To the Graduate Council:

I am submitting herewith a dissertation written by Guoping Xia entitled "Anisotropic Adaptation on Unstructured Grids." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

Charles L. Merkle

Major Professor

We have read this dissertation
and recommend its acceptance:

John E. Caruthers

K.C. Reddy

John S. Steinhoff

Accepted for the Council:

Anne Mayhew

Vice Provost and Dean of
Graduate Studies

(Original Signatures are on file with official student records.)

# Anisotropic Adaptation on Unstructured Grids

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Guoping Xia

May 2003

# Acknowledgements

I sincerely acknowledge the long-time love and support from my parents in all the endeavors I have undertaken.

My foremost gratitude goes to my advisor, Prof. Charles Merkle, who has always provided me with academic advice and continual encouragement, during the past five years' study and research at UTSI.

I would like to thank my committee members Dr. John Caruthers, Dr. K.C. Reddy and Dr. John Steinhoff for their suggestions in improving my dissertation. Sincere thanks are extended to Dr. Ding Li for his valuable suggestions on my research and Mrs. Brenda Brown for her help during my stay in UTSI.

I would also like to express the appreciation to the friends who help me in UTSI. These include Dr. Meng Fan, Shenghong Qiu, Wei Su, Dr. Yuxing Sun, Kan-Wai Tong, Lesong Wang, Min Xiao, Xiaoqiang Zeng, Dr. Fanglin Zhu and Lin Zhu.

No words can express my heartfelt gratitude to my beloved wife Xiaomei Wang for her constant love and encouragement, without which nothing could have never been achieved.

# Abstract

The objective of the present research is to assess the use of grid adaptation to improve Computational Fluid Dynamics calculations. Many issues of the quality discretization of a flow domain are discussed. The representation of the highly directional features in a flow field, such as shocks and boundary layers, forms the focus of the analysis. Anisotropic adaptation, which uses stretched elements to resolve directional features, is more effective than isotropic adaptation. Anisotropic adaptation requires more degrees of freedom from the mesh and demands the use of unstructured grids in the adaptation.

The size and orientation of an anisotropic element require a matrix-like local feature indicator. The Hessian, a matrix composed of the second derivatives of an appropriate flow variable, is defined and used as a feature indicator in the adaptation. The Hessian provides a metric that defines the length of an edge and the lengths of all edges are equal in the optimized mesh. The techniques to minimize the differences among edge lengths are discussed and those chosen include node enrichment, node removal, edge swapping and point smoothing. A unified procedure based on the advancing front method is implemented to reconstruct the local connectivity that has been removed in the node removal and edge swapping processes.

The results indicate that the mesh in which the edge lengths are equalized is not correct for three major flow features one frequently encounters. The inflections existing near the wall in a boundary layer result in coarse grids there. A “wall” Hessian is defined to replace the second derivatives and give a more appropriate spacing for high Reynolds number flow modeling. Difficulties in the adaptation of discontinuities are addressed. These include the infinite refinement that tries to pull all the points close to the discontinuity and the deviation of the shock from the refinement region because it is too thin. Remedies proposed are to limit the minimum physical edge length and smooth the Hessian such that the refinement encompasses more layers of elements. The strength of a discontinuity is defined and methodology to refine the discontinuity equally is proposed. The invalidity of the Hessian in a free stream is corrected to give a reasonable grid size in that region. It is demonstrated that these suggested modifications improve the overall quality of the adapted mesh as well as the solution.

The concepts involved in the extension of the length-based approach to three dimensions are addressed. The difference and difficulties in three-dimensional adaptation are discussed. Barriers exist which prevent the equidistribution of the edge lengths, the goal of the current approach.

# Contents


**1 Introduction** 1

1.1 Mesh Generation Methods 6

1.1.1 Octree Method 6

1.1.2 Delaunay Method 7

1.1.3 Advancing-Front Method 9

1.1.4 Extensions to Anisotropic Grid Generation 12

1.2 Solution Adaptation Techniques 13

1.2.1 *p*-Method 14

1.2.2 *r*-Method 15

1.2.3 *h*-Method 16

1.2.4 Remeshing Method 16

1.2.5 Combined Methods 17

1.3 Present Research 18

**2 Feature Indicator for Anisotropic Adaptation** 19

2.1 Errors in CFD calculations 19

2.2 Evaluation of Discretization Errors 22

2.3 Length-Based Grid Adaptation 27

**3 Anisotropic Mesh Adaptation Methodology** 31

3.1 Procedures to Drive Edge Metric Lengths Equal 31

3.1.1 Node Enrichment and Its Implementation 32
3.1.2 Node Removal and Its Implementation 33
3.1.3 Edge Swapping and Its Implementation 34
3.1.4 Local Reconnecting with Advancing-Front Method 36
3.1.5 Adaptation Function and Point Movement 37
3.1.6 Mesh Smoothing 41
3.2 Sequencing the Process 41
**4 Data Structures in Mesh Adaptation** 44
4.1 Storage of Connectivity in Adaptation 44
4.2 AVL Binary Tree Structures in Storage of Connectivity 46
4.2.1 Definition and terminology of Binary Tree 47
4.2.2 Binary Tree Traversal 48
4.2.3 Inserting and Deleting Nodes 50
4.2.4 AVL Search Tree 51
4.2.5 Implementing an AVL Tree 52
4.3 Fibonacci Heap 53
4.3.1 Insertion into a Fibonacci Heap 55
4.3.2 DeleteMin 55
4.3.3 Deletion of Specified Element 56
**5 Assessment of the Adaptation Approach** 57
5.1 Supersonic Flow around NACA 0012 57
5.2 Equidistribution of Metric Lengths in Adaptation 67
5.3 Effectiveness of Generating Anisotropy in Adaptation 69
**6 Boundary Layer Enhancement for Solution-Adaptive Grids** 71

6.1 Boundary Layer Adaptation with Blasius Solution 72

6.2 Boundary Layer Adaptation for Subsonic Flow around NACA 0012 Airfoil 80

6.2.1 Adaptation with Un-corrected Hessian 80

6.2.2 Adaptation with Boundary-Layer-Enhanced Hessian 84

6.2.3 Comparison with Benchmark Solution 87

**7 Enhancement for Supersonic Flow Adaptation** 94

7.1 Adaptation to Artificial Discontinuity 94

7.2 Discontinuity-Enhanced Hessian 97

7.3 Difficulties in Realistic Discontinuity Adaptation and Their Remedies 104

7.4 Modification for Uniform Flow Regions 121

**8 Results of Enhanced Anisotropic Grid Adaptation** 124

8.1 Enhanced Approach for Anisotropic Adaptation 125

8.2 Parametric Adaptations to Flow around NACA 0012 Airfoil 127

8.3 Transonic Flow around NACA 0012 Airfoil and Comparison with Experimental Data 134

8.4 Internal Flow Adaptation Results 138

**9 Preliminary Research on Three-Dimensional Anisotropic Adaptation** 143

9.1 Anisotropic Adaptation Approach in Three Dimensions 143

9.2 Results of Three-Dimensional Adaptation 149

9.3 Barriers in Three-Dimensional Adaptation 151

**10 Conclusions** 153

10.1 Summary 153

10.2 Concluding Remarks 156

**References** 157

**Appendices** 171

**Appendix A 2-D Edge-based Hessian Formulas** 172

**Appendix B Stability Analysis and Convergence Enhancement of High Order Upwind Difference Schemes** 175

B.1 Introduction 175

B.2 Analysis of the Point-Jacobi Algorithm 177

B.3 Inconsistent System for High Order Upwind Schemes 184

B.4 Analysis of High Order Upwind Schemes via Variable Extrapolation 188

B.5 Application to 2-D Inviscid Flow Equations 194

B.6 Conclusions 200

**Vita** 201

# List of Figures


| | | |
|---|---|---|
| 1.1 | Delaunay Method. (a) Voronoi Tessellation, Corresponding Delaunay Triangulation, (b) the Empty Circumcircle Property | 8 |
| 1.2 | New Cell Generation in Advancing-Front Method in Two Dimensions | 11 |
| 2.1 | Illustration of One-Dimensional Error Estimate | 23 |
| 2.2 | Transformation of an Ellipse by **T** where $a = \lvert\lambda_1\rvert^{-1/2}$ and $b = \lvert\lambda_2\rvert^{-1/2}$. | 29 |
| 3.1 | Node Enrichment | 33 |
| 3.2 | Node Removal | 33 |
| 3.3 | Edge Swapping | 35 |
| 3.4 | Non-Swappable Connectivity | 36 |
| 4.1 | A Simple Binary Tree | 48 |
| 4.2 | Deletion of a Node from the Binary Tree | 51 |
| 4.3 | Different Implementations of the Binary Tree | 53 |
| 5.1 | Supersonic Flow around NACA 0012. (a) Initial Grid, (b) Solution | 58 |
| 5.2 | Supersonic Flow around NACA 0012. (a) 1st Cycle Grid, (b) Solution | 60 |
| 5.3 | Supersonic Flow around NACA 0012. (a) 2nd Cycle Grid, (b) Solution | 61 |
| 5.4 | Supersonic Flow around NACA 0012. (a) 3rd Cycle Grid, (b) Solution | 63 |
| 5.5 | Supersonic Flow around NACA 0012. (a) 4th Cycle Grid, (b) Solution | 64 |
| 5.6 | Supersonic Flow around NACA 0012. (a) 5th Cycle Grid (b) Solution | 65 |
| 5.7 | Supersonic Flow around NACA 0012. (a) 6th Cycle Grid, (b) Solution | 66 |
| 5.8 | Distribution of Riemann Edge Lengths of the Initial Mesh | 68 |

| | | |
|---|---|---|
| 5.9 | Distribution of Riemann Edge Lengths of the Final Mesh | 68 |
| 5.10 | Definition of the Grid Aspect Ratio in Two Dimensions | 70 |
| 5.11 | History of the Grid Aspect Ratio | 70 |
| 6.1 | Adapted Boundary Layer Mesh -- Hessian Defined from Shear Velocity. (a) Far View, (b) Local View | 73 |
| 6.2 | Boundary Layer Hessian Profile -- Blasius Solution | 74 |
| 6.3 | Adapted Boundary Layer Mesh -- Hessian Defined from Stream Function | 75 |
| 6.4 | Adapted Boundary Layer Mesh – with Boundary Layer Enhancement. (a) Far View, (b) Local View | 79 |
| 6.5 | Boundary Layer Hessian Profile – with Boundary Layer Enhancement | 80 |
| 6.6 | Original Isotropic Grid for Boundary Adaptation (2000 nodes) | 82 |
| 6.7 | Contours of Mach Number from Original Grid | 82 |
| 6.8 | Adapted Mesh without Boundary Layer Enhancement (10000 nodes). (a) Far View, (b) Local View | 83 |
| 6.9 | Contours of Mach Number, Adapted Mesh without Boundary Layer Enhancement | 84 |
| 6.10 | Adapted Mesh, with Boundary Layer Enhancement (10000 nodes). (a) Far View, (b) Local View | 85 |
| 6.11 | Contours of Mach Number, Adapted Mesh with Boundary Layer Enhancement | 86 |
| 6.12 | C_Type Grid around the Airfoil (29849 nodes). (a) Far View, (b) Local View | 88 |
| 6.13 | Contours of Mach Number, C_Type Mesh (29849 nodes) | 89 |
| 6.14 | Pressure Distribution for NACA 0012 (Benchmark Solution) | 89 |
| 6.15 | Skin Friction Coefficients for NACA 0012 (Benchmark Solution) | 90 |

| | | |
|---|---|---|
| 6.16 | Comparison of Pressure Distribution for NACA 0012 | 91 |
| 6.17 | Comparison of Skin Friction Coefficients for NACA 0012 | 91 |
| 6.18 | Comparison of Upper Surface $y^+$ Distribution for NACA 0012 | 93 |
| 7.1 | Artificial Discontinuity – Initial Grid | 95 |
| 7.2 | Artificial Discontinuity – Grid after Second Iteration | 96 |
| 7.3 | Artificial Discontinuity – Grid after 20 iterations | 97 |
| 7.4 | Artificial Discontinuity –Grid After 20 Iterations with Maximal Eigenvalue Control | 99 |
| 7.5 | Hessian Evaluation on an Isotropic Grid. (a) Grid, (b) Hessian | 101 |
| 7.6 | Hessian Evaluation on an Anisotropic Grid (AR = 10). (a) Grid, (b) Hessian | 102 |
| 7.7 | Hessian Evaluation on an Anisotropic Grid (AR = 1000) | 103 |
| 7.8 | Supersonic flow around NACA 0012 – Initial Mesh | 105 |
| 7.9 | Front of Bow Shock Region and Interpolated Mach Numbers – Initial Mesh | 105 |
| 7.10 | Front of Bow Shock Region and Interpolated Mach Numbers – First Iteration | 106 |
| 7.11 | Front of Bow Shock Region and Interpolated Mach Numbers – Second Iteration | 106 |
| 7.12 | Front of Bow Shock Region and Interpolated Mach Numbers – Fifth Iteration | 107 |
| 7.13 | Locations of the Shock | 108 |
| 7.14 | Front of Bow Shock Region and Interpolated Mach Numbers – Mesh with Smoothed Hessian | 110 |
| 7.15 | Mesh Adapted from the Smoothed Hessian | 110 |
| 7.16 | Contous of Mach Number Calculated with the Mesh in Fig. 7.15 | 111 |

7.17 Front of Bow Shock Region in the Mesh Adapted from Smoothed Hessian 111

7.18 Rear Part of Bow Shock in the Mesh Adapted from Smoothed Hessian 112

7.19 Smoothing the Discontinuity Equally 113

7.20 New Approach to Smooth and Make Shocks Equal 114

7.21 Inflection Point in the Center of Shock 117

7.22 Front of Bow Shock Region in the Mesh Adapted from Discontinuity-Enhanced Hessian 118

7.23 Rear part of Bow Shock in the Mesh Adapted from Discontinuity-Enhanced Hessian 118

7.24 Mesh Adapted from the Discontinuity-Enhanced Hessian 119

7.25 Contours of Mach Number Calculated with the Mesh in Fig. 7.24 120

7.26 Contours of Mach Number around the Leading Edge (Smoothed Hessian) 120

7.27 Contours of Mach Number around the Leading Edge (Discontinuity-Enhanced Hessian) 121

7.28 Adaptation without Uniform Flow Modification 123

8.1 Initial Mesh for Adaptation of Flow Around NACA 0012 128

8.2 Adaptation for Flow at Different Attack Angles around NACA 0012 (Mesh) 129

8.3 Adaptation for Flow at Different Attack Angles around NACA 0012 (Mach Number Contours) 130

8.4 Adaptation for Flow at Different Mach Numbers and Reynolds Numbers around NACA 0012 (Mesh) 132

8.5 Adaptation for Flow at Different Mach Numbers and Reynolds Numbers around NACA 0012 (Mach Number Contours) 133

| | | |
|---|---|---|
| 8.6 | Adaptation for Transonic Flow around NACA 0012 (Mesh) | 136 |
| 8.7 | Adaptation for Transonic Flow around NACA 0012 (Mach Number Contours) | 136 |
| 8.8 | Comparison between Results from Adaptation and Experimental Data | 137 |
| 8.9 | Geometry Boundary Conditions of the Convergent/Divergent Nozzle | 138 |
| 8.10 | Initial Mesh of the Convergent/Divergent Nozzle | 140 |
| 8.11 | Mach Contours from the Calculation on the Mesh in Fig. 8.10 | 140 |
| 8.12 | Mesh of the Nozzle after $4^{th}$ Iteration | 141 |
| 8.13 | Mach Contours from the Calculation on the Mesh in Fig. 8.12 | 141 |
| 8.14 | Pressure Contours from the Calculation on the Mesh in Fig. 8.12 | 142 |
| 8.15 | Zoomed View of the Boundary Layer Mesh at the Throat | 142 |
| 9.1 | Illustration of Node Enrichment in Three Dimensions | 144 |
| 9.2 | Three-Dimensional Illustration of Node Removal | 145 |
| 9.3 | The Tetrahedralizable and Non-Tetrahedralizable Manners of Subdividing a Prism | 146 |
| 9.4 | Edge Swapping for Three-Dimensional Mesh Adaptation | 147 |
| 9.5 | Face Swapping for Three-Dimensional Mesh Adaptation | 148 |
| 9.6 | Surface Grid from Three-Dimensional Anisotropic Adaptation | 150 |
| 9.7 | Volume Grid from Three-Dimensional Anisotropic Adaptation | 150 |
| 9.8 | Anisotropic Grid on the Surface | 152 |
| A.1 | Vertex $P_0$ and Adjacent Neighbors. | 173 |
| B.1 | A.F. of the Outer Iteration(I/I) | 180 |
| B.2 | A.F. of the Inner Iteration(I/I) | 180 |
| B.3 | A.F. with 1 Inner Iteration(I/I) | 181 |
| B.4 | A.F. with 2 Inner Iterations(I/I) | 181 |

B.5 A.F. with 20 Inner Iterations(I/I) 181

B.6 A.F. with Infinite Inner Iterations(I/I) 181

B.7 A.F. of the Outer Iteration(II/II) 182

B.8 A.F. of the Inner Iteration(II/II) 182

B.9 A.F. with 1 Inner Iteration(II/II) 182

B.10 A.F. with 2 Inner Iteration(II/II) 182

B.11 A.F. with 20 Inner Iterations(II/II, CFL=1) 182

B.12 A.F. with 20 Inner Iterations(II/II, CFL=10) 182

B.13 A.F. with 1 Inner Iteration(I/II) 183

B.14 A.F. with 2 Inner Iterations(I/II) 183

B.15 A.F. with 20 Inner Iterations(I/II) 183

B.16 A.F. with Infinite Inner Iterations(I/II) 183

B.17 A.F. of the Outer Iteration(II/II, Scalar) 186

B.18 A.F. of the Inner Iteration(II/II, Scalar) 186

B.19 A.F. of the Outer Iteration(I/II, Scalar) 187

B.20 A.F. of the Inner Iteration(I/II, Scalar) 187

B.21 A.F. of the Outer Iteration(Mod/II, Scalar) 187

B.22 A.F. of the Inner Iteration(Mod/II, Scalar) 187

B.23 A.F. with 1 Inner Iteration(I/II) 187

B.24 A.F. with 1 Inner Iteration(Mod/II) 187

B.25 A.F. with 2 Inner Iteration(I/II) 188

B.26 A.F. with 2 Inner Iteration(Mod/II) 188

B.27 A.F. with 20 Inner Iteration(I/II) 188

B.28 A.F. with 20 Inner Iteration(Mod/II) 188

B.29 A.F. of the Outer Iteration (I/BSU, Scalar) 193
B.30 A.F. of the Outer Iteration (Mod/BSU, Scalar) 193
B.31 A.F. of the Outer Iteration (I/BSU, Euler) 193
B.32 A.F. of the Outer Iteration (Mod/BSU, Euler) 193
B.33 A.F. with 2 Inner Iterations (I/BSU, Euler) 193
B.34 A.F. with 2 Inner Iterations (Mod/BSU, Euler) 193
B.35 A.F. with 20 Inner Iterations (I/BSU, Euler) 194
B.36 A.F. with 20 Inner Iterations (Mod/BSU, Euler) 194
B.37 A.F. of the Outer Iteration (II/II, 2D Scalar) 196
B.38 A.F. of the Inner Iteration (II/II, 2D Scalar) 196
B.39 A.F. of the Outer Iteration (I/II, 2D Scalar) 196
B.40 A.F. of the Inner Iteration (I/II, 2D Scalar) 196
B.41 A.F. of the Outer Iteration (Mod/II, 2D Scalar) 197
B.42 A.F. of the Inner Iteration (Mod/II, 2D Scalar) 197
B.43 A.F. of the Outer Iteration (I/BSU, 2D Scalar) 197
B.44 A.F. of the Outer Iteration (Mod/BSU, 2D Scalar) 197
B.45 A.F. of the Outer Iteration (I/II, Euler) 198
B.46 A.F. of the Inner Iteration (Mod/II, Euler) 198
B.47 A.F. of the Outer Iteration (I/BSU, Euler) 199
B.48 A.F. of the Inner Iteration (Mod/BSU, Euler) 199
B.49 A.F. of the Outer Iteration (I/II, Euler, AR=100) 199
B.50 A.F. of the Outer Iteration (Mod/II, Euler, AR=100) 199
B.51 A.F. of the Outer Iteration (I/BSU, Euler, AR=100) 200
B.52 A.F. of the Outer Iteration (Mod/BSU, Euler, AR=100) 200

# Nomenclature

## English Symbols

| | |
|---|---|
| **A** | Jacobi matrix of the flux vector |
| A | Triangle area |
| **C** | Cosine function |
| $C$ | Scalar constant |
| $C_f$ | Skin viscous coefficient |
| $C_p$ | Pressure coefficient |
| CFL | Courant-Friedrichs-Lewy number |
| $c$ | Scalar variable, 1-D element or airfoil chord length |
| **D** | Jacobi matrix of the source vector |
| **E** | Flux vector |
| $E$ | Approximation error |
| $e$ | Edge between two nodes |
| $f$ | Scalar function or flow variable |
| **H** | Hessian matrix, or source term in Euler equation |
| $h$ | Element of Hessian matrix or scalar variable |
| **I** | Identity matrix |
| I | Point index |

| | |
|---|---|
| J | Point index |
| **L** | Temporary vector |
| $l$ | Edge length |
| **K** | Temporary vector |
| $k$ | Point index or scalar coefficient |
| $k'$ | Temporary scalar |
| **M** | Temporary vector |
| Ma | Mach number |
| N | Number of edges (triangles) around an edge |
| **P** | Eigenmatrix |
| **Q** | Primary transport variable |
| **R** | Eigenmatrix or temporary vector |
| Re | Reynolds number |
| RMS | Root-mean-square |
| $r$ | Length scale for VNN definition |
| **S** | Sine function |
| **T** | Transformation matrix |
| $T$ | Triangle |
| $t$ | Time |
| **U** | Temporary vector |
| $u$ | Scalar flow variable |
| **V** | Unit vector |
| VNN | Von Neumann number |
| **x** | Temporary vector |

| | |
|---|---|
| $x$ | Horizontal coordinate |
| $y^+$ | $(\rho y_w/\mu)\sqrt{\tau_w/\rho}$ |
| $y_w$ | Distance to a wall |

## Greek Symbols

| | |
|---|---|
| $\alpha$ | Flow attack angle, or angle in coordinate rotation |
| $\mathbf{\Lambda}$ | Diagonal matrix of the eigenvalues |
| $\lambda$ | Eigenvalues; CFL number |
| $\mu$ | Molecular viscosity |
| $\nu$ | Kinetic viscosity |
| $\psi$ | Stream function |
| $\omega$ | Relax coefficient or wave number |
| $\rho$ | Density |
| $\tau$ | Shear stress |
| $\Delta$ | Laplace operator $(\nabla \bullet \nabla)$ |
| $\Delta x$ | Grid spacing |
| $\phi$ | Weight function |
| $\Omega$ | Polygon formed by triangles around one node |
| $\nabla$ | Gradient operator |

## Subscripts

| | |
|---|---|
| $i$ | Point index |

| | |
|---|---|
| *j* | Point index |
| *max* | Maximum |
| *min* | Minimum |
| *n* | Point index |

## Superscripts

| | |
|---|---|
| *′* | First order derivative, or variable in a transformed coordinate |
| *″* | Second order derivative |
| - | Average |
| *R* | Riemann space |
| *E* | Euclidean space |

# Chapter 1

# Introduction

Computational Fluid Dynamics has been used to obtain numerical solutions for a wide range of engineering flows. The dramatic development in computing potential has allowed Computational Fluid Dynamics to become an important complement to experimental study. Equations governing flows, such as the Navier-Stokes equations, are mostly partial differential equations. Discretization, which includes both the spatial domain such that flows are represented by values at discrete points distributed all over the flow domain, and the equations where variables in the equations are replaced by those at points, must be conducted before computing. A straightforward example of discretization is shown below with the evaluation of the one-dimensional gradient,

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2\Delta x} - \frac{f'''\Delta x^2}{6} + \cdots\cdots \tag{1.1}$$

Suppose that values of the function $f$ are known at a series of equally spaced points, the gradient of the function at point *i* can be approximated as the difference of the values at neighboring points divided by their distance as shown in equation 1.1. The truncation error associated with such approximation, which is of order $\Delta x^2$, is also included in the formulation. Here $\Delta x$ is the distance between adjacent grid points.

The truncation error is proportional to the product of the local derivative of the function and the grid space. The order of the derivative and the coefficient in the truncation error depend upon how the points are chosen in the discretization scheme. Central differencing, which means that the gradient at the point is evaluated with the adjacent two nodes as shown in Equation 1.1, gives a truncation error of second order. The implication from this example tells us that truncation errors on an equal-spaced mesh will not be uniform unless the function is so trivial that it is constant or linear. Because the truncation error depends on the local derivatives of the function, it is desirable that smaller grid sizes appear in regions where the function experiences stronger variations.

The same fundamental concept exists in complex Computational Fluid Dynamics calculations. While larger and larger numbers of grid points are generally used in current CFD simulations, most of these points are positioned in regions where features such as boundary layers, discontinuities or flame fronts appear. The flow in these regions experiences more dramatic changes compared with that in the rest of the domain. Usually many functions as well as the gradient need to be calculated when the governing equations are solved in CFD. The accuracy of these calculations requires the use of much smaller grid sizes in relatively small regions embedded in the flow field where flow variables vary rapidly.

The grid generation process allows a certain extent of mesh adjustment according to local flow features. When a two-dimensional domain, for example, is discretized, it results in elements such as points, edges and faces. The connectivity of the mesh describes the manner in which the grid points are joined, i.e., how the edges and faces are formed. The degrees of freedom of the mesh represented by the point distribution and their connectivity must be defined during the mesh generation process.

From the perspective of a cell, the degrees of freedom in a mesh are reflected by the cell size, shape and orientation. It is worthwhile to note that although isotropic cells such as equilateral triangles only require one parameter to define each of their sizes, generally more parameters are necessary to determine the size of a cell that is non-isotropic in different directions. The distribution of grid points can be controlled by assigning cell sizes everywhere in the flow field, either one or more at each location, while local connectivity determines the shape and orientation of cells there.

Two distinct types of grids are widely used in Computational Fluid Dynamics. Structured grids, whose coordinates and connectivities can be mapped into elements of a matrix, originate historically from rectangular Cartesian grids. The lattice of the structured grid provides easy identification of neighboring points to be used in the representation of derivatives and is particularly useful when the finite difference approach is selected. Unstructured meshes represent another type of grid where an arbitrary distribution of points and any feasible connectivity among them are allowed. Unstructured grids cannot be represented as elements of a matrix. The points and connectivities of the unstructured grid do not possess a global structure. Unstructured grids offer more degrees of freedom than structured grids, because the connectivity of grid points which determines the element shape and orientation can be changed as well as the point locations.

Cells of many kinds of shapes are used in spatial discretization. Usually triangles and quadrilaterals are seen in two-dimensional grid generation, either alone or together in a hybrid grid. Hybrid grids are particularly useful when large grid stretching is desired near a wall. In this case, a body-fitted grid consisting of quadrilaterals is used close to the wall and triangles are generated in regions far from the wall where stretched cells are not

required. In three dimensions, tetrahedrons, hexahedrons, prisms (wedges) and even pyramids can be found in many practical grids. Triangles and quadrilaterals, which are cells themselves in two dimensions, become faces of the above three-dimensional cells. Generally quadrilaterals and hexahedrons are used in structured grids while any of the above cells and their combinations are acceptable in an unstructured grid.

The generation of quality grids continues to be a formidable task for most grid generators, either commercial or proprietary codes, since this relies upon the user's engineering sense of where to concentrate points and how to build connectivity within given constraints of memory, geometry and flow features. It is desirable that the distribution of points and their connectivity automatically take into account the flow features during the mesh generation process.

Solution-adaptive grids have received ever-increasing attention over the years [1-10]. Grid adaptation utilizes the mesh freedom by controlling the point distribution and their connectivity to satisfy the requirements of a more accurate solution. It is the goal of calculations with adaptive grid to obtain the best resolution with a given number of grid points. Effective adaptive grid methods should enable the resolution of complex flow fields more efficiently and with reduced computing resources.

The categorization of grid adaptation approaches in recent literature can be made according to the extent to which the degrees of freedom in the mesh are used. Two large categories of adaptation, namely isotropic adaptation and anisotropic adaptation, exist in the research area of grid adaptation. Isotropic adaptation produces isotropic grids and only requires the specification of the variation of the cell areas (volumes), one value at each location, over the computational domain, whereas anisotropic grids, which result

from anisotropic adaptation, require sizes which are defined with more than one parameter at each point, cell types and the cell orientation.

Isotropic grid adaptation distributes isotropic cells over the domain according to a prescribed criterion and has been attempted in both structured and unstructured meshes [1,2,6,7,10,19,22,34,35,56,66,84,85,91,102]. Larger variations in relatively small regions of the flow field require finer grids there. The only criterion necessary in the adaptation is the local size of the mesh element. Isotropic elements of different sizes are ideal for flow fields where flow variations are not highly directional, such as inviscid subsonic flows.

Anisotropic adaptation has also been the focus of recent work for both two and three-dimensional grids [4,5,8]. The interest in anisotropic adaptation arises from the features in fluid dynamics. Most of the phenomena in flow simulations that require more attention are highly directional, as for example in boundary layers and discontinuities. Efficient resolution of these kinds of flows demands the use of strongly anisotropic grids.

The feasibility of anisotropic adaptation comes mostly from the degrees of mesh freedom described earlier. Structured grids have a fixed connectivity. Although the fixed connectivity does allow the generation of anisotropic grids around prescribed geometries, such as boundary layer meshes for high Reynolds number flows, structured grids are generally not flexible enough to align the grid along arbitrary flow features. Unstructured grids provide more room for adaptation since the connectivity can be altered. Cell size and orientation can be changed during adaptation. Even different kinds of cells can be mixed to achieve better mesh quality. It is thus possible to align the cells according to the local flow features. Cells can be short across the flow feature and long in the directions parallel to it. For example, when a shock is computed with anisotropic

grids, it is possible to use much finer grid spacings across the shock compared with those along the shock.

Anisotropic adaptation on unstructured grids is chosen as the topic of current work. Although a wide range of types of cells can be used in unstructured grids, only a single cell type is adopted for simplicity. Triangles are chosen for two-dimensional grids while tetrahedrons are used in three dimensions.

## 1.1 Mesh Generation Methods

It is worthwhile to make a survey of the concurrent unstructured grid generation methods because many of the procedures used in grid generation are also useful in grid adaptation. Three commonly used techniques for generating unstructured meshes are the octree method, the Delaunay method and the advancing-front method. Each of these is described briefly below. Particular attention is paid to their capability to generate anisotropic grids.

### 1.1.1 Octree Method

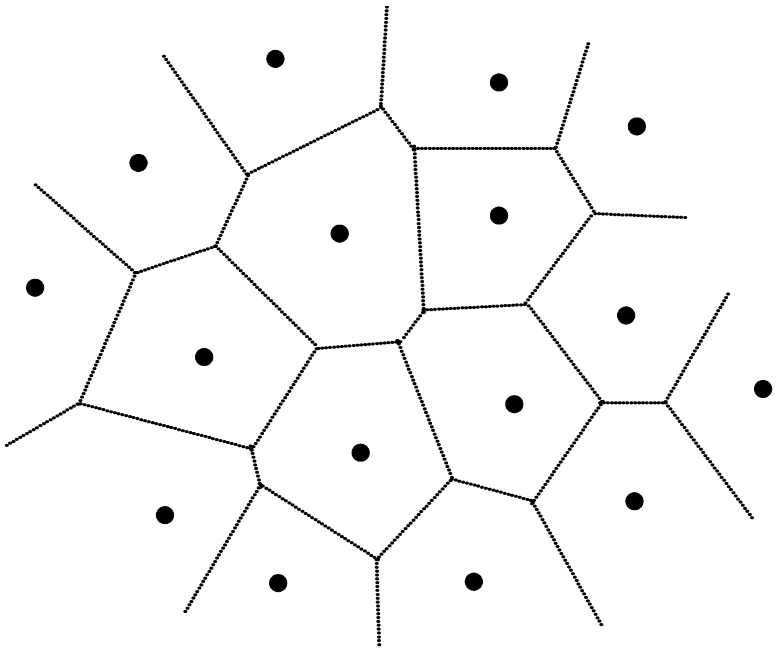
In the octree method [16,17], a master hexahedron is created which completely encompasses the three-dimensional domain to be meshed. The master hexahedron is recursively subdivided into eight child hexahedrons, called octants, by introducing new nodes at its centroid and at the centroid of each of its faces. The subdivision process continues until the sizes of the octants near the surface are on the same order as the sizes

of the elements of the local surface triangulation done earlier. In order to ensure a smooth variation in element size, additional subdivision is performed such that neighboring octants do not differ in depth by more than one. In some cases, the final mesh is obtained by further subdividing each octant into tetrahedrons. Two-dimensional versions of the approach are also straightforward.

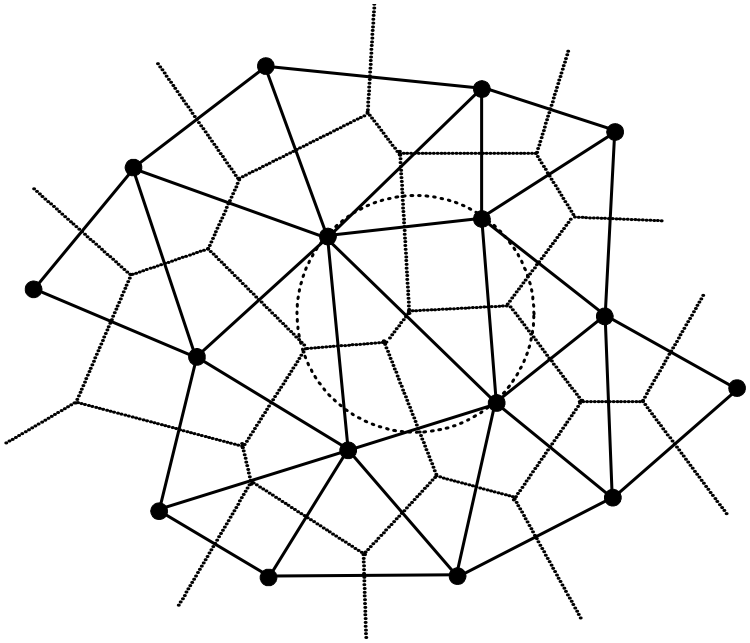
So long as each cell is divided into eight new cells, the degree of isotropy of the refined grid will be identical to that of the original master hexahedron. If it was a cube, all new elements will be cubes. If it was highly stretched, all new elements will be highly stretched also. It is clear that the degree of anisotropy could be changed by dividing a hexahedron into only four or two cells rather than eight. Most applications, however, have used the simpler "eight-children" method.

### 1.1.2 Delaunay Method

Given a set of points $\{P\}$ in a plane, there exist many ways to join the points together to form a set of non-overlapping triangles which completely covers the domain. The Delaunay triangulation represents a particular construction of this type which has various well-defined properties [104]. For example, the Delaunay triangulation is the dual of a particular way of tessellation, in which a graph is obtained by drawing the median line segments among discrete points that separate the plane into regions. Each of these regions is closer to a given point of $\{P\}$ than to any other point in the set $\{P\}$, as shown in Figure 1.1. This tessellation is called a Voronoi tessellation. If a line segment is drawn between any two points that are neighbors in this Voronoi diagram, the Delaunay triangulation of these points is obtained. Another well-known property of the Delaunay



*(a)*



*(b)*

*Fig. 1.1 Delaunay Method.*

*(a) Voronoi Tessellation and Corresponding Delaunay Triangulation, (b) Empty Circumcircle Property*

triangulation is the empty circumcircle property, which states that no point in the grid is contained inside the circumcircle of any triangle.

In the Delaunay method [18,19,20,21,22,23], the mesh connectivity is determined by constructing a Delaunay triangulation of the mesh points. A less strict version of the Delaunay triangulation, known as the constrained Delaunay triangulation in which the triangulation meets the Delaunay criteria locally instead of globally, is available in two dimensions but not in three dimensions [102,103]. A common approach in Delaunay grid generation is to use an incremental insertion algorithm one point at a time by successively adding points to an existing Delaunay triangulation. The locations of the field points can be either pre-specified or determined during the mesh generation process.

It is clear from these definitions that the Delaunay method will produce isotropic (or nearly isotropic) grids. Methods for producing stretched grids by modifying the Delaunay procedure are mentioned later.

### 1.1.3 Advancing-Front Method

The advancing-front method [25,9,8,26,27] is based largely on heuristics in which the mesh is generated one element at a time by propagating the boundary discretization inward. The generation process begins by choosing one edge (face in three dimensions) on the front as the base of a new element. A new element is then formed from the chosen edge (or face) either by introducing a new point or by connecting with an existing point. The front is updated and the process continues until the whole domain

is filled with elements. A coarse triangulation, called the background, is usually used to store the size information that controls the element sizes throughout the domain.

The nodal positions and connectivities are constructed simultaneously in the advancing-front procedure. Figure 1.2 shows a two-dimensional case where the front is a dynamic collection of edges. As noted before, the front is continually updated throughout the mesh generation process. Although the front can be composed of many disjoint pieces, each one must be a closed curve which is oriented so that the domain to be meshed lies to the left. The front is initialized to consist of the edges on the boundary of the domain to be triangulated. An edge on the front is selected to be the base of a new cell which will be generated. Based on the local spacing value obtained from the background grid, the location of an 'ideal' point to be the third vertex of the new element is computed. Nearby nodes on the front are then found and a decision is made to connect with one of these candidates, including the 'ideal' point. After a new cell is formed, the existing edge(s) used in the formulation of the new element are deleted from the front while any newly created edges are added to it. The process then repeats until there are no more edges on the front. The steps in the procedure can thus be described as following, with the illustration in Fig. 1.2.

1. Initialize the front.

2. Select an edge on the front to be the base of the new element. Usually the shortest one [8] is chosen first, as $AB$ in the figure.

3. Obtain from the background grid the local spacing value at the midpoint of the edge.

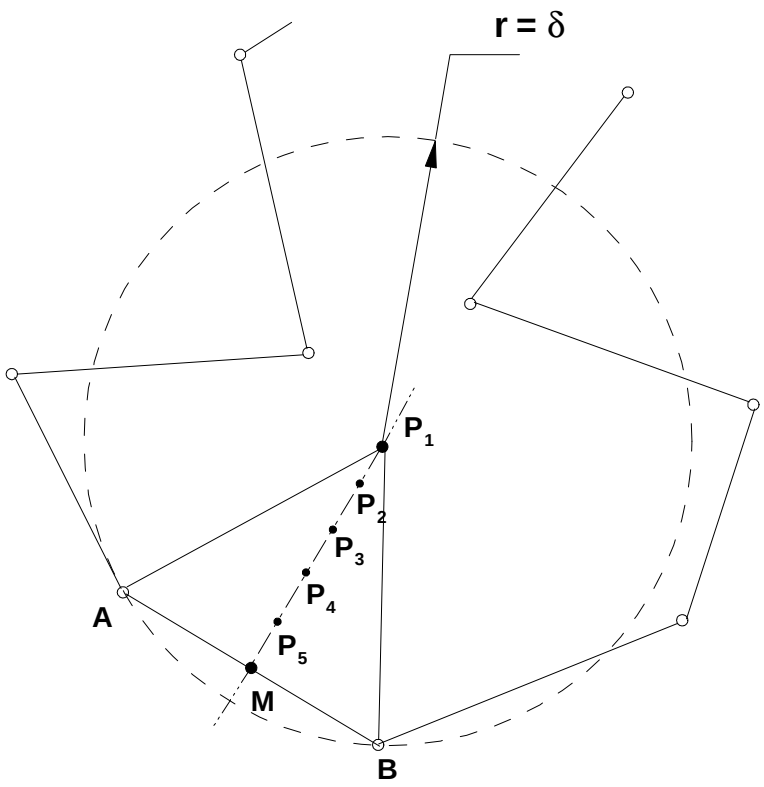



*Fig. 1.2 New Cell Generation in Advancing-Front Method in Two Dimensions*

4. Determine the location of a new 'ideal' point ($P_1$) to be the third vertex of the new element. This is done by choosing a $\delta$, which is the radius of the circle centered at $P_1$ and connecting points $A$ and $B$. The value of $\delta$ depends upon the length of $AB$. Details can be found in [8]. Note that the triangles developed by this method will be isotropic.

5. Find other possible choices from the nearby nodes on the front. This is judged by checking if the new connectivity intersects any edge in the front. These nodes are another group of candidates for forming the new cell.

6. Decide which one will be chosen from the nearby candidates. This is done by constructing a circle passing through *A*, *B* and each of the candidates. The

centers of all circles lie on the line $MP_1$, which is perpendicular to $AB$. The point that results in the furthest distance from $P_1$ in the direction of $P_1M$ is the ideal point. Note that $P_1$ and a sequence of points (usually 4) between $P_1$ and $M$ serve as the alternative candidates if none of the nearby points results in a valid connectivity. It can be seen that this approach tries to generate a triangle as close to isotropic as possible.

7. Form the new element and update the front.

8. If the front is not empty, go to step 2.

### 1.1.4 Extensions to Anisotropic Grid Generation

The descriptions of all three basic grid generation methods given above show that they all lead to isotropic or nearly isotropic grids. High Reynolds number fluid dynamics simulations, however, require strongly anisotropic grids. All three methods can be modified to produce anisotropic grids by appropriate extensions. In the case of the octree method, this can be done by subdividing cells in one or two directions (as noted above) instead of all three. The resulting grid anisotropy produced by the modified octree method, however, is not very useful in that while the grid aspect ratio can be controlled very effectively, the cell orientation is (essentially) fixed by the underlying master grid. Anisotropic grids generated by the octree method can be effective in limited conditions such as in flow fields where the major flow features are parallel to the master hexahedron, such as the boundary layer near a flat plate.

Both Delaunay and advancing-front methods are more readily adapted to anisotropy by simply forming the isotropic elements in a transformed (stretched) space. George *et al* [4] applied the Delaunay method to the generation of two-dimensional anisotropic grids by first transforming from a Euclidean space to a Riemann space, whose definition is given in the next chapter. The Delaunay criterion is then used to guide the insertion of the new point in the transformed space rather than the physical space. Although successful two-dimensional examples can be seen in his work, care must be taken when dramatic changes of the solution take place from point to point. Difficulties arise when the Delaunay criteria are extended to three-dimensional anisotropic grid generation and adaptation because of the lack of the constrained Delaunay triangulation in three dimensions [4].

The strategy of advancing-front methods has also been used to generate anisotropic mesh in two and three dimensions [9,8]. The criteria of the advancing-front method are again applied in a Riemann space into which the relevant connectivities are transformed. Elements with high aspect ratio will be obtained when these isotropic elements are transformed back to the Euclidean space. More detailed descriptions of this method are included in Chapter 3.

## 1.2 Solution Adaptation Techniques

Work on solution adaptive methods has also led to the development of their own peculiar techniques. These techniques either adjust the calculation or manipulate the mesh according to the flow features. Many of the mesh manipulations have their roots in grid generation methods.

Techniques used widely in solution adaptation can be categorized as the *p*-method in which the order of the scheme is varied locally, the *r*-method in which the points are repositioned and the *h*-method in which points are added or removed. A remeshing method in which a totally new mesh is generated at each adaptation level has also been used as well as combined methods that couple several of the above techniques. Analyses of the strengths and weaknesses of these methods are given below.

### 1.2.1 *p*-Method

In the *p*-method [11,12], the adaptation of a calculation is accomplished through increasing the local order of the scheme in the finite volume method (FVM) or finite difference method (FDM), or by increasing order of shape functions in the finite element method (FEM). The *p*-method does not involve grid movement, grid refinement and other mesh manipulations. In an optimized *p*-method, the solver would adjust the order of calculations from region to region as necessary. Although the concept of the *p*-method is widely accepted, it has seen little, if any, implementation in practice. The reason is that increasing the order of accuracy of a scheme rapidly increases its complexity so that it quickly becomes impractical. This is especially true in unstructured grids where the identification of an appropriate number of neighbors is difficult. One exception to this is an example in which the *p*-method is widely used to resolve shocks and other discontinuities by limiters [55,106]. Because high-order schemes have a tendency to oscillate in the presence of discontinuities, a locally low-order method is required to provide monotonicity in regions such as the vicinity of a shock. Although they are not normally thought of as an adaptation method, shock limiters effectively represent *p*-refinement. Because of the lack of previous applications and the difficulties of high-order

schemes in unstructured grids, the *p*-method is not considered in the present work (apart from shock limiters).

### 1.2.2 *r*-Method

The *r*-method of grid adaptation refers to repositioning the nodes of the mesh while maintaining the original connectivity. The *r*-method has seen application in numerous earlier works [2,13,14,75] and represents the primary technique that is used on structured grids. Structured grids have a fixed connectivity and changes of the point positions can help to better resolve local large gradients. In some moving boundary problems where the displacement is small or the geometry is simple, the *r*-method has been the primary approach because it requires no change in the solver and precludes the interpolation errors that are incurred if the domain is remeshed. For example, the flow in the cylinder of an internal combustion engine with a moving piston is usually modeled with a moving grid that compresses and expands layers of grid points in conjunction with the piston movement [105]. The primary limitation of the *r*-method lies in the fact that a change in connectivity is often necessary for better mesh alignment after the points have been displaced. Large movements of the nodes from their original positions can degrade the quality of the elements. In more complex applications, solution adaptation with node movement is limited. In addition to being used as a stand-alone method, the *r*-method is also routinely used in combination with other approaches. Most grid generators use grid smoothing as a final step in grid generation, which can also be considered as an *r*-adaptation.

### 1.2.3 *h*-Method

In the *h*-method new nodes are added or existing nodes are removed from a mesh in local regions of the flow field. Adaptations with the *h*-method usually start with a coarse grid and recursively refine the grid locally. In two dimensions, for example, nodes are added at some intermediate locations of either the elements or the edges. While this does not cause difficulty in unstructured grids because of the allowance of connectivity [15], it can result in hanging nodes when only one or two edges of a rectangle are flagged for refinement. When a structured grid is used as the initial frame (usually Cartesian grid), a hierarchical tree structure is produced if this refinement process is done repeatedly. Cartesian-grid methods are highly developed and have been applied to a wide variety of complex geometries [1,6,16,59,85]. The result of applying the *h*-method in Cartesian grids is a highly unstructured grid. Because Cartesian methods align the grids with the coordinate system rather than with the flow features, they are essentially limited to the Euler equations. Three-dimensional adaptive refinement on tetrahedral grids is reported in the work by Löhner [15] and Mavriplis [102]. It can be seen that the major drawback of the refining procedure in the *h*-method is the lack of capability to align the grid to directional features. It is therefore not well suited to problems that have highly anisotropic features, especially when these features are not in the directions of axes.

### 1.2.4 Remeshing Method

The remeshing approach generates a completely new mesh based on the analysis of the flow field on the previous mesh. The methods of grid generation discussed earlier

are thus applied directly here. Usually the previous mesh, where the information retrieved from the previous solution is stored, serves as the background grid for the generation of a new mesh. Successful applications can be seen in the work by Baker *et al.* [2,89] and Mavriplis [23] that are based on the Delaunay method and by Löhner *et al.* [91] with the advancing-front approach. However, a complete regeneration of the mesh at each adaptive step is not economical, especially for problems with large numbers of nodes. Remeshing becomes particularly inefficient when the mesh in steady flow simulation approaches convergence so that the new mesh has only minor changes over the previous one.

### 1.2.5 Combined Methods

All of the above adaptation methods have limitations and none of them is optimum for all grid adaptation applications. As a result, combined methods have been implemented by many researchers [2,3,5,10,69] in grid adaptation. Unstructured grids are mostly used for their allowance of connectivity optimization. In most cases adaptation is based on the modification of the current mesh. Operations on the mesh include those from the *r*, *h* methods and even those in the mesh generation procedure.

Grid adaptation procedures must reply upon some prescribed criteria that reflect the features of the flow field. Earlier adaptation research focused on the reduction of simulation errors in the flow domain, and the term Error Estimate is associated with much of the literature. Modeling errors are difficult to evaluate and their estimation itself has become a branch of grid adaptation [36,37,3,68]. An alternative term, a Feature Indicator, can also be used in adaptation [69]. A good feature indicator captures the key

features of the flow and can be used in the adaptation procedure to produce grids resolving these features.

## 1.3 Present Research

In the current approach, anisotropic adaptation is adopted with a goal of resolving highly directional flow features efficiently. Moreover, unstructured grids are chosen over structured meshes for their increased degrees of freedom. Triangles and tetrahedrons are chosen as the single type of elements in two and three dimensions respectively. The combined methods of mesh adaptation are used. In particular, some specific problems in the solution adaptation of general flow fields are addressed and suggested methods for circumventing some of these are presented.

The thesis is arranged as follows. Chapter 2 presents the feature indicator, composed of the second derivatives of the flow variables. Chapter 3 defines an adaptation function. The manipulations on the meshes in order to satisfy the optimization of the adaptation function are introduced also. The data structures in the current work are then explained in Chapter 4, followed by an assessment of the standard Hessian based approach in Chapter 5. In Chapter 6 the special difficulties in boundary layer adaptation are discussed and difficulties arising in the adaptation of flows with discontinuities (shocks) are presented in Chapter 7. A series of adaptation results is presented in Chapter 8. Chapter 9 is dedicated to some preliminary research on 3-D adaptation and is followed by a summary and concluding remarks in Chapter 10. Included in the Appendix are the derivation of an edge-based Hessian calculation and an assessment of convergence enhancement of upwind schemes accomplished during the thesis work.

# Chapter 2

# Feature Indicator for Anisotropic Adaptation

The purpose of grid adaptation is to allocate the grid points according to the flow features such that a desired solution accuracy will be obtained with a minimal number of grid points. Regions with strong variations in the flow variables require a finer mesh, while regions where the flow experiences less changes can be resolved with a coarser grid. In isotropic adaptation, the only local grid information needed to determine the cell geometry is the size of the cell. In anisotropic adaptation additional information is needed specify the mesh size (in multiple directions) and orientation. In this chapter, measures to detect the flow features are discussed and their ability for adaptation is analyzed. The error estimate, used in many references to reveal the levels of errors in different regions, is introduced first as a bridge to the feature detector that is used in the current approach.

## 2.1 Errors in CFD Calculations

Computational errors are a major issue in flow simulation. In general computational errors can be subdivided into three groups [28]. These are modeling

errors, discretization errors and convergence errors. Each of these is defined separately below.

- **Modeling Errors**

Modeling errors are defined as the difference between the actual flow and the exact solution of the mathematical model that describes the behavior of the system. In general, the model is composed of a system of coupled partial differential equations complemented by a companion set of algebraic relations. For laminar flows, the mathematical model (the Navier-Stokes equations) can be considered exact for many engineering purposes. However, in turbulent, two-phase or reacting flows, additional physical models must be added and these models do not always describe the underlying physics processes accurately and thus introduce potential modeling errors. These errors can only be rectified through deep understanding of the underlying physics.

We also note that modeling errors do not give useful information for adaptation, since the models are derived through analysis of the flow physics and do not rely on the CFD calculation. Adaptation may increase the accuracy with which the model is computed, but will not improve the effectiveness with which the model describes the flow physics. Errors of this type are not discussed further in the current grid adaptation approach.

- **Convergence Errors**

The system of algebraic equations constructed through discretization is usually solved using an iterative solver. The difference between the computed solution and the exact solution of the discretized system of equations is described by iterative

convergence errors. Although convergence errors can, in principle, be reduced to an arbitrary level by improving the algorithms and/or increasing the number of iterations, a lack of convergence exists in many CFD solutions and errors arising from the incomplete convergence are often seen in simulations.

Because convergence errors depend on the level of convergence, a fully converged solution in which the residual at every point is reduced to machine accuracy, results in the uniform distribution of residuals. This indicates that the convergence error is not useful for grid adaptation since a uniform residual will give a uniform mesh, which, obviously is not correct for resolving the large gradients in the flow field.

- **Discretization Errors**

Discretization errors constitute a third group of errors that originate from the approximation employed in the numerical methods used to solve the mathematical models. Discretization errors describe the difference between the exact solution of the system of discrete algebraic equations obtained by discretizing the governing equations on a given grid and the (usually unknown) exact solution of the continuous mathematical model. Discretization errors depend on the accuracy of the discretization procedure, as well as the discretization of the solution domain, as, for example, the number of cells, their sizes, distribution and shapes, etc.

Since the purpose of solution adaptation is to adapt the grid such that those regions experiencing large flow variations will either be represented by more grid points in *r* and *h* methods or their combination, or be discretized with a higher order scheme in the *p* method, discretization errors are suitable to serve as the primary guidance for adaptation.

## 2.2 Evaluation of Discretization Errors

An accurate estimation of discretization errors is very difficult to make because the exact solution of the set of partial differential equations is usually unknown. A so-called *a-posteriori* error estimator [29,30], which means that errors are approximated without knowing the exact solution, has been studied widely and used extensively in many adaptive procedures [31-37,5,8,3]. Two major methods of the *a-posteriori* error estimation are the Taylor series error estimate and the moment error estimate.

The Taylor series error estimate is based on a Taylor series truncation error analysis, while the moment error estimate is based on a higher moment balance [38]. Estimations with the two approaches are sometimes quite complicated [36]. Although more complicated estimation may give more accurate error indications of the solution, they require greater effort in computing. Moreover, the application of such complex error estimates in solution adaptation is usually also very difficult. It is not uncommon to use less precise but easily computed estimates as a feature of the flow field. It is also desirable that the evaluation of the error estimate takes a small portion of the overall adaptation and solution time.

Peraire *et al* [9,8] and Habashi *et al* [5] have developed an efficient and simple error estimate that forms the basis of the procedure followed here. In this approach, they use finite element interpolation theory as the basis. The resulting error estimate is interpreted as the lowest order term that is neglected in the interpolation functions when the solution is expanded as a Taylor series.

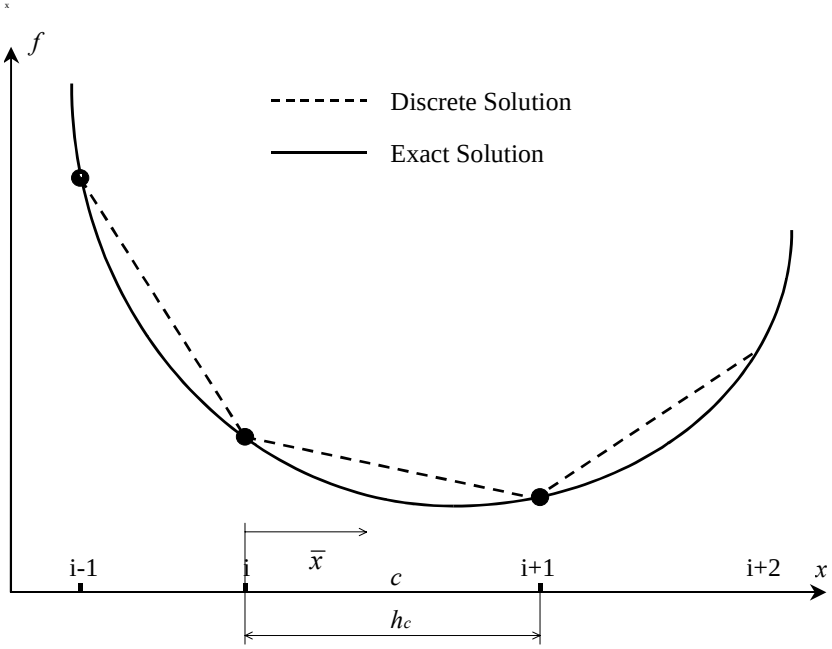



*Figure 2.1 Illustration of One-Dimensional Error Estimate*

The derivation of the above error estimate is first illustrated for a one-dimensional problem for the sake of simplicity and then generalized to the multi-dimensional case. Consider the problem in which the solution $f(x)$ is approximated by $f^h(x)$, with piecewise linear interpolation, as shown in Figure 2.1. A local approximation error, $E_c$, is defined at an arbitrary location over an element $c$ to be,

$$E_c(\bar{x}) = f(\bar{x}) - f_c^h(\bar{x}), \tag{2.1}$$

where $\bar{x}$ is defined as a local coordinate between point *i* and *i+1* and $h_c$ is the width of the element. Hence $\bar{x}$ is within the interval $[0, h_c]$.

The approximate solution at locations between *i* and *i+1*, $f_c^h(\bar{x})$, may be expressed as a function of its nodal values by means of linear interpolation in the form

$$f_c^h(\bar{x}) = \left(1 - \frac{\bar{x}}{h_c}\right) f_i + \frac{\bar{x}}{h_c} f_{i+1}. \tag{2.2}$$

By expanding $f_{i+1}$ in a Taylor series around node $i$, one obtains

$$f_{i+1} = f_i + h_c f_i' + \frac{h_c^2}{2} f_i'' + \cdots \tag{2.3}$$

After substituting Eq. (2.3) into Eq. (2.2) and making some simplifications, the solution $f_c^h$ may be rewritten as

$$f_c^h(\bar{x}) = f_i + \bar{x} f_i' + \frac{\bar{x} \cdot h_c}{2} f_i'' + \cdots \tag{2.4}$$

Since the error at the nodes is zero, the exact solution may also be expanded in the neighborhood of the node $i$ as

$$f(\bar{x}) = f_i + \bar{x} f_i' + \frac{\bar{x}^2}{2} f_i'' + \cdots \tag{2.5}$$

The elemental error at any point $\bar{x}$ is obtained by substituting the two equations (2.4) and (2.5) into Eq. (2.1) and neglecting third order terms, which gives

$$E_c(\bar{x}) = \left(\frac{\bar{x}^2}{2} - \frac{\bar{x} \cdot h_c}{2}\right) f'', \tag{2.6}$$

This can be seen as a departure of the quadratic interpolation in Eq. 2.5 from the linear one in Eq. 2.4. Note here that if the exact solution is a linear function of *x,* the error will vanish.

It is useful to define a root-mean-square (RMS) interpolation error over an element spanning the interval $[x_1, x_1 + h_c]$ following the work of Peraire *et al* [9], as,

$$E_c^{RMS} = \left\{ \int_0^{h_c} \frac{E_c^2}{h_c} dx \right\}^{1/2} = \frac{1}{\sqrt{120}} h_c^2 \left| \frac{d^2 f^h}{dx^2} \right|_c . \tag{2.7}$$

where $E_c$ has been replaced by Eq. 2.6.

Thus, the interpolation error for this 1-D problem is proportional to the product of the magnitude of second derivative and the square of the characteristic length of the element, $h_c$. The vertical bars of $\left| \quad \right|$ represent the magnitude of the derivatives. The function, $f$, used in evaluating the second derivative in Eq. 2.6, is replaced by the discrete solution on the grid points, $f^h$. This represents a reasonable assumption since the exact solution of the coupled system is usually unknown in the *a-posteriori* error estimate.

When extending this adaptation criterion to multi-dimensional cases, the second derivative of $f^h$ is now taken with respect to a given unit vector $\mathbf{V}$ in space as follows

$$\frac{\partial^2 f^h}{\partial \mathbf{V}^2} = \mathbf{V}^T \mathbf{H} \mathbf{V} , \tag{2.9}$$

where $\mathbf{H}(x) = (h_{ij}(x))$ represents the Hessian matrix of $f^h$ and is expressed as:

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f^h}{\partial x^2} & \frac{\partial^2 f^h}{\partial x \partial y} & \frac{\partial^2 f^h}{\partial x \partial z} \\ \frac{\partial^2 f^h}{\partial y \partial x} & \frac{\partial^2 f^h}{\partial y^2} & \frac{\partial^2 f^h}{\partial y \partial z} \\ \frac{\partial^2 f^h}{\partial z \partial x} & \frac{\partial^2 f^h}{\partial z \partial y} & \frac{\partial^2 f^h}{\partial z^2} \end{pmatrix} \tag{2.10}$$

The Hessian is thus defined as the matrix composed of the second derivatives of any variable, like $f^h$. For case where $f^h$ is linear for each element, its second derivatives vanish inside the elements but include jumps on edges between elements. A weak formulation has been developed to evaluate the second derivatives in the finite element method [9]. In the finite volume method, the Green-Gauss theorem is usually used to calculate the gradient and second derivatives [39], while a novel method in which only the node-to-node data structure (edge table) is used has been proven efficient for second derivative calculations [40]. This approach, originally developed by Barth [41] with the concept of the finite element method, gives the advantage that an edge-based algorithm can be implemented. A detailed illustration of the method is given in Appendix A.

One advantage of having this symmetric matrix appear in the error term is that a matrix provides information than can be used to define anisotropic grids with its eigenvectors and eigenvalues. A scalar or a vector such as the gradient of a function does not contain enough information to determine the sizes and orientation of a cell. The Hessian matrix, given by Eq. 2.9, is diagonalized for better illustration of its relation to anisotropy as

$$\mathbf{H} = \mathbf{R}\Lambda\mathbf{R}^T, \tag{2.11}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of the eigenvalues of $\mathbf{H}$ and $\mathbf{R}$ is the normalized matrix of the right eigenvectors. Note that since $\mathbf{H}$ is symmetric, it always has real eigenvalues. A discrepancy that often exists in the eigenvalues as well as the directions of the eigenvectors will be used to define the mesh anisotropy in the next section.

Only positive eigenvalues of the matrix are meaningful in space. Consequently the Hessian is modified by using its singular values. Because $\mathbf{H}$ is a real symmetric matrix, it is normal and its singular values are the absolute values of its eigenvalues. This results in:

$$\left|\mathbf{H}\right| = \mathbf{R}\left|\mathbf{\Lambda}\right|\mathbf{R}^T \tag{2.12}$$

where $\left|\mathbf{\Lambda}\right|$ denotes the matrix containing the singular values of $\mathbf{H}$ and $\left|\mathbf{H}\right|$ is the companion matrix obtained by the similarity transformation in Eq. 2.12.

## 2.3 Length-Based Grid Adaptation

In the current approach, the adaptation procedure is defined as one in which the differences among the errors along all edges are minimized. Ideally the errors are equidistributed over the edges of the elements. This criterion, which states that the errors are equal or nearly equal, in one dimension can be written as

$$h_c^2 \left| \frac{d^2 f^h}{dx^2} \right|_c \cong C, \tag{2.13}$$

where $C$ denotes a positive constant.

In multi-dimensional equations, when $\Delta\mathbf{x}$ represents the vector of the edge between node *I* and *J*, the above criterion can be written as

$$\Delta\mathbf{x}^T \left|\mathbf{H}\right| \Delta\mathbf{x} \cong C \,. \tag{2.14}$$

Here $\Delta\mathbf{x} = \mathbf{x}_J - \mathbf{x}_I$, $\mathbf{x}_I$ and $\mathbf{x}_J$ are the coordinates of nodes *I* and *J* respectively and $\left|\mathbf{H}\right|$ is given in Eq. 2.12. According to the definition in Eqs. 2.10 and 2.12, both **H** and $\left|\mathbf{H}\right|$ are symmetric. When $\left|\mathbf{H}\right|$ is replaced by its eigenvalues and eigenvectors, this is

$$\Delta\mathbf{x}^T \mathbf{R} \left|\mathbf{\Lambda}\right| \mathbf{R}^T \Delta\mathbf{x} \cong C \tag{2.15}$$

Note that this can be written as,

$$\Delta\mathbf{x}^T \mathbf{R} \left|\mathbf{\Lambda}\right|^{1/2} \left|\mathbf{\Lambda}\right|^{1/2} \mathbf{R}^T \Delta\mathbf{x} = \Delta\mathbf{x}^T \mathbf{T}^T \mathbf{T} \Delta\mathbf{x} \cong C \,. \tag{2.16}$$

where transformation matrix, **T**, has been defined as $\mathbf{T} = \left|\mathbf{\Lambda}\right|^{1/2} \mathbf{R}^T$. The criterion now can be written as

$$\Delta\tilde{\mathbf{x}}^T \Delta\tilde{\mathbf{x}} = \left|\Delta\tilde{\mathbf{x}}\right|^2 \cong C \,. \tag{2.17}$$

where $\Delta\tilde{\mathbf{x}}$ is the vector transformed from $\Delta\mathbf{x}$ in the Euclidean space with the matrix, **T.** This new space is called a Riemann space. The current adaptation is designed to equidistribute edge lengths in the Riemann space, which are Riemann lengths. The operation of the transformation matrix **T** on each of the eigenvectors of $\left|\mathbf{H}\right|$ results in multiplying them by the square root of the corresponding eigenvalues. For example, when an ellipse whose semi-major and semi-minor axes are the eigenvectors of $\left|\mathbf{H}\right|$, and

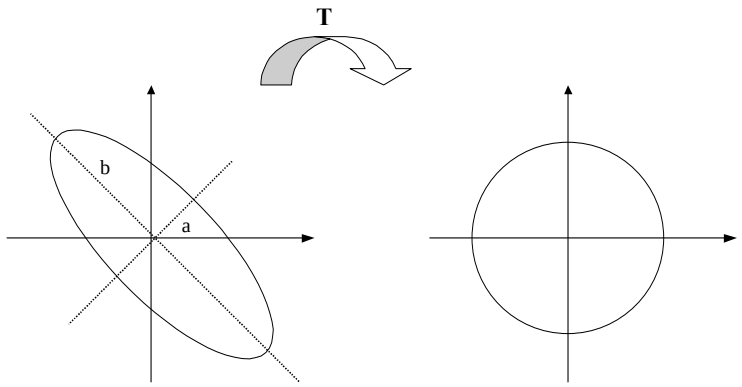



*Figure 2.2 Transformation of an Ellipse by* **T** *where* $a = |\lambda_1|^{-1/2}$ *and* $b = |\lambda_2|^{-1/2}$.

their lengths are the reciprocals of the square roots of singular values of $\mathbf{H}$, is transformed with **T**, it results in a unit circle in the Riemann space, as shown in Fig. 2.2.

A more convenient way is to define a Riemannian metric. The Riemann length of an edge $[\mathbf{x}_I, \mathbf{x}_J]$ in this metric is defined by:

$$l^R(\mathbf{x}_I, \mathbf{x}_J) = |\Delta\tilde{\mathbf{x}}| = \int_{\mathbf{x}_I}^{\mathbf{x}_J} \sqrt{d\mathbf{x}^T |\mathbf{H}|(\mathbf{x}) d\mathbf{x}} \tag{2.18}$$

Note that the variation of the Hessian along the edge is considered by the integration. Actually in real calculations an approximation is used by averaging the Hessian along the edge, for example, from node *I* to *J*, in this way,

$$l^R(\mathbf{x}_I, \mathbf{x}_J) = \sqrt{(\mathbf{x}_I - \mathbf{x}_J)^T |\tilde{\mathbf{H}}| (\mathbf{x}_I - \mathbf{x}_J)} \tag{2.19}$$

where $|\tilde{\mathbf{H}}|$ is the average Hessian with positive-definite eigenvalues.

The error estimate developed in this chapter will be used in the current approach. In the form of either a Riemannian metric or a transformation matrix, the criterion in the

adaptation is to equidistribute some manipulated length of the edge. This length originates from the regional error analysis along the edge developed for the 1-D problem, and a mesh with equal metric edge lengths corresponds to equal errors along the edges. The reason the term “feature indicator” is used, is because there are so many kind of errors in the solution and so many different approaches in the evaluation of the these errors, although only one of these is discussed in this chapter.

In the following chapters mesh optimization with the feature indicator is detailed and its weakness is also revealed.

# Chapter 3

# Anisotropic Mesh Adaptation Methodology

Mesh manipulations on the grid according to the feature indicator in the combined methods are detailed in this chapter. Although the initial mesh might be far from the optimal choice for the fluid dynamics solution, the purpose of grid adaptation is to drive the mesh toward the optimum. A second goal is that as the adaptation proceeds that the mesh approaches a stationary condition which indicates that the mesh adaptation has converged. The adaptation is converged when no further manipulation on the mesh is necessary, based on the prescribed criteria of the adaptation function.

## 3.1 Procedures to Drive Edge Metric Lengths Equal

In the last chapter, a feature indicator, introduced as the error along an edge, has been outlined. As noted before, the feature indicator can be used either in the form of a transformation matrix which transforms the Euclidean space to a Riemann space and calculates the edge length in that space or in the form of a Hessian metric which may be used to calculate the Riemann edge length explicitly. The grid adaptation procedure that

is used in the current approach is mathematically defined as one that will minimize the difference among the metric lengths of all edges in the field, such that the criterion of equal or nearly equal edge lengths is met. Operations on the mesh in order to meet the adaptation criterion include node enrichment/removal and edge swapping. In addition, point movement is allowed in the present study, but it is limited to node smoothing. These operations and their implementations are detailed below together with the reconnecting strategy that is required to rebuild the local connectivity removed by both node removal and edge swapping.

### 3.1.1 Node Enrichment and Its Implementation

Node enrichment is a procedure whereby additional nodes are added to the mesh. In the node enrichment process a new node is placed at some intermediate point of the longest edge in the mesh. The edge length used for determining this longest edge is the metric length. The specific procedure used to implement the node enrichment step is one that is analogous to that used for grid adaptation by many earlier researchers [3,4,5,8,9]. The longest edge is first identified and then split into two shorter edges by placing a new point in the middle, as seen in Fig. 3.1. In two dimensions, the neighboring triangles are also divided into two smaller ones, while volumes as well as faces around the edge are split in three dimensions.

Identification of the longest edge that is going to be split first is necessary before the node enrichment process. To facilitate the location of the longest length of all edges in the field, a heap data structure [43] is maintained. A heap is a kind of data structure

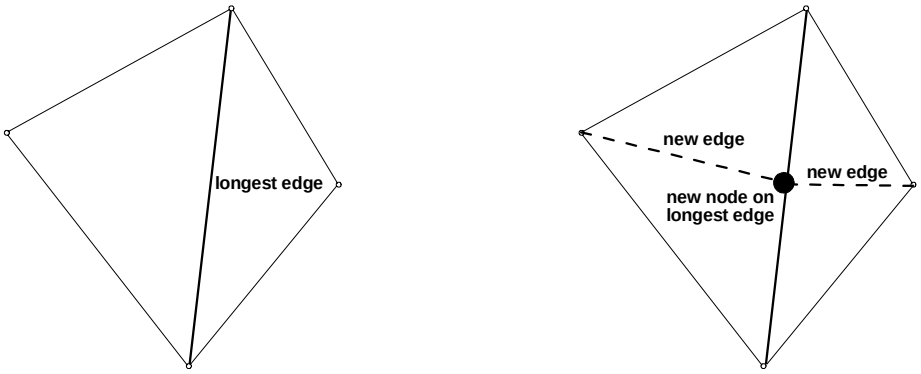



*Figure 3.1 Node Enrichment*

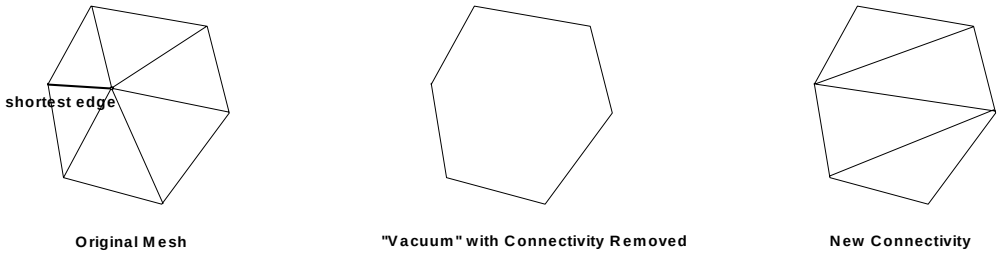



*Figure 3.2 Node Removal*

where the element with the smallest/largest key is placed on the top. A detailed description of the data structures in the method is given in the next chapter.

### 3.1.2 Node Removal and Its Implementation

Node removal removes the short edges in the mesh. When the shortest edge is identified, the node on one end of it is removed to create a “vacuum” in Fig. 3.2. Local reconnection is then used to construct a local connectivity inside this “vacuum”. A unique reconnection procedure based on criteria from the advancing-front method [5],

described in section 3.1.4 of this chapter, is used to remesh the "vacuum" created following node removal.

One other approach attempted for node removal is edge collapsing (Dompierre *et al*. [3]), which collapses the shortest edge by merging its two end points. The edge collapsing approach often leads to overlapping, especially in highly stretched meshes, while the reconnection approach can remove almost any edge in the mesh. Since the goal of the procedure is to make all the edge lengths nearly equal, this local reconnection procedure minimizes potential difficulties. A similar heap data structure is maintained in the node removal procedure in order to locate the shortest edge.

### 3.1.3 Edge Swapping and Its Implementation

In two dimensions edge swapping is used to break the edge between two triangles and connect the remaining two diagonals of the quadrilateral, if the two new triangles are closer to the equal-metric-length criterion than the previous two as seen in Fig. 3.3. Extension to three dimensions is similar and will be discussed in Chapter 9. Edge swapping is a complement to the above node insertion/removal operations in circumstances where these two operations are not effective. For instance, there are usually an upper and a lower edge length thresholds for the insertion and removal procedures respectively. A point is neither inserted to nor removed from the edge whose length is between the two thresholds though not close to the average length. This is to preclude infinite cycles, such as the removal of a new point which has just been inserted. This edge swapping procedure as well as the point smoothing procedure in next section improves the "quality" of the mesh resulting from node enrichment and removal in such situations. Here the quality of a mesh is defined as the extent to which its edge lengths

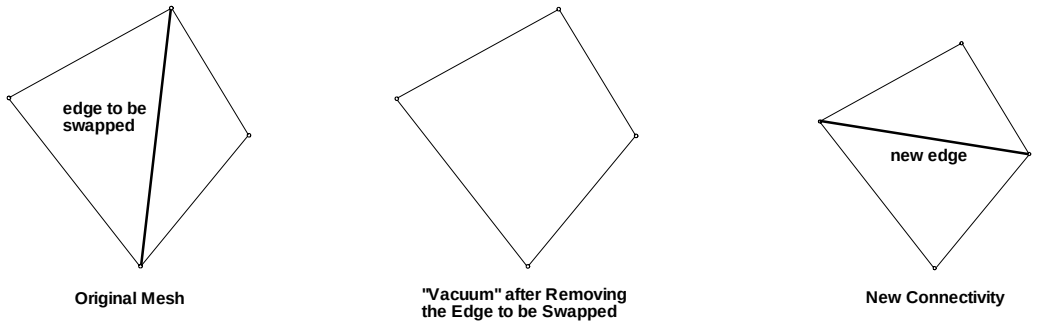



*Figure 3.3 Edge Swapping*

meet the equal-metric-length criterion. Edge swapping is widely used in both mesh generation and adaptation and has proven to be an efficient way to improve mesh quality [5].

Rather than implementing a distinct procedure for swapping, the advancing-front method used for the node removal procedure is also used to accomplish this effect. It is done by sweeping over all edges and checking the triangles on either side. During this checking procedure each edge is removed and the vacuum left is remeshed in the manner according to the advancing-front criteria. The original connectivity is kept track of so that if no change is made, the connectivity information is not updated.

Although the edge swapping procedure has the capability to equalize the lengths of edges, it is rather limited because of geometric constraints. For example, in Figure 3.4, the edge connecting *A* and *B* cannot be swapped because it will intersect the existing connectivity.

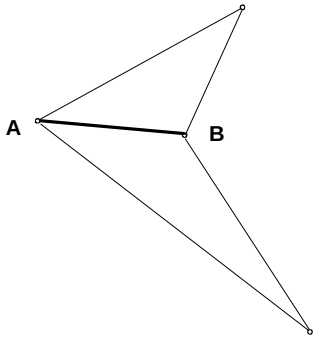



*Figure 3.4 Non-Swappable Connectivity*

### 3.1.4 Local Reconnecting with Advancing-Front Method

A revised version of the advancing front method discussed in Chapter 1 is used for local reconnection in the current approach. This reconnection procedure does not involve adding new points to the current mesh, although this is the general step of the advancing front method where it is used for grid generation. We also note that the characteristics of grid anisotropy must be accounted for in the modified advancing front method.

The steps invoked in the creation of the new connectivity inside the vacuum left by node removal and edge swapping are:

1. Select the shortest (metric) edge ( $M$ ) on the boundary of the hole (with ends $A$ and $B$, refer to Fig. 1.2) as the initial starting position. This has proven to be of special advantage in anisotropic mesh generation [8].

2. From all remaining points pick as the set of usable points those that can form a new triangle with edge ( $M$ ) without intersecting the existing mesh.

3. From the matrices in the background mesh, interpolate the local Hessian from which the local transformation matrix **T** can be derived by manipulating eigenvalues and corresponding eigenvectors, as in Chapter 2. Set the middle of the edge ($M$) picked in Step 1 as the origin and transform all usable points in Step 2 to the Riemann space.

4. For each usable point in the set (Step 2), a circle is constructed in Riemann space, which passes through A, B and the usable point. This is the same as in the standard advancing-front method (except that the circle is constructed in Riemann space), and the ideal point is the one whose circle has the longest distance from its center to an ideal point in the direction from the ideal point to the edge.

5. Construct the new triangle and a new front if it exists. Repeat Step 1 until no new front can be created.

### 3.1.5 Adaptation Function and Point Movement

As noted before, the grid adaptation procedure is mathematically defined as one that will minimize the difference among the metric lengths of all edges in the field. For the convenience of defining this adaptation function, a node-by-node basis consistent with the local Riemann space, is adopted. To provide an expression for measuring the amount by which the edges in a given mesh fails to meet an (nearly) equal metric edge length criteria, the average length of all edges, $\bar{l}_k^R$, surrounding any node $k$ is defined as

$$\bar{l}_k^R = \frac{1}{N_k} \sum_{i=1}^{N_k} l_i^R \tag{3.1}$$

where $N_k$ represents the number of edges that meet at point $k$. Using this definition of the average edge length at each mesh point, an adaptation function, $E_k(\mathbf{x})$, is then defined by summing the squares of the difference between the length of each individual edge and the average length for node $k$,

$$E_k(\mathbf{x}) = \sum_{i=1}^{N_k} \left( l_i^R - \bar{l}_k^R \right)^2 \tag{3.2}$$

The final adapted grid is obtained by minimizing this adaptation function over all nodes in the domain taking into account grid connectivity and node location. In the converged limit when the grid adaptation has been completed, the adaptation function at all nodes in the field will have (ideally) reached the minimum simultaneously. It is also expected that on the final mesh the differences among all edge lengths will be small such that the criterion of equal or nearly equal edge length is met.

While node removal/insertion and edge swapping operations add or remove points and optimize the connectivity to minimize the adaptation function, a smoothing process derived from the point movement equation below is used as a subsidiary procedure to further improve the mesh quality. Point movement has the potential to be the most important step in grid adaptation. It is identical in complexity in both two and three dimensions and can be accomplished by identical algorithms. Unfortunately, implementation of an effective point-movement algorithm is quite challenging and attempts to design such algorithm have been made for a long time, both in structured and unstructured grids. The spring analogy, which replaces the edges with springs, has been adopted in many references [13,50,5]. The position of the vertex is determined by balancing the forces on each of the edges (springs) around it, while the stiffness of the

edges is specified according to the edge length. The results, most of which have been obtained in structured grids, have been promising in applications such as free surface flow (Slikkeveer *et al* [44]), fluid-structure interaction (Blom *et al* [45]), and flow moving boundaries (Farhat *et al* [46] and Piperno [47]). Although the spring analogy has also been applied to unstructured grid adaptation [48,49,50], it has the disadvantage that the principle of it is to balance the force, not to equidistribute the edge lengths. The derivation of the equation below for point movement is purely for equalizing the metric lengths. Note that the results in the present work do not use point movement. This step is reduced to a smoothing procedure to complement the insertion, removal and swapping routines presented above. Here the general point movement problem is briefly summarized, and the approximate smoothing method is described.

A point movement equation can be obtained directly from the deviation function. This has the advantage of using two complementary steps to accomplish the same (edge length equalization) objective. The minimization of the function, $E_k$, proceeds in standard fashion by setting the partial derivatives of $E_k$ (Eq. 3.2) with respect to $\mathbf{x}_k$ to zero. This gives a nonlinear algebraic system for the new location of the node. Using primes to denote the gradient, the resulting system of equations becomes,

$$\begin{aligned}&\sum_{i=1}^{N_k}\left(l^R(\mathbf{x}_i,\mathbf{x})-\overline{l^R(\mathbf{x}_i,\mathbf{x})}\right)\cdot\left(l^R(\mathbf{x}_i,\mathbf{x})'-\overline{l^R(\mathbf{x}_i,\mathbf{x})}'\right)\\&=\sum_{i=1}^{N}\left(l^R(\mathbf{x}_i,\mathbf{x})-\overline{l^R(\mathbf{x}_i,\mathbf{x})}\right)\cdot l^R(\mathbf{x}_i,\mathbf{x})'-\sum_{i=1}^{N}\left(l^R(\mathbf{x}_i,\mathbf{x})-\overline{l^R(\mathbf{x}_i,\mathbf{x})}\right)\cdot\overline{l^R(\mathbf{x}_i,\mathbf{x})}'\end{aligned}\tag{3.3}$$

In the above equation, $l^R(\mathbf{x}_i,\mathbf{x})$ is the Riemann length of the edge connecting *i* and the node to be moved, $N_k$ is the number of edges around the node, and $\overline{l^R(\mathbf{x}_i,\mathbf{x})}$ is

the average lengths of these edges. Obviously the second term of the above equation is zero. The metric length is approximated as the Euclidean length times a coefficient in scale,

$$l^R(\mathbf{x}_i, \mathbf{x}) = k_i \cdot \|\mathbf{x}_i - \mathbf{x}\| \tag{3.4}$$

This is a fair approximation and accurate in isotropic grids, but becomes less appropriate in anisotropic cases where large discrepancies exist between eigenvalues. A justification of such approximation will be made later together with the results.

The derivative of Equation 3.4 with respect to $\mathbf{x}$ is,

$$l^R(\mathbf{x}_i, \mathbf{x})' = -\frac{k_i \cdot (\mathbf{x}_i - \mathbf{x})}{\|\mathbf{x}_i - \mathbf{x}\|} \tag{3.5}$$

The equation is now in the form,

$$\sum_{i=1}^{N_k} \left( l^R(x_i, \mathbf{x}) - \overline{l^R(\mathbf{x}_i, \mathbf{x})} \right) \cdot \frac{k_i (\mathbf{x}_i - \mathbf{x})}{\|\mathbf{x}_i - \mathbf{x}\|} = 0 \tag{3.6}$$

Solving this equation directly with Newton-type iteration is, unfortunately, unstable because of the lack of diagonal dominance. It is then cast into the form,

$$\sum_{i=1}^{N_k} l^R(\mathbf{x}_i, \mathbf{x}) \cdot \frac{k_i (\mathbf{x}_i - \mathbf{x})}{\|\mathbf{x}_i - \mathbf{x}\|} = \sum_{i=1}^{N_k} \overline{l^R(\mathbf{x}_i, \mathbf{x})} \cdot \frac{k_i (\mathbf{x}_i - \mathbf{x})}{\|\mathbf{x}_i - \mathbf{x}\|} \tag{3.7}$$

So the right hand side serves as a source term, and the equation can be solved iteratively as,

$$\mathbf{x}^{new} = \mathbf{x}^{old} + \omega \frac{\sum_{i=1}^{N_k} \left( l^R(\mathbf{x}_i, \mathbf{x}^{old}) - \overline{l^R(\mathbf{x}_i, \mathbf{x}^{old})} \right) \cdot \frac{k_i (\mathbf{x}_i - \mathbf{x})}{\|\mathbf{x}_i - \mathbf{x}\|}}{\sum_{i=1}^{N_k} k_i^2} \tag{3.8}$$

where $\omega$ is an under relaxation parameter.

Although applying this expression to all points in the domain appears straightforward, there are several mitigating circumstances that make it problematic. Primary among them is that this minimization sometimes causes the point, $\mathbf{x}$, to move to infinity. For example, since equalizing edge lengths is equivalent to fitting some sort of "average" sphere (circle) through the surrounding points and finding $\mathbf{x}$ as the center of this sphere, it is clear that three co-linear points will generate a circle whose center is at infinity.

### 3.1.6 Mesh Smoothing

Due to the difficulty in point movement, it is used only as a smoothing procedure which moves the grid points slightly to improve the local mesh quality in the current study. A point is allowed to move within the shell formed by the surrounding nodes. The under relaxation parameter $\omega$ is chosen to ensure that the point movement never entangles the connectivity. In practice, $\omega$ is initially set to one when a point is to be moved. It is continuously reduced by half until the new location is within the shell.

## 3.2 Sequencing the Process

Node enrichment and removal serve as the major procedures for eliminating the longest and shortest edges in the domain. Edge swapping is then used to improve the connectivity which can be distorted either because of node enrichment or removal. The

smoothing procedure further improves the grid locally. These procedures are applied iteratively several times during the solution procedure.

The complete adaptation process is performed by iterating the following operations:

1 Swap all the edges until convergence.

2 Remove edges whose length is below a specified threshold (usually one-half the average length of all edges) by the following sub-procedure:

2a Compute the average length of all edges in the field. Construct a heap data structure (discussed in next chapter), with the shortest edge at the top.

2b Remove the shortest edge (assuming that it satisfies the threshold) from the heap. Delete one of its nodes and define appropriate changes in connectivity. Update the heap structure.

2c Repeat 2b until a specific portion of the edges are removed (usually 10%).

2d Edge swap until the remaining triangles in the field satisfy the quality criterion.

2e Perform one sweep of the point smoothing equation throughout the entire field.

2f Repeat 2b through 2e until all edges below the threshold in Step 2 have been removed.

3 Perform edge swapping and one sweep of smoothing alternately on the whole mesh for three cycles.

4 Divide all edges whose lengths are above a threshold value (usually twice the average lengths of all the edges) by the following sub-procedure:

4a Compute the average length of all edges in the field. Construct a heap data structure, with the longest edge at the top.

4b Place a new node at the mid-point of the longest edge in the heap and form connections to all pertinent nodes.

4c Repeat 4b until a specific portion (usually 10%) of the edges are removed.

4d Edge swap until all triangles in the field satisfy the quality criterion.

4e Perform one sweep of the point smoothing equation throughout the entire field.

4f Repeat 4b through 4e until all edges are below the threshold in Step 4 or a pre-defined number of nodes has been added. This latter condition controls the total number of nodes.

5 Perform edge swapping and point smoothing alternately on the whole mesh for three cycles as in Step 3.

6 Repeat Steps 2 through 5 until no further edge is removed or divided.

7 Identify any critical points that cannot be removed due to boundary conditions and escape from infinite iterations.

# Chapter 4

# Data Structures in Mesh Adaptation

A mesh is a network of nodes, and the information on the connectivity is very important. The storage of the connectivity in the adaptation algorithm must not only consider the allocated memory size, but must also take into account the changes in connectivity brought by the combined methods used in the current approach.

## 4.1 Storage of Connectivity in Adaptation

An effective data structure for a grid adaptation procedure should meet the following contradictory requirements:

1. The data structures should provide a rapid identification of any connectivity needed during adaptation. In general, connectivity access time can be decreased by increasing the amount of connectivity information that is stored.

2. Storage should be minimized so that a change of the local connectivity involves minimal operations on the information stored. Fewer operations are necessary to alter the connectivity if less information is stored.

3. Data structures should be effective, considering the various connectivities in the mesh. For instance, if the number of edges around one node varies from point to point an array of fixed size will not be efficient for storing the connectivity. The same situation appears in three-dimensional unstructured grids where the number of faces adjacent to any edge is not always the same.

The first requirement is to store as much information as possible while the latter two state that information storage should be minimized and effective. These three requirements for the connectivity storage obviously compete with each other. An important point to note is adaptation will typically require more information about the local connectivity than the solver requires. For example, in the CFD solver identifying the edges around one node is usually not necessary, while in adaptation this connectivity is of the utmost importance, since it makes point smoothing and node removal more economical. A smaller amount of stored information makes the recovery of the local information difficult and thus increases the CPU time of the adaptation procedure.

The compromises made in the current approach result in the storage of the following information for two-dimensional meshes. As in all CFD data structures, each node, edge, face and cell in the domain must be given a unique identifier to distinguish them from all other elements. Once these elements are identified, the information below is associated with them.

1. Node information

   The coordinates and identity of each node, the identity of the edges containing the node and identity of the first neighbor nodes surrounding the given node

2. Edge information

   The identity of the end points of each edge, the identity of the cells adjacent to the edge

3. Cell information

   The identity of the vertices of each cell

The above connectivity information provides a reasonable balance between storage size and computing time.

## 4.2 AVL Binary Tree Structures in Storage of Connectivity

Most of the connnectivity information can be stored in an array of fixed size, since the size of these components is fixed. For example, the number of vertices of a cell is fixed if a single type of cell exists on the mesh. Similarly the number of cells adjacent to an edge in two dimensions (to a face in three dimensions) is always two. The only connectivity parameter whose number of elements is not a constant in two dimensions is the number of edges around a given node. In three dimensions the number of faces adjacent to an edge is also not constant. In the current approach, this kind of information is stored in an AVL binary tree [43].

Binary trees provide the basis for many searching algorithms, including the AVL binary tree to be presented here. It is therefore necessary to begin by introducing some basic concepts and terminology related to binary tree structures. More detailed expositions can be found in [43].

### 4.2.1 Definition and Terminology of Binary Tree

Tree structures provide a systematic way of sorting a collection of data items which not only enables quick access to the information stored, but also enables efficient insertions and deletions of items. This degree of flexibility requires the storage of data items in non-sequential locations of computer memory. To achieve such a data structure, each data item is extended by the addition of two integer values, known as the left and right links, stored in what is known as a node of the tree. Each added link will either be set to zero or to the position in memory where another node of the tree can be found. Hence, from any one node of the tree it is possible to reach at most two other nodes. Moreover, in order to ensure that every node can reached, these links must be such that for each node except one, known as the root, there is one and only one link pointing at it, as A in Fig. 4.1. This definition of a tree establishes a hierarchy of nodes: at the top level of the hierarchy the root points to zero, one or two nodes at the level below. Each of these in turn points to zero, one or two other nodes at the next level of the hierarchy; and so forth. This hierarchical structure leads to the graphic representation shown in Figure 4.1 for a simple tree comprising only eight nodes {A, B, C, D, E, F, G and H}.

Genealogical terms are normally used to describe the relative position of nodes in a tree: when one node points to a second node, the former is called the parent of the latter, and the latter the child of the former node. A node without children, that is, with both links blank, is called a terminal node. The only node without a parent is the root (node A in Fig. 4.1). For any given node, the set of nodes formed by the node itself together with all its descendants constitutes a subtree of the main tree. For instance, in

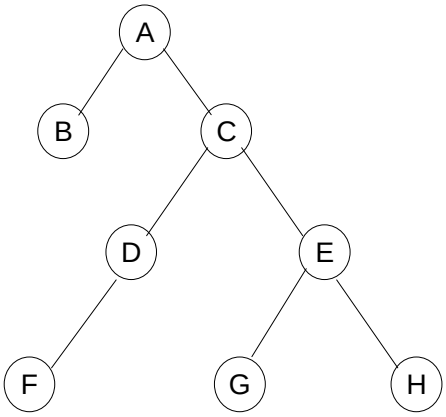



*Figure 4.1 A Simple Binary Tree*

Figure 4.1 the sets of nodes {C, D, E, F, G and H} and {E, G and H} are subtrees of the main tree rooted at C and E respectively.

### 4.2.2 Binary Tree Traversal

To retrieve the information stored in a given node requires knowledge of its location in memory which is kept by its parent. Hence, a node in the tree can only be examined or visited if all its ancestors are visited first. However, it is possible to systematically examine each node in such a way that every node is visited exactly once. Such an operation is known as traversing the tree and provides the basis for the searching method discussed below. Although several algorithms for traversing a binary tree can be found in the literature [43,51], attention will be centered here on the so-called pre-order traversal method [43]. This technique is embodied in the following three steps:

1. Visit the root of the current subtree

2. If the left link of the root is not zero then traverse the left subtree

3. If the right link of the root is not zero then traverse the right subtree

The procedure determined by these three steps is clearly recursive, that is step 2 and step 3 invoke again the algorithm that they define. In order to illustrate this process, again consider the tree shown in Figure 4.1. For this tree, the repeated application of the above algorithm yields the following sequence:

1 Visit A
2 Traverse the tree {B}
  2.1 Visit B
  2.2 Skip
  2.3 Skip
3 Traverse the tree {C, D, E, F, G, H}
  3.1 Visit C
  3.2 Traverse the tree {D, F}
    3.2.1 Visit D
    3.2.2 Traverse the tree {F}
      3.2.2.1 Visit F
      3.2.2.2 Skip
      3.2.2.3 Skip
    3.2.3 Skip
  3.3 Traverse the tree {E, G, H}
    3.3.1 Visit E
    3.3.2 Traverse the tree {G}
      3.3.2.1 Visit G
      3.3.2.2 Skip
      3.3.2.3 Skip
  3.3.3 Traverse the tree {H}
      3.3.3.1 Visit H
      3.3.3.2 Skip
      3.3.3.3 Skip

Thus, the nodes of the tree in Figure 4.1 in preorder are A, B, C, D, F, E, G and H. Additional details are given in Ref. 43. Traversing visits every element of a tree. If the identity of the edges surrounding one node is stored as the elements of a tree, these edges will be identified when this tree is traversed.

### 4.2.3 Inserting and Deleting Nodes

In order to add a new item to a binary tree, a node containing the new item of information must be created and stored with its left and right links set to zero. If the current tree is empty, the new node becomes the root of the tree, otherwise the node must be inserted or linked to the existing tree. To achieve this, the tree is followed downwards, starting from the root and jumping from parent to child, until a blank link is found. This link is then set to the memory position of the new node. When moving down the tree, a criterion must be provided at each node to choose between the left or right branches. This criterion determines the final position in the tree of the new node and, consequently, the shape of the tree itself.

Deleting a node from a binary tree is a straightforward operation if the undesired node is a terminal node. The only change necessary is to set the corresponding link of its parent to zero. In the case of an intermediate node, the process becomes slightly more complicated since a gap cannot be left in the tree. To overcome this problem, the unwanted node is replaced by a terminal node chosen from among its descendants according to some pre-set criterion. This operation can be carried out by modifying the links to suit the new structure of the tree without moving the nodes from their memory positions. Figure 4.2 illustrates the deletion of node C from the tree shown in Figure 4.1 and its replacement by node H.

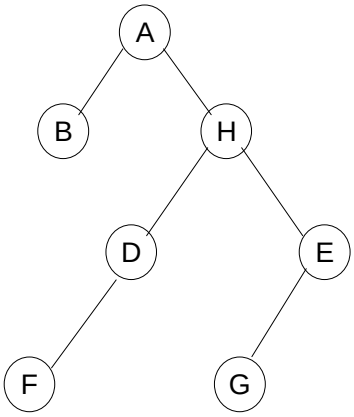



*Figure 4.2 Deletion of a Node from the Binary Tree*

### 4.2.4 AVL Search Tree

Binary trees are used to store the edges surrounding any given node. The search process, which is necessary for locating the edge for removal or any other operation, must be optimized such that the overall time consumption of the adaptation is minimized. The AVL search tree, named after Adelson- Velskii and Landis [52], is used to store the surrounding edges.

An AVL tree is a type of binary search tree that facilitates locating elements in the tree, as well as inserting and deleting nodes from the tree. A binary search tree may be empty, but if not, it satisfies the following properties:

- Every element has a key and no two elements have the same key.
- The keys (if any) in the left subtree are smaller than the key in the root.
- The keys (if any) in the right subtree are larger than the key in the root.
- The left and right subtrees are also binary search trees.

The difficulty with binary search trees is that while the average times for search, insertion, and withdrawal operations are all $O(\log_2 n)$, any one operation is still $O(n)$ in the worst case. This is so because one cannot say anything in general about the shape of the tree.

As an example, consider the two binary search trees shown in Figure 4.3 each of which contains the same set of keys with integer values from one to seven. The tree Ta is obtained by starting with an empty tree and inserting the keys in the following order

1, 2, 3, 4, 5, 6, 7.

The tree Tb is obtained by starting with an empty tree and inserting the keys in this order

4, 2, 6, 1, 3, 5, 7.

Clearly, Tb is a better search tree than Ta since visiting any node of it costs less than Ta. In fact, since Tb is a perfect binary tree, its height is $\log_2(n+1)-1$. Therefore, all three operations, search, insertion, and withdrawal, have the same worst-case asymptotic running time, $O(\log_2 n)$.

### 4.2.5 Implementing an AVL Tree

Inserting or removing an item is a two-part process in an AVL Tree. First, the item is inserted into the tree using the usual method for insertion in binary search trees. After the item has been inserted by attaching it to one of the terminals, it is necessary to

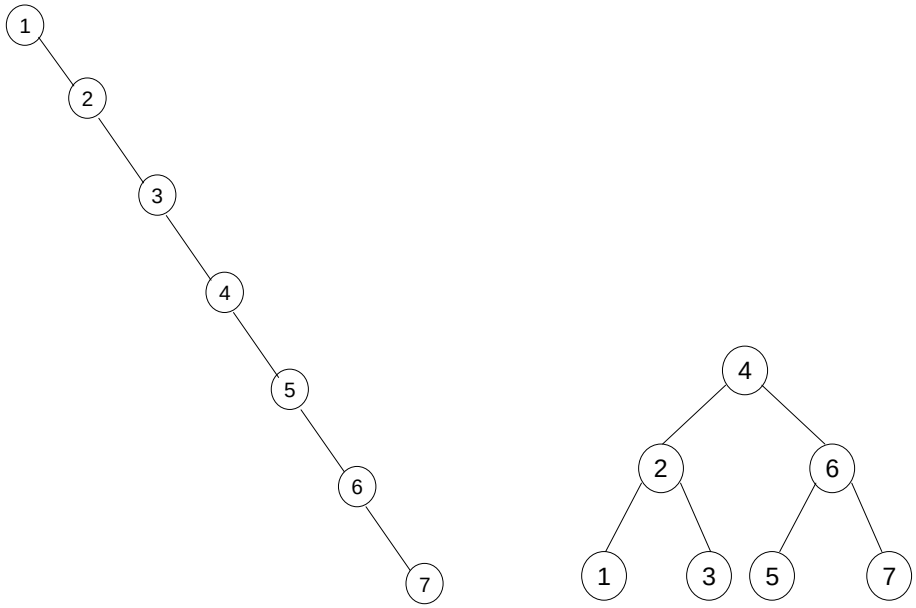



*Figure 4.3 Different Implementations of the Binary Tree*

check that the resulting tree is still AVL balanced (i.e., like Tb rather than Ta in Fig. 4.3) and to balance the tree when it is not.

When an AVL tree becomes unbalanced, it is possible to bring it back into balance by performing an operation called a rotation. For a general situation there are only four cases to consider and each case has its own rotation. These four cases include Left Left rotation(LL), Right Right(RR) rotation, Left Right(LR) rotation and Right Left rotation(RL). A detailed description of the four cases is given in Ref. 43.

## 4.3 Fibonacci Heap

Two major operations in the current approach are node insertion and node removal. In these operations, all edges in the field are sorted according to their lengths,

either ascendant or descendant. It is not economical to sort the edges every time an insertion/removal process is invoked. Similarly a basic data structure like a link-list is not appropriate for sorting edges, since the node insertion/removal process will produce edges with altered lengths. The insertion of such edges into the link-list takes an amount of time of *O(n)*, where *n* is the number of total edges [43]. A more efficient data structure, a Fibonacci heap, which always places the longest/shortest edge on the top and reduces the time to $O(\log_2 n)$, is maintained during the node removal/insertion.

A heap is a data structure where the item of top priority is placed on top. In node removal, the edge with the shortest length is placed on the top while the longest is on the top in node insertion. The Fibonacci heap implemented in the current work has the advantage of supporting all operations of the tree items that result from the operations on the edges during the node removal/insertion process.

Here the operations in the node removal and insertion processes are summarized. Before the node enrichment/removal process, edges are inserted into the heap, resulting in placing the longest/shortest edge on top. Thus an *Insertion* is needed here. During the process of node removal/insertion, locating the edge with longest/shortest length requires the *DeleteMin* operation. When one node is removed, edges other than the longest around the chosen point will also be removed. The removal of any specific node other than the min node is accomplished through a *Delete* operation. This operation also accompanies procedures like point smoothing and edge swapping, when relevant edge lengths have been changed. The accompanying operation on the heap is the deletion of the nodes associated with these edges from the heap being maintained, followed by the insertion of edges back into the same heap.

A min Fibonacci heap, which supports operations necessary for the current work, is a collection of min trees. A min tree is a tree in which each of the elements has a key less than its parent. The tree in a Fibonacci heap can have any number of children. Those children form a link list while the parent has a pointer member, pointing to one of the children. If one node has no child, the pointer to its child will be set to empty.

Several operations on the heap in order to build and maintain the heap are explained below. More detailed descriptions of the Fibonacci heap are given in Ref. 43.

### 4.3.1 Insertion into a Fibonacci Heap

An element may be inserted into a Fibonacci heap by first putting it into a new node and then inserting this node into the circular list pointed at by *min* which points to the node with the smallest key. The pointer *min* is reset to this new node only if the heap is empty or the key of the new node is smaller than the key in the node pointed to by *min*. It is evident that these insertion steps can be performed in *O(1)* time.

### 4.3.2 DeleteMin

Assume that *min* is not empty and that *min* points to the node that contains the min element. This node is deleted from the circular list. The new Fibonacci heap consists of the remaining min trees and the submin trees of the deleted root.

Before forming the circular list of min tree roots, pairs of min trees that have the same degree (the degree of a node is the number of its children) are repeatedly

joined together. This min tree joining is done by making the min tree whose root has the larger key a subtree of the other (ties are broken arbitrarily). When two min trees are joined, the degree of the resulting min tree is one larger than the original degree of each min tree and the number of min trees decreases by one. The amortized (statistical average) time for the DeleteMin operation is $O(\log_2 n)$ [43], where $n$ is the number of nodes in the heap.

### 4.3.3 Deletion of Specified Element

The deletion of a known element is necessary as the removal of a point or swapping an edge will result in the removal of some relative edges. The operations on the Fibonacci heap are to delete the corresponding nodes in the heap. This capability for removing a specified node is the reason why the relatively complicated heap is used in the current approach. The procedure of deleting a node other than the min from the heap is, however, more involved. It requires the addition of more pointers for each node and a procedure, *cascading cut***,** in order to improve the structure of the heap left by the deleting the node. A detailed description of the procedure can be found in literature [43].

# Chapter 5

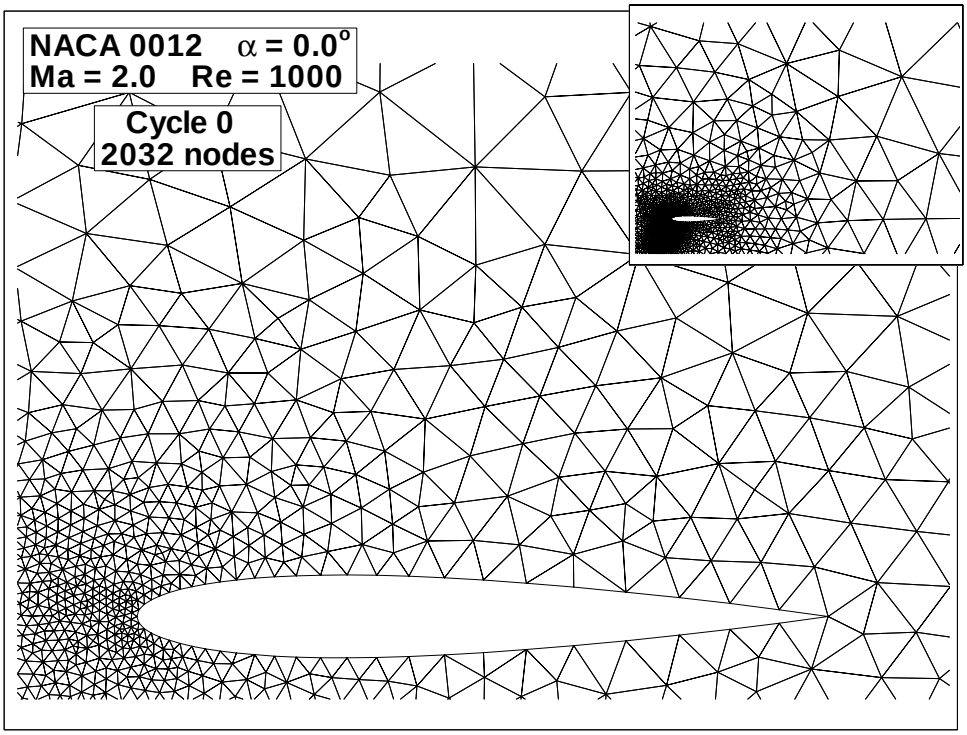
# Assessment of the Adaptation Approach

In order to assess the approach in the current work, adaptation is performed through a case demonstrated here in this chapter. The incorporation of node removal/insertion, edge swapping and grid smoothing is tested and their effectiveness for producing an anisotropic mesh according to the flow characteristics is demonstrated.

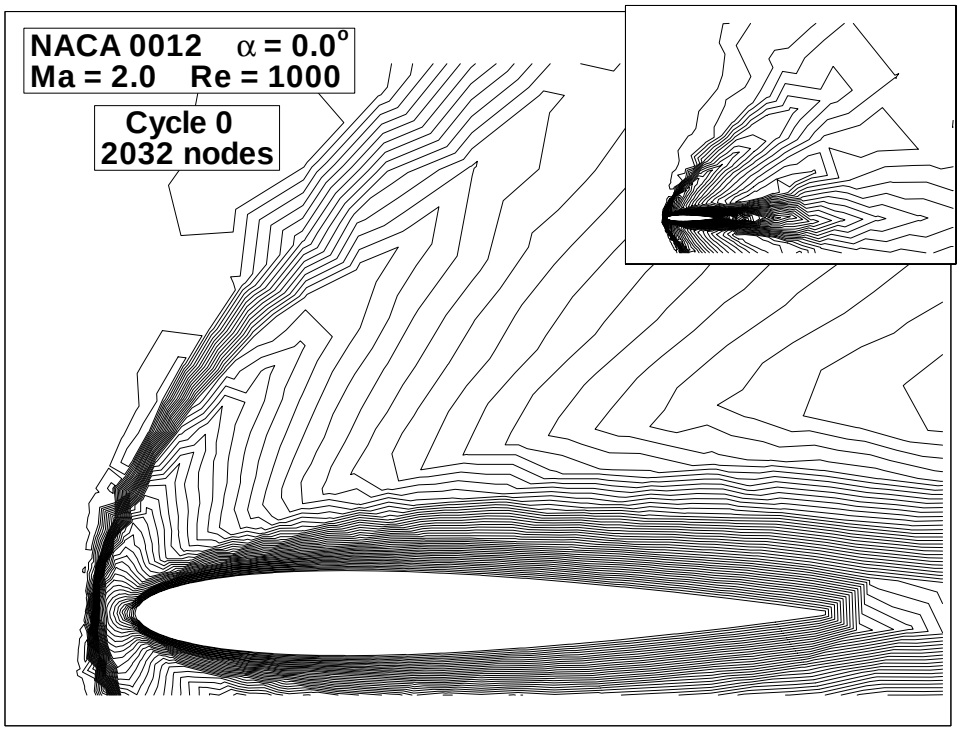
## 5.1 Supersonic Flow around NACA 0012

The example in this chapter for assessing the ability of current approach is the supersonic flow around a NACA 0012 airfoil at a Mach number of 2.0, with zero attack angle. To ensure steady solutions with laminar flow, the Reynolds number is set to 1000. The calculations in this chapter are done with FUN2D [39,53].

The initial grid for this calculation is shown in Fig. 5.1. The primary portion of part(a) shows the near-field mesh, while the plot in the top right of the figure shows the far-field view. This grid is generated by CFDRC-GEOM, a commercial grid generator [92]. As the figure indicates, the initial mesh, which contains 2032 nodes, is an isotropic




*(a)*




*(b)*

*Figure 5.1 Supersonic Flow around NACA 0012. (a) Initial Grid, (b) Solution*

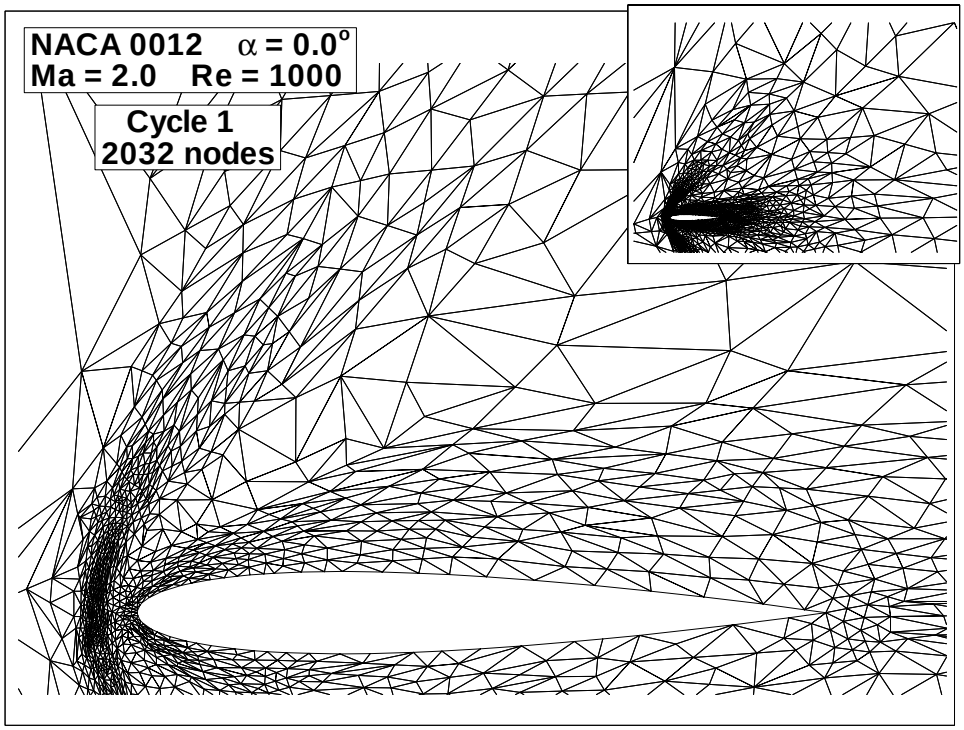
grid over the entire domain.

Clearly, the grid contains no provision for the boundary layer or the shock that are expected for viscous flow at a supersonic Mach number.

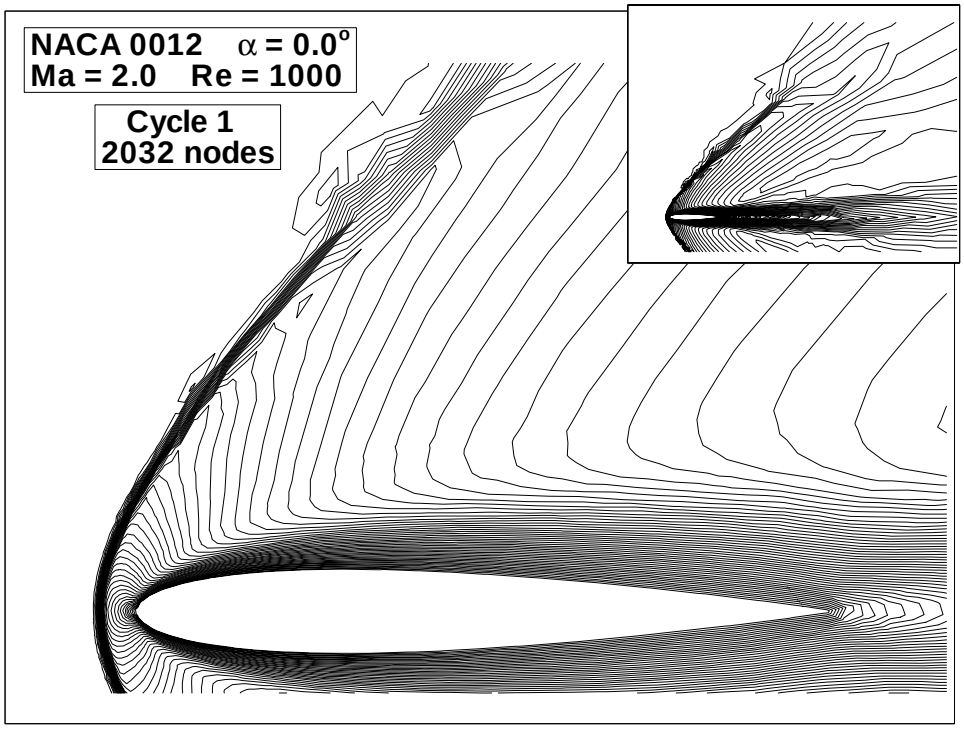
The converged Mach number contours obtained on this grid are shown in Fig. 5.1(b). The bow shock is visible, but is very diffuse and spread out while the shocks from the trailing edge are essentially non-existent. There is a semblance of a wake behind the airfoil and a wide, diffuse viscous region near the trailing edge of the airfoil. The flow is not well refined, with a coarse grid in both bow shock and boundary layer regions. This is clearly a very poor solution, but it is sufficient to initiate the adaptation procedure.

The refined mesh and Mach contours after one cycle of iteration are shown in Fig. 5.2. The Hessian is calculated with the Mach number, since it provides information for both a boundary layer and a shock wave. The number of points is frozen by inserting the same number of points as the number of points removed. It can be clearly seen that the mesh, though still coarse, has more points in the regions of the shock and the boundary layer. The bow shock is better refined on this initial refined mesh, but it remains quite diffuse. The boundary layer is reasonable considering the low Reynolds number chosen for this case. As can be seen from the top right plot that the regions of shock and boundary layer have attracted most of the grid points, but the wake grid remains very coarse.

Adaptation is furthered with the insertion of more points into the mesh, since the adaptation is not reasonable with the current number of points. The number of nodes has been increased to 5080. The results after two refinement cycles are given on Figure 5.3.

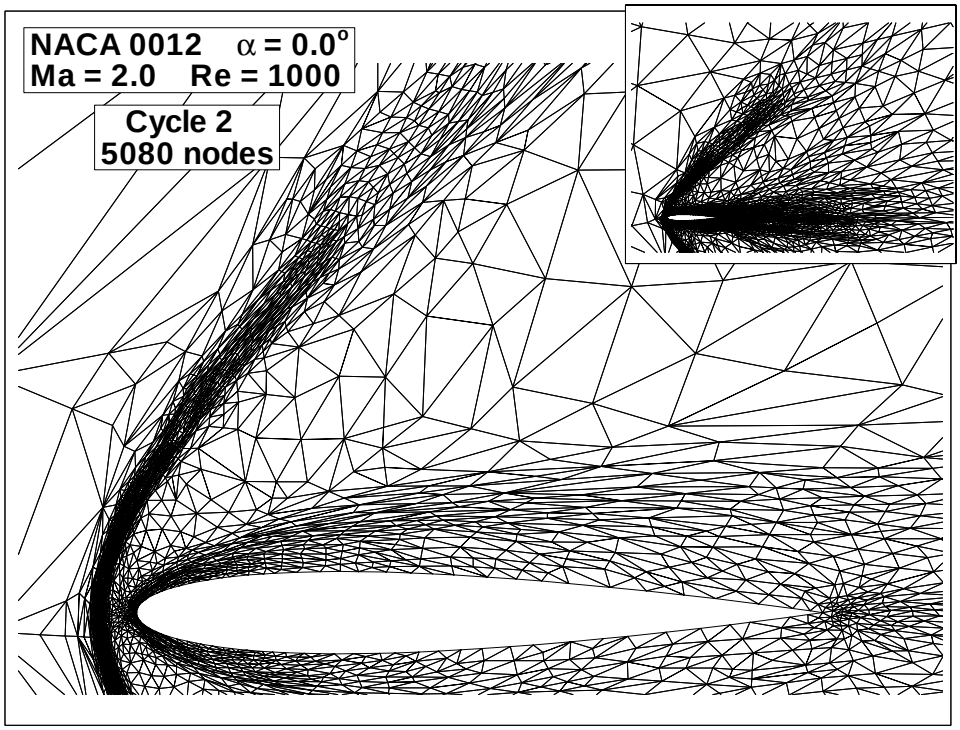



*(a)*

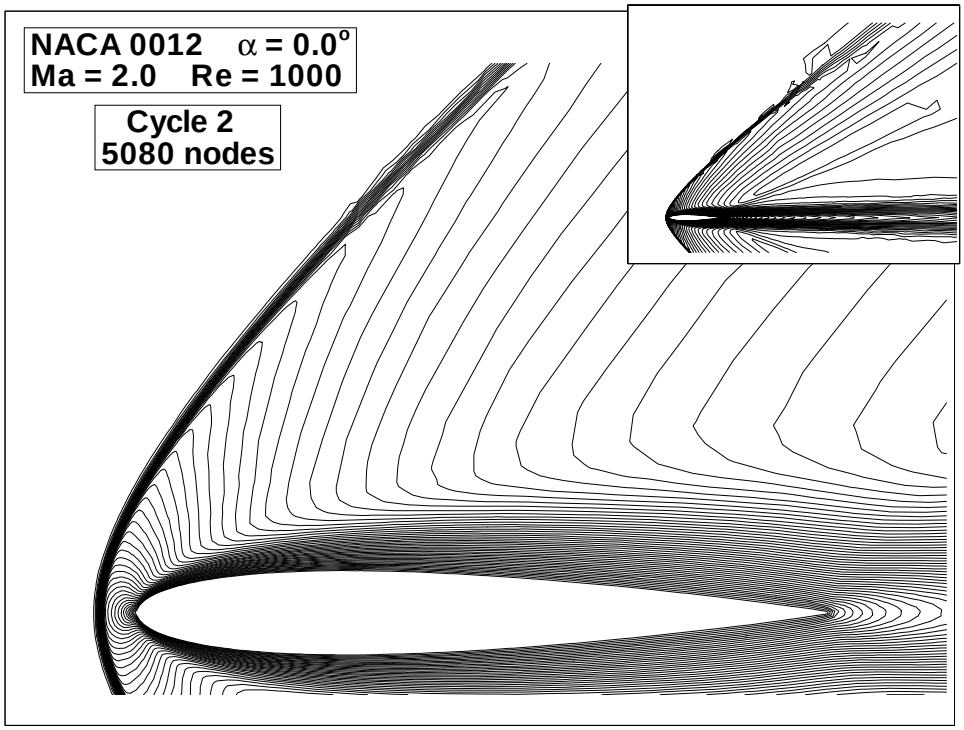



*(b)*

*Figure 5.2 Supersonic Flow around NACA 0012. (a) 1st Cycle Grid, (b) Solution*




*(a)*




*(b)*

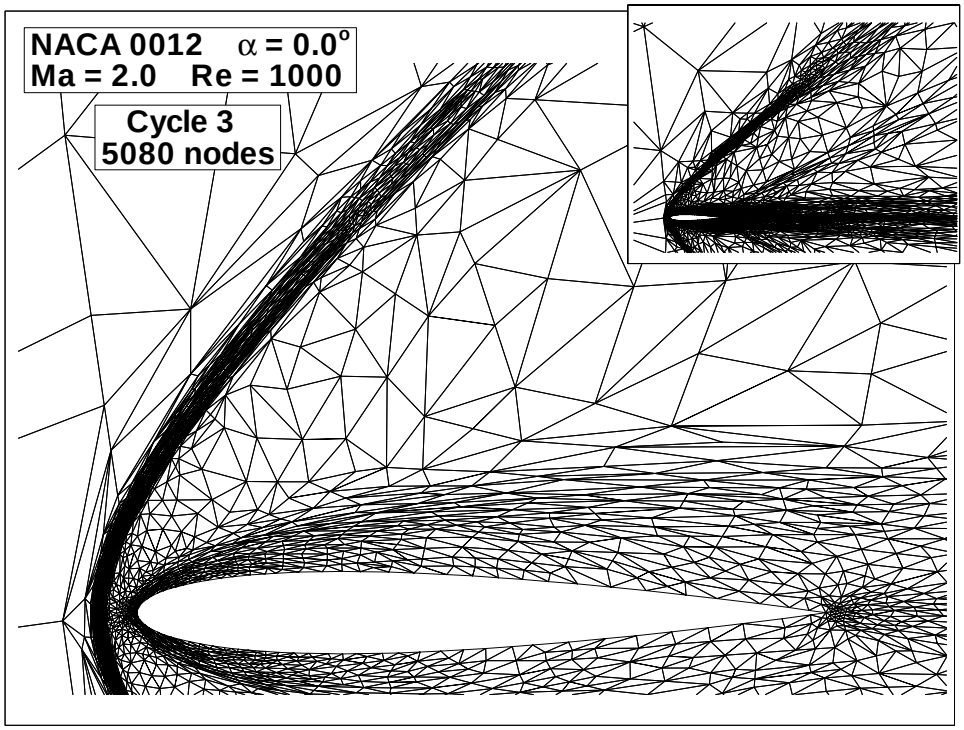
*Figure 5.3 Supersonic Flow around NACA 0012. (a) 2nd Cycle Grid, (b) Solution*

The grid in Fig. 5.3(a) clearly shows that the increased number of points gives a better outline of the shock and the viscous layer near the airfoil than the grid in Fig. 5.2(a). The Mach number contours in Fig. 5.3(b) also show that the refinement in the bow shock extends further from the leading edge. The wake is now starting to be resolved but still does not extend to the exit.

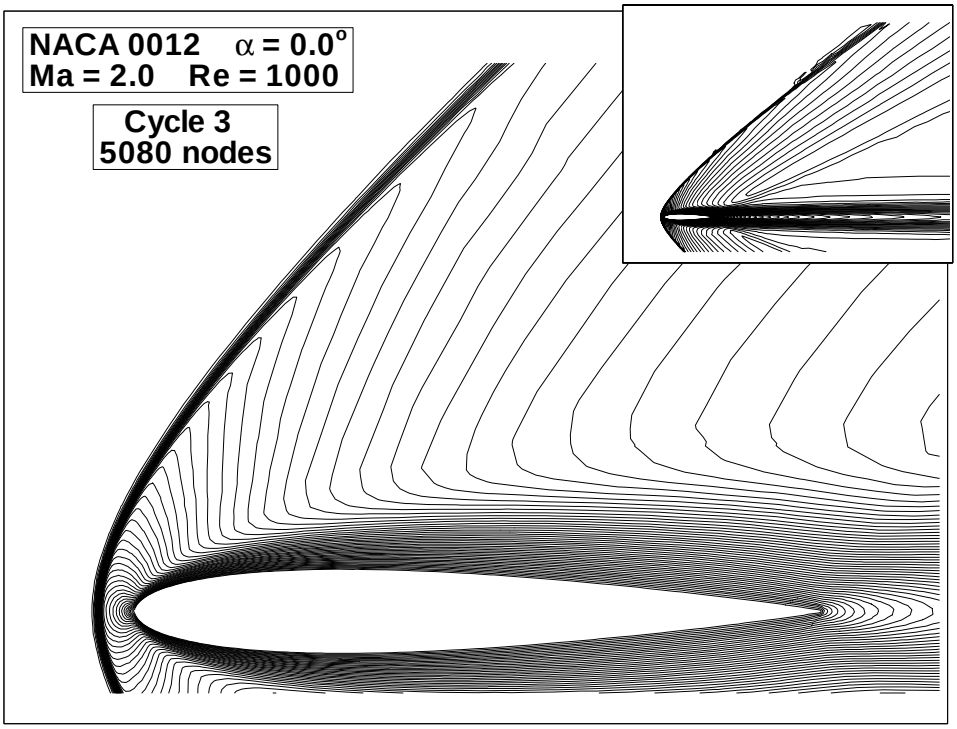
The mesh and computational results after three iterations are shown in Fig. 5.4. The bow shock appears much sharper than in Fig. 5.3(b) and a diffuse rear shock is now visible. (Notice that the rear shock is actually more visible in the mesh than in the Mach number contours.) The viscous layer is also more highly resolved. The grid and the Mach number contours extend to the end of the computational domain.

The results for two more adaptation cycles are given in Figs. 5.5 and 5.6. The number of nodes is frozen at 5080. Consequently, the modifications to the grid are accomplished by removing and replacing equal numbers of nodes. The incident shock shown in Fig. 5.5(b) after Cycle 4 is quite well resolved in both the near field and far field. The trailing edge shock appears to be relatively distinct in the far field, but remains a compression in the near field because of the low Reynolds number. Similarly, the boundary-layer region and the wake have become much more distinct.

The adaptation through the sixth cycle (see Fig. 5.7) shows only minor changes in both the shock and the viscous regions, suggesting that a stationary result has been attained. The mesh and the solution contours in the shock region are quite thick because of the low Reynolds number. More discussions on the shock adaptation will be detailed in a later chapter. Overall, the entire flow field is reasonably well resolved on this grid, and the solution quality appears to be high. In each cycle the duration of the adaptation is

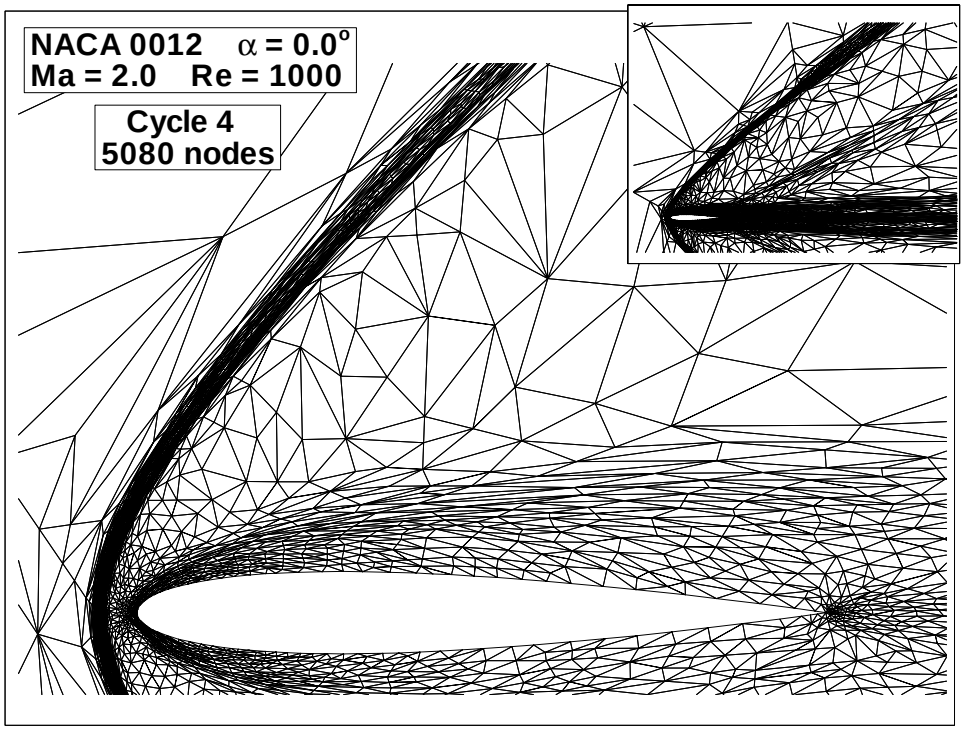



*(a)*

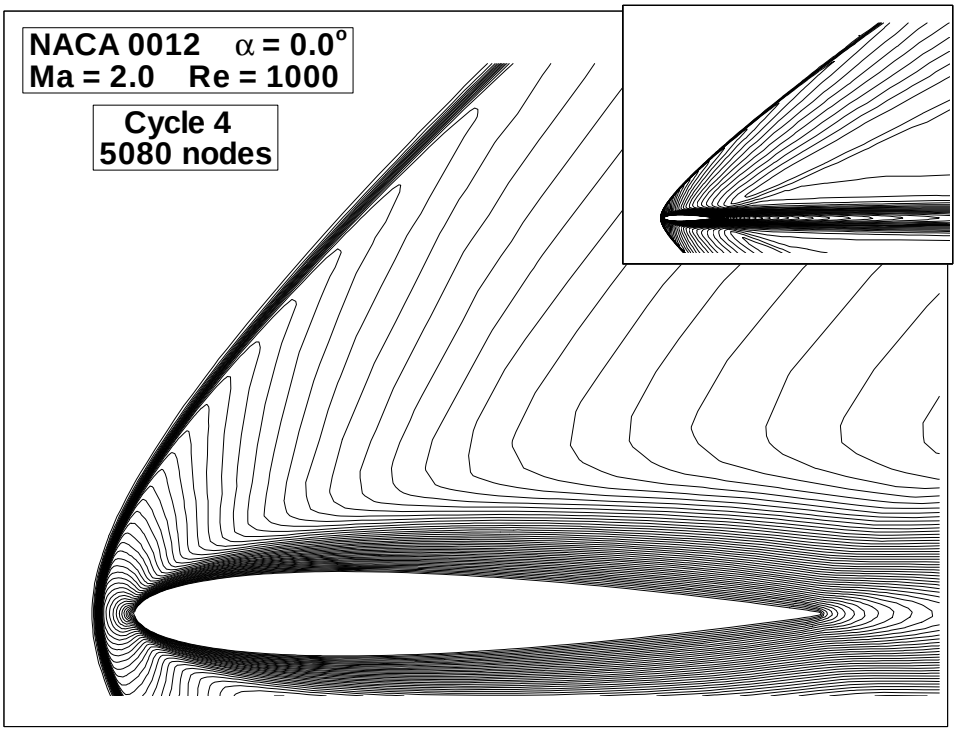



*(b)*

*Figure 5.4 Supersonic Flow around NACA 0012. (a) 3rd Cycle Grid, (b) Solution*

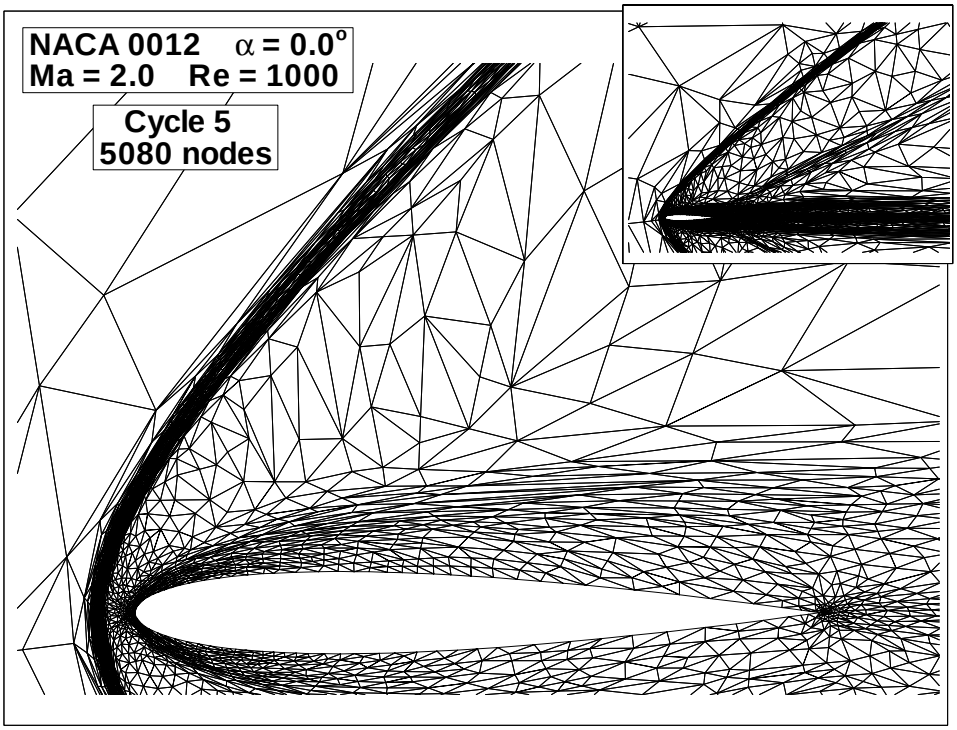



*(a)*

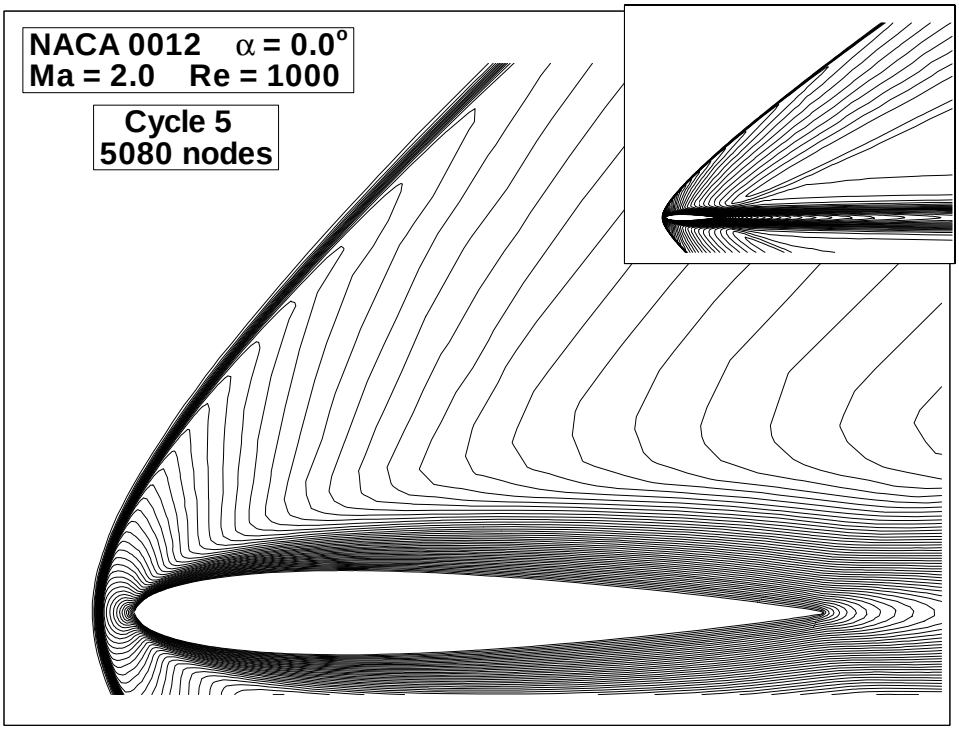



*(b)*

*Figure 5.5 Supersonic Flow around NACA 0012. (a) 4th Cycle Grid, (b) Solution*

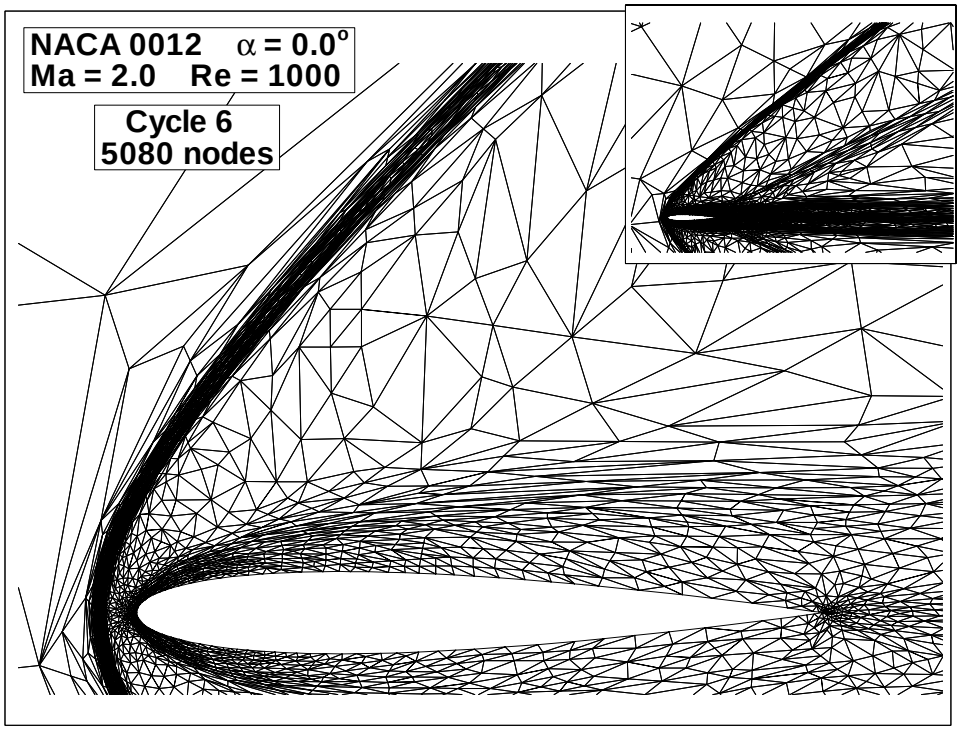



*(a)*

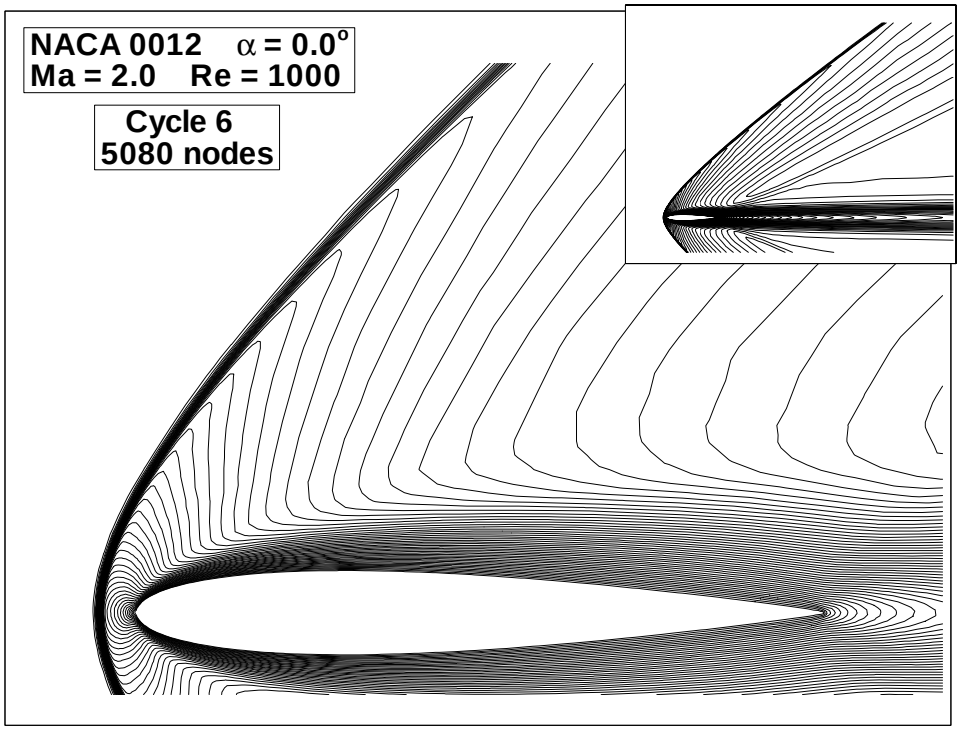



*(b)*

*Figure 5.6 Supersonic Flow around NACA 0012. (a) 5th Cycle Grid, (b) Solution*




*(a)*




*(b)*

*Figure 5.7 Supersonic Flow around NACA 0012. (a) 6th Cycle Grid, (b) Solution*

usually about twenty percent of the time that the solver takes to reach a fully converged solution.

## 5.2 Equidistribution of Metric Lengths in Adaptation

The goal of the adaptation in the current approach is to equidistribute the Riemann lengths of edges, which are defined by the solution Hessians. Charts comparing the distribution of the metric edge lengths in the initial grid and the final grid of the supersonic flow around NACA 0012 are shown in Figs. 5.8 and 5.9. The upper plot shows the distribution in the initial grid (which was isotropic, but not of equal size) in Riemann space, while the lower plot shows the final distribution after adaptation. Note that the edge lengths are plotted on a logarithmic scale. The metric edge lengths in the initial grid were distributed over six orders of magnitude, which indicates that a lot of grid points are not in the right part of the flow field. After adaptation, this spread has been decreased by nearly a factor of $10^3$. The lengths of the initial grid range from $10^{-7}$ to 1 while in the final mesh it is from $10^{-2.5}$ to $10^{-0.5}$. Edges with lengths relatively far from the center, which are about one order of magnitude, exist because of the constraints of boundaries. These constraints prevent the removal of critical points on the boundaries, since the disappearance of these points will lead to the loss of geometric integrity. Overall the Hessian edge-length re-distribution by node insertion/removal and point smoothing works well for this case.

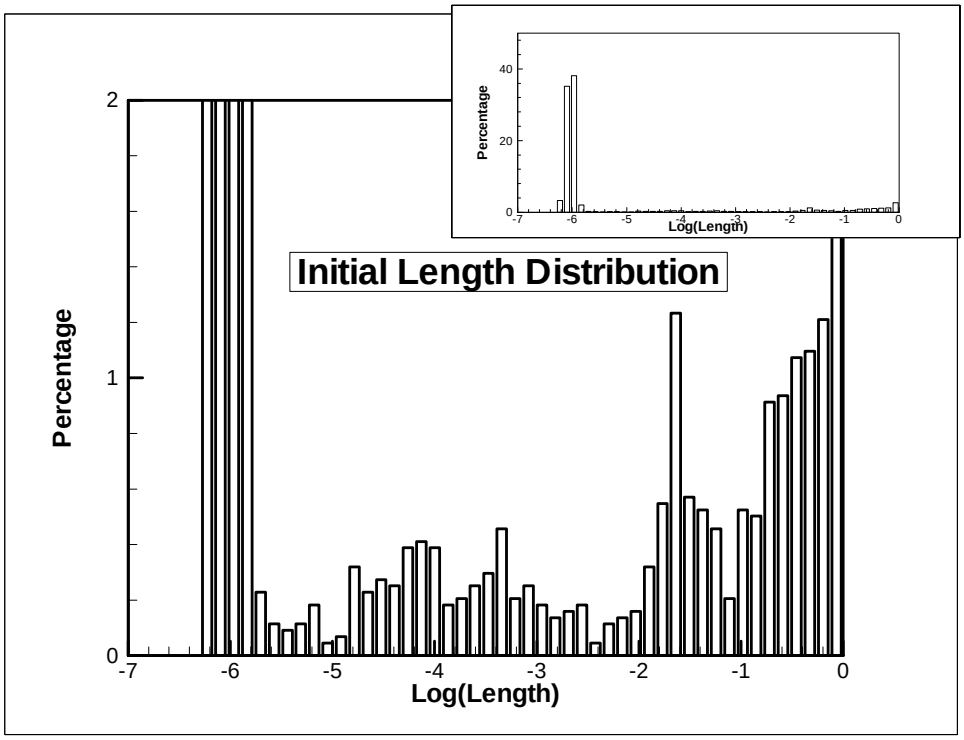



*Figure 5.8 Distribution of Riemann Edge Lengths of the Initial Mesh*

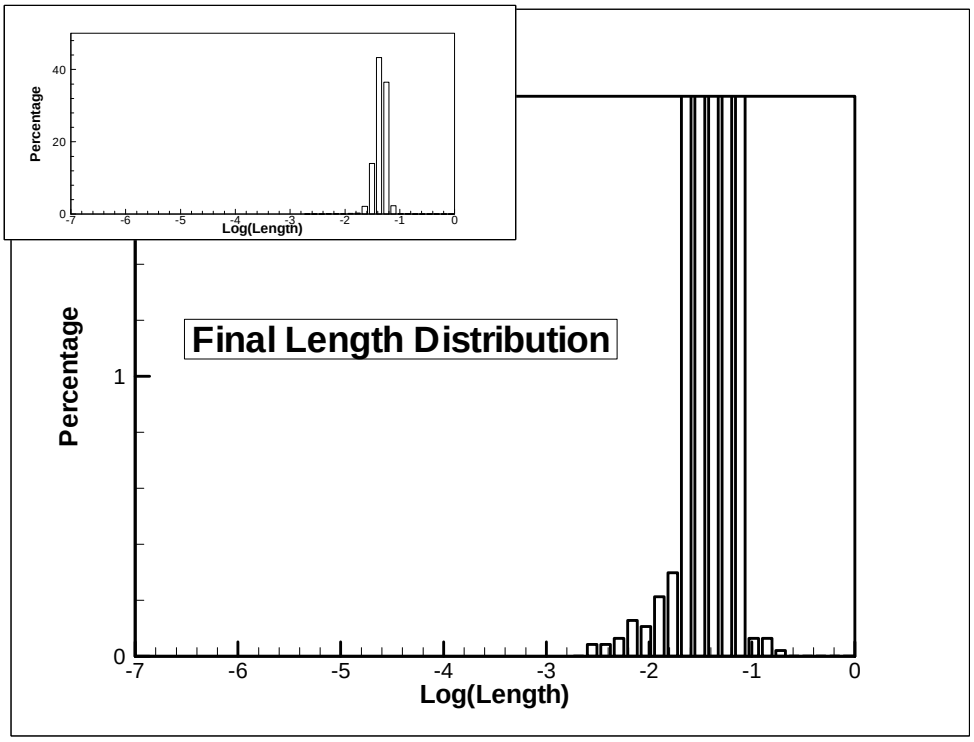



*Figure 5.9 Distribution of Riemann Edge Lengths of the Final Mesh*

## 5.3 Effectiveness of Generating Anisotropy in Adaptation

Anisotropic adaptation adjusts not only the local grid sizes, but also cell orientations. Anisotropy is introduced with the Hessian matrices of the solution variables. Stretched cells are oriented according to the flow features, such as the shocks and boundary layer in Fig. 5.7a.

A possible definition of the aspect ratio in a triangular cell [95] is illustrated in Fig. 5.10. The aspect ratio of triangles is defined as the ratio of the longest edge over the height associated with this edge.

$$Aspect\ Ratio = \frac{b}{h} \tag{5.1}$$

This straightforward definition can be easily extended to three dimensions where the choice of the face is based on the areas. Shown in Fig. 5.11 is the history of gird aspect ratios in the Euclidean (physical) space of the adaptation of supersonic flow around NACA 0012. The distribution of the aspect ratios of the initial mesh is a short segment in the upper-left corner, indicating that the mesh is isotropic. The number of cells with high aspect ratio keeps increasing in the mesh, which is consistent with the requirement of the shocks and boundary layer in this problem.

The example and analyses in this chapter show that the current adaptation procedure can equidistribute the metric edge lengths and generate a stretched grid aligned with key flow features. In the next two chapters, several key difficulties in the current adaptation will be discussed. One of them, which can be seen in Fig. 5.11, involves whether the increase in the number of high-aspect-ratio cells will ever stop.

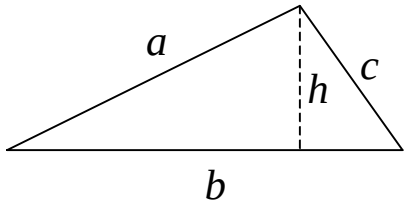



*Figure 5.10 Definition of the Grid Aspect Ratio in Two Dimensions*

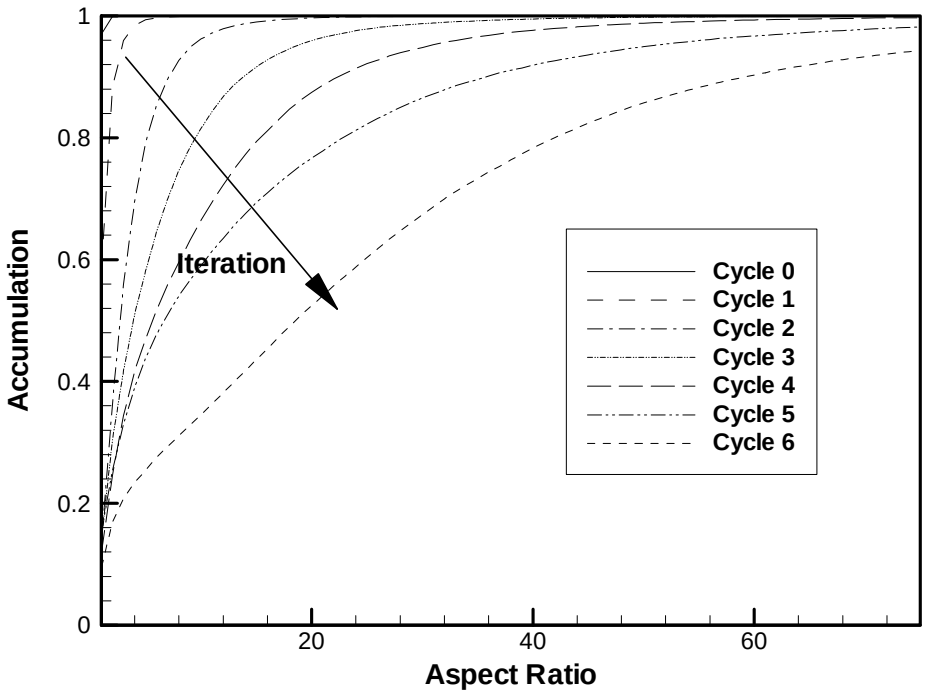



*Figure 5.11 History of the Grid Aspect Ratio*

# Chapter 6

# Boundary Layer Enhancement for Solution-Adaptive Grids

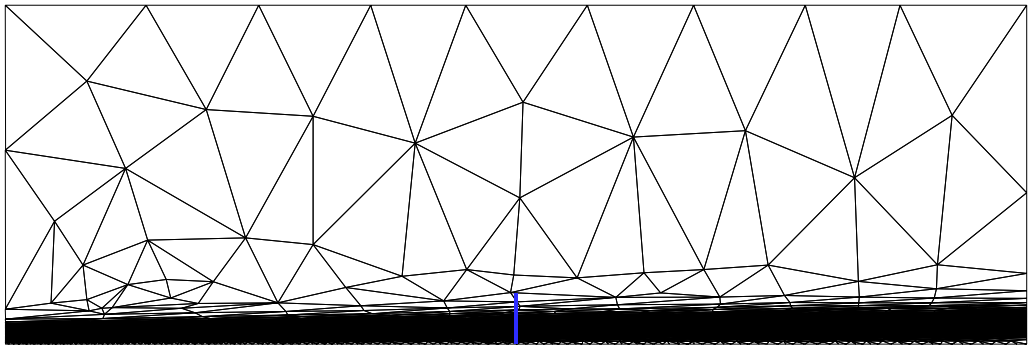
One of the regions of difficulty in grid adaptation based on a Hessian-like matrix occurs when the Hessian passes through an inflection point. When the Hessian is evaluated with tangential velocity or some similar variables, inflection points occur in the region near the wall in a viscous flow field or the mixing region of a shear layer. For the boundary layer case, an inflection point occurs on the wall when the free stream pressure gradient is zero. For favorable or adverse pressure gradients, this inflection is near the wall (either just outside, or just "inside"). The presence of this inflection point implies that the grid spacing near the wall will be wider than that far away from the wall, a condition opposite to the requirements for solution accuracy. The phenomena, both in the adaptation from a Blasius boundary layer solution and in the viscous calculation around an airfoil, are shown and remedies are proposed in this chapter.

## 6.1 Boundary Layer Adaptation with Blasius Solution

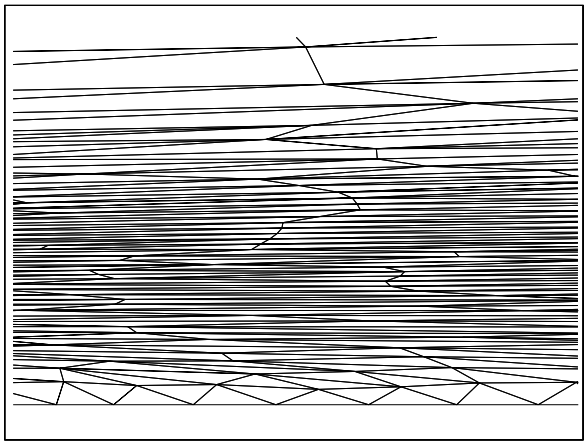
The Blasius solution [93] of the incompressible Navier-Stokes equations provides an opportunity to show the existence of the inflection points inside the boundary layer. The shear velocity, *u*, is the most representative flow variable and its second derivatives are used in the adaptation procedure.

Fig. 6.1 shows an adapted mesh based on a specified Blasius solution. The second derivatives of the shear velocity are used directly and adaptation stops when the criterion of equal edge length is met. The leading edge of the plate is set at $x = 0$ and the computing region, which is shown in the figure, ranges between values of $x = 1$ to $x = 4$. This corresponds to local Reynolds number $\mathrm{Re}_x$ increasing from 10000 to 40000. A zoomed view of the boundary layer grid at the center of the flat plate ($x = 2.5$) is shown in the lower part of Fig. 6.1.

It can be seen clearly in the zoomed mesh from the upside-down adaptation that a coarser mesh spacing appears close the wall, while adaptation gives good refinement farther from the wall. In boundary layer adaptation the second derivative of the shear velocity is the primary component in the Hessian and controls the grid spacing normal to the wall. The magnitude of the second derivative of *u* in the direction normal to the wall is shown in Fig. 6.2. It is extracted from the location shown in Fig. 6.1. The reason why the coarse grid is generated near the wall can now be seen clearly. The maximum of the second derivative appears in the middle of the boundary layer, while an inflection point occurs at the wall.



*(a)*



*(b)*

*Figure 6.1 Adapted Boundary Layer Mesh -- Hessian Defined from Shear Velocity.*
*(a) Far View, (b) Local View*

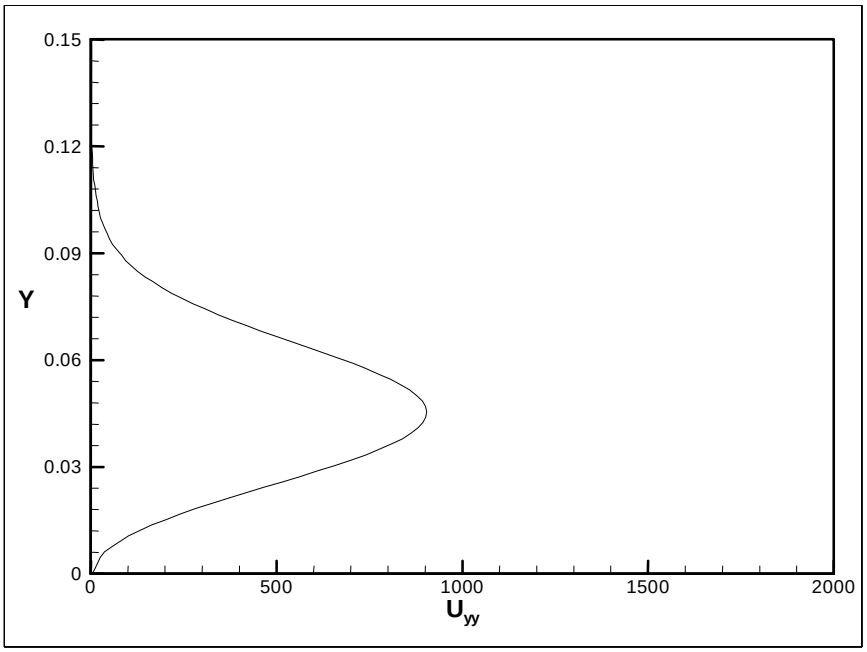



*Figure 6.2 Boundary Layer Hessian Profile -- Blasius Solution*

Since the near-wall region is a region of much concern in many flow simulations, usually the smallest grid spacing is required next to the wall for accurate prediction [39,53,55]. The Hessian must be reconstructed in order that a small spacing appears at the wall. The first remedy proposed here is to find an appropriate variable whose second derivatives give a more desirable profile near the wall. From Fig. 6.2 it can be seen that although the inflection point is at the wall, the maximal gradient of the shear velocity appears at the wall. The gradient (in vector form) itself is not an appropriate feature indicator for anisotropic adaptation because the gradient does not provide as much information to define the cell sizes and alignment as the Hessian does. However, in two dimensions it is possible to find other variables based upon which the calculated Hessian is composed of the gradients rather than the second derivatives of the velocities. Specifically a stream function meets the above requirement and is a potential choice in two dimensions. The definition of the stream function, $\psi$, is

$$\rho u = -\frac{\partial \psi}{\partial y}, \qquad \rho v = \frac{\partial \psi}{\partial x} \tag{6.1}$$

The Hessian calculated from the stream function is

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 \psi}{\partial x^2} & \frac{\partial^2 \psi}{\partial x \partial y} \\ \frac{\partial^2 \psi}{\partial y \partial x} & \frac{\partial^2 \psi}{\partial y^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial \rho v}{\partial x} & \frac{\partial \rho v}{\partial y} \\ -\frac{\partial \rho u}{\partial x} & -\frac{\partial \rho u}{\partial y} \end{pmatrix} \tag{6.2}$$

A result from adaptation with the above Hessian is shown in Fig. 6.3. It can be seen in the mesh that the minimal cell size (in the normal direction) appears close to the wall. This is understandable since the predominant second derivative of the stream function in the normal direction appears at the exact location in the Hessian (Equation 6.2). Although the Hessian defined from the stream function does a give reasonable solution for resolving the boundary layer, its capability is limited since the definition and evaluation of the stream function in three dimensions are quite involved.

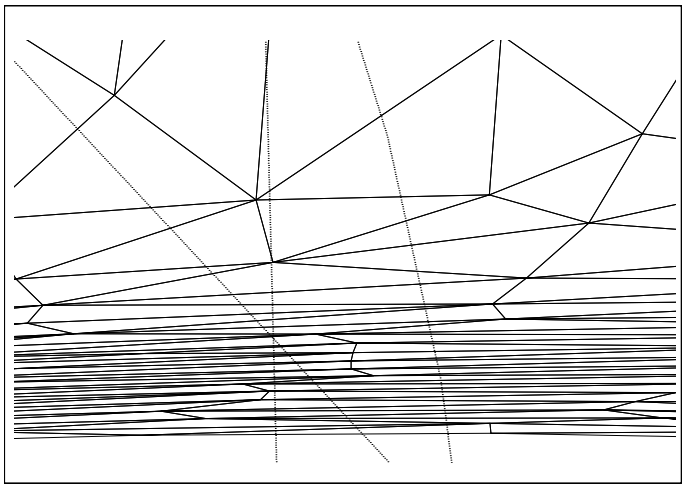


*Figure 6.3 Adapted Boundary Layer Mesh -- Hessian Defined from Stream Function*

An alternative to circumvent the near-wall difficulties is to replace the computed Hessian in such regions. The Hessian is replaced by a modified expression that is valid only in the near-wall region. At the beginning the location adjacent to the wall where the Hessian reaches its maximum (which means the eigenvalues of the Hessian are a maximum) is found. The Hessian, its eigenvalues and eigenvectors at this point are computed and denoted as $\mathbf{H}_{\max}$, $\lambda_{i,\max}$, etc. In order to define an appropriate near-wall spacing, a "wall" Hessian, $\mathbf{H}_w$, is introduced and the Hessian matrix between the point of maximum eigenvalues and the wall is obtained by linearly interpolating between $\mathbf{H}_{\max}$ and $\mathbf{H}_w$.

The implementation of the "wall" Hessian is illustrated in two dimensions for simplicity. In order to define $\mathbf{H}_w$, a rotated coordinate system, $\mathbf{x}'$ with components normal and parallel to the wall is introduced first. The wall Hessian is required to be diagonal in this rotated coordinate system, so that the exact spacing of the first cell from the wall can be defined. The transformation matrix that rotates the wall-normal coordinate system, $\mathbf{x}'$, to the reference coordinate system, $\mathbf{x}$, can be expressed as, $\mathbf{x} = \mathbf{P}\mathbf{x}'$ where the elements of $\mathbf{P}$ are the direction sines and cosines between the two coordinate systems. The requirement that the similarity transform of the wall Hessian be diagonal is made in the wall-normal system,

$$\mathbf{P}^{-1}\mathbf{H}_w\mathbf{P} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{6.3}$$

At the wall $\lambda_2$, whose corresponding eigenvector is normal to the wall, is picked to give a desired grid spacing (Euclidean) at the wall.

The value of $\lambda_2$ is determined as following. When one is going to specify a wall spacing, or the height of a cell adjacent to a wall, $l_{y'}^E$, on the wall, the Riemann length of an edge normal to the wall will be

$$l_y^R = \left(\lambda_1 (\Delta x')^2 + \lambda_2 (\Delta y')^2\right)^{1/2} = \sqrt{\lambda_2}\,\Delta y' = \sqrt{\lambda_2}\, l_{y'}^E \tag{6.4}$$

where $\Delta x'$ and $\Delta y'$ are the differences of the local coordinates of the edge. The tangential difference, $\Delta x'$, is zero for the edge normal to the wall.

The current average Riemann length, $\bar{l}^R$, is defined as the average of all edges in the domain. When the Riemann length of the edge normal to the wall, given in Equation 6.4, is equal to this average length, the eigenvalue of the local "wall" Hessian, $\lambda_2$, whose corresponding eigenvector is normal to the wall, is then given by

$$\lambda_2 = \left(\bar{l}^R / l_{y'}^E\right)^2 \tag{6.5}$$

The other eigenvalue of $\mathbf{H}_w$ is specified by setting it equal to the eigenvalue associated with the eigenvector that is directionally closest to the tangent of the wall at the point where the Hessian is a maximum. This completes the definition of the wall Hessian.

The linear "matrix" interpolation between the maximum Hessian point and the wall is accomplished by linearly interpolating the two eigenvalues and the rotation angles between the Hessian at the maximum location and the one at the wall. This gives the Hessian at intermediate locations between the wall and the maximum Hessian point.

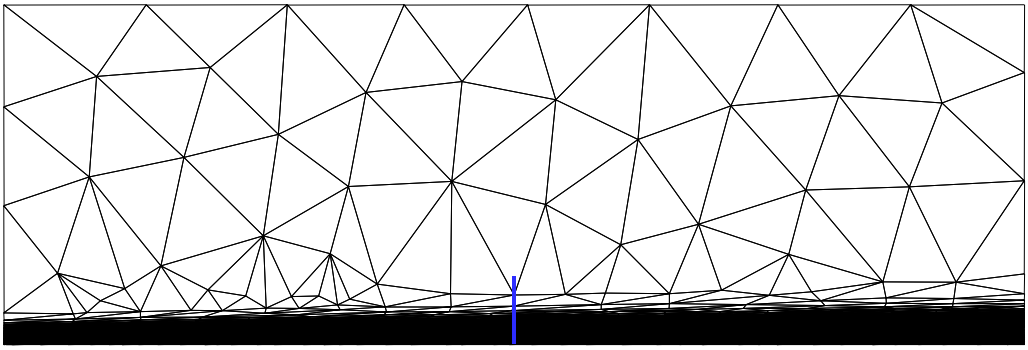
The improvement obtained with this near-wall correction for the Blasius boundary layer can be discerned by comparing Fig. 6.4 with Fig. 6.1. In Fig. 6.4 a small $l_{y'}^{E}$, which is about one tenth of the wall spacing in Fig. 6.1, is specified to define the wall Hessian. The second derivative along the short segment normal to the wall in Fig. 6.5 now increases monotonically from zero in the free stream to a maximum at the wall. The resulting grid spacing also decreases monotonically as one moves toward the wall. The finest grid is adjacent to the wall with the coarsest grid in the free stream.

It can be seen that this remedy allows the user to specify an arbitrary grid size adjacent to the wall as is normally done in structured grid generation. This is particularly useful for turbulence calculations since in most turbulence models an appropriate wall distance is required usually for accurate prediction. The specification of the grid spacing at the wall is often done by setting a constant $y^+$ (the dimensionless sublayer-scaled distance) along the wall. For example, to set the first grid point at a location of $y^+ = K$ in a turbulent flow, the physical (Euclidean) wall length $l_{y'}^{E}$ is picked as,

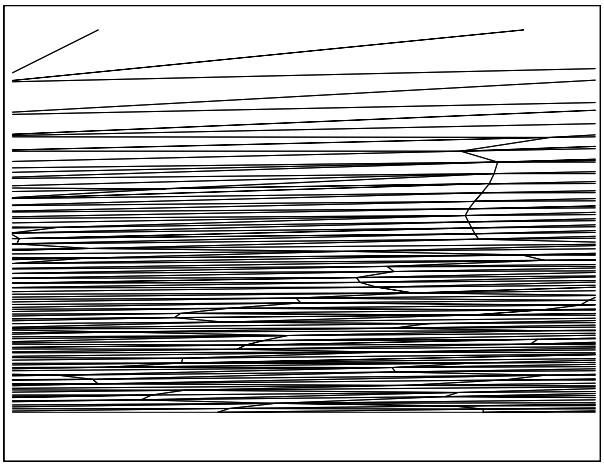
$$l_{y'}^{E} = K\sqrt{\nu / (\partial u' / \partial y')} \tag{6.6}$$

where $\nu$ is the kinetic viscosity and $u'$ is the velocity parallel to the wall.

The approach which improves the Hessian defined from flow velocity or similar parameter (such as Mach number) can be extended to three dimensional boundary layer adaptation more readily than the one in which the second derivatives of the stream function are used. The boundary layer enhancement with this approach is used routinely in the rest of the adaptation cases.



*(a)*



*(b)*

*Figure 6.4 Adapted Boundary Layer Mesh – with Boundary Layer Enhancement.*
*(a) Far View, (b) Local View*

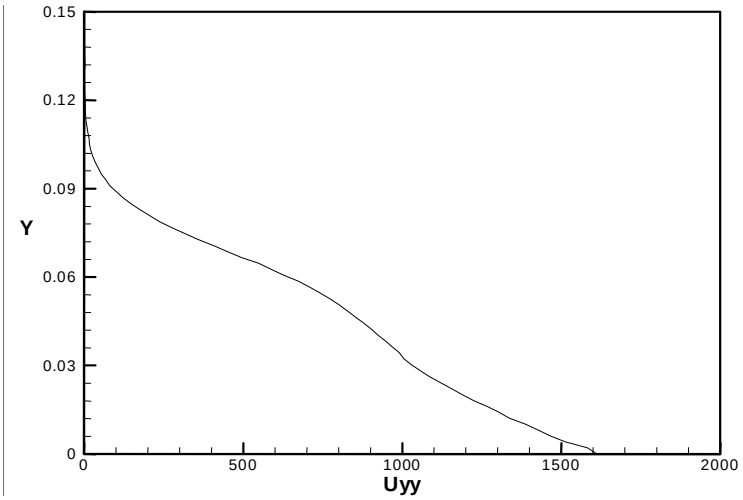



*Figure 6.5 Boundary Layer Hessian Profile – with Boundary Layer Enhancement*

## 6.2 Boundary Layer Adaptation for Subsonic Flow around NACA 0012 Airfoil

A more realistic boundary layer adaptation study is conducted in order to further illustrate that the fine grid near the wall is necessary and that the enhancement improves the accuracy of flow simulation. The justification of the boundary layer enhancement in last section is made by comparing the results from the corrected grid and uncorrected grid with a "benchmark solution".

### 6.2.1 Adaptation with Un-Corrected Hessian

The case calculated here is a subsonic flow around the NACA 0012 airfoil. The Mach number of the free stream flow is 0.5 and Reynolds number is $2.89\times10^{6}$ which results in a reasonably thin boundary layer. Zero attack angle is chosen for this case. The

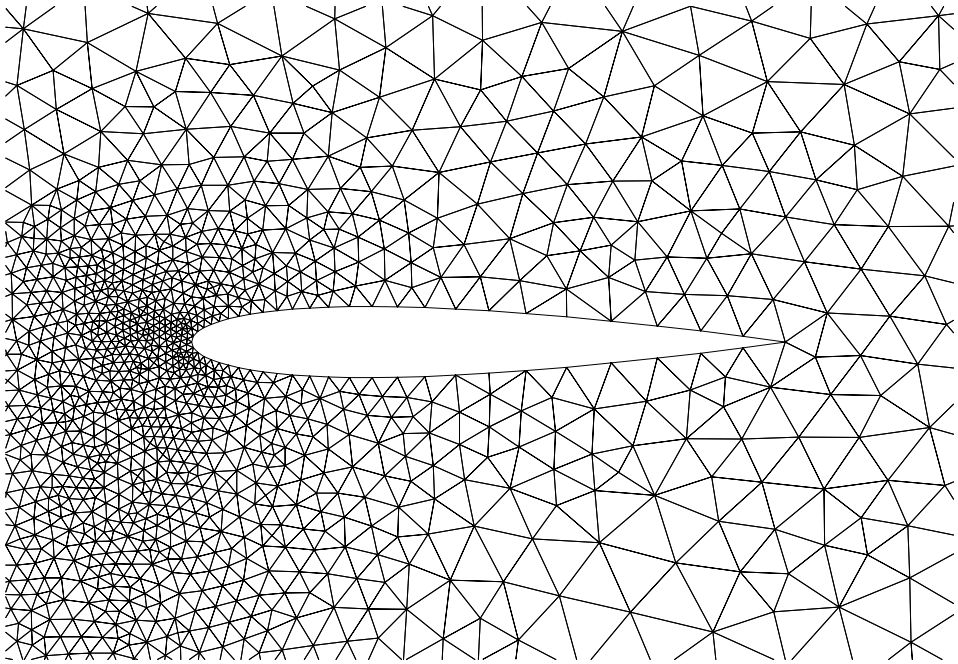


*Figure 6.6 Original Isotropic Grid for Boundary Adaptation (2000 nodes)*

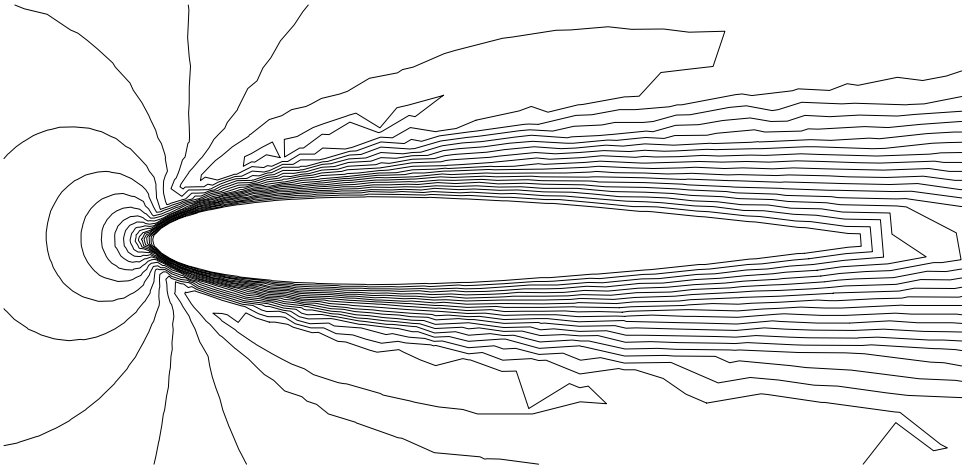
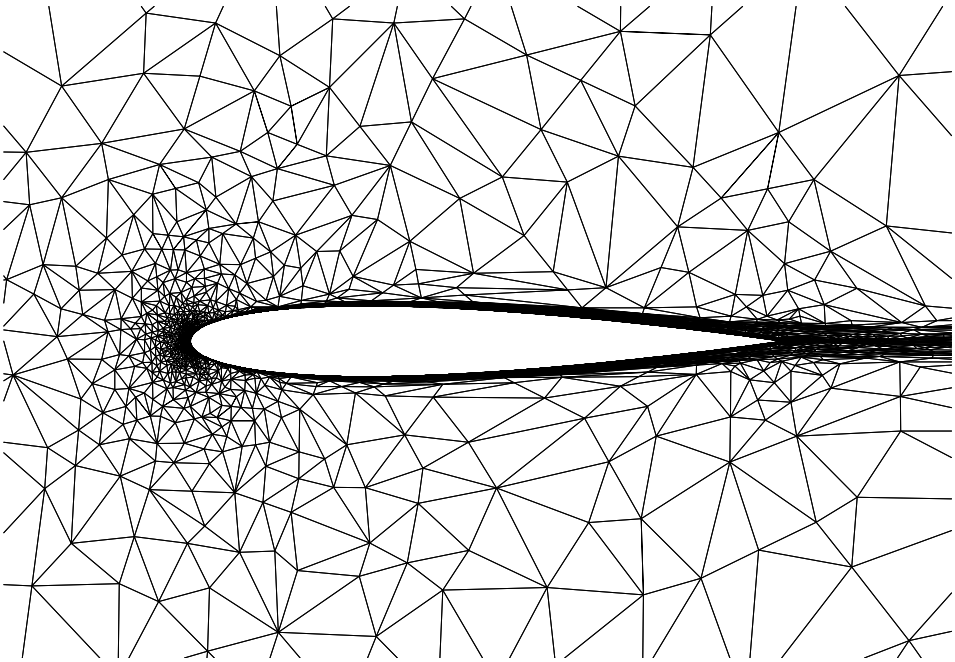


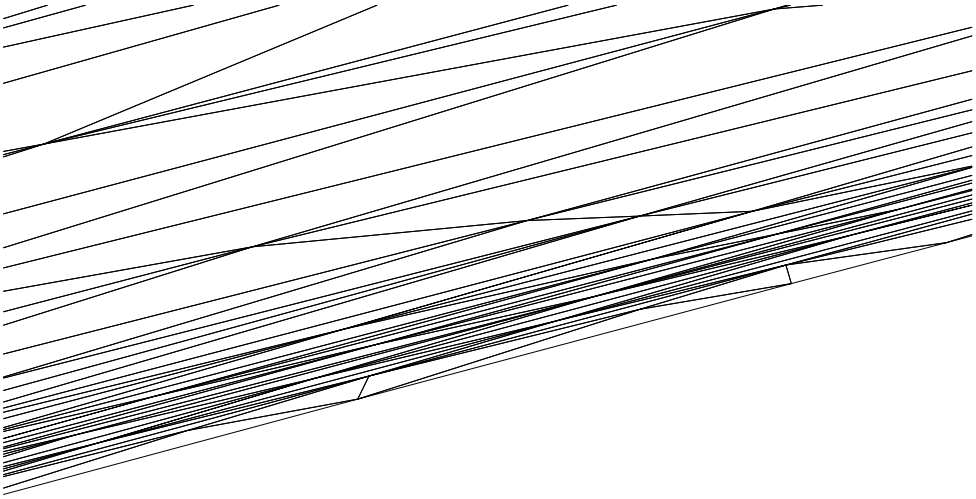
*Figure 6.7 Contours of Mach Number from Original Grid*

adaptation process is initiated from the isotropic grid shown in Fig. 6.6. The grid is generated by CFDRC-GEOM [92]. Fig. 6.7 shows the contours of the Mach number from the calculation on this grid as computed by the code FUN2D [39]. The one equation turbulence model used is that of Baldwin and Barth [54]. The boundary layer is very thick due to the large diffusion caused by the coarse grid near the wall. With only 2000 points, the grid is not sufficient even for inviscid flow simulation. Clearly this mesh is not adequate for boundary layer predictions. In following cycles, this grid is refined by adding more points than those being removed.

Shown in Fig. 6.8 are the grids after ten refinement iterations during which the proposed enhancement approach in last section have not been used. Since the shear velocity is cumbersome to calculate here, the Mach number, which provides adequate information of the flow field, is used to evaluate the Hessian. The calculated Hessian is not broken down near the wall and thus the "wall" Hessian is not introduced to give a specified spacing of the first point off the wall. The number of points in the refined grid in Fig. 6.8 is increased to 10000. The boundary layer has been refined, as shown in the upper part of the figure, such that it is now much thinner than that in Fig. 6.6. The result calculated with this grid is shown in Fig. 6.9 and gives a qualitatively much better solution. A coarse grid is generated close to the wall in a manner analogous to that seen in the Blasius boundary layer case as shown in the lower part of Fig. 6.8, which indicates the Mach number gives a similar inflection characteristic as the shear velocity does. The second derivatives on the wall are much smaller than the maximums in the middle of the boundary layer. The finest grids appear in the middle of the boundary layer. To evaluate the accuracy of the present calculation, quantitative information such as local pressure



*(a)*



*(b)*

*Figure 6.8 Adapted Mesh without Boundary Layer Enhancement (10000 nodes). (a) Far View, (b) Local View*

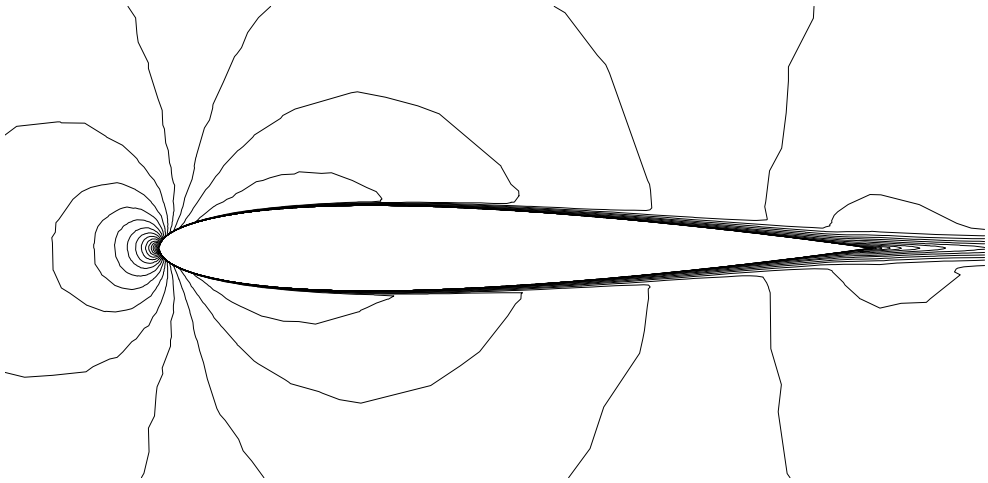
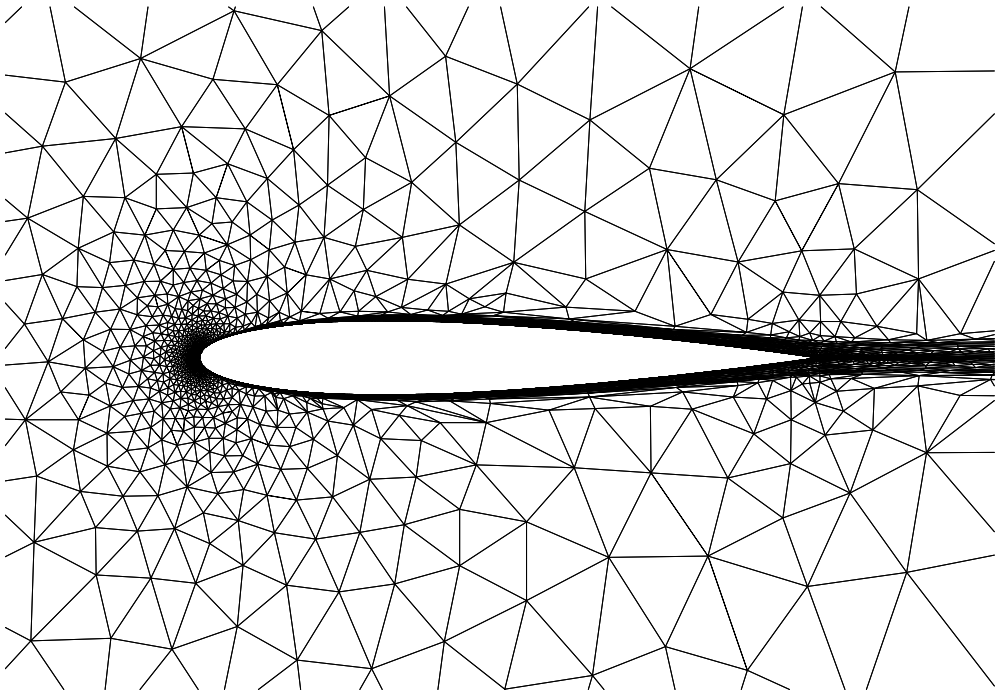


*Figure 6.9 Contours of Mach Number, Adapted Mesh without Boundary Layer Enhancement*

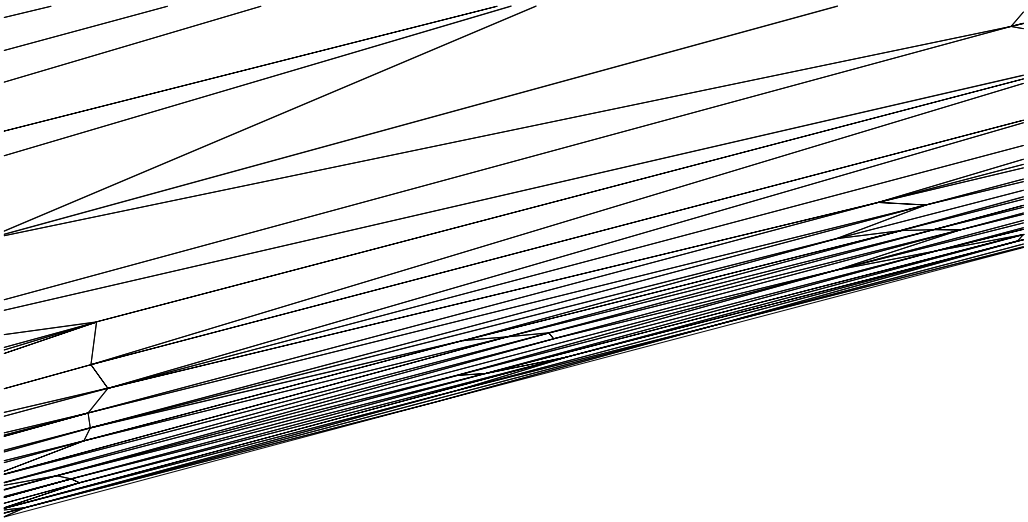
coefficients and skin friction coefficients must be extracted from the solution. This will be done later together with the calculation on the corrected grid.

### 6.2.2 Adaptation with Boundary-Layer-Enhanced Hessian

The adapted grid with boundary layer enhancement is shown in Fig. 6.10, also after 10 iterations. The number of points in this grid is again 10000, the same as that in Fig. 6.8 by controlling the number of nodes added and removed. The Hessian calculated from the Mach number is replaced in the region between where the Hessian reaches its maximum and the wall. The replacement Hessian starts from each of the cells adjacent to the wall and travels along the normal to the wall until the maximum value of the Hessian is located. The "wall" Hessian is introduced at the wall and the interpolation between the wall and the maximum Hessian is used to replace Mach-number-based formulation.



*(a)*



*(b)*

*Figure 6.10 Adapted Mesh, with Boundary Layer Enhancement(10000 nodes).*
*(a) Far View, (b) Local View*

The boundary layer enhancement here specifies that $y^+ = 1$ holds everywhere around the airfoil, which introduces a much larger Hessian on the wall. Note that $y^+ = 1$ of the first point adjacent to the wall has been suggested by most researchers [55,53,39] to give an accurate prediction of the skin friction coefficients. Globally the mesh shown in the first part of Fig. 6.10 looks very much like that in Fig. 6.8, while the boundary layer in the zoomed part shows that now finest grid spacings appear close to the wall compared with those in Fig. 6.9 and with the grid in the middle of the boundary layer. The Mach number contours in Fig. 6.11 are qualitatively the same as those in Fig. 6.9. In the latter part of this chapter, the justification of the boundary layer enhancement is made by extracting more information from the computation and comparing it with a "benchmark solution".

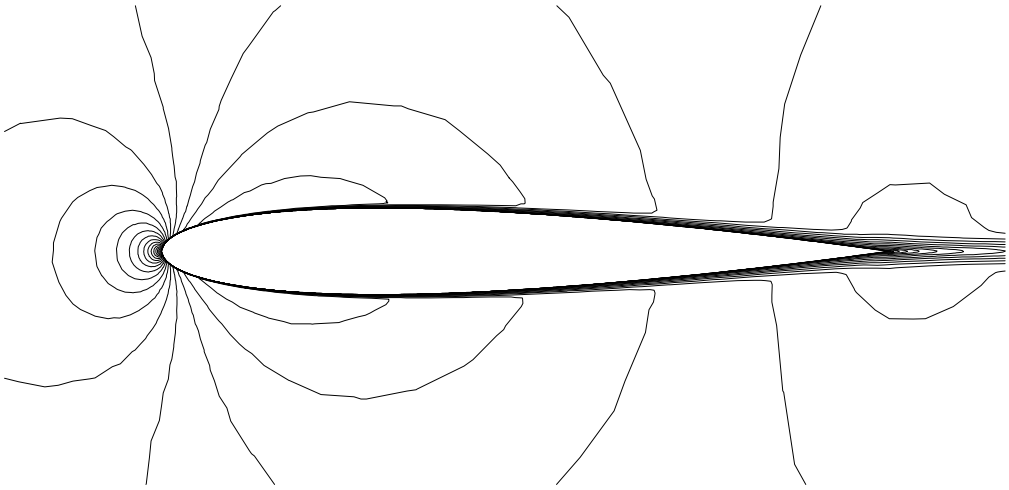
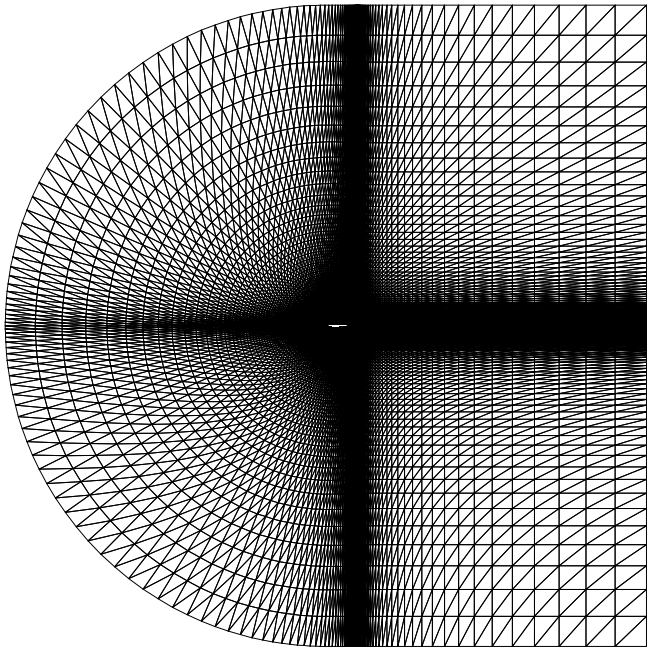


*Figure 6.11 Contours of Mach Number, Adapted Mesh with Boundary Layer Enhancement*

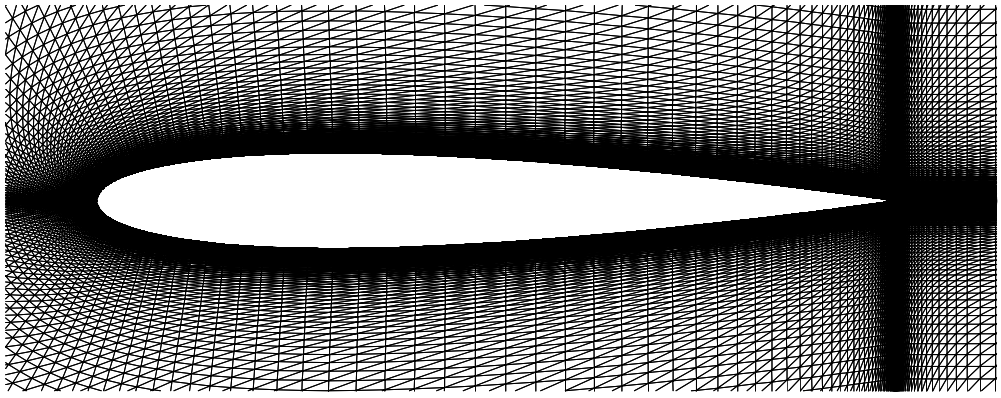
### 6.2.3 Comparison with Benchmark Solution

In order to evaluate the results calculated with the above two adapted meshes, a more accurate solution was set up to compare against. For the "benchmark" case, a well refined C_Type grid was generated around the NACA 0012 airfoil using CFDRC-GEOM [92]. Two grid sizes were chosen, $150\times200$ and $250\times400$, to verify that the resulting solution was grid independent. The former grid is shown in Fig. 6.12 and contains 29849 points and 59004 triangles. Note that every rectangle in the mesh is divided into two triangles since the FUN2D [39] solver used in the current adaptation can only accept pure triangular grids. Although many of the points, such as those above the tip and far behind the airfoil, are not placed in the appropriate regions, the details of interest in this problem are refined adequately in this grid. In particular, the boundary layer and wake regions are covered by highly anisotropic mesh in order to capture the strong directional variations there.

The Mach number contours of the converged solution are shown in Fig. 6.13 which is again globally similar to both of the results from the previous two grids. Local pressure coefficents and skin friction coefficients from the two "benchmark" calculations are shown in Fig. 6.14 and Fig. 6.15 respectively. Note that wiggles exist near the leading edge of the airfoil for both the pressure and skin friction coefficients. These wiggles in the skin friction coefficients are caused by the lack of convergence in these low Mach number regions, where the present CFD solver (FUN2D) has limited capability to damp errors. The indiscernible difference in these plots indicates that a grid-independent solution has been achieved. The results will therefore be considered as a benchmark solution with which the solutions from the two adapted meshes can be compared.



*(a)*



*(b)*

*Figure 6.12 C_Type Grid around the Airfoil(29849 nodes).*
*(a) Far View, (b) Local View*

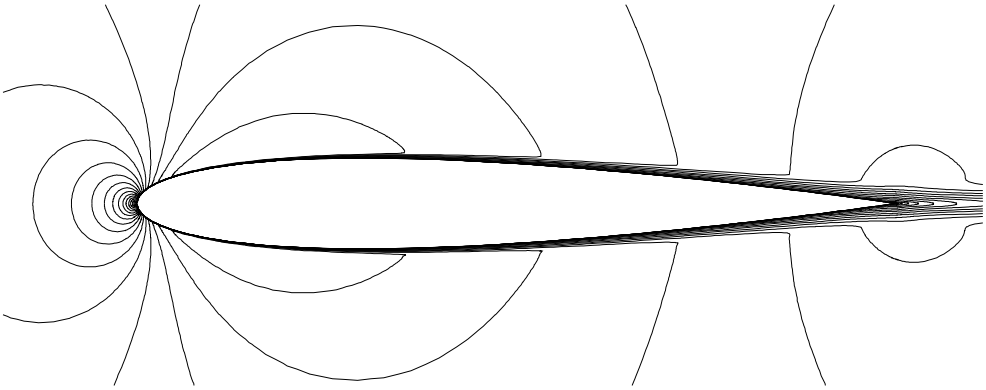


*Figure 6.13 Contours of Mach Number, C_Type Mesh (29849 nodes)*

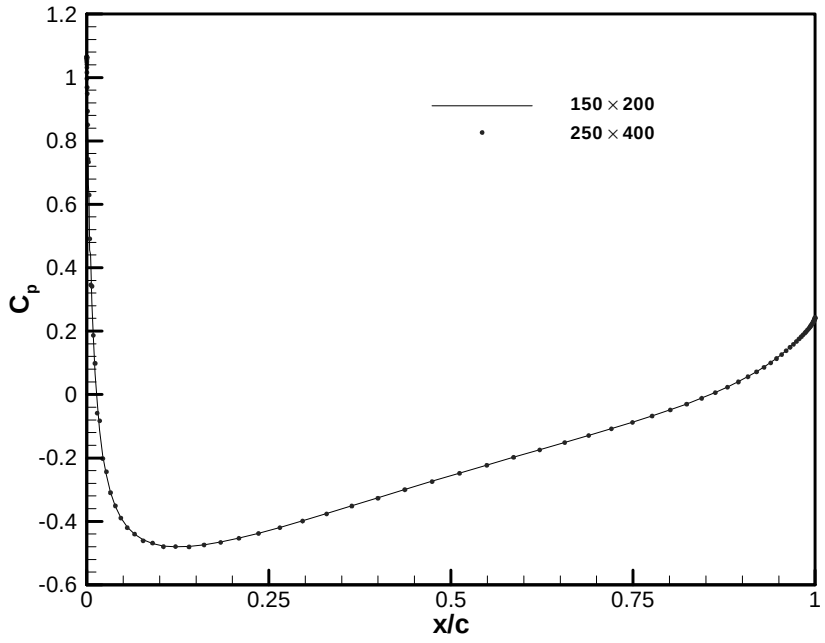



*Figure 6.14 Pressure Distribution for NACA 0012 (Benchmark Solution)*

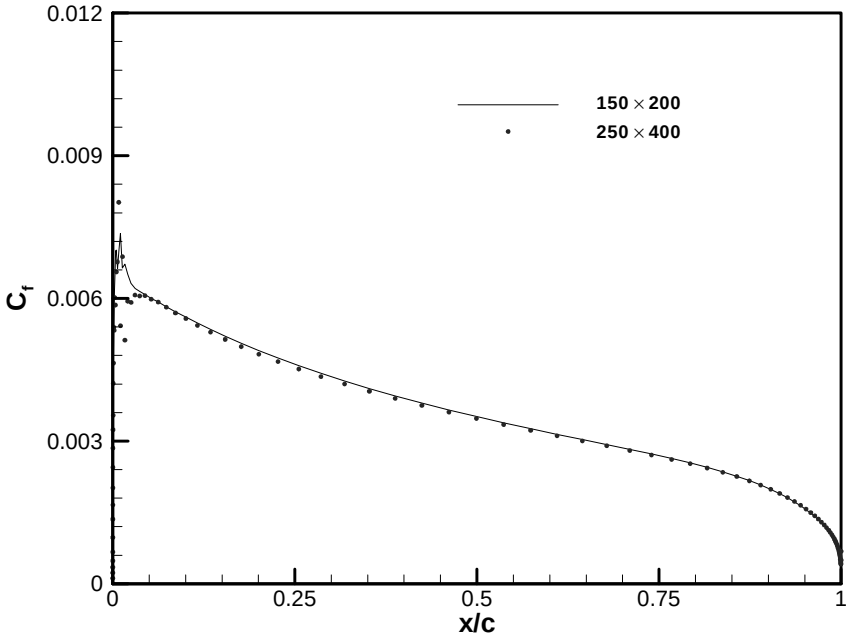



*Figure 6.15 Skin Friction Coefficients for NACA 0012 (Benchmark Solution)*

The pressure distribution from the calculation with the two adapted meshes is shown in Fig. 6.16, along with the benchmark solution. It can be seen clearly that the calculation from the enhanced mesh gives a better coincidence with the benchmark solution. The calculation with the uncorrected mesh gives a profile that differs considerably from the benchmark solution. The minimum pressure coefficient in the unenhanced mesh is smaller than that in the benchmark solution while the adapted solution with the boundary layer enhancement mesh assesses well in the minimum pressure region. This indicates that the predicted boundary layer in the uncorrected mesh is thicker than it should be, thereby accelerating the flow more and resulting in a smaller local static pressure. This increased boundary layer thickness is caused by the increased numerical diffusion caused by the large grid spacing adjacent to the wall. In Fig. 6.17, the skin friction coefficients from the two meshes are shown, again indicating that the

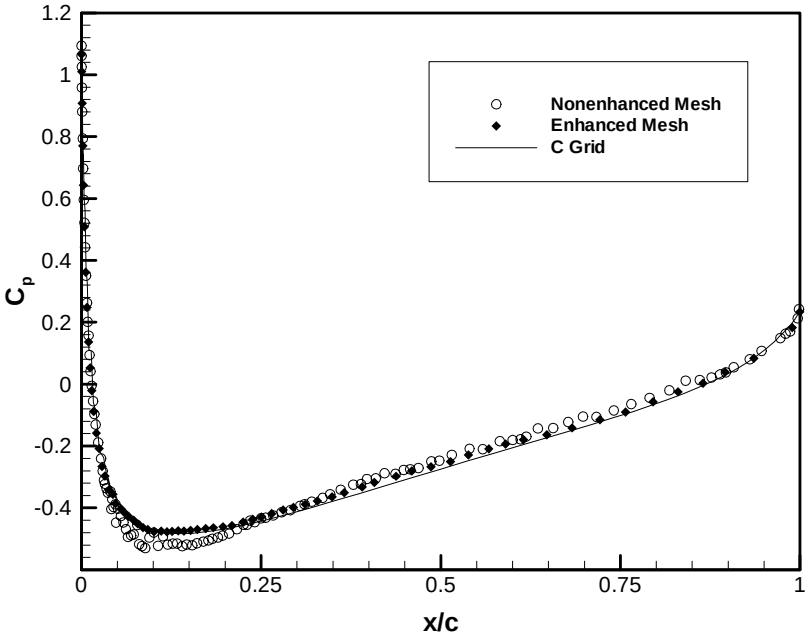



*Figure 6.16 Comparison of Pressure Distribution for NACA 0012*

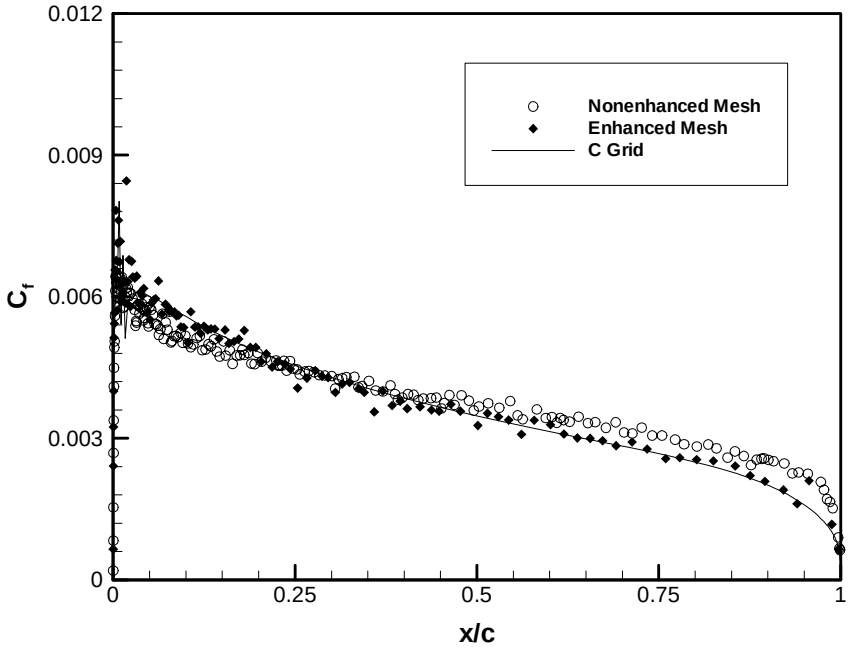



*Figure 6.17 Comparison of Skin Friction Coefficients for NACA 0012*

enhanced mesh is necessary for better resolution. Although the distribution of the tangential velocity inside the sub-layer is linear, which means the friction coefficient might not be sensitive to the height of the first cell if it is within the sub-layer, the uncorrected wall spacing loses direct control of the heights of the cells close to the wall and the inflection can cause first point off the wall to move outside the sub-layer region. The thicker boundary layer also causes the accuracy of local flow field resolution to deteriorate. These are the two factors causing the skin friction coefficients calculated on the uncorrected grid to deviate from the benchmark solution. Fig. 6.18 shows how the enhanced mesh refines the grid near the wall in terms of $y^+$, the controlling parameter for the distance of the first grid point off the wall. As can be seen, $y^+$ is almost constant for the corrected mesh. The uncorrected mesh gives both a large value of $y^+$ and a wider variation of $y^+$, and thus results in a less accurate calculation of the flow field.

The remedy for the adaptation close to a wall gives a better solution. The enhancement developed in this chapter is implemented in the adaptation procedure and will be turned on routinely for high Reynolds number viscous flows where the height of the first grid point to a wall is very important for accurate prediction. The inflection points existing in the remainder of the flow field, such as those in the wake, are not corrected in the current work. Lacking a mechanism to locate the inflection region and define an appropriate Hessian, the correction of these kinds of inflection requires additional effort although extensions of the present method appear feasible.

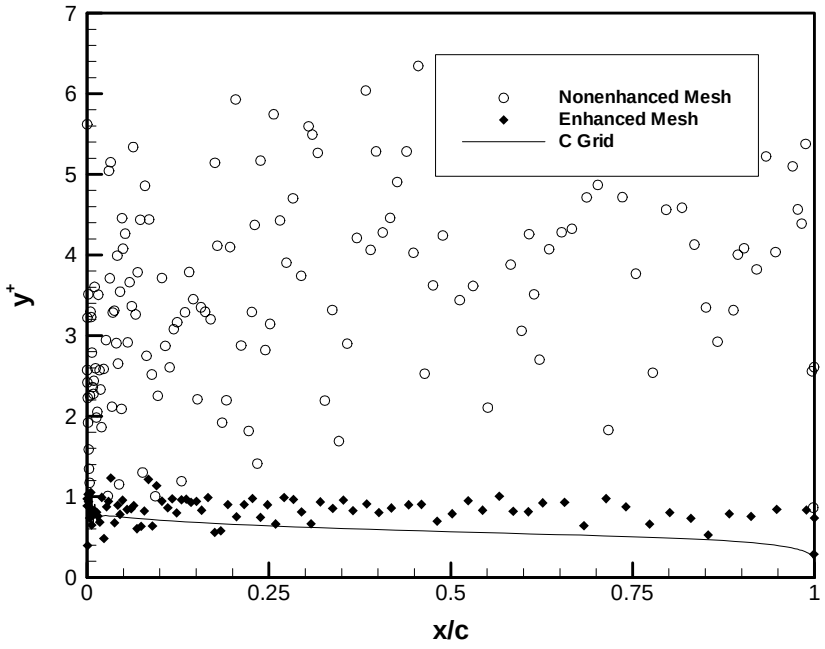



*Figure 6.18 Comparison of Upper Surface $y^+$ Distribution for NACA 0012*

# Chapter 7

# Enhancement for Supersonic Flow Adaptation

A shock represents a discontinuity in a flow field. In the vicinity of shock waves, the adaptation criterion continues to reduce the grid spacing, while the reduced spacing will further increase the magnitude of the Hessian. No matter how finely packed the grid is, the shock will never be completely resolved. In principle, all grid points in the field will eventually be drawn to the shock region if the adaptive procedure is allowed to continue, due to the ever-increasing magnitudes of the Hessian. Consequently, some criterion must be specified to limit refinement to ensure that an ample number of elements are available for other solution features.

## 7.1 Adaptation to Artificial Discontinuity

In order to investigate how a discontinuity attracts grid points, a discontinuous function, $f$, is specified as,

$$f = \begin{cases} 0 & X < -5 \\ 10 & X \geq -5 \end{cases} \tag{7.1}$$

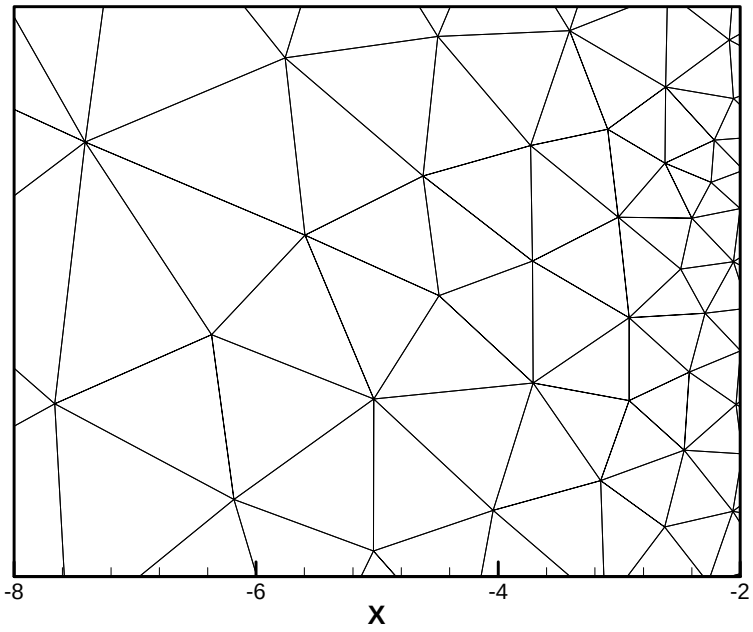



*Figure 7.1 Artificial Discontinuity – Initial Grid*

This function is used to compute the Hessian on the initial grid shown in Fig. 7.1. The adaptation procedure is then used to adapt the grid to this specified function. Again note that for this initial example of a discontinuous function the adaptation is based upon a specified function that is not dependent on the grid.

In Fig. 7.2 the second adaptive grid is shown after the adaptation procedure is applied. The mesh in the region of the discontinuity in Fig. 7.2 is quite coarse and wide since the cell across the discontinuity in Fig. 7.1 is very large which results in small and diffusive second derivatives. This is obvious since the magnitude of the jump in the function is fixed and small grid size results in large derivatives. This artificial discontinuous function is reasonable since many discontinuities such as a shock have a nearly stable difference across them no matter how well they are resolved. The function in Eq. 7.1 is discontinuous at one unique $x$ coordinate, which means that in the refined mesh in Fig. 7.2 the function will be discontinuous across several cells and it is constant

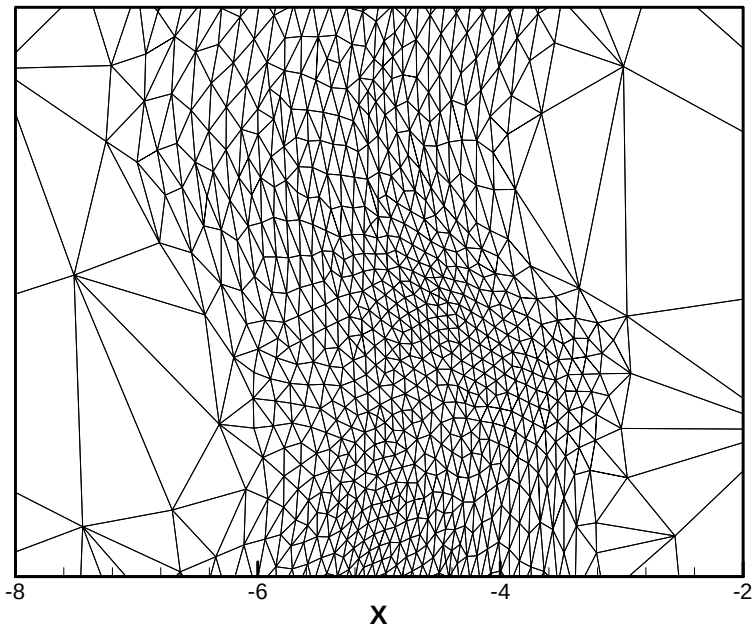



*Figure 7.2 Artificial Discontinuity – Grid after Second Iteration*

at most of the elements that have been previously added. Only several cells sitting across the discontinuity have a large Hessian.

As the refinement proceeds, these cells are refined in each iteration and those added previously away from the discontinuity keep being removed. The process will never end provided that the machine accuracy allows. The grid after 20 iterations is shown in Fig. 7.3. It can be seen that gradually the points far from the jump are removed and only a narrow zone is refined. The Hessian is getting larger and larger across the discontinuity and in the rest of the domain it is zero. It is worthwhile to note that during the adaptation a maximum physical length is applied. An edge whose physical length is above the criterion will not be removed, thereby preventing the grid from becoming nonsense. In this case the maximum length is 0.3 (referred to the scale in Figs. 7.1-7.3.) A better strategy to achieve this will be given at the end of this chapter. All points will eventually be pulled into the discontinuity region if no limit on the maximum physical

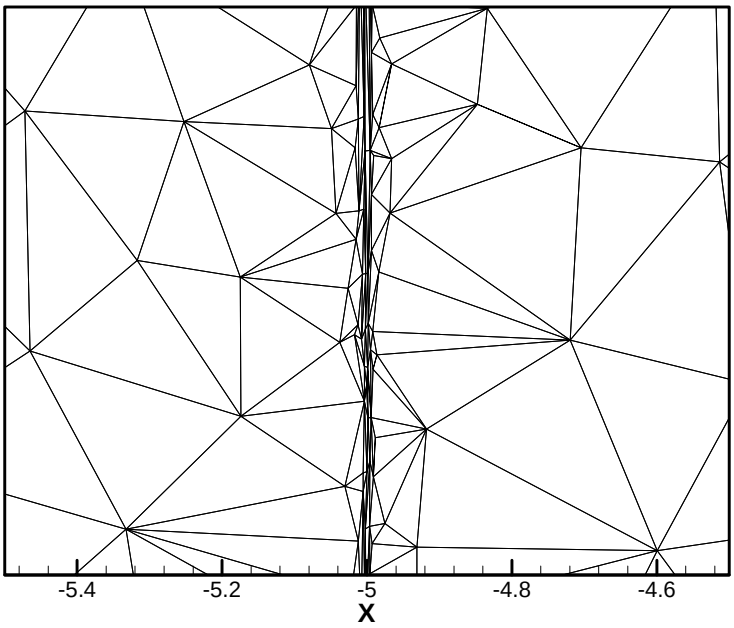



*Figure 7.3 Artificial Discontinuity – Grid after 20 iterations*

edge length is imposed.

## 7.2 Discontinuity-Enhanced Hessian

To circumvent the infinite refinement of a discontinuity, a restriction is placed on the minimum Euclidean edge length in the computational domain. Since the eigenvalues are inversely related to the Euclidean edge length, this will impose a corresponding maximum magnitude on the eigenvalues of the Hessian.

The eigenvalues of the Hessian are obtained by multiplying by the left and right eigenvector matrices to obtain the diagonal matrix, $\mathbf{\Lambda}$ ,

$$\mathbf{R}^{-1}\mathbf{H}\mathbf{R} = \mathbf{\Lambda} = diag(\lambda_1, \lambda_2) \tag{7.2}$$

where $\lambda_i$ represents the two eigenvalues of the Hessian. Only two dimensions are considered here for simplicity. To limit the range of the eigenvalues, an allowable maximum eigenvalue, $\lambda_{\max}$, is defined according to some pre-set formula. (The details used to determine its value are given below.) The limited eigenvalue, $\lambda_{i,\lim}$, is then defined as,

$$\frac{1}{\lambda_{i,\lim}} = \frac{1}{\lambda_i} + \frac{1}{\lambda_{\max}} \tag{7.3}$$

which is similar to the harmonic average of the two and gives

$$\lambda_{i,\lim} = \frac{\lambda_{\max}\lambda_i}{\lambda_{\max} + \lambda_i}. \tag{7.4}$$

The art of limiting in Eq. 7.4 lies in that the pre-specified maximum, $\lambda_{\max}$, will almost not affect the eigenvalues of the Hessian if they are much smaller than $\lambda_{\max}$, but $\lambda_i$ is close to or larger than $\lambda_{\max}$, $\lambda_{i,\lim}$ will approach $\lambda_{\max}$. The limited Hessian is then defined as

$$\mathbf{H}_{\lim} = \mathbf{R}\begin{pmatrix} \lambda_{1,\lim} & 0 \\ 0 & \lambda_{2,\lim} \end{pmatrix}\mathbf{R}^{-1} \tag{7.4}$$

The value of $\lambda_{\max}$ can be determined in the same manner as the "wall" Hessian introduced in the last chapter. Once the average edge Riemann length, $\bar{l}^R$, is found and an allowable minimum Euclidean length of the edges, $l^E_{\min}$, is picked, $\lambda_{\max}$ is given by

$$\lambda_{\max} = \left(\bar{l}^R / l^E_{\min}\right)^2 \tag{7.5}$$

In Fig. 7.4 the mesh resulting from limiting the maximum eigenvalues of the Hessians is shown. The minimum Euclidean length of the edges is set to 0.1. Rather than being overly refined in the region of the discontinuity, the minimum grid spacing is well controlled and an anisotropic grid can be seen across the discontinuity.

Another phenomena in anisotropic adaptation across discontinuities observed here is that the grid across the shock becomes small in both directions first. It will then become thin if the length in one of the directions is limited. This can be seen in Figs. 7.3 and 7.4. In Fig. 7.3 although the cells are very thin in the normal direction of the discontinuity, they are still small in the parallel direction. When the maximum edge length is applied, the size of cells in the parallel direction is approaching the allowed maximum cell size, which is 0.3 in this case. This indicates that the Hessian is becoming more accurate since ideally it should be zero in the parallel direction.

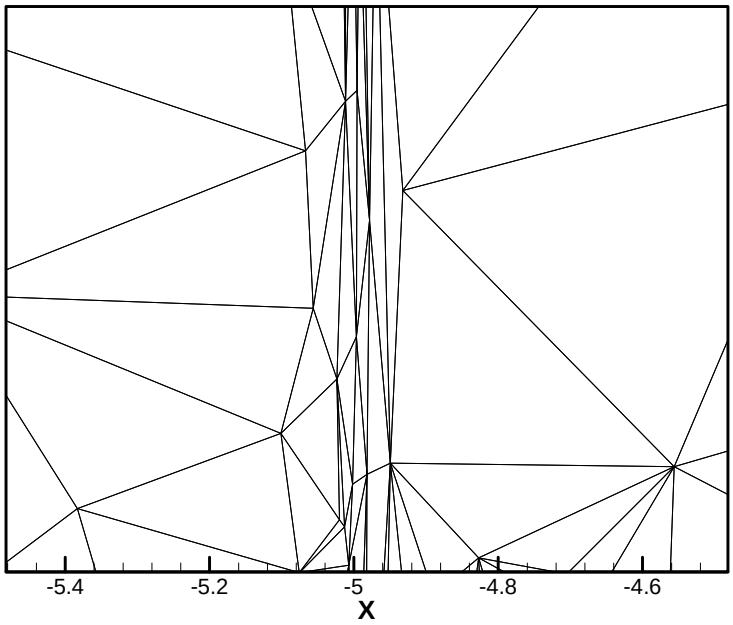
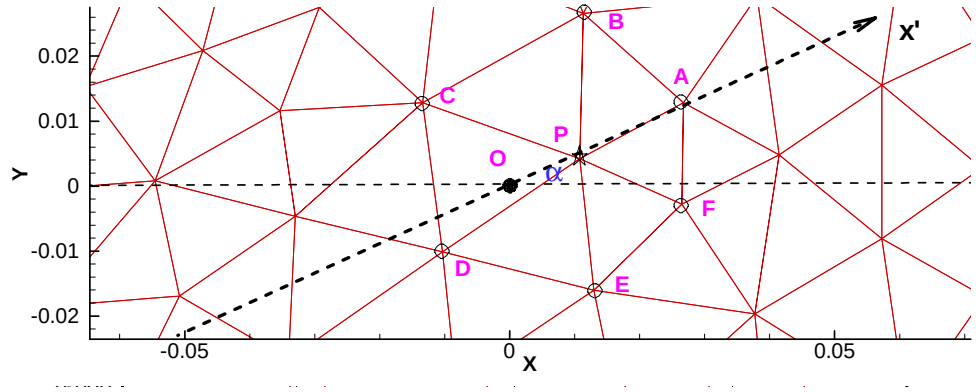


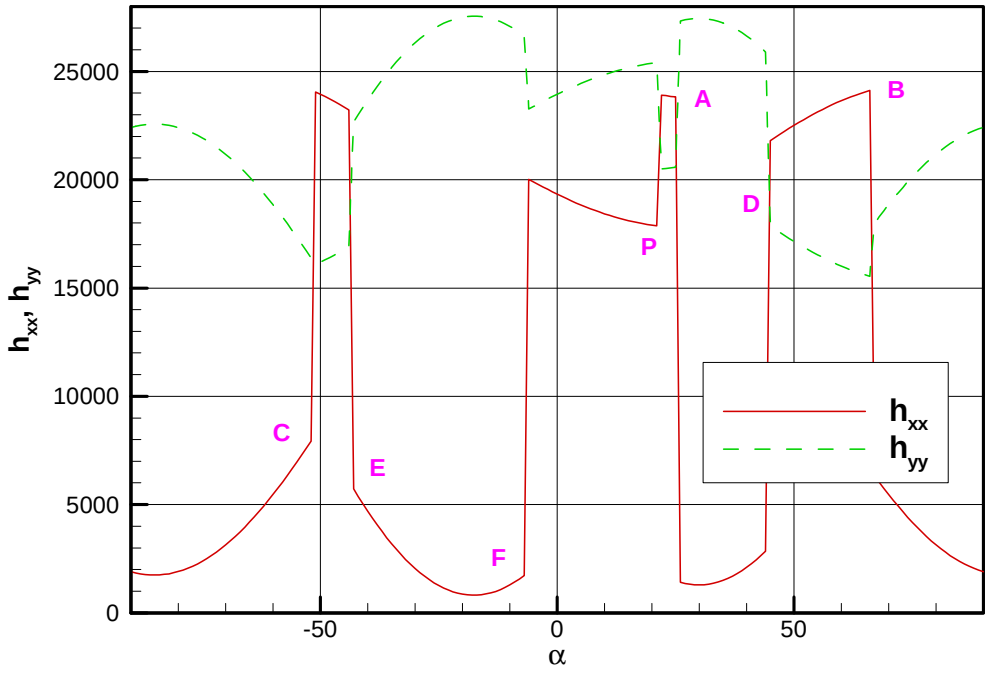

*Figure 7.4 Artificial Discontinuity –Grid After 20 Iterations with Maximal Eigenvalue Control*

The phenomenon of the discontinuity refinement can be explained more clearly from the calculation of the Hessian in different grids. In Fig. 7.5, a discontinuous function is set up, which is one for $y > 0$ and zero for $y \le 0$. The Hessian on point *P* is calculated. It can be seen that the second derivatives at *P* depend upon the values of the function at points *P, A, B, C, D, E* and *F*. In order to discover how the alignment of the elements affects the Hessian at *P*, the coordinate is rotated through an angle $\alpha$. Since the function is discontinuous across the *x*-axis, it can be expected that the Hessian will vary more dramatically when the *x'*-axis travels across *P* and each of the surrounding points. In the lower part of Fig. 7.5 the magnitudes of the Hessian in the axial directions are shown. The corresponding points on the mesh are also marked in the plot. The Hessian varies considerably, particularly in the *x*-direction. Although the second derivatives parallel to the discontinuity should be zero, its computed value along the discontinuity (the *y*' direction) is sometimes even larger than that across the discontinuity (the *y*-direction) where an isotropic grid is used. Although occasionally the aspect ratio can reach two at some locations as one can deduce from the Hessian in Fig. 7.5, this explains why the adapted grid from a discontinuity on an isotropic mesh results in a nearly isotropic grid,

To assess the impact of increasing grid aspect ratio, the *x* coordinate is fixed and the *y* coordinate is multiplied by 0.1. The resulting grid and Hessian are shown in Fig. 7.6. In the stretched grid, the Hessian becomes more sensitive to the angle of rotation. Compared with the second derivatives on the isotropic grid, the Hessian is accurate within a small region between two angles. One of them is formed by the vector $\overrightarrow{OB}$ with the horizontal axis and other one is by $\overrightarrow{OE}$. The other points, however, become less important in the evaluation. This indicates that the calculation of the second derivatives

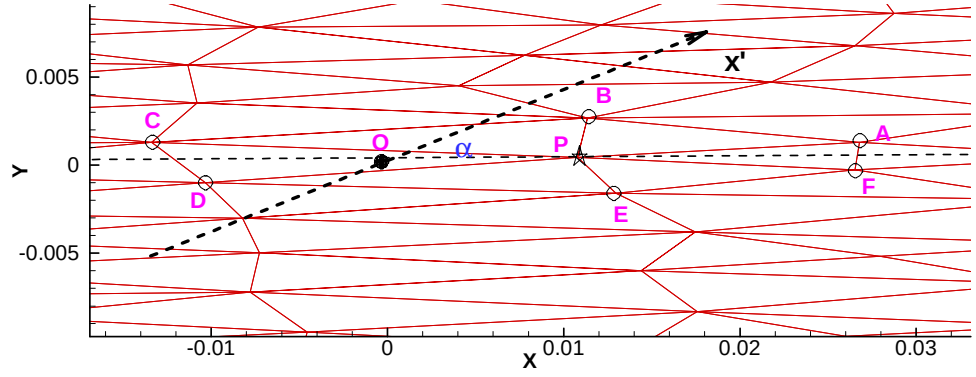



*(a)*

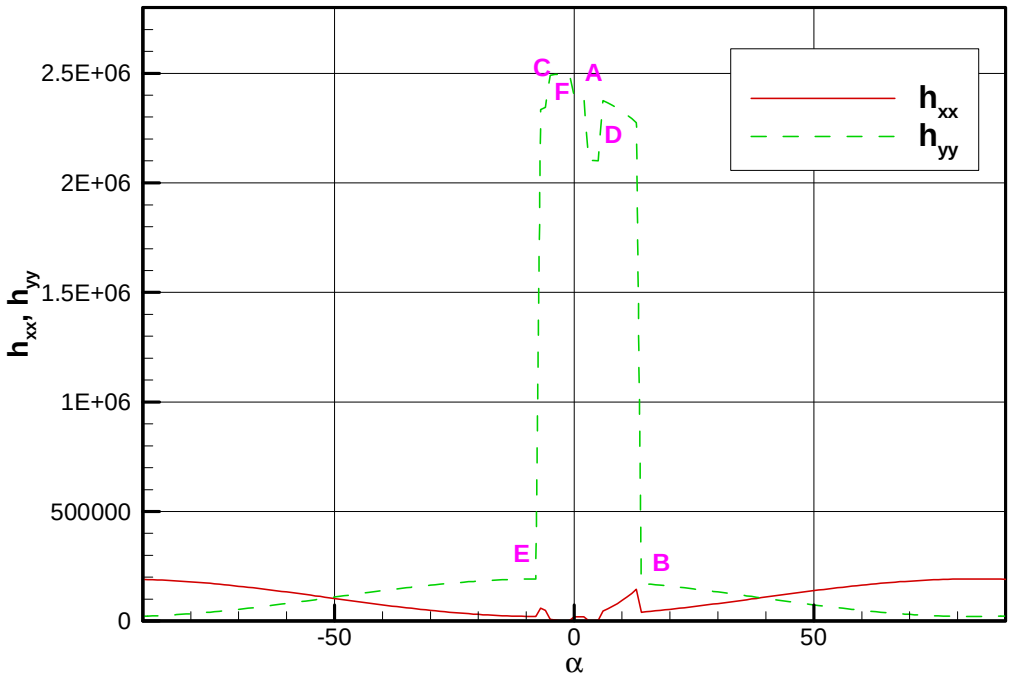



*(b)*

*Fig. 7.5 Hessian Evaluation on an Isotropic Grid. (a) Grid, (b) Hessian*




*(a)*




*(b)*

*Fig. 7.6 Hessian Evaluation on an Anisotropic Grid (AR = 10).*

*(a) Grid, (b) Hessian*

of a directional function is more accurate on a stretched grid, provided that the mesh is aligned with the function. In Fig. 7.7, a grid aspect ratio of 1000 is shown. This increased aspect ratio gives a narrower angle in which the discontinuity must fall for the Hessian to be accurate and the errors of the second derivative in the *x* direction to be negligible.

This example explains the growth of the anisotropic elements in the adapted mesh. The number of stretched cells is small on the initial mesh, but will grow steadily as the Hessian is improved at iteration. The adaptation of the discontinuity is thus a gradual process.

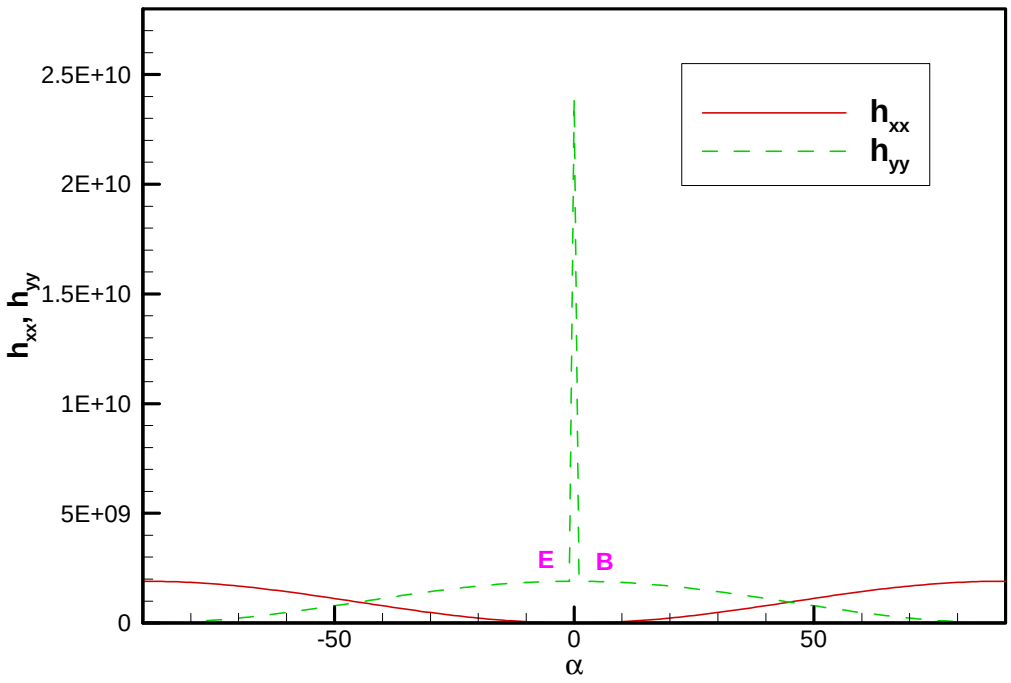



*Fig. 7.7 Hessian Evaluation on an Anisotropic Grid (AR = 1000)*

## 7.3 Difficulties in Realistic Discontinuity Adaptation and Their Remedies

Although a stationary discontinuity can now be adapted with the mesh above, this mesh is not practical for realistic flows with discontinuities, such as shocks. An example is next given to show how coupled shock adaptation differs from the uncoupled discontinuity given above. In Fig. 7.8 an isotropic mesh is used as the initial mesh for the calculation and adaptation of supersonic flow around airfoil NACA0012. This mesh is similar to the one used in the last chapter and starts with about 2000 points. A Mach number of 2.0 and zero attack angle are chosen in the case studied. A zoomed view of the initial and the adapted mesh in front of the leading edge is shown in Figs. 7.9 to 7.13, as well as the Mach numbers interpolated from the solution along the center line ( $y = 0$ ) of the flow approaching the airfoil. The number of points is increased to 6000 in the second step, as shown in Fig. 7.10. It is then frozen in the following iterations. The values of the Mach number can be identified from the axis located at the right of the plot. In Fig. 7.9, it can be seen that the solution is diffusive and the shock is limited in a region about two to three cells wide. In Fig. 7.10 only the region within this width is refined with nearly isotropic mesh. As noted before, the reason why an anisotropic mesh does not appear is that the Hessian calculated on the mesh in Fig. 7.9 is diffusive.

As the adaptation proceeds through additional iterations, the region near the discontinuity is further refined. The second derivatives in the normal direction continue to increase because of the finer grid spacings, and the second derivatives further reduce the grid spacings. The mesh across the shock becomes thinner and thinner until the specified minimum length is reached while most of the points outside the shock are

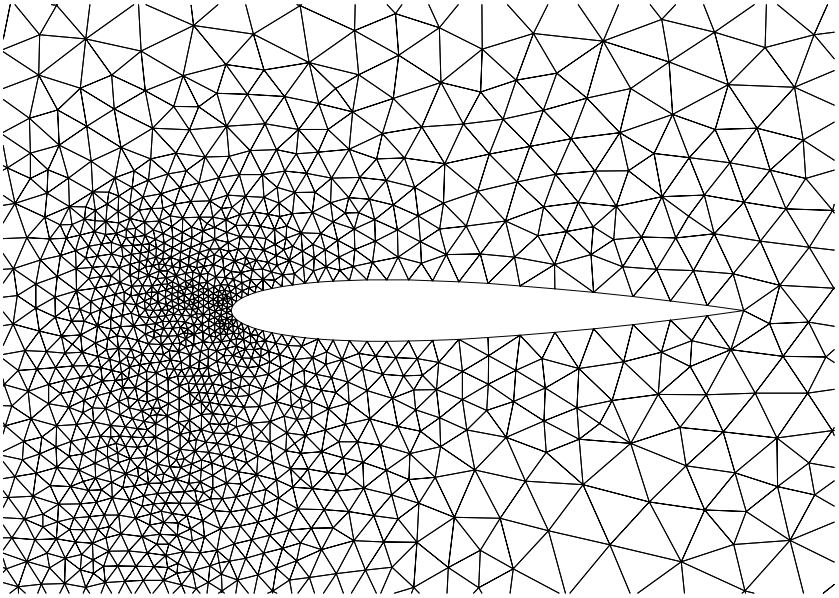


*Fig. 7.8 Supersonic flow around NACA 0012 – Initial Mesh*

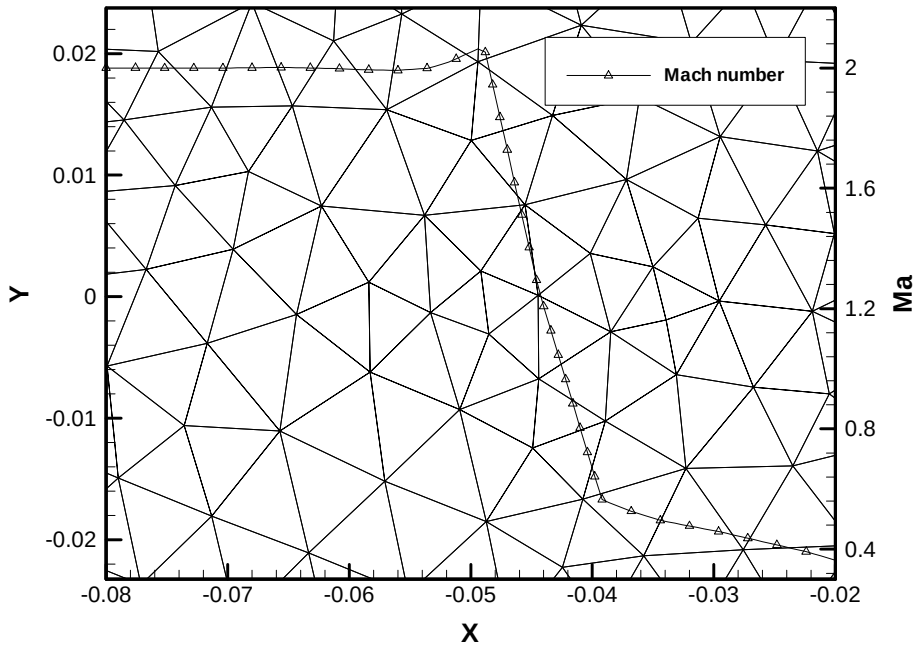



*Figure 7.9 Front of Bow Shock Region and Interpolated Mach Numbers – Initial Mesh*

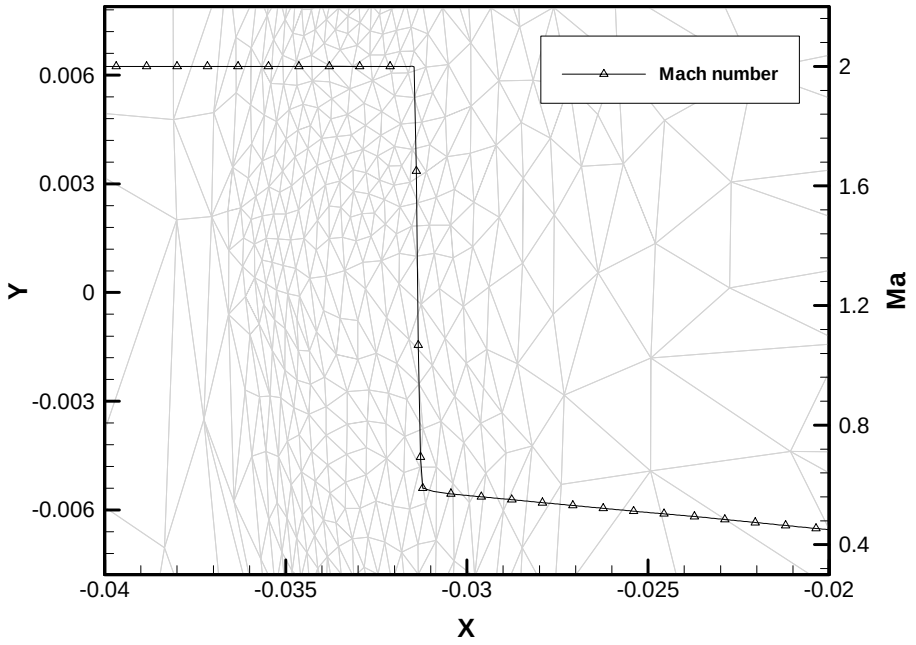



*Figure 7.10 Front of Bow Shock Region and Interpolated Mach Numbers – First Iteration*

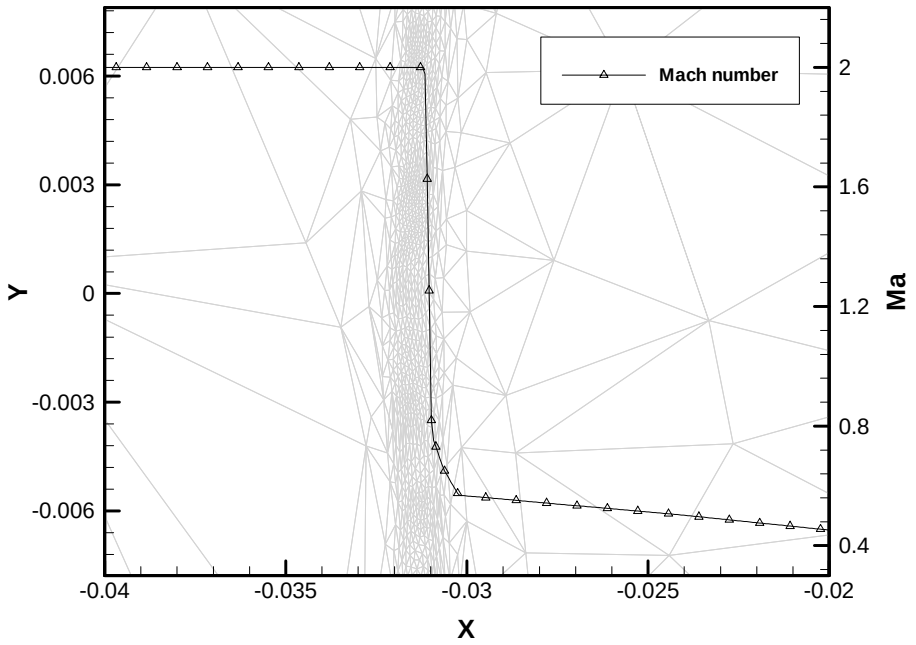



*Figure 7.11 Front of Bow Shock Region and Interpolated Mach Numbers – Second Iteration*

removed. Here the allowable minimum length across the discontinuity is set as $10^{-5}$ times the airfoil chord. Further adaptation iteration causes the shock to leave the center of the refined zone, as seen in Fig. 7.12. This means that the CFD solver requires a wider refinement to encompass the discontinuity. In Fig. 7.13 the locations of the shocks in the above iterations are shown. The convergence of the adaptation is lost finally due to the very thin shock zone refined.

In order to keep the discontinuity fixed, the region being refined should not be allowed to become so thin that it only covers a very small number of layers of elements. It is expected that on the final mesh the shock region will cover more layers rather than two or three such that the shock will still stay inside the refinement even it moves in the following iterations. One possible remedy is to smooth the Hessian such that the refined

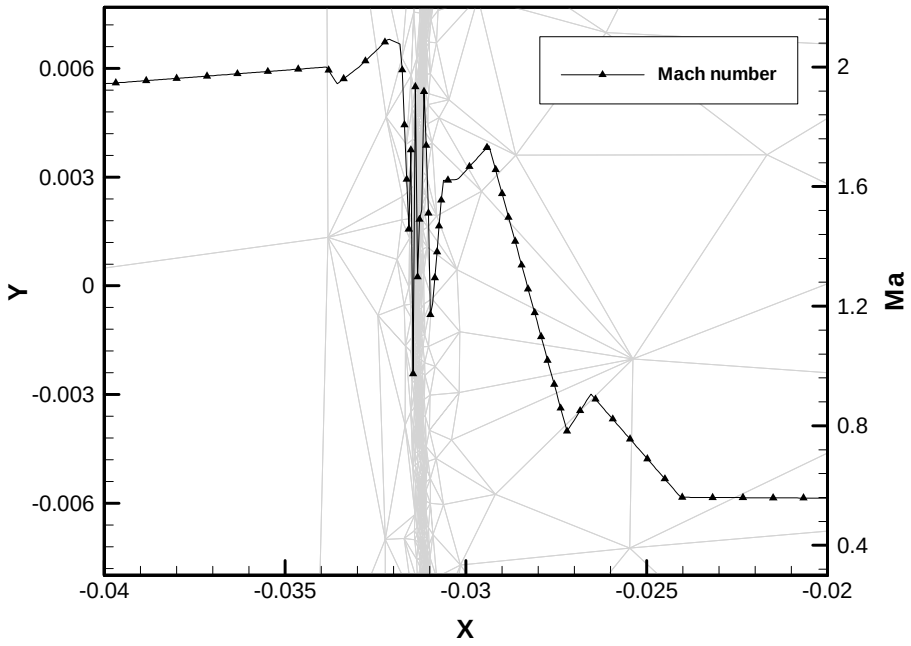



*Figure 7.12 Front of Bow Shock Region and Interpolated Mach Numbers – Fifth Iteration*

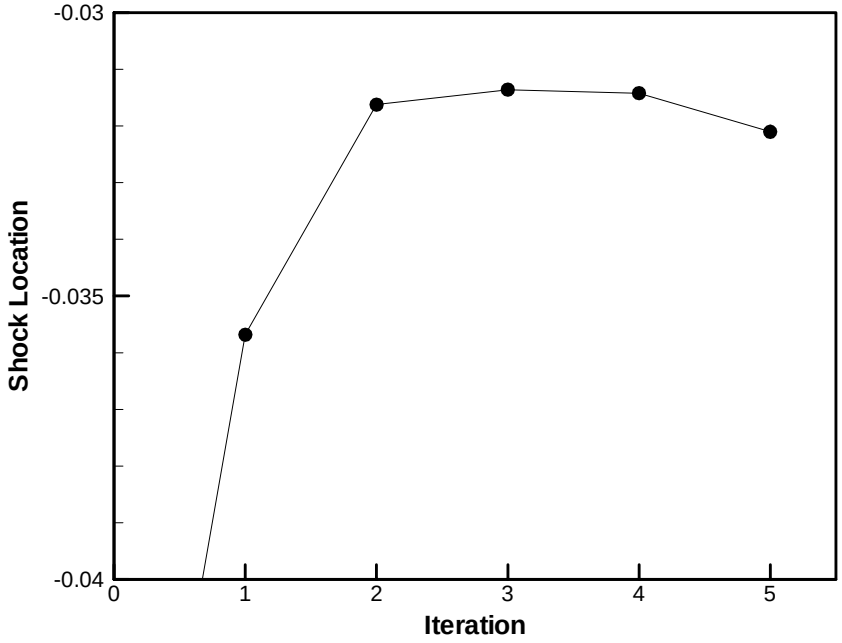



*Figure 7.13 Locations of the Shock*

region is fine enough to resolve the discontinuity but yet thick enough to keep the shock inside. A Laplace equation is solved with each element of the Hessian as the variable $h_{ij}$,

$$\Delta h_{ij} = 0 \qquad (i=1,ndim, j=1,ndim) \tag{7.6}$$

where *ndim* is the number of dimensions. In two dimensions the equation is solved by adding a time-dependent of $h_{ij}$, in the form

$$\frac{\partial h_{ij}}{\partial t} + \frac{\partial^2 h_{ij}}{\partial x^2} + \frac{\partial^2 h_{ij}}{\partial y^2} = 0\,. \tag{7.7}$$

The spatial discretization of Eq. 7.7 is similar to the evaluation of the second derivatives in the Hessian and thus the scheme in Appendix A is used. The time derivative is treated implicitly. Extension of the smoothing on the Hessian is controlled

by the time step and number of iterations. The Hessian is usually smoothed for 30 iterations and time step is determined by a fixed von Neumann number (here defined as $\Delta t / r^2$, and the length scale $r$ is the shortest of the three heights of a triangle) in the field, which is around 5. After adaptation with the smoothed Hessian, the mesh in the front of the bow shock and the Mach number are shown in Fig. 7.14. The limitation of the maximum edge length, which is again the same as in Fig. 7.12, is applied after the smoothing process is done. The thickness of the refined region is well controlled and the shock is reasonably resolved. Note that the same scale is used in Fig. 7.14 as in Fig. 7.12. The width of the refined zone is increased by a factor of nearly 10. Although the shock itself lies across two or three layers of cells and many other cells are "wasted", the mesh in Fig. 7.14 does represent a more reasonable grid for CFD solvers. The global mesh and Mach contours of the solution are shown in Fig. 7.15 and Fig. 7.16 respectively.

Although the mesh and result shown in Fig. 7.15 and 7.16 suggest that the adaptation captures the major characteristics of the flow, such as the bow shock in front of the airfoil, the trailing shock and the wake, careful examination of the adapted mesh shows that it is less satisfying. Zoomed views of the different parts of the bow shock are shown in Fig. 7.17 and 7.18. In Fig. 7.17 the mesh in the front of the bow shock is almost isotropic, which means a lot of the grid points in the tangential direction of the shock are wasted, considering the highly directional feature of the normal discontinuity. On the other side, the mesh adapted at the rear part of the bow shock in Fig. 7.18 shows that it is rather anisotropic, but too coarse compared with that in the front. It seems that the refinement in front of the leading edge has not reached the stage that the above limiting effects. Points have been attracted mostly by the shock here, rather than evenly distributed at different locations of the discontinuity.

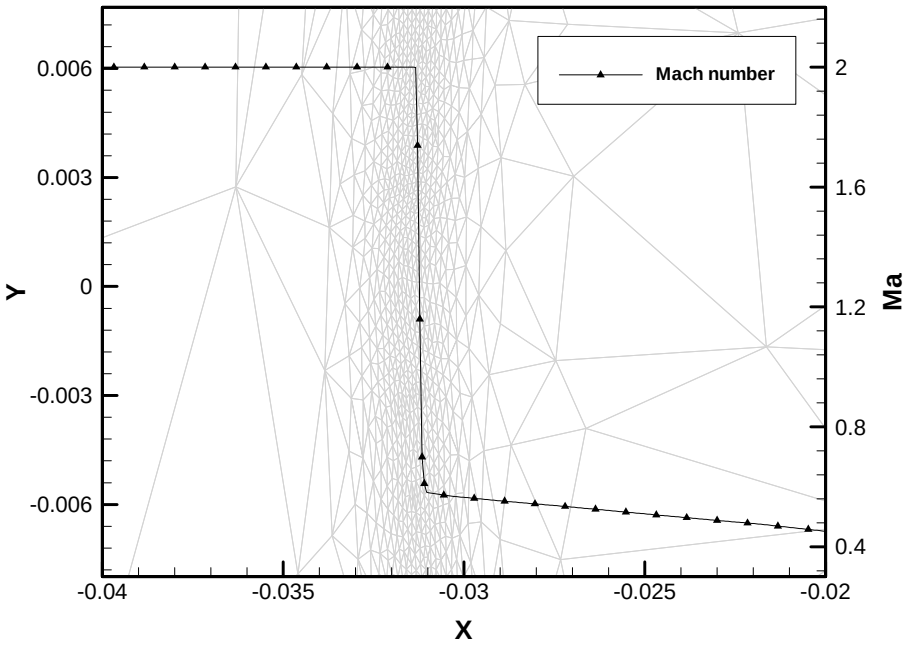



*Figure 7.14 Front of Bow Shock Region and Interpolated Mach Numbers – Mesh with Smoothed Hessian*

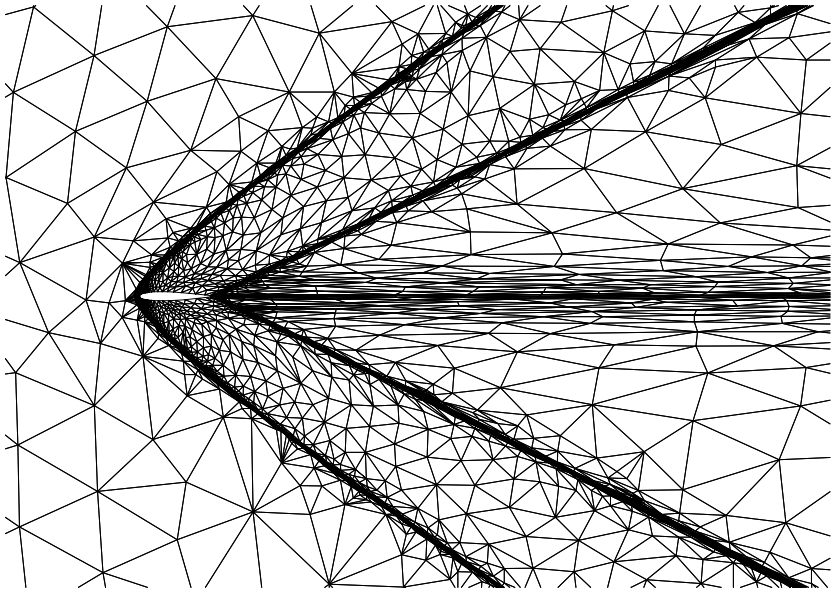


*Figure 7.15 Mesh Adapted from the Smoothed Hessian*

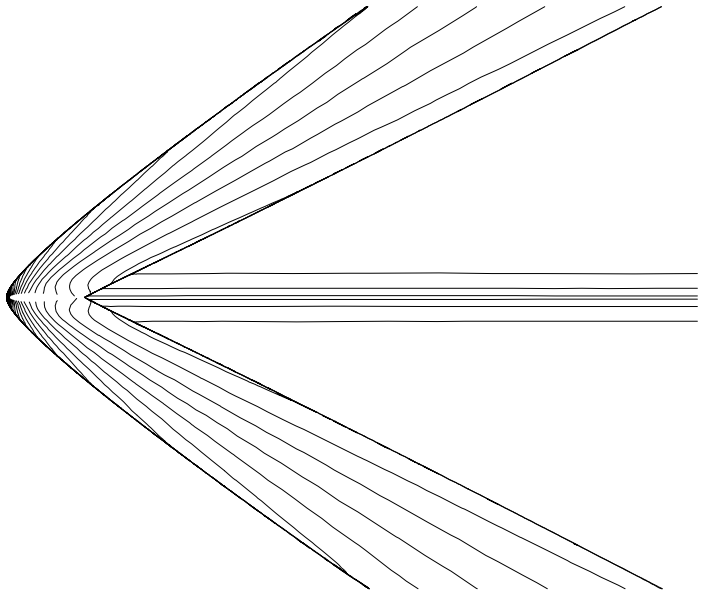


*Figure 7.16 Contours of Mach Number Calculated with the Mesh in Fig. 7.15*

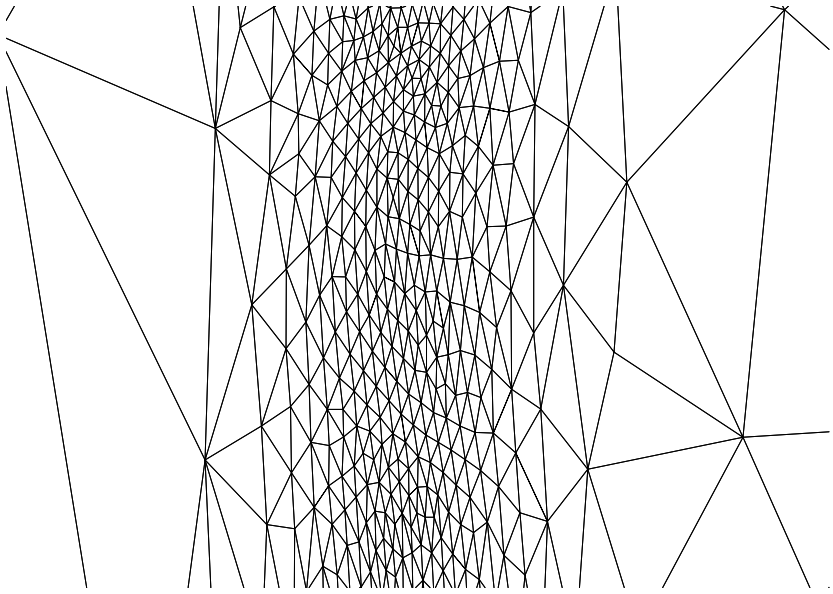


*Figure 7.17 Front of Bow Shock Region in the Mesh Adapted from Smoothed Hessian*

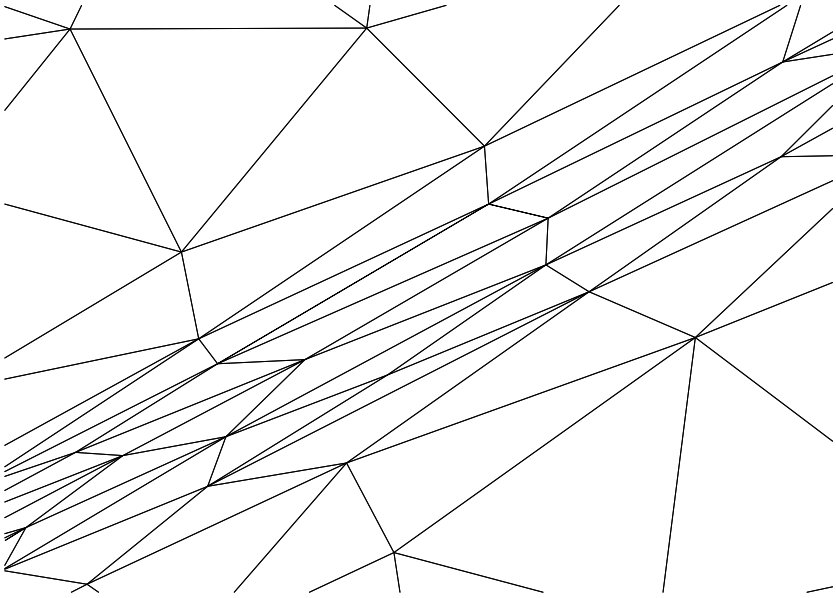


*Figure 7.18 Rear Part of Bow Shock in the Mesh Adapted from Smoothed Hessian*

In order to better explain the discontinuity refinement, the strength of a discontinuity is defined here as the magnitude of the jump across the discontinuity. A strong discontinuity is thus the one with a large jump. The Hessian, which is calculated based on the local Mach number, determines the mesh across the oblique shock in Fig. 7.18 is coarser since an oblique shock is much weaker than the normal shock in front of the leading edge, due to the smaller Mach number drop. A large number of grid points are drawn into the region of the stronger shock, leaving the rest of the discontinuity under-refined. Since a shock is a discontinuity, it is more reasonable to adapt the mesh equally everywhere across the discontinuity regardless of its strength.

In Fig. 7.19 the manner in which discontinuities of different strengths are smoothed is illustrated. The upper part of the figure represents a Hessian calculated from a weak discontinuity while the lower part is from a strong one. When the Laplace equation is solved over the entire domain, the results of the two Hessians will be

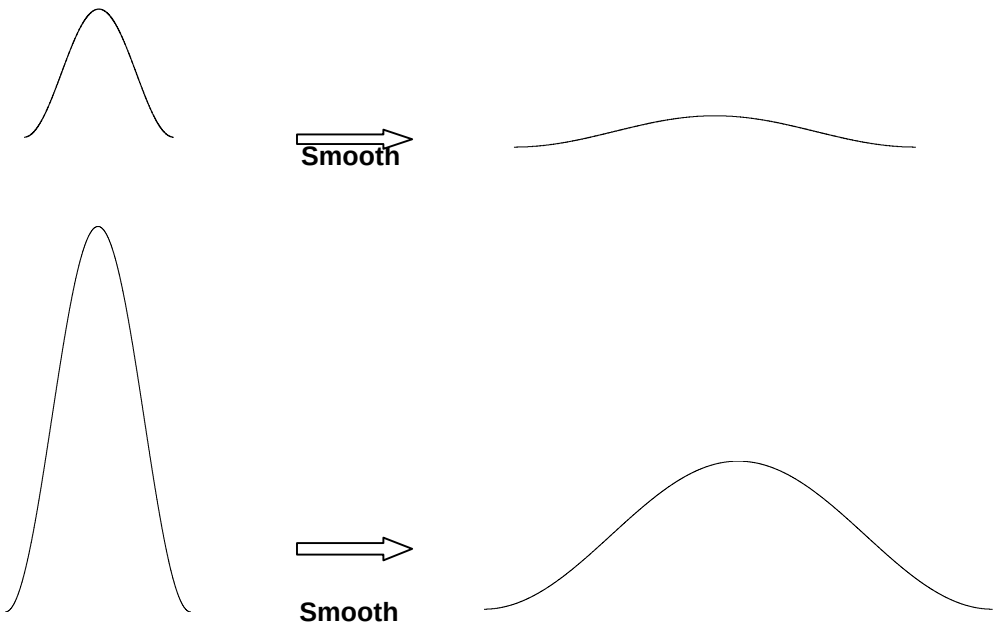



*Figure 7.19 Smoothing the Discontinuity Equally*

different, as can be seen in the figure. The mesh adapted from this smoothed Hessian will be too fine near strong discontinuity, and too coarse near the weak one, as shown above. The stronger shock results in a fine, almost isotropic mesh because the second derivatives in the tangential direction are also very large. The large second derivatives in both directions compared with other part of the domain results in overly refined elements in the wrong direction of the normal shock while the rest is under-refined.

A new approach is adopted here for the smoothing of the second derivatives. The purpose is to smooth the Hessian such that the resulting matrices will be almost equal everywhere in the shock. Note that here equality of Hessian means their eigenvalues are comparable. The procedure shown in Fig. 7.20 explains how this is accomplished.

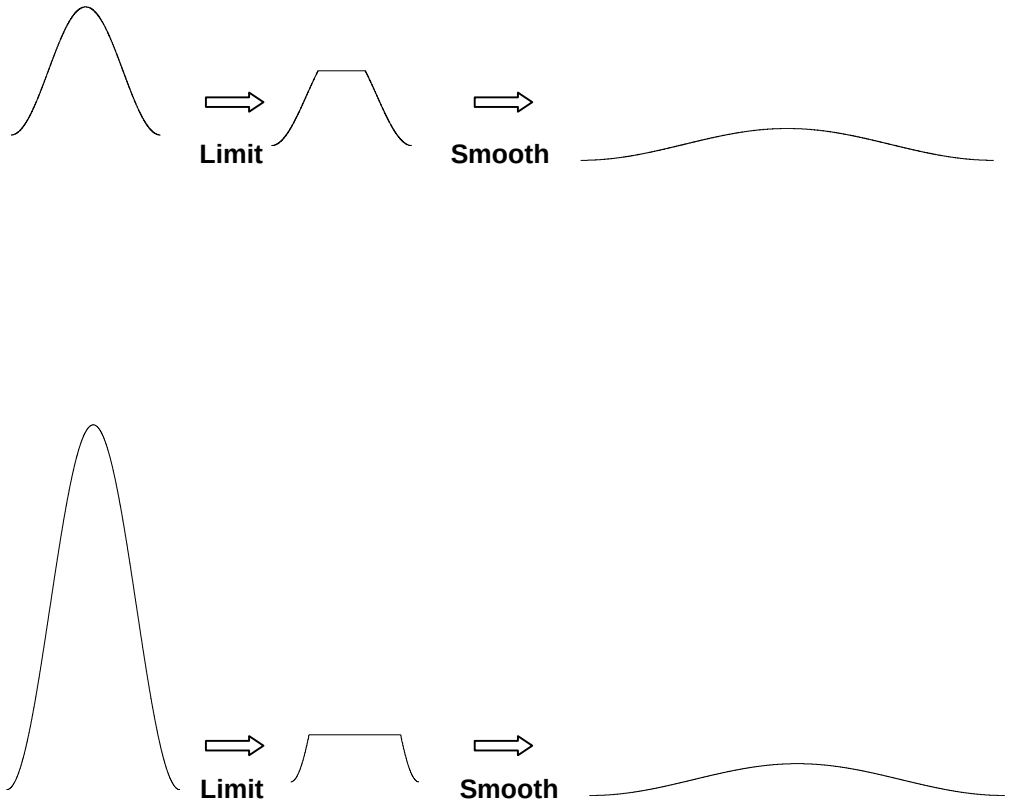



*Figure 7.20 New Approach to Smooth and Make Shocks Equal*

The magnitudes of the Hessian everywhere in the flow field are first limited and then smoothed. The limit here is applied in the same way the minimum length of the mesh is controlled in the earlier part of this chapter. However, the purposes of the two limits are different. The limit described earlier is to control the local minimum length while the latter is to equalize the Hessian across the discontinuity before smoothing. Although the calculated Hessian is disparate at different locations of the discontinuity, such as those two in Fig. 7.20, the limiting process makes them almost equal. This is followed by a smoothing process which makes the refining region wide enough to stabilize the discontinuity.

When calculated from the Mach numbers in the grid points, the Hessian is actually different from those before being smoothed in Fig. 7.19. There is an inflection point across the shock and the second derivatives in the normal direction look like that in the left of Fig. 7.21. After choosing the positive-definite eigenvalues of the Hessian, the profile across the shock will be like the one in the right of Fig. 7.21. In this approach, the Hessian is first smoothed by solving the Laplace equation for a few iterations, such that the inflection points inside the shock are smeared. The above revised limiting and smoothing processes are applied thereafter. Finally the limiting of the maximum allowable Euclidean edge length limitation is imposed to give the desired spacing across the shock. These steps are mathematically described as,

- Smooth the Hessian to remove the inflection point inside the discontinuity, shown in Fig. 7.21. This is done by solving,

$$\frac{\partial h_{ij}}{\partial t} + \frac{\partial^2 h_{ij}}{\partial x^2} + \frac{\partial^2 h_{ij}}{\partial y^2} = 0 \tag{7.8}$$

  for a few steps for each element of $|\mathbf{H}|$.

- Limit the Hessian, such that Hessian across all discontinuities is almost "equal".

$$|\mathbf{H}| = \mathbf{R}\begin{pmatrix} \lambda_{1,\lim} & 0 \\ 0 & \lambda_{2,\lim} \end{pmatrix}\mathbf{R}^{-1} \tag{7.9}$$

where

$$\lambda_{i,\lim} = \frac{\lambda_{\max}\lambda_i}{\lambda_{\max} + \lambda_i}. \tag{7.10}$$

The value of $\lambda_{\max}$ has to be determined manually during the adaptation. This means one has to examine the flow field and find the eigenvalue of the Hessian at the location where the discontinuity is the weakest.

- Smooth the resulting Hessian for more time steps. The equation here is that of 7.8.

- Limit the Hessian to give the desired spacing across the shock. The same formulas in 7.11 and 7.12 are used. $\lambda_{\max}$, however, is determined by

$$\lambda_{\max} = \left(\bar{l}^R / l^E_{\min}\right)^2 \tag{7.11}$$

where $l^E_{\min}$ is the minimum edge length allowed in the discontinuity. The resulting Hessian is denoted as $\mathbf{H}_{\lim}$.

The results after the above process are shown in the figures below. In Fig. 7.22 the discontinuity-enhanced Hessian gives a far better refinement and aspect ratio when compared with those in Fig. 7.17. In this case the average aspect ratio in the center of the shock region is bigger than 10, while in Fig. 7.17 the value is around 2. Fig. 7.23 shows the same region as in Fig. 7.18, which is under-refined since most of the points are drawn

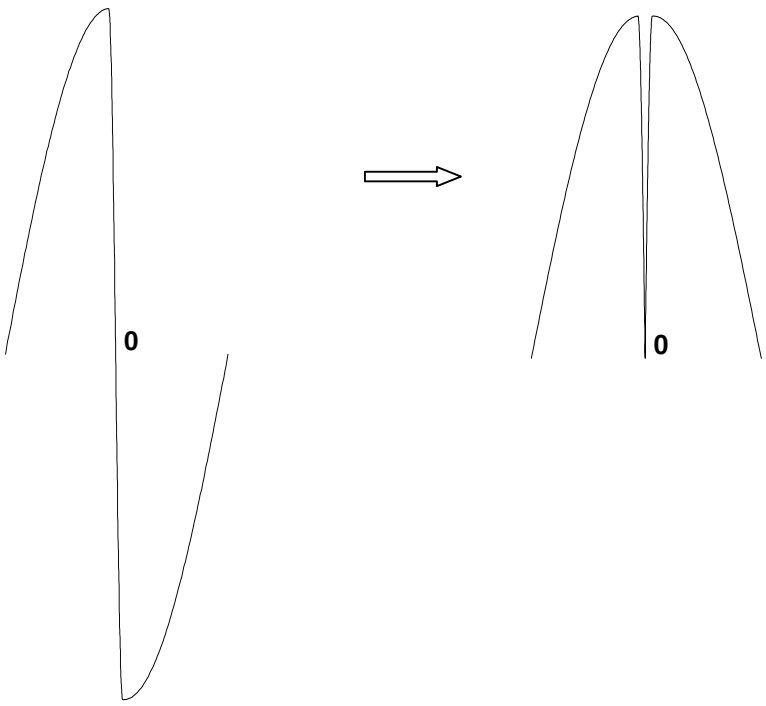



*Figure 7.21 Inflection Point in the Center of Shock*

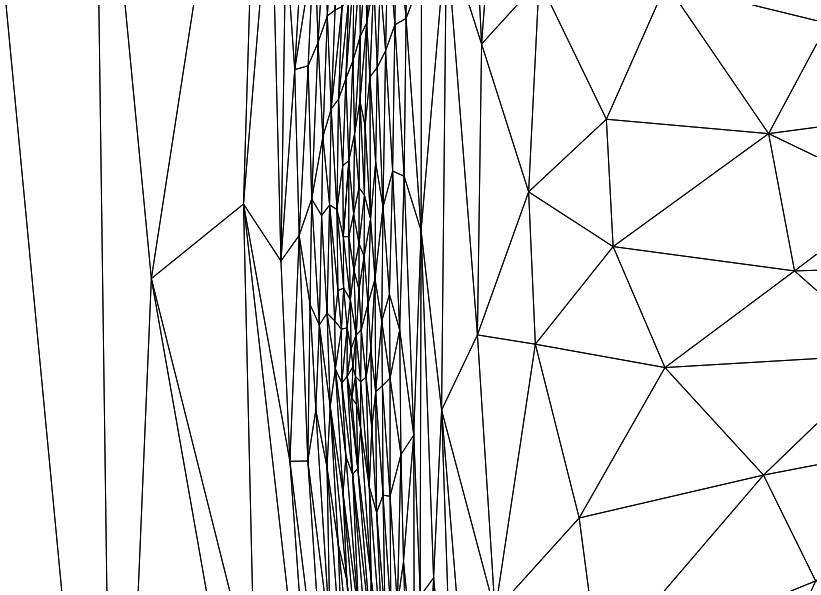


*Figure 7.22 Front of Bow Shock Region in the Mesh Adapted from Discontinuity-Enhanced Hessian*

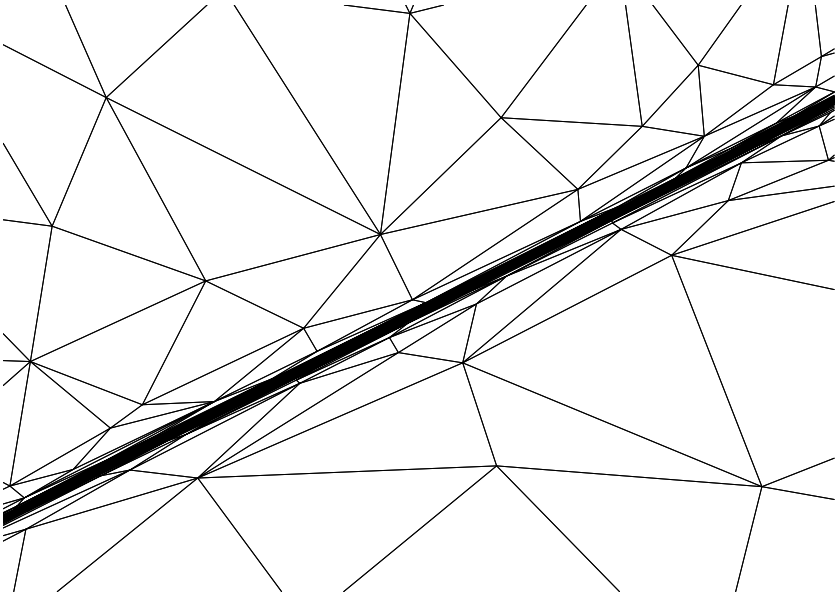


*Figure 7.23 Rear part of Bow Shock in the Mesh Adapted from Discontinuity-Enhanced Hessian*

into the discontinuity region with larger Hessians. Shocks now are equally refined by choosing the discontinuity-enhanced Hessian, which means the widths of the shocks are almost identical. The aspect ratios usually are not equal since they depend on the curvature of the shocks.

The resultant mesh is shown globally in Fig. 7.24 and the Mach number contours are shown in Fig. 7.25. Although it is difficult to detect the improvement in the computation in these global contours, the improvement is really seen from more detailed pictures. The contours in front of the bow shock on the two different meshes, as in Fig. 7.26 and 7.27, show the improvement more clearly. The revised method leaves the shock better refined with higher-aspect-ratio grid in a thinner region. The points saved from this region are moved into other part of the shocks such that the whole domain is better adapted.

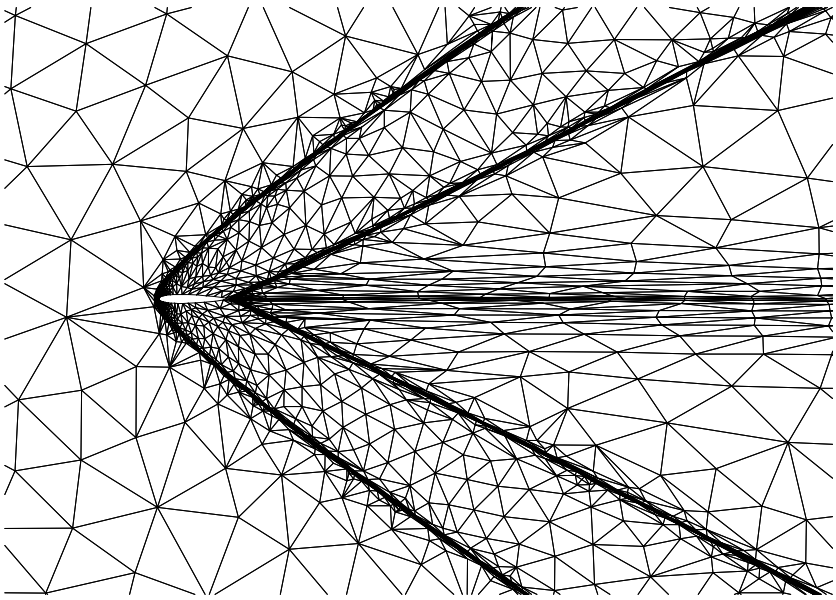


*Figure 7.24 Mesh Adapted from the Discontinuity-Enhanced Hessian*

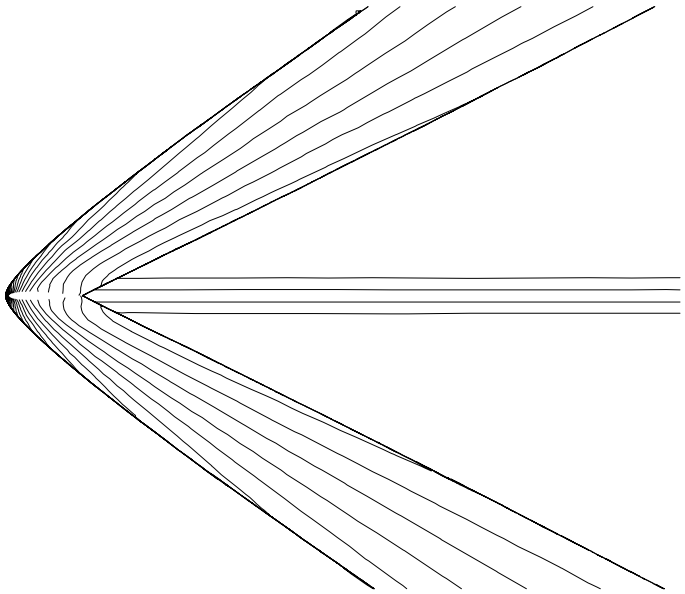


*Figure 7.25 Contours of Mach Number Calculated with the Mesh in Fig. 7.24*

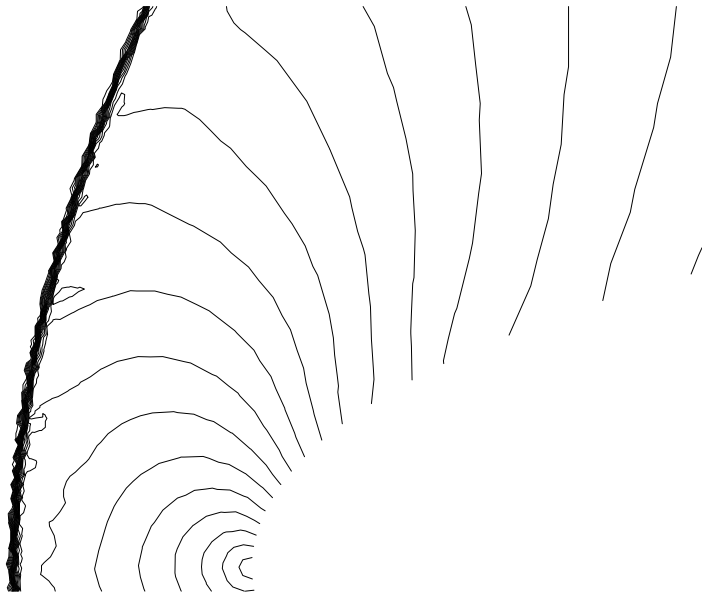


*Figure 7.26 Contours of Mach Number around the Leading Edge (Smoothed Hessian)*

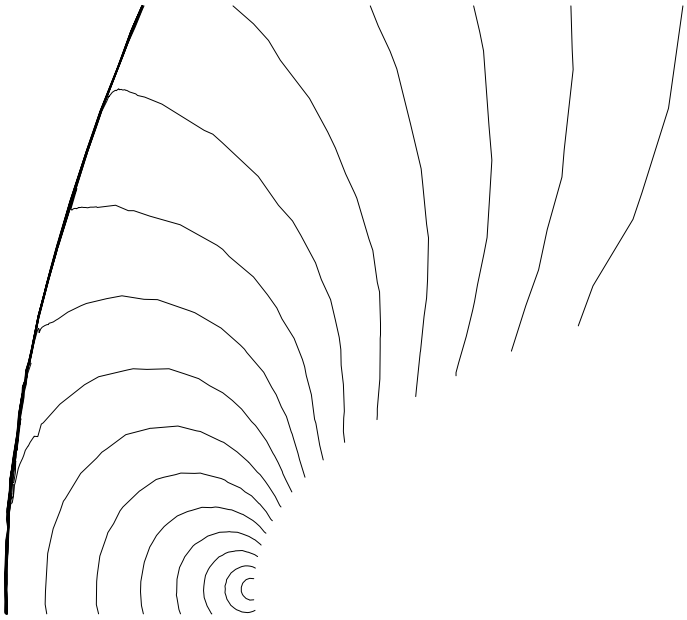


*Figure 7.27 Contours of Mach Number around the Leading Edge (Discontinuity-Enhanced Hessian)*

## 7.4 Modification for Uniform Flow Regions

In a uniform flow region both the Hessian and the solution gradient are zero. Uniform flow is often seen in supersonic flow where flow in front of the shock is essentially the same everywhere. The vanishing Hessian in front of the bow shock implies that all lengths in the Riemann plane vanish no matter what their Euclidean length so that there is no longer a mechanism for controlling the grid density. In principle a uniform flow region does not require resolution. In practice, however, generally some maximum grid size and some degree of grid uniformity are desired in this region. At the beginning of this chapter a maximum Euclidean edge length is imposed during the

adaptation which prevents the removal of an edge that is already very long physically. A new approach that can be applied more easily is introduced in this session. This is again through the manipulation of the Hessian.

To accomplish this control, a minimum eigenvalue limit, $\lambda_{\min}$, times the identity matrix is added to obtain the modified Hessian,

$$\mathbf{H}_{\text{mod}} = \mathbf{H}_{\text{lim}} + \lambda_{\min}\mathbf{I} \tag{7.12}$$

The value of $\lambda_{\min}$ is determined by

$$\lambda_{\min} = \left(\bar{l}^{R} / l^{E}_{\max}\right)^2 \tag{7.13}$$

where $l^{E}_{\max}$ is the maximum Euclidean length of edge that is allowed. The modified Hessian is then used in the adaptation.

In a region of non-uniform flow, the addition of the identity matrix will cause a minor change in the eigenvalues of the Hessian. In a uniform flow region, the eigenvalues are just $\lambda_{\min}$, while the principal directions are aligned with the coordinates (the free-stream). To show the improvement of the grid in front of the bow shock in Fig. 7.24 in which the maximum Euclidean length was enforced, a similar solution is shown in Fig. 7.28 where it is not used. The grid in Fig. 7.28 still contains cells in front of the shock, but they are generated because of the non-removable points on the boundary that are retained to maintain the integrity of the domain. The vanishing Hessian in front of the shock leads to the distorted mesh and unacceptable grid sizes that are not observed in the improved results in Fig. 7.24.

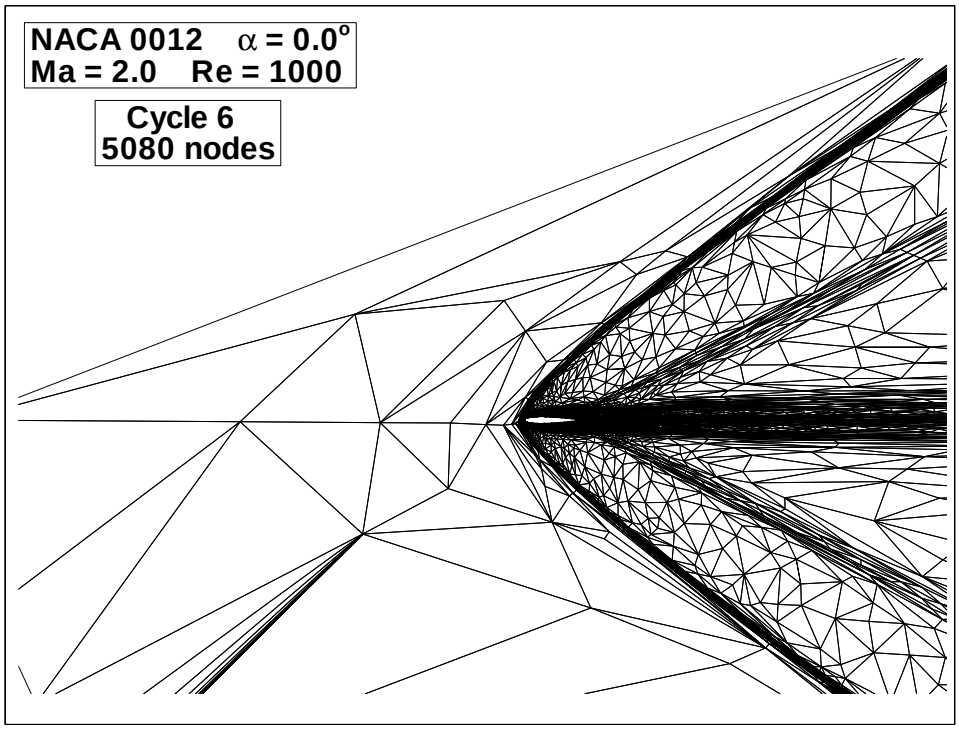



*Figure 7.28 Adaptation without Uniform Flow Modification*

# Chapter 8

# Results of Enhanced Anisotropic Grid Adaptation

The adaptation approach enhanced by the methods developed for boundary layer and discontinuities is tested by a sequence of calculations in this chapter. These include flows with various parameters, such as subsonic, transonic and supersonic flows, and different geometries, such as internal and external flows. The purpose of these calculations is to evaluate the capability of the current method to adapt the grid to practical problems.

The adapted grid satisfying the equal length criteria is not guaranteed to meet the requirements of Computational Fluid Dynamics calculations. Concerns about anisotropic adaptation have arisen in the accuracy of the computations based on adapted meshes [39,56-60]. In this chapter, the accuracy of computation on the adapted mesh has been studied by comparing the results with experimental data. Quality issues related to adapted mesh are tested by evaluating the calculation of FLUENT [94], a commercial CFD solver.

## 8.1 Enhanced Approach for Anisotropic Adaptation

Adaptation follows the sequence below when a converged solution is obtained on the previous mesh.

- Calculate the Hessian from chosen variable, $\mathbf{H} = \left(h_{ij}\right)$ where

$$h_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} . \tag{8.1}$$

The variable *f* should provide information for important flow features.

- Find the positive-definite Hessian $\left|\mathbf{H}\right|$,

$$\left|\mathbf{H}\right| = \mathbf{R}\left|\mathbf{\Lambda}\right|\mathbf{R}^T . \tag{8.2}$$

The transformation matrix **T** at each point can also be defined as

$$\mathbf{T} = \left|\mathbf{\Lambda}\right|^{1/2}\mathbf{R}^T \tag{8.3}$$

- For flows with boundary layer, start the boundary layer enhancement procedure, which specifies the near wall grid size by defining a "wall" Hessian.

$$\mathbf{H}_w = \mathbf{P}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\mathbf{P}^{-1} \tag{8.4}$$

where $\lambda_2$, the eigenvalue associated with the eigenvector normal to the wall, is given by

$$\lambda_2 = \left(\bar{l}^R / l_{y'}^E\right)^2 . \tag{8.5}$$

and $\bar{l}^R$ is the average Riemann length of all edges and $l_{y'}^E$ is the desired wall spacing. In a turbulent flow calculation, it is usually determined by a constant $y^+ = K$,

$$l_{y'}^E = K\sqrt{\nu/(\partial u'/\partial y')} \tag{8.6}$$

The eigenvalue parallel to the wall, $\lambda_1$, is specified by setting it equal to the eigenvalue associated with the eigenvector that is directionally closest to the tangent to the wall at the point where the Hessian is a maximum. The interpolation between the "wall" Hessian and the maximum Hessian replaces the original one in the near wall region.

- For flows with discontinuity, turn on the discontinuity enhancement approach. This is done with an initial smoothing the Hessian for a few time steps to remove the inflection inside the discontinuity, an initial limiting and a second smoothing procedures, resulting in comparable Hessians across the discontinuity everywhere within the enlarged zone. A maximum allowable grid size is specified with a minimum allowable eigenvalue in a second limiting step. The two smoothing processes are done by solving the Laplace equation

$$\frac{\partial h_{ij}}{\partial t} + \frac{\partial^2 h_{ij}}{\partial x^2} + \frac{\partial^2 h_{ij}}{\partial y^2} = 0 \tag{8.7}$$

and the limiting process

$$\mathbf{H}_{\lim} = \mathbf{R}\begin{pmatrix} \lambda_{1,\lim} & 0 \\ 0 & \lambda_{2,\lim} \end{pmatrix}\mathbf{R}^{-1} \tag{8.8}$$

is applied to the eigenvalues of the Hessian. Here $\lambda_{i,\lim}$ is defined as

$$\lambda_{i,\lim} = \frac{\lambda_{\max}\lambda_i}{\lambda_{\max} + \lambda_i}. \tag{8.9}$$

The Hessian is not smoothed and limited in the region close to the wall.

- For flows where any region of uniform variables exists, the Hessian is further modified as

$$\mathbf{H}_{\mathrm{mod}} = \mathbf{H}_{\lim} + \lambda_{\min}\mathbf{I} \tag{8.10}$$

where $\lambda_{\min}$ controls the maximum allowable maximum edge length on the mesh.

- The final Hessian, denoted as $\mathbf{H}_{adap}$, is stored on the previous mesh, and manipulations on it in order to equidistribute the metric edge length are done with procedures developed in Chapter 3.

- The updated mesh as well as the interpolated solution from the previous mesh is exported and used to start a new CFD calculation.

The above procedure is repeated until the mesh and solution no longer change. A converged adaptation is thus acquired.

## 8.2 Parametric Adaptations to Flow around NACA 0012 Airfoil

All adaptation cases for the flow around a NACA 0012 airfoil started from an isotropic initial grid show in Fig. 8.1 and were adapted through six cycles. Figures 8.2 and 8.3 show the results at four different angles of attack. The solver used is again FUN2D [39]. Mach numbers are used to evaluate the Hessian used in the adaptation. All four flows are at the same free stream Mach number ( $\mathrm{Ma} = 0.7$ ) and the same Reynolds

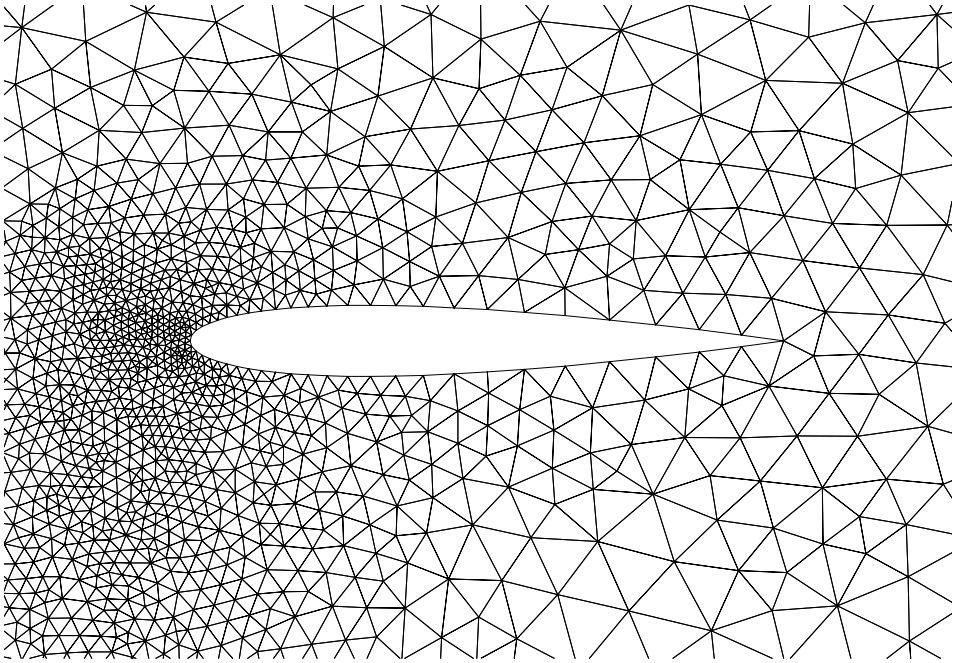


*Figure 8.1  Initial Mesh for Adaptation of Flow Around NACA 0012*

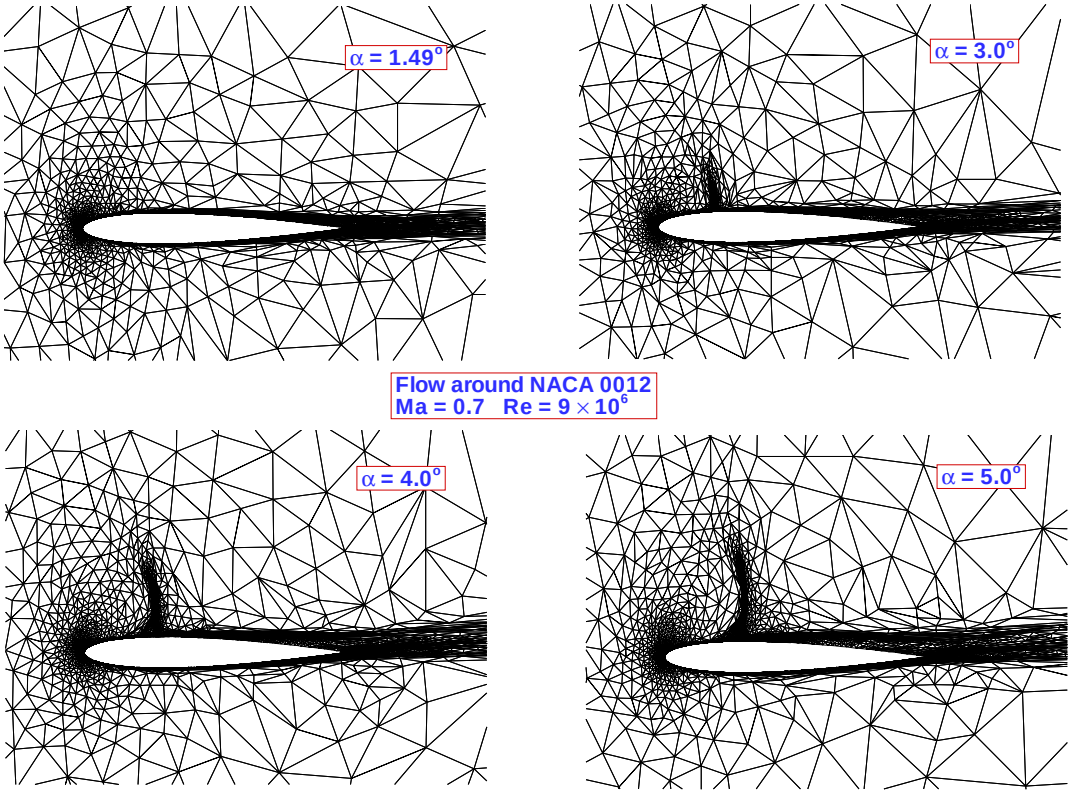



*Figure 8.2 Adaptation for Flow at Different Attack Angles around NACA 0012 (Mesh)*

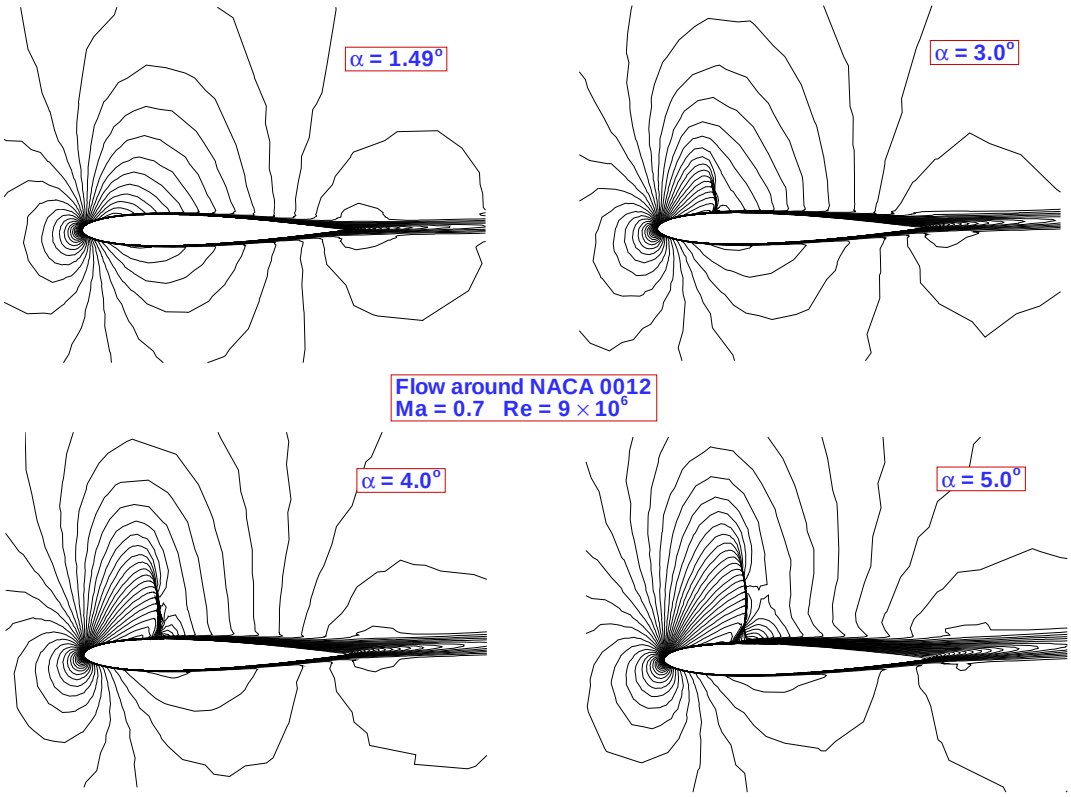



*Figure 8.3 Adaptation for Flow at Different Attack Angles around NACA 0012 (Mach Number Contours)*

number ( $\mathrm{Re} = 9 \times 10^6$ ), but the flow field goes from subsonic to transonic as the angle of attack is increased.

The results on Fig. 8.2 show the final adapted grid for angles of attack of $\alpha = 1.49$, 3.0, 4.0, and 5.0. The corresponding contour plots for the Mach number are given on Fig. 8.3. As both the Mach number contours and the adapted meshes show, the flow becomes transonic at $\alpha = 3.0$ degrees, and the transonic region continues to grow as the angle of attack increases. The enhancement for boundary layer adaptation is adopted here to improve the accuracy. Shock adaptation is also augmented with the discontinuity-enhanced Hessian, which prevents the infinite refinement and oscillation of the shock. The minimum Euclidean length allowed in the adaptation is $10^{-5} c$ ( $c$ is the airfoil chord length), which is effective in zones other than the boundary layer. This is done by limiting the Hessian at points far enough from the wall, which is judged by their distance to the wall. The Hessian is smoothed with 30 iterations to widen the refined zone near the shock. The time step is usually adjusted for each individual case to give the best result. The results are for a relatively small number of nodes (approximately 5000), but the Mach number contours and the boundary layer are reasonably well resolved. The grid also shows good refinement in the shock and boundary layer/wake regions.

Figures 8.4 and 8.5 show a second series of parametric calculations, again for a NACA 0012 airfoil, but this time the free-stream Mach number is changed. Adaptation also started from the isotropic grid in Fig. 8.1. Four different Mach number conditions have been computed, Ma = 0.5, 0.7, 0.799, and 2.0. Note that the Reynolds numbers vary considerably in these four examples, but overall the subsonic, transonic and supersonic free-stream Mach numbers show the global characteristics of the adaptation scheme for a variety of flow field conditions. In the first case (upper left plots in Figs. 8.4

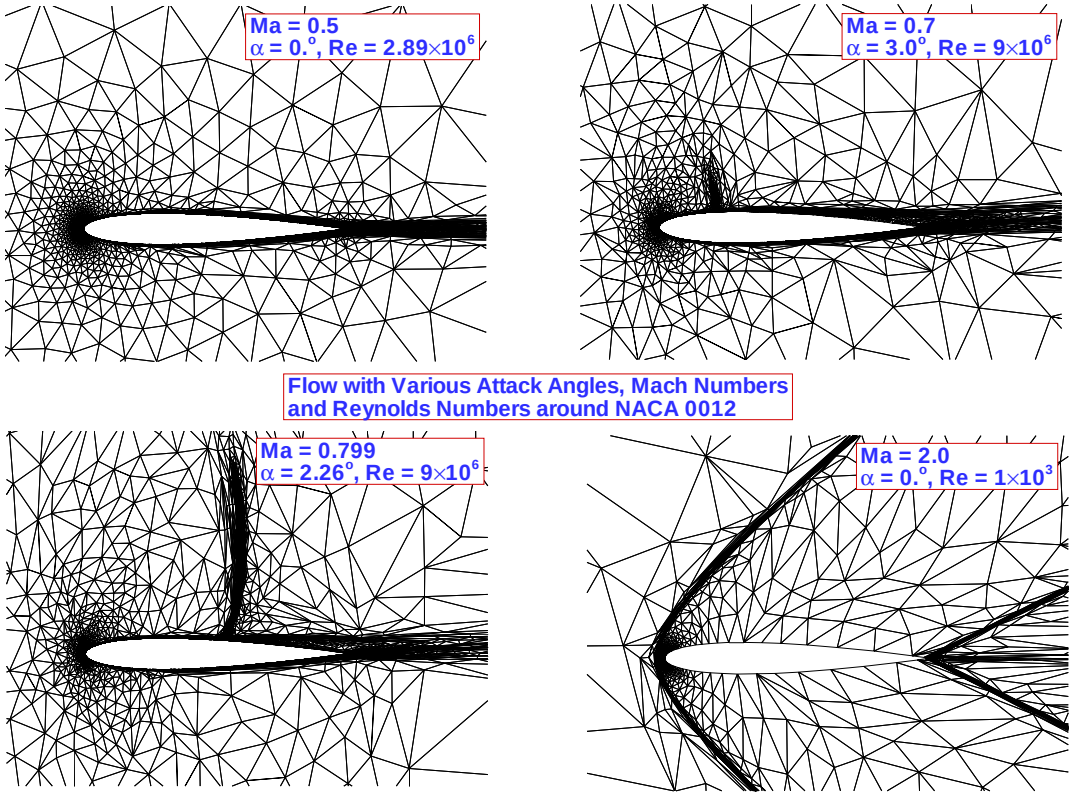



*Figure 8.4 Adaptation for Flow at Different Mach Numbers and Reynolds Numbers around NACA 0012 (Mesh)*

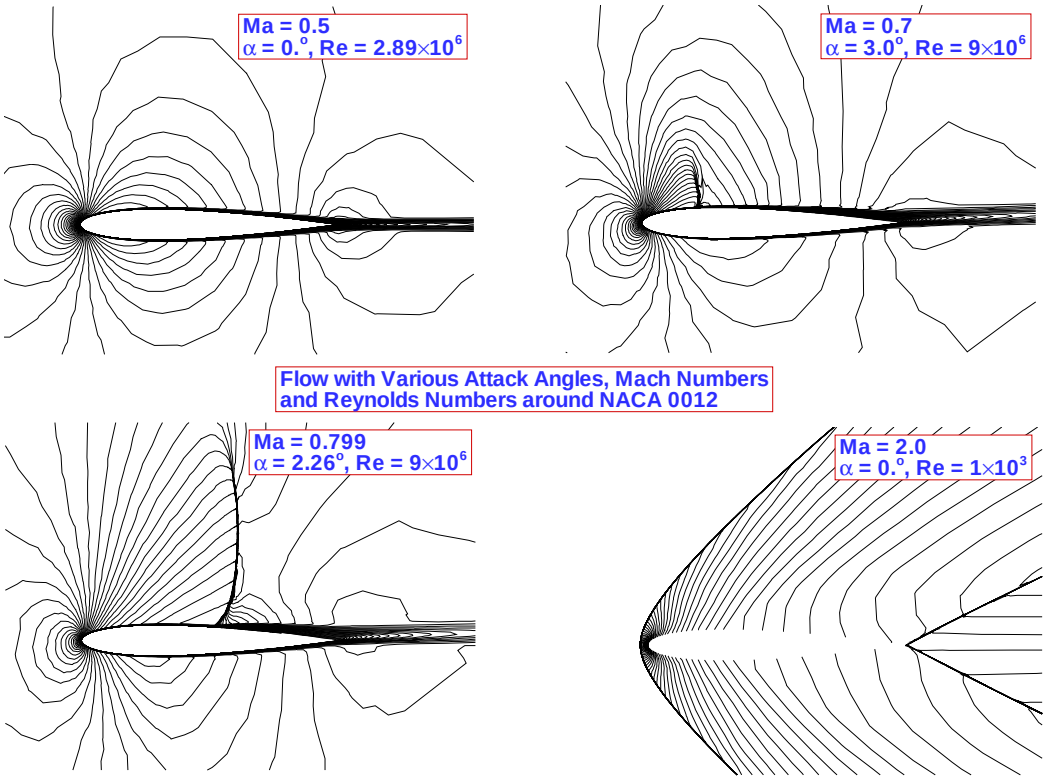



*Figure 8.5 Adaptation for Flow at Different Mach Numbers and Reynolds Numbers around NACA 0012 (Mach Number Contours)*

and 8.5), zero degree angle of attack is chosen, the Mach number is 0.5 and the Reynolds number is set as $2.89\times10^6$. The boundary layer is well resolved. The spacing of the first point off the wall is corrected by the "wall" Hessian, which is defined according to the criterion of $y^+=1$. Since no shock appears in this case, the discontinuity enhancement is not used here and the Hessian is not smoothed. In the second case in the upper right plot, the Mach number is 0.7 and the Reynolds number is $9\times10^6$. The attack angle of 3 degrees generates a small transonic zone above the airfoil. In the lower left plot the increased Mach number induces a stronger normal shock. In both plots the discontinuity enhancement is used and helps to stabilize the refinement of the normal shocks. The allowed minimum edge length is $10^{-5}c$, leaving enough grid points for the boundary layer. The same wall spacing correction is used as that in the first plot. The last case shown in the lower right plot in the supersonic flow with Mach number set to 2. The flow is inviscid and zero angle of attack is chosen. With the discontinuity enhancement the shocks, including the front bow shock and trailing shock, are equally refined. The allowed minimum length is again $10^{-5}c$, which prevents the attraction of all the points to the discontinuity regions. The Hessian in the uniform flow region in front of the bow shock is modified to give an isotropic grid whose size is around $0.3\ c$.

## 8.3 Transonic Flow around NACA 0012 Airfoil and Comparison with Experimental Data

The adaptation of the transonic flow past a NACA 0012 airfoil at a Reynolds number of nine million, one of the cases shown in last section, is further investigated.

The final mesh shown in Fig. 8.6 contains approximately 16000 nodes. The Mach number contours are shown in Fig. 8.7. As stated in the last section, the shock wave generates a narrow separated region on the upper surface of the airfoil that has a tendency to cause the shock wave to become unusually sensitive to the mesh distribution. The technique described in Chapter 7 is again used in this case. The discontinuity-enhanced Hessian makes the grid region across the shock become wide enough that the shock remains inside the fine grid. The allowed minimum physical length of the edges in regions other than the boundary layer is $10^{-5}c$ . The smoothing is done with 30 iterations and the VNN number is set as 5. With the enhancement implemented, the shock quickly settled to a stationary location. The adaptation procedure begins to increase the grid aspect ratio in the shock region after the refinement reaches the minimum cell size. Grids with high aspect ratio along the shock can be seen in Fig. 8.6.

The value of $y^+$ of the first grid point off the wall is set as unity. The Hessian computed inside the boundary layer, which is the boundary-layer-enhanced Hessian described in Chapter 6, is the interpolation between the "wall" Hessian and the maximum Hessian inside the boundary layer. The Hessian in this region does not need smoothing while that outside the boundary layer does. Excluding the smoothing from the boundary layer region during the adaptation is done by comparing the distance of grid points to the wall with prescribed criterion. In this case the criterion is an estimated thickness of the boundary layer, which is around $10^{-2}c$ . This enables us to capture the shock quite well without losing the boundary layer features.

Comparisons with experimental data [61] are shown in Fig. 8.8. The dashed line represents the calculation on the initial isotropic mesh (Fig. 8.1). The widely discrepant

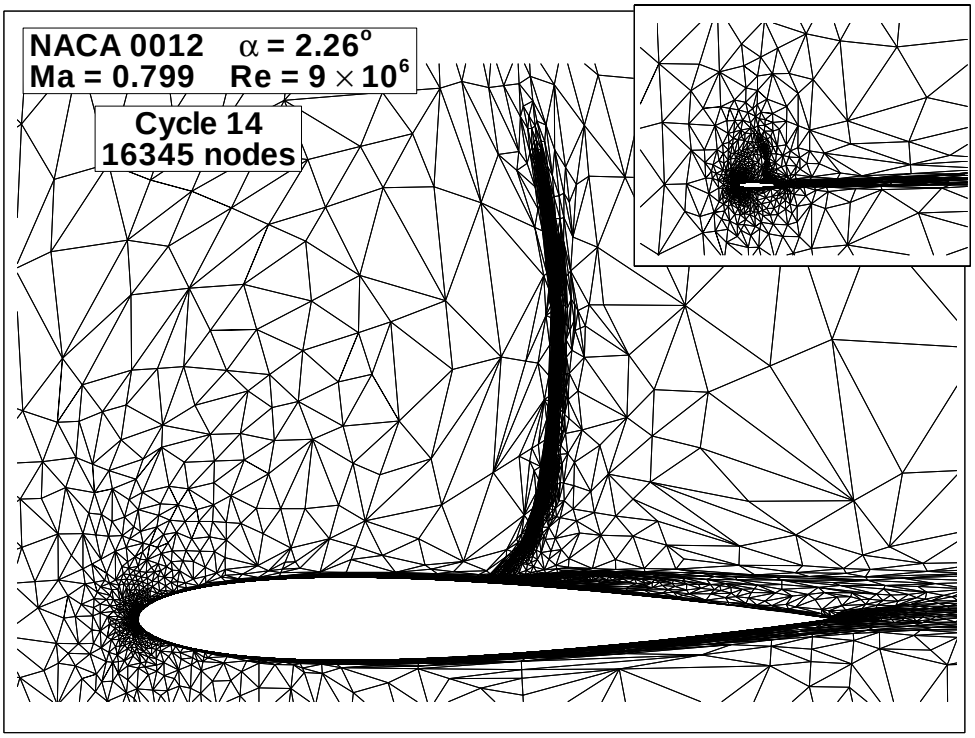



*Figure 8.6 Adaptation for Transonic Flow around NACA 0012 (Mesh)*

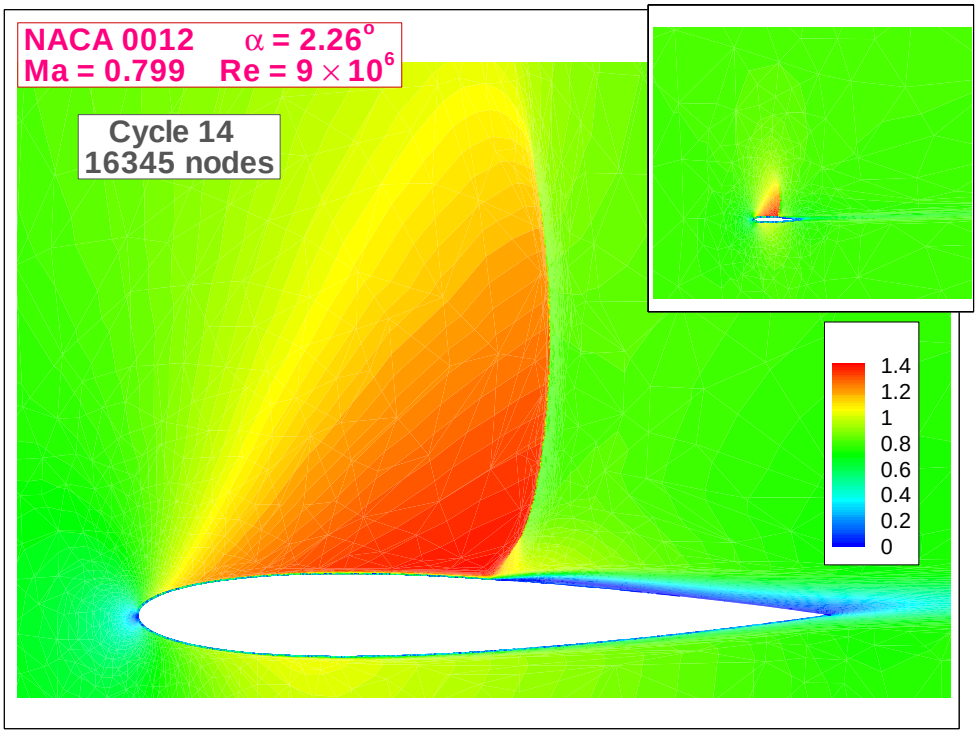



*Figure 8.7 Adaptation for Transonic Flow around NACA 0012 (Mach Number Contours)*

results do initiate the adaptation in this case, and it can be seen as the adaptation proceeds, the pressure coefficients shown in Fig. 8.8 approach the experimental data. The Mach number contours in Fig. 8.7 look excellent. Although the final result does not match the data very well, the primary reason is that wall effects are present in the wind tunnel, as has been verified by numerous CFD solutions. The significance of adaptation can be seen as adaptation is started from a trivial mesh.

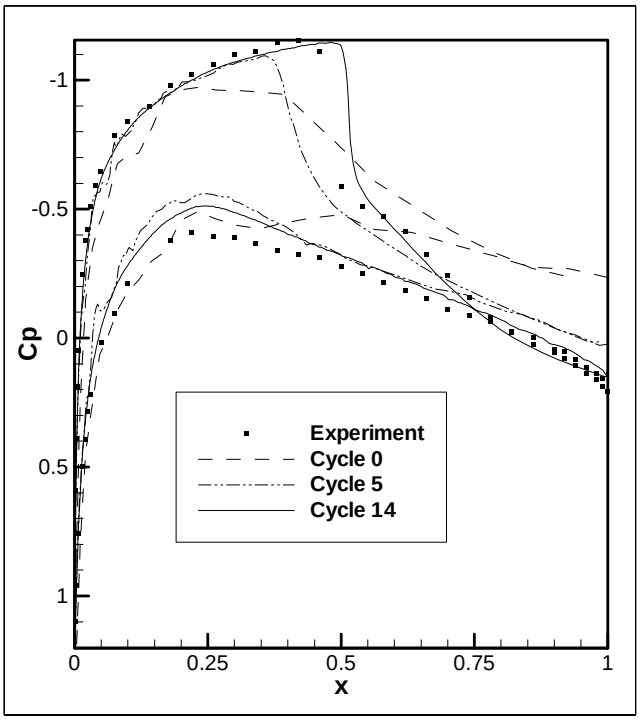



*Figure 8.8 Comparison between Results from Adaptation and Experimental Data*

## 8.4 Internal Flow Adaptation Results

In order to further verify the current adaptation approach, subsonic/supersonic flow through a nozzle is studied here. The computations of the internal flow are accomplished with the commercial CFD solver, FLUENT [94]. A data transfer facility is made possible by exporting the mesh in PATRAN format [62], which is widely used in commercial packages.

The geometry and boundary conditions of the supersonic nozzle are shown in Fig. 8.9. The total pressure at the inlet is set at 10 atmospheres, while the outlet back pressure is one atm. The total temperature is 500 K at the inlet and flow angle is zero. The ideal gas assumption is made for the air and two-dimensional rather than an axisymmetric case [63,64] is calculated for the demonstration of adaptation here. The two equation k-ω model of Wilcox [65] is used for turbulence.

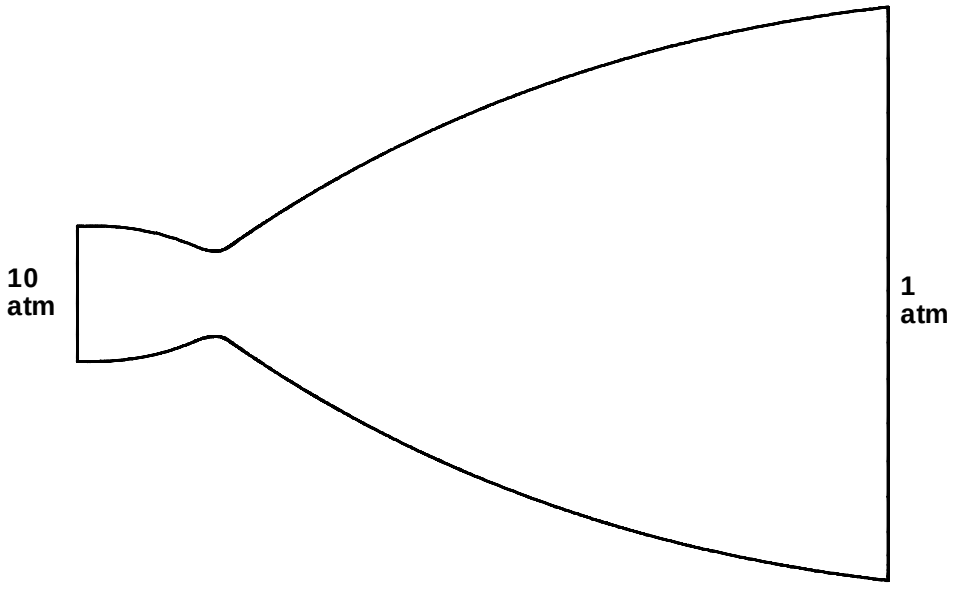



*Figure 8.9 Geometry and Boundary Conditions of the Convergent/Divergent Nozzle*

The initial mesh shown in Fig. 8.10 with 1500 grid points is generated with CFDRC-GEOM. The Mach contours from the calculation on this initial mesh are given in Fig. 8.11 and are quite diffusive near the wall, due to the large, isotropic cells there. The mesh after four cycles of adaptation is shown in Fig. 8.12. The number of points has now been increased to 20000, in order to give a more reasonable result. It can be seen from the mesh in Fig. 8.12 and Mach contours in Fig. 8.13 that the boundary layer is well resolved by the refined mesh. The pressure contours in the entire nozzle and in the convergent section are shown in Fig. 8.14. The absolute static pressure (static pressure plus the ambient reference pressure, which is one atm) of the exit flow is lower than the back pressure. An oblique shock appears in a small region near the exit line and can be seen in the pressure contours. The shock-induced separation is also captured by the adaptation. Due to the high Reynolds number in the flow, the Hessian far from the wall the nozzle surface is small compared with the second derivatives inside the boundary layer. The control of the maximum edge length, introduced in Chapter 7 in order to give a reasonable grid size in the uniform flow region, helps to improve the mesh in the core of the nozzle.

A zoomed view of the mesh inside the boundary layer is shown in Fig. 8.15. The region shown is just downstream of the throat of the nozzle, and the highly stretched cells with smooth transition from the boundary layer to free stream can be seen. The convergence criterion set by FLUENT, three orders of magnitude down from the initial condition, is met for the calculations on each adapted mesh of the nozzle. Overall, quality mesh and results have been acquired through the adaptation for this case.

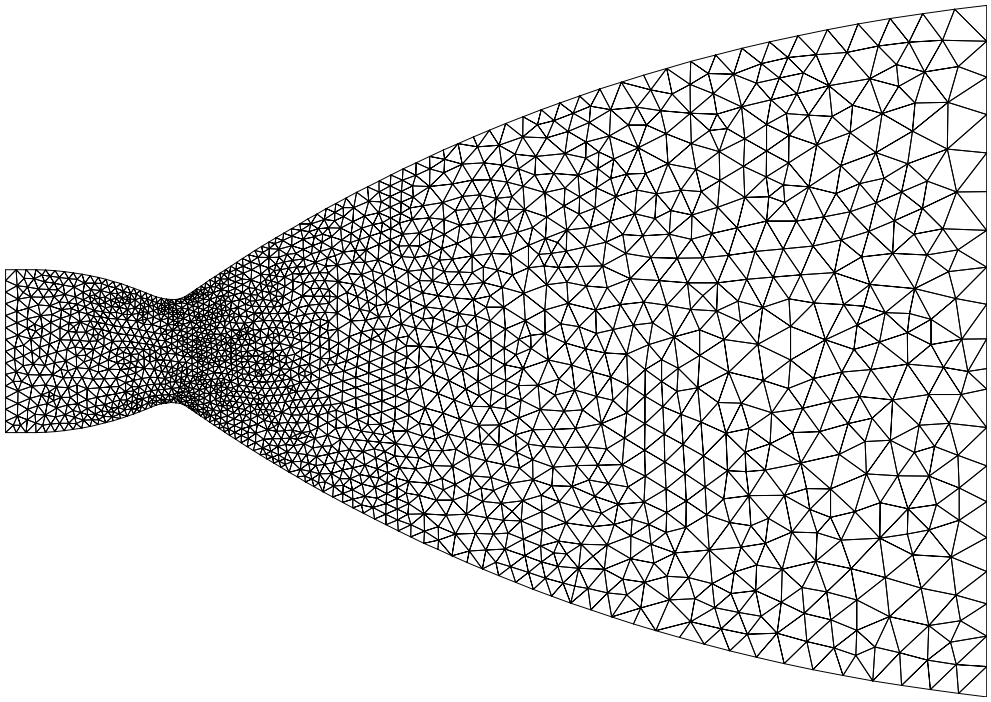


*Figure 8.10 Initial Mesh of the Convergent/Divergent Nozzle*

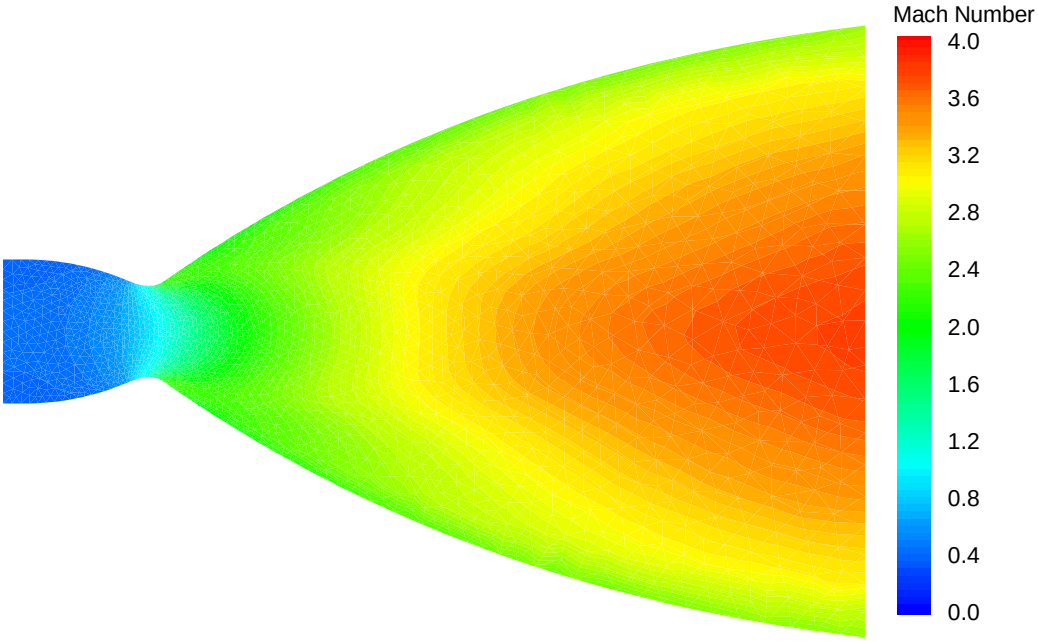



*Figure 8.11 Mach Contours from the Calculation on the Mesh in Fig. 8.10*

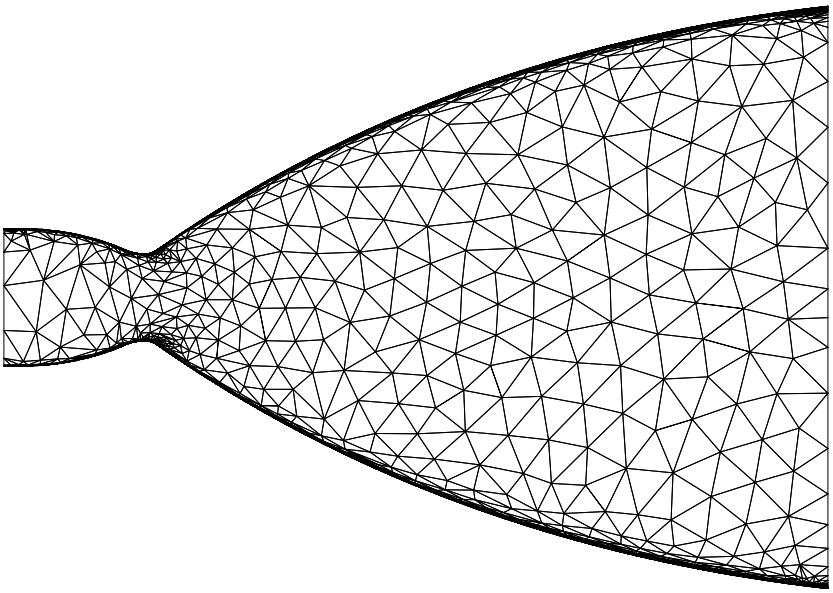


*Figure 8.12 Mesh of the Nozzle after 4th Iteration*

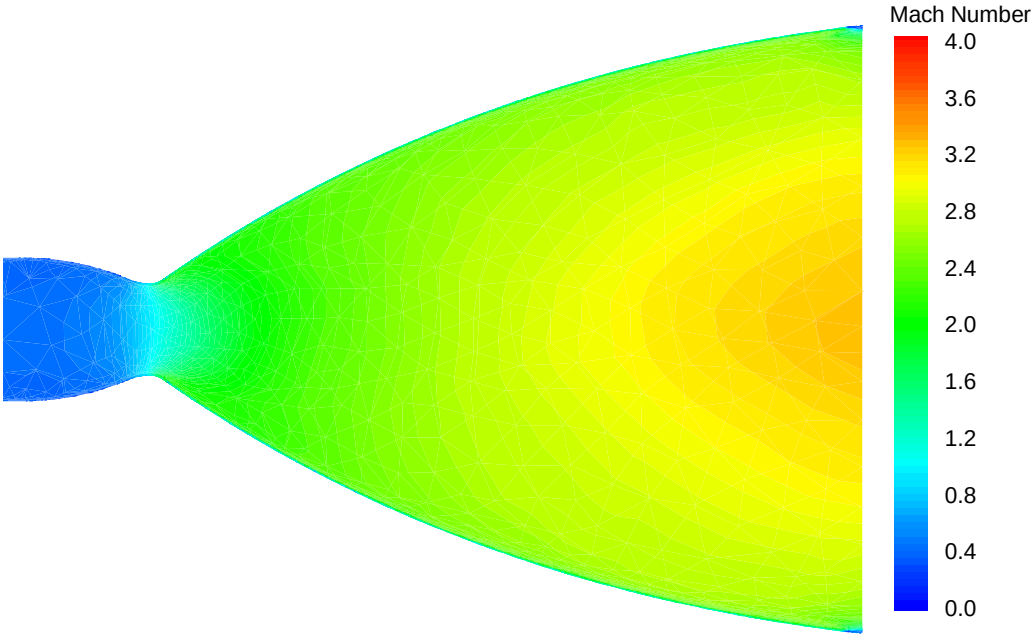



*Figure 8.13 Mach Contours from the Calculation on the Mesh in Fig. 8.12*

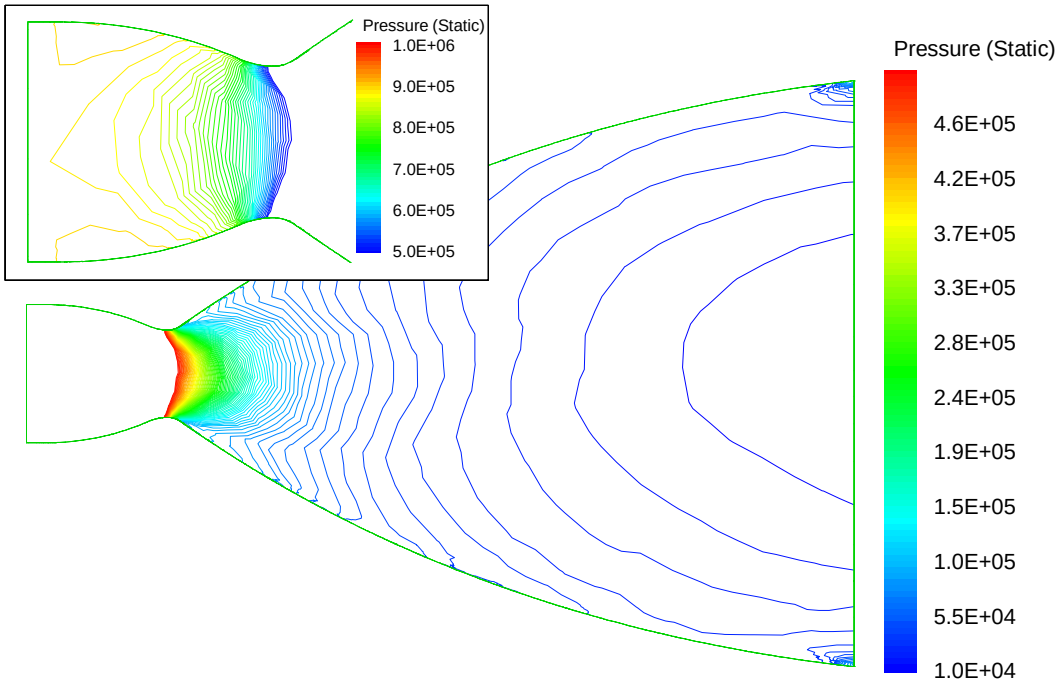



*Figure 8.14 Pressure Contours from the Calculation on the Mesh in Fig. 8.12*

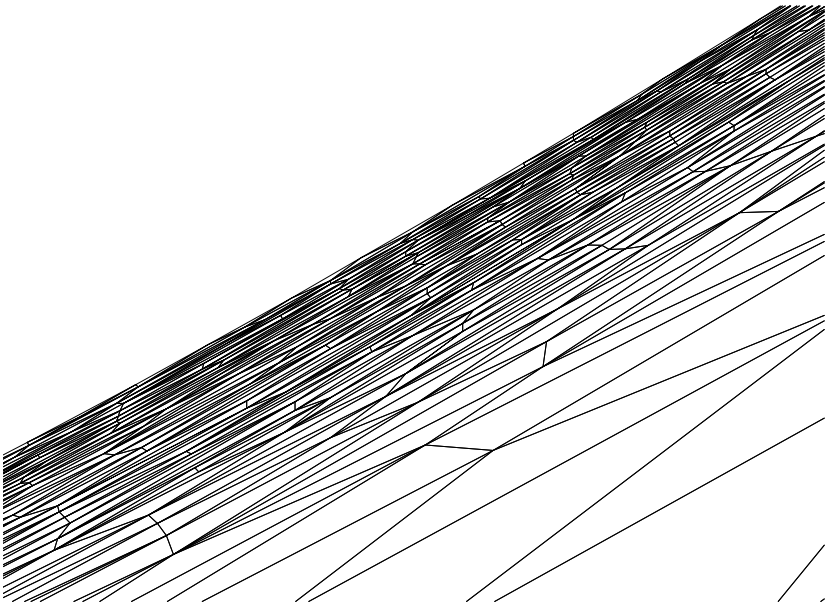


*Figure 8.15 Zoomed View of the Boundary Layer Mesh at the Throat*

# Chapter 9

# Preliminary Research on Three-Dimensional Anisotropic Adaptation

The previous chapters have addressed grid adaptation in two dimensions. The interest and effort for adaptation will clearly come in three dimensions where high quality grids are more difficult to generate. Three-dimensional adaptation has been the target of research efforts [84-90] for several years. In this chapter, the extension of the current approach to three-dimensional anisotropic grid adaptation is attempted. Some major differences in three dimensions are discussed. The barriers in anisotropic adaptation will also be pointed out in the preliminary study in this chapter.

## 9.1 Anisotropic Adaptation Approach in Three Dimensions

The problem formulation and adaptation function in three dimensions are the same as those in the two-dimensional study. Again the goal is to drive the edge lengths in a domain defined by some metric towards equality. Ideally the edges in the transformed

Riemann space will be equal-sized, while the corresponding mesh in the Euclidean space is the desired anisotropic solution.

As in the two-dimensional cases, procedures used to equalize the edge length include node enrichment, node removal, edge swapping and point smoothing. Point smoothing is exactly the same as that in two dimensions, as pointed out earlier. Node enrichment and removal are used to insert or delete nodes locally, based on the length of edges. Each edge whose Riemann length is longer than the allowable upper limit is divided into two new edges and a new node is introduced. Faces surrounding the edge as well as the tetrahedrons are divided into two. A representative example of node enrichment is shown in Fig. 9.1.

Wherever a short edge exists, the node removal procedure is invoked to remove one end of the edge. Edges meeting in this node will be removed, as well as tetrahedrons surrounding these edges. A vacuum is thus left. The method to construct the new connectivity inside this vacuum is similar to that used in two dimensions, which is the modified version of the advancing front method introduced in Chapter 1. An example in which the shortest edge is identified, removed and new cells are generated in the resulting vacuum is shown in Fig. 9.2.

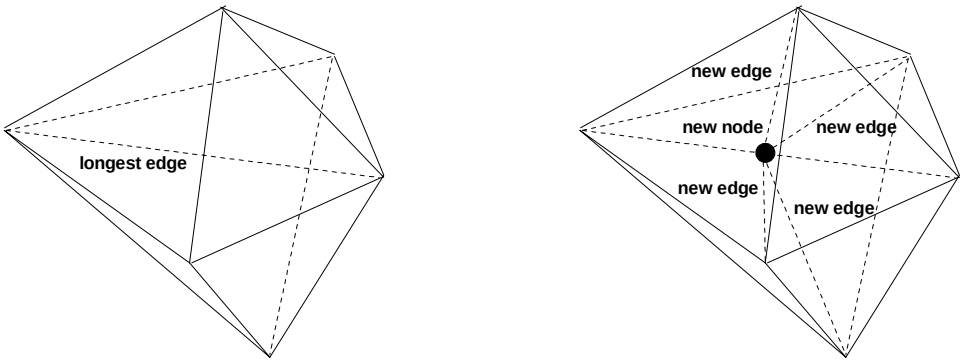



*Figure 9.1 Illustration of Node Enrichment in Three Dimensions*

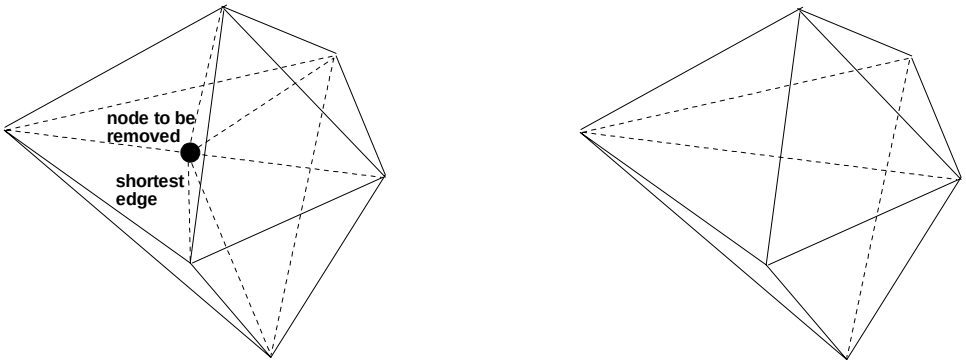



*Figure 9.2 Three-Dimensional Illustration of Node Removal*

The reconnecting method works well in two dimensions since a planar polygon can always be divided into triangles without introducing a new node, independent of any adjacent connectivity pattern. Difficulties, however, exist in three-dimensional applications. Not all polyhedrons in three dimensions can be converted to tetrahedrons. An example is the family of prisms shown in Figure 9.3. Each face of a prism can be split into triangles in two different manners, giving a total number of eight different possible tetrahedral configurations. Among these eight configurations, six can be subdivided into three tetrahedrons and two cannot. Similarly for general polyhedron with many possible encompassing connectivities, the tetrahedralization may not exist. If removal of the end of any short edge results in a non-tetrahedralizable connectivity, such as the prism shown in the right of Fig. 9.3, it indicates that this end of the edge cannot be removed and thus the node removal fails.

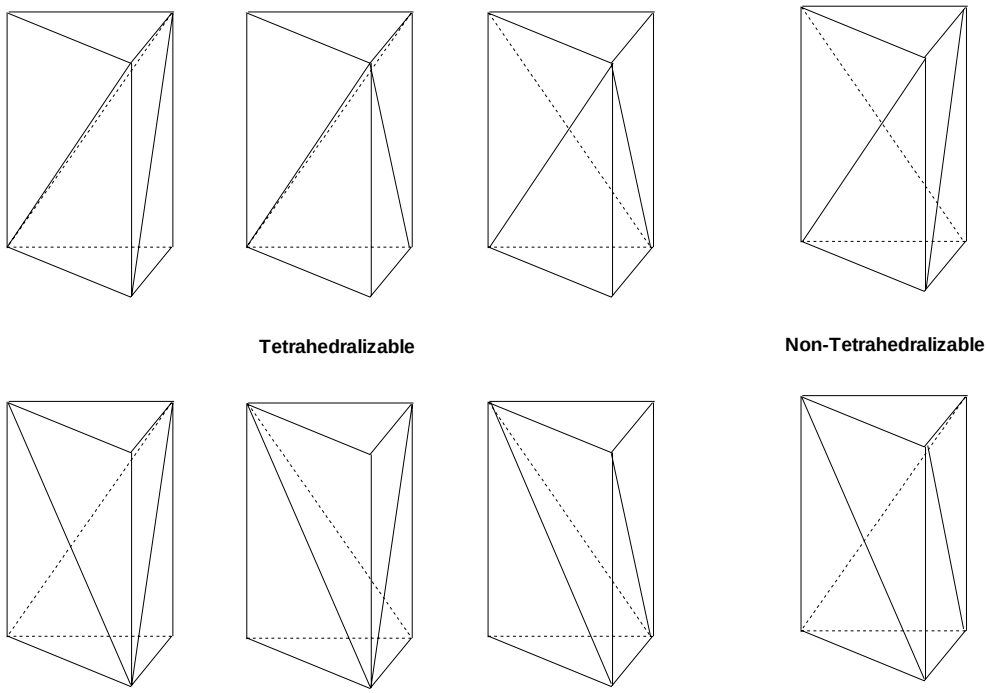



*Figure 9.3 The Tetrahedralizable and Non-Tetrahedralizable Manners of Subdividing a Prism*

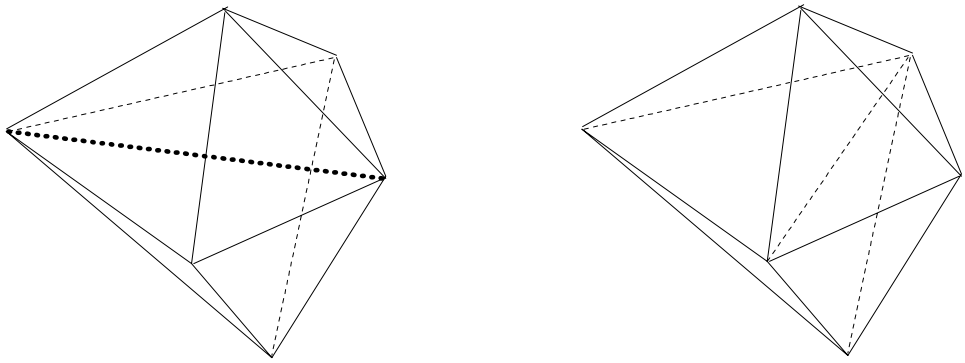


*Figure 9.4 Edge Swapping on Three-Dimensional Mesh*

The edge swapping operation in the current work is to remove every edge iteratively. The faces and cells around the edge are removed and a vacuum is thus created. A new connectivity, which is expected to be at least as good as the previous one, is generated so that local improvement is possible. This is done by the same reconnecting procedure as in the above node removal process. An example of edge swapping is shown in Fig. 9.4.

Because edges and faces are distinct elements in a three-dimensional mesh, edge swapping and face swapping likewise become distinct operations. Face swapping as demonstrated in Fig. 9.5 considers the two cells sharing a triangular face. When the common face is replaced by an edge, three new tetrahedrons are generated. If the overall quality of these three tetrahedrons is better than the previous two, face swapping improves the local quality. However, as can be easily seen that face swapping will only increase the number of edges and faces in a mesh with given number of nodes, which is obviously not reasonable for the optimization of mesh. Face swapping can only change the left configuration to the right one, while edge swapping can alter the connectivity in

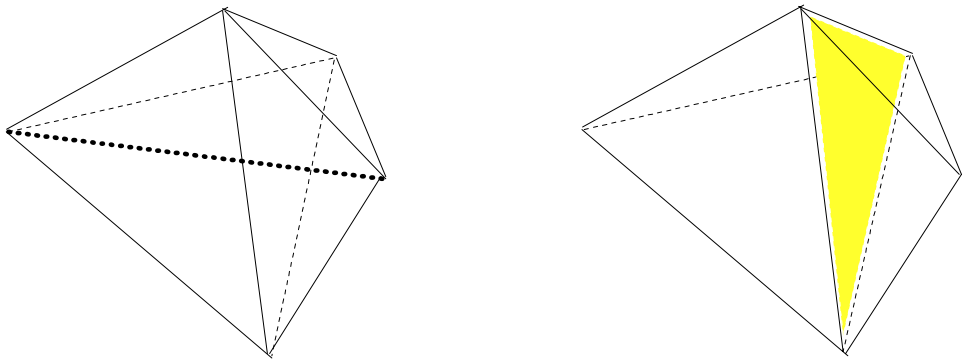


*Figure 9.5 Face Swapping on Three-Dimensional Mesh*

both ways. Thus, face swapping itself is not a useful approach for improving the global quality of the mesh and is not used in the current work.

The advancing front method in the node removal and edge swapping procedures requires special attention in three dimensions, since occasions that no valid connectivity can be found exist at any stage when the advancing front method is followed. Thus in the current approach for every face used to advance, all the possible connectivities should be stored and tried. If none of the possibilities are valid, any connectivity created previously should be removed, until returning to the original vacuum. For edge swapping, there is always a valid connection (the original one before the edge is removed). The node removal process, however, will frequently result in a non-tetrahedralizable vacuum as pointed out earlier.

## 9.2 Results of Three-Dimensional Adaptation

Following the procedure of the earlier work on two-dimensional mesh adaptation, a code for the anisotropic adaptation over three-dimensional unstructured meshes is developed. Only meshes with pure tetrahedrons are considered.

The sequence of procedures is similar to those in Chapter 3. Specific modifications of the Hessian for boundary layer and discontinuity are not yet included in the current work. An example here is to show the feasibility of the current approach. An uncoupled analytical (Blasius) solution of a two-dimensional boundary layer flow is taken to give the velocity profile so that Hessian metric can be evaluated. The flow is symmetric in the *z*-direction. A constant is added to the three eigenvalues in order that the edge length in the *z*-direction would not be zero, a situation similar to the uniform flow region in Chapter 7. Shown in the Figures 9.6 and 9.7, $x$ ranges from 1 to 4, corresponding to local Reynolds number $\mathrm{Re}_x$ of 1000 to 4000. The shear velocity, which is in the $y$ direction, is used to calculated Hessian. Colors in the mesh are shown according to the magnitudes of $h_{yy}$, which control the cell sizes in the *y*-direction. The values near the wall are smaller than the peaks above the wall due to the inflections on the wall, which we have explained in our work earlier. A coarse mesh appears near the wall since boundary layer enhancement is not yet implemented. The surface mesh is shown in Figure 9.6 and the inner grid is shown in Figure 9.7. Although visualization is difficult, especially for cells inside the domain, the mesh on the surface clearly shows that high aspect ratio can be achieved with the current approach.

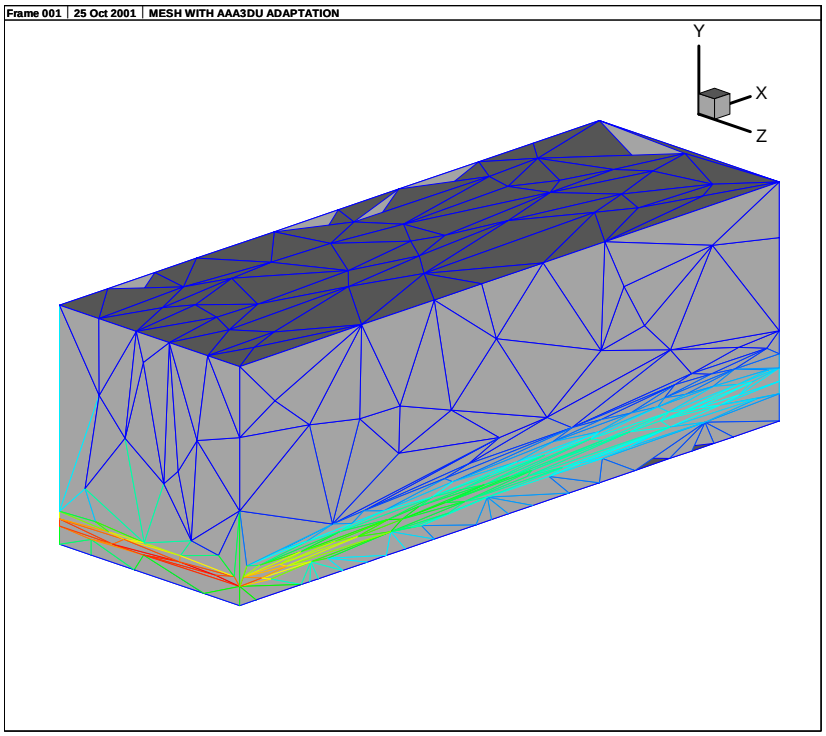



*Figure 9.6 Surface Grid from Three-Dimensional Anisotropic Adaptation*

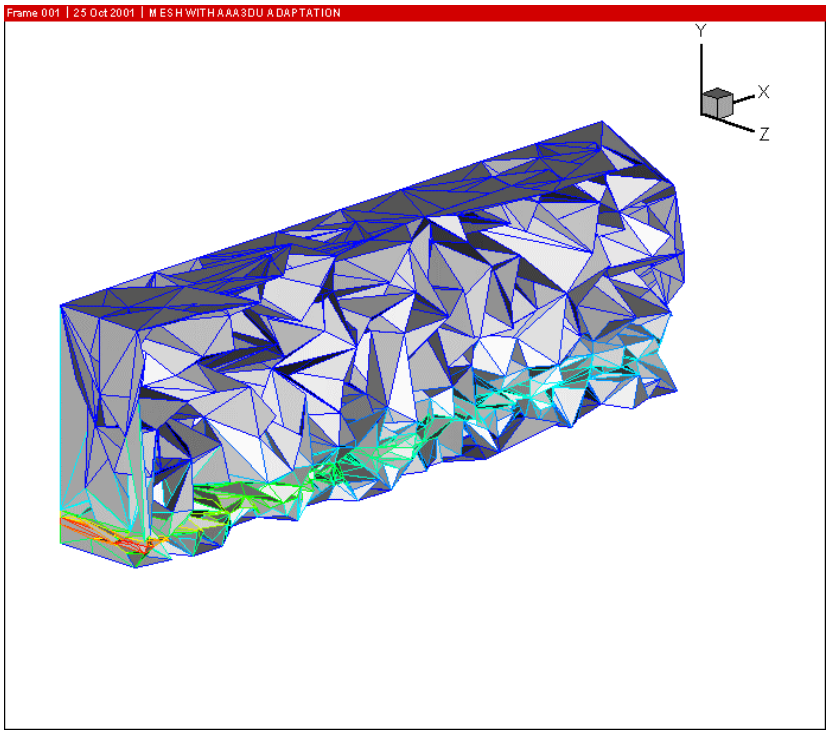



*Figure 9.7 Volume Grid from Three-Dimensional Anisotropic Adaptation*

## 9.3 Barriers in Three-Dimensional Adaptation

Although anisotropic elements are generated both on the surface and inside the volume, as can be seen in Figs. 9.6 and 9.7, difficulties can also be seen in these pictures. In Fig. 9.8 the side view of the surface grid shows that poor quality triangles are not improved by edge swapping. One of them is sitting in the lower left corner, right above the boundary layer. Points in the lower right border are not removed, although edges there are short. These difficulties are amplified by the relatively coarse grid used in the present example.

The poor quality grids are caused by the major difficulties in three-dimensional anisotropic adaptation. The inability of tetrahedralizing the vacuum left by the removal of either a point or an edge prevents equidistribution of the edges and improvement of the elements with iterative edge swapping. These difficulties, moreover, cannot be overcome since all possible connectivity patterns have been attempted.

The result in this chapter shows that significant difficulties exist when the current approach is extended to three-dimensional grids. Although there are many degrees of freedom when discrete points are to be connected, the situation occurs where no valid connectivity discretization exists for simple geometry, such as the non-tetrahedralizable prism mentioned earlier. This kind of difficulty, which does not exist in two dimentions, is a major barrier in three dimensions. One possible future approach is to utilize a larger number of degrees of freedom in unstructured grids. Anisotropic adaptation in three dimensions thus requires more intelligence and effort.

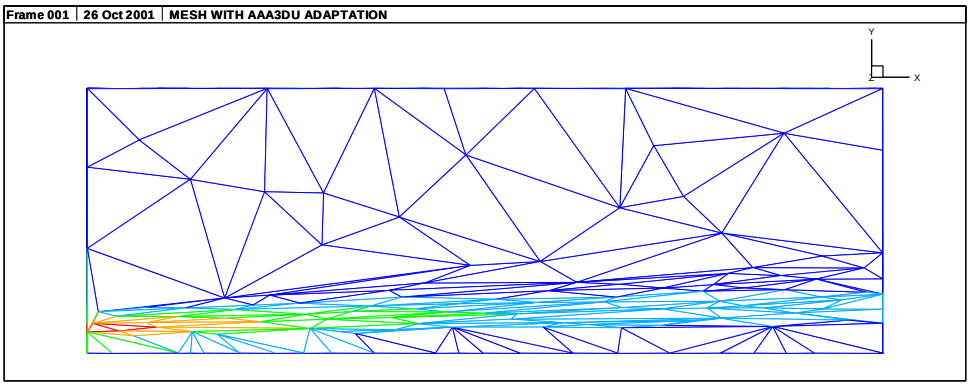



*Figure 9.8 Anisotropic Grid on the Surface*

# Chapter 10

# Conclusions

## 10.1 Summary

A solution adaptive method has been analyzed that enables the simulation of two-dimensional flow fields with reasonably stretched meshes. In order to model highly directional flow features more effectively, anisotropic meshes are chosen over an isotropic mesh. The anisotropic approach aligns the elements along the main flow features, which makes the adaptation more economical since the length scale required in one direction might be several orders of magnitude smaller than that in the other direction. Unstructured grids offer more degrees of freedom and are chosen for the adaptation.

Anisotropic adaptation requires the definition of cell orientations as well as cell size. Gradients, efficient for isotropic adaptation, cannot provide enough information for anisotropic adaptation, since the gradient only gives the direction of growth. In order to identify the large variations of the flow field, feature detection is made possible with the introduction of Hessian matrices. The Hessian matrices, composed of the second derivatives of one or more of the flow variables, are efficient for the definition of the cell

orientations and cell sizes. The eigenvectors of a matrix give the directions of the cells while the eigenvalues specify the sizes in these directions.

The manipulations on the mesh in order to equidistribute the edge lengths are detailed. Those include node insertion and removal. Edge swapping is used to improve the connectivity of grids and a point smoothing process further helps. An equation to re-locate a point surrounded by edges is developed for this purpose. The triangulation of the vacuum left by node removal and edge swapping is done through a unified local reconnection procedure which uses a modified version of the advancing front method. Data structures to facilitate the adaptation are described, while validation of the current approach is made with the grid adaptation of modeling of a supersonic flow around NACA 0012.

The feature detector in the form of a Hessian matrix does not always generate high quality grids for key flow features. Inflection points in boundary and shear layers cause difficulties in the adapted grids. These inflection points result in coarse grids in regions where fine elements are required, such as the boundary layer grid near the wall. The remediation is to introduce the boundary-layer-enhanced Hessian, which controls the grid size next to the wall. The definition of the wall Hessian in turbulence modeling follows the value of $y^+$ of the first point off the wall. The Hessian between the wall and the location where it reaches a maximum is obtained through interpolation. The control of the mesh size in a uniform flow region seen usually in a supersonic flow is augmented. Justification of the correction is made through the comparison of the results with those from a benchmark solution.

The difficulties in the adaptation of discontinuities, such as shocks, are investigated. Discontinuities tend to attract all the grid points in the field, packing most of the points in their vicinity. Although limiting the maximum value of the Hessian across the discontinuity prevents over-refinement, the resulting thin region covers only two or three layers of cells and is not enough to encompass the discontinuity. The deviation of the shock from the refined region can be prevented by enlarging the width of refinement by smoothing the Hessian. Issues in the discontinuity adaptation also include the unbalanced refinement at different regions of the discontinuity. The unbalanced refinement is shown to be detrimental to the adaptation and improvement is made through limiting the Hessian before it is smoothed.

Parametric adaptation results for flows around a NACA 0012 are obtained. Flows with different Mach numbers, going from subsonic to supersonic, are attempted, as well as transonic flows with attack angles. All adaptations are initiated with coarse, isotropic grids. The result from a detailed investigation of the transonic flow around NACA 0012 is compared with an experimental study. The overall quality of the adapted mesh is further validated through the calculation with FLUENT, a commercial CFD solver.

Preliminary work on three-dimensional anisotropic adaptation with the current length-based approach is conducted. The reason why edge swapping instead of face swapping is used is explained. Although essentially the same advancing-front method is used for tetrahedralizing the vacuum left by node/edge removal, difficulties are explained in this approach in the three-dimensional case. The major difficulty is the possible non-existence of connectivity inside an arbitrary vacuum, which prevents the equidistribution of edge length and improvement of the tetrahedral quality by iterative swapping. Though

high-aspect-ratio elements appear on the mesh, poor grid quality also accompanies them because of the difficulties of the current approach in three dimensions.

## 10.2 Concluding Remarks

The adaptation procedure developed in the current work facilitates Computational Fluid Dynamics simulations of two-dimensional flows with stretched adapted mesh. The anisotropic approach provides an economical solution for flows with highly directional features. The feature detector, developed and improved in the current approach, can successfully resolve major flow features, such as boundary layers and discontinuities. The unified reconnection procedure in adaptation enables the easy removal of short edges and simple implementation of the edge swapping procedure. Demonstrations show that the anisotropic adaptation resolves flows with a substantially reduced number of points.

Poor quality grids in three dimensions resulting from the extended version of the current methodology indicate the magnitudes of the problems to be overcome in three-dimensional anisotropic adaptation.

# REFERENCES

# References


1. Aftosmis, M., Berger, M. and Melton, J., Adaptation and Surface Modeling for Cartesian Mesh Methods, AIAA Paper 95-1725-CP, 1995.

2. Baker, T. and Cavello, P. Dynamic Grid Adaptation for Deforming Tetrahedral Meshes, AIAA Paper 99-3253-CP, 1999.

3. Dompierre, J., Labbé, P., Garon, A. and Camarero, R., Unstructured Tetrahedral Mesh Adaptation for Two-Dimensional Space-Time Finite Elements, AIAA 2000-0810, 2000.

4. George, P. L. and Hecht, F. Nonisotropic Grids, in Handbook of Grid Generation, Edited by Thompson, Soni and Weatherill, CRC Press, pp20-1, 20-30, 1999.

5. Habashi, W. G., Fortin, M., Dompierre, J., Vallet, M. –G., Ait-Ali-Yahia, D., Bourgault, Y., Robichaud, M. P., Tam, A., Boivin, S., Anisotropic Mesh Optimization for Structured and Unstructured Meshes, *Von Karman Institute Lecture Series,* 1997-02, 1997.

6. Khokhlov, A. M., Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics simulations, *J. Comp. Phys.*, 143, 519-543, 1998.

7. Neaves, M.D., McRae, D.S., Edwards, J. R., High-Speed Flow Calculations Using a Three-Dimensional Coupled Implicit Solution Adaptive Mesh Algorithm, AIAA Paper 99-0665, 1999.

8. Peraire, J., Peiró, J., Morgan, K., Advancing Front Grid Generation, in Handbook of Grid Generation, Edited by Thompson, Soni and Weatherill, CRC Press, pp17-1, 17-22, 1999.

9. Peraire, J., Vahdati, M., Morgan, K. and Zienkeiwicz, O.C., Adaptive Remeshing for Compressible Flow Computations, *J. Comp. Phys.*, **72**, 449-466, 1987.

10. Pirzadeh, S., An Adaptive Unstructured Grid Method by Grid Subdivision, Local Remeshing and Grid Movement, AIAA Paper 99-3255-CP, 1999.

11. W. Rachowicz, J.T. Oden and L. Demkowicz. Toward a universal *h-p* adaptive finite element strategy, Part3. Design of *h-p* meshes. *Comp. Meth. In Appl. Mech. and Engng*., 77:181-212,1989.

12. W. Gui and I. Babuška, The *h*, *p* and *h-p* version of the finite element method in one dimension. Part III: The adaptive *h-p* version. *Nümerische Methematik*, 48, 1986.

13. Frederic J. Blom, Considerations on the spring analogy. *Int. J. Numer. Meth. Fluids*, 32: 647-668, 2000.

14. D.F. Hawken, J.J. Gottlieb and J.S. Hansen. Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations. *J. Comp. Phys*., 95(2):254-302, August 1991.

15. R. Löhner, Finite-element methods in CFD: Grid generation, adaptivity and parallelization. In *von Karmon Institute Lecture Series*, AGARD Pub. R-787, 1992.

16. M.A. Yerry and M.S. Shephard., Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, vol. 20, pp. 1965-1990, 1984.

17. S. Ward and Y. Kallinderies, Hybrid prismatic/tetrahedral grid generation for complex 3-D geometries, AIAA Paper 93-0669, 1993.

18. Cavendish, J.C., Field, D.A., and Frey, W.H., An approach to automatic three-dimensional finite element mesh generation, *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 329-347, 1985.

19. Baker, T.J., Three dimensional mesh generation by triangulation of arbitrary point sets, AIAA Paper 87-1124, 1987.

20. Weatherill, N.P., A method for generating irregular computational grids in multiply connected planar domains, *Internal Journal for Numerical Methods in Fluids*, vol. 8, pp. 181-197, 1988

21. Holmes, D.G. and Synder, D.D., The generation of unstructured triangular meshes using Delaunay triangulation, In *Numerical Grid Generation in Computational Fluid Mechanics '88* (S. Sengupta *et al*., eds.), pp. 643-652, Pineridge Press Limited, 1988

22. Baker, T.J., Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation, *Engineering with Computers*, vol. 5, pp. 161-175, 1989

23. Mavriplis, D.J., Adaptive mesh generation for viscous flows using Delaunay triangulation, *Journal of Computational Physics*, vol. 90, pp. 271-291, October 1990.

24. Mavriplis, D.J., Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes, *AIAA J.*, 28(2): 213-221, 1990.

25. Lo, S.H., A new mesh generation scheme for arbitrary planar domains, *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1403-1426, 1985

26. Löhner, R., and Parikh, P., Generation of three-dimensional unstructured grids by the advancing-front method, AIAA Paper 88-0515, 1988

27. Jin, H., and Wiberg, N.-E., Two-dimensional mesh generation, adaptive remeshing and refinement, *Internal Journal for Numerical Methods in Engineering*, vol. 29, pp. 1501-1526, 1990.

28. Lilek, Ž. and Perić, M., A fourth–order Finite Volume method with collocated variable arrangement, *Computers and Fluids*, 24(3):239-252, 1995.

29. Berger, M.J. and Oliger, J., Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comp. Physics*, 53:484-512, 1984.

30. Berger, M.J. and Collela, P., Local adaptive mesh refinement for shock hydrodynamics, *J. Comp. Physics*, 82:64-84, 1989.

31. Dwyer, H.A., Grid adaptation for problems in fluid dynamics, *AIAA Journal*, 22(12): 1705-1712, December, 1984.

32. Berger, M.J. and Jameson, A., Automatic adaptive grid refinement for the Euler equations, *AIAA Journal*, 23(4): 561-568, 1985.

33. Löhner, R., An adaptive Finite Element scheme for transient problems in CFD, *Comp. Meth. Appl. Mech. Engineering*, 61:323-338, 1987.

34. Ramakrishnan, R., Bey, K.S., and Thornton, E.A., Adaptive quadrilateral and triangular Finite Element scheme for compressible flows, *AIAA Journal* 28(1): 51-59, January 1990.

35. Speares, W. and Toro, E.F., A high resolution algorithm for finite dependent shock dominated problems with adaptive mesh refinement, *Z. Flugwiss. Weltraumforsch.*, 19:267-281, 1995.

36. Jasak, H.: Error analysis and estimation in the Finite Volume method with applications to fluid flows, PhD thesis, Imperial College, University of London, 1996.

37. Muzaferija, S. and Gosman, D., Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology, *J. Comp. Physics*, 138(2):766-787, 1997.

38. Haworth, D.C., El Tahry, S.H., and Huebler, M.S., A global approach to error estimation and physical diagnostics in multidimensional fluid dynamics, *Int. J. Numer. Meth. Fluids*, 17(1): 75-97, 1993.

39. Anderson, W. K., Bonhaus, D. L., An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids, *Computers Fluids* **23**, 1-21 (1994).

40. Mavriplis D.J. and Venkatakrishnan V., A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes, ICASE Report, No. 95-30.

41. Barth, T.J., Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes, AIAA Paper, 91-0721, 1991.

42. Vallet, M.-G., Génération de maillages élément finis anisotropies et adaptatifs, PhD thesis, Université Pierre et Marie Curie, Paris VI, France, 1992.

43. Horowitz, E., Sahni, S., Mehta, D., Fundamentals of Data Structures in C++, Computer Science Press, 1995.

44. Slikkeveer P.J., van Loohuizen E.P., O'Brien S.B.G., An implicit surface tension algorithm for Picard solvers of surface-tension-dominated free and moving boundary problems, *International Journal for Numerical Methods in Fluids*, 22: 851-865, 1996.

45. Blom F.J., Leyland P., Analysis of fluid-structure interaction on moving airfoils by means of an improved ALE method. AIAA Paper 97-1770, 1997.

46. Farhat C., Lesoinne M., Maman N., Mixed explicit/implicit time integration of coupled aeroelastic problems: three field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21: 807-835, 1995

47. Piperno S., Explicit/implicit fluid/structure staggered procedures with a structural prediction and fluid subcycling for 2D inviscid aeroelastic simulations. *International Journal for Numerical Methods in Fluids*, 25: 1207-1226, 1997.

48. Richter R. and Leyland P., Entropy correcting schemes and non-hierarchical auto-adaptive dynamics finite element type meshes: applications to unsteady aerodynamics, *International Journal for Numerical Methods in Fluids*, 20(6): 853-868, 1995.

49. Weatherhill, N.P., Gaither K.P., and Gaither J.A., Building unstructured grids for computational fluid dynamics, *Computational Fluid Dynamics Journal*, 4(1): 1-28, 1995.

50. Blom F.J., Considerations on the spring analogy, *International Journal for Numerical Methods in Fluids*, 32: 647-668, 2000.

51. Preiss B.R., Data Structures and Algorithms: with Objected-oriented Design Patterns in C++, John Wiley & Sons, Inc., 1999.

52. Adelson-Velskii G.M. and Landis E.M., *Dokl. Acad. Nauk.*, SSR (Soviet Math), 3:1259-1263, 1962.

53. Anderson, W.K., Rausch R.D. and Bonhaus D.L., Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids, *AIAA Paper 95-1740*, 1995.

54. Baldwin, B.S., and Barth, T.J., A One-Equation Turbulence Transport Model for High Reynolds Number Wall Bounded Flows, *NASA Technical Memorandum 102847*, Aug. 1991.

55. Barth, T., and Jespersen, D., The Design and Application of Upwind Schemes on Unstructured Meshes, *AIAA 89-0366*, 1989.

56. Ashford, G.A., An Unstructured Grid Generation and Adaptive Solution Technique for High-Reynolds-Number Compressible Flows, Ph. D Thesis, The University of Michigan, 1996.

57. Deister, F., Hirschel, E.H., Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometries, *AIAA Paper 99-0782*, 1999.

58. Löhner, R., Sharov, D., Luo, H., and Ramamurti, R., Overlapping Unstructured Grids, *AIAA Paper 01-0439*, 2001.

59. Deister, F., and Hirschel, E.H., Self-Organizing Hybrid Cartesian Grid/Solution System with Multigrid, *AIAA Paper 02-0112*, 2002.

60. Murayama, M., Nakahashi, K., and Matsushima, K., Unstructured Dynamic Mesh for Large Movement and Deformation, *AIAA Paper 02-0122*, 2002.

61. Harris, C. D., Two-Dimensional Aerodynamics Characteristics of the NACA 0012 Airfoil in the Langley 8-Foot Transonic Pressure Tunnel, *NASA TM 81927*, 1987.

62. CFD-GEOM Tutorials, Version 5, CFD Research Corporation, October 1998.

63. Hunter, C.A., Experimental, Theoretical, and Computational Investigation of Separated Nozzle Flows, *AIAA Paper 98-3107*, 1998.

64. Graß, A., and Weiland, C., Investigation of Shock Patterns and Separation Behavior of Several Subscale Nozzles, *AIAA Paper 00-3293*, 2000.

65. Wilcox, D., Reassessment of the Scale-Determining Equation for Advanced Turbulence Models, *AIAA Journal*, Vol. 26, No. 11, pp. 1299-1310, 1988.

66. Wood, W.A., and Kleb, W.L., On Multi-Dimensional Unstructured Mesh Adaption *AIAA Paper 99-3254*, 1999.

67. Shahyar Z. Pirzadeh, S.Z., An Adaptive Unstructured Grid Method By Grid Subdivision, Local Remeshing, and Grid Movement, *AIAA Paper 99-3255*, 1999.

68. Zhang, X.D., Pelletier, D., Trepanier, J.-Y. and Camarero, R., Error Control and Mesh Adaptation for the Euler Equations, *AIAA Paper 01-0441*, 2001.

69. Xia, G., Li, D., and Merkle, C.L., Grid Adaptation on Unstructured Meshes, *AIAA Paper 01-0443*, 2001.

70. Groth, C.P.T., De Zeeuw, D. L., Powell, K. G., Gombosi, T. I., and Stout, Q. F., A Parallel Solution-Adaptive Scheme for Ideal Magnetohydrodynamics, AIAA Paper 99-3273, 1999.

71. Venditti, D.A., and Darmofal, D.L., A Multilevel Error Estimation and Grid Adaptive Strategy for Improving the Accuracy of Integral Outputs, *AIAA Paper 99-3292*, 1999.

72. Kang, H.J., and Kwon, O.J., Numerical Prediction of Rotor Hover Performances Using Unstructured Adaptive Meshes, *AIAA Paper 00-0258*, 2000.

73. Turkgeldi, I.E., and Kandil, O.A., Development and Application of Adaptive Dynamic Grid for Wake-Vortex/Ground Interaction AIAA, *Paper 00-0789*, 2000.

74. Murayama, M., Nakahashi, K., and Sawada, K., Numerical Simulation of Vortex Breakdown Using Adaptive Grid Refinement with Vortex-Center Identification, AIAA Paper 00-0806, 2000.

75. Carpenter, J.G.V., and McRae, D.S., Progress toward a Conservative Time Accurate Unstructured Adaption Algorithm, *AIAA Paper 00-0811*, 2000.

76. Venditti, D.A., and Darmofal, D.L., Grid Adaptation for Functional Outputs of 2-D Compressible Flow Simulations, *AIAA Paper 00-2244*, 2000.

77. Nae, C., Flow Solver and Anisotropic Mesh Adaptation Using a Change of Metric Based on Flow Variables, *AIAA Paper 00-2250*, 2000.

78. Ahuja, V., Cavallo, P.A., and Hosangadi, A., Multi-Phase Flow Modeling on Adaptive Unstructured Meshes, *AIAA Paper 00-2662*, 2000.

79. Cavallo, P.A., and Dash, S.M., Aerodynamics of Multi-Body Separation Using Adaptive Unstructured Grids, *AIAA Paper 00-4407*, 2000.

80. Liou, B.H., and Balsara, D., An Implicit Unstructured Adaptive-Grid Approach for Compressible Flows with Moving Boundaries, AIAA Paper 01-0440, 2001.

81. Neaves, M.D., McRae, D.S., and Edwards, J.R., High-Speed Inlet Unstart Calculations Using an Implicit Solution Adaptive Mesh Algorithm, *AIAA Paper 01-0825*, 2001.

82. Park, Y.M., and Kwon, O.J., Unsteady Flow Computations Using a 3-D Parallel Unstructured Dynamic Mesh Adaptation Algorithm, *AIAA Paper 01-0865*, 2001.

83. Soni, B., Thompson, D., Koomullil, R., and Thornburg H., GGTK - A Tool Kit for Static and Dynamic Geometry-Grid Generation and Adaptation, *AIAA Paper 01-1164*, 2001.

84. Baker, T.J., and Cavallo, P.A., Dynamic Adaptation for Deforming Tetrahedral Meshes, *AIAA Paper 99-3253*, 1999.

85. Rocher, D., Deister, F., Monnoyer, F., and Hirschel E. H., Flow Simulation on Adaptive Cartesian and Hybrid Prismatic-Cartesian Grids around Arbitrary Geometries, *AIAA Paper 99-3313*, 1999.

86. Lacor, C., Hirsch, C., Leonard, B., and Lessani, B., A 3D Navier-Stokes Solver Using Unstructured, Hexahedral Meshes with Adaptation, *AIAA Paper 99-3364*, 1999.

87. Aftosmis, M. J., Berger, M. J., and Adomavicius, G., A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries, *AIAA Paper 00-0808*, 2000.

88. Lahur, P.R., and Nakamura, Y., Anisotropic Cartesian Grid Adaptation, *AIAA Paper 00-2243*, 2000.

89. Cavallo, P.A., and Baker, T.J., Efficient Delaunay-Based Solution Adaptation for Three-Dimensional Unstructured Meshes, *AIAA Paper 00-0809*, 2000.

90. Robichaud, M., Ait-Ali-Yahia, D., Peeters, M., Baruzzi, G., Kozel, V., and Habashi, W. G., 3-D Anisotropic Adaptation for Eexternal and Turbomachinery Flows on Hybrid Unstructured Grids, *AIAA Paper 00-2248*, 2000.

91. Löhner R. and Baum J.D., Three-Dimensional Store Separation Using a Finite Element Solver and Adaptive Remeshing, *AIAA Paper 91-0602*, 1991.

92. see web page www.cfdrc.com.

93. Schlicting, H., *Boundary Layer Theory*, 7th Edition, McGraw-Hill, 1979.

94. see web page www.fluent.com.

95. Li, D., Venkateswaran, S., Fakhari, K., and Merkle, C.L., Convergence Assessment of General Fluid Equations on Unstructured Hybrid Grids, *AIAA Paper 01-2557,* 2001.

96. Merkle, C.L. and Yu S.–T., *Computational Fluid Dynamics of Inviscid and High Reynolds Number Flows*. (to be published)

97. Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2, *Computional Methods for Inviscid and Viscous Flows*, John Wiley and Sons, 1991.

98. Venkateswaran, S., and Merkle, C.L., Analysis of Preconditioning Methods for the Euler and Navier-Stokes Equations, *Von Karman Institute Lecture Series*, March 8 – March 12, 1999.

99. Van Leer, B., Towards the ultimate conservative difference scheme. V. A second order sequel to Godunov's method. *Journal Computational Physics*, 32, 101-36, 1979.

100. Roe, P.L., Some Contributions to the modelling of discontinuous flows. *Proc. 1983 AMS-SIAM Summer Seminar on Large Scale Computing in Fliud Mechanics, Lectures in Applied Mathematics,* Vol. 22, pp. 169-93. SIAN, Philadelphia, 1985.

101. Roe, P.L. and Baines, M.J., Algorithms for advection and shock problems. *Proc. 4th GAMM Conference on Numerical Methods in Fluid Mechanics*, Braunschweig, Vieweg, 1982.

102. Mavriplis D.J., Unstructured Mesh Generation and Adaptivity, ICASE Report, No. 95-26, 1995.

103. Chew, L.P., Constrained Delaunay Triangulation. *Algorithmica*, 4:97-108, 1989.

104. Preparata, F.P. and Shamos, M.I., *Computational Geometry, An Introduction*. Texts and Monographs in Computer Science, Springer-Verlag, 1985.

105. Celik, I., Yavuz, I., Smirnov, A., Large Eddy Simulations of In-Cylinder Turbulence for IC-Engines: A Review, *Int. Journal of Engine Research*, Vol. 2, No. 2, August, 2001.

106. Venkatakrishnan, V., On the Accuracy of Limiters and Convergence to Steady State Solutions, *AIAA Paper 93-0880*, 1993.

# APPENDICES

# Appendix A

# 2D Edge-based Hessian Formulas

We begin by approximating the following matrix of second derivatives

$$\Delta^2 f = \begin{bmatrix} f_{xx} & f_{yx} \\ f_{xy} & f_{yy} \end{bmatrix} \tag{A.1}$$

using a standard Galerkin approximation for the region $\Omega_0$ formed from all triangles that share the vertex $P$. Multiplying by the weight function $\phi$ and integration by parts over $\Omega_0$ assuming $\phi = 0$ on the boundary $\partial\Omega_0$ produces the so-called weak form,

$$\int_{\Omega_0} \phi \nabla(\nabla f)^T \, da = -\int_{\Omega_0} (\nabla \phi)(\nabla f)^T \, da = -\sum_{i=1}^{N} \int_{T_{i+1/2}} (\nabla \phi)(\nabla f)^T \, da \tag{A.2}$$

where $T_{i+1/2}$ is the triangle formed by $P_i$, $P_{i+1}$ and $P_0$.

Using the notation of figure A.1, gradients of the piecewise linear functions $\phi^h$ and $u^h$ are

$$\left(\nabla \phi^h\right)_{T_{i+1/2}} = -\frac{1}{2A_{i+1/2}} \vec{n}_{i+1/2} \tag{A.3}$$

and

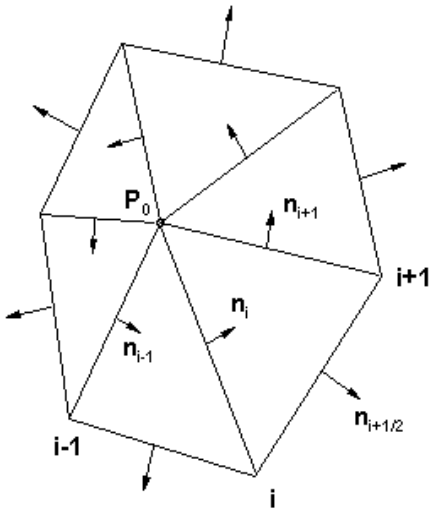



*Figure A.1 Vertex $P_0$ and Adjacent Neighbors.*

$$\left(\nabla f^{h}\right)_{T_{i+1/2}} = -\frac{1}{2A_{i+1/2}}\left(f_0^{h}\vec{n}_{i+1/2} + f_i^{h}\vec{n}_{i+1} - f_{i+1}^{h}\vec{n}_i\right) \tag{A.4}$$

where $A_{i+1/2}$ is the area of $T_{i+1/2}$ and $\vec{n}_{i+1/2}$ is the vector normal to the edge $e(P_i, P_{i+1})$ with magnitude equal to the length of the edge.

For piecewise linear $u^h$ the gradient is constant in each triangle. The integral average matrix of second derivatives simplifies to the following form:

$$\begin{aligned}\int_{\Omega_0} \phi^h \nabla(\nabla f)^T \, da & \\ &= \sum_{i=1}^{N} \frac{1}{2A_{i+1/2}} \vec{n}_{i+1/2} \int_{T_{i+1/2}} \left(\nabla f^{h}\right)^T da \\ &= \sum_{i=1}^{N} \frac{1}{2A_{i+1/2}} \vec{n}_{i+1/2} \left(\nabla f^{h}\right)^T_{T_{i+1/2}} \int_{T_{i+1/2}} da\end{aligned}$$

$$= \sum_{i=1}^{N} \frac{1}{2} \vec{n}_{i+1/2} \left( \nabla f^{h} \right)_{T_{i+1/2}}^{T} \tag{A.5}$$

Inserting the triangle gradient formula we obtain a discretized formula for the Galerkin integral.

$$\int_{\Omega_0} \phi^{h} \nabla \left( \nabla f^{h} \right)^{T} da = -\frac{1}{4} \sum_{i=1}^{N} \frac{1}{A_{i+1/2}} \vec{n}_{i+1/2} \left( f_0^{h} \vec{n}_{i+1/2}^{T} + f_i^{h} \vec{n}_{i+1}^{T} - f_{i+1}^{h} \vec{n}_i^{T} \right) \tag{A.6}$$

Regrouping of terms and removal of a constant solution yields the following simplified form:

$$\int_{\Omega_0} \phi^{h} \nabla \left( \nabla f^{h} \right)^{T} da = \int_{\Omega_0} \nabla \left( \nabla \left( f^{h} - f_0^{h} \right) \right)^{T} da = \sum_{i=1}^{N} M_i \left( f_i^{h} - f_0^{h} \right) \tag{A.7}$$

with

$$M_i = -\frac{1}{4} \left[ \frac{1}{A_{i+1/2}} \vec{n}_{i+1/2} \left( \vec{n}_{i+1} \right)^{T} - \frac{1}{A_{i-1/2}} \vec{n}_{i-1/2} \left( \vec{n}_{i-1} \right)^{T} \right] \tag{A.8}$$

# Appendix B

# Stability Analysis and Convergence Enhancement of High Order Upwind Difference Schemes

Performance of several high order difference schemes is studied with von Neumann (or Fourier) stability analysis. The inconsistent first-order/second-order upwind scheme system is examined carefully. We propose a new methodology to design the left hand side (LHS) so that the potential of the inconsistent difference system can be fully utilized. The generally used second order upwind scheme in unstructured mesh calculation is carefully studied and its convergence property is presented.

## B.1 INTRODUCTION

Upwind difference schemes are widely used in CFD applications due to their fast damping rate [96,97]. First order upwind system introduces a large amount of dissipation and is considered not sufficient in many cases. People seek to more accurate alternatives,

such as second and third order upwind schemes. Both of these two schemes introduce high order dissipation, which, in turn, use the second neighborhood nodes in the difference formulas. However, experiences have shown that the consistent high order upwind systems (with the same form in the LHS and right hand side (RHS)) are not stable for many algorithms, such as Point-Jacobi iteration. It is easy to find that both second order and third order upwind schemes result in a non-diagonally dominating matrix, which is found to be unstable in some modes with Fourier analysis. Since the LHS is a transient term and does not affect the steady state solution, people find that second order scheme can be stable by replacing the LHS difference by first order upwind scheme [98]. The inconsistent difference scheme, although not accurate for unsteady computation, resolves the main difficulty related to high order upwind schemes. The present authors investigate the reason of the success of inconsistent schemes and the potential of them with von Neumann analysis. We propose a new method to design the LHS by setting the damping rate at given modes, which is shown to be viable in the designing of the new LHS for the second order upwind scheme.

The second order upwind scheme in the unstructured mesh calculation is implemented by the linear reconstruction of the solution variables at the interfaces. This method for the generation of second order upwind schemes via variable extrapolation is often referred to in the literature as the MUSCL approach. In our work, the prototype of this scheme is examined in its one-dimensional version. This biased second order upwind scheme differs from standard second order upwind scheme, which is obtained through one-sided extrapolation. The optimal LHS corresponding to this scheme is found with the methodology, which, occasionally, is the one used in all the calculations.

## B.2 Analysis of the Point-Jacobi Algorithm

We start from the one-dimensional Euler equation

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} = \mathbf{H} \tag{B.2.1}$$

First-order upwind scheme results in following difference form

$$\left(\mathbf{I} - \mathbf{D}\Delta t + \frac{\Delta t}{\Delta x}|\mathbf{A}|\right)\Delta \mathbf{Q}_i + \frac{\Delta t}{\Delta x}\mathbf{A}^{-}\Delta \mathbf{Q}_{i+1} - \frac{\Delta t}{\Delta x}\mathbf{A}^{+}\Delta \mathbf{Q}_{i-1} = -\Delta t\left(\frac{\partial \mathbf{E}}{\partial x} - \mathbf{H}\right)^{n} \tag{B.2.2}$$

When the source term $\mathbf{H}$ does not present, von Neumann stability analysis shows that the amplification factor of the system is

$$\mathbf{G}_t = \frac{\mathbf{I}}{\mathbf{I} + \frac{\Delta t}{\Delta x}\mathbf{A}^{+}(1 - \mathbf{C} + i\mathbf{S}) - \frac{\Delta t}{\Delta x}\mathbf{A}^{-}(1 - \mathbf{C} - i\mathbf{S})} \tag{B.2.3}$$

where $\mathbf{C} = \cos(\omega)$ , $\mathbf{S} = \sin(\omega)$, $\omega$ is wavenumber.

For simplicity, we define the following variables before introducing the subiteration

$$\mathbf{U} \equiv \Delta \mathbf{Q} \tag{B.2.4}$$

$$\mathbf{L}_t = \mathbf{I} - \mathbf{D}\Delta t + \frac{\Delta t}{\Delta x}|\mathbf{A}| \tag{B.2.5}$$

$$\mathbf{R}^n = \Delta t\left(\frac{\partial \mathbf{E}}{\partial x} - \mathbf{H}\right)^{n} \tag{B.2.6}$$

With Block-Jacobi iteration used to solve the linear equations, we have

$$\mathbf{L}_t \mathbf{U}_i^{k+1} = -\mathbf{R}^n + \frac{\Delta t}{\Delta x}\mathbf{A}^+ \mathbf{U}_{i-1}^k - \frac{\Delta t}{\Delta x}\mathbf{A}^- \mathbf{U}_{i+1}^k \tag{B.2.7}$$

For the inner iteration, $\mathbf{R}^n$ is considered as a constant. Again von Neumann stability analysis can be used to find the amplification factor of the iteration

$$\mathbf{G}_i = \frac{\frac{\Delta t}{\Delta x}\mathbf{A}^+ (\mathbf{C} - i\mathbf{S}) - \frac{\Delta t}{\Delta x}\mathbf{A}^- (\mathbf{C} + i\mathbf{S})}{\mathbf{L}_t} \tag{B.2.8}$$

We find that the amplification factor of the outer iteration can be achieved only when the full convergence of the inner iteration is obtained. However, the inner iteration is only computed for finite times between two outer iterations. Therefore, the overall amplification factor of such system is somewhat complicated and depends on several above factors. In order to find the overall application factor of dual system, we do the first two steps of the inner iteration.

Given $\mathbf{U}^{(0)} \equiv 0$, we compute $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$ from the above expression

$$\mathbf{U}^{(1)} = \mathbf{L}_t^{-1}\left(-\mathbf{R}^n\right) \tag{B.2.9}$$

$$\mathbf{U}^{(2)} = \mathbf{L}_t^{-1}\left(-\mathbf{R}^n + \frac{\Delta t}{\Delta x}\mathbf{A}^+ \mathbf{U}_{i-1}^{(1)} - \frac{\Delta t}{\Delta x}\mathbf{A}^- \mathbf{U}_{i+1}^{(1)}\right) \tag{B.2.10}$$

Let

$$\mathbf{U}^{(1)} = \mathbf{G}^{(1)}\mathbf{Q}^n \tag{B.2.11}$$

$$\mathbf{U}^{(2)} = \mathbf{G}^{(2)}\mathbf{Q}^n \tag{B.2.12}$$

we have

$$\mathbf{G}^{(1)} = -\mathbf{L}_t^{-1}(\mathbf{R}) \tag{B.2.13}$$

where

$$\mathbf{R} = \frac{\Delta t}{\Delta x}\left(\mathbf{A}^{+}(1-\mathbf{C}-i\mathbf{S}) - \mathbf{A}^{-}(1-\mathbf{C}+i\mathbf{S})\right) \tag{B.2.14}$$

$$\mathbf{G}^{(2)} = -\mathbf{L}_t^{-1}\left(\mathbf{R} - \frac{\Delta t}{\Delta x}\mathbf{A}^{+}\mathbf{G}^{(1)}(\mathbf{C}-i\mathbf{S}) + \frac{\Delta t}{\Delta x}\mathbf{A}^{-}\mathbf{G}^{(1)}(\mathbf{C}+i\mathbf{S})\right) \tag{B.2.15}$$

After careful analysis, we find that the amplification factor can be defined as

$$\mathbf{G} = \mathbf{I} + \mathbf{L}\mathbf{M}_m\mathbf{R} \tag{B.2.16}$$

where

$$\mathbf{M}_m = \mathbf{I} + \mathbf{K}\mathbf{L} + (\mathbf{K}\mathbf{L})^2 + \cdots + (\mathbf{K}\mathbf{L})^{m-1} \tag{B.2.17}$$

which corresponds to $m$ inner iterations. $\mathbf{K}$ and $\mathbf{L}$ are

$$\mathbf{K} = \mathbf{R} - \frac{\Delta t}{\Delta x}|\mathbf{A}| \tag{B.2.18}$$

$$\mathbf{L} = -\mathbf{L}_t^{-1} \tag{B.2.19}$$

The consistent second order upwind system gives the similar expressions, which are given by defining the following different components $\mathbf{R}$ and $\mathbf{K}$,

$$\mathbf{K} = \mathbf{R} - \frac{3}{2}\frac{\Delta t}{\Delta x}|\mathbf{A}| \tag{B.2.20}$$

$$\begin{aligned}\mathbf{R} = {} & \frac{\Delta t}{\Delta x}\left(\frac{3}{2}|\mathbf{A}| - \frac{1}{2}\mathbf{A}^{-}\left(2\mathbf{C}^2 - 1 + 2i\mathbf{S}\mathbf{C}\right) + 2\mathbf{A}^{-}(\mathbf{C}+i\mathbf{S})\right) + \\ & \frac{\Delta t}{\Delta x}\left(-2\mathbf{A}^{+}(\mathbf{C}-i\mathbf{S}) + \frac{1}{2}\mathbf{A}^{+}\left(2\mathbf{C}^2 - 1 - 2i\mathbf{S}\mathbf{C}\right)\right)\end{aligned} \tag{B.2.21}$$

For the I/II inconsistent upwind system, second order upwind scheme is used for the right hand side, while first order is used in the LHS. This results in

$$\mathbf{K}=\mathbf{R}-\frac{\Delta t}{\Delta x}\left(\frac{3}{2}|\mathbf{A}|+\frac{1}{2}\mathbf{A}^{-}\left(2\mathbf{C}^{2}-1+2i\mathbf{SC}\right)-2\mathbf{A}^{-}\left(\mathbf{C}+i\mathbf{S}\right)\right)+\frac{\Delta t}{\Delta x}\left(+2\mathbf{A}^{+}\left(\mathbf{C}-i\mathbf{S}\right)-\frac{1}{2}\mathbf{A}^{+}\left(2\mathbf{C}^{2}-1-2i\mathbf{SC}\right)\right) \tag{B.2.22}$$

$$\mathbf{R}=\frac{\Delta t}{\Delta x}\left(\frac{3}{2}|\mathbf{A}|-\frac{1}{2}\mathbf{A}^{-}\left(2\mathbf{C}^{2}-1+2i\mathbf{SC}\right)+\mathbf{A}^{-}\left(\mathbf{C}+i\mathbf{S}\right)\right)+\frac{\Delta t}{\Delta x}\left(-\mathbf{A}^{+}\left(\mathbf{C}-i\mathbf{S}\right)+\frac{1}{2}\mathbf{A}^{+}\left(2\mathbf{C}^{2}-1-2i\mathbf{SC}\right)\right) \tag{B.2.23}$$

With the above expressions, we can check how the inner iteration affects the overall amplification factors (A.F.). The I/I consistent difference system is examined first. In the cases we study, CFL number is set to 50.

Fig. B.1 is the amplification factor of the outer iteration. Remember that it can be obtained only when the inner iteration is fully converged. From Fig. B.2 we know that the inner iteration is convergent. We can see from Fig. B.3 to Fig. B.6 that the property of the outer amplification factor can be realized with the increase of the numbers of inner iterations. It is worthwhile to mention that the series is replaced by its exact value in Fig. B.6.

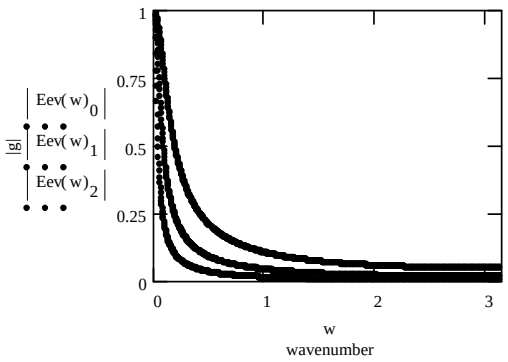



*Figure B.1 A.F. of the Outer Iteration(I/I)*

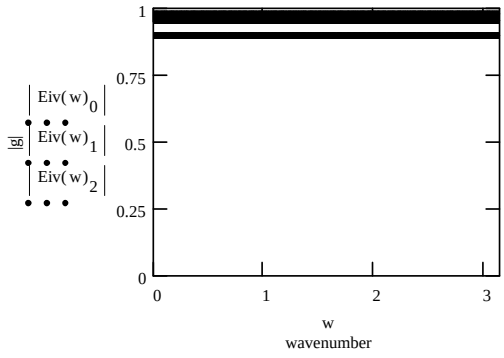



*Figure B.2 A.F. of the Inner Iteration(I/I)*

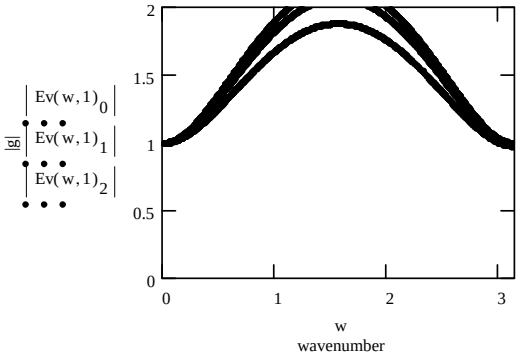



*Figure B.3 A.F. with 1 Inner Iteration(I/I)*

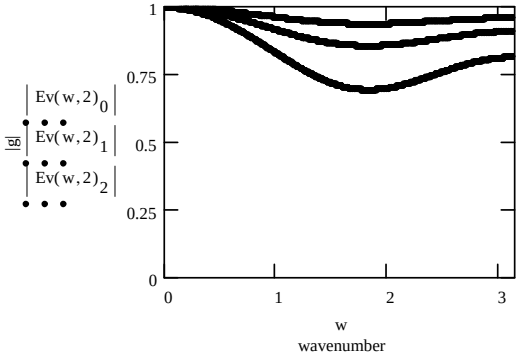



*Figure B.4 A.F. with 2 Inner Iterations(I/I)*

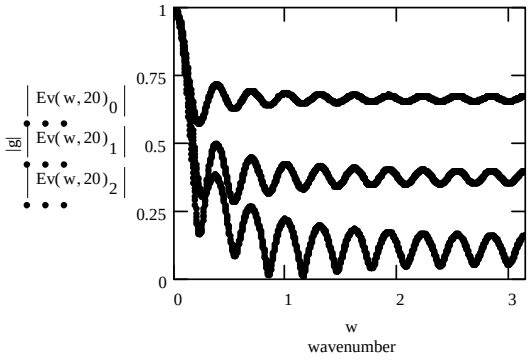



*Figure B.5 A.F. with 20 Inner Iterations(I/I)*

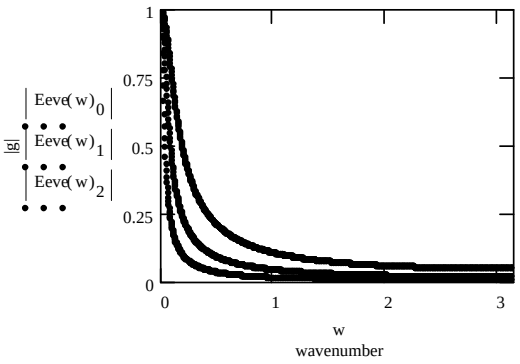



*Figure B.6 A.F. with Infinite InnerIterations(I/I)*

From Fig. B.7 to Fig. B.10 are the results of the second/second consistent upwind system with CFL number set to 50. Although the outer iteration is stable as we see from Fig. B.7, the inner iteration shown in Fig. B.8 is unstable. Therefore, we can expect that the overall stability cannot be obtained with the dual system, which are shown in Fig. B.9 and B.10. As we can see from Fig. B.10, one of the eigenvalues is less than unity, which may result in a stable approach. In Figures B.11 and B.12, stability of the Second/Second system is present with difference CFL numbers. Although theoretical analysis shows that his consistent system has its best performance with CFL number round 10, later we can see that the inconsistent system works much better.

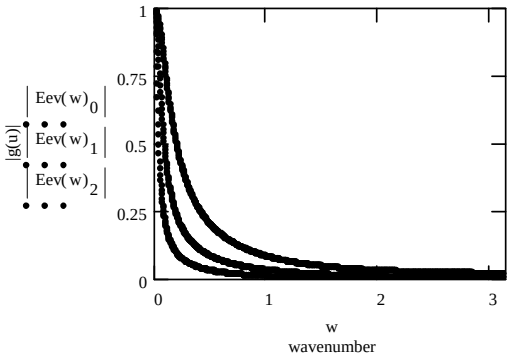



*Figure B.7 A.F. of the Outer Iteration(II/II)*

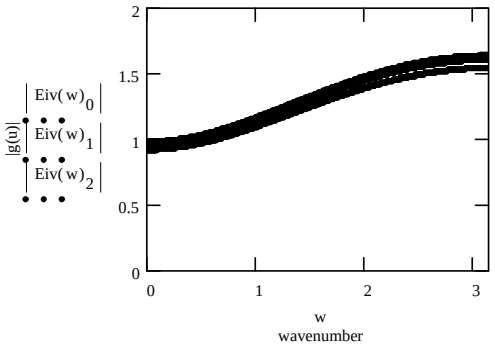



*Figure B.8 A.F. of the Inner Iteration(II/II)*

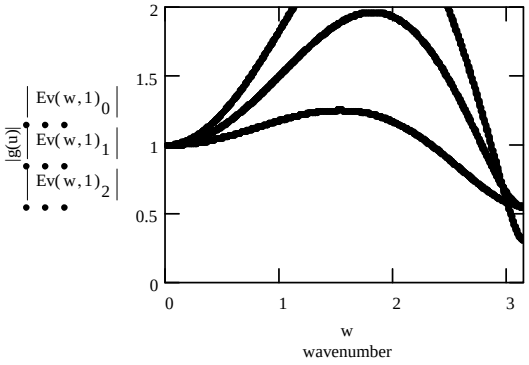



*Figure B.9 A.F. with 1 Inner Iteration(II/II)*

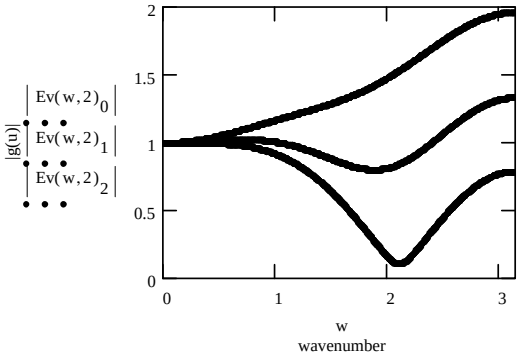



*Figure B.10 A.F. with 2 Inner Iterations(II/II)*

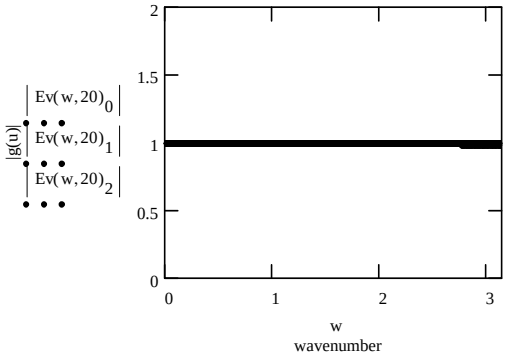



*Figure B.11 A.F. with 20 Inner Iterations (II/II,CFL=1)*

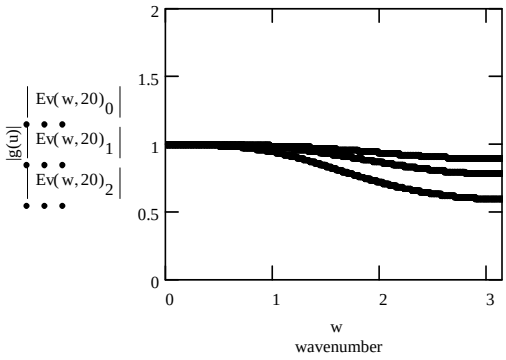



*Figure B.12 A.F. with 20 Inner Iterations (II/II,CFL=10)*

An alternative to the consistent system is the so-called inconsistent system, which means we use different schemes on sides of the equations. We analyze the I/II system from Fig. B.13 to Fig. B.16. Figure B.16 shows the amplification factor with infinite inner iterations. It is also the outer iteration amplification factor. We find that it results in a relatively large amplification factor at high wavenumbers. It is also freezing at the above profile even when CFL increases to infinity. Applications in the next session show that the inconsistent I/II upwind system with second order scheme in the RHS is unstable in some applications. It is also found that the generally used second order upwind scheme in unstructured mesh computation, which introduces less dissipation, is different from the scheme above.

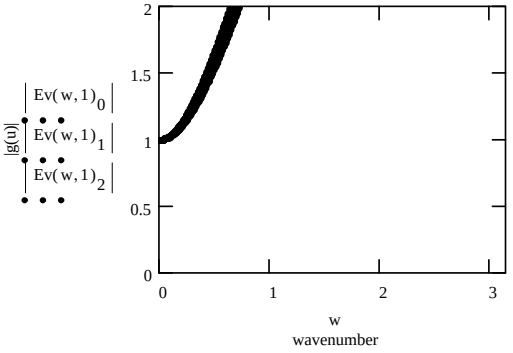



*Figure B.13 A.F. with 1 Inner Iteration(I/II)*

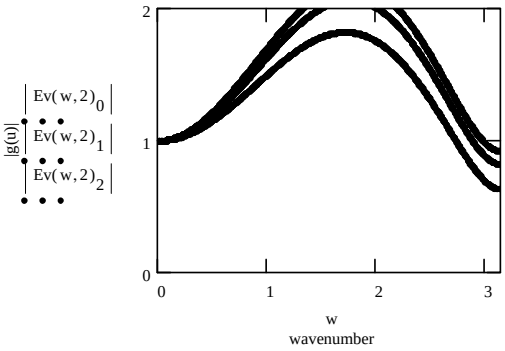



*Figure B.14 A.F. with 2 Inner Iterations(I/II)*

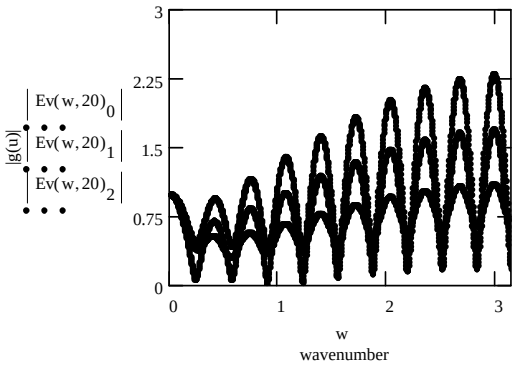



*Figure B.15 A.F. with 20 Inner Iterations(I/II)*

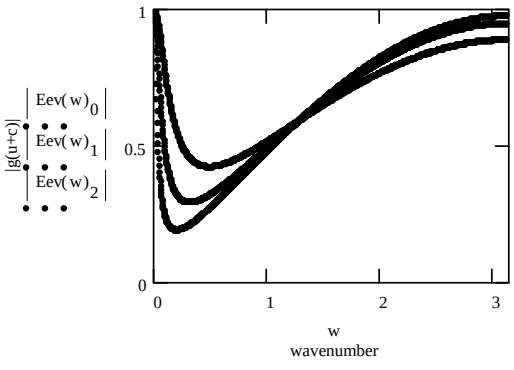



*Figure B.16 A.F. with Infinite Inner Iterations(I/II)*

## B.3 Inconsistent System for High Order Upwind Schemes

From the above work we find that the inconsistent upwind difference resolves the difficulties related to high order difference schemes. However, inconsistent system cannot recover the exact property of high order consistent difference when the convergence of the inner iteration is not fully achieved, especially at high wavenumber. For example, at wavenumber equal to $\pi$, it is found that the amplification factor is $\lambda/(1+\lambda)$, where $\lambda$ is CFL number. So at high CFL number, the damping rate at high wavenumber is very poor, compared with that of the consistent difference, as seen in Figures B.15 and B.7.

We can try to improve the inconsistent difference by designing new schemes for the LHS, since they don't affect the steady state solution. For simplicity, we start from the one-dimensional scalar equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \qquad a > 0 \tag{B.3.1}$$

Second-order upwind difference gives

$$\frac{\partial u}{\partial x} = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2} \tag{B.3.2}$$

is used for the RHS. Instead of putting the one-order upwind difference for the LHS, we define it with arbitrary coefficients,

$$\frac{\partial u}{\partial x} = au_{i+1} + bu_i + cu_{i-1} \tag{B.3.3}$$

In order that it is an appropriate expression for the gradient, we have the following restrictions on them

$$a+b+c=0 \tag{B.3.4}$$

$$a-c=1 \tag{B.3.5}$$

Another freedom is left for the optimization of the LHS. The amplification factor of this general difference is

$$g=\frac{1+\lambda\left(1-2C-2iS+C^2+iSC\right)+a\lambda\left(C+iS\right)+b\lambda+C\lambda\left(C-iS\right)}{1+a\lambda\left(C+iS\right)+b\lambda+C\lambda\left(C-iS\right)} \tag{B.3.6}$$

We can find the other relation by setting $|g|=0$ at wavenumber $\pi$

$$g=\frac{1+\lambda\left(4-a+b-c\right)}{1+\left(-a+b-c\right)} \tag{B.3.7}$$

Now the remaining relation is

$$b-a-c=-4 \tag{B.3.8}$$

and the final result is

$$a=-\frac{1}{2} \qquad b=2 \qquad c=-\frac{3}{2} \tag{B.3.9}$$

Recall that one-order upwind difference scheme is the central difference with second order dissipation,

$$\frac{u_i-u_{i-1}}{\Delta x}=\frac{u_{i+1}-u_{i-1}}{2\Delta x}-\frac{1}{2}\Delta x\frac{u_{i+1}-2u_i+u_{i-1}}{\Delta x^2} \tag{B.3.10}$$

the new scheme is

$$\frac{-u_{i+1}+4u_i-3u_{i-1}}{2\Delta x}=\frac{u_{i+1}-u_{i-1}}{2\Delta x}-\Delta x\frac{u_{i+1}-2u_i+u_{i-1}}{\Delta x^2} \qquad \text{(B.3.11)}$$

We find that it is the central difference plus more dissipation than first order upwind scheme, although we derive it in the other way.

Results from the above analysis are shown in Figs. B.17 ~ B.22. Figures B.21 and B.22 show the amplification factors of the modified system. Compared with the I/II system in Fig. B.19, it loses a little in middle wavenumber, while gains pretty much at high wavenumber. The inner iteration is also improved. Same as the old system, the profile will freeze at the CFL limit, as shown in Fig. B.21.

The application of the new system to the Euler equation is also tried (Figs.B.23-B.28). From the results we can see great improvement on the damping rate because of the combination of outer and inner iterations influences.

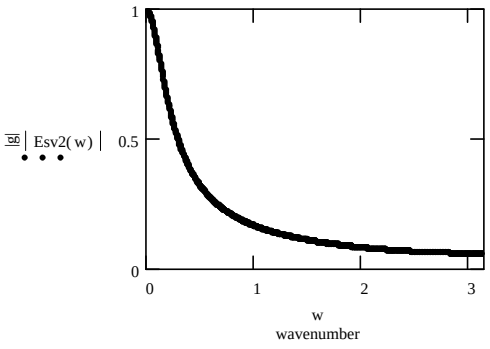



*Figure B.17 A.F. of the Outer Iteration (II/II, Scalar)*

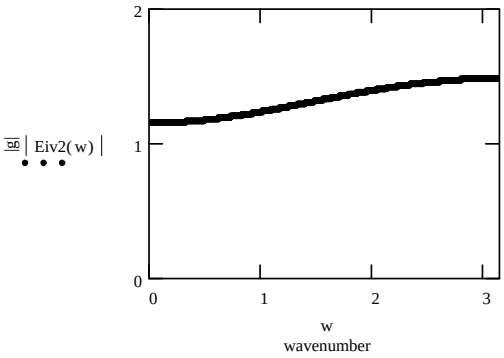



*Figure B.18 A.F. of the Inner Iteration (II/II, Scalar)*

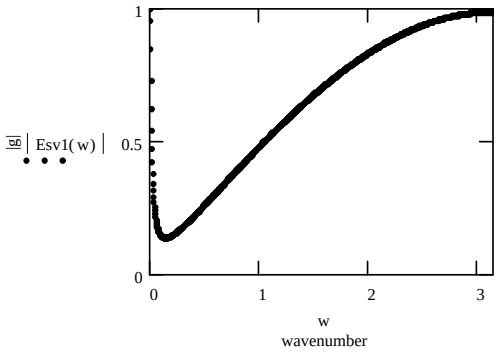



*Figure B.19 A.F. of the Outer Iteration (I/II, Scalar)*

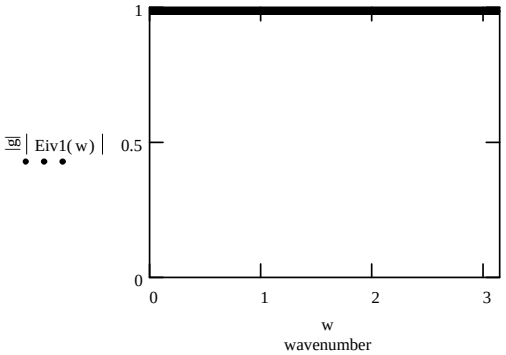



*Figure B.20 A.F. of the Inner Iteration (I/II, Scalar)*

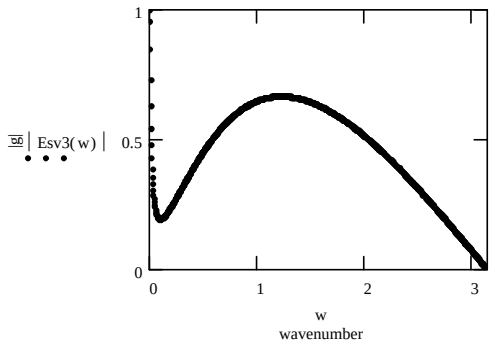



*Figure B.21 A.F. of the Outer Iteration (Mod/II, Scalar)*

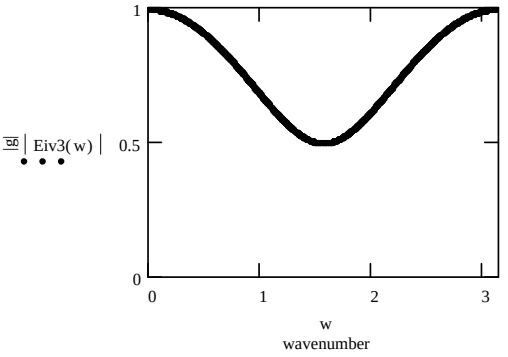



*Figure B.22 A.F. of the Inner Iteration (Mod/II, Scalar)*

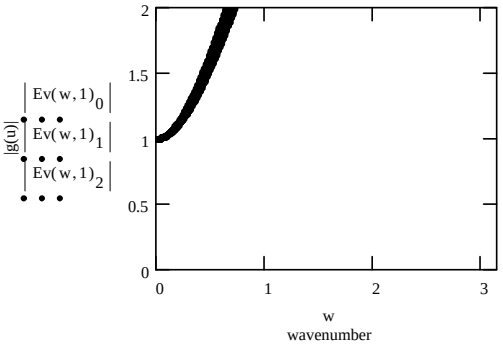



*Figure B.23 A.F. with1 Inner Iteration(I/II)*

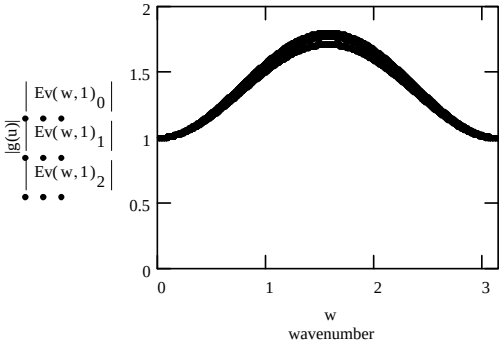



*Figure B.24 A.F. with1 Inner Iteration(Mod/II)*

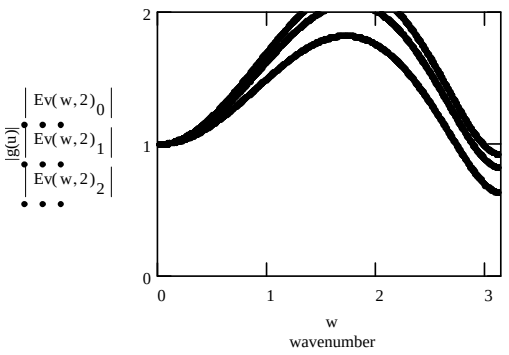



*Figure B.25 A.F. with 2 Inner Iterations(I/II)*

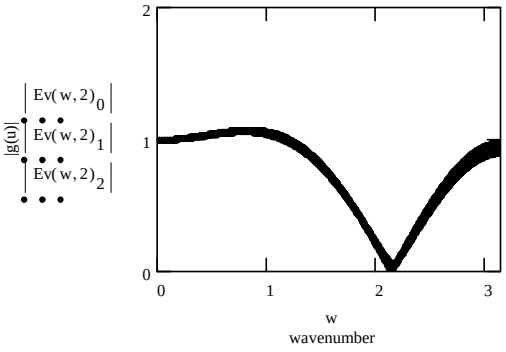



*Figure B.26 A.F. with 2 Inner Iterations(Mod/II)*

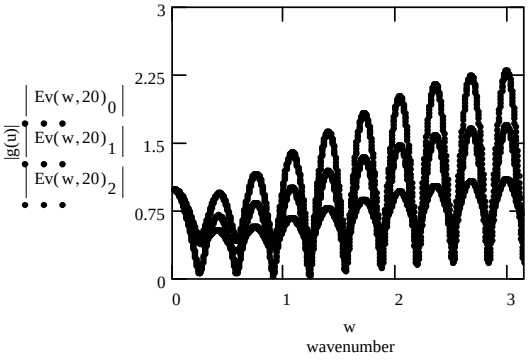



*Figure B.27 A.F. with 20 Inner Iterations (I/II)*

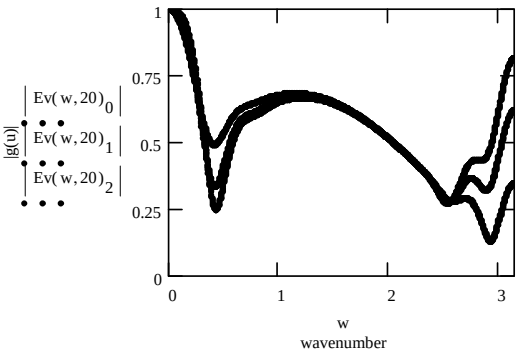


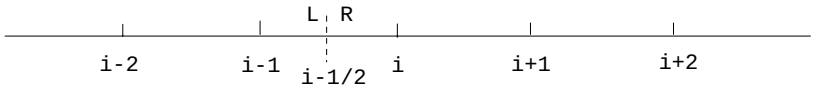

*Figure B.28 A.F. with 20 Inner Iterations (Mod/II)*

## B.4 Analysis of High Order Upwind Schemes via Variable Extrapolation

In the previous section, we have studied the performance of inconsistent system. The schemes we work on are essentially from finite difference method, where derivative is replaced by biased interpolation or extrapolation. In finite volume method, MUSCL approach [99,100], where upwind schemes are generated with variable extrapolation at the control volume interfaces, is used quite often both structured mesh and unstructured

mesh CFD solver. The fluxes are evaluated by the approximate Riemann solver, which is to resolve the difference between sides of the interface. Artificial dissipation is introduced by the Riemann solver, so overall it is still an upwind scheme.

Since our purpose is to investigate the performance of inconsistent systems, we are interested in the behavior of the second order schemes in MUSCL approach when inconsistent LHS is used. In the work below we study the discretized form of the scheme in one-dimensional application.




In this demonstration work, we will show the resulting difference schemes from several upwind approaches in current CFD simulation. The one-dimensional scalar equation is used in our work,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \qquad a > 0 \tag{B.4.1}$$

here we assume that $a$ is positive constant so that we can get around that trouble to evaluate $a$ in the control volume interfaces. Standard first, second and third order upwind difference schemes give

$$\frac{\partial u}{\partial x}_i = \frac{u_i - u_{i-1}}{\Delta x} \tag{B.4.2}$$

$$\frac{\partial u}{\partial x}_i = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2\Delta x} \tag{B.4.3}$$

$$\frac{\partial u}{\partial x}_i = \frac{2u_{i+1} + 3u_i - 6u_{i-1} + u_{i-2}}{6\Delta x} \tag{B.4.4}$$

In MUSCL approach, the solution variables at both sides of the cell interfaces are defined by a combination of backward and forward extrapolation

$$u^L_{i-1/2} = u_{i-1} + \frac{1}{4}\left[(1-k)(u_{i-1} - u_{i-2}) + (1+k)(u_i - u_{i-1})\right] \tag{B.4.5}$$

$$u^R_{i-1/2} = u_i - \frac{1}{4}\left[(1+k)(u_i - u_{i-1}) + (1-k)(u_{i+1} - u_i)\right] \tag{B.4.6}$$

For most compuations, linear interpoation is used between the upstream and downstream cell, which corresponds to the case $k = 0$. Thus we have

$$u^L_{i-1/2} = u_{i-1} + \frac{1}{4}(u_i - u_{i-2}) \tag{B.4.7}$$

$$u^R_{i-1/2} = u_i - \frac{1}{4}(u_{i+1} - u_{i-1}) \tag{B.4.8}$$

Upwind is introduced when evaluating the fluxes across the cell interface

$$(a\tilde{u})_{i-1/2} = \frac{au^L_{i-1/2} + au^R_{i-1/2}}{2} - \frac{1}{2}|a|\left(u^R_{i-1/2} - u^L_{i-1/2}\right) \tag{B.4.9}$$

Here since $a$ is assumed to be a positive constant, we can have the equivalent $\tilde{u}$ at the interface that can be used to evaluate the flux. That is

$$\tilde{u}_{i-1/2} = \frac{u^L_{i-1/2} + u^R_{i-1/2}}{2} - \frac{1}{2}\left(u^R_{i-1/2} - u^L_{i-1/2}\right) = u^L_{i-1/2} \tag{B.4.10}$$

This can also be expressed as the average value with dissspation added

$$\tilde{u}_{i-1/2} = \frac{u_{i-1} + u_i}{2} - \frac{1}{4}\left(u_i - 2u_{i-1} + u_{i-2}\right) \qquad \text{(B.4.11)}$$

Similarily, we can have the expression for the equivalent $\tilde{u}$ at the interface $i+1/2$

$$\tilde{u}_{i+1/2} = \frac{u_i + u_{i+1}}{2} - \frac{1}{4}\left(u_{i+1} - 2u_i + u_{i-1}\right) \qquad \text{(B.4.12)}$$

From the viewpoint of finite difference method, we can find the scheme corresponding to the discretized form of finite volume expression

$$\frac{\partial u}{\partial x}_i = \frac{\tilde{u}_{i+1/2} - \tilde{u}_{i-1/2}}{\Delta x} = \frac{u_{i+1} + 3u_i - 5u_{i-1} + u_{i-2}}{4\Delta x} \qquad \text{(B.4.13)}$$

This four point biased second order upwind scheme is different from the standard four point scheme mentioned above. In order to understand this scheme well, we can express the equivalent interface flux in the form below

$$\tilde{u}_{i-1/2} = \tilde{u}^L_{i-1/2} = \frac{u_{i-1} + u_i}{2} - k'\left(u_i - 2u_{i-1} + u_{i-2}\right) \qquad \text{(B.4.14)}$$

Then we can find different values of the upwind schemes

$k' = \frac{1}{2}$ second order upind

$k' = \frac{1}{6}$ third order upwind

$k' = \frac{1}{4}$ second order biased upwind

The upwind used in many finite volume approach is different from what we think it may be, the second order upwind scheme. Actually it is another four point biased upwind scheme with less dissipation than the second order upwind scheme. With more dissipation added, it is less accurate than the third order upwind scheme.

Now we may wonder how this new scheme performs in the inconsistent system. The amplification factor of the coefficients, which will be optimized later, is

$$g = \frac{1+\lambda\left(-\frac{1}{2}+C-\frac{3}{2}iS-\frac{1}{2}C^2+\frac{1}{2}iSC\right)+a\lambda(C+iS)+b\lambda+C\lambda(C-iS)}{1+a\lambda(C+iS)+b\lambda+C\lambda(C-iS)} \tag{B.4.15}$$

With the same method to determine the LHS scheme in our work on the second order upwind scheme, we find that the optimal scheme for the LHS is

$$a=0 \qquad b=1 \qquad c=-1 \tag{B.4.16}$$

So we find that first order upwind scheme is the optimal choice for the LHS, which can perform best with the second order upwind scheme generated from variable extrapolation in most finite volume solvers. Figures B.29 and B.30 show the amplification factors of the outer iterations when different LHS are used together with the second order biased upwind scheme(which is named as Biased Second order Upwind scheme) . It shows that first order upwind scheme presents a faster convergence rate than the optimal choice for the standard second order upwind system(modified LHS). The curves in Fig. B.29 and B.31 meet the criteria we use to optimize the left hand side.

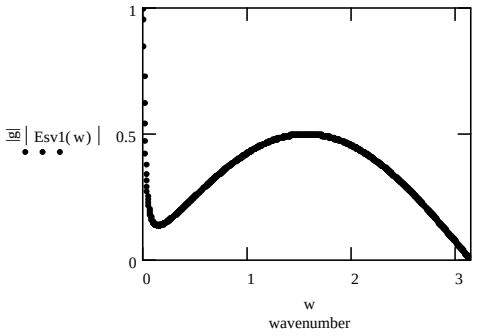



*Figure B.29 A.F. of the Outer Iteration (I/BSU, Scalar)*

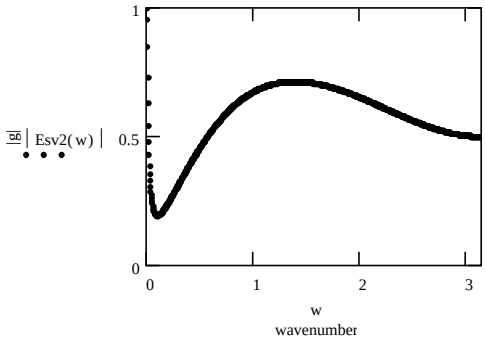



*Figure B.30 A.F. of the Outer Iteration (Mod/BSU, Scalar)*

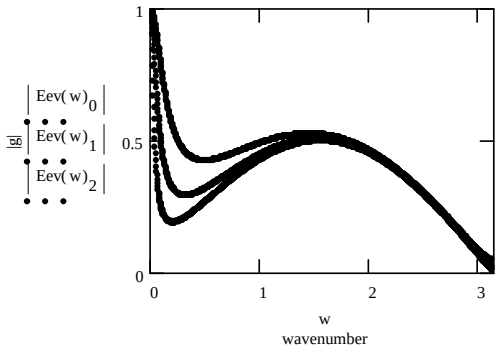



*Figure B.31 A.F. of the Outer Iteration (I/BSU, Euler)*

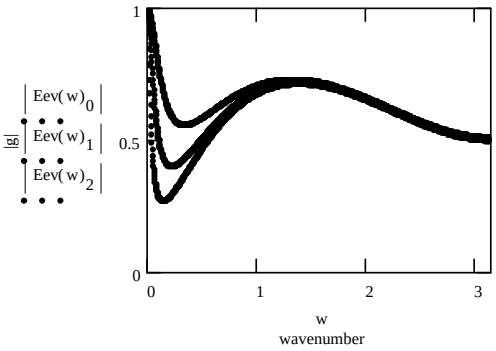



*Figure B.32 A.F. of the Outer Iteration (Mod/BSU, Euler)*

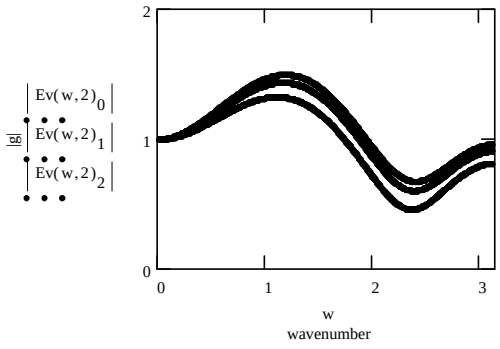



*Figure B.33 A.F. with 2 Inner Iterations (I/BSU, Euler)*

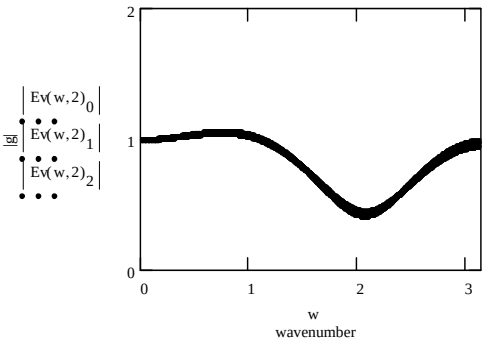



*Figure B.34 A.F. with 2 Inner Iterations (Mod/BSU, Euler)*

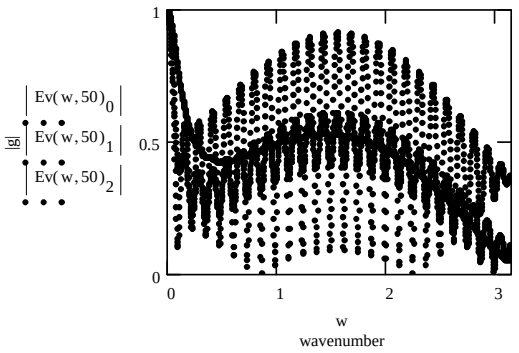



*Figure B.35 A.F. with 50 Inner Iterations (I/BSU, Euler)*

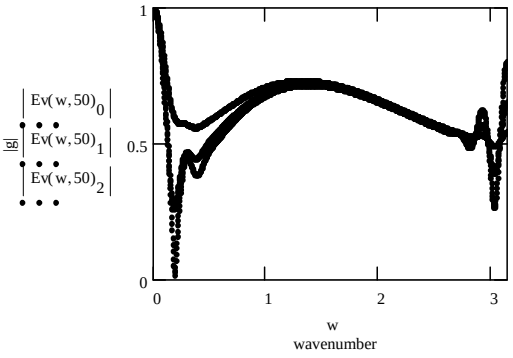



*Figure B.36 A.F. with 50 Inner Iiterations (Mod/BSU, Euler)*

## B.5 Application to 2-D Inviscid Flow Equations

In order for the further understanding of the inconsistent system for high order upwind scheme, it is helpful to apply the analysis to the two-dimensional equations. Before proceeding to the vector equations, the procedure is first applied to the scalar convection equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = 0 \qquad\qquad a > 0 \quad b > 0 \qquad\qquad \text{(B.5.1)}$$

Von Neumann analysis gives the amplification factor for the I/II upwind scheme as following

$$\left[1 + \lambda_x\left(1 - C_x + iS_x\right) + \lambda_y\left(1 - C_y + iS_y\right)\right](g-1) = -\lambda_x\left[\frac{3}{2} - 2\left(C_x - iS_x\right) + \frac{1}{2}\left(2C_x^2 - 1 - 2iS_xC_x\right)\right]$$
$$-\lambda_y\left[\frac{3}{2} - 2\left(C_y - iS_y\right) + \frac{1}{2}\left(2C_y^2 - 1 - 2iS_yC_y\right)\right]$$

(B.5.2)

where $\lambda_x = a\frac{\Delta t}{\Delta x}$, $\lambda_y = b\frac{\Delta t}{\Delta y}$.

Similiarily, we can find the formula for the Mod/II and I/BSU systems

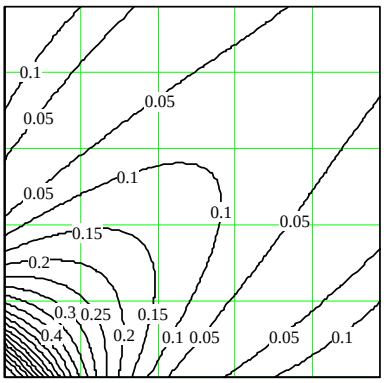
$$\left[1+\lambda_x\left(2-2C_x+iS_x\right)+\lambda_y\left(2-2C_y+iS_y\right)\right](g-1)=-\lambda_x\left[\frac{3}{2}-2\left(C_x-iS_x\right)+\frac{1}{2}\left(2C_x^2-1-2iS_xC_x\right)\right]$$
$$-\lambda_y\left[\frac{3}{2}-2\left(C_y-iS_y\right)+\frac{1}{2}\left(2C_y^2-1-2iS_yC_y\right)\right] \tag{B.5.3}$$

$$\left[1+\lambda_x\left(1-C_x+iS_x\right)+\lambda_y\left(1-1C_y+iS_y\right)\right](g-1)=$$
$$-\lambda_x\left[\frac{1}{4}\left(C_x+iS_x\right)+\frac{3}{4}-\frac{5}{4}\left(C_x-iS_x\right)+\frac{1}{4}\left(2C_x^2-1-2iS_xC_x\right)\right]$$
$$-\lambda_y\left[\frac{1}{4}\left(C_y+iS_y\right)+\frac{3}{4}-\frac{5}{4}\left(C_y-iS_y\right)+\frac{1}{4}\left(2C_y^2-1-2iS_yC_y\right)\right] \tag{B.5.4}$$

CFL numbers for both directions are set to 50.

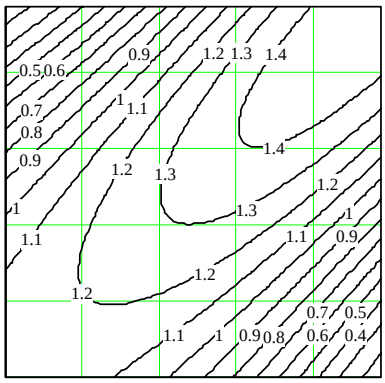
Fig. B.37 and FIg. B.38 are the stability results of the Second/Second system. We can find that although the outer iteration presents good stability, overall stability of the II/II system with Point-Jacobi iteration cannot be ensured because of the unstable inner iteration, which is shown in Fig. B.38. Comparison of the results in Figures B.39 and B.41, which are the stability of the outer iteration of the First/Second system and the Modified/Second system respectively, shows that the modified LHS does improve the stability of the second order upwind scheme. The inner iteration can also be enhanced much, as what can be seen from Figures B.40 and B.42.

Results for the biased upwind scheme are shown in Figures B.43 and B.44. First order upwind scheme presents better stability than the modified scheme as the LHS, which is consistent the results from one dimension study.



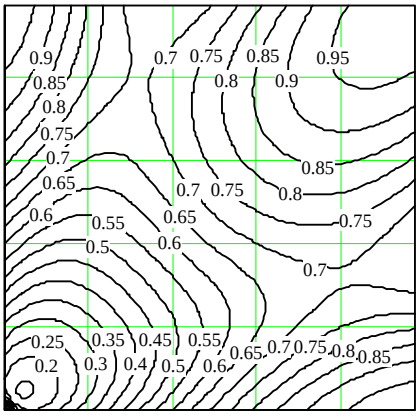

Es1

*Figure B.37 A.F. of the Outer Iteration (II/II, 2D Scalar)*



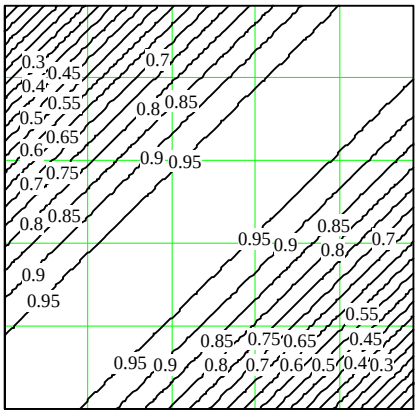

Ei1

*Figure B.38 A.F. of the Inner Iteration (II/II, 2D Scalar)*



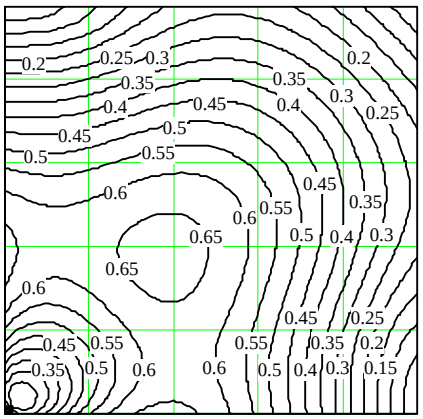

Es2

*Figure B.39 A.F. of the Outer Iteration (I/II, 2D Scalar)*



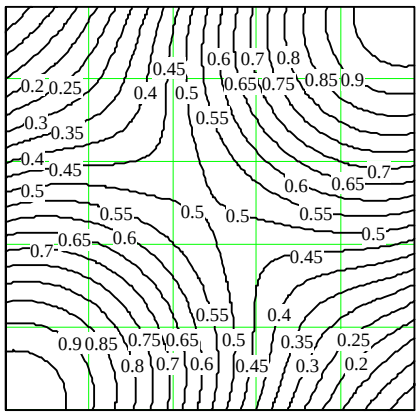

Ei2

*Figure B.40 A.F. of the Inner Iteration (I/II,2D Scalar)*



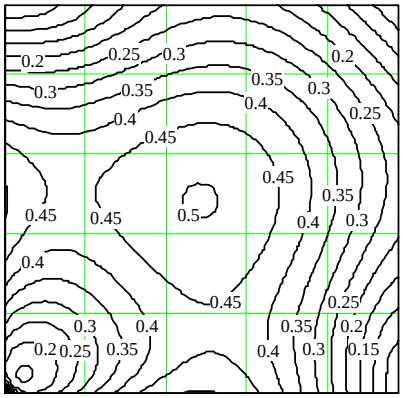

Es3

*Figure B.41 A.F. of the Outer Iteration (Mod/II, 2D Scalar)*



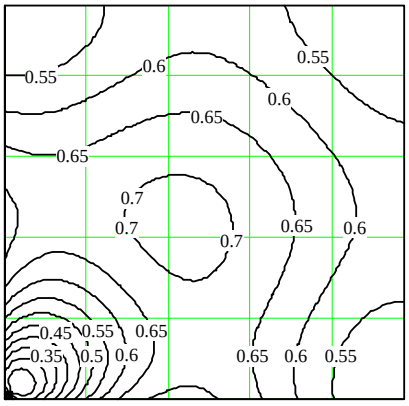

Ei3

*Figure B.42 A.F. of the Inner Iteration (Mod/II, 2D Scalar)*




Es4

*Figure B.43 A.F. of the Outer Iteration (I/BSU, 2D Scalar)*




Es5

*Figure B.44 A.F. of the Outer Iteration (Mod/BSU, 2D Scalar)*

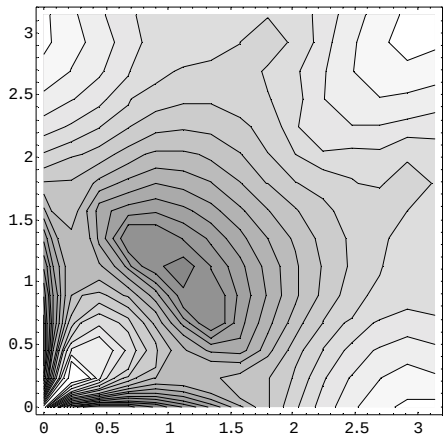



*Figure B.45 A.F. of the Outer Iteration (I/II, Euler)*

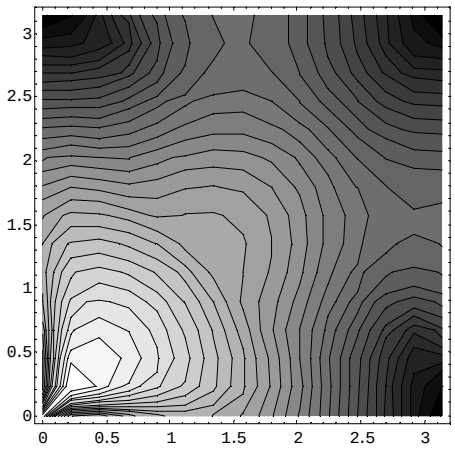



*Figure B.46 A.F. of the Outer Iteration (Mod/II, Euler)*

Extension of the stability study to Euler equations is studied also. Figures B.45 and B.46 are the out iteration stability properties of second order upwind scheme, when first order upwind scheme and the modified scheme are used as LHS. The magnitude of one of the eigenvalues is shown with contour. In both contours, light center regions represent large amplification values, which approach to unity at the origin. As we can see, the new scheme works better than first order upwind scheme.

With contrast to the second order upwind scheme, first order upwind scheme is a better choice for the LHS to work with the biased second order upwind scheme, which is used quite often in finite volume method with MUSCL approach. The contours in Figures B.47 and B.48 show how it works compared with the scheme which works very well with second order upwind scheme.

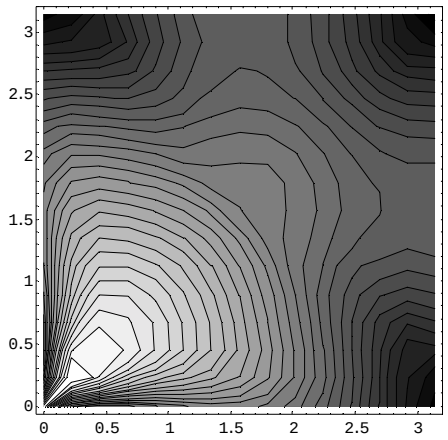



*Figure B.47 A.F. of the Outer Iteration (I/BSU, Euler)*

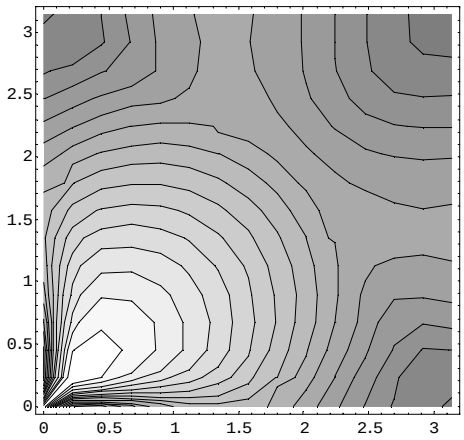



*Figure B.48 A.F. of the Outer Iteration (Mod/BSU, Euler)*

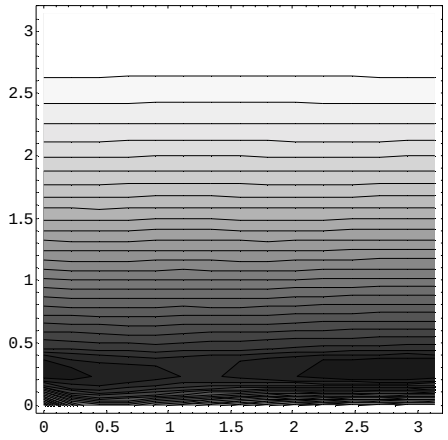



*Figure B.49 A.F. of the Outer Iteration (I/II, Euler,AR=100)*

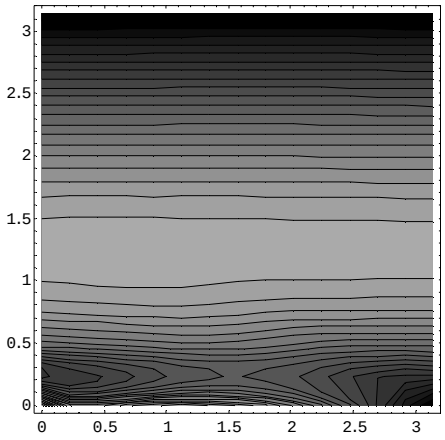



*Figure B.50 A.F. of the Outer Iteration (Mod/II, Euler,AR=100)*

The effect of grid aspect ratio(AR) is examined by the stability analysis (Figs. B.49~B.52). Grid aspect ration of 100 is used in calculation. CFL number in one direction $(y)$ is one hundred times greater than the other direction, which makes the pictures similar to the one dimensinal curves. Obviously the conclusions we have from the previous work stand in the above four figures.

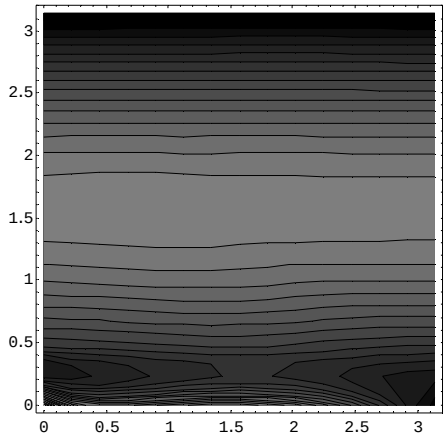



*Figure B.51 A.F. of the Outer Iteration (I/BSU, Euler,AR=100)*

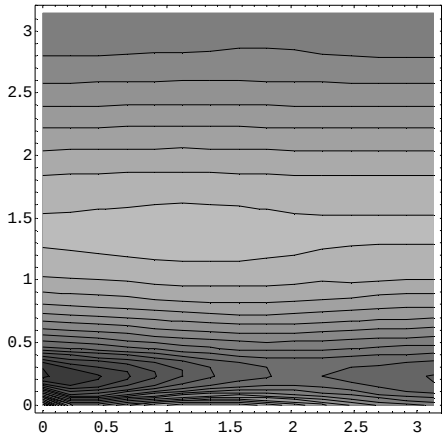



*Figure B.52 A.F. of the Outer Iteration (Mod/BSU, Euler,AR=100)*

## B.6 Conclusions

Stability analysis has been conducted for the implementation of different upwind schemes. Dual system is studied when Point-Jacobi iteration is used. The properties of the outer iteration can give the reason of the advantage and disadvantage of different LHS schemes no matter which kind of inner iteration is choosed. A new methodology for the design of LHS scheme is proposed ,with which we modify the LHS scheme for second order upwind scheme. This new LHS scheme renders second order upwind scheme stable, which is shown to be not possible with first order upwind. Extension of the analysis is made to other second order upwind scheme generated by MUSCL approach. We carefully study the performance of the second order biased upwind scheme. Although the optimal choice is adopted by nearly all the computations, the difficulty rooted in the inconsistent system is also shown in the calculations. Work towards the consistent LHS is under investigation.

# Vita

Guoping Xia was born in Jiangyan county, Jiangsu Province, China on May 29, 1973. He completed his five-year study in elementary school and six-year study in high school from 1980 to 1991 in his hometown. In 1991 he attended Xi'an Jiaotong University for his college study majoring in Internal Combustion Engine and earned his B.E. degree in July 1995. After that he began his Ph. D study under Professor Wenquan Tao's guidance at the same university. In early 1998 he came to the United States to pursue his Ph. D degree at the University of Tennessee Space Institute. Guoping got his Doctoral degree in May 2003.