cylinder, used a Cartesian quadtree grid except near non-Cartesian boundaries, where "cut cells" were used. The introduction of cut cells produced oscillations in the flow solution near the cylinder, which lead to the search for an alternate flow solution method.

The second flow solution method described in this study was the second-order upwind method of Mathur and Murthy [68]. This method approximated the convective flux across a face between two cells through a linear reconstruction of flow values at the upwind cell. The diffusive flux was determined
through a transformation to coordinates that were parallel to a line connecting the cell centers and tangent to the cell face. This flux could then be expressed as the sum of a primary flux component and a secondary component. The secondary or cross-diffusion component was expressed as the difference between the total diffusion and the primary component. The reconstruction gradients based on a Green-Gauss formulation were prevented from introducing extrema by using the Venkatakrishnan limiter. Flow gradients used in the secondary diffusion terms were computed by applying the Green-Gauss approximation over the average reconstructed flow values at cell faces. The Rhie and Chow interpolation procedure was used to avoid any “checkerboard” instabilities. A pressure-correction equation was derived from the continuity and momentum equations to close the system of equations. The system of equations was solved through the use of the SIMPLE algorithm. Solution adaptive grid refinement was used to increase flow solution accuracy.

The upwind method was first applied to a lid-driven cavity flow, and produced results more accurate than the power-law method, but at a slower convergence rate due to the use of the SIMPLE algorithm and a higher-order scheme. The flow over a backward-facing step was then analyzed with the upwind method and produced results that showed good agreement with previously published results. Finally, the flow over a circular cylinder was computed, using cut cells to form the cylinder’s boundary. It was found that the upwind method did not eliminate the oscillations in the flow solution near the cut-cell boundary. This led to the use of a mixed-element hybrid grid combined with the upwind method to predict the flow over the cylinder. The hybrid grid was composed of unstructured quadrilateral cells in the viscous region, a Cartesian quadtree grid in inviscid regions, and triangular cells at the grid interface. This process succeeded in eliminating the oscillations in the flow solution.

To predict fluid flow in three-dimensional geometries, the upwind method was used in conjunction with an octree grid and data structure. The octree grid was composed of hexahedral cells which were successively subdivided into eight smaller hexahedra. The results of two test cases, a 3-D driven cavity flow and the flow through 90 degree curved duct, showed good agreement with previous results.

Several extensions to the present method can be made, each of which may result in improvements in different areas. First and foremost, a detailed study of the effectiveness of different adaption criteria should be performed. In the present study, each test case used an adaption criterion that best captured the flowfield of the particular test case. It would be desirable to possess a single criterion that could be used for a wide variety of flow configurations.

To improve the convergence properties of the present method, two recommendations are made. A sparse matrix linear systems solver would enhance the solution of the discretized equations over the
point Gauss-Seidel method now employed. A sparse matrix solver would allow for faster propagation of boundary information to internal control volumes. Multigrid methods have been proven to greatly accelerate the convergence of the solution of an elliptic problem [70]. With the present method's inherent hierarchical relationship between cells of different levels in the quadtree grid and its use of the non-staggered grid arrangement, an extension to a multigrid formulation seems logical.

The incorporation of unsteady flow solution techniques would enhance the stability of the present method and allow for the solution of time-dependent flows. However, since high-gradient regions of a flow may develop in time into low-gradient regions which require fewer grid points, the extension of the present method to unsteady flows would require the addition of a cell derefinement procedure.

The ability to solve high Reynolds number flows would be essential to prove the effectiveness of the present method. An attempt was made to solve a high Re test case for inclusion in this study, with only modest success. This test case consisted of the flow over a NACA-0012 airfoil at a Reynolds number of $2.88 \times 10^6$. The flow was solved on a body-fitted, O-type grid. It was found that the addition of unsteady terms in the upwind method was necessary to prevent the solution from diverging. A converged solution was found for very low angles of attack on a relatively coarse grid of size 101 points on the airfoil surface by 33 points in the radial direction. This grid produced an accurate pressure distribution on the airfoil, but was insufficiently resolved to capture the boundary layer and predict accurate shear-stress values on the airfoil. All attempts to find a solution at a higher angle of attack, using a finer grid, or using grid adaption, resulted in non-converging solutions. It is believed, however, that these problems could be remedied with a modest amount of effort.

Pressure-based methods have been used in the past to solve the compressible Navier-Stokes equations [80], [81]. The incorporation of these techniques into the present method would result in a solution procedure for flows at all speeds.

Finally, it is believed that the implementation of the present method would benefit from the use of an object-oriented computer language with dynamic memory allocation, such as C++ or FORTRAN 90, to compute and store the quadtree data structure. The use of FORTRAN 77 in the present method required that the data structure was represented as a number of arrays of primitive data types. An object-oriented language would allow the data structure to be represented in abstract data types and greatly decrease the difficulties associated with grid adaption. Object-oriented routines that compute data structure could be combined with FORTRAN 77 libraries that have been optimized for the solution of a system of equations. Also, in the present method parent cells unnecessarily stored certain flow variable and geometric values. A language with dynamic memory would not require that these values
be stored, thus reducing the amount of memory needed.

This study has shown that an adaptively-refined quadtree grid, used in conjunction with a pressure-based flow solution method for the incompressible Navier-Stokes equations, accurately and efficiently predicts solutions for a variety of fluid flow problems. Several test cases have shown that the present method effectively captures features of the flow through localized grid refinement, thus increasing solution accuracy and requiring less computational time. It is clear that more research in this direction would be beneficial.
BIBLIOGRAPHY


