ON GRIDS AND SOLUTIONS FROM RESIDUAL MINIMIZATION

by

Hiroaki Nishikawa

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering and Scientific Computing)
in The University of Michigan
2001

Doctoral Committee:

Professor Philip L. Roe, Chairperson
Professor Smadar Karni
Professor Kenneth G. Powell
Professor Bram van Leer

sit difficile; experiar tamen.

ACKNOWLEDGEMENTS

I would like to thank Professor Phillip L. Roe for being my adviser, from whom I learned so many. It is his backup with full of ideas that has kept me working on this challenging research topic. I am grateful also to Professor Bram van Leer, Professor Kenneth G. Powell and Professor Smadar Karni for being on my committee and for their constructive comments and suggestions.

On the financial side, I would like to thank Professor Bram van Leer and the Department of Aerospace Engineering for the opportunity of the teaching assistantship. Many thanks go to Mr. Yoshihisa Kobayashi at Koby International Academy in Novi Michigan who has been providing me with a teaching job at his school.

I would like to thank my parents in Japan for their understanding and support. I thank my friends here in Michigan and in Japan for their encouragement. Finally and principally, I thank my wife for her understanding, support, and forbearance.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF FIGURES . . . . . **vi**

LIST OF TABLES . . . . . **x**

LIST OF APPENDICES . . . . . **xi**

CHAPTER

I. INTRODUCTION **1**

1.1 Additional Degrees of Freedom 1
1.2 Grid Adaptation 3
1.3 Thesis Outline 8

II. THE LEAST-SQUARES MOVING GRID METHOD **12**

2.1 The Least-Squares Moving Grid Method 12
2.2 The Choice of The Grid 14
2.3 The Choice of The Norm 19
2.4 Relationship to Other Numerical Schemes 21
2.5 Implementation 24

III. PROBLEMS IN ONE DIMENSION **28**

3.1 The Least-Squares Method for Friedrichs' Model 28
3.2 A Geometric Interpretation 33
3.2.1 Geometric Solution of Friedrichs' Model 34
3.2.2 Discretization 38
3.2.3 The Least-Squares Method 39
3.3 Additional Property 43
3.4 Numerical experiments 45
3.5 A Nonlinear Example 48

IV. HYPERBOLIC PROBLEMS IN TWO DIMENSIONS **50**

4.1 Linear Hyperbolic Equations 50
4.1.1 Linear Advection 51
4.1.2 Linearized Aerodynamics 56
4.1.3 Burgers' Equation 60
4.2 Nonlinear Hyperbolic Equations 62
4.2.1 Conservation Laws 62
4.2.2 Quadrature Formulae 63
4.2.3 Formulas for the Residual 64
4.2.4 Detecting Compression/Expansion 68
4.2.5 Least-Squares Formulation 69
4.2.6 Results 70

V. ELLIPTIC PROBLEMS IN TWO DIMENSIONS 73

5.1 Cauchy-Riemann Equations: $\phi - \psi$ Formulation 73
5.1.1 Governing equations 73
5.1.2 Residual 74
5.1.3 The Least-Squares Method 76
5.1.4 Equivalence with a finite-element method 79
5.1.5 Grid Movement 85
5.1.6 Results 91
5.2 Cauchy-Riemann Equations: $u - v$ Formulation 96
5.2.1 A Problem of Computing Lifting Flows 96
5.2.2 An Accurate Least-Squares Scheme 102
5.2.3 A Third-Order Least-Squares Scheme 108
5.2.4 The Least-Squares Moving Grid Method 113
5.2.5 Moving Quadrilateral Grids 115

VI. CONCLUSIONS 121

6.1 One Dimension 121
6.2 Two Dimensions 122
6.3 Three Dimensions and Beyond 125

APPENDICES 128

BIBLIOGRAPHY 207

LIST OF FIGURES

1.1 A typical grid adapted by an h-method to the solution of a circular advection problem, $y\,\partial_x u - x\,\partial_y u = 0$.[0] 5

1.2 A typical grid adapted by an r-method for the same circular advection problem. 5

1.3 The adaptive grid generated by Roe's minimization scheme [80]. . . 7

1.4 The solution contours obtained simultaneously by the residual minimization. 7

2.1 Residual distribution on a triangular element 23

2.2 A pair of triangles for which diagonal swapping is feasible. The dashed line indicates the alternative. 27

2.3 A pair of triangles for which diagonal swapping is NOT feasible. The dashed line indicates the alternative that violates the validity of the grid. 27

3.1 The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes. 31

3.2 The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes. 32

3.3 The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes. 33

3.4 A geometric representation of the exact solution of the Friedrichs' model for a=0.3 and $\epsilon = 0.04$. The plane is defined by $\epsilon\,J + u - ay = constant$, and the curved surface is defined $y + \epsilon\,ln(J - a) = constant$ where the constants have been determined by the boundary conditions in (3.2). 37

3.5 The solution from an oscillatory initial solution and the exact solution as the dotted curve. 8 edges. 42

3.6 The solution from a straight initial solution and the trajectories of the vertices(dots). 8 edges. 42

3.7 The solution from a straight initial solution. 64 edges. 42

3.8 The initial approximation and the exact solution (the sold line). . . 45
3.9 The solution curve in (u,J,y) space and the trajectories of the 2nd, the 5th, and the 8th vertices(dots). 8 edges. 47
3.10 The projections of the curve onto $u-y$ and $J-y$ planes with the initial and the final locations of the vertices on y-axis indicated by $*$. 47
3.11 The solution curve in (u,J,y) space. 64 edges. 48
3.12 The projections of the curve onto $u-y$ and $J-y$ planes with the initial and the final locations of the vertices on y-axis indicated by $*$. 48
3.13 3D View of the solution 49
3.14 The final solutions u and J and the grid. 49

4.1 The residual represents the area of the triangle projected onto the characteristic plane. 52
4.2 A triangle in the physical plane which has zero area in the characteristic plane. 52
4.3 Typical nodal movement within a triangular element. 53
4.4 Initial Solution 54
4.5 Initial Grid 54
4.6 Converged Solution for $a = 0.7$ and $b = 1.0$. 55
4.7 The Final Grid 55
4.8 Left: Contours of u computed on a fixed grid indicated by the thicker lines. $\mathcal{F} \approx 10^{-3}$. Right: Contours of u and an optimal grid computed simultaneously, starting from the grid on the left. 18 nodes have been removed during the solution process. $\mathcal{F} \approx 10^{-12}$. 59
4.9 Converged Solution for the residual (4.31) 61
4.10 Converged solution for the residual (4.32) 61
4.11 Shock Recognition: $\alpha = 1$ 66
4.12 Characteristic Recognition: $\alpha = 0$ 67
4.13 Left: Element in expansion. Right: Element in compression. Arrows indicate the characteristic speed vectors. 68
4.14 The final grid for the curved shock. 71
4.15 Solution contours for the shock. 71
4.16 The final grid for the expansion. 72
4.17 Solution contours for the expansion. 72

5.1 A 40x20 O-grid generated by the least-squares method. 86
5.2 The grid in the solution space with an exponential stretching in the ψ direction 86
5.3 Definition of side vectors for a triangle T. 88
5.4 Definitions of edge vectors and angles in a group of triangles, $\{T_j\}$. . 88

5.5 A network of springs. 90
5.6 Contour plot (70 Levels) of the exact solution (k=22) of ϕ. 92
5.7 Contour plot (30 Levels) of the exact solution (k=22) of ψ. 92
5.8 Initial Delaunay grid with 568 triangles and 309 nodes. 93
5.9 Adaptive grid 93
5.10 A 160x80 O-grid. 97
5.11 C_p distribution around the airfoil and the airfoil geometry. Solid curve represents the exact solution. Circles indicate the numerical solution. 97
5.12 A mesh plot of the flow speed q. A 20x10 O-grid. 101
5.13 ru versus r^2 along the radial grid line at $\theta = 90°$. 101
5.14 A nonuniform regular triangular grid. 104
5.15 C_p distribution around the airfoil and the airfoil geometry. Solid curve represents the exact solution. Circles indicate the numerical solution. 160x80 O-grid. 106
5.16 The solid line indicates the line of convergence rate 1.7. The error in u and v were measured in L_1 norm, indicated by circles and stars respectively. N_E is the number of nodes in the radial direction of the O-grids. 106
5.17 C_p distribution on a Joukowsky airfoil at the angle of attack 10°, and the geometry of the airfoil. Circles are numerical solutions on a 80x40 O-grid while the solid curve represents the exact solution. . . 112
5.18 The solid line indicates the line of convergence rate 3. The error in u and v were measured in L_1 norm, indicated by circles and stars respectively. N_E is the number of nodes in the radial direction of the O-grids. 112
5.19 Initial 40x20 O-grid around a Joukowsky airfoil. 114
5.20 Adaptive grid. 114
5.21 C_p distribution on the initial grid. 115
5.22 C_p distribution on the adaptive grid. 115
5.23 A set of quadrilaterals $\{Q_j\}$ that share a node j. Each quadrilateral is divided into a pair of triangles A and B by dashed lines for the purpose of computing the residual and the gradient. 117
5.24 C_p contours on a regular O-grid. 118
5.25 C_p distribution on the regular grid. 118
5.26 C_p contours on an adaptive grid. 119
5.27 C_p distribution on the adaptive grid. 119

A.1 A piecewise linear approximation to a function in one dimension. . . 134
A.2 A piecewise linear approximation to a curve in three dimension. . . 134

A.3 Function (A.28) and a piecewise linear representation on a uniform grid with N_E =5. 141
A.4 Top: The first derivative of (A.28). Upper-Middle: Elementwise relative error. Lower-Middle: Actual local L_2 errors. Bottom: Elementwise relative error with the modification (A.27). The last three are plotted in the middle of each element with N_E =160. 141
A.5 The exact solution (A.30) with $c_s = 1.0$, and the numerical solution by a third-order Runge-Kutta Method with N_E =20. 143
A.6 $-log_{10}(N_E)$ vs. log_{10} (Relative Error) and the linear least-squares fits. 143
A.7 Example (a). $\mathcal{E}_{2(I)}$ =1.0E-04. 152
A.8 Example (b). $\mathcal{E}_{2(I)}$ =1.0E-02 152
A.9 Example (c). $\mathcal{E}_{2(I)}$ =1.0E-02 153
A.10 Example (d). $\mathcal{E}_{2(I)}$ =1.0E-01 153
A.11 Numerical and exact solutions for the scalar equation. $\mathcal{E}_{2(I)}$ =1.0E-03 159
A.12 Numerical and exact solutions for the stiff system. $\mathcal{E}_{2(I)}$ =1.0E-03 . 159
A.13 The transient region of the solutions shown in Figure A.12. 160
A.14 The numerical and exact solution curves in the phase space. 160
A.15 The exact solution and the numerical solution obtained after the first step. 164
A.16 The exact solution and the numerical solution after the second step. 164

B.1 Example (a). $\mathcal{E}_{2(I)}$ =1.0E-04. 171
B.2 Example (b). $\mathcal{E}_{2(I)}$ =1.0E-02 171
B.3 Example (c). $\mathcal{E}_{2(I)}$ =1.0E-02 171
B.4 Example (d). $\mathcal{E}_{2(I)}$ =1.0E-01 171

C.1 $-log_{10}(L)$ vs. log_{10} (Relative Error) and the linear least-squares fit. Equilateral Triangles. 178
C.2 $-log_{10}(L)$ vs. log_{10} (Relative Error) and the linear least-squares fit. Skew Triangles. 178

LIST OF TABLES

3.1 The results of the numerical experiments. 46

5.1 The results of the numerical experiments 94

A.1 The results for (A.28) 140
A.2 3rd order Runge-Kutta Method. 142
A.3 4th order Runge-Kutta Method. 142
A.4 Example (a): $p = 2$. 150
A.5 Example (b): $p = 3$. 150
A.6 Example (c): $p = 8$. 151
A.7 Example (d): $p = 4$. 151
A.8 Example (a): $p = 2$. 154
A.9 Example (b): $p = 3$. 154
A.10 Example (c): $p = 8$. 155
A.11 Example (d): $p = 4$. 155
A.12 The results for the scalar case. $p = 5$. 158
A.13 The results for the stiff system. $p = 2$. 158
A.14 The results for (A.68). $p = 5$. 162

B.1 Example (a): $p = 2$. 169
B.2 Example (b): $p = 3$. 169
B.3 Example (c): $p = 8$. 169
B.4 Example (d): $p = 5$. 169

LIST OF APPENDICES

A. An Alternative Method for One-Dimensional Problems 129

A.1 Introduction 129
A.2 Error estimates 132
A.2.1 Local error analysis 132
A.2.2 Leading terms 133
A.2.3 Error estimates 136
A.2.4 Numerical tests for accuracy 139
A.3 Algorithm 142
A.3.1 Equidistribution 142
A.3.2 Algorithm 145
A.3.3 Properties of the iteration formula 146
A.4 Applications 148
A.4.1 Approximation of functions and curves 148
A.4.2 Numerical integration 152
A.4.3 Initial value problems for ordinary differential equations 156
A.4.4 Boundary value problems for ordinary differential equations 160
A.5 Remarks 164

B. A Shooting Method for A Node Generation Algorithm 166

B.1 Node-Generation Algorithm 166
B.2 Shooting Method 168
B.3 Results 170

C. L_2 Error Estimates On Triangular Grids 173

C.1 L_2 Error for Quadratic Functions 173
C.2 Numerical Tests 177

D. A Geometrical View in Two Dimensions 179

D.1 Hyperbolic Equations 180

D.1.1 Linear Advection 180
D.1.2 Linearized Aerodynamics 183
D.1.3 Burgers' Equation 185
D.2 Cauchy-Riemann equations 187

E. A Least-Squares Norm for the Euler Equations 190

E.1 Introduction 190
E.2 A Dimensionally Consistent Set of Variables and Its Equations 191
E.3 A Decomposition Matrix 194
E.4 A Matrix for the Acoustic Subsystem 196
E.5 The Norm for the Euler Equations 198
E.6 Implementation 199
E.7 Summary 199

F. Computing the Exact Solutions for Airfoil Problems 201

CHAPTER I

INTRODUCTION

1.1 Additional Degrees of Freedom

One strategy to solve a problem is to introduce additional degrees of freedom. A good example is the method of Lagrange multipliers for the problem of finding constrained extrema for which a straightforward procedure to find extrema fails because of the coupling of the variables through the constraint. A constraint is linked with the minimization(or maximization) via a Lagrange multiplier, an additional degree of freedom, which converts the problem into an unconstrained optimization where all the variables can be treated as independent, and therefore a regular procedure becomes applicable. The implication is that a difficult problem becomes simple by introducing extra variables.

In computational fluid dynamics, extra degrees of freedom are often introduced to improve the solution accuracy. In the language of finite element methods, additional degrees of freedom are introduced to construct high-order elements in the form of either an increased number of nodes or storing extra unknowns at nodes such as derivatives of the solutions. The former corresponds to a local grid refinement(or enlarging a stencil) while the latter to the use of a high-order interpolation formula to represent a solution. In these cases, the problem does not necessarily become

simple, but that is the price one has to pay to obtain a better solution.

Another good example is the adjoint methods first introduced by Jameson[47] for aerodynamic optimization using computational fluid dynamics. A typical problem is a drag minimization for an airfoil with the governing equations such as the Euler equations as constraints. A naive approach is to evaluate the gradient, which indicates the direction of improvement, by making a small change in each design variable, and then recompute both the grid and flow variables for each change. But the cost of solving the governing equations becomes prohibitively expensive for a large number of the design variables. In the adjoint methods, Lagrange multipliers (or adjoint variables) are introduced so as to eliminate the flow variables in the expression of the gradient, resulting governing equations for Lagrange multipliers (adjoint equations). The cost of evaluating the gradient now amounts to mainly solving the governing equations and the adjoint equations once for each. This is in fact a well-established technique in optimal control theory[42] with the aim of reducing the cost of solving the constraint equations which is another interpretation of the role of Lagrange multipliers. The method is now well established in optimal design problems also, and besides it has recently found a new application area, error estimation for functional outputs such as lift or drag wherein the design variables are truncation errors (or local grid size)[76, 96]. In these cases, the extra variables not only reduce the cost of solution methods but also make it possible to obtain useful information that is otherwise difficult.

Yet another possibility, that is not as well established as these approaches, is to introduce the elements with adjustable nodes: create extra degrees of freedom in the form of nodal positions. Consider a linear advection problem which is equivalent to saying that a solution is constant along the advection direction. A reason-

able way to move the grid points is along this advection direction. Obviously, this produces the exact solution at any instant of time, and now the difficult problem becomes trivial. Unfortunately this procedure does not generalize easily to systems or higher-dimensional problems. It is therefore desired to create a more general guiding principle that drives the mesh movement. A class of methods called moving finite elements[22, 3] first introduced by Miller[64, 65] is one promising approach in this context, in which the mesh movement, as well as the solution, is driven by a least-squares residual minimization. The method has been studied primarily for time-dependent problems: tracking steep fronts as they move in time or keeping an accurate representation of a smooth solution moving in time. Methods carefully tuned for each particular problem have been shown to give impressive results, that otherwise requires an excessive number of nodes to resolve rapidly varying solutions moving in time. Notice that tailoring the computational grid is no less (or even more) important than improving a numerical scheme on a fixed grid because a good grid does not demand an exceptionally good scheme whereas a good scheme still does demand a good grid. Introduction of the extra degrees of freedom in the nodal positions is therefore worthy of a particular attention.

1.2 Grid Adaptation

This thesis explores the possibility of moving grids, but the focus is on steady-state problems. In other words, we are concerned with a class of techniques called grid adaptation: modify the grid to capture the solution features at a steady-state. This is an essential item in computational fluid dynamics to deal with an inevitable dilemma: the computational grid, on which the solutions are to be computed, must be tailored to all the flow features which, however, are available only after the computation. A

good example is shock waves on an airfoil at transonic speed, which require extra resolution, yet whose positions are known only *a posteriori*. Grid adaptation is therefore an iterative process in which solution computation and grid alteration is alternatively repeated as desired. There have been numerous attempts in the past couple of decades to generate solution adaptive grids, *automatically*, *efficiently* and *effectively*, but nothing universal seems to have been developed yet, and it remains an important research area in computational fluid dynamics.

Many of the current adaptation techniques rely on one or both of two types of strategies for mesh alteration. One is local mesh refinement/derefinement (h-method) [11, 26, 75, 59, 28, 2] in which a computational cell is subdivided into smaller cells for higher resolution or a group of neighboring cells are merged into a larger cell to avoid unnecessary resolution, thus attempting equidistribution of error. At the cost of a simple (I,J,K) data structure, methods based on this strategy offer a great flexibility to deal with a wide range of length scales, a typical feature of computational fluid dynamics. The other strategy is mesh movement (r-method) [43, 62, 73, 33, 40, 69], in which the grid points are redistributed for better overall resolution based on a certain principle such as functional minimization or error equidistribution. An advantage over h-methods is that both the grid size and the data structure can be kept invariant, and therefore it is relatively easy to apply even for structured grids[33, 40, 69, 88].

A disadvantage of h-methods is its isotropic nature: no directional information is taken into account, and only the resolution is altered. This usually results an excessive grid refinement for anisotropic flows such as a shock or a boundary layer (Figure 1.1). R-methods also have a drawback that unless an initial mesh is fine enough, a significant number of nodes may be removed from the regions that are judged to be less important, but that still require a fair number of nodes, which

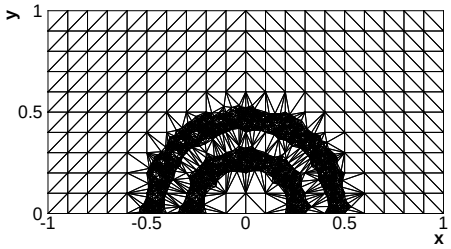

Figure 1.1: A typical grid adapted by an h-method to the solution of a circular advection problem, $y\,\partial_x u - x\,\partial_y u = 0$.[0]

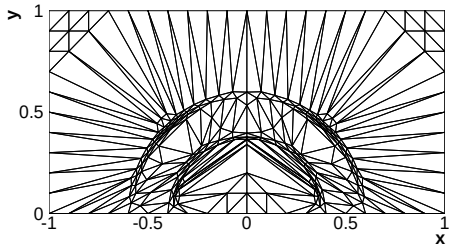

Figure 1.2: A typical grid adapted by an r-method for the same circular advection problem.

often creates a severe distortion of the grid leading to deterioration of the numerical accuracy (Figure 1.2). This is where a combination of the two strategies come into play: local refinement can be used to cure such a problem of moving mesh methods. But their combination does not seem meaningful if used merely to cluster nodes in the regions where solutions undergo large variations; an h-method alone would do the job. In other words, a combination of the two strategies with a common aim does not make sense.

One of the problems that techniques based on these ideas suffer from is the isotropic nature of the resulting meshes. For instance, this necessitates clustering an excessive number of nodes around shock waves that could be captured more efficiently simply by aligning cell edges along them. Other examples include vortical flows, boundary layers, flow separation, etc. This is because many methods focus on merely optimizing the grid point density (resolution), paying no attention at all to the directionality of the solution. The importance of anisotropic meshes has recently been recognized in the context of grid adaptation[41, 23], and also in the related

[0]Solutions are initially set zero except at two grid points on the x-axis $x = -0.4$ and -0.3, and these two values are to be convected along circular paths.

problem of interpolating a known function[79, 29], and has become one of the major topics in computational fluid dynamics.

The most powerful, and perhaps essential, tool for the anisotropic adaptation is the mesh movement which provides the most efficient way to alter the the local cell-orientation. Many mesh movement methods have been developed merely as a means to move the grid points towards the regions that require high resolution, and not as a means to alter the cell-orientation. Unlike h-methods, this is, however, not an inherent limitation of the redistribution strategy itself, but largely due to the heuristic nature of the guiding principle. Commonly, equidistribution of an error indicator is attempted which is based on solution gradient or second derivatives(the choice of the variable for the derivatives is another source of the ambiguity). Methods based on such error equidistribution tend to cluster too many nodes around discontinuous solutions because such error indicators do not converge there, and therefore all the nodes are meant to be moved to the discontinuities to achieve the equidistribution. This suggests that we abandon the equidistribution approach, and seek a new approach that is not based on just the solution variation. A class of methods, called variational methods, is one promising approach in this respect[58, 53, 15, 16, 49], in which not only adaptive criteria but also grid qualities such as orthogonality or smoothness can be incorporated into the minimization, and notably it is capable of producing a grid that is aligned with a prescribed vector field[16, 35]. However, the methods are currently applicable to structured grids only which limits the potential of the technique.

Recently, a new approach was proposed by Roe[80] that simultaneously produces numerical solutions and the grid tailored for the solutions, based on a least-squares residual minimization principle similar to the moving finite elements. The grid move-

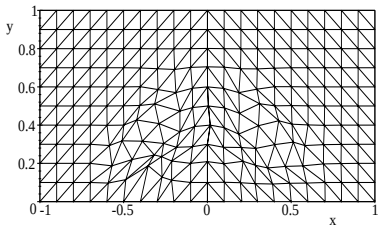

Figure 1.3: The adaptive grid generated by Roe's minimization scheme [80].

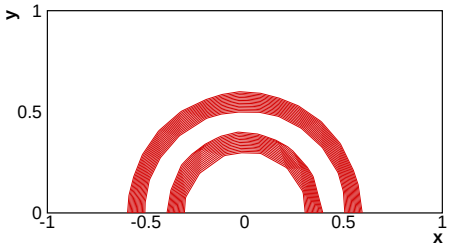

Figure 1.4: The solution contours obtained simultaneously by the residual minimization.

ment however significantly differs from that of moving finite elements in that the movement does not follow the physical path such as the advection direction, and rather it is perpendicular to that direction, trying to construct the physical path on which the exact solutions can be found at steady-state(Figures 1.3 and 1.4). Herein lies the distinction between time-dependent and steady-state problems. A little surprisingly, the grid movement driven by this simple minimization principle can be shown to reflect faithfully the physics of the governing equations: automatically creating characteristic grids for hyperbolic equations, and isotropic grids, but responsive to singularities, for elliptic equations. Note that a combination of an h-method and an r-method now makes sense because they have two different aims and truly compensate for each other's disadvantages. A least-squares approach to the numerical solution of differential equations is, however, not new: it is related to first-order system least-squares finite element methods[48, 18, 17]. But the unification of the two processes, computation of numerical solutions and an optimized grid for steady-state problems, is a new idea, which motivated the studies presented in this thesis.

It is important to note that the solution updates and the grid movement are driven by the same equations, i.e. there are no extra equations prepared specifically for the grid movement. This means that the method requires a highly overdetermined

problem with enough extra degrees of freedom available to move the grid. As will be discussed in Chapter II, this is possible in two and three dimensions, but it demands the use of specific types of cells for computational grids : triangles in two dimensions and tetrahedra in three dimensions. Fortunately, grids composed of these cells are very flexible in generation of a grid for arbitrary geometries as well as in alterations such as removing/adding nodes or changing connectivity. These are very important items for grid movement methods for avoiding mesh distortion often caused by large mesh movement especially in quadrilateral or hexahedral grids.

1.3 Thesis Outline

The thesis begins with the general description of the moving grid method in Chapter II. The principle of the solution strategy is given, the choice of triangular and tetrahedral grids is justified, the relationship with other methods is discussed, and the detailed algorithm is described in the general setting. The focus of the rest of the chapters is then on what grids and solutions can be obtained for each particular problem *and* how they are created, especially the mechanism of the grid movement.

In Chapter III, the least-squares method is applied to a one-dimensional boundary-value problem which entails a boundary layer behavior calling for an adaptive grid. This is a typical solution of the Navier-Stokes equations, and our aim is then, although at a fundamental level, to give insight into the problem of efficiently representing the boundary layer solution. Also, this is an instructive example from which we see what it means to include grid coordinates among the variables and what the least-squares method attempts to do. A common thought about adaptive grids is that numerical solutions are computed on an adjustable grid. Here, a geometrical view is taken that a numerical approximation is constructed for the solution manifold

that resides in a higher dimensional space whose coordinates are solution variables and grid coordinates, i.e. all the variables are regarded as independent. This interpretation proves to be useful to understand the nodal movement driven by the least-squares method.

In two dimensions, any partial differential equations can be decomposed into a certain number of hyperbolic advection equations and elliptic systems [81]. Based on this fact, we consider hyperbolic and elliptic problems separately. The hyperbolic part is then the scope of Chapter IV. It is also necessary to distinguish linear and nonlinear problems within the hyperbolic problems because of the different nature of discontinuous solutions. It is known that the least-squares method works well for linear hyperbolic problems[80, 82]. As will be demonstrated in this thesis, the method is capable of creating characteristic grids on which exact solutions can be found simultaneously. It is shown that the method attempts to minimize the area of the computational cell(or element) projected onto a characteristic plane, thereby satisfying the characteristic relations. Hence, with the extra degrees of freedom, the problem of computing a discontinuous solution on a computational grid becomes a simple problem of minimizing the cell area in the characteristic plane. Also shown is the possibility to capture discontinuous solutions *perfectly* by means of degenerate elements (elements with zero physical area). The method however fails at nonlinear shocks where characteristics merge. As will be discussed in detail in Chapter IV, the method must be designed to generate not a characteristic mesh but a shock mesh and simultaneously computes solutions satisfying jump relations. We show that a shock wave can be captured very economically, and also that it is achieved, contrary to the usual practice, with nodes removed rather than added.

As a model system for elliptic problems, we consider the Cauchy-Riemann equa-

tions in Chapter V. It is shown that the least-squares method is equivalent to the Galerkin finite-element method for solutions and an elliptic grid generation method for grid movement. Hence, the method seeks a simultaneous solution of the two methods. Also shown is the fact that the mesh movement driven by the discrete Cauchy-Riemann system is equivalent to a well-known mesh movement technique called spring analogy with the solution Jacobian as a stiffness, which in turn implies the equivalence of elliptic grid generation and the spring analogy technique. It has been discovered however that the least-squares method applied to compute a flow around a lifting airfoil loses its accuracy almost completely: the accuracy is somewhere between the zeroth order and the first order. Although many researchers have been using the least-squares method to solve the Cauchy-Riemann equations, this problem seems not to have been noticed. A detailed discussion on this problem is given, and a remedy is proposed subsequently. Then, the method is extended to achieve the third-order accuracy, and shown to produce very accurate solutions and even more when coupled with grid movement. Finally, it is demonstrated that a checkerboard error mode that is a typical problem in cell-vertex methods for quadrilateral grids [67] can be eliminated completely by the grid movement, creating a checkerboard mode in the grid instead. An important message here is that a grid generally believed to be good is not necessarily good; and a grid generally regarded as bad can help a scheme produce a highly accurate solutions.

The study presented here is fundamental and by no means ready for practical applications. It remains to be extended to more complex equations such as the Euler and the Navier-Stokes equations. A trouble encountered in such applications is that the least-squares method is not an accurate scheme on triangular grids. In fact, Jiang suggests not to use triangular grids based on the argument that the problem is too

overdetermined on triangular grids[48]. This seems not to be a valid argument since, when grid movement is permitted, the problem is hardly at all overdetermined, but the solutions are still poor. We believe that the real difficulty is that the residual is not always a good measure of the error and that, even if it were, minimizing a wrong norm still could produce poor solutions. The key is therefore the discovery of the right residual and the norm. For all the problems we have considered so far, an appropriate norm has actually been found. We believe that it can be done for more complex problems, and we have just embarked upon the application to the Euler equations. And the essence is contained in this thesis.

CHAPTER II

THE LEAST-SQUARES MOVING GRID METHOD

2.1 The Least-Squares Moving Grid Method

In this chapter, the least-squares approach introduced by Roe[80], to which we will refer as a least-squares moving grid method, is described. Suppose that we seek a numerical solution of the following equations written in an abstract form,

$$\mathcal{L}(u) = 0 \tag{2.1}$$

in a domain of interest of one, two, or three dimensions. We divide the domain into a set $\{C\}$ of cells, e.g. triangles in two dimensions. Then, storing the solutions at the vertices that form the cells, and assuming a certain variation of the solutions within each cell, we define the residual, a measure of the error in satisfying the equations, to be the integral of the governing equations over the cell.

$$\Phi_C = \int_C \mathcal{L}(u) \tag{2.2}$$

which defines a vector in general. Typically, the number of residuals exceeds the number of vertex values, thus leading to an overdetermined problem. Therefore the residuals cannot be made to vanish within every cell. We then consider solving the

equations in a least-squares sense by minimizing some functional

$$\mathcal{F} = \sum_{C\in\{C\}} F_C = \frac{1}{2}\sum_{C\in\{C\}} \Phi_C^t Q_C \Phi_C \tag{2.3}$$

where the superscript t denotes transpose and Q_C is a positive definite symmetric matrix. The matrix Q_C has two important roles; first, it assigns relative weight to the components of the residual, and second, it weights the errors in each cell relative to others. Now, exploiting the fact that the problem is overdetermined, implying extra degrees of freedom available, Roe proposed to minimize the norm with respect to not only the nodal solutions $\mathbf{u}_j$ but also the nodal coordinates $\mathbf{x}_j$, say by the steepest descent technique [80],

$$\delta\mathbf{u}_j = -\omega_u \frac{\partial\mathcal{F}}{\partial\mathbf{u}_j} \tag{2.4}$$

$$\delta\mathbf{x}_j = -\omega_x \frac{\partial\mathcal{F}}{\partial\mathbf{x}_j} \tag{2.5}$$

where δ denotes the change made to nodal quantities , and ω_u and ω_x are small constants. Thus the grid is now set in motion. The gradients are obtained simply by taking the derivative of $\mathcal{F}$ with respect to the nodal solution and the coordinate, each of which is the sum of terms arising from each of the set of cells $\{C_j\}$ that share that node. Thus

$$\frac{\partial\mathcal{F}}{\partial\mathbf{u}_j} = \sum_{C\in\{C_j\}} \left(Q_C \frac{\partial\mathbf{\Phi}_C}{\partial\mathbf{u}_j}\right)^t \mathbf{\Phi}_C \tag{2.6}$$

$$\frac{\partial\mathcal{F}}{\partial\mathbf{x}_j} = \sum_{C\in\{C_j\}} \left(Q_C \frac{\partial\mathbf{\Phi}_C}{\partial\mathbf{x}_j}\right)^t \mathbf{\Phi}_C \tag{2.7}$$

where the matrix Q_T has been assumed to be frozen. Note that this is a scheme of distribution type[84, 82, 31] that can be coded in a very simple way. In a loop over elements, we evaluate the residual $\mathbf{\Phi}_C$ for each element, and then distribute it to the nodes forming that cell with the distribution coefficient (or matrix) $\left(Q_C \frac{\partial\mathbf{\Phi}_C}{\partial\mathbf{x}_j}\right)^t$,

thus accumulating the contribution from the surrounding cells for each node as in (2.6) and (2.7). The nodal solution and the coordinate can then be updated by the contributions gathered at that node.

This simple minimization strategy created a new approach to mesh movement methods, i.e. driven not by solution variations but by nonvanishing residuals. In other words, it is based not on the nature of the solution but *on the nature of the equations*. This is the principle of the least-squares moving grid method. Unlike other moving mesh techniques, in this approach, the mesh movement can be shown to be responsive faithfully to the physics of the governing equations. However, the choice of quadrature for Φ and the weighting matrix Q_C are both subtle issues as will be discussed in details in the rest of the thesis.

2.2 The Choice of The Grid

As mentioned earlier, the number of residuals is not always equal to the number of unknowns. In other words, the number of elements is not always equal to the number of nodes. Specifically, in order to move the grid, we need extra degrees of freedom, meaning that the number of elements must exceed the number of nodes. This is however not always possible. For example, on a one-dimensional grid with N_C cells and N_V vertices (or nodes), we have obviously

$$N_V - N_C = 1. \tag{2.8}$$

Suppose that we wish to solve p differential equations with p unknowns. Then, we have pN_C residuals and pN_V unknowns which are related by

$$pN_V - pN_C = pN_V - pN_V + p = p \tag{2.9}$$

where (2.8) has been used. This shows that we do not have any extra degrees of freedom to move the grid because the number of unknowns exceeds the number of residuals. Particularly, if we include the position of the vertices as unknowns, we have

$$(p+1)N_V - pN_C = (p+1)N_V - pN_V + p = N_V + p, \tag{2.10}$$

resulting a highly underdetermined problem[1]. It is therefore necessary to introduce additional equations to move the grid. We shall discuss this in the next chapter.

In two dimensions, there are many possibilities because there exist a variety of choices for the type of cells. We begin with Euler's formula which relates the number of vertices N_V, edges N_E and faces N_F of a compact surface dissected by polygons:

$$N_V - N_E + N_F = 2. \tag{2.11}$$

Suppose that we have generated a computational grid around N_B objects which extends to a finite distance from the objects. Then we have, noting that the outside of the grid is also a face,

$$N_F = N_C + (N_B + 1) \tag{2.12}$$

where N_C is the number of cells in the grid. Substituting this into (2.11), we obtain Euler's formula for computational grids in two dimensions,

$$N_E = N_V + N_C + (N_B - 1). \tag{2.13}$$

On the other hand, visiting the cells and marking edges, we obtain

$$2N_{E_i} + N_{E_b} = sN_C \tag{2.14}$$

---

[1]Exactly speaking, we must take into account the boundary conditions. But they are typically $2p$ conditions at most. So the problem remains underdetermined by $N_V - p$ equations.

where N_{E_i} and N_{E_b} are the numbers of edges in the interior and on the boundaries of the grid respectively, and s is the number of edges of which each polygonal cell is composed[2]. Eliminating N_{E_i} from (2.13) and (2.14), we have

$$2N_{E_b} = 2N_V + (2-s)N_C + 2(N_B - 1). \tag{2.15}$$

Yet, since $N_{E_b} = N_{V_b}$, we obtain

$$(s-2)N_C = 2N_{V_i} + N_{V_b} + 2(N_B - 1) \tag{2.16}$$

which is a formula useful in relating the number of cells to the number of vertices on a two-dimensional grid. For instance, on a quadrilateral grid ($s = 4$), we have

$$2N_C = 2N_{V_i} + N_{V_b} + 2(N_B - 1) \tag{2.17}$$

which shows that the number of residuals is almost the same as the number of unknowns because we have typically $N_{V_b}, N_B \ll N_C, N_{V_i}$. Therefore, if we introduce the nodal coordinates as additional unknowns, we will have to introduce additional equations as well to move the grid. On the other hand, on a triangular grid ($s = 3$), we have

$$N_C = 2N_{V_i} + N_{V_b} + 2(N_B - 1) \tag{2.18}$$

which shows that there are almost twice as many triangles as the vertices. The excess of the number of residuals, however, depends on the problem we wish to solve. Suppose we wish to solve p differential equations for p unknowns. Then, the number of extra degrees of freedom available is given by

$$pN_C - pN_V = 2pN_{V_i} + pN_{V_b} + 2p(N_B - 1) - p(N_{V_i} + N_{V_b}) = pN_{V_i} + 2p(N_B - 1). \tag{2.19}$$

---

[2]Here we assume that the domain is divided into a set of polygons of the same type, e.g. a set of triangles. Therefore, for example, $s = 3$ for triangles, and $s = 4$ for quadrilaterals.

In two dimensions, the number of unknowns for the grid coordinates is two (x_j, y_j) at each vertex. Hence, we need $2N_{V_i}$ extra degrees of freedom at least. Clearly, this is possible only for $p \geq 2$. Therefore, the problem will be well overdetermined for any equations except for scalar ones ($p = 1$). For scalar equations, the problem will be underdetermined even with mesh movement. But at the same time this implies the possibility that residuals could be made to vanish. It turns out that the method works well for scalar equations without adding extra equations. For these reasons, and also for its great flexibility in grid generation as well as in local point insertion/removal that is a valuable option to the moving grid method, we choose triangular grids for the least-squares method.

In three dimensions, again we have Euler's formula for any compact three-dimensional object dissected by polyhedra,

$$N_V - N_E + N_F - N_C = 0 \tag{2.20}$$

where N_V, N_E, and N_F are the numbers of vertices, edges, faces, and cells respectively. Similarly to the 2D case, replacing N_C by $N_C + (N_B + 1)$ so that N_C now denotes the number of computational cells, we have

$$N_V - N_E + N_F - N_C = (N_B + 1) \tag{2.21}$$

for any three-dimensional computational grids. On the other hand, visiting the cells and marking faces, we obtain

$$2N_{F_i} + N_{F_b} = qN_C \tag{2.22}$$

where N_{F_i} and N_{F_b} are the numbers of faces in the interior and boundary of the grid respectively, and q is the number of faces of which each cell composed. Eliminating N_{F_i} from (2.21) and (2.22), we obtain

$$2N_V - 2N_E + (q-2)N_C + N_{F_b} = 2(N_B + 1). \tag{2.23}$$

For surface grids of inner and outer boundaries which are both assumed to be compact two-dimensional surface, we have

$$N_{V_b} - N_{E_b} + N_{F_b} = 2 \tag{2.24}$$

$$2N_{E_b} = sN_{F_b} \tag{2.25}$$

where s is determined by the type of elements employed to divide the surface ($s = 3$ for triangles, $s = 4$ for quadrilaterals, etc). Eliminating N_{E_b} from these equations, we get

$$(s-2)N_{F_b} = 2(N_{V_b} - 2). \tag{2.26}$$

Now, substituting (2.24) into (2.23) to eliminate N_{E_b}, we get

$$2N_{V_i} - 2N_{E_i} + (q-2)N_C + 4 - N_{F_b} = 2(N_B + 1) \tag{2.27}$$

from which N_{F_b} can be eliminated by using (2.26) to yield

$$(q-2)N_C = 2N_{E_i} - 2N_{V_i} + \frac{2}{s-2}(N_{V_b} - 2) + 2(N_B - 1). \tag{2.28}$$

Unfortunately, there is no general formula that determines explicitly N_{E_i} for a given N_{V_i} in three dimensions, except for a simple structured grid. For a structured grid composed of hexahedra ($q = 6$ and $s = 4$), without any geometric singularities, visiting the internal edges and marking vertices, we obtain

$$2N_{E_i} = 6N_{V_i} + N_{V_b}. \tag{2.29}$$

Then, the formula (2.28) becomes

$$4N_C = 4N_{V_i} + 2(N_{V_b} - 1) + 2(N_B - 1) \tag{2.30}$$

which shows that there are not enough degrees of freedom to move the grid. On the other hand, consider a grid generated by this hexahedral grid by dividing each

hexahedron into 5 or 6 tetrahedra without introducing additional vertices. We then have 5 or 6 times as many tetrahedra as the vertices because the number of vertices remains intact. If we have p differential equations with p unknowns, the excess of the number of residuals will be

$$pN_C - pN_V \approx 4pN_V \text{ or } 5pN_V. \tag{2.31}$$

This shows that we have enough degrees of freedom to move the grid even for a scalar equation ($p = 1$) since we wish to introduce three unknowns at each vertex (x_j, y_j, z_j). Therefore, tetrahedral grids are the candidates for the least-squares method in three dimensions. Similarly to triangular grids in two dimensions, tetrahedral grids offer an ease with node insertion/removal and geometric flexibility for generation, which leads to the conclusion that tetrahedral grids are suitable for least-squares moving grid methods.

2.3 The Choice of The Norm

The choice of the norm is of great importance to the least-squares method. It is the matrix Q_C that endows properties to the resulting numerical scheme and also to the movement of the grid. One important role of the matrix is to scale different equations in a system of equations. Consider Stokes' equations that govern slow motions of fluids in two dimensions.

$$\partial_x p - \mu \left(\partial_x^2 u + \partial_y^2 u \right) = 0 \tag{2.32}$$

$$\partial_y p - \mu \left(\partial_x^2 v + \partial_y^2 v \right) = 0 \tag{2.33}$$

$$\partial_x u + \partial_y v = 0 \tag{2.34}$$

To apply the least-squares method, we introduce vorticity ω as an additional unknown to reduce the system into a first-order system.

$$m_x = \partial_x p + \mu \partial_y \omega = 0, \quad m_y = \partial_y p - \mu \partial_x \omega = 0 \tag{2.35}$$

$$\lambda = \partial_x u + \partial_y v = 0, \quad \xi = \partial_x v - \partial_y u - \omega = 0. \tag{2.36}$$

Assuming on a triangular grid that all the variables vary linearly within each triangle we define the residual Φ_T as integrals of the equations

$$\Phi_T = [M_{xT}, M_{yT}, \Lambda_T, \Xi_T]^t \tag{2.37}$$

where the components expressed by upper-case letters denote the integral values of the lower-case counterparts over the triangle T, and the superscript t denotes transpose. It would be reasonable to weight the first two components equally, and the second ones also, but the pairs should be distinguished from each other on dimensional grounds. Roe suggested based on a simple dimensional analysis[80] that we set

$$Q_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k\frac{\mu^2}{S_T} & 0 \\ 0 & 0 & 0 & k\frac{\mu^2}{S_T} \end{bmatrix} \tag{2.38}$$

where k is an arbitrary constant, so that the norm is written compactly

$$\mathcal{F} = \frac{1}{2} \sum_{T \in \{T\}} \left[M_{xT}^2 + M_{yT}^2 + k\frac{\mu^2}{S_T} \left(\Lambda_T^2 + \Xi_T^2 \right) \right]. \tag{2.39}$$

This is a dimensionally correct norm. Our numerical experiments showed that the method produced a correct solution with this norm, but that wrong solutions could be found with other dimensionally inconsistent norms. Dimensional consideration

should be carefully given in defining the norm for a system of equations. The importance of weighting for least-squares methods has been pointed out by Zeitoun *et al.* [100] and also by Carey *et al.* [21].

Another important consideration is a volume weight. Consider the norm of the form

$$\mathcal{F} = \sum_{C\in\{C\}} F_C = \frac{1}{2} \sum_{C\in\{C\}} \frac{\Phi_C^t Q_C \Phi_C}{V_C} \tag{2.40}$$

where it has been assumed that Q_C does not contain V_C among its arguments. It may be perceived from (2.40) that minimizing this with respect to the grid points could involve maximizing the volume V_C. This turns out to be true; *de facto*, this offers a mechanism for eliminating small cell volumes. In spite of this favorable property, the norm without the volume weight will become important in some cases.

$$\mathcal{F} = \sum_{C\in\{C\}} F_C = \frac{1}{2} \sum_{C\in\{C\}} \Phi_C^t Q_C \Phi_C. \tag{2.41}$$

We will usually refer to (2.40) as the weighted norm, and to (2.41) as the unweighted norm. Note that the unweighted norm will remain positive whatever happens; the minimization proceeds. Mesh tangling, which is always a concern for any moving mesh algorithm, will never be noticed unless one actually looks at the mesh. This type of norm, although with such a disadvantage, is not without its advantages as we shall discuss in the chapter for elliptic problems in two dimensions.

2.4 Relationship to Other Numerical Schemes

As far as the solution method is concerned, the method belongs to a class of least-squares finite-element methods[48, 102, 18, 17] weighted by the matrix Q_C because we can write the norm as

$$\mathcal{F} = \frac{1}{2} \int R_C^t Q_C R_C \tag{2.42}$$

where the integral is all over the domain, and R_C is a discrete version of the differential equations we wish to solve, which is an alternative definition of the residual, typically defined by

$$R_C = \mathcal{L}(u_h) \tag{2.43}$$

where u_h is an approximate representation of the solution. If a piecewise linear function with triangular elements is chosen for u_h and moreover the equations are linear and first-order, it becomes equivalent to Φ_C/V_C. Thus we find readily

$$\frac{1}{2}\int R_C^t Q_C R_C = \frac{1}{2}\sum_{C\in\{C\}} \frac{\Phi_C^t Q_C \Phi_C}{V_C}. \tag{2.44}$$

It corresponds precisely with the weighted norm. The methods are therefore related very closely to one another. But they can be different for nonlinear problems where different kinds of linearizations may be employed. As pointed out earlier, however, the least-squares methods are seldom used for triangular grids because they are not accurate on such grids. But again we emphasize that this is not because the problem is overdetermined for triangles, for when grid movement is introduced, the problem is scarcely overdetermined, but the solutions are still poor. It is simply, we believe, because we do not minimize the right quantity.

Equations are associated with cells while unknowns are associated with vertices. Numerical schemes that employ this strategy belong to the class of methods called cell-vertex methods[27, 67] that includes residual distribution schemes (fluctuation splitting schemes) [84, 63, 78, 31], and the least-squares method. In cell-vertex methods, it is common that the number of equations is not always equal to the number of unknowns as shown in the last section, which is often called a counting problem. Therefore, instead of making the residuals vanish individually, the effort has been put on making some weighted sum of them vanish at each vertex. The

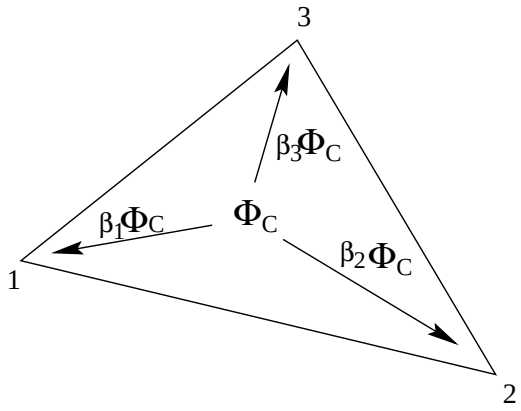

Figure 2.1: Residual distribution on a triangular element

choice of the weight is the central subject of the cell-vertex methods, and it has been extensively debated[31, 63, 78]. In the least-squares methods, the weight is the derivative of the norm with respect to a nodal solution, and its property is given through the residuals and the norm.

As a distribution scheme, the least-squares method can be coded as a simple double-loop process: loop over the elements computing the residuals and distribute them to the nodes, and then loop over the nodes updating the solutions and the grid points by the amount accumulated in the previous element loop. The distribution process in two dimensions is depicted in Figure 2.1, which is a typical picture found in the literature on the distribution schemes, where β_i is the distribution coefficient that determines the fraction of the residual sent to a particular node. In the least-squares method with the unweighted norm, we have

$$\beta_i = \left(Q_C \frac{\partial \mathbf{\Phi}_C}{\partial \mathbf{u}_i} \right)^t \tag{2.45}$$

and similarly for the grid coordinate

$$\beta_i = \left(Q_C \frac{\partial \mathbf{\Phi}_C}{\partial \mathbf{x}_i} \right)^t . \tag{2.46}$$

After the element loop, we complete the computation of the gradients (2.6) and (2.7) at each node, and the solution and the grid coordinate can be updated by using these

gradients making a loop over the node. Note that we have typically

$$\sum_{i=\{1,2,3\}} \beta_i = 0 \tag{2.47}$$

for residuals linearized in the cell, so that the resulting scheme will be conservative, although not in the usual sense, and for the grid coordinate this means that the centroid of the element does not move.

2.5 Implementation

The simplest method to minimize the norm would be a steepest descent method. The changes made to vertex j that minimize the weighted norm (2.40) are given by

$$\delta \mathbf{u}_j = -\omega_u \frac{\partial \mathcal{F}}{\partial \mathbf{u}_j} = -\omega_u \sum_{C \in \{C_j\}} \frac{\partial F_C}{\partial \mathbf{u}_j} = -\omega_u \sum_{C \in \{C_j\}} \frac{\Phi_c^t}{V_C} \left(Q_C \frac{\partial \Phi_C}{\partial \mathbf{u}_j} \right) \tag{2.48}$$

for the solution vector, and

$$\delta \mathbf{x}_j = -\omega_x \frac{\partial \mathcal{F}}{\partial \mathbf{x}_j} = -\omega_x \sum_{C \in \{C_j\}} \frac{\partial F_C}{\partial \mathbf{x}_j} = -\omega_x \sum_{C \in \{C_j\}} \left\{ \frac{\Phi_c^t}{V_C} \left(Q_C \frac{\partial \Phi_C}{\partial \mathbf{x}_j} \right) - \frac{F_C}{V_C} \frac{\partial V_C}{\partial \mathbf{x}_j} \right\} \tag{2.49}$$

for the position vector of the vertex, where ω_u and ω_x are small constants, $\{C_j\}$ is a group of cells that share the vertex j, and it has been assumed that the norm is defined by (2.40) and that Q_C is frozen during the minimization. It is important to remember that when taking the derivative of the residual, any coefficients derived from nonlinearity should be frozen, so that we deal with a linear problem with respect to the solution values at each iteration step[48]. We note in passing that the change in $\mathbf{x}_j$ has an additional term, and that this term is in the direction of increasing cell volumes with larger weights given to smaller cells, thus creating a mechanism by which cells can compete for volume.

It is well-known that the steepest descent method converges very slowly. One reason is its dimensional inconsistency. For instance, in (2.49), it is seen that the

right hand side does not have the dimension of length while the left hand side clearly does. The use of Newton's method cures this dimensional inconsistency and also speeds up the convergence. But instead of inverting the Hessian matrix (the second derivative of the norm), we may use only the diagonal elements for simplicity which makes the inversion extremely simple. Hence each change will be divided by the second derivative of the norm with respect to the corresponding variable. For a component u_j of $\mathbf{u}_j$, we have

$$\delta u_j = -\omega_u \frac{\partial \mathcal{F}}{\partial u_j} \Big/ \frac{\partial^2 \mathcal{F}}{\partial u_j^2} \tag{2.50}$$

and for a component x_j of $\mathbf{x}_j$,

$$\delta x_j = -\omega_x \frac{\partial \mathcal{F}}{\partial x_j} \Big/ \frac{\partial^2 \mathcal{F}}{\partial x_j^2} \tag{2.51}$$

where

$$\frac{\partial^2 \mathcal{F}}{\partial u_j^2} = \sum_{C \in \{C_j\}} \frac{1}{V_C} \frac{\partial \Phi_C^t}{\partial u_j} \left(Q_C \frac{\partial \Phi_C}{\partial u_j} \right) \tag{2.52}$$

$$\frac{\partial^2 \mathcal{F}}{\partial x_j^2} = \sum_{C \in \{C_j\}} \frac{1}{V_C} \left\{ \frac{\partial \Phi_C^t}{\partial x_j} \left(Q_C \frac{\partial \Phi_C}{\partial x_j} \right) - 2 \frac{\partial F_C}{\partial x_j} \frac{\partial V_C}{\partial x_j} \right\}. \tag{2.53}$$

Observe that the equalities (2.50) and (2.51) are now dimensionally consistent. Another way to improve the convergence is to apply sequential updates such as a Gauss-Seidel type iteration[91]. Since this makes the grid perturbation completely local, it provides the opportunity to control the mesh validity at each iteration, which is actually a usual practice for Laplacian smoothing techniques for unstructured grid generation methods. That is to say, we check, before we actually move the node, if the movement of a node creates mesh folding, and if it does, reject the movement. Upwind relaxation proposed by Baines and Leary[55, 7] is another effective strategy to accelerate the convergence where among the set of cells $\{C_j\}$ only those in the

upwind side contribute to the updates at node j. In this thesis, however, we will not consider this upwind strategy, and use only the diagonal scaling and the Gauss-Seidel iteration mentioned above, focusing on what we can achieve rather than how fast we can get there.

So far, it has been implicitly assumed that the grid connectivity is preserved, but the initial connectivity might not be satisfactory. The final mesh would strongly depend on the initial grid since nodal movement is limited by the fixed connectivity. On triangular grids, there is a simple way to change the connectivity called diagonal swapping: a well-known trick used to construct Delaunay triangulations[9]. This technique can be easily incorporated into the minimization scheme, that is, for a pair of adjacent triangles that forms a quadrilateral, the diagonal is replaced whenever the other would reduce the norm (Figure 2.2). But it is not always possible in general. A pair of triangles may exists that form a concave quadrilateral as shown in Figure 2.3. In such a case, swapping the diagonal will introduce two overlapping triangles. To avoid this situation, it must be checked whether the swapping is feasible or not for each diagonal in consideration. The check can be made by computing the area of each triangle created by the edge swapping: if either one has a negative area, the swapping is not feasible. As a general procedure, we first search for the candidate edges by checking the feasibility and then comparing the sum of F_T's of two triangles that share an edge with that of the other possible pair of the triangles. Then, we put them in order of large reduction of the norm and start the swapping in the modified order, following Tourigny and Hulsemann who developed a moving grid algorithm for second-order elliptic equations in the form of variational minimization[91]. However every time we swap one diagonal, the rest of the edges in the list might yield larger norms if the edges are in the neighborhood of the already swapped edge. Therefore

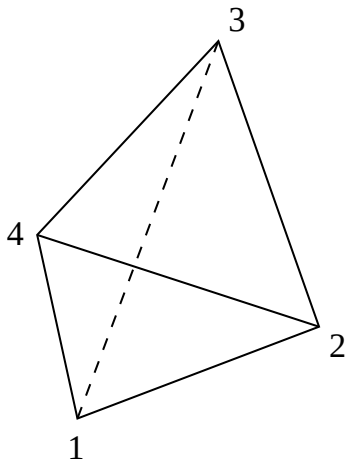

Figure 2.2: A pair of triangles for which diagonal swapping is feasible. The dashed line indicates the alternative.

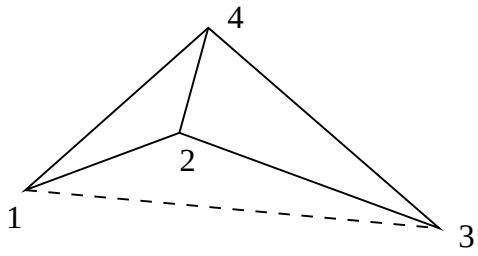

Figure 2.3: A pair of triangles for which diagonal swapping is NOT feasible. The dashed line indicates the alternative that violates the validity of the grid.

it is necessary to check the effectiveness of the swapping again before an edge is actually swapped. In three dimensions, face swapping can be utilized in a similar manner[9].

The flexibility of triangular and tetrahedral grids makes it easy to insert or remove nodes. This is also incorporated into the minimization scheme easily; a large norm motivates node insertion; a small norm motivates node removal. In this thesis, node insertion will not be considered because the interest here is to see how much the numerical solution can be improved by grid movement; it is quite obvious that adding more nodes improves the accuracy. But it becomes necessary to implement node removal technique in some cases, and it turned out that node removal simply based on the norm reduction did not work well. This issue will be addressed when we discuss hyperbolic equations that involves discontinuous solutions.

CHAPTER III

PROBLEMS IN ONE DIMENSION

In this chapter, we describe the least-squares moving grid method for a simple boundary value problem. It is demonstrated that in one dimension including grid points as unknowns always results an underdetermined problem, thus leading to nonunique solutions, but also that equidistribution of a certain quantity, which provides additional equations, yields satisfactory solutions for the problem. Of particular interest is a geometrical interpretation of the least-squares method: the method attempts to construct a numerical approximation to a geometrical object that represents the solution. We discuss via geometrical analogy what it means to include the node positions among the variables.

3.1 The Least-Squares Method for Friedrichs' Model

We consider the following second-order ordinary differential equation,

$$\epsilon \frac{d^2u}{dy^2} + \frac{du}{dy} - a = 0 \tag{3.1}$$

with

$$u(0) = 0, \quad u(1) = 1 \tag{3.2}$$

which was proposed by Friedrichs[94] as a model of the boundary-layer momentum equations with rough resemblances, $\epsilon = \frac{1}{Re}$ where Re is the Reynolds number and

$a(< 1)$ modeling the pressure gradient. The exact solution is given by

$$u(y) = ay + (1-a)\frac{1 - exp(-y/\epsilon)}{1 - exp(-1/\epsilon)} \quad . \tag{3.3}$$

This solution exhibits a severe transition region near $y = 0$ whose width is proportional to ϵ. Our expectation for the least-squares method is that it will automatically move the nodes of a computational grid into the boundary layer for better resolution. To apply the least-squares method, we first need to reduce the equation to a first-order system by introducing $J = \frac{du}{dy}$ as an additional unknown. We thus have

$$\epsilon\frac{dJ}{dy} + \frac{du}{dy} - a = 0 \tag{3.4}$$

$$\frac{du}{dy} - J = 0 \tag{3.5}$$

We begin by dividing the domain $0 \leq y \leq 1$ into a set $\{E\}$ of elements(segments) each of which is defined by two consecutive nodes denoted by L and R where L is the node closer to $y = 0$. Within each element, assuming that u and J vary linearly, we define the residuals as integral values of the above equations over the element.

$$\Phi_E = \epsilon\Delta J_E + \Delta u_E - a\,\Delta y_E \tag{3.6}$$

$$\Psi_E = \Delta u_E - \bar{J}_E\,\Delta y_E \tag{3.7}$$

where Δu_E is the difference of u between the right and left nodes: $u_R - u_L$, and $\bar{J}_E$ denotes a value of J at the midpoint of the edge E. Now, observe that the number of unknowns is $3N_V$ and the number of residuals is $2N_E$ where N_V is the number of nodes and N_E is the number of elements. Therefore there are more unknowns than equations. In fact, this is a valid general statement in one dimension as discussed in Chapter II. Suppose there are p differential equations for p dependent variables. Then, including grid points as additional unknowns, the excess of the unknowns N_{ex}

become

$$N_{ex} = (p+1)N_V - pN_E. \tag{3.8}$$

For one-dimensional grid, we have $N_E = N_V - 1$. Thus

$$N_{ex} = N_V + p. \tag{3.9}$$

It follows from this that the problem is always underdetermined if the grid points are included among the variables. In our case, $p = 2$,

$$N_{ex} = N_V + 2. \tag{3.10}$$

Giving the boundary conditions, which are 4 conditions: $(u, y) = (0, 0)$ and $(u, y) = (1, 1)$, and writing the number of interior vertices N_{Vi}, we obtain

$$N_{ex} = N_{Vi}. \tag{3.11}$$

Therefore the problem is underdetermined by N_{Vi} conditions, which in turn implies that the solution will not be unique. But at the same time, this suggests that an extra condition may be imposed on the solution or the grid at each vertex. This is discussed in a later section.

The next step is to define the norm to be minimized. The choice of the norm is the central subject of the least-squares method, and can greatly affect the resulting solution. This means conversely that one might aim for solutions with a desired property by defining a suitable norm. We will return to this point in later sections. In this study, we define the following weighted norm.

$$\mathcal{F} = \sum_{E\in\{E\}} F_E = \sum_{E\in\{E\}} \frac{1}{2\Delta y_E}\left[\Phi_E^2 + \Psi_E^2\right]. \tag{3.12}$$

The weight $1/\Delta y_E$ has been introduced in order to penalize small cells, and hence tends to prevent Δy_E from becoming negative or zero, otherwise $u(y)$ and $J(y)$ will

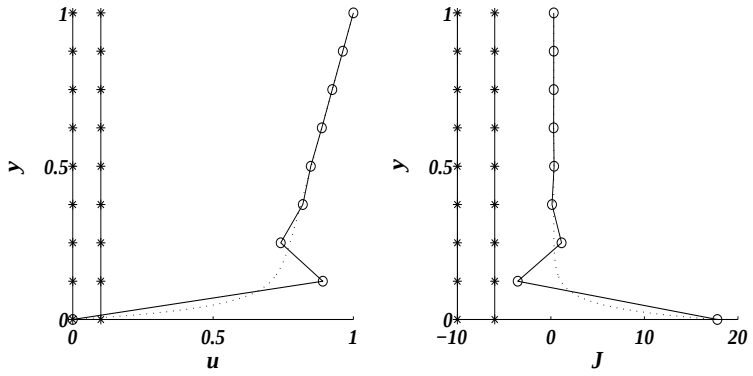

Figure 3.1: The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes.

become multi-valued. Now, the residuals being nonlinear, the norm is not in general quadratic in the unknowns including y. Therefore some iterative scheme must be employed to carry out the minimization. The simplest method would be a steepest descent method.

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - c\,\frac{\partial \mathcal{F}}{\partial \mathbf{U}_j} \tag{3.13}$$

where $\mathbf{U}_j = (u_j, J_j, y_j)$ and c is a positive constant which we call a relaxation factor. In order to see how this simple method works, we present a few results here. In computing all the results here, we used the steepest descent technique with a Gauss-Seidel type procedure and a diagonal scaling as described in Chapter II. Figure 3.1 shows the least-squares solutions obtained on a fixed grid of eight elements. The solutions are oscillatory in the boundary layer because of the lack of resolution. And moving the grid does not improve the solutions at all because the residuals have already been driven to zero (the least-squares solutions on a fixed grid is unique). In other words, the minimization problem with grid movement is highly underdetermined, and the norm does have an undesired minimum. Shown in Figures 3.2 is another set least-squares solutions obtained from initial solutions u and J that are

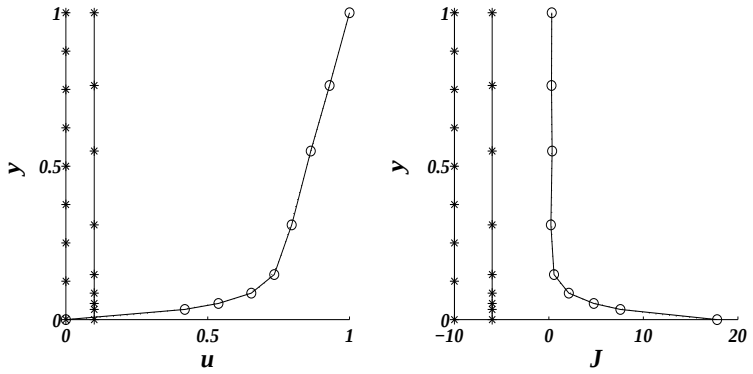

Figure 3.2: The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes.

both linear between the two boundary values 0 and 1. The method converged for 362 iterations with the relaxation factor 1.0. The solutions are much better than the previous ones. It is seen that some vertices have been moved into the boundary layer region, thereby suppressing the oscillation that would have been created on the fixed grid. But at the same time this proves the nonuniqueness of the solution as well: the solutions depend highly on the initial solutions. Figure 3.3 shows solutions with 64 elements, obtained from the same linear initial solution. We found experimentally that the relaxation factor must be reduced as the number of elements increase, and therefore the relaxation factor was set 0.7 in this case which resulted 66963 iterations for convergence. It is seen in the figure that the method does not move the nodes deeply enough into the boundary layer while placing many unnecessary nodes in the linear portion. The L_2 error, defined by $\sum_{nodes} \sqrt{(u_{exact} - u_{numerical})^2 / N_{Vi}}$, of $u(y)$ is 9.69E-04 while the L_2 error of the solution on the fixed grid of the same size is 1.05E-03. Therefore the solution has not been improved very much. Also considering the fact that the error for the case of 8 edges is already 4.89E-03, we see that the convergence to the exact solution is extremely slow. It is, of course, possible to improve the solution by choosing a better initial solution. However, this is not a

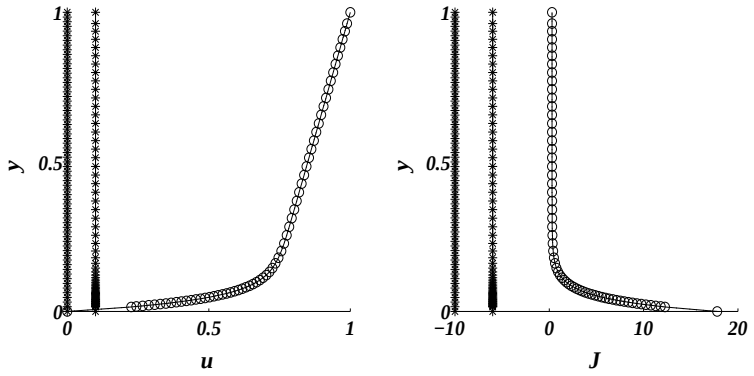

Figure 3.3: The numerical and the exact solutions for u and J indicated by circles connected by line segments and dotted curves respectively. $*$ indicates the initial and the final positions of the nodes.

meaningful strategy because choosing a good initial curve demands the knowledge of the exact solution. Since we know that the discrete problem is underdetermined, the simplest strategy would be to introduce additional equations, possibly giving an overdetermined problem, which impose some extra conditions on the least-squares solutions. We will return to this point later.

3.2 A Geometric Interpretation

In the previous section, it was shown that the least-squares method automatically moved the nodes into the boundary layer for better resolution, although depending on initial solutions. But we did not know why it did. It is the purpose of this section to understand what it does, and what it means to treat the grid points as unknowns. It is a geometrical viewpoint that sheds a light on these questions. That is, we abandon the common thought that we are computing u and J on a computational grid laid along y axis. Instead, we consider the problem as a geometric approximation: approximation of a solution curve in three-dimensional space (u, J, y). In order to illustrate its geometrical nature, the presentation will be given in geometrical terms. The mathematics suitable for discussing the geometry of differential equations is the

calculus of differential forms, also known as the exterior calculus. For the details of the theory, the interested reader is referred to Edwards[34] which is readable and a good introduction to the subject, and Schutz[86] for more advanced topics.

3.2.1 Geometric Solution of Friedrichs' Model

In order to discuss the geometrical property of Friedrichs' model, we shall derive the exact solution using the calculus of differential forms. In the geometric formulation, differential equations are expressed as a system of differential forms(exterior system). The solution to the system is then a vector field which annuls all the differential forms in the system. There are basically two methods to find the solution. We shall demonstrate these here for Friedrichs' model. To define the system of the differential forms equivalent to (3.1), we first reduce the equation to the first-order system by introducing $J = \frac{du}{dy}$ as an additional unknown. We thus have

$$\epsilon \frac{dJ}{dy} + \frac{du}{dy} - a = 0 \tag{3.14}$$

$$\frac{du}{dy} - J = 0 \quad , \tag{3.15}$$

rewritten as

$$\epsilon dJ + du - a\, dy = 0 \tag{3.16}$$

$$du - J\, dy = 0 \tag{3.17}$$

On a three-dimensional manifold with the coordinates (u, J, y), denoted by $\mathbf{R}^3$, we define two one-forms, α_1 and α_2, by

$$\alpha_1 = \epsilon\, \tilde{d}J + \tilde{d}u - a\, \tilde{d}y \tag{3.18}$$

$$\alpha_2 = \tilde{d}u - J\, \tilde{d}y \tag{3.19}$$

where $\tilde{d}u$, $\tilde{d}J$ and $\tilde{d}y$ are now one-forms. One-forms are in fact what are called covariant vectors, such as gradients, in tensor analysis. Therefore one may think

of these expressions as defining two covariant vectors $(1, \epsilon, -a)$ and $(1, 0, -J)$ for the basis $\{\tilde{d}u, \tilde{d}J, \tilde{d}y\}$. The solution is then a tangent vector(contravariant vector) which annuls these two one-forms. In other words, the solution is a vector that is perpendicular to the both covariant vectors. The solution submanifold is then the integral curve of the vector field. A question may arise as to whether the problem is equivalent to the original problem. The equivalence to the original problem may be verified by sectioning the forms into a solution submanifold. The sectioning is done by substituting $\tilde{d}u = u_y \tilde{d}y$ and $\tilde{d}J = J_y \tilde{d}y$ into the above equations and then requiring the components of the resulted forms to vanish[1]. In order to assert the equivalence it still remains to show that the set of forms is closed. A closed set is defined as a set whose elements generate an ideal[2] and the exterior derivatives of the elements are also in the ideal. It is easy to show that a set of forms $\{\beta_i\}$ is closed if exterior derivatives of the forms can be written as a linear combination of $\{\beta_i\}$, i.e. $\sum \gamma_i \wedge \beta_i$ where $\wedge$ denotes wedge product[86] and γ_i is any form. In fact we find

$$\tilde{d}\alpha_1 = 0 \tag{3.20}$$

$$\tilde{d}\alpha_2 = -\tilde{d}J \wedge \tilde{d}y = \frac{1}{\epsilon}\tilde{d}y \wedge \alpha_1 - \frac{1}{\epsilon}\tilde{d}y \wedge \alpha_2 \quad . \tag{3.21}$$

Therefore they constitute a closed ideal of differential forms[3]. The Frobenius' theorem[86] then guarantees the existence of the solution submanifold of the forms, i.e. any local surface element that annul α_1 and α_2 will fit together to form a solution submanifold. Now we describe how to find the solution submanifold.

---

[1]The solution submanifold is represented here by functions $(u(y), J(y))$ satisfying (3.2) with y as a parameter.

[2]A set of differential forms defines a tangent vector space as its dual space. An ideal is all the forms whose restriction to the tangent vector space vanishes.

[3]This is in fact trivial in the present case since any two linearly independent one-forms in a three-dimensional manifold automatically constitute a closed ideal.

One approach to the exact solution is to find an analytic expression of the tangent vector field whose integral curve becomes the solution submanifold. In general, a tangent vector V in $\mathbf{R}^3$ can be written as

$$V = \frac{d}{d\lambda} = \frac{du}{d\lambda}\frac{\partial}{\partial u} + \frac{dJ}{d\lambda}\frac{\partial}{\partial J} + \frac{dy}{d\lambda}\frac{\partial}{\partial y} \tag{3.22}$$

where λ is a parameter of a curve and $\frac{du}{d\lambda}, \frac{dJ}{d\lambda}$, and $\frac{dy}{d\lambda}$ are the components of V for the basis $\{\frac{\partial}{\partial u}, \frac{\partial}{\partial J}, \frac{\partial}{\partial y}\}$ which is dual to the one-form basis, $\{\tilde{du}, \tilde{dJ}, \tilde{dy}\}$. The components are found by requiring α_1 and α_2 to vanish when contracted with (3.22), which we write

$$\alpha_1(V) = \epsilon\,\tilde{dJ}(V) + \tilde{du}(V) - a\,\tilde{dy}(V) = 0 \tag{3.23}$$

$$\alpha_2(V) = \tilde{du}(V) - J\,\tilde{dy}(V) = 0 \ , \tag{3.24}$$

and we obtain

$$\epsilon\frac{dJ}{d\lambda} + \frac{du}{d\lambda} - a\frac{dy}{d\lambda} = 0 \tag{3.25}$$

$$\frac{du}{d\lambda} - J\frac{dy}{d\lambda} = 0 \ . \tag{3.26}$$

These are two linear equations for the components, and thus the tangent vector is determined up to a length. A quick algebra shows that

$$\left(\frac{du}{d\lambda}, \frac{dJ}{d\lambda}, \frac{dy}{d\lambda}\right) = p\,\left(J, -\frac{J-a}{\epsilon}, 1\right) \tag{3.27}$$

where p is any function of u, J, y or a constant. Then the solution can be found by solving the system of ordinary differential equations given by

$$\frac{du}{d\lambda} = J, \ \ \frac{dJ}{d\lambda} = -\frac{J-a}{\epsilon}, \ \ \frac{dy}{d\lambda} = 1 \ . \tag{3.28}$$

This method therefore leads us to the original system of ordinary differential equations by choosing $\lambda = y$.

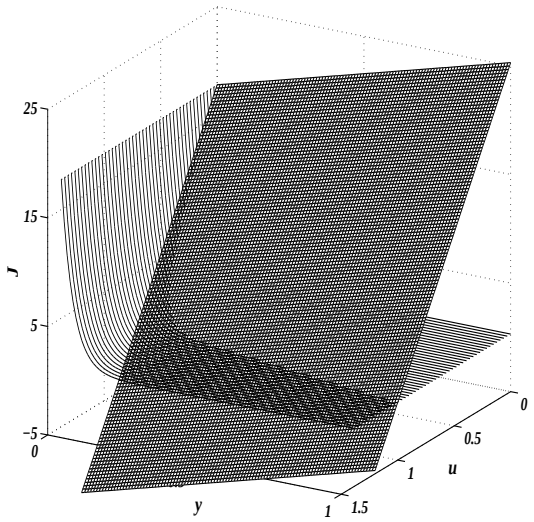

Figure 3.4: A geometric representation of the exact solution of the Friedrichs' model for a=0.3 and $\epsilon = 0.04$. The plane is defined by $\epsilon\, J + u - ay = constant$, and the curved surface is defined $y + \epsilon\, ln(J - a) = constant$ where the constants have been determined by the boundary conditions in (3.2).

The other approach gives another interesting view of the solution curve. The Frobenius' theorem is now stated more precisely for Friedrichs' model as follows. If and only if the one-forms, α_1 and α_2, are closed, there exist functions h, k, l, m, f, and g such that

$$\alpha_1 = h\,\tilde{d}f + k\,\tilde{d}g \tag{3.29}$$

$$\alpha_2 = l\,\tilde{d}f + m\,\tilde{d}g. \tag{3.30}$$

Then the solution is a one-dimensional submanifold defined by $f = constant$ and $g = constant$[4]. In general, it is however not easy and often impossible to find such functions that are valid globally. The theorem in fact guarantees only the local

[4]Each function defines a family of surfaces in $\mathbf{R}^3$. And the intersections are the general solution curve. The functions k, l, and m are arbitrary and have no meaning. The point is that the differential forms can be written as a combination of two total differentials, so that if these functions are constant, they vanish.

integrability of the exterior system. For Friedrichs' model, however, we find

$$\alpha_1 = \tilde{d}(\epsilon J + u - ay) \tag{3.31}$$

$$\alpha_2 = \tilde{d}(\epsilon J + u - ay) - (J - a)\,\tilde{d}(y + \epsilon\, ln(J - a)). \tag{3.32}$$

Therefore the solution submanifold is given by

$$\epsilon J + u - ay = constant \tag{3.33}$$

$$y + \epsilon\, ln(J - a) = constant, \tag{3.34}$$

i.e. each of which defines a family of surfaces in $\mathbf{R}^3$, and the set of their intersections is understood as the general solution submanifold. The solution of Friedrichs' model is one of the intersections which satisfies the boundary conditions (3.2). See Figure 3.4. It turns out that this unique representation of the solution reveals a property of the least-squares method.

3.2.2 Discretization

We will approximate the solution curve by a piecewise linear curve which consists of a set of linear edges $\{E\}$ and a set of vertices $\{V\}$ where every vertex is incident to exactly two distinct edges, i.e. the number of vertices = the number of edges + 1. The orientation of the curve may be defined as positive in the direction from $(u, J, y) = (0, J, 0)$ to $(u, J, y) = (1, J, 1)$. The properties of the solution submanifold is that its tangent vectors V annul the one-forms α_1 and α_2 everywhere on it, which is again written

$$\alpha_1(V) = \epsilon\tilde{d}J(V) + \tilde{d}u(V) - a\,\tilde{d}y(V) = 0 \tag{3.35}$$

$$\alpha_2(V) = \tilde{d}u(V) - J\,\tilde{d}y(V) = 0\ . \tag{3.36}$$

We remark here that the tangent space is defined at a point on the submanifold and therefore there exist infinite tangent spaces on the exact submanifold. In our

approximate submanifold, we have a finite number of tangent spaces each of which is defined by the edge, $V_E = (\Delta u_E, \Delta J_E, \Delta y_E)$ where Δu_E denotes the projected length of the edge E onto u-axis and similarly for others. On the assumption that the tangent vector and the one-forms are defined at the midpoint of the edge, we obtain the discrete versions of (3.35) and (A.10),

$$\Phi_E = \alpha_1(V_E) = \epsilon \Delta J_E + \Delta u_E - a\, \Delta y_E = 0 \tag{3.37}$$

$$\Psi_E = \alpha_2(V_E) = \Delta u_E - \bar{J}_E\, \Delta y_E = 0 \tag{3.38}$$

where $\bar{J}_E$ denotes a value of J at the midpoint of the edge E. These are identical to (3.6) and (3.7). This shows that we may interpret the residual as a geometrical error, i.e. an error in aligning a tangent vector along the solution manifold.

3.2.3 The Least-Squares Method

The least-squares method is formulated in exactly the same manner as before. But our viewpoint is now geometrical: construct an approximate curve by minimizing the residuals, in a least-squares norm, with respect to the the positions of the vertices in $\mathbf{R}^3$. Consider the change made to each vertex by the steepest descent method.

$$\delta \mathbf{U}_j^n = -c\, \frac{\partial \mathcal{F}}{\partial \mathbf{U}_j}. \tag{3.39}$$

The gradient is given by

$$\frac{\partial \mathcal{F}}{\partial \mathbf{U}_j} = \begin{bmatrix} (\Phi_L + \Psi_L)/\Delta y_L - (\Phi_R + \Psi_R)/\Delta y_R \\ \epsilon \left(\frac{\Phi_L}{\Delta y_L} - \frac{\Phi_R}{\Delta y_R} \right) - \frac{1}{2} (\Psi_L + \Psi_R) \\ \frac{\bar{J}_R}{\Delta y_R} \Psi_R - \frac{\bar{J}_L}{\Delta y_L} \Psi_L + a \left(\frac{\Phi_R}{\Delta y_R} - \frac{\Phi_L}{\Delta y_L} \right) + \frac{F_R}{\Delta y_R} - \frac{F_L}{\Delta y_L} \end{bmatrix} \tag{3.40}$$

where the subscripts L and R again indicate the edges on the left and right of the vertex j and the right means the positive direction of the approximate curve. The method is usually understood as updating the solution $\mathbf{U}_j^n$ in the direction of

steepest descent. This interpretation is global in nature, and does not give any useful information as to what the formula tries to do locally. In fact, an interesting property is hidden in the gradient, which can be seen clearly by decomposing the gradients into four parts. The change made to each vertex is then written, with $c = 1$,

$$\delta \mathbf{U}_j^n = -(c_1+c_2)\begin{bmatrix} 1 \\ \epsilon \\ -a \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ \epsilon \\ J_j - a \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 0 \\ -\left(\Delta y_L \frac{\Psi_L}{\Delta y_L} + \Delta y_R \frac{\Psi_R}{\Delta y_R}\right) \\ \left(\Delta J_L \frac{\Psi_L}{\Delta y_L} + \Delta J_R \frac{\Psi_R}{\Delta y_R}\right) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{F_L}{\Delta y_L} - \frac{F_R}{\Delta y_R} \end{bmatrix}. \tag{3.41}$$

where

$$c_1 = \left(\frac{\Phi_L}{\Delta y_L} - \frac{\Phi_R}{\Delta y_R}\right), \quad c_2 = \left(\frac{\Psi_L}{\Delta y_L} - \frac{\Psi_R}{\Delta y_R}\right). \tag{3.42}$$

It is now observed that each component indicates a particular direction in which vertex j is updated. The first and the second component will move the vertex j in the direction normal to the surfaces defined by $\epsilon\, J + u - ay = \mathit{constant}$ and $y + \epsilon\, ln(J - a) = \mathit{constant}$, respectively. And updates in these directions will cease when c_1 and c_2 reduce to zero, i.e. when the weighted residuals are equidistributed over the edges. Here it is important to note that such analytical functions of solution surfaces are not in general available as mentioned in Section 1. Therefore a more general interpretation would be that the first two components move the vertex in the directions of the two one-forms, α_1 and α_2, which can be clearly seen by rearranging the first two components in the form $-c_1[\,] - c_2[\,]$. The interpretation of the third component is not so obvious. However, it can be shown that when c_2 vanishes, it moves a vertex such that the projected area onto (J, y)-plane of the triangle formed by the two adjacent edges is minimized locally until the residuals Ψ_L and Ψ_R are driven to zero. Besides, we see that this projected area is nothing but the value of two-form (3.21) contracted with the two tangent vectors V_L and V_R. Now recall

that the integrability condition states that the exterior derivatives must be also in the ideal, i.e. must vanish also on the solution submanifold. On the exact solution submanifold, this is automatically true because any two-form is identically zero on a one-dimensional manifold. However on a discrete curve it cannot be true in general because two adjacent edges are very likely to form a finite angle at the vertex that they share. In effect, the third component locally attempts to minimize the values of two-form (3.21) at the vertices. The last term has been produced by the weight. It can be seen that this will limit the vertex movement such that no edge vanishes when projected onto y-axis. These are quite accurate pieces of information about the exact solution, and thus should be preserved. This suggests that we must use the same relaxation factor for all variables.

Let us look again at the results presented before, but geometrically this time. In Figure 3.5, we see that the oscillatory solution curve lies on one of the surface $\epsilon\, J + u - ay = constant$ but not on the other. This seems to suggest that the oscillation is caused by the second equation, and that it could be satisfied in many different ways apart from the exact orientation. In Figure 3.6, the trajectories of the vertices are plotted (at every 5 iterations). The reason that some points moved into the boundary layer is the movement of the vertices perpendicular to the surface $\epsilon\, J + u - ay = constant$. A close look at the geometrical solution will reveal that any vertex near the boundary with $u = y = 0$ is moved away from the boundary no matter how close they are. This explains why many nodes are left outside the boundary layer in the case of 64 edges as shown in Figure 3.7.

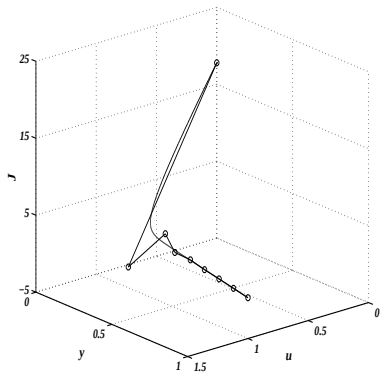

Figure 3.5: The solution from an oscillatory initial solution and the exact solution as the dotted curve. 8 edges.

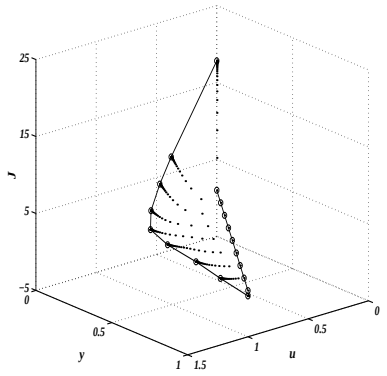

Figure 3.6: The solution from a straight initial solution and the trajectories of the vertices(dots). 8 edges.

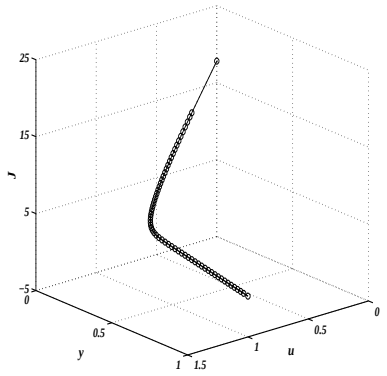

Figure 3.7: The solution from a straight initial solution. 64 edges.

3.3 Additional Property

It is important, when additional terms are added to the norm, to ensure the dimensional consistency. Also it is desired to define the additional quantity based on edges rather than vertices in order to avoid introducing additional complexity in the norm. Under these requirements, in this section we seek an additional quantity to be minimized which will make the solution as smooth as possible to exclude the minima with oscillatory solutions, and also by which a reasonable rate of convergence will be realized.

We begin with a rather simple argument. It is obvious that a small Δy_E is necessary for the element with large variations in u and J. Therefore, the equidistribution of $\Delta t_E \Delta J_E$, which is dimensionally consistent with the residuals, may be a good strategy. Returning to the original viewpoint that the solution u is a function of y, we first notice that

$$\Delta y_E \Delta J_E \approx {\Delta y_E}^2 \frac{dJ}{dy} \approx {\Delta y_E}^2 \frac{d^2 u}{dy^2}. \tag{3.43}$$

This means that the equidistribution of this quantity has the effect of concentrating the grid points in regions of high-curvature of $u(y)$. This is a desired property.

Although specific to Friedrichs' model, this takes into account the second derivative of J as well. On the assumption that J satisfies the differential equation (4), we can replace the first derivative of J by the first derivative to get

$$\Delta y_E \Delta J_E \approx -\epsilon {\Delta y_E}^2 \frac{d^2 J}{dy^2}, \tag{3.44}$$

which proves the statement. Also note that the right hand side is proportional to the local truncation error of Ψ_E. Hence the equidistribution of $\Delta y_E \Delta J_E$ is equivalent to that of the truncation error. This is, however, also specific to Friedrichs' model.

Another interpretation of $\Delta y_E \, \Delta J_E$ is based on the reconstruction of the solution $u(y)$. For our piecewise linear approximation, $u(y)$ is represented by a linear Lagrange interpolating polynomial $u_L(y)$ within each element Δy_E. Also a higher order interpolation is possible using the values of J at the vertices, i.e. a cubic Hermite interpolating polynomial of u, denoted by $u_H(y)$. Then it may be legitimate to estimate the error of $u(y)$ by the difference between $u_L(y)$ and $u_H(y)$. In fact it can be shown that

$$\frac{1}{\Delta y_E} \int_E \left(u_H(y) - u_L(y)\right)^2 dy = \frac{1}{120} {\Delta y_E}^2 {\Delta J_E}^2 + \frac{1}{210} \left(\Delta u_E - \bar{J}_E \, \Delta y_E\right)^2 \qquad (3.45)$$

where the second term is identical to ${\Psi_E}^2$ and therefore is already to be minimized. It is easy to see that the first term cannot be driven to zero in general unless Δy_E is zero. Therefore, it is reasonable to equidistribute ${\Delta y_E}^2 {\Delta J_E}^2$ so that exceptionally large local errors are reduced. In addition, if the solution is convex, the equidistribution of $\Delta y_E \, \Delta J_E$ will suffice, and this is in fact the case for Friedrichs' model. Yet another interpretation is that it is a leading term of the L_2 error of the piecewise linear approximation. This will be discussed in great detail in Appendix A.

The equidistribution of this quantity can be carried out, together with the minimization of the residuals, by minimizing the norm,

$$\mathcal{F} = \sum_{E \in \{E\}} F_E = \sum_{E \in \{E\}} \frac{1}{2\Delta y_E} \left[\Phi_E^2 + \Psi_E^2 + \Omega_E^2\right] \qquad (3.46)$$

where

$$\Omega_E = \Delta y_E \, \Delta J_E - \overline{\Delta y \, \Delta J} \qquad (3.47)$$

and $\overline{\Delta y \, \Delta J}$ is the average value of $\Delta y_E \, \Delta J_E$ over the edges which is to be recomputed at each iteration. In practice, we minimize Ω_E with respect to only interior vertices to keep physically meaningful boundary values of J.

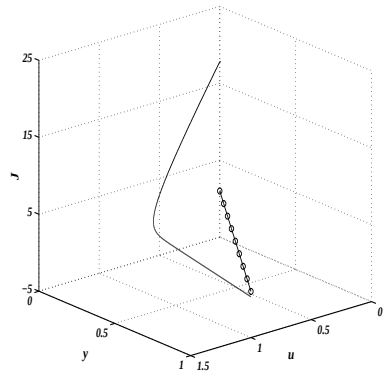

Figure 3.8: The initial approximation and the exact solution (the sold line).

3.4 Numerical experiments

For all the computations presented here, we employed the same strategies: descent technique with a Gauss-Seidel type procedure and a diagonal scaling. As for the stopping criterion, the gradient of the norm will be a reasonable choice. The iteration will be terminated when all the L_1 norms ($\sum_{nodes} |\cdot|/N_{Vi}$) of $\frac{\partial \mathcal{F}}{\partial u_j}$, $\frac{\partial \mathcal{F}}{\partial J_j}$ and $\frac{\partial \mathcal{F}}{\partial y_j}$ become less than 1.0E-06. The relaxation factor c in the update formula (3.39) will be set 1.0 whenever it is possible. It has been experimentally found that the method blows up for any value exceeding 1.0, and that the relaxation factor must be decreased as the grid gets finer.

We consider Friedrichs' model with $\epsilon = 0.04$ and $a = 0.3$. The initial approximation with 8 edges and the exact solution curve are shown in Figure 3.8. The results are given in Figures 3.9 to 3.12 for the cases of 8 and 64 edges. As can be seen in the figures, the solutions exhibit a surprising improvement over the results shown in Section 4. Observe that the vertices are concentrated in the large curvature region as desired. And no unnecessary vertices are placed in the linear portion anymore. For 8 edges, it has been found also that the method produces nearly the same results as in Figures 3.9 and 3.10 even when started with the oscillatory initial solution as

The number of edges	Iterations	$L_2(u - u_{exact})$	$L_2(J - J_{exact})$	$L_2(yj - y_{j_{L_2FITS}})$
8	3332	1.55E-03	2.77E-02	1.25E-02
16	27183	3.92E-04	7.05E-03	1.54E-02
32	399479	1.00E-04	1.82E-03	1.73E-02
64	4855967	2.46E-05	4.77E-04	1.86E-02

Table 3.1: The results of the numerical experiments.

in Figures 3.1 or 3.5. Hence the undesired minimum has been successfully excluded. The accuracy achieved by 8 edges is in fact comparable to that obtainable by 64 edges of fixed grid. The trajectories of three vertices are plotted in Figure 3.9, at every 10 iterations. These paths are significantly different from those shown in Figure 3.6. It is of particular importance to observe the large turns near the final solution curve. Since such a largely curved path simply corresponds to a long path, these trajectories can be thought of as a geometrical interpretation of the slow convergence of the present method. Viewed in this light, iterative methods better than the present one would be expected to move the vertices along the shorter paths to their final positions.

The number of iterations and the L_2 errors, including the cases of 16 and 32 edges, are given in Table 1. We observe that the errors now decrease as the number of grid points is refined, and that the quadratic convergence has been achieved[5]. Also in order to see how good the final grids are, L_2 norm of the difference between the final grid and the grid obtained by applying Baines' L_2 fits algorithm with adjustable nodes[5] to the exact solution (3.3). As can be seen, the differences are small for all the cases, indicating that the final grids are nearly optimal in the L_2 sense. The connection of the two methods seems to lie in our L_2 error estimate described in the

[5]Note that all the computations in this thesis were performed with double precision, but the results have been truncated for brevity.

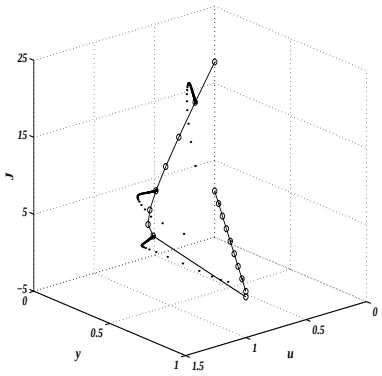

Figure 3.9: The solution curve in (u,J,y) space and the trajectories of the 2nd, the 5th, and the 8th vertices(dots). 8 edges.

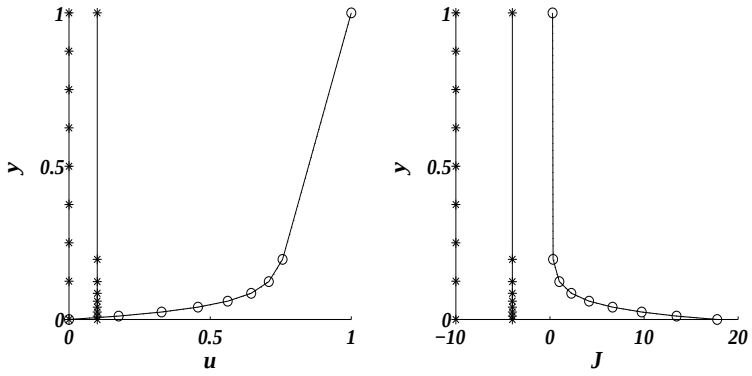

Figure 3.10: The projections of the curve onto $u-y$ and $J-y$ planes with the initial and the final locations of the vertices on y-axis indicated by $*$.

previous section. However a rather disappointing result is the large increase in the number of iterations against the refinement of the grid. The results indicate that doubling the number of edges requires nearly an order of magnitude larger number of iterations.

Finally the method was applied for various values of ϵ. It has been found that the method converges to a wrong solution for approximately $\epsilon \leq$ 5.00E-04. What happens is that Ψ_E are driven to zero at convergence, but Φ_E are not driven to zero but just equally distributed with the weight $\frac{1}{\Delta y_E}$. We have found one possible way to overcome this difficulty. The method works for extremely small ϵ's, if the weight $\frac{1}{\Delta y_E}$ is removed from the norm. In fact it has been confirmed that the method works even for ϵ=1.0E-10 with 8 edges. However this norm also has a defect that it allows zero or negative Δy_E because the norm is always positive irrespective of the sign of Δy_E, and thus the minimization can proceed. For instance, for $\epsilon = 0.04$ with 16 edges, it has been observed that the method indeed produces a negative Δy_E during

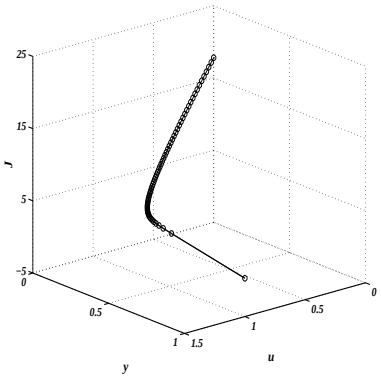

Figure 3.11: The solution curve in (u,J,y) space. 64 edges.

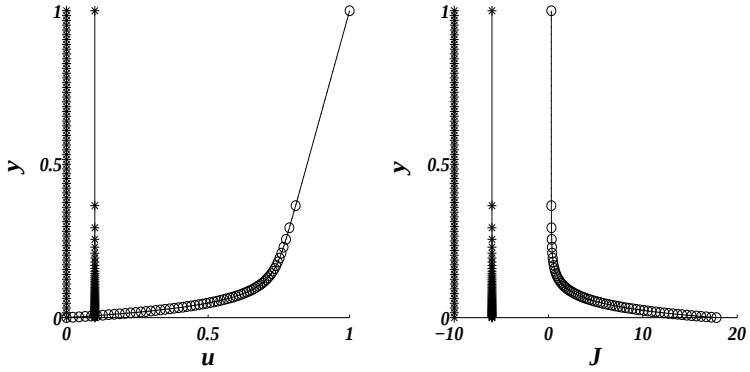

Figure 3.12: The projections of the curve onto $u-y$ and $J-y$ planes with the initial and the final locations of the vertices on y-axis indicated by $*$.

the iteration. In this case, however, the method finally recovered the solution close to the one shown above, i.e. removed all of the negative Δy_E's. But we have found also that with 32 edges the method fails to converge, creating negative Δy_E's. Even after ten million of iterations, a negative Δy_E remains. Further study is necessary concerning with a small ϵ.

3.5 A Nonlinear Example

It is straightforward to apply the method to any two-point boundary-value problems provided a proper norm can be defined. We give one example here: the viscous Burgers equation in one-dimension given by

$$\epsilon \frac{d^2u}{dy^2} - u\,\frac{du}{dy} = 0 \tag{3.48}$$

The equivalent first-order system is

$$\begin{aligned} \epsilon \frac{dJ}{dy} - u\frac{du}{dy} &= 0 \\ \frac{du}{dy} - J &= 0 \end{aligned}$$

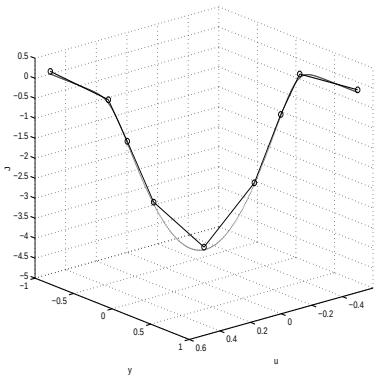

Figure 3.13: 3D View of the solution

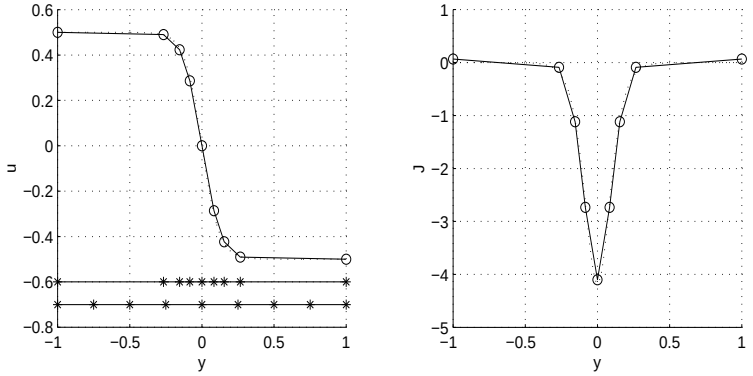

Figure 3.14: The final solutions u and J and the grid.

where $J = \frac{du}{dy}$ has been introduced. As before, dividing the domain $-1 \leq y \leq 1$ into a set $\{E\}$ of elements(segments) and assuming that u and J vary linearly within each element, we integrate the equations to obtain the residuals.

$$\Phi_E = \epsilon \Delta J_E - \bar{u}_E \Delta u_E \tag{3.49}$$

$$\Psi_E = \Delta u_E - \bar{J}_E \, \Delta y_E \tag{3.50}$$

Note that the two residuals have different dimensions, and therefore we need to weight them to define a dimensionally consistent norm. Reasonable quantities that would play the role are ϵ and Δy_E. A simple dimensional analysis tells that we define the norm in the following form.

$$\mathcal{F} = \sum_E \frac{1}{\Delta y_E} \left[\Phi_E^2 + \left(\frac{\epsilon}{\Delta y_E} \right)^2 \Psi_E^2 + \left(\Delta {y_E}^2 \Delta {J_E}^2 - C \right)^2 \right] \tag{3.51}$$

where the equidistribution of the error estimate has been incorporated in a similar manner as before. The final solutions and the grid, for $\epsilon = 0.03$, are plotted in Figures 3.5 and 3.14. The asterisks indicate the final and the initial grid locations, and we see that they have moved to capture the large variation in the middle. The L_2 error on the final grid is $9.64E-03$ which is comparable to the solution with a fixed uniform grid of 42 edges.

CHAPTER IV

HYPERBOLIC PROBLEMS IN TWO DIMENSIONS

4.1 Linear Hyperbolic Equations

We now begin the discussion of the least-squares moving grid method for two-dimensional problems. Based on the fact that any partial differential equations in two dimensions can be decomposed into a certain number of hyperbolic(advection) equations and elliptic systems[81], we discuss the two kinds of problems separately. In this chapter, we consider hyperbolic problem. Specifically, we start with the problems of linear type, i.e. the advection vector (or coefficient matrix for systems) does not depend on the solution itself. The least-squares method is known to work remarkably well for the problems of this kind. The first result obtained by Roe[80] was a circular advection problem where a grid evolved into a characteristic grid, creating circular paths on which the exact solution was actually found (Figures 1.3 and 1.4). We first describe the method for a simple linear advection and introduce the use of degenerate elements. A system of equations is then considered. In the last section, we consider a nonlinear equation to illustrate a difficulty associated with nonlinear shocks.

4.1.1 Linear Advection

Consider the steady-state two-dimensional scalar advection equation,

$$a\partial_x u + b\partial_y u = 0 \tag{4.1}$$

where a and b are constants. This is a problem with a preferred direction $dy/dx = b/a$ along which u is constant, which may be called a characteristic relation. To solve this numerically, we first divide a domain of interest into a set $\{T\}$ of triangles and store the solution at vertices. Within each element $T \in \{T\}$, we define the residual by

$$\Phi_T = \int_T (a\partial_x u + b\partial_y u)\, dxdy \tag{4.2}$$

which, by Green's theorem, reduces to a closed line integral around the boundary ∂T,

$$\Phi_T = \oint_{\partial T} (audy - budx). \tag{4.3}$$

Assuming that u varies linearly within the triangle with vertices numbered 1,2,3 in counterclockwise order, we obtain

$$\begin{aligned} \Phi_T = \quad & a\frac{u_1+u_2}{2}(y2-y1) - b\frac{u_1+u_2}{2}(x2-x1) + a\frac{u_2+u_3}{2}(y3-y2) \\ - \quad & b\frac{u_2+u_3}{2}(x3-x2) + a\frac{u_3+u_1}{2}(y1-y3) - b\frac{u_3+u_1}{2}(x1-x3) \end{aligned} \tag{4.4}$$

which can be rearranged into

$$\Phi_T = \frac{1}{2}\sum_{i\in j_T} u_i \left(a\Delta y_i - b\Delta x_i\right) \tag{4.5}$$

where j_T is the set of vertices $\{1,2,3\}$ of triangle $T \in \{T\}$ and $\Delta(\,)_i$ denotes a difference taken counterclockwise along the side opposite to node i. Note that we can write

$$\Phi_T = \frac{1}{2}\sum_{i\in j_T} u_i \Delta \left(ay - bx\right)_i \tag{4.6}$$

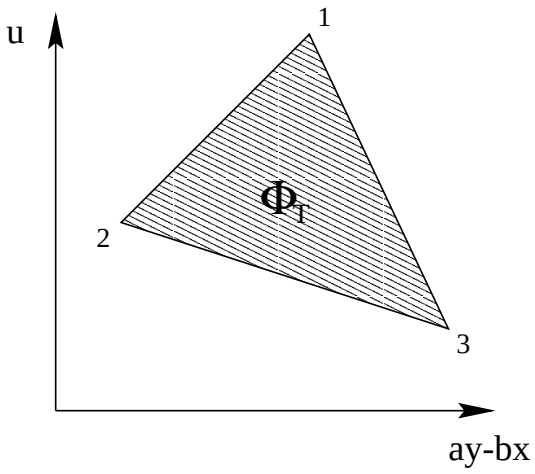

Figure 4.1: The residual represents the area of the triangle projected onto the characteristic plane.

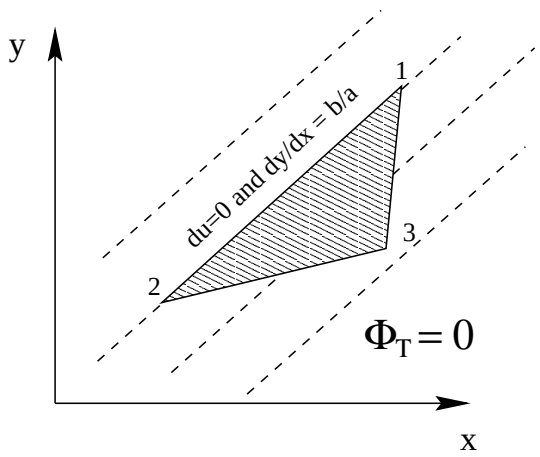

Figure 4.2: A triangle in the physical plane which has zero area in the characteristic plane.

which represents the area of the triangle in a characteristic plane that is defined here as a plane whose coordinates are u and $ay-bx$ (Figure 4.1). It is therefore immediate to see that this area will vanish if both u and $ay - bx$ are constant along one of the sides of the triangle because that side is reduced to a point in the characteristic plane. In the physical plane, this means that the residual will vanish if one edge is aligned with the characteristic direction $dy/dx = b/a$ and u is constant along that edge, satisfying the characteristic relation (Figure 4.2). Hence vanishing area in the characteristic plane is equivalent to satisfying the characteristic relation. It must be noted also that it is possible in general to drive the residual to zero only by allowing the grid to move. Now, we attempt to minimize

$$\mathcal{F} = \sum_{T \in \{T\}} F_T = \frac{1}{2} \sum_{T \in \{T\}} \Phi_T^2 \tag{4.7}$$

with respect to the nodal quantities, including the nodal position, $\mathbf{U}_j = (u_j, x_j, y_j)$. This is an unweighted norm (without the area weight $1/S_T$). We choose this here for the purpose of showing the ability of the method for computing the exact dis-

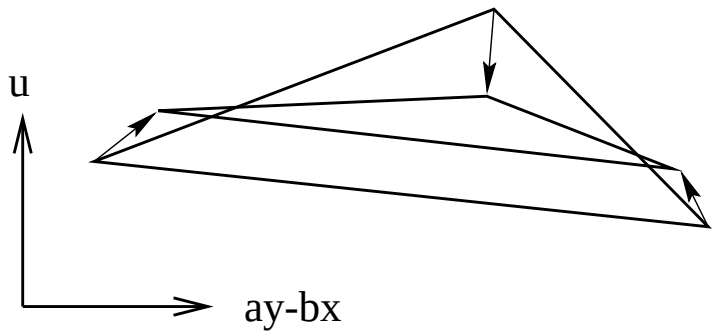

Figure 4.3: Typical nodal movement within a triangular element.

continuous solution, introducing degenerate elements: elements with zero area. Note that this norm has no trouble with zero triangle areas, i.e. always nonnegative and therefore minimization proceeds. Suppose that we use the steepest descent method to minimize the norm. Then the changes made to each vertex are given by

$$\delta \mathbf{U}_j = -c \frac{\partial \mathcal{F}}{\partial \mathbf{U}_j} \tag{4.8}$$

where $\mathbf{U}_j = [u_j, x_j, y_j]^t$ and c is a small constant that may be different for different variables.

Here, it is instructive to see the action taken place within a single triangle. For each vertex i of triangle T, the following changes are made.

$$\delta u_i = \frac{c_u}{2}(a\Delta y_i - b\,\Delta x_i)\Phi_T \tag{4.9}$$

$$\delta x_i = -\frac{c_x}{2} b \Delta u_i \Phi_T \tag{4.10}$$

$$\delta y_i = \frac{c_x}{2} a \Delta u_i \Phi_T \tag{4.11}$$

where c_u and c_x are small constants. Now, multiplying δx_i by b and δy_i by a, and subtracting, we obtain

$$\delta u_i = \frac{c_u}{2}(a\Delta y_i - b\,\Delta x_i)\Phi_T \tag{4.12}$$

$$a\delta y_i - b\delta x_i = \frac{c_x}{2}(a^2 + b^2)\Delta u_i \Phi_T. \tag{4.13}$$

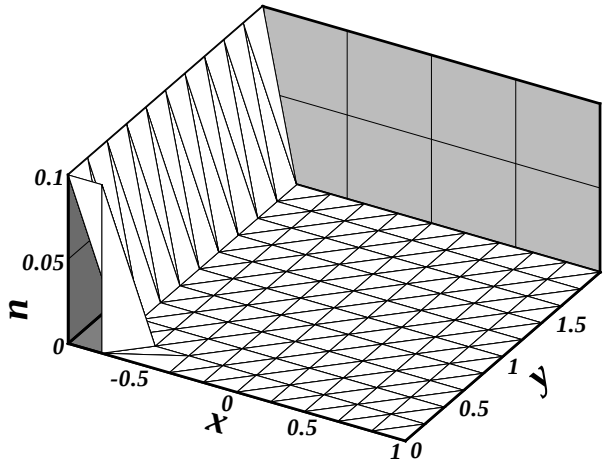

Figure 4.4: Initial Solution

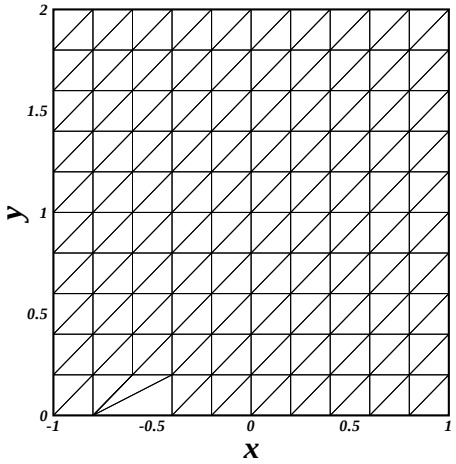

Figure 4.5: Initial Grid

Then, the choice $c_x = c_u/(a^2 + b^2)$ gives

$$\delta u_i = \frac{c_u}{2}(a\Delta y_i - b\,\Delta x_i)\Phi_T \tag{4.14}$$

$$\delta(ay - bx)_i = \frac{c_u}{2}\Delta u_i \Phi_T \tag{4.15}$$

which shows that the scheme will move the vertex i, in the characteristic plane, towards the opposite side in its normal direction and therefore it minimizes the area of the triangle projected onto this plane is minimized as quickly as possible (see Figure 4.3). The actual change given to node i is a sum of this actions to the surrounding triangles. Recall now that the problem is underdetermined on triangular grids for scalar equations. Then, we expect that the residuals can be made to vanish for every triangle, and therefore at a minimum the grid forms a characteristic mesh, which is not necessarily unique, but on any of which the exact solution can be found. We consider a square domain with an initial solution with a discontinuity as shown in Figures 4.5 and 4.4. Note that the discontinuity is introduced by inserting a

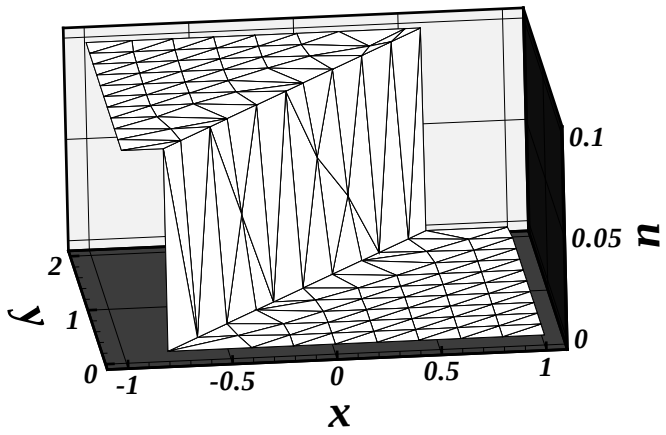

Figure 4.6: Converged Solution for $a = 0.7$ and $b = 1.0$.

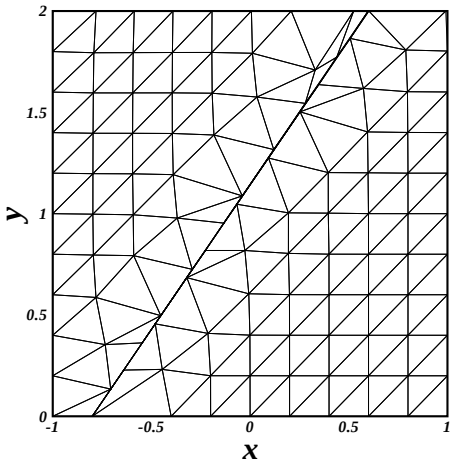

Figure 4.7: The Final Grid

degenerate element. The advection vector (a, b) is chosen to be $(0.7, 1)$. The results are shown in Figures 4.6 and 4.7 where all the residuals have vanished; the solution is exact. It is worth mentioning that the boundary nodes at out-flow boundary (top and right) have been moved in the same way as the interior nodes, but their movement is restricted along the boundary. The boundary node movement is in fact an essential item to obtain such a remarkable solution because the initial node distribution might not be satisfactory. In our case here, two nodes need to be placed in the location where the discontinuous solution reaches the top boundary, but the initial grid is not tailored in such a way. As can be seen in Figure 4.7, the nodes at the top boundary has been moved automatically by the minimization to capture the discontinuous solution successfully. On the right boundary, nothing much has happened because residuals are zero from the beginning. This shows that the mesh movement (also the solution updates) responds to nonzero residuals only, and therefore the grid has

been altered only in an important region. The advantage of the method is obvious: a discontinuity can be captured without clustering many nodes as would have been done by the usual curvature-based adaptation [41, 23] and also its operation is local not creating significant grid distortion. Similar results are reported in [8] where they showed the possibility of extending the use of degenerate elements to nonlinear hyperbolic equations.

4.1.2 Linearized Aerodynamics

We consider the *Generalized Cauchy-Riemann equations*

$$(1 - M^2)\partial_x u + \partial_y v = 0, \qquad \partial_x v - \partial_y u = 0, \tag{4.16}$$

which are physically the description of small irrotational perturbations to a uniform flow of Mach number M; hyperbolic if $M > 1$; elliptic otherwise. This system has been well studied theoretically in [80, 82], and therefore our discussion will be brief. We consider supersonic cases where we can find the characteristic equations.

$$\beta u + v = \text{const., along } x + \beta y = \text{const.} \tag{4.17}$$

$$\beta u - v = \text{const., along } x - \beta y = \text{const.} \tag{4.18}$$

where $\beta = \sqrt{M^2 - 1}$.

On the assumption that u and v vary linearly within each triangle, the residuals are obtained by integrating (4.16).

$$\Delta_T = (1 - M^2)\frac{1}{2}\sum_{i \in j_T} u_i \Delta y_i - \frac{1}{2}\sum_{i \in j_T} v_i \Delta x_i \tag{4.19}$$

$$\Omega_T = \frac{1}{2}\sum_{i \in j_T} v_i \Delta y_i + \frac{1}{2}\sum_{i \in j_T} u_i \Delta x_i. \tag{4.20}$$

The norm is given by

$$\mathcal{F} = \sum_{T \in \{T\}} F_T = \frac{1}{2}\sum_{T \in \{T\}} \frac{\Delta_T^2 + |M^2 - 1|\Omega_T^2}{S_T} \tag{4.21}$$

which has special properties as demonstrated in [80, 82]: in the subsonic case, it decouples the variables u and v completely and the discretization can be shown to be equivalent to the Galerkin finite-element method for the second-order equivalents of (4.16)

$$(1 - M^2)u_{xx} + u_{yy} = 0 \tag{4.22}$$

$$(1 - M^2)v_{xx} + v_{yy} = 0; \tag{4.23}$$

in the supersonic case, F_T vanishes when any two edges(or one edge for simple wave solutions) of the triangle T oriented along the two characteristics and the corresponding characteristic equations are satisfied on these edges.

One way to understand its property in the supersonic case is to find characteristic form of the residuals as we have seen for a linear advection equation. It is easy to show that the residuals can be combined to form another interesting pair of residuals C_T and D_T

$$C_T = \frac{1}{2}\sum_{i \in j_T} (v + \beta u)_i \Delta(x + \beta y)_i = \Delta_T + \beta\Omega_T \tag{4.24}$$

$$D_T = \frac{1}{2}\sum_{i \in j_T} (v - \beta u)_i \Delta(x - \beta y)_i = \Delta_T - \beta\Omega_T \tag{4.25}$$

Clearly, C_T represents the area of the triangle projected onto a plane with coordinates $v + \beta u$ and $x + \beta y$, and similarly for D_T. For simple wave solution, say, with $v + \beta u$ constant, C_T will be identically zero, and D_T will vanish if one of the side is directed along the characteristic direction $dy/dx = 1/\beta$ and $v - \beta u$ is constant along that edge. Yet, by aligning another edge along the other characteristic direction and having the corresponding characteristic variable invariant on that edge, they can be both made to vanish simultaneously for non-simple wave solutions. Then, the

least-squares norm formed by these residuals

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2}\sum_{T\in\{T\}} \frac{C_T^2 + D_T^2}{S_T} \tag{4.26}$$

has a property that it recognizes the characteristic equations. Note that written in terms of Δ_T and Ω_T, this norm becomes identical to (4.21), thus justifying the choice of the weight $\beta = |M^2 - 1|$. We must remark however that the property of the characteristic recognition is not special to the norm (4.21). We may minimize any norm without losing this particular property. To see this, solve (4.24) and (4.25) for Δ_T and Ω_T to get

$$\Delta_T = \frac{C_T + D_T}{2} \tag{4.27}$$

$$\Omega_T = \frac{C_T - D_T}{2\beta}. \tag{4.28}$$

It is clear from these relations that the original residuals Δ_T and Ω_T will vanish when the characteristic residuals C_T and D_T vanish, i.e when the element is fitted to a simple or non-simple wave solution. This implies that we may minimize instead the norm without the weighting factor β^2

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2}\sum_{T\in\{T\}} \frac{\Delta_T^2 + \Omega_T^2}{S_T}, \tag{4.29}$$

and still have the characteristic recognition property. The difference, however, lies in the speed of convergence. With the weighting factor, we are minimizing the characteristic residuals directly, or the element area projected onto the two characteristic planes. It is easy to show that the movement of each node for a particular element is perpendicular to the side opposite to that node, that is the most effective direction to minimize the area[80, 82]. We therefore expect that minimizing a norm other than (4.21) would slow down the convergence.

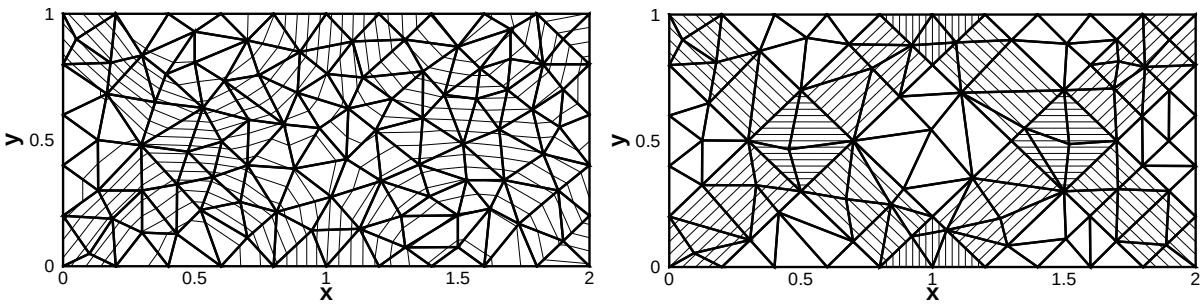

Figure 4.8: Left: Contours of u computed on a fixed grid indicated by the thicker lines. $\mathcal{F} \approx 10^{-3}$. Right: Contours of u and an optimal grid computed simultaneously, starting from the grid on the left. 18 nodes have been removed during the solution process. $\mathcal{F} \approx 10^{-12}$.

A computational result is available for a flow at $M = \sqrt{2}$ (from the left to the right) through a parallel duct placed at a small incidence, so that compression and rarefaction waves, created at the leading edge, reflect down the duct. As the boundary condition, we specify $v = 0$ at along the duct and the freestream condition at the inlet. For simplicity, the boundary nodes are fixed for this problem. The initial and the final grids are shown in Figure 4.8 where the former is composed of 180 elements and 106 nodes. On the left, the solution contours(thin lines), obtained on the fixed grid, are plotted on the initial Delaunay grid(thick lines). As can be expected, the solution diffuses a lot on this coarse grid. But once the grid is allowed to move, the method generates elements having one or two sides aligned with the characteristic directions, and simultaneously computes the solutions that are perfectly diffusion-free. Note that the grid has been altered only in regions where waves are actually present. Because two edges sometimes try to represent the same characteristic we implemented a scheme for removing redundant nodes: in a group of triangles that share a node, compute the length of each edge connecting the node and a neighboring node, also compute the distance between the node and the edge opposite to that node, and if the minimum length is less than 5% of the maximum, remove

the node. This is purely geometrical. In fact, we implemented the scheme based on the norm reduction, but the method removed all the interior points in the end, actually reducing the norm locally. Further investigation is required for the node removal procedure. The boundary nodes are fixed in this example because the initial distribution is satisfactory (exactly in the places where they can catch the waves). The possibility of creating a characteristic grid in the supersonic case means that the adapted solution is virtually exact.

4.1.3 Burgers' Equation

We now return to the scalar case, but a nonlinear equation for which the least-squares method experiences a difficulty. We consider the inviscid Burgers equation.

$$\partial_y u + u\partial_x u = 0 \tag{4.30}$$

On the assumption that u varies linearly within a triangle, we integrate the equation to get the residual.

$$\Phi_T = \frac{1}{2}\sum_{i\in j_T} u_i \left(\Delta y_i - \bar{u}_T \Delta x_i\right) \tag{4.31}$$

where $\bar{u}_T = \frac{u_1+u_2+u_3}{3}$. Note that this is exact for linear solutions. This means that Green's theorem holds exactly in the discrete form as well, and therefore summing up the residuals all over the domain yields cancellation of interior line integrals, thus leaving a single boundary integral, i.e. telescoping property. For this reason, it is called conservative linearization in the context of residual distribution (or fluctuation splitting) schemes[30]. Similarly to the linear case, it can be shown that the residual vanishes for the element when one of the edges is aligned along $dy/dx = \bar{u}_T$, i.e. the characteristic with the averaged speed. This is acceptable for a smooth solution, but not for a discontinuity. A solution is computed for the case of a stationary shock, and is shown in Figure 4.9 where the nodes at the top boundary were moved in the same

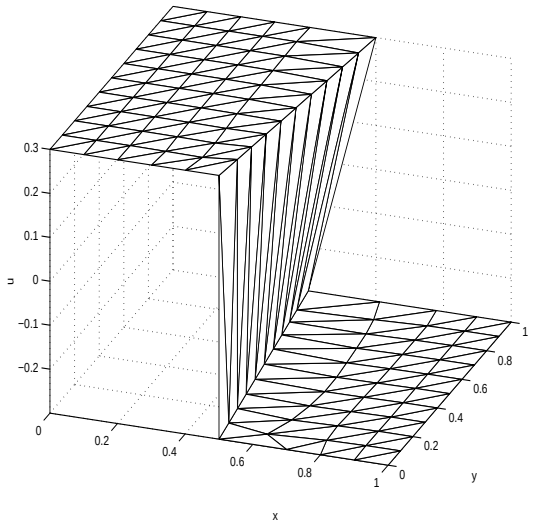

Figure 4.9: Converged Solution for the residual (4.31)

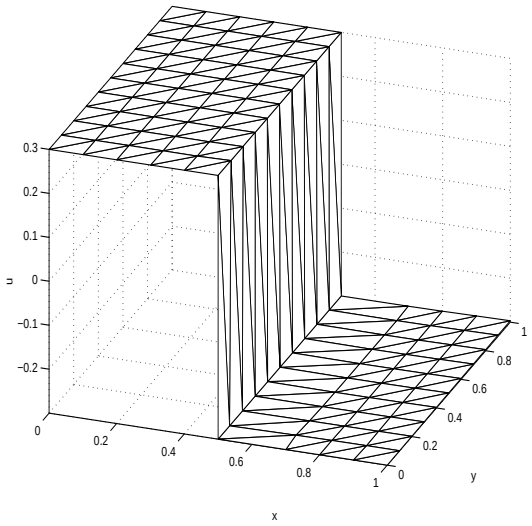

Figure 4.10: Converged solution for the residual (4.32)

way as the interior nodes but only in the x-direction. By minimizing the norm of the form (4.7), the residuals have been driven to zero for all elements. And the method indeed converges to an overturned shock. This is the reason that the area-weighted norm fails to converge at a nonlinear shock. It is important to note that the residual (4.31) is not unique. In general, there exist more than one form of discretization for nonlinear equations, e.g. the trapezoidal rule, the midpoint rule, etc. In fact, an alternative discretization that ensures a correct shock speed can be constructed by enforcing it to satisfy the jump condition along each edge of the element. The discretization takes the following form.

$$\Phi_T = \frac{1}{2} \sum_{i \in j_T} u_i \left(\Delta y_i - u_i \Delta x_i \right) \tag{4.32}$$

This residual vanishes when one of the sides of the element is aligned along a shock. In other words, it vanishes when it becomes parallel to the line with the slope $(u_1 + u_2)/2$ where the other nodal solution u_3 is assumed to be equal to u_1 or u_2. As can be seen in Figure 4.10, the shock is captured exactly. One problem with

this discretization is that it does not give a correct shock if more than one element is involved in the shock. It will give an overturned shock again. Also, it admits non-physical shocks(entropy-violating shocks). The detailed discussion on nonlinear problems is now given in the next section.

4.2 Nonlinear Hyperbolic Equations

We have seen that the least-squares method is capable of producing exact solutions for linear hyperbolic equations, automatically adjusting the mesh into a characteristic configuration. It is an advantage of the method that a discontinuity can be captured with many fewer nodes than the usual curvature adaptation [41, 23]. We have also seen, however, a straightforward extension to nonlinear problems fails, and the residual must be designed such that a shock relation is satisfied across shocks. The main focus of this section is then on the computation of residual for nonlinear hyperbolic problems.

4.2.1 Conservation Laws

We consider sets of two-dimensional conservation laws of the form

$$\partial_x \mathbf{F}(\mathbf{w}) + \partial_y \mathbf{G}(\mathbf{w}) = 0 \tag{4.33}$$

where $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{w}$ are all $\in \mathcal{R}^m$ in which each component of the conservative variable and fluxes is a bilinear function of the components of $\mathbf{w}$.

$$\mathbf{F}(\mathbf{w}) = \mathbf{w}^t \mathbf{C} \mathbf{w}, \quad \mathbf{G}(\mathbf{w}) = \mathbf{w}^t \mathbf{D} \mathbf{w} \tag{4.34}$$

where the superscript t denotes transpose, and $\mathbf{C}$ and $\mathbf{D}$ are constant symmetric third-order tensors. We can easily prove that for a small change $d\mathbf{w}$, we have, on account of symmetry,

$$d\mathbf{F} = \mathbf{w}^t \mathbf{C} d\mathbf{w}, \quad d\mathbf{G} = \mathbf{w}^t \mathbf{D} d\mathbf{w} \tag{4.35}$$

and for large changes we have the discrete version

$$\Delta\mathbf{F} = \mathbf{w}^t\mathbf{C}\Delta\mathbf{w}, \quad \Delta\mathbf{G} = \mathbf{w}^t\mathbf{D}\Delta\mathbf{w}. \tag{4.36}$$

This structure includes the Euler equations of compressible inviscid flow if $\mathbf{w}$ is taken to be Roe's parameter vector. The steady-state version of (4.33) can be written

$$\mathbf{w}^t\mathbf{C}\partial_x\mathbf{w} + \mathbf{w}^t\mathbf{D}\partial_y\mathbf{w} = 0. \tag{4.37}$$

The Hugoniot condition, $S\Delta\mathbf{F} - \Delta\mathbf{G}$, where $S = dy/dx$ is the slope of a discontinuity, becomes

$$\overline{\mathbf{w}}^t(S\mathbf{C} - \mathbf{D})\Delta\mathbf{w} = 0 \tag{4.38}$$

where $\overline{\mathbf{w}}$ is the arithmetic average of the variable across the discontinuity. To obtain a characteristic equation we pre-multiply (4.37) by a row vector ℓ such that

$$\ell\mathbf{w}^t\mathbf{C} = \ell\mathbf{w}^t\mathbf{D}/\lambda, \tag{4.39}$$

that is to say we have to solve the generalized eigenvalue problem for the pair of $m \times m$ matrices $\mathbf{w}^t\mathbf{C}$, $\mathbf{w}^t\mathbf{D}$. Given such an ℓ, the characteristic equation becomes

$$\ell\mathbf{w}^t\mathbf{C}(\partial_x + \lambda\partial_y)\mathbf{w} = 0 \tag{4.40}$$

or

$$\ell\mathbf{w}^t\mathbf{C}d\mathbf{w} = \ell\mathbf{w}^t\mathbf{D}d\mathbf{w} = 0 \qquad \text{on} \quad dy/dx = \lambda \tag{4.41}$$

Note that the characteristic equation (4.41) relates values *along* characteristic, but the Hugoniot condition relates values *across* the shock.

4.2.2 Quadrature Formulae

We are interested in computing solutions of (4.33) by a least-squares method on triangular grids in which the first step is to compute the residual.

$$\mathbf{\Phi}_{123} \;=\; \iint_{123} \left[\partial_x\mathbf{F}(\mathbf{w}) + \partial_y\mathbf{G}(\mathbf{w})\right] dxdy = \oint_{123} \left[\mathbf{F}(\mathbf{w})dy + \mathbf{G}(\mathbf{w})dx\right] \tag{4.42}$$

where the data is available at the vertices 1,2,3. The second step is then to distribute some fraction of the residual to each vertex. In the least-squares method, these fractions typically sum to zero over any given element, so that the method will be conservative however the residuals are evaluated. Therefore we focus first on different properties of various evaluations. Consider a single term, arising from integrating just one component of the flux vector over on edge of the element.

$$\mathbf{F}_{12} = \int_1^2 \mathbf{F}(\mathbf{w})\ dy = \int_1^2 \mathbf{w}^t \mathbf{C}\mathbf{w}\ dy \tag{4.43}$$

To evaluate this integral there is a class of simple formulas;

$$\mathbf{F}_{12}(\alpha) = \frac{(y_2 - y_1)}{2}\left[(\mathbf{w}_1^t \mathbf{C}\mathbf{w}_2 + \mathbf{w}_2^t \mathbf{C}\mathbf{w}_1) + \alpha(\mathbf{w}_1 - \mathbf{w}_2)\mathbf{C}(\mathbf{w}_1 - \mathbf{w}_2)\right] \tag{4.44}$$

where α is a parameter that has only a second-order relative effect on the accuracy of the estimate.

4.2.3 Formulas for the Residual

The formula for the residual is obtained by summing up the contributions from three sides. Introducing the notation,

$$\mathbf{w}_{ij}(\alpha) = \frac{\alpha}{2}(\mathbf{w}_i + \mathbf{w}_j) + (1 - \alpha)\mathbf{w}_k \tag{4.45}$$

where i, j, k are cyclically permuted for $1, 2, 3$, and rearranging terms, we obtain

$$\begin{aligned} \mathbf{\Phi}_{123}(\alpha) \quad = \quad & \mathbf{w}_{31}^t(\alpha)\left\{\Delta y_3 \mathbf{C} - \Delta x_3 \mathbf{D}\right\}(\mathbf{w}_3 - \mathbf{w}_1) + \\ & \mathbf{w}_{12}^t(\alpha)\left\{\Delta y_2 \mathbf{C} - \Delta x_2 \mathbf{D}\right\}(\mathbf{w}_2 - \mathbf{w}_1) \end{aligned} \tag{4.46}$$

where Δy_i is the difference of y taken anticlockwise along the edge opposite to the node i, similarly for Δx_i. Note that this formula is not symmetric with respect to

the vertices. There are two other rearrangements possible,

$$\begin{aligned}\mathbf{\Phi}_{123} &= \mathbf{w}_{12}^T(\alpha)\left\{(y_3-y_2)\mathbf{C}-(x_3-x_2)\mathbf{D}\right\}(\mathbf{w}_2-\mathbf{w}_1)+\\ &\quad \mathbf{w}_{23}^T(\alpha)\left\{(y_2-y_1)\mathbf{C}-(x_2-x_1)\mathbf{D}\right\}(\mathbf{w}_2-\mathbf{w}_3),\end{aligned} \tag{4.47}$$

$$\begin{aligned}\mathbf{\Phi}_{123} &= \mathbf{w}_{31}^T(\alpha)\left\{(y_3-y_2)\mathbf{C}-(x_3-x_2)\mathbf{D}\right\}(\mathbf{w}_3-\mathbf{w}_1)+\\ &\quad \mathbf{w}_{23}^T(\alpha)\left\{(y_1-y_3)\mathbf{C}-(x_1-x_3)\mathbf{D}\right\}(\mathbf{w}_3-\mathbf{w}_2),\end{aligned} \tag{4.48}$$

and all three will give the same numerical value. The particular arrangement (4.46) can be written,

$$\begin{aligned}\mathbf{\Phi}_{123}(\alpha) &= \left\{\frac{\partial \mathbf{F}}{\partial \mathbf{w}}\Delta y_3-\frac{\partial \mathbf{G}}{\partial \mathbf{w}}\Delta x_3\right\}(\mathbf{w}_3-\mathbf{w}_1)+\\ &\quad \left\{\frac{\partial \mathbf{F}}{\partial \mathbf{w}}\Delta y_2-\frac{\partial \mathbf{G}}{\partial \mathbf{w}}\Delta x_2\right\}(\mathbf{w}_2-\mathbf{w}_1)\end{aligned} \tag{4.49}$$

where the Jacobian matrices in the first term are evaluated at the state $\mathbf{w}_{31}(\alpha)$ and those in the second terms at the state $\mathbf{w}_{12}(\alpha)$, which shows that the parameter α acts on the Jacobian matrices. For example, taking $\alpha = 2/3$ we obtain the conservative linearization which has been utilized in the residual splitting method [30].

Shock Recognition: $\alpha = 1$

Suppose that two of the states happen to be equal, $\mathbf{w}_1 = \mathbf{w}_2 = \mathbf{w}_c$. Then the second line of (4.46) vanishes and the first line becomes

$$\mathbf{\Phi}_{123}(\alpha) = \mathbf{w}_{31}^t(\alpha)\left\{\Delta y_3\mathbf{C}-\Delta x_3\mathbf{D}\right\}(\mathbf{w}_3-\mathbf{w}_c). \tag{4.50}$$

For the special choice $\alpha = 1$, which gives $\mathbf{w}_{31}(\alpha) = \overline{\mathbf{w}}_{31} = (\mathbf{w}_3+\mathbf{w}_c)/2$ (arithmetic average across the shock), this corresponds precisely with the Hugoniot condition (4.38) along the edge 31, provided the edge 12 is aligned in shock of the speed $dy/dx = S$ (Figure 4.11). This is independent of the position of the other node, and

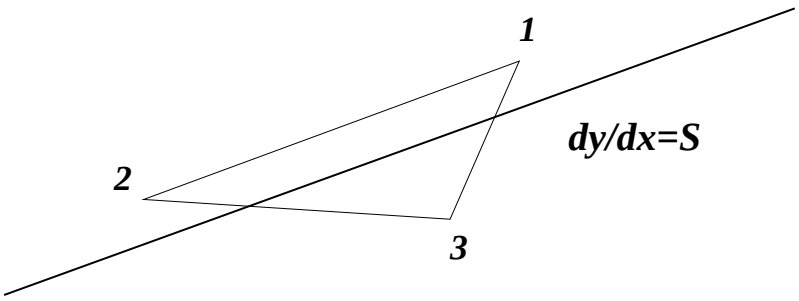

Figure 4.11: Shock Recognition: $\alpha = 1$

introduces the possibility of capturing (or fitting) shocks by means of degenerate elements. However, unfortunately, this admits non-physical shocks as well.

Special Properties: $\alpha = 0$

For scalar problems, the choice $\alpha = 0$ has a special property. With $\alpha = 0$, the residual becomes

$$\begin{aligned} \mathbf{\Phi}_{123}(0) &= \mathbf{w}_2^t \left\{\Delta y_3 \mathbf{C} - \Delta x_3 \mathbf{D}\right\} (\mathbf{w}_3 - \mathbf{w}_1) + \\ &\quad \mathbf{w}_3^t \left\{\Delta y_2 \mathbf{C} - \Delta x_2 \mathbf{D}\right\} (\mathbf{w}_2 - \mathbf{w}_1) \end{aligned} \tag{4.51}$$

where all quantities are thought of as scalar. Now suppose that the edge 12 is aligned with the characteristic whose speed evaluated at the state 2, i.e. *on the edge itself.* Then the first term vanishes. And the second represents the characteristic equation $d\mathbf{w} = 0$ exactly, again for any position of the third node (Figure 4.12). For systems, all this can actually happen but only approximately. Note that the choice $\alpha = 0$ relates the solution values *along* the edge rather than *across* as in the case $\alpha = 1$.

Another important property associated with this choice is the ability to compute physical rarefaction. To see this, consider Burgers' equation $\partial_t u + \partial_x f = 0$ where $f = u^2/2$. Discretizing the spatial derivative term by the quadrature formula (4.44), rearranging the terms, we find

$$\frac{du_j}{dt} + \frac{f_{j+1} - f_{j-1}}{2\Delta x} = (1 - \alpha)\Delta x^2 \frac{u_{j+1} - u_{j-1}}{2\Delta x} \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} \tag{4.52}$$

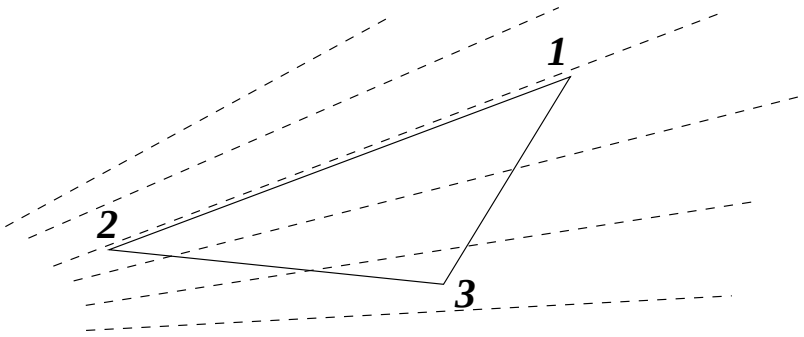

Figure 4.12: Characteristic Recognition: $\alpha = 0$

which is a second order discretization of

$$\partial_t u + \partial_x f = (1-\alpha)\Delta x^2\, \partial_x u\; \partial_x^2 u. \tag{4.53}$$

This shows that for diverging characteristics, $\partial_x u > 0$, taking $\alpha < 1$ gives positive dissipation. Experimentally we find that α needs to be well below unity, and that zero is a good choice. The same effect can be explained by means of entropy argument. Write the semi-discrete equation (4.52) in the conservation form.

$$\frac{du_j}{dt} = -\frac{\Delta t}{\Delta x}[f_{j+1/2} - f_{j-1/2}] \tag{4.54}$$

with the flux function $f_{j+1/2} = u_j u_{j+1} + \frac{\alpha}{2}(u_{j+1} - u_j)^2$. Define a non-increasing quadratic entropy $\frac{1}{2}u_j^2$ [56], sum them up all over the nodes, and arrange the result to get

$$\frac{d}{dt}\sum_j \frac{1}{2}u_j^2 = \sum_j \left(\frac{\alpha}{2} - \frac{1}{3}\right)(u_{j+1} - u_j)^3. \tag{4.55}$$

This shows that the entropy is conserved by taking the choice $\alpha = 2/3$, reduced by $\alpha > 2/3$ for the compressive data $u_{j+1} < u_j$, and reduced also by $\alpha < 2/3$ for the expansive data $u_{j+1} > u_j$. Therefore we have to take $\alpha > 2/3$ for shock waves, but $\alpha < 2/3$ for expansion waves. This is consistent with the above discussion: $\alpha = 1$ recognizes shocks but admits non-physical shocks and $\alpha = 0$ avoids non-physical shocks.

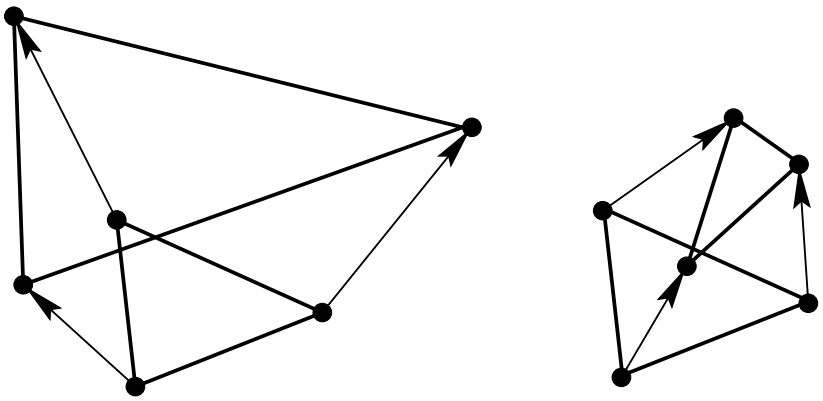

Figure 4.13: Left: Element in expansion. Right: Element in compression. Arrows indicate the characteristic speed vectors.

4.2.4 Detecting Compression/Expansion

One possible way to take advantage of the formulas for the residual is to use $\alpha = 1$ for elements in shocks and $\alpha = 0$ for others, thus capturing shocks exactly at the same time avoiding possible rarefaction shocks. The decision can be made based on the rate of change of the triangle area due to the virtual vertex motion caused by the characteristic speeds. Imagine that a triangle is convected in a characteristic field. If the characteristics are diverging, implying expansion, the triangle area would increase. On the other hand, if the characteristics are converging, implying compression, the triangle area would decrease. See Figure 4.13. It is easy to show that the rate of change of the area is given by

$$\frac{dS_T}{dt} = \frac{1}{2} \sum_{i=1,2,3} \vec{\lambda}_i \cdot \mathbf{n}_i \tag{4.56}$$

where $\mathbf{n}_i$ is the scaled inward normal vector of the edge opposite to the vertex i (its magnitude = the length of that edge), and $\vec{\lambda}_i$ is a characteristic speed vector at vertex i. That is to say, the element is being compressed if

$$\frac{dS_T}{dt} < 0 \tag{4.57}$$

whereas being expanded if

$$\frac{dS_T}{dt} > 0. \tag{4.58}$$

For system of equations, we compute this for each wave-like component.

4.2.5 Least-Squares Formulation

The solutions are sought that minimize the norm

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2} \sum_{T\in\{T\}} \frac{\mathbf{\Phi}_T^t Q_T \mathbf{\Phi}_T}{S_T} \tag{4.59}$$

over a set $\{T\}$ of triangular elements that divides the domain of interest. Q_T is a positive definite symmetric matrix that assigns relative weight to the different equations. The change made to each vertex $\{j\}$ is the sum of the contribution from the surrounding triangles $\{T_j\}$, and can be written in the residual distribution format as follows.

$$\delta \mathbf{w}_j = -\omega_u \sum_{T\in\{T_j\}} \mathcal{A}_j^T \mathbf{\Phi}_T \tag{4.60}$$

$$\delta \mathbf{x}_j = -\omega_x \sum_{T\in\{T_j\}} \left\{ \mathcal{B}_j^T \mathbf{\Phi}_T - \frac{F_T}{2S_T} \mathbf{n}_j \right\} \tag{4.61}$$

where $\mathbf{w}_j$ is a solution vector at the node, $\mathbf{x}_j$ is the nodal position vector, and ω_u and ω_x are small constants. The second term in (4.61) comes from differentiating the weight $1/S_T$ in the norm. Note that the scheme is equivalent to the steepest descent method. $\mathcal{A}_j^T$ and $\mathcal{B}_j^T$ are the distribution matrices given by

$$\mathcal{A}_j^T = \frac{1}{S_T} \left(Q_T \frac{\partial \mathbf{\Phi}_T}{\partial \mathbf{w}_j} \right)^t \tag{4.62}$$

$$\mathcal{B}_j^T = \frac{1}{S_T} \left(Q_T \frac{\partial \mathbf{\Phi}_T}{\partial \mathbf{x}_j} \right)^t. \tag{4.63}$$

Here the computation of the derivative in $\mathcal{A}_j^T$ requires careful consideration. We have observed that taking the derivative straightforwardly can result in completely

wrong solutions. As suggested in [48], it is important to linearize the equations first, e.g.

$$\tilde{\mathbf{A}}\partial_x \mathbf{w} + \tilde{\mathbf{B}}\partial_y \mathbf{w} = 0 \tag{4.64}$$

where $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are locally linearized constant matrices, and then apply the least-squares method: minimize

$$\mathcal{F} = \sum_{\{T\}} \left[\iint_T \left(\tilde{\mathbf{A}}\partial_x \mathbf{w} + \tilde{\mathbf{B}}\partial_y \mathbf{w} \right) \, dxdy \right]^2 . \tag{4.65}$$

This means that we take the derivatives assuming $\mathbf{w}_{ij}(\alpha)$ are constant in (4.46) because these quantities act on the Jacobian matrix as mentioned before. Yet this raises another important point of consideration. Recall that the residual has two other rearrangements other than (4.46). The derivative computed as above now depends on this arrangement. Then it would be reasonable to take the arithmetic average of the three possible forms and take the derivative, which yields

$$\begin{aligned} \frac{\partial \mathbf{\Phi}_T}{\partial \mathbf{w}_1} &= \frac{\mathbf{w}_{31}^T(\alpha)}{3} \left[(\Delta y_1 - \Delta y_3)\mathbf{C} - (\Delta x_1 - \Delta x_3)\mathbf{D} \right] + \\ &\quad \frac{\mathbf{w}_{12}^T(\alpha)}{3} \left[(\Delta y_1 - \Delta y_2)\mathbf{C} - (\Delta x_1 - \Delta x_2)\mathbf{D} \right], \end{aligned} \tag{4.66}$$

and this has been actually found to work.

4.2.6 Results

Results are available for Burgers' equation $\partial_y u + u\ \partial_x u = 0$, for which Q_T was taken to be unity. Figures 1 and 2 show the final grid and solution for a right-moving curved shock for the boundary conditions; $u = 3$ for $x \leq -0.8$ and $\frac{30}{17}(x-1)$ for $x \geq -0.7$ where the data is interpolated linearly between $x = -0.8$ and $x = -0.7$, modeling an initial discontinuity. The grid and the solution were obtained by repeating the following cycle.

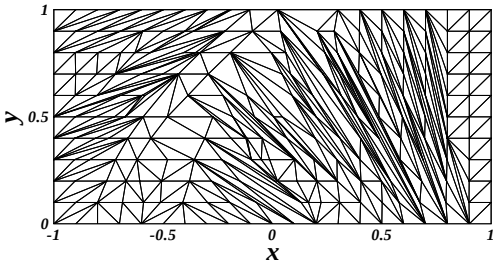

Figure 4.14: The final grid for the curved shock.

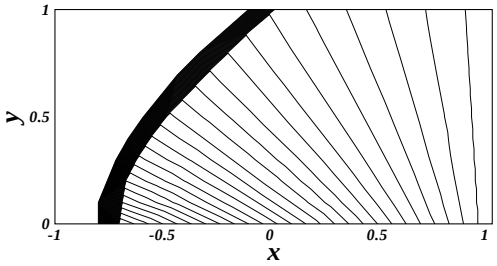

Figure 4.15: Solution contours for the shock.

1. Converge the solution on a fixed grid

2. Assign α to each element, depending the sign of (4.56)

3. Remove undesirable nodes

4. Update solutions and coordinates, with edge swapping interleaved (2000 iterations maximum)

and the method terminates when the changes to solutions and coordinates are both small in the step 4. The edge swapping is based on the norm reduction, which attempts to create a characteristic mesh as clearly seen in the final grid. Also, the scheme for removing redundant nodes was implemented as described in Section 4.1.2. The nodes on the upper boundary were also allowed to move by the regular procedure but only along the x-axis. The method converged at 10 cycles. And the final values of α are 1 for the elements forming a shock, and 0 elsewhere. As pointed out at the end of the section 4.1.3, if more than one element is involved in the shock, implying the existence of nodes inside the shock, it will create an overturned shock. In this example, we did not encounter this problem. But it can be fixed easily by removing the nodes that is associated with triangles that have negative area.

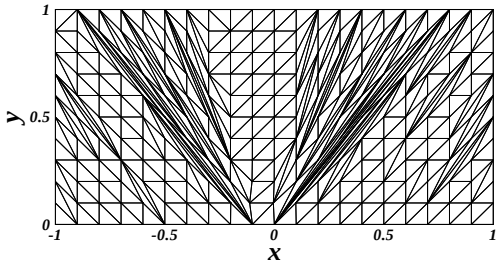

Figure 4.16: The final grid for the expansion.

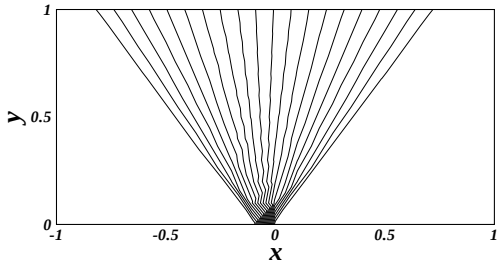

Figure 4.17: Solution contours for the expansion.

A grid aligned expansion wave was computed with $u = -0.8$ on the left and $u = 0.8$ on the right, to demonstrate the ability of the method to compute entropy-satisfying solutions. The values of α are automatically set to be zero everywhere in the step 2, and as can be seen the final solution is the correct smooth expansion fan. The method converges at just 1 cycle. Increasing the value of α, we found that a rarefaction shock appeared in the middle of the rarefaction and it became finally a perfectly-resolved rarefaction shock when $\alpha = 1$. This is consistent with the observation in section 4.2.3.

CHAPTER V

ELLIPTIC PROBLEMS IN TWO DIMENSIONS

5.1 Cauchy-Riemann Equations: $\phi - \psi$ Formulation

In this Chapter, we consider the Cauchy-Riemann equations, a prototype of first-order elliptic equations. In particular, we consider the Cauchy-Riemann equations for streamfunction and velocity potential, but the interpretation of the variables is not important here, and therefore the discussion will be valid for any two variables. For elliptic problems, there is no longer a preferred direction along which information travels. A property of grids required for such problems is local isotropy, but the grids must be tailored for side effects that can arise due to the presence of singularities. We shall see that the mechanism for grid movement is quite different from the one for hyperbolic problem and reflects faithfully the nature of elliptic problems, and also that the solution method is equivalent to a well-known finite-element method.

5.1.1 Governing equations

In two dimensions, an incompressible and irrotational flow is governed by

$$\partial_x \phi + \partial_y \psi = 0 \tag{5.1}$$

$$\partial_x \psi - \partial_y \phi = 0 \tag{5.2}$$

where ϕ and ψ are a velocity potential and a streamfunction respectively. These equations are accompanied by the boundary conditions: Dirichlet condition for ψ that specifies streamlines along boundaries, and Neumann condition for ϕ that constraints the flow across any internal bodies. It is possible to exchange the independent and dependent variables to get governing equations for x and y in $\phi - \psi$ plane, which is known as hodograph transformation.

$$\partial_\phi x = \partial_y \psi / j, \;\; \partial_\psi x = -\partial_y \phi / j, \;\; \partial_\phi y = -\partial_x \psi / j, \;\; \partial_\psi y = \partial_x \phi / j \tag{5.3}$$

where

$$j = \partial_x \phi \partial_y \psi - \partial_y \phi \partial_x \psi = (\partial_\psi y \partial_\phi x - \partial_\psi x \partial_\phi y)^{-1}. \tag{5.4}$$

Note that the Jacobian is always negative, at least for a converged solutions in the discrete case. We thus have another set of Cauchy-Riemann equations in the hodograph plane (ϕ, ψ).

$$\partial_\phi x + \partial_\psi y = 0 \tag{5.5}$$

$$\partial_\psi x - \partial_\phi y = 0. \tag{5.6}$$

Assume that u and v vary linearly within a triangle. Then, any first derivatives are constant and all the above results hold discretely as well.

5.1.2 Residual

Residuals are in general obtained by integrating the governing equations over an element with some assumption on the variation of the solutions, typically piecewise linear. But since linear variation implies constant first derivatives and our equations consist of only first derivatives, the integration is a simple product of the governing equations, with the piecewise linear approximation inserted, and the element area

S_T. The piecewise linear interpolation formulas ϕ and ψ within each element are given by

$$\phi = \sum_{i \in j_T} \phi_i N_i^T(x,y), \quad \psi = \sum_{i \in j_T} \psi_i N_i^T(x,y) \tag{5.7}$$

where ϕ_i and ψ_i are the values at node i, and

$$N_i^T(x,y) = \frac{a_i + b_i x + c_i y}{2S_T}, \quad a_i = x_k y_l - y_k x_l, \quad b_i = \Delta y_i, \quad c_i = -\Delta x_i \tag{5.8}$$

and S_p is the area of the element given by

$$S_T = \frac{1}{2} \sum_{i \in j_T} x_i \Delta y_i = -\frac{1}{2} \sum_{i \in j_T} y_i \Delta x_i \tag{5.9}$$

where $\Delta\{\}_i$ denotes a difference taken counterclockwise along the side opposite to j and the subscripts k and l take 1, 2 and 3, and are permuted cyclically for i. Then it is straightforward to show that

$$\frac{\partial \phi}{\partial x} = \frac{1}{2S_T} \sum_{i \in j_T} \phi_i \Delta y_i, \quad \frac{\partial \phi}{\partial y} = -\frac{1}{2S_T} \sum_{i \in j_T} \phi_i \Delta x_i \tag{5.10}$$

$$\frac{\partial \psi}{\partial x} = \frac{1}{2S_T} \sum_{i \in j_T} \psi_i \Delta y_i, \quad \frac{\partial \psi}{\partial y} = -\frac{1}{2S_T} \sum_{i \in j_T} \psi_i \Delta x_i \tag{5.11}$$

which are all constants in an element. Substituting these and multiplying by S_T, we arrive at the following residuals of the Cauchy-Riemann equations in the physical plane.

$$U_T = \frac{1}{2} \sum_{i \in j_{T_p}} \phi_i \Delta y_i - \frac{1}{2} \sum_{i \in j_{T_p}} \psi_i \Delta x_i \tag{5.12}$$

$$V_T = \frac{1}{2} \sum_{i \in j_{T_p}} \psi_i \Delta y_i + \frac{1}{2} \sum_{i \in j_{T_p}} \phi_i \Delta x_i \tag{5.13}$$

This particular formulation applies to the hodograph equations also. Consider the image $\{T_h\}$ of the triangulation $\{T\}$ onto the hodograph plane. On the assumption that x and y vary linearly within each element with respect to ϕ and ψ, we obtain

the first derivatives.

$$\frac{\partial x}{\partial \phi} = \frac{1}{2S_T^{'}} \sum_{i \in j_{T_h}} x_i \Delta\psi_i, \quad \frac{\partial x}{\partial \psi} = -\frac{1}{2S_T^{'}} \sum_{i \in j_{T_h}} x_i \Delta\phi_i \tag{5.14}$$

$$\frac{\partial y}{\partial \phi} = \frac{1}{2S_T^{'}} \sum_{i \in j_{T_h}} y_i \Delta\psi_i, \quad \frac{\partial y}{\partial \psi} = -\frac{1}{2S_T^{'}} \sum_{i \in j_{T_h}} y_i \Delta\phi_i \tag{5.15}$$

where $S_T^{'}$ is the element area in the hodograph plane given by

$$S_T^{'} = \frac{1}{2} \sum_{i \in j_{T_h}} \phi_i \Delta\psi_i = -\frac{1}{2} \sum_{i \in j_{T_h}} \psi_i \Delta\phi_i. \tag{5.16}$$

By inserting these into the hodograph equations and multiplying by S_h, we get

$$U_T^{'} = \frac{1}{2} \sum_{i \in j_{T_h}} x_i \Delta\psi_i - \frac{1}{2} \sum_{i \in j_{T_h}} y_i \Delta\phi_i \tag{5.17}$$

$$V_T^{'} = -\frac{1}{2} \sum_{i \in j_{T_h}} y_i \Delta\psi_i - \frac{1}{2} \sum_{i \in j_{T_h}} x_i \Delta\phi_i. \tag{5.18}$$

which can be written, by the antisymmetry implied in equation (5.9),

$$U_T^{'} = \frac{1}{2} \sum_{i \in j_{T_h}} \phi_i \Delta y_i - \frac{1}{2} \sum_{i \in j_{T_h}} \psi_i \Delta x_i = U_T \tag{5.19}$$

$$V_T^{'} = \frac{1}{2} \sum_{i \in j_{T_h}} \psi_i \Delta y_i + \frac{1}{2} \sum_{i \in j_{T_h}} \phi_i \Delta x_i = V_T. \tag{5.20}$$

Evidently, these are identical to those in the physical plane, (5.12) and (5.13). Hence, the residuals are unified representation of the error in both physical and hodograph planes. This suggests another interpretation of the moving grid method: by moving the grid, we are in effect attempting to solve the hodograph equations that govern the independent variables x and y.

5.1.3 The Least-Squares Method

We consider minimizing

$$\mathcal{F} = \sum_{T \in \{T\}} F_T = \sum_{T \in \{T\}} \frac{1}{2S_T} \left[{U_T}^2 + {V_T}^2 \right] \tag{5.21}$$

which is a norm with area weight. Based on the assumption of the linear variation, this can be written as

$$\mathcal{F} = \sum_{T\in\{T\}} \frac{1}{2} \iint_T \nabla\phi\cdot\nabla\phi\, dx\, dy + \sum_{T\in\{T\}} \frac{1}{2} \iint_T \nabla\psi\cdot\nabla\psi\, dx\, dy - \sum_{T\in\{T_h\}} S_T' \quad (5.22)$$

where S_T' denotes the area of the triangle in the hodograph plane. This norm is well known as energy norm in the variational formulation of the Laplace equations on which finite-element method is based. It is in fact equivalent to the Galerkin finite-element method with piecewise linear elements as shall be proven in the next section.

In the light of hodograph plane, we can also write this norm as

$$\mathcal{F} = \sum_{T\in\{T_h\}} \frac{J_T}{2} \iint_T \nabla x\cdot\nabla x\, d\phi\, d\psi + \sum_{T\in\{T_h\}} \frac{J_T}{2} \iint_T \nabla y\cdot\nabla y\, d\phi\, d\psi - \sum_{T\in\{T_h\}} S_T' \quad (5.23)$$

where J_T is the discrete version of the Jacobian in element T. This is not precisely the form of energy norm because of the presence of J_T. Therefore, the resulting method will be different for grids and solutions. This is not surprising because the norm (5.22) is biased towards the physical plane by the weight $1/S_T$. In other words, it is not symmetric with respect to physical and hodograph planes. It is not hard to see that the resulting method for grid movement would be nothing but an elliptic grid generation: a map between physical and hodograph planes. We will discuss this in detail later. A final remark is that the resulting moving mesh method is very similar to that developed by Tourigny and Hulsemann [91] in which the energy norm is directly utilized for mesh movement for a single Poisson's equation.

Minimization is performed by the method described in Chapter II. Accordingly, the update formulas are given by

$$\phi_j^{n+1} = \phi_j^n - c_\phi \frac{\partial\mathcal{F}}{\partial\phi_j}, \quad \psi_j^{n+1} = \psi_j^n - c_\psi \frac{\partial\mathcal{F}}{\partial\psi_j} \quad (5.24)$$

$$x_j^{n+1} = x_j^n - c_x \frac{\partial \mathcal{F}}{\partial x_j}, \quad y_j^{n+1} = y_j^n - c_y \frac{\partial \mathcal{F}}{\partial y_j} \tag{5.25}$$

where n is the iteration number and

$$\frac{\partial \mathcal{F}}{\partial \phi_j} = \sum_{T \in \{T_j\}} \frac{\partial F_T}{\partial \phi_j} = \sum_{T \in \{T_j\}} \frac{1}{2S_T} \left[\Delta y_T U_T + \Delta x_T V_T \right] \tag{5.26}$$

$$\frac{\partial \mathcal{F}}{\partial \psi_j} = \sum_{T \in \{T_j\}} \frac{\partial F_T}{\partial \psi_j} = \sum_{T \in \{T_j\}} \frac{1}{2S_T} \left[-\Delta x_T U_T - \Delta y_T V_T \right] \tag{5.27}$$

$$\frac{\partial \mathcal{F}}{\partial x_j} = \sum_{T \in \{T_j\}} \frac{\partial F_T}{\partial x_j} = \sum_{T \in \{T_j\}} \frac{1}{2S_T} \left[\Delta \psi_T U_T - \Delta \phi_T V_T - F_T \Delta y_T \right] \tag{5.28}$$

$$\frac{\partial \mathcal{F}}{\partial y_j} = \sum_{T \in \{T_j\}} \frac{\partial F_T}{\partial y_j} = \sum_{T \in \{T_j\}} \frac{1}{2S_T} \left[-\Delta \phi_T U_T - \Delta \psi_T V_T - F_T \Delta x_T \right] \tag{5.29}$$

where $\{T_j\}$ is a set of triangles that share vertex j and $\Delta\{\}_T$ denotes a difference taken counterclockwise along the side of triangle T. The diagonal scaling is incorporated into the coefficients.

$$c_\phi = \omega_u / \frac{\partial^2 \mathcal{F}}{\partial \phi_j^2}, \quad c_\psi = \omega_u / \frac{\partial^2 \mathcal{F}}{\partial \psi_j^2} \tag{5.30}$$

$$c_x = \omega_x / \frac{\partial^2 \mathcal{F}}{\partial x_j^2}, \quad c_y = \omega_x / \frac{\partial^2 \mathcal{F}}{\partial y_j^2} \tag{5.31}$$

where ω_u and ω_x are small constants, and

$$\frac{\partial^2 \mathcal{F}}{\partial \phi_j^2} = \frac{\partial^2 \mathcal{F}}{\partial \psi_j^2} = \sum_{T \in \{T_j\}} \frac{1}{4S_T} \left\{ (\Delta x_T)^2 + (\Delta y_T)^2 \right\} \tag{5.32}$$

$$\frac{\partial^2 \mathcal{F}}{\partial x_j^2} = \sum_{T \in \{T_j\}} \frac{1}{S_T} \left[\frac{1}{4} \{ (\Delta \phi_T)^2 + (\Delta \psi_T)^2 \} - \frac{\partial F_T}{\partial x_j} \Delta y_T \right] \tag{5.33}$$

$$\frac{\partial^2 \mathcal{F}}{\partial y_j^2} = \sum_{T \in \{T_j\}} \frac{1}{S_T} \left[\frac{1}{4} \{ (\Delta \phi_T)^2 + (\Delta \psi_T)^2 \} + \frac{\partial F_T}{\partial y_j} \Delta x_T \right]. \tag{5.34}$$

Note that the second term in (5.33) and (5.34) vanish at convergence. For faster convergence, updates are performed sequentially as described in Chapter II.

5.1.4 Equivalence with a finite-element method

Cauchy-Riemann System and Laplace Equations

Consider the Cauchy-Riemann equations for u and v

$$\partial_x u + \partial_y v = 0 \tag{5.35}$$

$$\partial_x v - \partial_y u = 0. \tag{5.36}$$

in a domain Ω, with the boundary conditions

$$\partial_n u = \partial_s v \tag{5.37}$$

$$v = g(x, y) \tag{5.38}$$

on $\partial\Omega$, where $g(x, y)$ is a given function, ∂_n is the derivative normal to the boundary and ∂_s is the derivative along the boundary. The variables u and v are left arbitrary; it is not important here. Note, however, that the condition (5.37) will become irrotationality condition if u and v are taken as velocity components in x and y directions respectively. It is well-known that the Cauchy-Riemann system implies a set of Laplace equations

$$-\nabla^2 u = 0 \tag{5.39}$$

$$-\nabla^2 v = 0 \tag{5.40}$$

in Ω with the same boundary conditions. We will show that the least-squares method applied to the Cauchy-Riemann system is equivalent to the Galerkin finite-element method for the two Laplace equations.

Basis Functions

Although it is not necessary to introduce the concept of basis functions in the case of least-squares method, it is useful to define local piecewise linear interpolat-

ing functions when the two methods are compared. For this purpose, here we will establish the relation between the two functions.

We begin by triangulating the domain Ω into a set of triangles $\{T\}$ and nodes $\{J\}$ consisting in the interior nodes $\{J_i\}$ and the boundary nodes $\{J_b\}$. A group of triangles that shares node j is denoted by $\{T_j\}$, equivalently this local domain will be denoted also by Ω_j with its boundary being $\partial\Omega_j$. We also introduce the notations N_i and N_b for the number of interior nodes and boundary nodes respectively. The total number of nodes N is then equal to $N_i + N_b$.

In finite-element methods, the solution is sought in a vector subspace which is spanned by a set of basis functions denoted by $\phi_j(x,y)$ $j = 1, 2, ...N$. Each $\phi_j(x,y)$ is unity at the vertex j , zero on $\partial\Omega_j$, and varies linearly within each element belonging to $\{T_j\}$. The approximate solutions u_h and v_h are then represented by

$$u_h = \sum_{i\in\{J\}} u_i\,\phi_i(x,y) \tag{5.41}$$

$$v_h = \sum_{i\in\{J\}} v_i\,\phi_i(x,y) \tag{5.42}$$

where u_i and v_i are the nodal values of u and v. On the other hand, the functions useful in the least-squares method are piecewise linear interpolating functions $N_i^T(x,y)$ as described in the last section: $N_i^T(x,y)$ is a linear function that takes the value unity at the vertex i and zero at the others. The solutions are then written within each element

$$u^T = \sum_{i\in\{i_T\}} u_i\,N_i^T(x,y) \tag{5.43}$$

$$v^T = \sum_{i\in\{i_T\}} v_i\,N_i^T(x,y). \tag{5.44}$$

where $\{i_T\}$ is a set of vertices that defines the element T. Important relations are

$$u^T = u_h|_T, \quad v^T = v_h|_T \tag{5.45}$$

where $|_T$ means the restriction on the element T.

Finite Element Method

Finite-element method begins with variational formulations for the pair of Laplace equations (5.39) and (5.40). Let U and V be the trial spaces for u and v respectively defined by

$$U = \{u \in H^1(\Omega)\} \tag{5.46}$$

$$V = \{v \in H^1(\Omega) \mid v = g(x,y) \text{ on } \partial\Omega\} \tag{5.47}$$

where $H^1(\Omega)$ denotes the Sobolev space defined by

$$H^1(\Omega) = \{v : \int_\Omega (|\nabla v|^2 + v^2)\ dxdy < \infty\} \tag{5.48}$$

and U will be used also as the test space for u. For v, we define the test space V_0 by

$$V_0 = \{v_0 \in H^1(\Omega) \mid v_0 = 0 \text{ on } \partial\Omega\}. \tag{5.49}$$

The variational forms are easily found by multiplying the Laplace equations by test functions $u_0 \in U$ and $v_0 \in V_0$, integrating over Ω, and by using Green's formula. We are then led to the following variational formulations: find $u \in U$ and $v \in V$ such that

$$\iint_\Omega \nabla u \cdot \nabla u_0\ dxdy - \oint_{\partial\Omega} \frac{\partial v}{\partial s} u_0\ ds = 0 \quad \forall u_0 \in U \tag{5.50}$$

$$\iint_\Omega \nabla v \cdot \nabla v_0\ dxdy = 0 \quad \forall v_0 \in V_0 \tag{5.51}$$

where we have used the Neumann condition (5.37) to rewrite the boundary integral for u. Note that the implementation of the Neumann boundary condition is not automatic here.

We discretize these problems on piecewise linear triangular elements. The finite element subspaces for (5.46), (5.47) and (5.49) are denoted by U_h, V_h and V_{0h} respectively, with the basis function ϕ_j that is associated with node $j \in \{J\}$. Choosing

this basis function as a test function, we obtain the following discrete problems: find $u_h \in U_h$ and $v_h \in V_h$ such that

$$\iint_\Omega \nabla u_h \cdot \nabla \phi_j \; dxdy - \oint_{\partial\Omega} \frac{\partial v_h}{\partial s} \phi_j \; ds = 0 \quad \forall j \in \{J\} \tag{5.52}$$

$$\iint_\Omega \nabla v_h \cdot \nabla \phi_j \; dxdy = 0 \quad \forall j \in \{J_i\}. \tag{5.53}$$

Least-squares Method

In the least-squares method, we begin by defining the residuals Ω_T and Δ_T as the integrals of (1) and (2) over each element.

$$\Delta_T = \iint_T \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dxdy = \left(\frac{\partial u^T}{\partial x} + \frac{\partial v^T}{\partial y} \right) S_T \tag{5.54}$$

$$\Omega_T = \iint_T \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) dxdy = \left(\frac{\partial v^T}{\partial x} - \frac{\partial u^T}{\partial y} \right) S_T \tag{5.55}$$

where the second equalities are due to the linear approximations of u and v. We then define the norm $\mathcal{F}$ to be minimized by

$$\mathcal{F} = \sum_{T \in \{T\}} F_T = \sum_{T \in \{T\}} \frac{1}{2S_T} \left[\Delta_T^2 + \Omega_T^2 \right] \tag{5.56}$$

which can be expanded as

$$\begin{aligned} \mathcal{F} = & \quad \frac{1}{2} \sum_{T \in \{T\}} \left[\left(\frac{\partial u^T}{\partial x} \right)^2 + \left(\frac{\partial u^T}{\partial y} \right)^2 \right] S_T + \frac{1}{2} \sum_{T \in \{T\}} \left[\left(\frac{\partial v^T}{\partial x} \right)^2 + \left(\frac{\partial v^T}{\partial y} \right)^2 \right] S_T \\ & + \sum_{T \in \{T\}} \left[\frac{\partial u^T}{\partial x} \frac{\partial v^T}{\partial y} - \frac{\partial u^T}{\partial y} \frac{\partial v^T}{\partial x} \right] S_T \end{aligned} \tag{5.57}$$

or

$$\begin{aligned} \mathcal{F} = & \quad \sum_{T \in \{T\}} \frac{1}{2} \iint_T \nabla u^T \cdot \nabla u^T \, dx \, dy + \sum_{T \in \{T\}} \frac{1}{2} \iint_T \nabla v^T \cdot \nabla v^T \, dx \, dy \\ & + \sum_{T \in \{T\}} \left[\frac{\partial u^T}{\partial x} \frac{\partial v^T}{\partial y} - \frac{\partial u^T}{\partial y} \frac{\partial v^T}{\partial x} \right] S_T. \end{aligned} \tag{5.58}$$

By (5.45), we have

$$\mathcal{F} = \frac{1}{2}\iint_{\Omega} \nabla u_h \cdot \nabla u_h \, dx dy + \frac{1}{2}\iint_{\Omega} \nabla v_h \cdot \nabla v_h \, dx dy + \sum_{T \in \{T\}} \left[\frac{\partial u^T}{\partial x}\frac{\partial v^T}{\partial y} - \frac{\partial u^T}{\partial y}\frac{\partial v^T}{\partial x} \right] S_T. \tag{5.59}$$

Now there are two arrangements we can make for the last term. The first option is to use an algebraic identity given by

$$S_T^{'} = \left[\frac{\partial u^T}{\partial x}\frac{\partial v^T}{\partial y} - \frac{\partial u^T}{\partial y}\frac{\partial v^T}{\partial x} \right] S_T \tag{5.60}$$

where $S_T^{'}$ is the area of the image of the element T in the solution space (u, v). This is a discrete analog of

$$du\, dv = (\partial_x u \partial_y v - \partial_y u \partial_x v)\, dx\, dy \tag{5.61}$$

which holds true exactly for piecewise linear functions. It now follows immediately that the last term reduces to the entire area of the domain in the solution space. Hence we can write

$$\mathcal{F} = \frac{1}{2}\iint_{\Omega} \nabla u_h \cdot \nabla u_h \, dx\, dy + \frac{1}{2}\iint_{\Omega} \nabla v_h \cdot \nabla v_h \, dx\, dy + S_{\mathcal{H}} \tag{5.62}$$

where $S_{\mathcal{H}}$ represents the entire area of the domain in the solution space. It is important to note that $S_{\mathcal{H}}$ involves only the boundary values of u_h and v_h. The second option is to use the relations (5.45) again. This gives

$$\mathcal{F} = \frac{1}{2}\iint_{\Omega} \nabla u_h \cdot \nabla u_h \, dx dy + \frac{1}{2}\iint_{\Omega} \nabla v_h \cdot \nabla v_h \, dx dy + \iint_{\Omega} \left[\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial y} - \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial x} \right] dx dy. \tag{5.63}$$

To simplify the last integral, we first use the following identity

$$\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial y} - \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial x} = \frac{\partial}{\partial x}\left(u_h \frac{\partial v_h}{\partial y} \right) - \frac{\partial}{\partial y}\left(u_h \frac{\partial v_h}{\partial x} \right) \tag{5.64}$$

to rewrite the integrand, and then use Green's formula to obtain

$$\mathcal{F} = \frac{1}{2}\iint_{\Omega} \nabla u_h \cdot \nabla u_h \, dx\, dy + \frac{1}{2}\iint_{\Omega} \nabla v_h \cdot \nabla v_h \, dx\, dy + \oint_{\partial\Omega} u_h \frac{\partial v_h}{\partial s} ds. \tag{5.65}$$

Note that the least-squares method implements the Neumann condition (5.37) automatically.

To find a minimum, we need to solve the following equations

$$\frac{\partial \mathcal{F}}{\partial u_j} = 0 \quad \forall\, j \in \{J\} \tag{5.66}$$

$$\frac{\partial \mathcal{F}}{\partial v_j} = 0 \quad \forall\, j \in \{J_i\} \tag{5.67}$$

for the nodal unknowns. Now we substitute (5.65) into the first equation and (5.62) into the second equation to get

$$\frac{\partial \mathcal{F}}{\partial u_j} = \iint_\Omega \nabla u_h \cdot \nabla \phi_j \, dx \, dy - \oint_{\partial\Omega} \phi_j \frac{\partial v_h}{\partial s} ds = 0 \quad \forall\, j \in \{J\} \tag{5.68}$$

$$\frac{\partial \mathcal{F}}{\partial v_j} = \iint_\Omega \nabla v_h \cdot \nabla \phi_j \, dx \, dy = 0 \quad \forall\, j \in \{J_i\}. \tag{5.69}$$

These equations are identical to those of finite element method (5.52) and (5.53), and this completes the proof of the equivalence.

We have proved the equivalence of the least-squares method for the Cauchy-Riemann system with the Galerkin finite element method for the associated Laplace equations. The resulting linear systems are identical, and therefore have the same unique solution regardless of the choice of the solution algorithm. In particular, if an iterative algorithm is used such as Gauss-Seidel, which can be interpreted also as a variant of the steepest descent method when applied to the least-squares method, discrete updates will be the same, within a relaxation factor.

Also shown is that the Neumann boundary condition is automatically incorporated into the least-squares method. The implication of this fact is that the Laplace equations are solved with irrotationality condition on a boundary when we choose the variables u and v to be velocity components. We will return to this point later when we discuss Cauchy-Riemann equations for velocity components.

5.1.5 Grid Movement

Returning to the Cauchy-Riemann equations for velocity potential ϕ and stream-function ψ, we now understand that we are in effect solving Laplace equations for ϕ and ψ by minimizing the least-squares norm. In view of grid movement, this implies that we are solving the Laplace equations inversely for independent variables x and y by minimizing the least-squares norm, which is nothing but the principle of elliptic grid generation methods. The solution contours of a pair of Laplace equations, say for ϕ and ψ,

$$\phi_{xx} + \phi_{yy} = 0 \tag{5.70}$$

$$\psi_{xx} + \psi_{yy} = 0 \tag{5.71}$$

on a physical domain are smooth and nonintersecting, thus suitable for forming a computational grid. But it is impossible to solve the equations on the physical plane because the physical grid is what we want to generate and not yet available. Therefore, we solve the Laplace equations for x and y on a fixed grid in ϕ-ψ plane. For this purpose, it is customary to transform (5.70) and (5.71) into the governing equations for x and y

$$\alpha x_{\phi\phi} - 2\beta x_{\phi\eta} + \gamma x_{\psi\psi} = 0 \tag{5.72}$$

$$\alpha y_{\phi\phi} - 2\beta y_{\phi\eta} + \gamma y_{\psi\psi} = 0 \tag{5.73}$$

where

$$\alpha = x_\psi^2 + y_\psi^2, \quad \beta = x_\phi x_\psi + y_\phi y_\psi, \quad \gamma = x_\phi^2 + y_\phi^2, \tag{5.74}$$

and this complex nonlinear system is then solved by a finite-difference method, usually on a rectangular domain in ϕ-ψ plane. In the least-squares method, we are doing exactly the same thing, but in a much simpler way, i.e. by minimizing the norm with

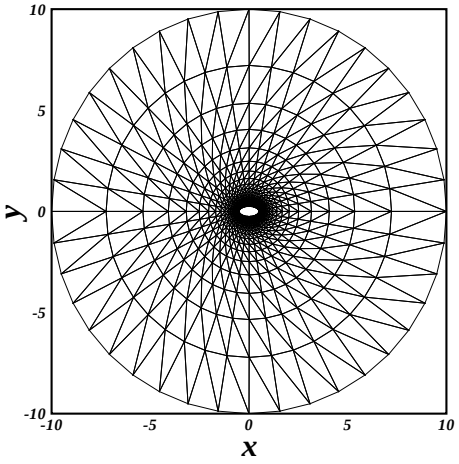

Figure 5.1: A 40x20 O-grid generated by the least-squares method.

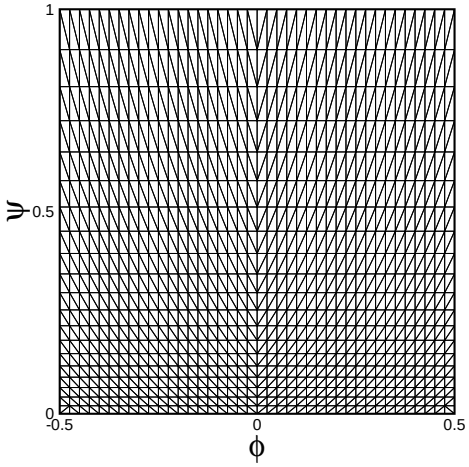

Figure 5.2: The grid in the solution space with an exponential stretching in the ψ direction

respect to x and y. As an example, Figure 5.1 shows an O-grid around an ellipse generated by the least-squares method with a fixed regular triangular grid in ϕ-ψ plane as in Figure 5.2.

This equivalence implies also that elliptic grid generation can be easily applied to any types of unstructured grids, quadrilateral, triangular or even mixed ones based on the least-squares approach while the conventional elliptic grid generation appears to be applicable to structured quadrilateral grids only. In order for the method to be practical, however, there must be introduced some mechanisms to control the grid qualities such as boundary orthogonality, which is common in the elliptic grid generation methods. In the least-squares method, this could be realized by introducing source terms in the Cauchy-Riemann equations.

In the context of the least-squares moving grid method, the movement of the grid is therefore nothing but an elliptic grid generation. However, the grid in the hodograph plane is also allowed to evolve by the standard finite-element method, and

so the method attempts to find a simultaneous solution of the finite-element method and the elliptic grid generation.

It is interesting that the mechanism of the grid movement by elliptic grid generation can be explained by the spring analogy that is a well-known technique for any node-moving algorithms as discussed in Blom[14] for quadrilateral grids. We show this equivalence in terms of the least-squares method for triangular grids. Recall that the gradient of the norm with respect to the nodal coordinates $\mathbf{x}_j = (x_j, y_j)$ is given by

$$\frac{\partial \mathcal{F}}{\partial \mathbf{x}_j} = \sum_{T \in \{T_j\}} \frac{1}{2S_T} \begin{bmatrix} \Delta\psi_T U_T - \Delta\phi_T V_T - F_T \Delta y_T \\ -\Delta\phi_T U_T - \Delta\psi_T V_T + F_T \Delta x_T \end{bmatrix} \tag{5.75}$$

Note that the third term in each pair is in the direction of the outward normal to the edge opposite to the node j within each T. This will contribute to preventing vanishing area, for the change made to each vertex is proportional to the negative gradient. Expand the first component of (5.75), which drives the corrections to x, as

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial x_j} &= \frac{1}{2} \sum_{T \in \{T_j\}} \frac{\Delta\psi_T}{S_T} U_T - \frac{1}{2} \sum_{T \in \{T_j\}} \frac{\Delta\phi_T}{S_T} V_T - \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta y_T & (5.76) \\ &= \frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\psi_T}{S_T} \sum_{i \in j_T} \phi_i \Delta y_i - \frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\psi_T}{S_T} \sum_{i \in j_T} \psi_i \Delta x_i & (5.77) \\ &\quad -\frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\phi_T}{S_T} \sum_{i \in j_T} \psi_i \Delta y_i - \frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\phi_T}{S_T} \sum_{i \in j_T} \phi_i \Delta x_i - \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta y_T, \end{aligned}$$

or

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial x_j} &= -\frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\psi_T}{S_T} \sum_{i \in j_T} y_i \Delta\phi_i + \frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\psi_T}{S_T} \sum_{i \in j_T} x_i \Delta\psi_i & (5.78) \\ &\quad +\frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\phi_T}{S_T} \sum_{i \in j_T} y_i \Delta\psi_i + \frac{1}{4} \sum_{T \in \{T_j\}} \frac{\Delta\phi_T}{S_T} \sum_{i \in j_T} x_i \Delta\phi_i - \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta y_T \quad . \end{aligned}$$

By using the following formulas that hold within a triangle for any nodal quantity a_i (Figures 5.3),

$$\Delta\psi_j \sum_{i \in j_T} a_i \Delta\psi_i + \Delta\phi_j \sum_{i \in j_T} a_i \Delta\phi_i = \vec{l}_j \cdot \vec{l}_j a_j + \vec{l}_1 \cdot \vec{l}_j a_1 + \vec{l}_2 \cdot \vec{l}_j a_2 \tag{5.79}$$

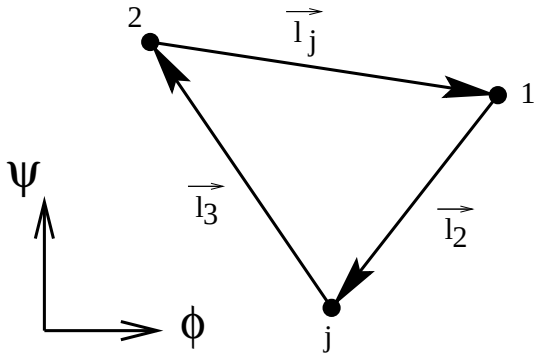

Figure 5.3: Definition of side vectors for a triangle T.

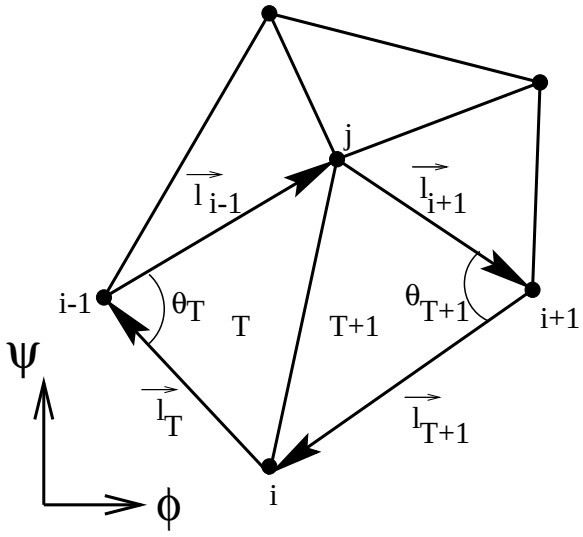

Figure 5.4: Definitions of edge vectors and angles in a group of triangles, $\{T_j\}$.

$$\Delta\psi_j \sum_{i\in j_T} a_i \Delta\phi_i - \Delta\phi_j \sum_{i\in j_T} a_i \Delta\psi_i \;=\; 2S_T^{'} \Delta a_j \tag{5.80}$$

we obtain [1](cf. Figures 5.4)

$$\frac{\partial \mathcal{F}}{\partial x_j} = \left(\frac{1}{4} \sum_{T\in T_j} \frac{\vec{l}_T \cdot \vec{l}_T}{S_T} \right) x_j + \frac{1}{4} \sum_{i\in i_j} \left(\frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S_T} + \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S_{T+1}} \right) x_i - \frac{1}{2} \sum_{T\in T_j} \Delta y_T - \sum_{T\in\{T_j\}} \frac{F_T}{2S_T} \Delta y_T. \tag{5.81}$$

Note that the side vectors are evaluated in the hodograph plane, but that S_T is in the physical plane. Similarly for y, we have

$$\frac{\partial \mathcal{F}}{\partial y_j} = \left(\frac{1}{4} \sum_{T\in T_j} \frac{\vec{l}_T \cdot \vec{l}_T}{S_T} \right) y_j + \frac{1}{4} \sum_{i\in i_j} \left(\frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S_T} + \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S_{T+1}} \right) y_i + \frac{1}{2} \sum_{T\in T_j} \Delta x_T + \sum_{T\in\{T_j\}} \frac{F_T}{2S_T} \Delta x_T. \tag{5.82}$$

Evidently the third sum identically vanishes for interior nodes. Then, we write, using the Jacobian $J_T S_T = S_T^{'}$,

$$\frac{\partial \mathcal{F}}{\partial x_j} \;=\; \left(\frac{1}{4} \sum_{T\in\{T_{h_j}\}} J_T \frac{\vec{l}_T \cdot \vec{l}_T}{S_T^{'}} \right) x_j + \frac{1}{4} \sum_{i\in i_j} \left(J_T \frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S_T^{'}} + J_{T+1} \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S_{T+1}^{'}} \right) x_i - \sum_{T\in\{T_j\}} \frac{F_T}{2S_T} \Delta y_T$$

[1]This is obtainable also by differentiating (5.23) directly. At least, the last term is easily obtained this way.

$$\frac{\partial \mathcal{F}}{\partial y_j} = \left(\frac{1}{4} \sum_{T \in \{T_{h_j}\}} J_T \frac{\vec{l}_T \cdot \vec{l}_T}{S'_T} \right) y_j + \frac{1}{4} \sum_{i \in i_j} \left(J_T \frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S'_T} + J_{T+1} \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S'_{T+1}} \right) y_i + \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta x_T$$

Expressing the vector $\vec{l}_T$ by the other two edge vectors in a triangle T, and converting the sum over the triangles to the one over the surrounding nodes, we find their alternative forms.

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial x_j} &= \frac{1}{4} \sum_{i \in i_j} \left(J_T \frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S'_T} + J_{T+1} \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S'_{T+1}} \right) (x_i - x_j) - \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta y_T \\
\frac{\partial \mathcal{F}}{\partial y_j} &= \frac{1}{4} \sum_{i \in i_j} \left(J_T \frac{\vec{l}_{i-1} \cdot \vec{l}_T}{S'_T} + J_{T+1} \frac{\vec{l}_{i+1} \cdot \vec{l}_{T+1}}{S'_{T+1}} \right) (y_i - y_j) + \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta x_T
\end{aligned}$$

Furthermore, we can write also

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial x_j} &= \frac{1}{4} \sum_{i \in i_j} \left(J_T \cot\theta_T + J_{T+1} \cot\theta_{T+1} \right) (x_i - x_j) - \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta y_T \\
\frac{\partial \mathcal{F}}{\partial y_j} &= \frac{1}{4} \sum_{i \in i_j} \left(J_T \cot\theta_T + J_{T+1} \cot\theta_{T+1} \right) (y_i - y_j) + \sum_{T \in \{T_j\}} \frac{F_T}{2S_T} \Delta x_T
\end{aligned}$$

which can be proved easily by combining the definition of dot product and a formula for a triangle area (two sides and the sine of the angle in between). At a minimum, these gradients vanish for every node which can be realized only by iterations because the equations are nonlinear in (x_j, y_j). Hence, we may solve the equations for x_j and y_j, freezing x_j and y_j that do not appear explicitly. By setting the gradients equal to zero and solving for x_j and y_j[2], we obtain

$$\mathbf{x}_j = \frac{\sum_{i \in i_j} \alpha_i \mathbf{x}_i}{\sum_{i \in i_j} \alpha_i} - \frac{1}{2} \frac{\sum_{T \in \{T_j\}} F_T / S_T}{\sum_{i \in i_j} \alpha_i} \mathbf{n}_T \tag{5.83}$$

[2]This is possible only if the Jacobian is nonzero, but it is the case for Cauchy-Riemann system, except for trivial solutions of course.

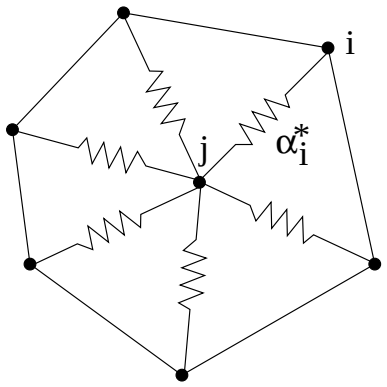

Figure 5.5: A network of springs.

where $\mathbf{x}_j = (x_j, y_j)$, $\mathbf{n}_T$ is a scaled inward normal vector of the edge opposite to the node j for triangle T, and

$$\alpha_i = \frac{1}{4}\left(J_T \cot\theta_T + J_{T+1}\cot\theta_{T+1}\right). \tag{5.84}$$

Yet, note that the Jacobian is always negative for our Cauchy-Riemann equations (5.3), and therefore we can write.

$$\mathbf{x}_j = \frac{\sum_{i\in i_j}\alpha_i^*\mathbf{x}_i}{\sum_{i\in i_j}\alpha_i^*} + \frac{1}{2}\frac{\sum_{T\in\{T_j\}}F_T/S_T}{\sum_{i\in i_j}\alpha_i^*}\mathbf{n}_T \tag{5.85}$$

where

$$\alpha_i^* = \frac{1}{4}\left(|J_T|\cot\theta_T + |J_{T+1}|\cot\theta_{T+1}\right). \tag{5.86}$$

Finally, we have, in an iterative form,

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \frac{\sum_{i\in i_j}\alpha_i^*(\mathbf{x}_i^n - \mathbf{x}_j^n)}{\sum_{i\in i_j}\alpha_i^*} + \frac{1}{2}\frac{\sum_{T\in\{T_j\}}F_T/S_T}{\sum_{i\in i_j}\alpha_i^*}\mathbf{n}_T \tag{5.87}$$

where n is the iteration number. Note that the last two terms on the right are to be multiplied by a small constant (the relaxation factor) in the actual update. The first term is precisely the form of the spring analogy. In particular, this is of the vertex-spring type with the stiffness defined by (5.86). See Figure 5.5. First of all, we see that the changes to x_j and y_j are thus biased toward the elements

with large Jacobian, which is a desirable property for grid adaptation. Second of all, the nodes will move towards the elements with small angle θ_T in the hodograph plane which is controlled by the cotangent functions. In other words, a large weight will be given to nodes closer to the node j in the hodograph plane. In fact, this is a property of Laplacian, and therefore a natural consequence of the least-squares method. Finally, the second term is the one that compete for area. It has an effect to enlarge small elements. This is in fact equivalent to the term introduced by Palmerio and Dervieux [72] for the purpose of preventing mesh folding in grid movement by a spring analogy. They added the terms, arising from minimizing $1/S_T$ with a small parameter multiplied, to the mesh movement formula based on a spring analogy.

We note here that the iterative formula (5.87) is similar to the update formula that would have been obtained from the approach described in 5.1.3, but slightly different. This iterative formula is missing in the denominator the additional term that comes from differentiating the second term of the gradient, but our experience shows that its effect is so small that the two updates do not differ significantly. This makes sense because the missing term is directly proportional to the gradient, and therefore vanishes at convergence. At any rate, the difference is merely a matter of linearization.

5.1.6 Results

Numerical experiments were conducted for a two-dimensional flow in a square domain with a source in one corner and a sink in the other. There is an exact solution for this problem which is expressed in terms of the complex potential $F(z)$,

$$F(z) = \phi + i\psi = m \ln \left[\sinh\left(\frac{\pi z}{2a}\right) \frac{\prod_{k=\text{even}} \left(1 - \sinh^2 \frac{\pi z}{2} / \sinh^2 \frac{k\pi}{2a}\right)}{\prod_{k=\text{odd}} \left(1 - \sinh^2 \frac{\pi z}{2} / \sinh^2 \frac{k\pi}{2a}\right)} \right] \tag{5.88}$$

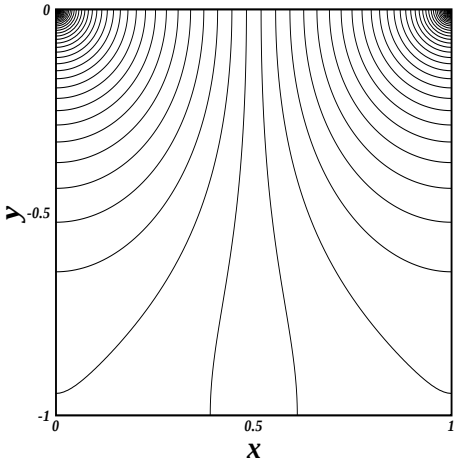

Figure 5.6: Contour plot (70 Levels) of the exact solution (k=22) of ϕ.

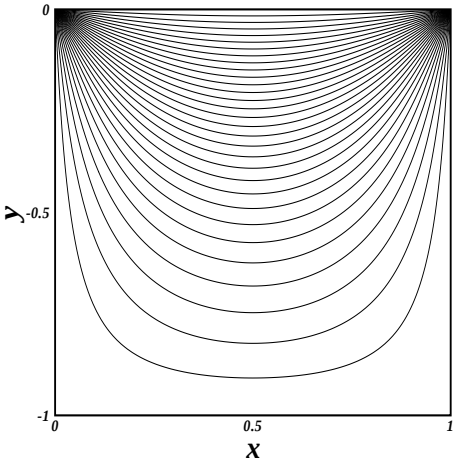

Figure 5.7: Contour plot (30 Levels) of the exact solution (k=22) of ψ.

where $z = x + iy$ and m determines the source(and sink) strength. In fact, this solution produces a series of square regions which are divided by streamlines. The solution can be shown to converge quite rapidly, so that $k = 22$ which we used is sufficient. We choose the domain of interest Ω as the unit square whose upper left corner coincides with the origin, i.e. $\Omega = \{(x, y)|\ 0 \leq x \leq 1,\ -1 \leq y \leq 0\}$. The source strength was chosen to be $m = -\frac{2}{\pi}$ and a was set to be 1, then the stream function takes the value of zero when $y = 0$ and the value of unity elsewhere on the boundary. These were used as the boundary conditions for our numerical experiments. And also the quantity inside the square bracket was divided by itself with $z = \frac{a}{2}$ inserted, so that it becomes unity at $x = \frac{a}{2}$, i.e. the potential becomes zero at the mid point of the upper boundary which makes the solution unique. The exact solutions in Ω are shown in Figures (5.8) and (5.9). The most difficult part of this problem is the singularities at two upper corners: ϕ becomes infinite and ψ becomes multi-valued. For this reason, the solutions at these corners are included

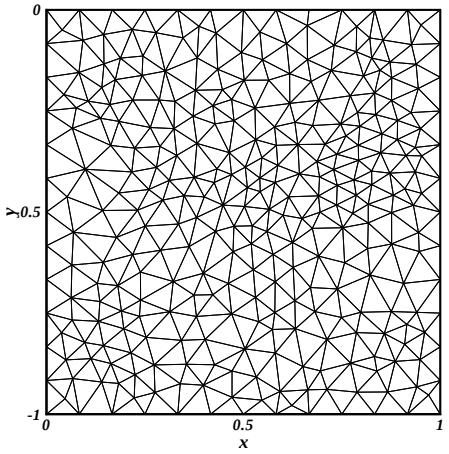

Figure 5.8: Initial Delaunay grid with 568 triangles and 309 nodes.

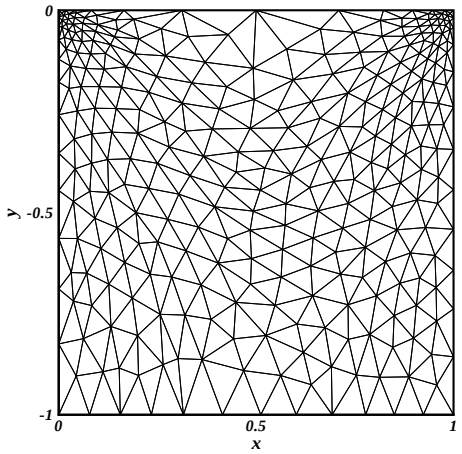

Figure 5.9: Adaptive grid

among the unknowns. For the initial values for ϕ and ψ, we always use the values of zero for both. Diagonal swapping is performed for every 50 iteration. With the diagonal scaling and the Gauss-Seidel iteration described in Section 5.1.3, the scheme becomes indistinguishable from SOR for ϕ and ψ. Therefore, we already know that the best value of ω_u lies between 1.0 and 2.0 and approaches 2.0 as the size of the problem increases. On the other hand, for grid movement we found that it is necessary to underrelax the iteration to prevent mesh tangling, and therefore in the actual computations, we used $\omega_x = 0.15$. The iteration may be taken to converge when the L1-norm of the gradient goes below the tolerance 10^{-5}. Boundary nodes must be moved also, for it is very likely that mesh movement inside the domain is very likely to create severe distortion near boundaries if the boundary nodes are fixed. However, we do not move the boundary nodes by the least-squares method because we found that their movement was not consistent with that of internal nodes nearby each boundary node. Similar observation has been reported in a moving mesh

	Fixed grid	Moving grid
Iteration number	444	20000
$L_2(\psi)$	1.22999E-02	3.02695E-03
$L_2(\phi)$	2.40661E-02	6.16447E-03
Norm	1.18088E-01	1.47477E-01
$\mathcal{F}'$	2.74647E-04	4.88475E-05

Table 5.1: The results of the numerical experiments

method for interpolating a known function[5]. Here, we instead move the boundary nodes, except for the four corners, by a Laplacian smoothing. That is, we compute the average position of the surrounding nodes, and project it onto the boundary so that the nodes remain on the boundary. Equivalently, we may form a mirror image, outside the boundary, of the group of triangles that share a boundary node, and then apply a Laplacian smoothing. Although this is quite effective in the sense that mesh distortion created near the boundary by the movement of the internal nodes is removed, it does not necessarily minimize the norm. Therefore we cannot necessarily achieve convergence. Our procedure is then first compute a solution on a fixed grid, which is going to be our initial solution, and then apply the least-squares moving grid method for some specified number of iterations, and finally compute the solution again with the grid fixed. In this particular example, the moving grid algorithm was applied for 20000 iterations.

Shown in Figures 5.8 and 5.9 are the initial Delaunay grid and the final grid respectively. As we expected, the nodes have moved towards the upper two corners to better resolve the rapid variations of the solutions. The errors and other related quantities are compared between these grids in Table 5.1. Firstly, we see that the error has been reduced significantly for both solutions. Because of the presence of

singularity in the solutions, the error is measured in the L_2 norm defined by

$$L_2(\phi) = \sum_{T \in \{T\}} \int_T (\phi_{\text{exact}} - \phi_{\text{numerical}})^2 \, dxdy \tag{5.89}$$

where the integration is performed by a Gaussian quadrature. It would be worth mentioning that the reduction of the errors was achieved only with diagonal swapping, emphasizing the importance of the connectivity. In fact, without diagonal swapping technique, the resulting grid would have suffered from highly skew and flat triangles with very low error reduction. Another important fact seen in Table 5.1 is that the norm is greater on the moving grid. This indicates that the minimizing the energy norm does not necessarily minimize the L_2 error although it is not a bad choice for grid adaptation. In fact, as is well known in finite-element methods, the residual is not the only error indicator: the jump in the first derivatives across linear elements is also directly related to the solution error[36]. On the other hand, the residuals have been reduced significantly as seen in the bottom of the table where $\mathcal{F}'$ is the unweighted norm,

$$\mathcal{F}' = \sum_{T \in \{T\}} \frac{1}{2} \left[{U_T}^2 + {V_T}^2 \right]. \tag{5.90}$$

These results suggest that we minimize not $\mathcal{F}$ but $\mathcal{F}'$. As mentioned earlier, it is difficult with this norm to move the grid because it does not offer any mechanism for competing for cell area. In practice, however, it does not deny the possibility for moving the grid; it just has to be done carefully. In the next section, we shall encounter the situation where we must use this unweighted norm.

5.2 Cauchy-Riemann Equations: $u - v$ Formulation

Alternatively, the equations that govern 2D incompressible and irrotational flows can be cast, using the velocity components, (u, v), in the form.

$$\partial_x u + \partial_y v = 0 \tag{5.91}$$

$$\partial_x v - \partial_y u = 0. \tag{5.92}$$

This form may be preferred possibly for two reasons. Firstly, in case of dealing with multiply-connected region, the velocity potential will be discontinuous when circulation is involved whereas the velocity components are continuous. Secondly, this system is a model system for the elliptic subsystem hidden in the steady-state Euler equations in two dimensions. It is also a role of the least-squares method for providing a good solver for this elliptic subsystem (See [78] for details).

5.2.1 A Problem of Computing Lifting Flows

It is straightforward to apply the least-squares method for this system. However, it was soon discovered that the least-squares method as a solver for u and v had a serious flaw especially in the case of a flow around a lifting body: the solution is extremely inaccurate for flows with nonzero circulation. To illustrate this problem, a flow around a Joukowsky airfoil at an angle of attack 10° was computed by the least-squares method with the area-weighted norm, on a 160x80 O-grid as shown in Figure 5.10 which should be fine enough to obtain accurate solutions. As the boundary conditions, we give exact solutions at the outer boundary[3] which is located at the distance of 10-chord-length away from the airfoil, and use a tangency condition on the

[3]This is of course not a practical condition since the circulation is not known in practice. But it is true also that if the method does not work with this condition, it will not work with the vortex correction[89]: a usual trick in practice.

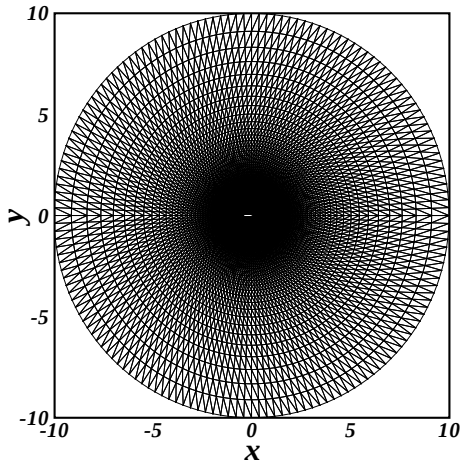

Figure 5.10: A 160x80 O-grid.

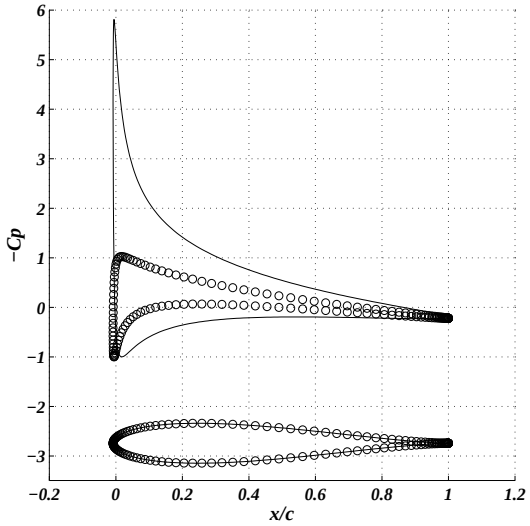

Figure 5.11: C_p distribution around the airfoil and the airfoil geometry. Solid curve represents the exact solution. Circles indicate the numerical solution.

airfoil which is simply to eliminate the component of the velocity normal to the body after each iteration. This tangency condition can be interpreted as a constraint to the minimization in the context of the least-squares method[80], which is equivalent also to replacing the variables (u, v) by a tangential velocity on the boundary as usually done in finite-element methods. At the cusped trailing edge, the solutions are computed without the tangency condition, i.e. the node is treated just like an interior node. The method converged for 9883 Gauss-Seidel iterations making the L_1 norm $(\sum_{nodes} |\cdot|/N_V)$ where N_V denotes the number of nodes except for outer boundary) of the updates below 1.0E-08. The pressure distribution over the airfoil is shown in Figure 5.11. It is evident that the numerical solution is far off the exact solution, yielding an extremely small lift. The circulation around the airfoil was found to be 1.92797E-01 whereas the one specified at the outer boundary is 5.99931E-01, implying nonzero vorticity inside the domain. The vorticity evaluated elementwise

is small indeed, but does not satisfy the integral constraint,

$$\iint \omega \, dxdy = \Gamma_{\text{out}} - \Gamma_{\text{airfoil}} \tag{5.93}$$

where the integration is all over the domain, and Γ denotes the circulation that is positive counterclockwise. This constraint is difficult to impose and in practice seldom enforced explicitly, whereas the continuity constraint, no net flux through the boundaries, can be easily ensured by the tangency condition. As shown by Baines[6] recently, the least-squares residual minimization can be interpreted as approximate equidistribution of the residuals, and therefore it may be inferred that the method merely redistributes nonzero residuals over the elements, and therefore the sum of the residuals are not necessarily small, thus creating a significant amount of gap between the outer and inner circulations (5.93). The result shows also that the scheme is not vorticity-preserving[68] because the iteration was started from the exact solutions. If it had been so, we could have obtained a very accurate value for the lift. Surprisingly, we observed from additional results on coarser grids that the method is not even first-order accurate: the numerical solution is converging to the exact one, but extremely slowly. Considering the nature of incompressible potential flows, we suspect that the method has a difficulty in computing a potential vortex that is the fundamental solution responsible for the circulation[4].

Consider solving the Cauchy-Riemann system in a domain bounded by two co-centric circles. At the outer boundary, a uniform tangential velocity is given with no cross flows, while the tangency condition is applied at the inner boundary. Obviously,

---

[4]It is a fundamental result of the theory of incompressible potential flows that the force on an object due to fluid motion comes exclusively from the fundamental solution of the Laplace equation that describes a vortex motion[66].

the exact solution is a potential vortex defined by

$$q = u_\theta = \frac{k}{r} \tag{5.94}$$

where $q^2 = u^2 + v^2$, $r^2 = x^2 + y^2$, and k is a constant related to the circulation. This is the only possible solution to the Cauchy-Riemann equations, but the associated Laplace equations allow another solution. In cylindrical coordinates, Laplace's equation for u reads

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = 0. \tag{5.95}$$

Assume the solution of the form

$$u = R(r)\Theta(\theta) \tag{5.96}$$

which leads to, on substitution and some arrangements,

$$-\frac{r^2}{R}\frac{d^2R}{dr^2} - \frac{r}{R}\frac{dR}{dr} = \frac{1}{\Theta}\frac{d^2\Theta}{d\theta^2} = -\beta^2 \tag{5.97}$$

where the negative constant $-\beta^2$ has been chosen for the periodicity in the θ direction. Therefore, we have

$$\frac{d^2\Theta}{d\theta^2} + \beta^2\Theta = 0 \tag{5.98}$$

$$r^2\frac{d^2R}{dr^2} + r\frac{dR}{dr} - \beta^2 R = 0 \tag{5.99}$$

where the first equation is a simple harmonic equation and the second is the Euler-Cauchy equation. The general solution is given by

$$u = (A_u r^{-\beta} + B_u r^{\beta})(C_u \cos(\beta\theta) + D_u \sin(\beta\theta)) \tag{5.100}$$

where A_u,B_u,C_u, and D_u are constants determined by boundary conditions. Similarly for v, we have

$$v = (A_v r^{-\beta} + B_v r^{\beta})(C_v \cos(\beta\theta) + D_v \sin(\beta\theta)) \tag{5.101}$$

where A_v,B_v,C_v, and D_v are arbitrary constants. These constants can be reduced by boundary conditions. Firstly, we have $C_u = 0$ because $u = 0$ at $\theta = 0$ on both boundaries. Then, we immediately see that β must be nonzero integers since u vanishes also at $\theta = \pi$. Secondly, $D_v = 0$ by the symmetry of v with respect to θ, by which we find also β to be odd integers since v vanishes at $\pm\pi/2$. Thirdly, the constancy of the flow speed along the outer boundary requires that the constants in u and v must be the same in magnitude, but with the opposite sign so that the velocity is tangent to the boundary, thus yielding

$$u = (Ar^{-\beta} + Br^{\beta})\sin(\beta\theta) \tag{5.102}$$

$$v = -(Ar^{-\beta} + Br^{\beta})\cos(\beta\theta). \tag{5.103}$$

Finally, the tangency condition at the inner boundary, $u\cos\theta + v\sin\theta = 0$, determines that $\beta = 1$, and therefore we obtain

$$u = \left(\frac{A}{r} + Br\right)\sin\theta \tag{5.104}$$

$$v = -\left(\frac{A}{r} + Br\right)\cos\theta \tag{5.105}$$

where A corresponds to k in (5.94). This shows that the Laplace equations have another solution component proportional to r. It is easy to eliminate this solution analytically (just require that the solution be bounded as $r \to \infty$), but not easy to do so numerically. This solution represents a solid body rotation and has a nonzero vorticity given by $2B$. Therefore, it does not satisfy one of the Cauchy-Riemann equations, but it does satisfy the Laplace equations as just shown above. Recall that the least-squares method is equivalent to solving the Laplace equations by the Galerkin finite-element method. Then, it would be reasonable to conjecture that this is the solution that the least-squares scheme mistakenly computes. In fact, we

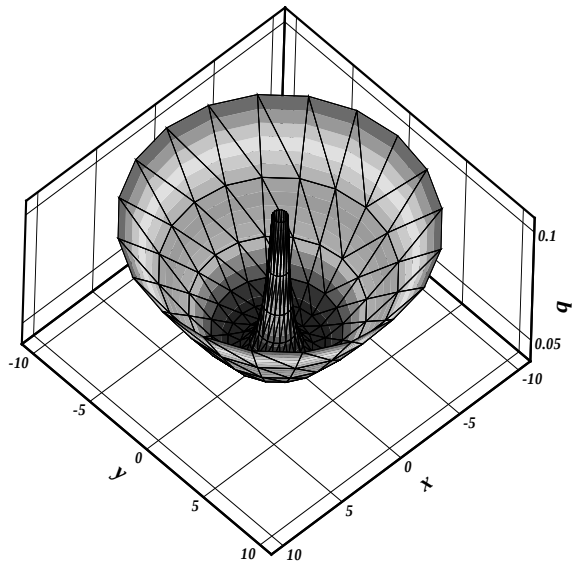

Figure 5.12: A mesh plot of the flow speed q. A 20x10 O-grid.

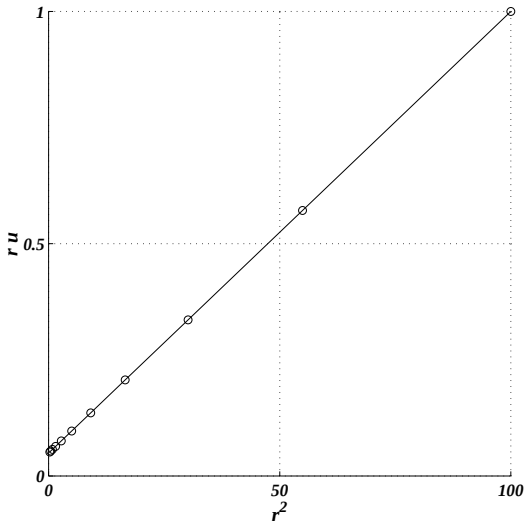

Figure 5.13: ru versus r^2 along the radial grid line at $\theta = 90°$.

found by a simple computation that this was exactly what was happening. Figure 5.12 shows a mesh plot of the flow speed obtained by the least-squares method on a 20x10 O-grid with two circular boundaries located at $r = 0.5$ and $r = 10$, for the exact solution with $k = A = 1$. As clearly seen, the flow speed decays from the inner boundary(correct behavior), but soon starts to rise towards the outer boundary(wrong behavior). The reason for the correct behavior near the inner boundary is that the scheme automatically implements the zero vorticity condition as shown in Section 5.1.4. But it is not effective enough to completely suppress the wrong solution component, and the solution is extremely inaccurate at the inner boundary where $q \approx 0.1$ whereas the exact solution is $q = 2$. The circulation along the inner boundary is 3.19607E-01, which is too small compared to the value at the outer boundary 6.18034E+00. The presence of the two types of solutions was confirmed by Figure 5.13 which shows a plot of ru verses r^2 along a radial grid line. The plot shows a perfect straight line which implies $ru = a + br^2$ for some constants a and b

and which in turn yields (5.104).

These results suggest that we should find a way to solve the Cauchy-Riemann equations, not the Laplace equations. But remember that the problem is specific to triangular grids. There is no problem with the least-squares method on quadrilateral grids. The method is actually capable of producing the exact solution to this problem. The reason is that a bilinear variation on a quadrilateral cell can represent the solution of the form (5.94) exactly in the sense that the residuals vanish with the exact nodal values. On the other hand, the piecewise linear variation on a triangular cell cannot represent the correct solution exactly. But this is not a problem; the problem is that the residual can represent the wrong solution exactly because $q = Br$ implies linear variations in u and v. It is therefore natural that the method would prefer to compute this wrong solution because the norm can be brought further down by producing this solution. One thing is clear: minimizing residuals is not equivalent to minimizing errors. This is one possible explanation for the problem, but it does not give a single clue as to how to overcome this difficulty in least-squares methods, for all it implies is that minimizing the residual will suffer from the same problem whatever the norm is.

5.2.2 An Accurate Least-Squares Scheme

In this section, we propose one way to recover the accuracy of the least-squares method. As mentioned in the previous section, we should solve the Cauchy-Riemann equations, not the Laplace equations. This means that the decoupling that occurs in the discretization must be prevented, and that we must retain the coupling as in the Cauchy-Riemann equations. One way to do this is to minimize the norm of the

form,

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \sum_{T\in\{T\}} \frac{1}{2}\left[{\Delta_T}^2 + {\Omega_T}^2\right], \tag{5.106}$$

i.e. a norm without the area weight $1/S_T$, where

$$\Delta_T = \int_T (\partial_x u + \partial_y v)\, dxdy, \quad \Omega_T = \int_T (\partial_y u - \partial_x v)\, dxdy. \tag{5.107}$$

It is easy to show that this can be written as

$$\mathcal{F} = \sum_{T\in\{T\}} \frac{S_T}{2} \iint_T \nabla u\cdot\nabla u\, dxdy + \sum_{T\in\{T\}} \frac{S_T}{2} \iint_T \nabla v\cdot\nabla v\, dxdy - \sum_{T\in\{T\}} S_T S_T^{'} \tag{5.108}$$

where $S_T^{'}$ is again the area in the hodograph plane (u, v). The last term will not vanish identically this time, and this is the term that links the two variables u and v. Following the steps described in section 5.1.5, we obtain the gradient

$$\frac{\partial\mathcal{F}}{\partial u_j} = \frac{1}{4}\sum_{i\in i_j} \left(S_T \cot\theta_T + S_{T+1}\cot\theta_{T+1}\right)(u_i - u_j) - \frac{1}{2}\sum_{T\in T_j} S_T \Delta v_T \tag{5.109}$$

$$\frac{\partial\mathcal{F}}{\partial v_j} = \frac{1}{4}\sum_{i\in i_j} \left(S_T \cot\theta_T + S_{T+1}\cot\theta_{T+1}\right)(v_i - v_j) + \frac{1}{2}\sum_{T\in T_j} S_T \Delta u_T \tag{5.110}$$

(cf. Figure 5.4, but remember that we are now in $x-y$ plane.) which shows that the last term in each equation does not vanish identically, and the variables are coupled. There are however occasions when they vanish; all the triangles surrounding the node j have the same area except for boundary nodes, such as uniform grids; the solution is uniform along the sides opposite to j; give all Δu_T or Δv_T , and all S_T except one which can be solved for. To show that we are solving the Cauchy-Riemann equations, now we are going to derive the equivalent (or modified) equations of the least-squares scheme. Unifying the summation, we can write the gradient also as

$$\frac{\partial\mathcal{F}}{\partial u_j} = \frac{1}{4}\sum_{i\in i_j} \left[\left(S_T \cot\theta_T + S_{T+1}\cot\theta_{T+1}\right)(u_i - u_j) - 2\left(S_T - S_{T+1}\right)(v_i - v_j)\right] \tag{5.111}$$

$$\frac{\partial\mathcal{F}}{\partial v_j} = \frac{1}{4}\sum_{i\in i_j} \left[\left(S_T \cot\theta_T + S_{T+1}\cot\theta_{T+1}\right)(v_i - v_j) + 2\left(S_T - S_{T+1}\right)(u_i - u_j)\right], \tag{5.112}$$

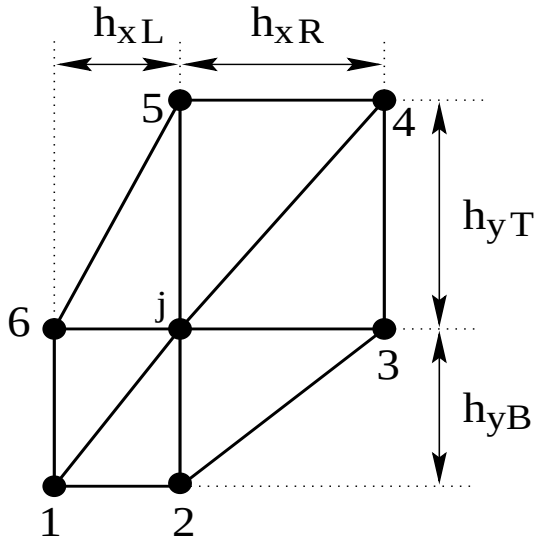

Figure 5.14: A nonuniform regular triangular grid.

Consider a nonuniform stencil shown in Figure 5.14. It is clear that the contribution from nodes 1 and 4 disappear because $\cot 90^\circ = 0$ and two triangles that share each node have the same area. Now, we may assume without loss of generality that the node j is located at the origin. Expanding u as a Taylor series around the node j, we have, up to the second-order,

$$u = u_j + u_x x + u_y y + \frac{1}{2}u_{xx}x^2 + u_{xy}xy + \frac{1}{2}u_{yy}y^2, \tag{5.113}$$

analogously for v,

$$v = v_j + v_x x + v_y y + \frac{1}{2}v_{xx}x + v_{xy}xy + \frac{1}{2}v_{yy}y^2. \tag{5.114}$$

Substituting these into (5.111) and (5.112), after some algebra, we obtain

$$A_1(u_x + v_y) + A_2(v_x - u_y) - B_1(u_{xx} + u_{yy}) - B_2 v_{xx} + B_3 v_{yy} = 0 \tag{5.115}$$

$$-A_2(u_x + v_y) + A_1(v_x - u_y) - B_1(v_{xx} + v_{yy}) + B_2 u_{xx} - B_3 u_{yy} = 0 \tag{5.116}$$

where

$$A_1 = \frac{1}{2}(h_{yT}^2 + h_{yB}^2)(h_{xL} - h_{xR}), \quad A_2 = \frac{1}{2}(h_{xR}^2 + h_{xL}^2)(h_{yT} - h_{yB})$$

$$B_1 = \frac{1}{4}(h_{xR}^2 + h_{xL}^2)(h_{yT}^2 + h_{yB}^2), \quad B_2 = \frac{1}{4}(h_{xL}^3 - h_{xR}^3)(h_{yT} - h_{yB}),$$
$$B_3 = \frac{1}{4}(h_{yT}^3 - h_{yB}^3)(h_{xL} - h_{xR}).$$

u and v being regarded as solutions of the discrete problem, these equations represent equivalent equations, i.e. the equations we are actually solving. Note first that, as expected, these equations reduce to the Laplace equations on a uniform grid, $h_{xR} = h_{xL} = h_{yT} = h_{yB} = h$, for which all constants vanish except for B_1. But suppose $h_{xR} = h_{xL}$ but $h_{yT} \neq h_{yB}$. Then, $A_1 = B_2 = B_3 = 0$, and we obtain

$$\begin{aligned} u_x + v_y &= -\frac{B_1}{A_2}\nabla^2 v = -\frac{1}{2}\left(\frac{h_{yT}^2 + h_{yB}^2}{h_{yT} - h_{yB}}\right)\nabla^2 v \\ v_x - u_y &= \frac{B_1}{A_2}\nabla^2 u = \frac{1}{2}\left(\frac{h_{yT}^2 + h_{yB}^2}{h_{yT} - h_{yB}}\right)\nabla^2 u \end{aligned}$$

which shows that we are solving Cauchy-Riemann equations with some added dissipation. Similarly, if $h_{yT} = h_{yB}$ but $h_{xR} \neq h_{xL}$, we obtain

$$\begin{aligned} u_x + v_y &= \frac{B_1}{A_1}\nabla^2 u = \frac{1}{2}\left(\frac{h_{xR}^2 + h_{xL}^2}{h_{xL} - h_{xR}}\right)\nabla^2 u \\ v_x - u_y &= \frac{B_1}{A_1}\nabla^2 v = \frac{1}{2}\left(\frac{h_{xR}^2 + h_{xL}^2}{h_{xL} - h_{xR}}\right)\nabla^2 v. \end{aligned}$$

In general, if the grid is nonuniform in both directions, we have

$$\begin{aligned} u_x + v_y &= \frac{1}{A_1^2 + A_2^2}\left[B_1(A_1\nabla^2 u - A_2\nabla^2 v) + B_2(A_1 v_{xx} + A_2 u_{xx}) - B_3(A_1 v_{yy} + A_2 u_{yy})\right] \\ v_x - u_y &= \frac{1}{A_1^2 + A_2^2}\left[B_1(A_2\nabla^2 u + A_1\nabla^2 v) + B_2(A_2 v_{xx} - A_1 u_{xx}) - B_3(A_2 v_{yy} - A_1 u_{yy})\right]. \end{aligned}$$

In the case of doubly-connected region, where the circulation becomes important, computational grids would have to be nonuniform. Therefore, this least-squares scheme will be solving the Cauchy-Riemann equations, and we expect that it gives accurate results for the problems with nonzero circulation. The method was applied for the same problem of the flow around the Joukowsky airfoil at an angle of attack

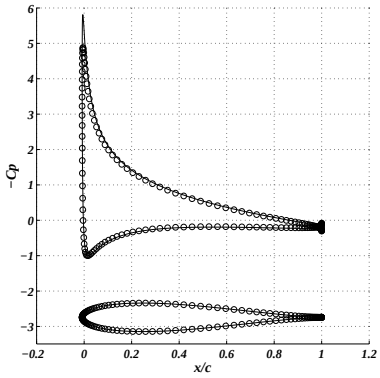

Figure 5.15: C_p distribution around the airfoil and the airfoil geometry. Solid curve represents the exact solution. Circles indicate the numerical solution. 160x80 O-grid.

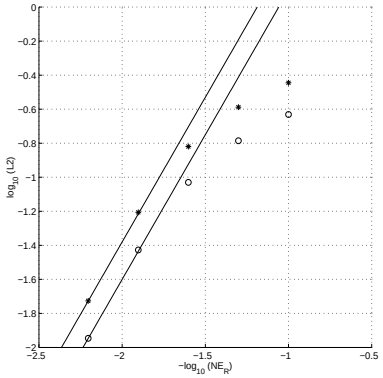

Figure 5.16: The solid line indicates the line of convergence rate 1.7. The error in u and v were measured in L_1 norm, indicated by circles and stars respectively. N_E is the number of nodes in the radial direction of the O-grids.

10° on the same 160x80 O-grid. The solutions converged for 1758 iterations, again making the L_1 norm of the updates below 1.0E-08. It is interesting that the method converges faster for the unweighted norm than for the weighted norm. The pressure distribution around the airfoil is shown in Figure 5.15, which is significantly more accurate than the one in Figure 5.11, A little oscillation is seen however at the trailing edge. This may be expected because the coefficients of the diffusion terms in the equivalent equations are not necessarily positive. The rate of convergence is not precisely two as seen in Figure 5.16 where the errors are computed at all the nodes except on the outer boundary (See Appendix F for computing the exact solution)

and measured in an L_1 norm. The grids used are 10x05, 20x10, 40x20, 80x40, and 160x80 O-grids. But the order of accuracy is nearly 1.7, and the improvement is substantial compared with the previous less-than-first-order accuracy.

A final remark is that this is not the only way to couple the variables, u and v; other types of norms may be suggested. In particular, consider the norm in the form

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2} \sum_{T\in\{T\}} \frac{1}{S_T^m} \left[{\Delta_T}^2 + {\Omega_T}^2 \right]. \tag{5.117}$$

where m is an integer. The weighted norm corresponds to the choice $m = 1$, and the unweighted one corresponds to $m = 0$. The equivalent equations derived from minimizing this norm can be written in the same form as (5.115) and (5.116), but the coefficients now involve the parameter m as follows.

$$A_1 = 2^{m-1}(h_{yT}^{2-m} + h_{yB}^{2-m})(h_{xL}^{1-m} - h_{xR}^{1-m}), \quad A_2 = 2^{m-1}(h_{xR}^{2-m} + h_{xL}^{2-m})(h_{yT}^{1-m} - h_{yB}^{1-m})$$

$$B_1 = 2^{m-2}(h_{xR}^{2-m} + h_{xL}^{2-m})(h_{yT}^{2-m} + h_{yB}^{2-m}), \quad B_2 = 2^{m-2}(h_{xL}^{3-m} - h_{xR}^{3-m})(h_{yT}^{1-m} - h_{yB}^{1-m}),$$

$$B_3 = 2^{m-2}(h_{yT}^{3-m} - h_{yB}^{3-m})(h_{xL}^{1-m} - h_{xR}^{1-m}).$$

This clearly shows that all the coefficients but B_1 will vanish identically for $m = 1$ and the equations revert to the Laplace equations, and moreover any choice other than $m = 1$ will keep some other coefficients finite and therefore retain the variable coupling [5]. Our numerical experiments show however that taking $m > 0$ leads to the same problem in computing lifting flows, making solutions even worse as m gets larger, and that the solutions with $m < 0$ are good in the sense that they have a reasonable amount of lift, but reducing m (i.e. increasing $|m|$) gives spurious oscillation at the trailing edge. Therefore, it would be reasonable to believe that

[5] The same observation can be made by looking at (5.109) and (5.110).

there must be some other explanation to this problem which not only explains the problem, but also offers a cure in a more precise manner.

5.2.3 A Third-Order Least-Squares Scheme

In order to obtain even more accurate solutions to the Cauchy-Riemann equations, we consider a simple third-order extension of the least-squares scheme described in the last section. It is based on the proposition claimed by Caraeni and Fuchs[19]: for residual distribution schemes, if the distribution coefficients are bounded, the accuracy of the scheme is determined solely by the accuracy with which the residual is evaluated over a cell. They developed a third-order residual distribution scheme for the Euler equations using a high-order Gaussian quadrature to evaluate the residual. Here we develop a simple third-order scheme in which we just add a correction term to the second-order residual rather than employing the Gaussian quadrature procedure. One way to compute the residual with higher order accuracy is to use a higher order quadrature formula. Consider

$$\mathbf{\Phi}_{123} = \int\int_{123} [\partial_x f + \partial_y g]\, dxdy = \oint_{123} f\, dy - g\, dx \tag{5.118}$$

where 123 denotes a triangle formed by three vertices $1, 2, 3$, and f and g can be considered as flux vectors but we assume here for simplicity that these are the variables we compute as in the case of Cauchy-Riemann equations (i.e. $f = u$ and $g = v$ for example). Suppose that we have f and g at the midpoint of each edge. Then, the line integral can be evaluated by Simpson's rule.

$$\int_1^2 f\, dy - g\, dx = \frac{(y_2 - y_1)}{6}(f_1 + 4f_m + f_2) - \frac{(x_2 - x_1)}{6}(g_1 + 4g_m + g_2) \tag{5.119}$$

where the subscript m indicates the value at the midpoint. Note that this is 4th order accurate. Introducing the notations $\overline{(\ )} = \{(\ )_2 + (\ )_1\}/2$ and $\Delta(\ ) = (\ )_2 - (\ )_1$, we

rewrite this as

$$\int_1^2 f\,dy - g\,dx = (\overline{f}\Delta y - \overline{g}\Delta x) - \frac{2}{3}\left[(\overline{f} - f_m)\Delta y - (\overline{g} - g_m)\Delta x\right] \tag{5.120}$$

which clearly shows that the second term is a correction term to the standard second-order evaluation(trapezoidal rule). Now we turn to the computation of the midpoint values. First, for simplicity, we do not compute these as additional unknowns, instead we recover these values by way of reconstruction. Yet, in order to preserve the compactness of the scheme, we reconstruct the solution in the form of a Hermite polynomial along each edge. This requires the gradients of the solutions at nodes, but these are easily estimated by area-weighted average over the surrounding triangles using the solutions of the previous iteration. Then the midpoint values can be obtained by Hermite interpolation with the pre-computed gradients, which gives

$$f_m = \overline{f} - \frac{1}{8}(p_2 - p_1), \quad g_m = \overline{g} - \frac{1}{8}(q_2 - q_1) \tag{5.121}$$

where p_i and q_i $(i = 1, 2)$ are the gradients evaluated at node i and then projected onto the edge 12, of f and g respectively. Substituting these into (5.120) and collecting the contributions from all sides, we arrive at the final expression.

$$\mathbf{\Phi}_{123} = \sum_{edges} (\overline{f}\Delta y - \overline{g}\Delta x) - \frac{1}{12}\sum_{edges} (\Delta p\Delta y - \Delta q\Delta x)\,. \tag{5.122}$$

The same expression can be obtained by constructing the Hermite polynomial along each edge and performing the line integral exactly. Note that the first term is exactly the second-order residual. Hence, to achieve high-order accuracy, we just need to add a correction term that is the second term on the right. For systems of equations (f and g are now flux vectors $\mathbf{f}$ and $\mathbf{g}$), we linearize the equations first if nonlinear and then follow the same procedure choosing a variable for reconstruction, say $\mathbf{u}$. Thus

$$\mathbf{\Phi}_{123} = \sum_{edges} (\mathbf{A}\overline{\mathbf{u}}\Delta y - \mathbf{B}\overline{\mathbf{u}}\Delta x) - \frac{1}{12}\sum_{edges} (\mathbf{A}\Delta\mathbf{r}\Delta y - \mathbf{B}\Delta\mathbf{r}\Delta x) \tag{5.123}$$

where $\mathbf{A} = \partial\mathbf{f}/\partial\mathbf{u}$ and $\mathbf{B} = \partial\mathbf{g}/\partial\mathbf{u}$ are Jacobian matrices whose elements are constant or made constant within the triangle by linearization, and $\mathbf{r}$ is the nodal gradient of $\mathbf{u}$ projected on the edge. Again, the first term is the second-order residual and the second term is the correction term that improves the accuracy. The method will however not be fourth-order accurate because of the error committed in the gradient recovery. In fact, it will be third-order accurate only if the estimates of the gradients are second-order accurate. Now back to the scalar case, suppose that the gradient is estimated at node j with m-th order accuracy.

$$(\nabla f)_j = \nabla f + \mathcal{O}(h^m) \tag{5.124}$$

where h is a length scale of the grid size. Then, the projected slope, say p_1 on edge 12, is written

$$p_1 = (\nabla f)_j \cdot \vec{s_{12}} = (f_x)_j(x_2 - x_1) + (f_y)_j(y_2 - y_1) = \nabla f \cdot \vec{s_{12}} + \mathcal{O}(h^{m+1}). \tag{5.125}$$

Therefore, the line integral along the edge 12 is, in fact,

$$\int_1^2 f\,dy - g\,dx = (\overline{f}\Delta y - \overline{g}\Delta x) - \frac{1}{12}\left(\Delta p \Delta y - \Delta q \Delta x\right) + \mathcal{O}(h^{m+2}) \tag{5.126}$$

which shows that the method will be third-order if $m = 2$. This can be realized on a fairly regular grid. Consider a group $\{T_j\}$ of triangles that share node j. Given function values f at the nodes, we want to estimate the gradient of f at node j. The simplest way to do this would be compute the area-weighted average, assuming that f is a piecewise linear function within each triangle denoted by f^T.

$$(f_x)_j = \frac{\sum_{\{T_j\}} S_T f_x^T}{\sum_{\{T_j\}} S_T}, \quad (f_y)_j = \frac{\sum_{\{T_j\}} S_T f_y^T}{\sum_{\{T_j\}} S_T} \tag{5.127}$$

where S_T is the area of the triangle $T \in \{T_j\}$ and

$$f_x^T = \frac{1}{2S_T}\sum_{\{i_T\}} f_i \Delta y_i, \quad f_y^T = -\frac{1}{2S_T}\sum_{\{i_T\}} f_i \Delta x_i \tag{5.128}$$

where $\{i_T\}$ is a list of three vertices of triangle T. We are now interested to see how accurate these formulas can be.

Without loss of generality, we assume that the node j is at the origin, and assuming f is smooth, we write

$$f = f_j + \sum_{n=1}^{\infty} \frac{1}{n!} \left[x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} \right]^n f, \tag{5.129}$$

which is a Taylor series expansion around the node j. Then, for a triangle $j12$, after some simple algebra, we find

$$\sum_{\{i_T\}} f_i \Delta y_i = 2S_T f_x + y_2 \sum_{n=2}^{\infty} \frac{1}{n!} \left[x_1\frac{\partial}{\partial x} + y_1\frac{\partial}{\partial y} \right]^n f - y_1 \sum_{n=2}^{\infty} \frac{1}{n!} \left[x_2\frac{\partial}{\partial x} + y_2\frac{\partial}{\partial y} \right]^n f, \tag{5.130}$$

This can be written,

$$\sum_{\{i_T\}} f_i \Delta y_i = 2S_T f_x + \frac{1}{2} \left(E_{xx}^T f_{xx} + E_{xy}^T f_{xy} + E_{yy}^T f_{yy} \right) + \mathcal{O}(h^4) \tag{5.131}$$

where

$$E_{xx}^T = (y_2 x_1^2 - y_1 x_2^2),\ E_{xy}^T = 2x_1 y_2 (y_1 - y_3),\ E_{yy}^T = (y_1^2 y_2 - y_1 y_2^2) \tag{5.132}$$

and higher order terms have been neglected. Collecting the contribution from other triangles, we have

$$(f_x)_j = f_x + \frac{1}{4} \frac{\sum_{\{T_j\}} \left(E_{xx}^T f_{xx} + E_{xy}^T f_{xy} + E_{yy}^T f_{yy} \right)}{\sum_{\{T_j\}} S_T} + \mathcal{O}(h^2). \tag{5.133}$$

The error constants are identical to those derived and studied by Roe[83] in great details for cell-vertex schemes, who discovered various error-free polygons. Here, it is obvious, at least, that the constants will vanish for a pair of triangles that are symmetric with respect to node j. Therefore, the estimate will be second-order accurate on a group of triangles that can be generated by these pairs such as the one shown in Figure 5.14 but with uniform spacings.

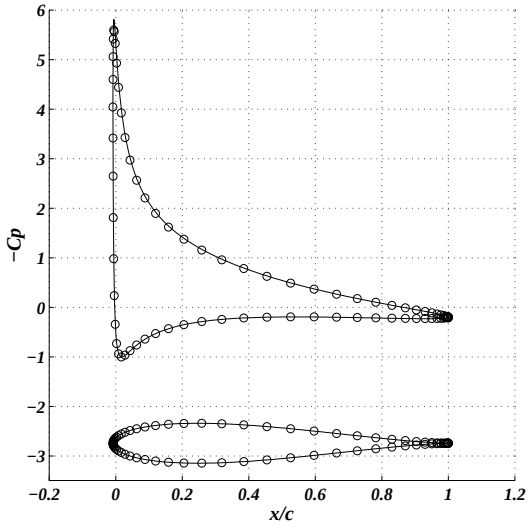

Figure 5.17: C_p distribution on a Joukowsky airfoil at the angle of attack 10°, and the geometry of the airfoil. Circles are numerical solutions on a 80x40 O-grid while the solid curve represents the exact solution.

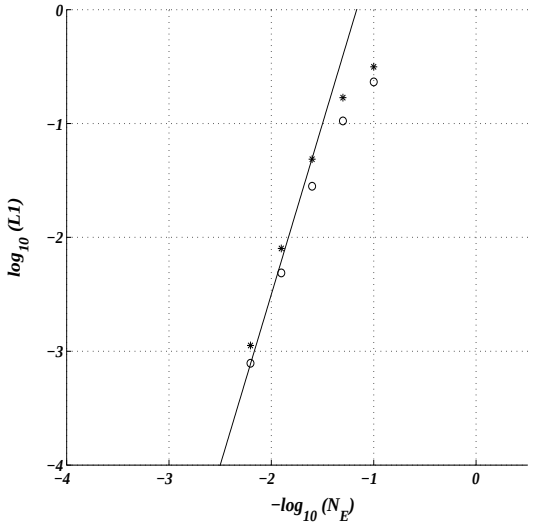

Figure 5.18: The solid line indicates the line of convergence rate 3. The error in u and v were measured in L_1 norm, indicated by circles and stars respectively. N_E is the number of nodes in the radial direction of the O-grids.

The evaluation of the residual is therefore third-order accurate on a fairly regular grid such as the one shown in Figure 5.10. And it can be distributed in the same way as the second-order scheme without deteriorating the accuracy, based on the proposition mentioned earlier. This means that the second-order least-squares scheme is upgraded into a third-order scheme simply by adding the correction term to the residual. The second term in (5.122) is simply a numerical correction to the second-order least-squares scheme, and that this is the only change made to the second-order scheme. The results for the Joukowsky airfoil at an angle of attack 10° confirm the predicted third-order accuracy as shown in Figure 5.18. The grids used are again 10x05, 20x10, 40x20, 80x40, and 160x80 O-grids. A sample solution is shown in Figure 5.17. Note that third-order solution is very accurate compared

with the second-order solutions such as shown in Figure 5.11 and 5.15 which were obtained on twice as fine a grid as the one used here.

5.2.4 The Least-Squares Moving Grid Method

We now improve the solution further by introducing the mesh movement into the third-order least-squares scheme. As pointed out earlier, the unweighted norm that is necessary to compute accurate solutions does not give any mechanism with which mesh folding is countered. But this does not mean that it is impossible to move the grid by minimizing the norm. It is possible, and in fact expected to improve the solutions because now we know that minimizing the unweighted norm gives accurate solutions. The only problem is that mesh tangling can occur with the unweighted norm. Here, we apply the least-squares moving grid method straightforwardly, but it is checked, every time a node is moved, whether or not the node stays inside of the group of triangles that share that node. In other words, a check is made whether the nodal movement creates triangles with negative area, and if it does, that movement will be rejected. This way, in principle, mesh validity can be preserved during the mesh movement.

We consider again a flow around a Joukowsky airfoil at an angle of attack 10°. The same techniques, the diagonal scaling and the Gauss-Seidel iteration, were employed for both solutions and nodal coordinates. The nodal movement is allowed only when it yields positive area for all the neighboring triangles. We now go down to even coarser grid 40x20 O-grid (Figure 5.19), the half size in each direction of the previous 80x40 O-grid. Figure 5.20 shows the C_p distribution obtained by the third-order least-squares scheme without grid movement on this coarse grid (after 233 iterations with the relaxation factor 1.8). It is not accurate enough: it fails to capture the suction

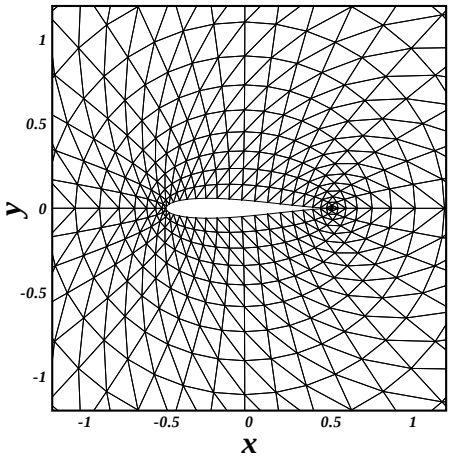

Figure 5.19: Initial 40x20 O-grid around a Joukowsky airfoil.

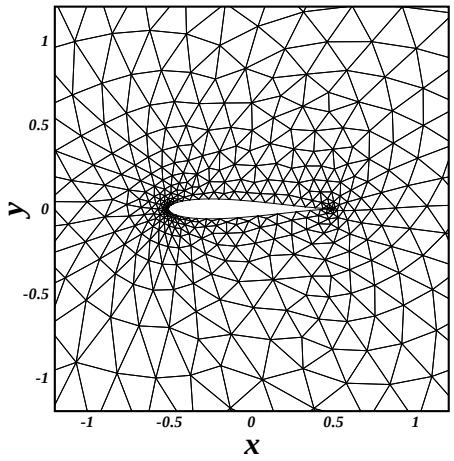

Figure 5.20: Adaptive grid.

peak, thus resulting a relatively small circulation around the airfoil, 4.17198E-01, whereas the exact value is 5.99931E-01. The least-squares moving grid method is then applied for 10000 iterations with the relaxation factors 1.8 for the solutions and 0.1 for the grid, followed by the final iteration for solutions with the grid fixed. The boundary nodes are moved in the same way as was done in Section 5.1.6 (the Laplacian smoothing technique) to follow the internal node movement, and each node is projected onto the airfoil immediately after the movement, and they are moved only if the mesh validity is preserved. Diagonal swapping technique is also used to alleviate the topological limitation of the grid movement (applied at every 100 iterations). The correction term in the third-order accurate residual (5.122) is again regarded as a constant, and therefore the distribution of the residual for nodal movement is performed just as in the second-order method. The grid and C_p obtained by the least-squares moving grid method are shown in Figures 5.20 and 5.22. As can be seen, the grid points have been concentrated at the leading and the trailing edge,

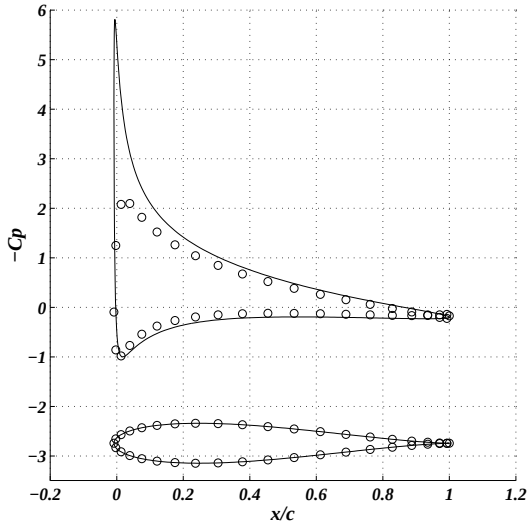

Figure 5.21: C_p distribution on the initial grid.

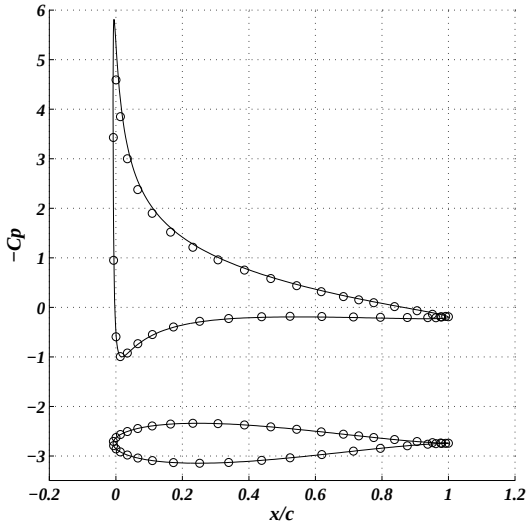

Figure 5.22: C_p distribution on the adaptive grid.

and the solution accuracy has been improved in these regions. Especially, the method has accurately captured the suction peak, and produced the circulation that is very accurate, 5.78325E-01. Note also that the resulting grid is very smooth, which would not have been attained without diagonal swapping techniques. Finally, the norms were found to be 4.394E-4 for the fixed grid case and 2.386E-4 for the moving grid case, which shows clearly that the smaller norm now implies smaller errors.

5.2.5 Moving Quadrilateral Grids

For quadrilateral grids, we know from the discussion in Chapter II that there are not many degrees of freedom left to move the grid. But it is also true that there exist some. Consider computing a flow around an airfoil by solving the Cauchy-Riemann equations (for u and v) on a fixed regular O-grid. There are $2N_V$ solutions at N_V nodes, and $2N_C$ equations in N_C quadrilateral elements. Then, using (2.17), the excess of the unknowns is found to be

$$2N_V - 2N_C = N_{Vb}. \tag{5.134}$$

Assume that the exact solution is imposed at the outer boundary, and apply the tangency condition at the inner boundary. Then, the former eliminates N_{Vb} unknowns, and the latter $N_{Vb}/2$ unknowns[6]. Therefore, we actually have $N_{Vb}/2$ more equations than the unknowns: the problem is overdetermined by $N_{Vb}/2$ equations. Although a very small number, we can make use of these extra degrees of freedom to move the grid. We do not expect, however, that the method would adapt the grid as in the triangular grid case (Figure 5.20) because the residuals would be all driven to zero with a little movement of the grid points. Nevertheless, we may hope that something interesting would happen since the residuals can all be made to vanish with the grid movement. And we will see here that the something does happen.

It is straightforward to apply the least-squares method on quadrilateral grids. To compute a residual within a quadrilateral, we first break the quadrilateral into two triangles separated by a diagonal (Figure 5.23). And we compute the residual within each triangle as before, and add them to obtain the residual for the quadrilateral cell.

$$\Delta_Q = \int_Q (\partial_x u + \partial_y v)\, dxdy = \oint_{\partial Q} (udy - vdx) = \Delta_A + \Delta_B \tag{5.135}$$

$$\Omega_Q = \int_Q (\partial_y u - \partial_x v)\, dxdy = -\oint_{\partial Q} (udx + vdy) = \Omega_A + \Omega_B \tag{5.136}$$

where A and B denote the two subtriangles. The line integral along the diagonal is counted twice with opposite direction, thus canceling out each other, and therefore the choice of the diagonal does not matter. This procedure makes it very similar to the triangular case to formulate the problem and write a code. In fact, it can be shown that this is equivalent to the exact integration with an assumed bilinear variation of the solution, to discretization by using the area-weighted averages of

[6]Note that the outer and inner boundaries have the same number of nodes for the regular O-grid.

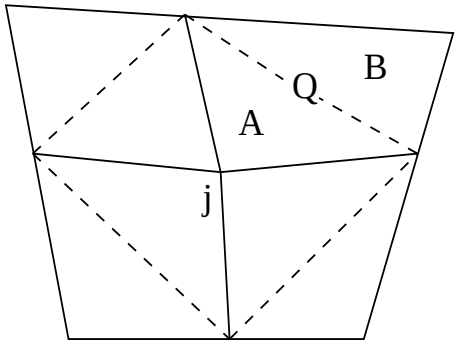

Figure 5.23: A set of quadrilaterals $\{Q_j\}$ that share a node j. Each quadrilateral is divided into a pair of triangles A and B by dashed lines for the purpose of computing the residual and the gradient.

the solution derivatives over the two triangles, and also to the use of the one-point Gaussian quadrature on a quadrilateral.

For quadrilateral grids, there is no problem in computing a lifting airfoil as mentioned in Section 5.2.1, and therefore we minimize the weighted norm

$$\mathcal{F} = \sum_{Q \in \{Q\}} F_Q = \sum_{Q \in \{Q\}} \frac{1}{2} \frac{{\Delta_Q}^2 + {\Omega_Q}^2}{S_Q} \tag{5.137}$$

where the sum is over the set $\{Q\}$ of quadrilaterals that covers the domain, S_Q denotes the area of the quadrilateral cell Q. The gradients with respect to the solution $\mathbf{u} = (u, v)$ and the nodal coordinates $\mathbf{x} = (x, y)$ are obtained as a sum of the contribution from all the surrounding quadrilaterals.

$$\frac{\partial \mathcal{F}}{\partial \mathbf{u}_j} = \sum_{Q \in \{Q_j\}} \frac{\partial F_Q}{\partial \mathbf{u}_j} \tag{5.138}$$

$$\frac{\partial \mathcal{F}}{\partial \mathbf{x}_j} = \sum_{Q \in \{Q_j\}} \frac{\partial F_Q}{\partial \mathbf{x}_j} \tag{5.139}$$

where

$$\frac{\partial F_Q}{\partial \mathbf{u}_j} = \frac{\Delta_Q}{2S_Q} \begin{bmatrix} \Delta y_A \\ -\Delta x_A \end{bmatrix} + \frac{\Omega_Q}{2S_Q} \begin{bmatrix} \Delta x_A \\ \Delta y_A \end{bmatrix} \tag{5.140}$$

$$\frac{\partial F_Q}{\partial \mathbf{x}_j} = \frac{\Delta_Q}{2S_Q} \begin{bmatrix} \Delta v_A \\ -\Delta u_A \end{bmatrix} - \frac{\Omega_Q}{2S_Q} \begin{bmatrix} \Delta u_A \\ \Delta v_A \end{bmatrix} - \frac{F_Q}{2S_Q} \mathbf{n}_A \tag{5.141}$$

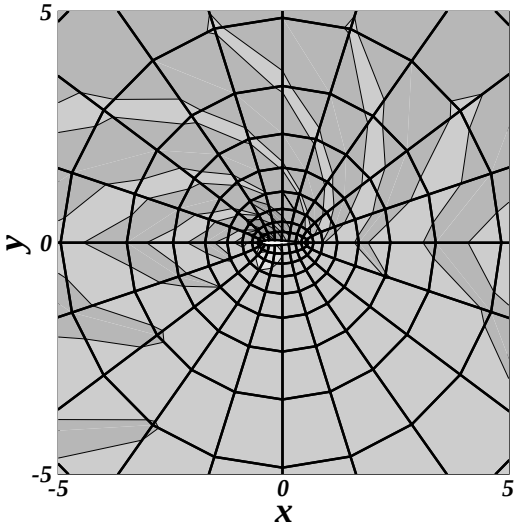

Figure 5.24: C_p contours on a regular O-grid.

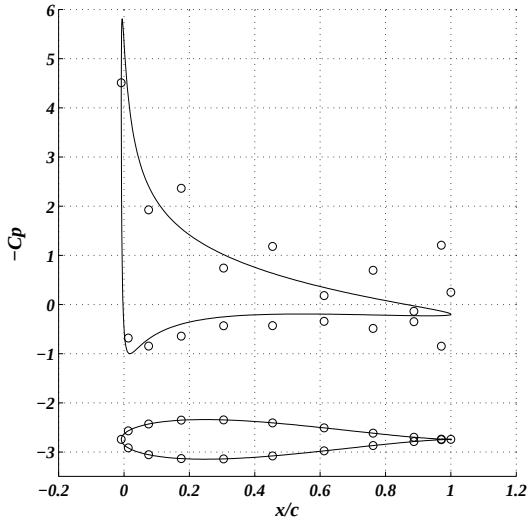

Figure 5.25: C_p distribution on the regular grid.

where the quadrilateral is divided into two triangles (Figure 5.23): one which has the node j as its vertex and the other which does not, and the subscript A denotes the former, so that the notations are the same as in the triangular case. With these gradients above, the steepest descent method can be used to find a minimum. The same techniques such as sequential updates or diagonal scaling can also be used to accelerate the convergence.

We consider again the Joukowsky airfoil at an angle of attack 10°. First, the least-squares solutions were computed on a coarse grid (20x10 O-grid with the outer boundary at $r = 10$) without grid movement. At the trailing edge, no special condition is imposed, and the solutions are computed there just like at interior nodes. The method converged for 133 Gauss-Seidel iterations, and the final value of the norm was 3.09525E-02. Contours of pressure coefficients are shown in Figure 5.24, with the final grid overlaid. This shows clearly that a spurious checkerboard mode has appeared and contaminated the solution. C_p distribution around the airfoil is disas-

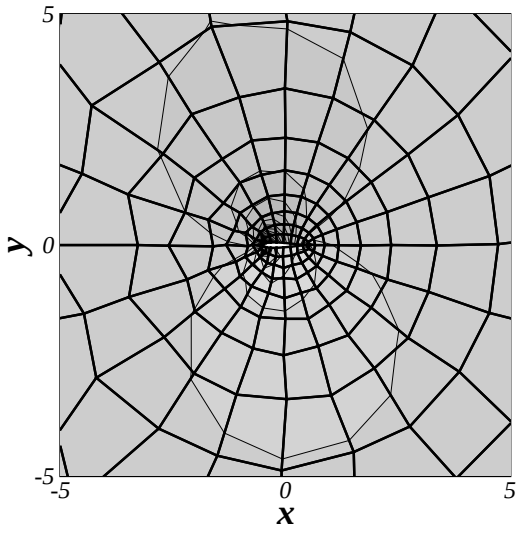

Figure 5.26: C_p contours on an adaptive grid.

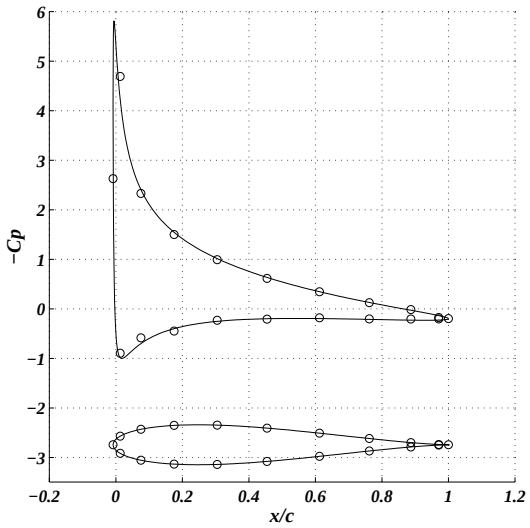

Figure 5.27: C_p distribution on the adaptive grid.

trous as shown in Figure 5.25. In fact, it is well-known that cell-vertex schemes on quadrilateral grids suffer from this particular error mode [67]. One way to suppress this mode is through the boundary conditions. Imposing $v = 0$ at the trailing edge, i.e. Kutta condition, we actually obtained a little better solution, but only around the trailing edge and the checkerboard mode in the pressure coefficient remained. A remedy suggested by Morton[67] is to add a fourth-order artificial dissipation, an essential item for central schemes. Here, we demonstrate that it is possible to eliminate this problem by the grid movement. Although we have a very small number of the degrees of freedom, we allow all the interior nodes to move, thus creating a highly underdetermined problem. This implies that the residuals can be driven to zero for all the elements at the cost of the uniqueness of the solution. With the relaxation factors 1.0 and 0.5 for solutions and nodal coordinates respectively, the method converged quickly for 255 iterations, reducing the norm by several orders of magnitude down to 3.19750E-07. The results are shown in Figures 5.26 and 5.27. It is remark-

able that with a little grid movement the solution has been improved significantly, which is very accurate for this size of a coarse grid. Yet the most striking result is that the perturbation on the grid produced by the moving grid algorithm is now checkerboard as if the error mode has been transfered from the solution to the grid. The implication is that a difficult problem may be overcome easily by introducing additional degrees of freedom; the grid movement is one possible approach. Finally, the result shows that grid qualities such as orthogonality or smoothness, which have been widely believed to be important for solution accuracy in general, are not always important: the general-purpose criteria for grid quality do not serve, and a *bad* grid can actually be a good grid.

CHAPTER VI

CONCLUSIONS

6.1 One Dimension

A least-squares moving grid method has been developed for one-dimensional boundary-value problems, and shown to be capable of producing highly accurate solutions by adjusting the node distribution. In one-dimension, because of the lack of degrees of freedom, it is necessary to introduce additional equations in order to guide the grid movement. For this purpose, an error estimate has been derived by computing the difference between the piecewise linear approximation and the Hermite polynomial. The equidistribution of this quantity over the elements has proved to be a very effective strategy and be easily incorporated with the residual minimization. In addition, this study has introduced an interesting interpretation of the numerical approximation to the solution of differential equations: approximation of a geometrical object. There is no longer the distinction between independent and dependent variables, i.e. grids and solutions. And this is the only means by which we are able to grasp the mechanism of the moving grid method. It tries to place the nodes onto the solution curve as quickly as possible. This work has just opened a new paradigm: geometric approach. In higher dimensions, however, it is somewhat unclear how to exploit it in a practical sense although it can be extended naturally

to higher dimensions (See Appendix D).

In fact, taking this geometrical point of view, another new approach has been devised. Considering a problem of approximating a one-dimensional curves in $\mathbf{R}^n$ ($n > 0$), parameterized by t, we have devised an algorithm to generate a node distribution that achieves equidistribution of the error estimate. And there is more. It is capable of achieving a desired L_2 error with remarkable accuracy. The generality of the problem allows the technique to be applicable to a wide range of problems: interpolation, numerical integration, initial-value problems for ordinary differential equations, and boundary-value problems for ordinary differential equations. In this approach, in stead of adjusting the grid points, we generate nodes from one end to the other such that the error estimate is equidistributed. The algorithm has been successfully applied to all the problems mentioned above. See Appendix A for further details.

The work on one-dimensional problem might become important when we reach a stage of tackling the Navier-Stokes equations. Here, there is great practical interest on the accurate prediction of the minimum number of nodes to well resolve the boundary layer.

6.2 Two Dimensions

For two-dimensional problems, the least-squares moving grid method is definitely an promising approach. Above all, it provides the capability of efficiently capturing discontinuous solutions with many fewer nodes than the conventional methods. For linear hyperbolic problems, we have shown that the problem becomes a simple minimization of a triangular area in characteristic planes, by introducing the extra degrees of freedom. For 2x2 systems, we apply the method for each characteristic

equation, thereby minimizing the area in each characteristic plane. For systems more than 2x2, we expect that the method would respond to one or two waves that locally dominates the flow because, as we have seen, grids and solutions respond only to nonzero residuals. But if more than two waves happened to be important, say three, a possibility would be to introduce a quadrilateral element, thus increasing an additional edge to capture the third wave. It was also demonstrated that the method is capable of computing an exact discontinuity by the use of degenerate elements. Presumably, no one actually ever considered seriously introducing cells with zero area. This is natural because many numerical schemes would break down on such a grid due to the presence of the factor $1/S_T$ somewhere in their update formulas. Fortunately, in the least-squares scheme, such a singularity can be easily removed simply by minimizing the unweighted norm. Interpreted geometrically, the method approximates a two-dimensional submanifold by triangulation treating all the variables as independent. The manifold is a geometrical object, and therefore discontinuous solution is not really discontinuous but a sharp corner which can be represented accurately by linear triangular elements. In fact, the residual can be interpreted as representing a geometrical error. Further details can be found in Appendix D.

Accurate and efficient nonlinear shock capturing was demonstrated. An adaptive evaluation of the residual was shown to be effective to capture a shock as well as avoid non-physical shocks. Also, the use of the rate of change of the element area was proposed to tell if an element was in compression or expansion. This quantity might find its application in other areas as a simple shock-detector. A more sophisticated mechanism for removing nodes should, however, be devised. Note again that the focus is on removing redundant nodes rather than inserting nodes.

Accurate elliptic solvers have been developed that incorporate with an automatic grid adaptation. In particular, a third-order scheme has been developed with a little additional cost. In addition, the scheme has a good smoothing property, and therefore can be combined with a multigrid technique to further accelerate the convergence[12], which is left as a future work. Also, a hitherto unknown problem of computing lifting airfoils was discovered, studied, and resolved. It is however worth further investigation to fully understand the problem. Another way to recover the accuracy may be found. Finally, the method was applied to quadrilateral grids for which there are only a few extra degrees of freedom available. But it was shown that the grid movement eliminated completely the checkerboard mode in the solution which then appeared in the grid instead. This is an interesting result in the sense that a difficult problem can be easily resolved by introducing additional degrees of freedom, and that the general-purpose criteria for grid quality are not necessarily useful: a good grid depends on the problem and may not be determined without reference to the nature of the problem.

In this study, the relaxation factor for grid movement was determined experimentally. For robustness, it is desired to establish a precise way to choose it, possibly for some stability condition on the grid movement. The difficulty comes from the fact that the equations for grid coordinates are typically nonlinear even for linear problems, so that useful linear stability analyses do not apply. Nevertheless, taking a viewpoint that we are solving equations on a grid laid in the solution space, some nonlinear analysis may be developed. The future work should include this important topic.

The method can be applied to any two-dimensional partial differential equations by decomposing into hyperbolic and elliptic subproblems. The Euler equations,

which have both hyperbolic and elliptic parts, are naturally the next target for the least-squares method. A least-squares norm for the Euler equations that possesses a certain number of attractive properties have actually been discovered (See Appendix E), but no numerical results are available at the time of writing this thesis.

6.3 Three Dimensions and Beyond

A challenge is to develop a moving grid method for three-dimensional problems where the benefit of moving grid would be enormous, especially for discontinuous solutions. In three dimensions, as discussed in Chapter II, tetrahedral grids can be used, which provides us with sufficient amount of extra degrees of freedom and facility for grid alteration, face swapping and inserting/deleting nodes. In fact, it is easy to show that for a simple linear advection problem the residual vanishes if the solution is constant along one edge of a tetrahedron and that edge is aligned with the characteristic direction. Further study on fundamental equations is however necessary. For nonlinear shocks, a quadrature formula must be found that satisfies a jump condition. For elliptic problems, its accuracy on a fixed grid must be confirmed first before we move the grid.

Four dimensional problems, i.e. time-dependent problems, are just another challenge. In this case, for instance, the least-squares method is expected to create a characteristic grid and compute the solution simultaneously on it in a time-accurate manner. Again, the key feature of the method is its local property: modify the grid locally responding to nonzero residuals alone. As mentioned earlier, it is very important to develop accurate least-squares schemes before we move the grid. It is therefore desired to first develop a time-accurate least-squares scheme. The development of residual distribution schemes for unsteady problems has begun recently[61, 37, 45].

A least-squares scheme as one of the distribution schemes should be worth particular attention in the development, for its success as a numerical scheme means that as a grid adaptation method as well.

The study presented in this thesis has shown that quite impressive results can be obtained by computing grids and solutions simultaneously through residual minimization, especially for shock-capturing, giving also the explanations as to why and how such results were generated, for the mechanism of the grid movement in particular. This thesis has just opened the door to the development of a new class of grid adaptation techniques by showing that there do exist better grids and solutions than those we are currently content with and the residual minimization does lead us to these remarkable results.

APPENDICES

APPENDIX A

An Alternative Method for One-Dimensional Problems

A.1 Introduction

Here, we extend the geometrical viewpoint taken in Chapter III to other classes of one-dimensional problems. We begin by pointing out that in many branches of numerical analysis, piecewise linear continuous functions are often employed to approximate one-dimensional submanifolds, whether or not one intends to do so, such as discretization of boundaries for computational grids in two dimension, interpolation of a function in one dimension, numerical solutions of ordinary differential equations, and so on. In a terminology of modern mathematics, these problems may be categorized as a problem of numerically approximating one-dimensional submanifolds(curves), typically by piecewise linear elements. In any problems of this kind, it is well known that adjusting the node distribution is a very effective way to improve the approximation. A popular strategy is to place nodes so as to equally distribute the arc length of the manifold. In White[98], such an approach is studied in detail to improve the numerical solutions of two-point boundary value problems. Another area of the same kind is L_2 fits to functions with adjustable nodes. In this area, since the function is known, algorithms have been developed based on a direct

minimization of the L_2 error such as the one in Baines[5]. Numerical integration is yet another area in which the trapezoidal rule can be considered as approximating a function by a piecewise linear continuous function. On the other hand, particularly in the area of numerical initial value problems for ordinary differential equations, a different approach is usually taken that focuses on the error at grid points. Adaptive step-size control techniques have been developed based on an error estimate given by the difference of numerical solutions obtained by two schemes with two consecutive orders of accuracy such as embedded Runge-Kutta methods[39, 54, 32].

Here, we approach these problems from a unified viewpoint that all these problems can be thought of as a numerical approximation to one-dimensional submanifolds by piecewise linear continuous elements. Choosing the L_2 error, the integral value of the pointwise error squared, as a measure of the error of the approximation, we consider the problem of constructing geometry(or solution)-adaptive node distribution in the parameter space of the submanifold to reduce the error. A key idea to the determination of the node distribution derives from the fact that the L_2 error does not vanish even with the exact nodal values, i.e. the interpolation error. This simple fact implies that there exist error components in the L_2 that are essentially independent of the nodal approximation. Hence no numerical schemes, which play a role of computing the nodal values, can control these errors, thus further implying that they can be controlled only by changing the node distribution in the parameter space. In other words, the development of algorithms for adaptive node distribution should be guided by these error components. As a consequence, such algorithms, unlike those which compute the nodal values, should be equally applicable to all the areas of application whose objective is to construct an accurate piecewise linear continuous approximation of a one-dimensional submanifold.

The error components that are incurable by numerical schemes can be derived by directly computing the L_2 error for a smooth submanifold as demonstrated in Section A.2. It is shown that the leading term of these error components estimates the L_2 error with second-order accuracy provided the nodal approximation is more than third order accurate. Therefore it plays an important role not only in guiding the node distribution but also in estimating the error. Exploiting this fact, an algorithm that constructs a piecewise linear continuous approximation for is devised in Section A.3 based on an equidistribution strategy, that ensures a specified global L_2 error or equivalently a specified local average error. The algorithm generates nodes successively, advancing from one end of the manifold to the other rather than adjusting an existing nodes, which makes the computation very quick. A difficulty arises by a singular behavior of the generating equation which is solved iteratively: a curvature term in the denominator causes a trouble when it is very small. A modification of the equation is proposed that damps a large number created by extremely small curvature.

Applications of the algorithm are described in Section A.4. The first example is the approximation of a curve in $\mathbf{R}^n$. Results demonstrate that the method be capable of highly accurate approximations to known functions in one dimension that ensures a specified L_2 error. The second is numerical integration in which case the upper bound of the integration error can be specified. The third is initial value problems for ordinary differential equations in which the algorithm is combined with a fourth-order numerical scheme that computes nodal solutions. Finally, the method is applied to boundary value problems for ordinary differential equations via the shooting technique, which requires a slight modification of the algorithm for initial value problems.

A.2 Error estimates

In this section, we derive an L_2 error estimate for a piecewise linear continuous approximation to a smooth one-dimensional submanifold. For this purpose, we begin by studying the simple case that the manifold is merely a function in one dimension. The error will be calculated directly assuming that the manifold we wish to approximate is smooth. From this result, the leading terms of the error components independent of nodal errors will be identified. The analysis will then be extended to one-dimensional submanifolds in higher dimensions to derive a local and global L_2 error estimates. Some numerical results are shown that verify the accuracy of the error estimates.

A.2.1 Local error analysis

Suppose we have a piecewise linear continuous approximation of unknown accuracy, $u(t)$, to a function $x(t)$ in the interval $I = [0, 1]$ which is subdivided into a set of elements $\{E\}$ (See Figure A.1). Within each element $E = [t_l, t_r] \in \{E\}$, we have

$$u_E(t) = \frac{\Delta u_E}{\Delta t_E} t + \frac{u_l t_r - u_r t_l}{\Delta t_E} \tag{A.1}$$

where $\Delta u_E = u_r - u_l$, $\Delta t_E = t_r - t_l$, and u_l and u_r are some approximations to $x(t_l)$ and $x(t_r)$ respectively. Assume that $x(t)$ is smooth within the element E and therefore expressible as a Taylor series around the midpoint of E,

$$x(t) = x_m + \sum_{n=1}^{\infty} \frac{x_m^{(n)}}{n!} (t - t_m)^n \tag{A.2}$$

where $x_m = x(t_m = (t_l + t_r)/2)$, and $x_m^{(n)}$ is the nth derivative of $x(t)$ evaluated at t_m. Assuming Δt_E is small, we are going to compute the square of the local L_2 error,

$$\|x - u_E\|_{L_2(E)}^2 = \int_E (x(t) - u_E(t))^2 \, dt \tag{A.3}$$

using (A.1) and (A.2) up to some significant order. Hence we begin with

$$\|x - u_E\|^2_{L_2(E)} = \int_E \left(x_m - u_E(t) + \sum_{n=1}^{\infty} \frac{x_m^{(n)}}{n!}(t - t_m)^n \right)^2 dt. \tag{A.4}$$

Using the following relation, which is derived by approximating x_m by $\frac{x(t_l)+x(t_r)}{2}$,

$$\begin{aligned} x_m - u_E(t) &= \frac{x(t_l) + x(t_r)}{2} + \text{“truncation error”} - u_E(t) && \text{(A.5)} \\ &= \overline{e_E} - \frac{\Delta u_E}{\Delta t_E}(t - t_m) - \sum_{n=2,4,6} \frac{x_m^{(n)}}{2^n \cdot n!} {\Delta t_E}^n + \mathcal{O}\left({\Delta t_E}^8\right) && \text{(A.6)} \end{aligned}$$

where $\overline{e_E} = \frac{1}{2}(e_l + e_r) = \frac{1}{2}\{(x(t_l) - u_l) + (x(t_r) - u_r)\}$, and neglecting high order terms, we obtain

$$\begin{aligned} \|x - u_E\|^2_{L_2(E)} &= \int_E \left(\overline{e_E} - \Psi_E(t - t_m) - \sum_{n=2,4,6} \frac{x_m^{(n)}}{2^n \cdot n!} {\Delta t_E}^n + \sum_{n=2}^{6} \frac{x_m^{(n)}}{n!}(t - t_m)^n \right)^2 dt \\ &\quad + \mathcal{O}\left({\Delta t_E}^9\right) \end{aligned} \tag{A.7}$$

where $\Psi_E = \frac{\Delta u_E}{\Delta t_E} - x_m^{(1)}$. Note that the integral of $(t - t_m)^n$ vanishes when n is odd, and results in $\frac{{\Delta t_E}^{(n+1)}}{2^n(n+1)}$ when n is even. Consequently, the resulting expression will contain only odd order terms. The result is

$$\begin{aligned} \|x - u_E\|^2_{L_2(E)} &= \overline{e_E}^2 \Delta t_E - \frac{1}{12}\left[2\,\overline{e_E} x_m^{(2)} - {\Psi_E}^2\right] {\Delta t_E}^3 \\ &\quad - \frac{1}{240}\left[\overline{e_E} x_m^{(4)} + \Psi_E x_m^{(3)} - 2\left\{x_m^{(2)}\right\}^2\right] {\Delta t_E}^5 \\ &\quad - \frac{1}{26880}\left[\overline{e_E} x_m^{(6)} + \Psi_E x_m^{(5)} - \frac{20}{3} x_m^{(2)} x_m^{(4)} - \frac{5}{3}\left\{x_m^{(3)}\right\}^2\right] {\Delta t_E}^7 \\ &\quad + \mathcal{O}\left({\Delta t_E}^9\right). \end{aligned} \tag{A.8}$$

Although this looks a little complicated, it contains valuable pieces of information about the error.

A.2.2 Leading terms

The simplest case to begin with would be a piecewise linear continuous approximation of a function $x(t)$ with the exact nodal values. In this case (A.8) simplifies

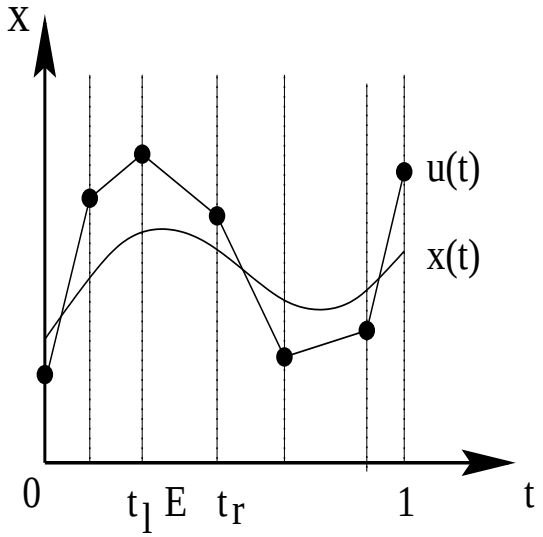

Figure A.1: A piecewise linear approximation to a function in one dimension.

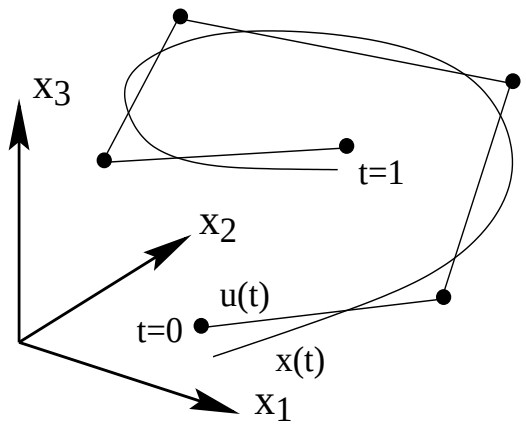

Figure A.2: A piecewise linear approximation to a curve in three dimension.

to

$$\|x - u_E\|^2_{L_2(E)} = \frac{\left\{x_m^{(2)}\right\}^2}{120}{\Delta t_E}^5 + \frac{\left\{x_m^{(3)}\right\}^2}{30240}{\Delta t_E}^7 + \frac{x_m^{(2)} x_m^{(4)}}{4032}{\Delta t_E}^7 + \mathcal{O}\left({\Delta t_E}^9\right) \quad \text{(A.9)}$$

where we have used the fact that

$$\Psi_E = \frac{1}{2} x_m^{(3)} {\Delta t_E}^2 + \mathcal{O}\left({\Delta t_E}^4\right) \quad \text{(A.10)}$$

when the nodal values are exact. This shows that the second derivative of $x(t)$ is the main contribution to the error, which clearly agrees to our intuition. We notice also that the leading term estimates the error with second-order accuracy. It is, of course, exact for quadratic functions. It is then tempting to define it as a second-order error estimator neglecting the rest of the terms. If this is to be done, we can replace $x_m^{(2)}$ by a second-order difference approximation, without altering the accuracy of the estimate. Therefore we can write

$$\|x - u_E\|^2_{L_2(E)} = \frac{{\Delta f_E}^2 {\Delta t_E}^3}{120} + \mathcal{O}\left({\Delta t_E}^7\right) \quad \text{(A.11)}$$

where f denotes the first derivative of $x(t)$. Now, we remark that the leading term fails to estimate the error accurately in the neighborhood of an inflection point where $x_m^{(2)}$ would be extremely small no matter what difference approximation is employed. Clearly, in this case, the error must be estimated by a quantity which carries the information of $x^{(3)}$. Since the error terms of second-order difference approximations to $x_m^{(2)}$ generally do not contain $x^{(3)}$, it follows that the largest term of such is only the second term in (A.9). However this means also that the error will be $\mathcal{O}\left({\Delta t_E}^2\right)$ smaller than that in the other region. Besides, since inflection points will exist usually at a finite number of discrete points, the failure might be tolerable if we are interested only in relatively large errors. But if that is not the case, we must retain the second term. Then, it may be convenient to replace $x_m^{(3)}$ by Ψ_E by using (A.10). We thus obtain

$$\|x - u_E\|_{L_2(E)}^2 = \frac{{\Delta f_E}^2 + \frac{16}{7}\Psi_E^2}{120}{\Delta t_E}^3 + \mathcal{O}\left({\Delta t_E}^7\right). \tag{A.12}$$

Note that we have included only a part of the next largest term which is relevant to $x_m^{(3)}$. Therefore the accuracy is improved near inflection points only, and so the overall accuracy of the estimate remains second order.

Next we consider the solution of ordinary differential equations where the nodal values are computed by a numerical scheme. In this case, we can deduce from (A.8) that

$$\|x - u_E\|_{L_2(E)}^2 = \frac{{\Delta f_E}^2 {\Delta t_E}^3}{120} + \mathcal{O}\left({\Delta t_E}^6\right) \tag{A.13}$$

provided the nodal approximation is of $\mathcal{O}\left({\Delta t_E}^3\right)$, and that we recover (A.11) with the nodal approximations higher than third order. Therefore we must use, at lowest, third-order methods to compute the nodal values in order for the leading term to estimate the error. This is possible in most cases, for example using the fourth-

order Runge-Kutta method for initial value problems or boundary value problems via shooting technique. Then the leading term can be used to estimate $\|x-u_E\|^2_{L_2(E)}$ with second-order accuracy, and (A.12) can also be used to improve it near inflection points. Yet, even if the leading term is disqualified for the error estimator, it will still be a useful quantity. Suppose that a perfect scheme were devised which finds the exact nodal solution on a grid with a finite number of elements. Then, it is seen from (A.9) that even with such a scheme, the error, expressed by $\|x-u_E\|^2_{L_2(E)}$, cannot be driven to zero, and that it will be dominated by the leading term or the one in (A.13). Hence the leading term dominates the error components that are incurable by numerical schemes. And we see that it is possible to control this error component only through the adjustment of the grid. In other words, minimization of this quantity can be considered as a theme of the selection or the adjustment of the grid.

A.2.3 Error estimates

We now consider L_2 error of piecewise linear approximations to one-dimensional submanifolds in $\mathbf{R}^n$. The extension of the previous analysis to one-dimensional submanifolds in higher dimensions is in fact straightforward. Suppose we have a piecewise linear continuous approximation in $\mathbf{R}^n$ (See Figure A.2)

$$\mathbf{u} = (u_1, \cdots, u_n) = (u_1(t), \cdots, u_n(t)) \tag{A.14}$$

where $t \in I = [0,1]$ which is again divided into a set of elements $\{E\}$, to a smooth one-dimensional submanifold parameterized by

$$\mathbf{x} = (x_1, \cdots, x_n) = (x_1(t), \cdots, x_n(t)). \tag{A.15}$$

Here n denotes the dimension of the ambient space, and thus $n = 1$ corresponds to functions in one dimension. Within each element, we have a local linear function u_{iE}

similar to (A.1) for each coordinate. In n dimensions, it would be natural to define the local L_2 error by

$$\|\mathbf{x}-\mathbf{u}\|^2_{L_2(E)} = \int_E \sum_{i=1}^{n} \left(x_i(t) - u_{iE}(t)\right)^2 dt. \tag{A.16}$$

Geometrically, the quantity thus defined is the measure of the square of the area of the surface formed by joining the two submanifolds by line segments at points with the same t. Then the previous analysis applies to each i, and introducing the notation $f_i = dx_i/dt$, we have the result

$$\|\mathbf{x}-\mathbf{u}\|^2_{L_2(E)} = \frac{\left\{\sum_{i=1}^{n} \Delta {f_{iE}}^2\right\} {\Delta t_E}^3}{120} + \mathcal{O}\left({\Delta t_E}^7\right) \tag{A.17}$$

for a smooth one-dimensional submanifolds with the exact or fourth-order accurate nodal values. Note that the leading term approximates

$$\frac{\left|\frac{d^2\mathbf{x}}{dt^2}\right|^2 {\Delta t_E}^5}{120} \tag{A.18}$$

where the second derivative can be written in terms of the arc length, s,

$$\frac{d^2\mathbf{x}}{dt^2} = \left(\frac{ds}{dt}\right)^2 \frac{d^2\mathbf{x}}{ds^2} + \frac{d^2 s}{dt^2}\frac{d\mathbf{x}}{ds}. \tag{A.19}$$

This means that the leading term contains the information not only about the curvature $\frac{d^2\mathbf{x}}{ds^2}$ but also about the acceleration along the manifold $\frac{d^2 s}{dt^2}$. In particular, if the manifold is parameterized by its arc length, it becomes precisely the measure of the magnitude of the curvature.

Now, taking the square root of (A.17) and expanding, we obtain the following local L_2 error estimate

$$\|\mathbf{x}-\mathbf{u}\|_{L_2(E)} = \mathcal{E}_{2(E)}\left[1 + \mathcal{O}\left({\Delta t_E}^2\right)\right] \tag{A.20}$$

where

$$\mathcal{E}_{2(E)} = \frac{C_E\sqrt{\Delta t_E}}{\sqrt{120}}, \quad C_E = \Delta t_E \sqrt{\sum_{i=1}^{n} {\Delta f_{iE}}^2}. \tag{A.21}$$

Note that C_E is related to the local average error,

$$\sqrt{\frac{1}{\Delta t_E}\int_E \sum_{i=1}^{n} \left(x_i(t) - u_{iE}(t)\right)^2 dt} \;=\; \sqrt{\frac{\|\mathbf{x}-\mathbf{u}\|^2_{L_2(E)}}{\Delta t_E}} \;=\; \frac{C_E}{\sqrt{120}}\left[1 + \mathcal{O}\left({\Delta t_E}^2\right)\right]. \tag{A.22}$$

In a similar manner, we can estimate the global L_2 error defined by

$$\|\mathbf{x}-\mathbf{u}\|_{L_2(I)} = \sqrt{\int_I \sum_{i=1}^{n} \left(x_i(t) - u_i(t)\right)^2 dt}. \tag{A.23}$$

Clearly we have

$$\|\mathbf{x}-\mathbf{u}\|^2_{L_2(I)} = \sum_{\{E\}} \|\mathbf{x}-\mathbf{u}\|^2_{L_2}. \tag{A.24}$$

Then, inserting (A.17), taking the square root and expanding, we obtain

$$\|\mathbf{x}-\mathbf{u}\|_{L_2(I)} = \mathcal{E}_{2(I)}\left[1 + \mathcal{O}\left({\Delta t_E}^2\right)\right] \tag{A.25}$$

where

$$\mathcal{E}_{2(I)} = \sqrt{\sum_{\{E\}} \mathcal{E}^2_{2(E)}} = \sqrt{\sum_{\{E\}} \frac{{C_E}^2 \Delta t_E}{120}} \tag{A.26}$$

The accuracy of $\mathcal{E}_{2(E)}$ and $\mathcal{E}_{2(I)}$ as error estimators, as mentioned earlier, will be second-order with the nodal approximation better than third-order, and first-order with third-order nodal approximations, except in the neighborhood of inflection points. To improve it near inflection points as discussed earlier, it can be shown, extending (A.12), that we need to modify C_E as follows.

$$C_E = \Delta t_E \sqrt{\sum_{i=1}^{n} \left\{ {\Delta f_{iE}}^2 + \frac{16}{7}\Psi^2_{iE} \right\}} \tag{A.27}$$

where $\Psi_{iE} = \frac{\Delta u_{iE}}{\Delta t_E} - x^{(1)}_{im}$. Although the error estimators thus defined require the knowledge of the first derivatives at nodes, they do not have to be exact. The estimates will be second order accurate as long as the nodal values approximating the first derivatives f_i are second order accurate at least. If the parameterization $x(t)$ is available, we may apply a finite-difference approximation to compute $f(t)$ using a

suitable choice of a small interval. In the case of initial-value problems for ordinary differential equations, f is always available as the right hand side of the equations.

A.2.4 Numerical tests for accuracy

This section gives the results that verify the accuracy of the error estimate. First we consider the case where the nodal values are exact, i.e. linear interpolation of a function. Then we consider the solution of an initial-value problem for ordinary differential equations. In the latter, we discuss how numerical schemes affect the accuracy of the error estimator.

We consider approximating the following function

$$x(t) = e^{-3t}\sin(4\pi t) \tag{A.28}$$

by a piecewise linear function on a uniform grid with N_E elements. The function and the approximation with $N_E = 5$ are shown in Figure A.3. The computed errors are given in Table A.1, where the actual L_2 errors were computed by using, within each element, the five-point Gaussian quadrature formula which is exact for 9th order polynomials and therefore is sufficiently accurate for our comparison purpose. Note that all the computations in this paper were performed with double precision, but the results have been truncated for brevity. The table shows a very good agreement between the actual L_2 error and $\mathcal{E}_{2(I)}$ for which we used C_E defined by (A.21) and $f(t)$ is computed by using the central difference formula with a step size 1.0E-05 which was found to give the identical result as that obtained by using the exact derivative of (A.28). In the last column of the table, the relative errors, defined by $\left|1 - \frac{\mathcal{E}_{2(I)}}{\|x-u\|_{L_2(I)}}\right|$, are shown. Starting with 1.15% error for $N_E = 5$, the relative error goes down as the grid is refined. The rate of convergence is found to be 1.98 which verifies the result in the last section. As mentioned earlier, the local L_2 error

N_E	$\|x-u\|_{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
5	$1.947E-01$	$1.722E-01$	$1.152E-01$
10	$4.657E-02$	$4.513E-02$	$3.100E-02$
20	$1.173E-02$	$1.164E-02$	$7.740E-03$
40	$2.941E-03$	$2.935E-03$	$1.933E-03$
80	$7.359E-04$	$7.356E-04$	$4.833E-04$

Table A.1: The results for (A.28)

cannot be predicted accurately near inflection points. To see this, we computed the relative error locally for each element with N_E =160. The plots are shown in Figure A.4. Note that the ordinates of the lower three plots are logarithmically scaled. On the top, the first derivative $f(t)$ is plotted so that the inflection points can be easily located, and the upper-middle one is the plot of the elementwise relative error $|1-\frac{\mathcal{E}_{2(E)}}{\|x-u\|_{L_2(E)}}|$ plotted in the middle of each element. It is seen that the relative error is significantly larger near inflection points than in the other region. However, as seen in the plot in the lower-middle, the actual local L_2 errors $\|x-u\|_{L_2(E)}$ are significantly smaller near inflection points than in the other region. The relative error obtained with the modification (A.27) is in the plot on the bottom. It is seen that the errors have been reduced nearly by an order of magnitude at inflection points.

Next we consider numerical solutions of the following linear initial-value problem.

$$\frac{dx(t)}{dt} = -c_s x + c_s \sin(at) + \cos(at) \qquad t \in [0,1] \tag{A.29}$$

with $c_s = 20$ and $a = 6$. The exact solution is given by

$$x(t) = e^{-c_s t} + \sin(at) \tag{A.30}$$

which is shown in Figure A.5 with a numerical solution. The equation was integrated by a third-order explicit Runge-Kutta method[39] and the classical 4th order Runge-Kutta method. Table A.2 shows the result for the third-order method. The actual

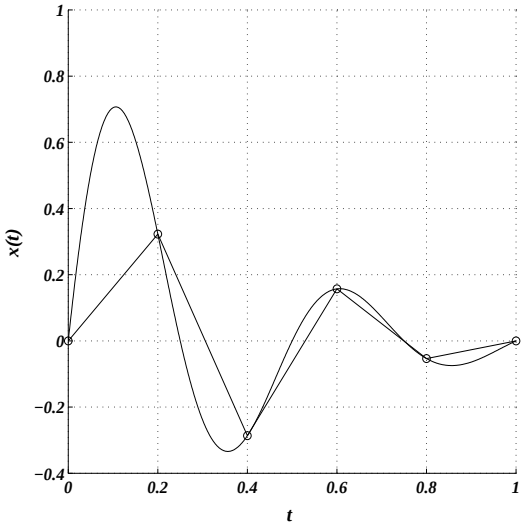

Figure A.3: Function (A.28) and a piecewise linear representation on a uniform grid with N_E =5.

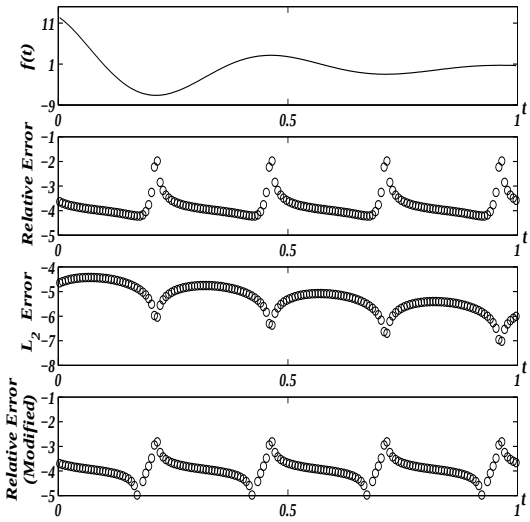

Figure A.4: Top: The first derivative of (A.28). Upper-Middle: Elementwise relative error. Lower-Middle: Actual local L_2 errors. Bottom: Elementwise relative error with the modification (A.27). The last three are plotted in the middle of each element with N_E =160.

L_2 errors were computed in the same way as in the previous section, and $\mathcal{I}_2$ was computed without the modification (A.27). We see that $\mathcal{E}_2$ estimates the actual error fairly well in the sense that their first digits agree except for $N = 80$. The rate of convergence of the relative error is 1.11 as we expected. For the fourth-order method, it is seen in Table A.3 that the errors are estimated more accurately, for example the relative error is 8% for $N_E = 20$ while it is 33% for the third-order method. The rate of convergence of the relative error is 2.00 as we expected. We also applied the modified Euler's method which is only second-order accurate. Including this, Figure A.6 shows the plot of the convergence histories. As we expected, using a second-order method deteriorates the accuracy of the error estimate, and the relative

N_E	$\|x-u\|_{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
20	$1.07705E-02$	$1.43262E-02$	$3.30132E-01$
40	$3.13290E-03$	$3.69313E-03$	$1.78821E-01$
80	$8.65301E-04$	$9.30703E-04$	$7.55830E-02$
160	$2.25419E-04$	$2.33147E-04$	$3.42850E-02$
320	$5.73809E-05$	$5.83164E-05$	$1.63025E-02$

Table A.2: 3rd order Runge-Kutta Method.

N_E	$\|x-u\|_{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
20	$1.42137E-02$	$1.43262E-02$	$7.92006E-03$
40	$3.68583E-03$	$3.69313E-03$	$1.98266E-03$
80	$9.30242E-04$	$9.30703E-04$	$4.95826E-04$
160	$2.33118E-04$	$2.33147E-04$	$1.23966E-04$
320	$5.83146E-05$	$5.83164E-05$	$3.09921E-05$

Table A.3: 4th order Runge-Kutta Method.

error remains large for any grids.

A.3 Algorithm

In this section, we give a basic idea of the algorithm that generates a node distribution to produce a smooth piecewise linear approximation. It is based on an equidistribution property which is discussed below. The algorithm is described for a simple example.

A.3.1 Equidistribution

It is desired that the node distribution is determined such that the quantity $\mathcal{E}_{2(E)}$ is minimized. But since a direct minimization seems rather difficult, we propose to place the nodes so as to equally distribute C_E over the elements, thus concentrating the nodes in the region of large $\left|\frac{d^2\mathbf{x}}{dt^2}\right|$ or equivalently equidistributing the local average error. We advocate this strategy also because of the following useful property.

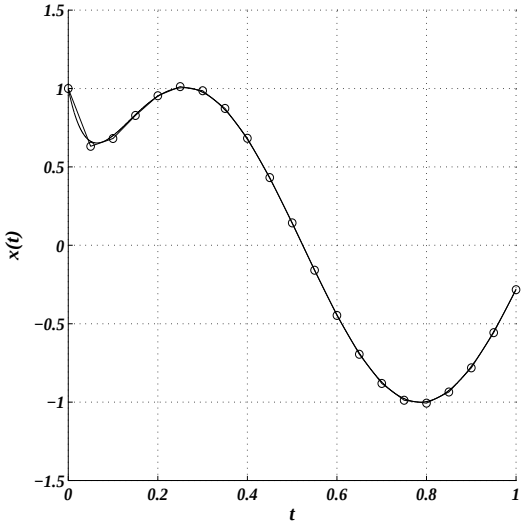

Figure A.5: The exact solution (A.30) with $c_s = 1.0$, and the numerical solution by a third-order Runge-Kutta Method with N_E =20.

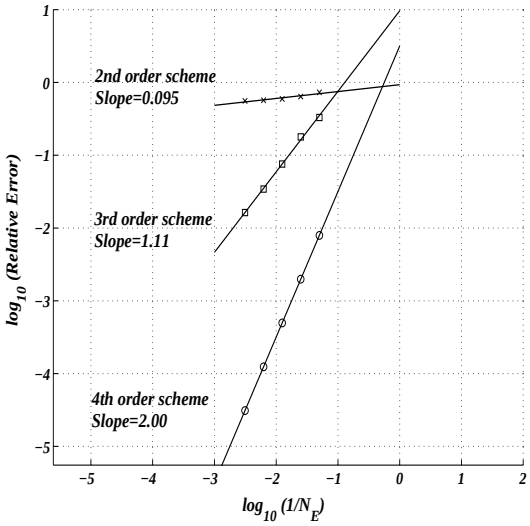

Figure A.6: $-log_{10}(N_E)$ vs. log_{10} (Relative Error) and the linear least-squares fits.

Suppose that we generated nodes such that C_E is equidistributed. Then we find

$$\mathcal{E}^2_{(I)} = \sum_{\{E\}} \frac{{C_E}^2 \Delta t_E}{120} = \sum_{\{E\}} \frac{C^2 \Delta t_E}{120} = \frac{C^2}{120} \tag{A.31}$$

where $C = C_E$ for all $E \in \{E\}$. This shows that the global L_2 error becomes identical to the local average error on equidistribution grids. Also, more importantly, we notice that it provides us with a means to determine C for a desired L_2 or the local average error. This will be useful when we seek an approximation that gives a desired error. Thus this leads us to consider an algorithm that generates nodes such that $C_E = C$ for every element to be generated. Note that the equidistribution implies a uniform spacing for quadratic functions whose curvature is constant.

At this point, it is not clear yet if the grid thus generated gives a better approximation than that with the uniform grid of the same size. To investigate this, we ask how the equidistribution affects the first variation of $\mathcal{E}^2_{(I)}$. For this purpose, without loss of generality, we may consider the case $n = 1$ with the exact nodal values. Sup-

pose that we have successfully equidistributed $C_E = \Delta t_E |\Delta f_E|$ over $\{E\}$. We give a small perturbation dt to a node at $t = t_j$. The first variation of $\mathcal{E}^2_{(I)}$ is then given by

$$\begin{aligned} d\mathcal{E}^2_{(I)} = & \quad \frac{1}{40}\left[\Delta {t_L}^2 \Delta {f_L}^2 - \Delta {t_R}^2 \Delta {f_R}^2\right] dt \\ + & \quad \frac{1}{60}\left[\Delta {t_L}^3 \Delta f_L - \Delta {t_R}^3 \Delta f_R\right] f^{(1)}(t_j) dt \end{aligned} \tag{A.32}$$

where $f^{(1)} = df/dt$, and the subscripts, L and R, denote the elements that share the node j. Note that the both sides are $\mathcal{O}\left(\Delta t^4\right)$, meaning that $\mathcal{E}^2_{(I)}$ and the gradient go down at the same rate. The first term vanishes by the assumption, and we have

$$\frac{\partial \mathcal{E}^2_{(I)}}{\partial t_j} = \frac{1}{60}\left[\Delta {t_L}^3 \Delta f_L - \Delta {t_R}^3 \Delta f_R\right] f^{(1)}(t_j). \tag{A.33}$$

Note that this shows that we achieve a minimum for a quadratic function because the equidistribution implies a uniform grid, and that it is a minimum of the true L_2 error because $\mathcal{E}^2_{(I)}$ becomes identical to the L_2 error for quadratic functions. Yet another interesting observation is that if $\Delta f_R \Delta f_L < 0$, i.e. nonconvex, the only way to achieve a minimum is to place t_j at the inflection point of $x(t)$ where $f^{(1)}$ vanishes. Next we assume that $\Delta f_R \Delta f_L > 0$ and that $\Delta t_L \neq \Delta t_R$, and write, using the assumption of the equidistribution,

$$\frac{\partial \mathcal{E}^2_{(I)}}{\partial t_j} = \frac{C}{60}\left[\Delta {t_L}^2 - \Delta {t_R}^2\right] f^{(1)}(t_j) \tag{A.34}$$

where $C = \Delta t_R \Delta f_R = \Delta t_L \Delta f_L$. It is clear that we cannot have a minimum in this case. However, if we replace $f^{(1)}(t_j)$ by a first-order finite difference approximation in each element, we obtain

$$\begin{aligned} \frac{\partial \mathcal{E}^2_{(I)}}{\partial t_j} = & \quad \frac{C}{60}\left[\Delta {t_L}^2 \left\{\frac{\Delta f_L}{\Delta t_L} + \frac{f^{(2)}(t_j)}{2}\Delta t_L + \mathcal{O}\left(\Delta {t_L}^2\right)\right\}\right. \\ & \left. - \Delta {t_R}^2 \left\{\frac{\Delta f_R}{\Delta t_R} - \frac{f^{(2)}(t_j)}{2}\Delta t_R + \mathcal{O}\left(\Delta {t_R}^2\right)\right\}\right]. \end{aligned} \tag{A.35}$$

By the assumption and also noting that C is $\mathcal{O}\left(\Delta t^2\right)$, we finally get

$$\frac{\partial \mathcal{E}_{(I)}^2}{\partial t_j} = \frac{C}{120}\left[{\Delta t_L}^3 + {\Delta t_R}^3\right] f^{(2)}(t_j) + \mathcal{O}\left(\Delta t^6\right). \tag{A.36}$$

Hence the right hand side is now $\mathcal{O}\left(\Delta t^5\right)$, implying that a smaller gradient can be achieved by equidistribution and that the gradient goes down faster than $\mathcal{E}_{(I)}^2$ does, implying that the error is closer to a minimum. Therefore, we conclude that the equidistribution grid give, if not minimum, a smaller error than the uniform grid of the same size.

A.3.2 Algorithm

Given a function $x(t)$ in I together with its first derivative, we consider constructing a piecewise linear continuous approximation for a given error C where C is a desired local average error multiplied by $\sqrt{120}$. The nodal values are assigned simply by $x(t)$, and therefore the error estimates are second order accurate. Now the idea is to generate nodes successively starting from the initial point $(t_0, x(t_0)) = (0, x(0))$ such that $C_E = C$ over all the elements to be constructed. To generate the next node t_{j+1} from t_j which together define the element $E \in \{E\}$, we need to iterate for t_{j+1} until we have

$$(t_{j+1} - t_j)|\Delta f_E| - C = 0. \tag{A.37}$$

where $\Delta f_E = f(t_{j+1}) - f(t_j)$. A good iteration formula is

$$t_{j+1}^{(k+1)} = t_j + \left[\frac{C(t_{j+1}^{(k)} - t_j)^{p-1}}{|\Delta f_E^{(k)}|}\right]^{\frac{1}{p}} \tag{A.38}$$

where the superscript $k(\geq 1)$ indicates the number of iterations, p is a positive real number. It is easy to show that (A.37) is satisfied at convergence. The role of p is to damp an excessively large change, i.e. an extremely small Δf_E^k which will occur when the iteration enters a region that is near an inflection point or where the curve

is locally linear. Also the modification (A.27) will be effective to prevent the zero division. But if the curve is truly linear, i.e. $|\Delta f_E^k|$ as well as Ψ_E are identically zero, any choice of p or the modification (A.27) will not work. In this case, a practical tip is to add a small constant in the denominator, thus adding a small curvature δ. Yet, a more sophisticated way to add the artificial curvature is to add the term in the following form.

$$\delta = \frac{C}{\tilde{h}} \exp\left(-\lambda |\Delta f_E^{(k)}|\right) \tag{A.39}$$

where $\tilde{h}$ is the desired spacing in the region where the curve is linear and λ is a constant that controls the effect of the artificial curvature. This will, of course, destroy the error equidistribution property, but will not increase the error because its effect is to increase the nodes.

A.3.3 Properties of the iteration formula

For simplicity, we assume that the function in consideration is quadratic, which would be approximately valid locally in a small subdomain of nonquadratic functions unless the curvature is zero. We write (A.38) as

$$\Delta t^{(k+1)} = Q^{1/p} (\Delta t^{(k)})^{1-\frac{2}{p}} \tag{A.40}$$

where

$$\Delta t^{(k+1)} = t_{j+1}^{(k+1)} - t_j, \quad Q = \frac{C}{|\Delta f_E^{(k)}|/\Delta t^{(k)}}. \tag{A.41}$$

For quadratic functions, $\Delta f/\Delta t$ represents the constant curvature exactly, and therefore Q is constant. Observe that for the choice $p = 2$, (A.40) becomes

$$\Delta t^{(k+1)} = \sqrt{Q}. \tag{A.42}$$

That is to say, the iteration is independent of k as well as the initial guess, and will converge at the first try for any initial values, producing uniform node distribution

in t. Of course, we then achieve the equidistribution.

$$C_E^{(k+1)} = \Delta t^{(k+1)} |\Delta f_E^{(k+1)}| = \left\{\Delta t^{(k+1)}\right\}^2 \frac{C}{Q} = C. \tag{A.43}$$

In general, we have, for quadratic functions,

$$\Delta t^{(k+1)} = \sqrt{Q} \left[\frac{\Delta t^{(1)}}{\sqrt{Q}}\right]^{(1-\frac{2}{p})^k}. \tag{A.44}$$

It can be easily proved by induction that this satisfies (A.40). In terms of C_E, we have

$$C_E^{(k+1)} = \Delta t^{(k+1)} |\Delta f_E^{(k+1)}| = C \left[\frac{\left\{\Delta t^{(1)}\right\}^2}{Q}\right]^{(1-\frac{2}{p})^k}. \tag{A.45}$$

It is straightforward to show that for $p > 1$ we have

$$\left|1 - \frac{2}{p}\right| < 1 \tag{A.46}$$

and $C_E^{(k+1)}$ will converge to the limit C, and that for $p \leq 1$ we find

$$1 - \frac{2}{p} \leq -1, \tag{A.47}$$

and hence it will not converge. Therefore, the value of p must be greater than 1 for the iteration to converge. It is also important to note that it takes longer to converge with a larger p since $\left|1 - \frac{2}{p}\right| \to 1$ as $p \to \infty$. For nonquadratic functions with nonvanishing curvature, we therefore conclude that $p = 2$ give the fastest convergence which takes only one iteration provided the spacing Δt that achieves $C_E = C$ is so small that the curvature is locally constant. For functions with inflection points, however, a large value must be assigned to p to make the iteration proceed as mentioned in the last subsection although this will make the iteration take longer in convex part of the function. Note also that the iteration will converge faster for smaller C, i.e. smaller Δt, since the function will behave more and more quadratically in the neighborhood of such a small subdomain.

A.4 Applications

A.4.1 Approximation of functions and curves

In this section, we describe an algorithm to construct an accurate piecewise linear continuous approximation to a given curve in $\mathbf{R}^n$, with the equidistribution property. The curve is assumed to be parameterized analytically or by some approximation such as cubic splines. The main objective is to determine the distribution of the nodes in the parameter space such that the equidistribution is achieved. The results are however shown for the case $n = 1$ only.

Algorithm

Given an L_2 error, we compute C such that $\mathcal{E}_2$ is equal to the specified error, and generate nodes successively starting from an initial point $(t_0, x_1(t_0), \cdots, x_n(t_0)) = (0, x_1(0), \cdots, x_n(t_0))$ such that $C_E = C$ for all the elements to be generated. The iteration formula for t_{j+1}, a generalization of (A.38) to n dimensions, is

$$t_{j+1}^{k+1} = t_j + \left[\frac{C}{C_E}\right]^{\frac{1}{p}} (t_{j+1}^k - t_j). \tag{A.48}$$

where k indicates the number of iterations, p is a positive real number and

$$C_E = (t_{j+1}^k - t_j) \left\{ \sqrt{\sum_{i=1}^{n} \left\{ {\Delta f_{iE}}^2 + \frac{16}{7} \Psi_{iE}^2 \right\}} + \delta \right\} \tag{A.49}$$

The second term in the square root may be removed for strictly convex functions. If it is desired to have a uniform grid spacing $\tilde{h}$ in the region where the curve is linear, we may set

$$\delta = \frac{C}{\tilde{h}} \exp\left(-\lambda \sqrt{\sum_{i=1}^{n} {\Delta f_{iE}}^2 + \frac{16}{7} \Psi_{iE}^2} \right) \tag{A.50}$$

where λ is a user-specified constant that limits the influence of this artificial curvature in the region where such correction is not desired.. The first derivative $f(t)$ will be

computed by a simple central difference formula,

$$f(t_j) = \frac{x(t_j + h) - x(t_j - h)}{2h} \tag{A.51}$$

with h =1.0E-5. The initial guess is always set to be $t_j + (t_j - t_{j-1})$ assuming the curve is locally quadratic for $j > 0$, and to be 0.001 as a sheer guess for $j = 0$. The iteration is taken to be converged when

$$\left| \frac{C_E}{C} - 1 \right| < \text{TOL} \tag{A.52}$$

where TOL is set to be 1.0E-03, thus allowing 0.1 % error. The process will be terminated when a new node exceeds $t = 1$. The node distribution in I will be determined independently, and therefore the nodal values can be assigned by the given parameterization at the end of the node generation process. The algorithm is summarized as follows.

1. compute C by $C = \sqrt{120}\ \mathcal{E}_{2(I)}$ for a desired error $\mathcal{E}_{2(I)}$;
2. set $t_{j+1} = t_j + (t_j - t_{j-1})$ $\quad$ ($t_{j+1} = 0.001$ for $j = 0$);
3. compute a new location t_{j+1} by (A.48);
4. if (A.52) is not satisfied, go to 3;
5. if $t_{j+1} < 1$, go to 2 (Next node).
6. choose the node at $t = 1$: end of the node-generation process.
7. assign $x(t)$ at all the nodes generated.

In the results that follow, the endpoint has been chosen as follows. The last node generated, the one that exceeds $t = 1$, will be moved to $t = 1$ if the size of the element defined by $t = 1$ and the previous node is greater than 20 % of its previous element size. Otherwise, the previous node will be moved to $t = 1$.

$\mathcal{E}_{2(I)}$	N	noi	$\|x-u\|_{L_2(I)}$	L_∞	L_{2U}
1.0E-02	6	7.4	4.711E-03	9.932E-03	6.053E-02
1.0E-04	40	2.5	6.069E-05	9.999E-05	2.073E-03
1.0E-06	384	1.2	7.409E-07	1.000E-06	2.199E-05

Table A.4: Example (a): $p = 2$.

$\mathcal{E}_{2(I)}$	N	noi	$\|x-u\|_{L_2(I)}$	L_∞	L_{2U}
1.0E-02	15	5.2	1.018E-02	1.260E-02	1.144E-01
1.0E-04	135	2.8	9.979E-05	1.086E-04	3.723E-03
1.0E-06	1337	1.7	1.000E-06	1.048E-06	3.863E-05

Table A.5: Example (b): $p = 3$.

Results

We consider the case $n = 1$, i.e. approximation of functions in one dimension. Numerical tests were performed for the following four different functions.

(a) $at + (1-a)\left\{1 - e^{(-t/\epsilon)}\right\} / \left\{1 - e^{(-1/\epsilon)}\right\}$ with $\epsilon = 0.04$ and $a = 0.6$

(b) $a(t-0.5)^2 - \frac{a}{4}(1 + 8\epsilon t) + (1 + 2\epsilon a)\left\{1 - e^{(-t/\epsilon)}\right\} / \left\{1 - e^{(-1/\epsilon)}\right\}$ with $\epsilon = 0.01$ and $a = 3.5$

(c) $\tanh\{20(t-0.5)\}$

(d) $10e^{(-10t)} + 20/\{1 + 400(t-0.7)^2\}$

(a) is a strictly convex function, and (b) has a single inflection point and also exhibits a large variation near $t = 0$. (c) is symmetric with a single inflection point in the middle, and (d) has two inflection points. Examples (c) and (d) are taken from [5]. We set $\delta = 0$ for all the cases.

The results are summarized in Table A.4 to A.7 where $\mathcal{E}_{2(I)}$ is the specified L_2 error, N is the total number of nodes, noi is the arithmetic average of the numbers of iterations to find the next node, $\|x-u\|_{L_2(I)}$ is the actual L_2 error computed by the

$\mathcal{E}_{2(I)}$	N	noi	$\|x-u\|_{L_2(I)}$	L_∞	L_{2U}
1.0E-02	12	33.4	5.870E-03	1.042E-02	3.311E-02
1.0E-04	104	16.2	7.507E-05	1.001E-04	7.930E-04
1.0E-06	1026	8.7	8.899E-07	1.001E-06	8.026E-06

Table A.6: Example (c): $p = 8$.

$\mathcal{E}_{2(I)}$	N	noi	$\|x-u\|_{L_2(I)}$	L_∞	L_{2U}
1.0E-01	25	10.5	9.483E-02	1.084E-01	3.685E-01
1.0E-03	234	6.4	9.978E-04	1.033E-03	4.627E-03
1.0E-05	2324	3.4	9.993E-06	1.003E-05	4.660E-05

Table A.7: Example (d): $p = 4$.

five point Gaussian quadrature formula, L_∞ is the maximum of the local average error, i.e. $L_\infty(\sqrt{\|x-u\|_{L_2(E)}/\Delta t_E})$, and finally L_{2U} is L_2 error on a uniform grid of the same size. The approximations constructed are shown in Figures A.7 to A.10. The results confirm that the specified L_2 errors are accurately achieved except for Examples (a) and (b) where the actual L_2 errors are smaller than expected. This happens because of the violation of the equidistribution in the last element. Since the functions are almost linear near $t = 1$ in both cases, the placement of the last node is very likely to give a smaller error for the last element. Indeed, we found that if we excluded the last element when computing the error, the actual L_2 errors were very close to the specified values. L_∞ norm of the local average errors are compared with $\mathcal{E}_{2(I)}$. A good agreement can be seen for every case. Although the only L_∞ norm is given here, the values do not vary significantly over the elements, again except for the last element, due to the equidistribution. L_2 errors on the uniform grid of the same sizes are given for comparison purposes. In all the cases, it is seen that extra orders of magnitude reduction has been achieved. node is usually a few in locally convex regions, and more in the neighborhood of inflection points. As seen in the tables, the average number of iterations decreases with the number of nodes

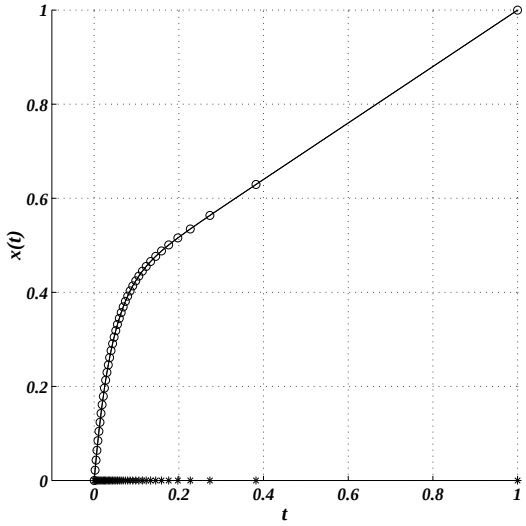

Figure A.7: Example (a). $\mathcal{E}_{2(I)}$ =1.0E-04.

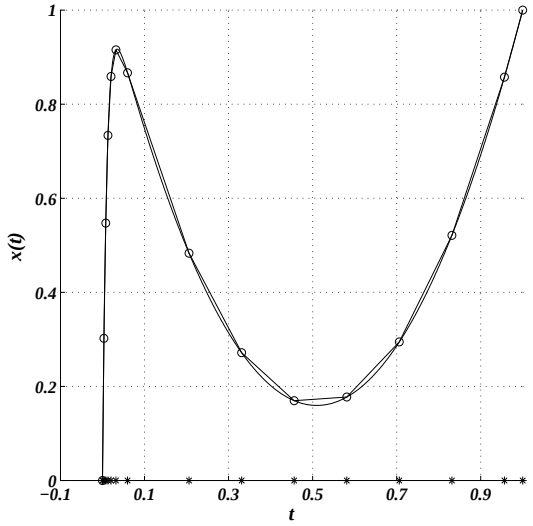

Figure A.8: Example (b). $\mathcal{E}_{2(I)}$ =1.0E-02

increased as we expected. Especially, in the first example which is a convex case, the number of iteration approaches one as the number of nodes increases, which is consistent with our observation in the last section. Similar behavior is observed also in the second example although $p = 3$ was used to deal with an inflection point. Note also that the number of iterations is large for a large p which again is consistent with our observation in the last section.

In some applications, it may be desired to capture the end point accurately. For instance, as shown in Figure A.9, the function is symmetric with respect to $x = 0.5$ but the grid is not quite symmetric, and one might wish to have a symmetric grid. One way to do this is to apply a shooting technique such as the one proposed by Ketzscher and Forth[52] in a related context. An algorithm developed based on this idea is presented in Appendix B.

A.4.2 Numerical integration

The algorithm described in the last subsection was shown to be capable of producing a highly accurate piecewise linear approximation, with a specified L_2 error, to

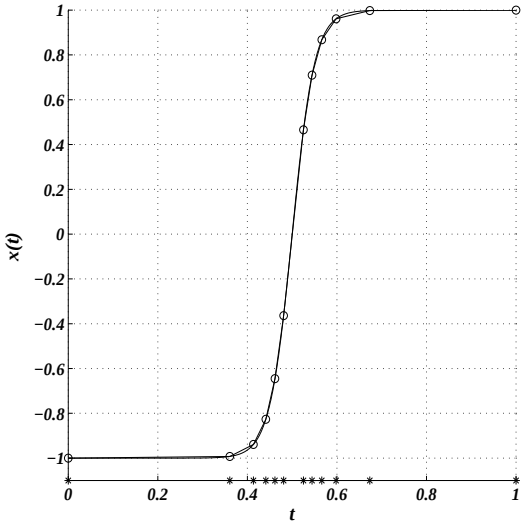

Figure A.9: Example (c). $\mathcal{E}_{2(I)}$ =1.0E-02

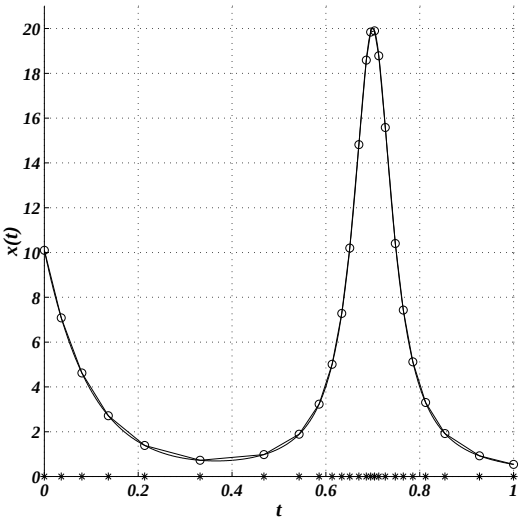

Figure A.10: Example (d). $\mathcal{E}_{2(I)}$ =1.0E-01

a known function. A byproduct is an accurate estimation of the integral of the function. But accurate prediction of the error cannot be made in this case, and only the upper bound of the error can be specified. The results show that the method gives errors lower than, but sufficiently close to, the specified upper bound for practical purposes.

Algorithm

Consider the Cauchy-Schwarz inequality[25],

$$\left| \int_I g(t)h(t)dt \right| \leq \|g(t)\|_{L_2(I)} \|h(t)\|_{L_2(I)} \tag{A.53}$$

where $g(t)$ and $h(t)$ are continuous functions in I. Let $g(t) = x(t) - u(t)$ and $h(t) = 1$, where $x(t)$ is a known function we wish to integrate and $u(t)$ is a piecewise linear approximation to $x(t)$, then we have

$$\left| \int_I x(t) - u(t)dt \right| \leq \|x(t) - u(t)\|_{L_2(I)} \tag{A.54}$$

The Integration Error	The Upper Bound($\mathcal{E}_{2(I)}$)	N
2.50E-03	1.0E-02	6
4.14E-05	1.0E-04	40
5.52E-07	1.0E-06	384

Table A.8: Example (a): $p = 2$.

The Integration Error	The Upper Bound($\mathcal{E}_{2(I)}$)	N
8.07E-3	1.0E-02	15
7.82E-5	1.0E-04	135
7.82E-7	1.0E-06	1337

Table A.9: Example (b): $p = 3$.

which becomes, by (A.25),

$$\left| \int_I x(t)dt - \int_I u(t)dt \right| \leq \mathcal{E}_{2(I)} \left[1 + \mathcal{O}\left({\Delta t_E}^2 \right) \right]. \tag{A.55}$$

The integral of $u(t)$, that is easily computable by the trapezoidal rule, being our numerical approximation to the exact integral, the left hand side represents precisely the error of the integration. Then, the above inequality shows that the integration error is bounded by the L_2 error estimate, $\mathcal{E}_{2(I)}$, with a possible second-order error. Therefore, with the algorithm that can construct $u(t)$ for a given $\mathcal{E}_{2(I)}$, we can specify the upper bound of the integration error within the second-order truncation error. The resulting algorithm is an adaptive step-size control technique for the trapezoidal rule.

The algorithm is exactly the same as before except that the last node, i.e. the one that exceeds $t = 1$, is always pulled down to $t = 1$ to avoid an increase in the upper bound $\mathcal{E}_{2(I)}$, and that the integration of the resulting piecewise linear function must be performed at the end of the process.

1. compute C by $C = \sqrt{120}\ \mathcal{E}_{2(I)}$ for a desired error bound $\mathcal{E}_{2(I)}$;

2. set $t_{j+1} = t_j + (t_j - t_{j-1})$ $\quad (t_{j+1} = 0.001$ for $j = 0)$;

The Integration Error	The Upper Bound($\mathcal{E}_{2(I)}$)	N
4.09E-4	1.0E-02	6
5.11E-6	1.0E-04	104
4.61E-8	1.0E-06	1026

Table A.10: Example (c): $p = 8$.

The Integration Error	The Upper Bound($\mathcal{E}_{2(I)}$)	N
7.47E-02	1.0E-01	25
8.05E-04	1.0E-03	234
8.07E-06	1.0E-05	2324

Table A.11: Example (d): $p = 4$.

3. compute a new location t_{j+1} by (A.48);

4. if (A.52) is not satisfied, go to 3;

5. if $t_{j+1} < 1$, go to 2 (Next node).

6. set $t_{j+1} = 1$: end of the node-generation process.

7. assign $x(t)$ at all the nodes generated, and compute the integral $\int_I u(t)dt$.

Results

We consider numerically integrating over I the four functions used in the last section. These functions can be integrated exactly over I, and therefore we can compute the actual errors of the numerical integration. The results are summarized in Tables A.8, A.9, A.10 and A.11 in which the actual integration error, i.e. the left hand side of (A.55), the upper bound we specified, and the number of nodes generated are shown from the left. For functions (a), (b), and (d), we see that the integrals are very accurately estimated with the errors that are smaller than but very close to the specified upper bounds, which demonstrate the effectiveness of the method. In the example (c), on the other hand, the errors were found to be always

nearly two orders of magnitude smaller than the upper bounds, but this does not mean that we generated more nodes than necessary. This is a special case: by the symmetry of the function, a large part of the quadrature error cancel out. It is easy to show that a perfectly symmetric piecewise linear approximation gives the exact value of the integration (zero) for any number of nodes. The approximations constructed by the algorithm are, however, not perfectly symmetric (see Figure A.9). Therefore, they do not give the exact value, but very small errors due to a large cancellation that still can occur.

A.4.3 Initial value problems for ordinary differential equations

It is well-known that the solution of an $n \times n$ system of ordinary differential equations is a one-dimensional submanifold in $\mathbf{R}^n$ (or the so-called phase space) with its independent variable as a parameter. We consider approximating this submanifold by a piecewise linear continuous function. Although the methodology is basically the same as the previous, we now need to consider numerical schemes that produce the nodal approximation since the parameterization of the solution submanifold is not known.

Algorithm

We consider approximating the solution manifold of an $n \times n$ system of ordinary differential equations,

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, t) \tag{A.56}$$

where $\mathbf{x}(t) = (x_1(t), \cdots, x_n(t))$ and $\mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), \cdots, f_n(\mathbf{x}, t))$ with suitable initial conditions, by a piecewise linear continuous function,

$$\mathbf{u}(t) = (u_1(t), \cdots, u_n(t)). \tag{A.57}$$

Given a desired L_2 error, starting from the initial point $(x_1(0), \cdots, x_n(0))$, we are going to find the grid points successively by the node generation algorithm using the right hand side $\mathbf{f}$ to compute the first derivatives. The algorithm is exactly the same as before except that we need to compute the nodal values by a numerical scheme this time at every iteration for t_{j+1}. The generation process will be terminated when a new node exceeds $t = 1.0$ and the node closest to the end point will be moved to the end point. A natural choice of the scheme would be the classical 4th order Runge-Kutta method, which makes the error estimate second-order accurate. For smooth solutions, the method is expected to produce nodal values accurate enough for our purpose. However, for stiff problems, the scheme is not recommended because of its poor stability property for such problems. Our choice here is a two-stage 4th order implicit Runge-Kutta scheme of Gauss-Legendre type which is A-stable[54]. Note however that A-stable methods does not necessarily imply high accuracy. Its accuracy depends on the step size; the node generation algorithm will play an important role in determining it. The algorithm is summarized as follows.

1. compute C by $C = \sqrt{120}\ \mathcal{E}_{2(I)}$ for a desired error $\mathcal{E}_{2(I)}$;

2. set $t_{j+1} = t_j + (t_j - t_{j-1})$ $\quad$ ($t_{j+1} = 0.001$ for $j = 0$);

3. compute $\mathbf{u}(t_{j+1})$ by a numerical scheme;

4. compute a new location t_{j+1} by (A.48);

5. compute new solutions $\mathbf{u}(t_{j+1})$ by a numerical scheme;

6. if (A.52) is not satisfied, go to 4;

7. if $t_{j+1} < 1$, go to 2 (Next node).

$\mathcal{E}_{2(I)}$	N	noi	$\|x-u\|_{L_2(I)}$	L_{2U}
1.0E-02	21	12.5	1.084E-02	1.37416E-02
1.0E-03	65	10.2	1.017E-03	1.36935E-03
1.0E-04	205	8.5	1.005E-04	1.35032E-04

Table A.12: The results for the scalar case. $p = 5$.

$\mathcal{E}_{2(I)}$	N	noi	$\|\mathbf{x}-\mathbf{u}\|_{L_2(I)}$	L_{2U}
1.0E-01	4	8.3	9.362E-02	1.328
1.0E-02	11	4.6	1.158E-02	8.701E-01
1.0E-03	34	3.5	1.014E-03	2.747E-01

Table A.13: The results for the stiff system. $p = 2$.

If it is known that the solution tends to become linear, including δ will be effective as described earlier, producing a uniform grid in such a region.

Results

We first consider a scalar problem,

$$\frac{dx(t)}{dt} = -c_s x + \left[0.3c_s - a\pi e^{-c_s t}\right]\sin(a\pi t) + 0.3a\pi\cos(a\pi t) \quad t \in [0,1] \tag{A.58}$$

with $c_s = a = 5$ and $x_1(0) = 1$. The exact solution is given by

$$x(t) = e^{-c_s t}\cos(a\pi t) + 0.3\sin(a\pi t). \tag{A.59}$$

Since this solution is smooth, we use the classical 4th order Runge-Kutta method for this problem. The modification (A.27) is not used here for simplicity, and δ is also set zero. The method produced results shown in Table A.12. Although the improvement of the error over the uniform grids is not impressive, the desired errors have been achieved quite accurately for all the cases. As seen in Figure A.11, the nodes are concentrated in the regions where the curvature is relatively large.

The second test case is a stiff 2×2 system of equations taken from [39].

$$\frac{dx_1(t)}{dt} = 998x_1 + 1998x_2 \tag{A.60}$$

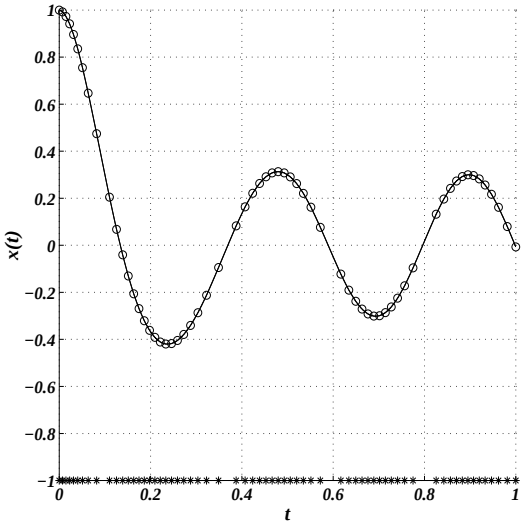

Figure A.11: Numerical and exact solutions for the scalar equation. $\mathcal{E}_{2(I)}$ =1.0E-03

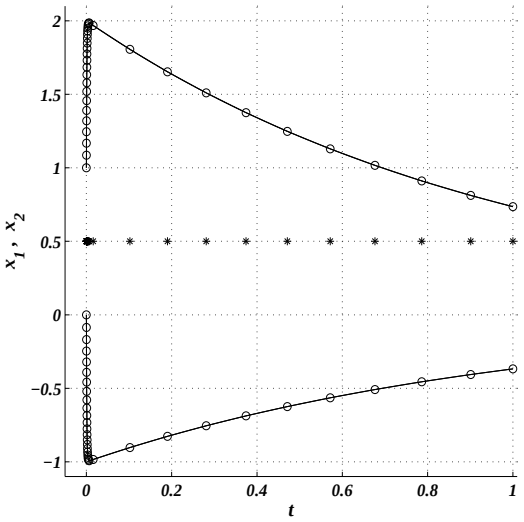

Figure A.12: Numerical and exact solutions for the stiff system. $\mathcal{E}_{2(I)}$ =1.0E-03

$$\frac{dx_2(t)}{dt} = -999x_1 - 1999x_2 \tag{A.61}$$

with the initial conditions $x_1(0) = 1$ and $x_2(0) = 0$. The exact solutions are given by

$$x_1 = 2e^{-t} - e^{-1000t}, \quad x_2 = -e^{-t} + e^{-1000t}. \tag{A.62}$$

We will compute the solution up to $t = 1$. The nodal values are computed by the two-stage 4th order implicit Runge-Kutta scheme. As can be seen in Table A.13, the desired errors have been achieved accurately for all the cases, and also nearly two orders of magnitude reduction of the error over the uniform grid can be seen. The numerical solutions and the exact solutions for $\mathcal{E}_{2(I)}$ =1.0E-03 are shown in Figures A.12 and A.13. We see that the generated grid is very smooth inside as well as outside the transient region. Figure A.14 shows the exact solution(solid line) and the numerical solution(circles) in the phase space which we actually intended to approximate. It is clearly seen, as we expected, that more nodes are placed between t =0 and t =0.005 where the arc length changes very rapidly with respect to t than

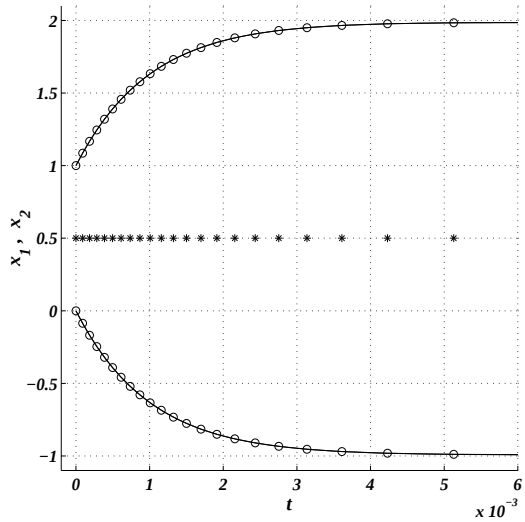

Figure A.13: The transient region of the solutions shown in Figure A.12.

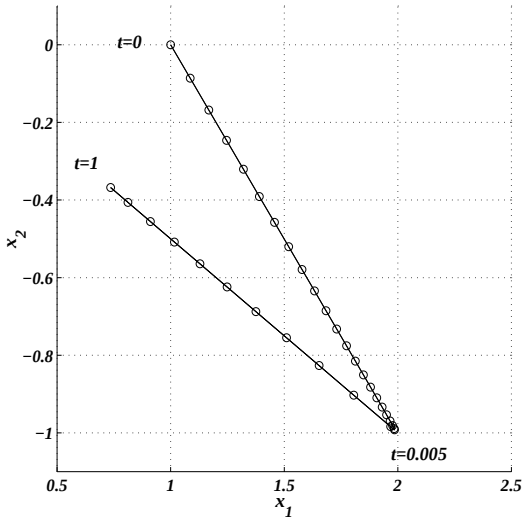

Figure A.14: The numerical and exact solution curves in the phase space.

those in the rest of the curve, and that the nodes are concentrated in the portion with large curvature near t =0.005.

A final remark is on extremely stiff problems. The algorithm was tested for the scalar case with c_s =1.0E+10 and $\mathcal{E}_{2(I)}$ =1.0E-01. We found that the method generated a few nodes in the extremely narrow transient region and about 700 nodes in the smooth region which are too many, but that the equidistribution was accurately achieved on that grid. It seems that this was caused by the large error in $f(u,t)$ produced by the product of c_s and a small error in the solution u. In other words, the method seems to have achieved the equidistribution by adjusting not the step size but the solution. Further study is necessary on this problem.

A.4.4 Boundary value problems for ordinary differential equations

In this section, we describe the application of the node generation algorithm to boundary value problems for ordinary differential equations. It is well known that the problem can be transformed into two initial value problems via the shooting

technique. Therefore our algorithm can be applied just as described in the previous section, but a slight modification is necessary.

Algorithm

We consider approximating the solution of the linear boundary value problem,

$$\frac{d^2x}{dt^2} - p(t)\frac{dx}{dt} - q(t)x = r(t) \quad t \in [0,1] \tag{A.63}$$

with $x(0) = \alpha$ and $x(1) = \beta$, by a piecewise linear continuous function $u(t)$. We assume that $p(t)$, $q(t)$ and $r(t)$ are continuous in I and $q(t) > 0$ in I. Then the problem has a unique solution[51]. It is well known that this unique solution can be written as

$$x(t) = x_1(t) + \frac{\beta - x_1(1)}{x_2(1)}x_2(t) \tag{A.64}$$

where x_1 and x_2 are the solutions of the following *initial value problems* [51].

$$\frac{d^2x_1}{dt^2} = p(t)\frac{dx_1}{dt} + q(t)x_1 + r(t), \quad x_1(0) = \alpha,\ \tfrac{dx_1}{dt}(0) = 0 \tag{A.65}$$

$$\frac{d^2x_2}{dt^2} = p(t)\frac{dx_2}{dt} + q(t)x_2, \quad x_2(0) = 0,\ \tfrac{dx_2}{dt}(0) = 1. \tag{A.66}$$

Therefore, each problem can be solved, rewritten as a first order system, by the algorithm described in the previous subsection. It is however important to notice that we are interested not in the solution manifold of these systems but the solution of (A.63). That is to say, we want to use $f = dx/dt$ to compute C_E. It appears possible to compute f during the integration of the initial value problems since we have from (A.64)

$$f(t) = f_1(t) + \frac{\beta - x_1(1)}{x_2(1)}f_2(t) \tag{A.67}$$

where $f_1 = dx_1/dt$ and $f_2 = dx_2/dt$ which are to be computed together with x_1 and x_2. However, it requires the knowledge of $x_1(1)$ and $x_2(1)$ which are available

$\mathcal{E}_{2(I)}$	N	noi_1	noi_2	$\|x - u\|_{L_2(I)}$	L_{2U}
1.0E-01	10	10.6	22.0	1.394E-01	5.043E-01
1.0E-02	17	10.1	15.8	1.105E-02	3.855E-01
1.0E-03	43	9.8	11.1	9.914E-04	1.518E-01

Table A.14: The results for (A.68). $p = 5$.

only after the integration. This leads us to a two-step method; first solve the initial value problems with $C_E = \Delta t_E \sqrt{\Delta f_{1E}^2 + \Delta f_{2E}^2}$, and second repeat the computation with $C_E = \Delta t_E \sqrt{\Delta f_E^2}$. In the first step, the focus is on to approximate the solution manifold in (x_1, x_2) space accurately because we seek accurate values of $x_1(1)$ and $x_2(1)$. In the second step, we focus on approximating the solution of (A.63) as f is now computable by (A.67). In general, the grids generated in the two steps can be quite different. The algorithm is summarized as follows.

1. compute C by $C = \sqrt{120}\ \mathcal{E}_{2(I)}$ for a desired error $\mathcal{E}_{2(I)}$;

2. Apply the node generation algorithm to the two initial value problems (A.65) and (A.66) with $C_E = \Delta t_E \sqrt{\Delta f_{1E}^2 + \Delta f_{2E}^2}$;

3. Repeat the step 2 with $C_E = \Delta t_E \sqrt{\Delta f_E^2}$.

Results

We consider the following linear problem.

$$\frac{d^2x}{dt^2} + \frac{1}{\varepsilon}\frac{dx}{dt} = \frac{n\pi}{\varepsilon}\left[\varepsilon n\pi \sin(n\pi t) - \cos(n\pi t)\right] \qquad t \in [0,1] \tag{A.68}$$

with the boundary conditions $x(0) = 0$ and $x(1) = 1$, and $\varepsilon = 0.001$ and $n = 1$. Although $q(t) = 0$, there exists a unique solution given by

$$x(t) = \frac{1 - e^{(-t/\epsilon)}}{1 - e^{(-1/\epsilon)}} - \sin(n\pi t). \tag{A.69}$$

We use the two-stage 4th order implicit Runge-Kutta scheme again to integrate the equivalent initial value problems. As can be seen in Table A.14, in which noi_1 is the average of the numbers of iterations at the first step and noi_2 at the second step, the specified errors are achieved accurately for each case. On the other hand, the errors on the uniform grids are significantly larger for each case and decrease very slowly as the grid is refined. The reason for the latter is that no grid points are placed inside the narrow region near $t = 0$ even with N =44. Figures A.15 and A.16 show the exact solution and the numerical solutions obtained at the first and the second steps respectively, for $\mathcal{E}_{2(I)}$ =1.0E-03, where grid points are indicated by stars as before. It can be seen clearly that the two grids differ significantly. As mentioned earlier, this happens because in the first step we seek a good approximation to the solution curve in $x_1 - x_2$ space, not $x(t)$. Although x_2 does have a narrow region of nearly the same width near $t = 0$, its magnitude is of the same order as that width. Therefore the method does not place many points in this region. At this stage, we found that $\|x - u\|_{L_2(I)}$ =3.310E-02; the prescribed error has not been achieved. In the second step, this solution is magnified by the constant $\frac{\beta - x_1(1)}{x_2(1)}$ which was found to be 996.8 after the first step, and therefore the narrow region becomes very important, thus making the method place many nodes within it. As shown in Table A.14, the iteration to generate each node took 10 to 20 on average with $p = 5$. The choice of $p = 5$ was necessary for the iteration to converge near the infection point.

Finally we remark that the method works for extremely stiff boundary value problems. We applied the algorithm to the case ε =1.0E-10 with $\mathcal{E}_{2(I)}$ =1.0E-01 and $p = 5$. The algorithm generated 721 interior nodes, giving the error $\|x - u\|_{L_2(I)}$ =1.045E-01. The resulting grid looks almost identical to the one in Figure A.16 outside the narrow region, but has many more nodes inside. This result is a

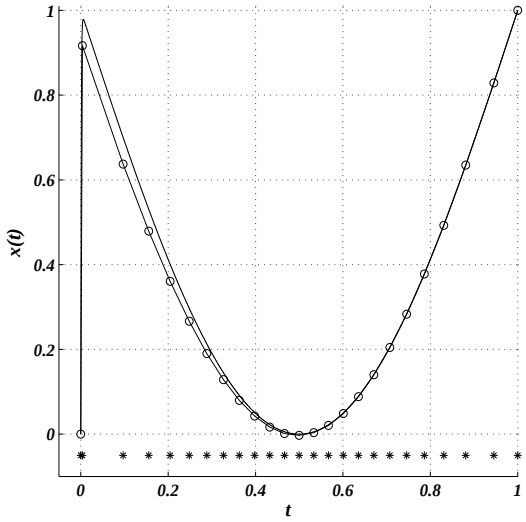

Figure A.15: The exact solution and the numerical solution obtained after the first step.

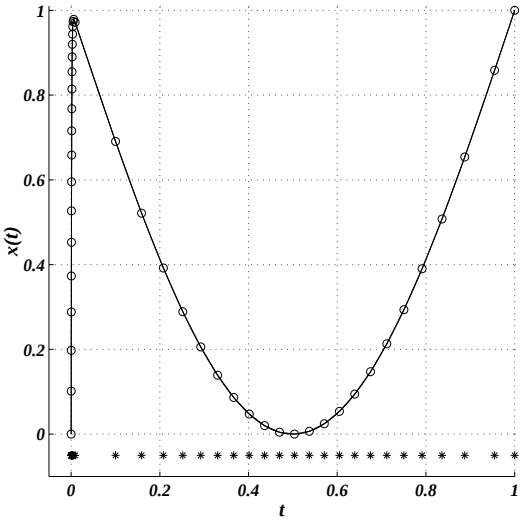

Figure A.16: The exact solution and the numerical solution after the second step.

total contrast to the one for a stiff initial value problem mentioned in the previous section. The reason for this success lies in the fact that the equivalent initial value problems are not really stiff and that the effect of the stiffness appears only in the constant $\frac{\beta - x_1(1)}{x_2(1)}$ and does not affect the numerical solutions.

A.5 Remarks

In this study, the value of p in the algorithm was fixed in the entire domain. Although very effective, it would be more efficient if it could be changed in such a way that a large value is assigned near inflection points and $p = 2$ is assigned in other convex part. This is because the larger the p is the more iterations it takes in convex regions where $p = 2$ is the best value. This would become an important issue when the method is applied to nonlinear stiff ordinary differential equations where the iteration is required for nodal values as well, and also for large scale problems for which the evaluation of the right hand side is very expensive. Further

study is necessary also for extremely stiff initial value problems, concerning accurate evaluations of the first derivatives of the solutions, i.e. the right hand sides.

Compared with node adjustment approach (LSMGM), the node generation approach is more efficient and robust. However, in terms of the extension to higher dimension, its ability may be limited because two-dimensional submanifolds, especially as solutions of partial differential equations, cannot be constructed always by marching from a boundary. In addition, although the expression for the L_2 error estimate has also been derived for a triangular element (See Appendix C), it seems too complicated for a practical use. But it may be possible that simpler error estimates could be found in other norms, which is left as a future work.

APPENDIX B

A Shooting Method for A Node Generation Algorithm

In this appendix, a shooting method is studied to capture the endpoint by adjusting C. The idea is taken from [52] where a shooting method for the construction of an L_1 optimum node distribution is studied. In [52], the nodes are generated successively starting from a pair of nodes to minimize L_1 error, and therefore the position of the last node depends upon the choice of the second node. The second node is then be found by the minimization technique of Brent's, which guarantees that the last node is placed at the specified endpoint. In this study, we need to find the value of C such that

B.1 Node-Generation Algorithm

We consider generating nodes for interpolation of a function $x(t)$ in $t \in I = [0, 1]$ which will equidistribute the leading term of the local average error (See Section A.4.1)

$$\frac{C_E}{\sqrt{120}} = \frac{\Delta t_E}{\sqrt{120}} \sqrt{{\Delta f_E}^2 + \frac{16}{7}\Psi_E^2}. \tag{B.1}$$

where $f(t) = dx/dt$, $\Delta t_E = t_{j+1} - t_j$, $\Delta f_E = f(t_{j+1}) - f(t_j)$,

$$\Psi_E = \frac{\Delta x_E}{\Delta t_E} - f(t_m), \quad t_m = \frac{t_{j+1} + t_j}{2}, \tag{B.2}$$

and E denotes the element defined by two consecutive nodes, t_j and t_{j+1}. The nodes can be generated successively starting from the initial point $(t, x(t)) = (0, x(0))$ once we specify the value of C_E. Let C denote this desired value, then given a node t_j, we must find t_{j+1} such that

$$C_E(t_{j+1}) - C = 0. \tag{B.3}$$

The iteration formula proposed in Chapter 2 is

$$t_{j+1}^{k+1} = t_j + \left[\frac{C}{C_E}\right]^{\frac{1}{p}} (t_{j+1}^k - t_j). \tag{B.4}$$

the superscript k indicates the number of iterations, p is a positive real number. The role of p is to damp an excessively large change, i.e. an extremely small Δf_E^k which will occur when the iteration enters a region that is near an inflection point or where the curve is locally linear. We terminate the iteration when the equidistribution is achieved within 3% error. The node generation algorithm is summarized as follows.

1. compute C by $C = \sqrt{120}\ \mathcal{E}_{2(I)}$ for a desired error $\mathcal{E}_{2(I)}$;

2. set $t_{j+1} = t_j + (t_j - t_{j-1})$ $\quad (t_{j+1} = 0.001$ for $j = 0)$;

3. compute a new location t_{j+1} by (B.4);

4. if $|C_E/C - 1| >$ 1.0E-03, go to 3;

5. if $t_{j+1} < 1.0$, go to 2 (Next node).

Let N denote the total number of nodes generated by the above algorithm, including the initial node. Then note that we always have $t_N > 1.0$. In order to make it finish at $t = 1$, we adjust the value of C by a shooting method. The idea is taken from [52] where a shooting method for the construction of an L_1 optimum node distribution is developed. In [52], the nodes are generated successively starting from a pair of

nodes to minimize L_1 error, and therefore the position of the last node depends upon the choice of the second node. The second node is then found by the minimization technique of Brent's, which guarantees that the last node is placed at the specified endpoint.

B.2 Shooting Method

Given a set of N nodes generated by the node generation algorithm, we consider solving the following nonlinear equation for C.

$$t_N(C) - 1.0 = 0. \tag{B.5}$$

Note that this has many solutions unless N is fixed. For a convex function, this equation has a unique solution for a fixed N since t_N is a monotone function of C [52]. However it is not guaranteed for nonconvex functions, and therefore we allow N to change, relaxing the requirement for the solution. But the solution we seek is the one with the total number of nodes close to the original number N. Since the explicit functional relationship between t_N and C is not known, we use the Secant method to solve the problem. Given another set of nodes with $C^{(2)} = 1.0001C$ and the initial one $C^{(1)} = C$, we therefore compute C by

$$C^{(n)} = C^{(n-1)} - \frac{t_N(C^{(n-1)}) - 1}{\left[t_N(C^{(n-1)}) - t_N(C^{(n-2)})\right] / \left[C^{(n-1)} - C^{(n-2)}\right]} \tag{B.6}$$

where n is the iteration index greater than two, and when generate another set of nodes and also at each iteration, we modify the step 5 such that the node generation terminates if the number of nodes reaches $N + 1$. The iteration will be terminated when the error in capturing the endpoint becomes less than 5% of the size of the last element,

$$\left|t_N(C^{(n)}) - 1.0\right| < 0.05 \left[t_N(C^{(n)}) - t_{N-1}(C^{(n)})\right]. \tag{B.7}$$

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	6	9	6	3.6504E-03
1.0E-04	40	3	40	6.2812E-05
1.0E-06	383	4	383	7.6735E-07

Table B.1: Example (a): $p = 2$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	15	5	16	8.0120E-03
1.0E-04	135	2	135	9.9683E-05
1.0E-06	1337	1	1337	1.0001E-06

Table B.2: Example (b): $p = 3$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	12	8	13	5.4500E-03
1.0E-04	104	5	105	7.6247E-05
1.0E-06	1026	6	1026	1.0001E-06

Table B.3: Example (c): $p = 8$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-01	25	3	26	8.5046E-02
1.0E-03	234	417	235	9.7102E-04
1.0E-05	2324	1	2325	9.9872E-06

Table B.4: Example (d): $p = 5$.

The algorithm is summarized as follows.

1. compute t_{N_1} for $C^{(1)} = C$;

2. compute t_{N_2} for $C^{(2)} = 1.0001C$ with $N_2 \leq N_1 + 1$, set $n = 3$;

3. compute $C^{(n)}$ by (B.6);

4. compute t_{N_n} for $C^{(n)}$ with $N_n \leq N_1 + 1$;

5. if (B.7) is not satisfied, set $n = n + 1$ and go to 3;

6. set $t_{N_n} = 1.0$.

B.3 Results

Numerical tests were performed for the same four different functions as those used in Section A.4.1.

(a) $at + (1-a)\left\{1 - \mathrm{e}^{(-t/\epsilon)}\right\} / \left\{1 - \mathrm{e}^{(-1/\epsilon)}\right\}$ with $\epsilon = 0.04$ and $a = 0.6$

(b) $a(t-0.5)^2 - \frac{a}{4}(1 + 8\epsilon t) + (1 + 2\epsilon a)\left\{1 - \mathrm{e}^{(-t/\epsilon)}\right\} / \left\{1 - \mathrm{e}^{(-1/\epsilon)}\right\}$ with $\epsilon = 0.01$ and $a = 3.5$

(c) $\tanh\{20(t-0.5)\}$

(d) $10\mathrm{e}^{(-10t)} + 20/\{1 + 400(t-0.7)^2\}$

(a) is a strictly convex function, (b) and (c) have a single inflection point, and (d) has two inflection points.

All the computations were done with double precision. The value of p was chosen such that the node generation algorithm converges for all the nodes. The results are summarized in Table B.1 to B.4 where $\mathcal{E}_{2(I)}$ is the specified L_2 error, N_1 is the initial number of nodes, N is the final number of nodes, and $\|x-u\|_{L_2(I)}$ is the actual L_2 error computed by the five point Gaussian quadrature formula. Some selected results are shown in Figures B.1 to B.4 where circles indicate nodes, and stars are their projection onto t-axis. In almost all cases, the shooting method converges quickly at less than 10 iterations. For nonconvex functions, this is due to the option that it is allowed to increase the total number of nodes by 1. Without it, it would take extremely long or even fail to converge. However there still can be seen a pathological case. In the last example, the method took very long to converge for $\mathcal{E}_{2(I)}$=1.0E-03. One way to make it converge faster is to change the input value C by a very small

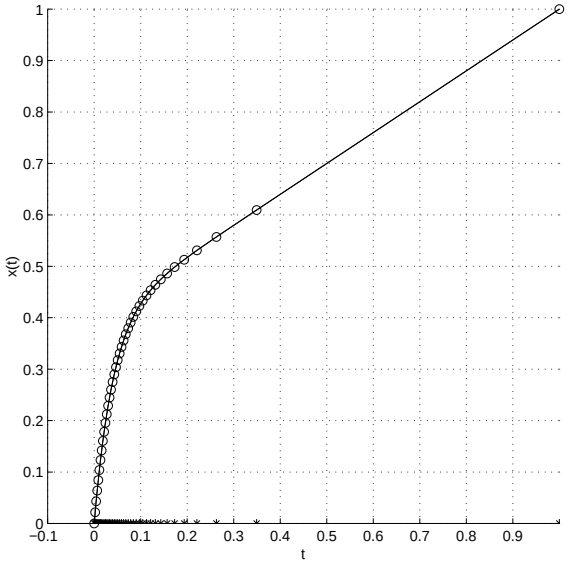

Figure B.1: Example (a). $\mathcal{E}_{2(I)}$ =1.0E-04.

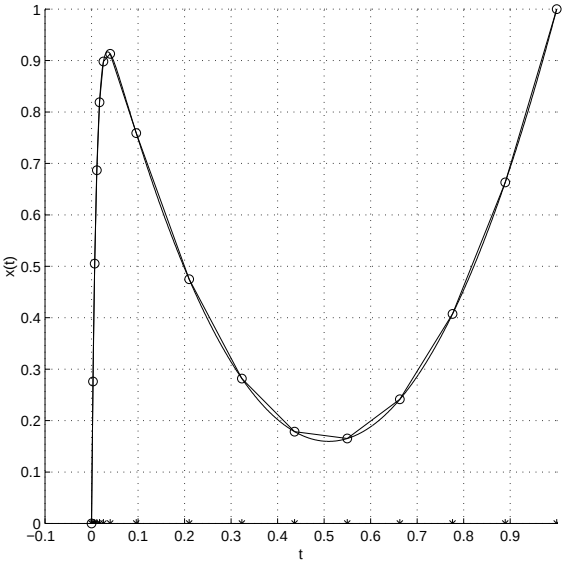

Figure B.2: Example (b). $\mathcal{E}_{2(I)}$ =1.0E-02

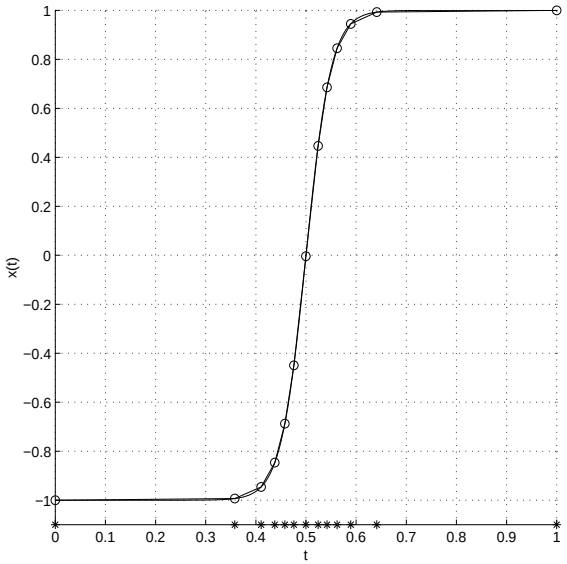

Figure B.3: Example (c). $\mathcal{E}_{2(I)}$ =1.0E-02

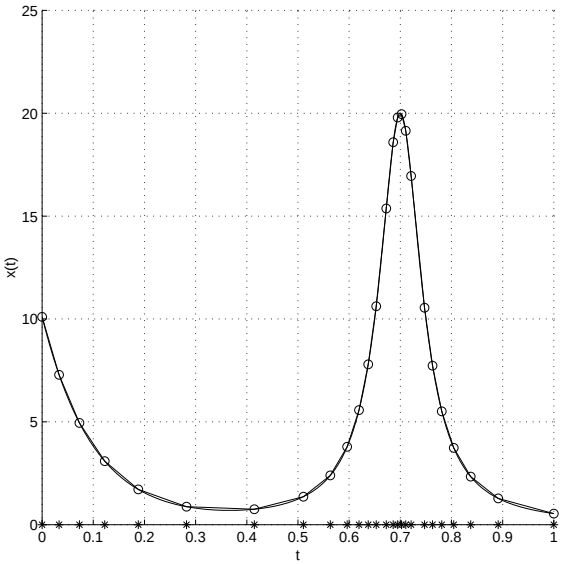

Figure B.4: Example (d). $\mathcal{E}_{2(I)}$ =1.0E-01

amount. We found that $\mathcal{E}_{2(I)}$=1.01E-03 instead of 1.0E-03 made it to converge at the first iteration.

However it should be remembered that there can be in principle two sets bad scenarios in the method. First, $C^{(n)}$ can go negative during the iteration. Second the final number of nodes can be significantly less than the starting value. These problems could arise when the method is applied to functions with a large number of inflection points. In fact we have found that $C^{(n)}$ became negative during the iteration for an oscillatory function such as $x(t) = \mathrm{e}^{-3t}\sin(10t)$. Further study is necessary on this robustness issue.

APPENDIX C

L_2 Error Estimates On Triangular Grids

C.1 L_2 Error for Quadratic Functions

Suppose we have a quadratic function $f(x,y)$ in a domain Ω and its piecewise linear continuous approximation of unknown accuracy, i.e. triangulation of the quadratic function. We are going to compute its local L_2 error on a triangle exactly. Let $\{T\}$ denote the set of triangles that fills Ω, and $T \in \{T\}$ be formed by vertices 1,2 and 3. We define the local L_2 error on triangle T by

$$\left(L_2^T\right)^2 = \int_T \left[f(x,y) - u^T(x,y)\right]^2 dx\, dy\ . \tag{C.1}$$

Here $u^T(x,y)$ is the restriction of the approximation to triangle T which can be written as

$$u^T(x,y) = \overline{u^T} + \frac{\partial u^T}{\partial x}\Delta x_c + \frac{\partial u^T}{\partial y}\Delta y_c \tag{C.2}$$

where

$$\frac{\partial u^T}{\partial x} = \frac{1}{2S^T}\sum_{i=1}^{3} u_i \Delta y_i, \quad \frac{\partial u^T}{\partial y} = -\frac{1}{2S^T}\sum_{i=1}^{3} u_i \Delta x_i, \tag{C.3}$$

u_i are approximations of f at vertices, $\overline{u^T} = (u_1+u_2+u_3)/3$, S^T is the area of T, $x_c = (x_1+x_2+x_3)/3$, $\Delta x_c = x - x_c$, Δx_i is the difference of x taken counterclockwise along the side opposite to j, and similarly for y. Being a quadratic function, $f(x,y)$

can also be written similarly in the form of Taylor series around the centroid of the triangle.

$$f(x,y) = f(x_c,y_c) + p_c\Delta x_c + q_c\Delta y_c + \frac{1}{2}\left[\Delta x_c\frac{\partial}{\partial x} + \Delta y_c\frac{\partial}{\partial y}\right]^2 f \qquad \text{(C.4)}$$

where

$$p_c = \frac{\partial}{\partial x}f(x_c,y_c), \quad q_c = \frac{\partial}{\partial y}f(x_c,y_c). \qquad \text{(C.5)}$$

The difference then becomes

$$f(x,y)-u^T(x,y) = f(x_c,y_c)-\overline{u^T}+\Psi_x^T\Delta x_c+\Psi_y^T\Delta y_c+\frac{1}{2}\left[\Delta x_c\frac{\partial}{\partial x} + \Delta y_c\frac{\partial}{\partial y}\right]^2 f \qquad \text{(C.6)}$$

where

$$\Psi_x^T = p_c - \frac{\partial u^T}{\partial x}, \quad \Psi_y^T = q_c - \frac{\partial u^T}{\partial y}. \qquad \text{(C.7)}$$

The first two terms can be written as

$$f(x_c,y_c) - \overline{u^T} = \overline{e^T} - \frac{1}{3\cdot 2}\left[\Delta x_{ic}\frac{\partial}{\partial x} + \Delta y_{ic}\frac{\partial}{\partial y}\right]^2 f \qquad \text{(C.8)}$$

where $\overline{e^T}$ is the average of the nodal errors defined by $\frac{1}{3}\sum_{i=1}^{3}\{f(x_i,y_i)-u_i\}$, and $\Delta x_{ic} = x_i - x_c$, $\Delta y_{ic} = y_i - y_c$. Therefore, introducing the notation,

$$\begin{aligned} S_c(\Delta x_c,\Delta y_c) &= \Psi_x^T\Delta x_c + \Psi_y^T\Delta y_c \\ Q_c(\Delta x_c,\Delta y_c) &= \frac{1}{2}\left[\Delta x_c\frac{\partial}{\partial x} + \Delta y_c\frac{\partial}{\partial y}\right]^2 f \\ Q_{ic} &= \frac{1}{6}\sum_{i=1}^{3}\left[\Delta x_{ic}\frac{\partial}{\partial x} + \Delta y_{ic}\frac{\partial}{\partial y}\right]^2 f \end{aligned}$$

we have

$$f(x,y) - u^T(x,y) = \overline{e^T} + S_c + Q_c - Q_{ic}. \qquad \text{(C.9)}$$

Now the local L_2 error can be written as

$$\left(L_2^T\right)^2 = \int_T \left[\overline{e^T}^2 + S_c^2 + Q_c^2 + Q_{ic}^2 + 2S_cQ_c - 2Q_cQ_{ic} + 2\overline{e^T}\left(Q_c - Q_{ic}\right) - 2S_cQ_{ic} + 2\overline{e^T}S_c\right] dx\,dy \qquad \text{(C.10)}$$

The integrals of the last two terms identically vanish, and we are left with

$$\left(L_2^T\right)^2 = \int_T \overline{e^T}^2\,dx\,dy + 2\int_T \overline{e^T}\,(Q_c - Q_{ic})\,dx\,dy + \int_T S_c^2\,dx\,dy + \int_T Q_c^2\,dx\,dy + \int_T Q_{ic}^2\,dx\,dy + 2\int_T rS_cQ_c\,dx\,dy - 2\int_T Q_cQ_{ic}\,dx\,dy. \tag{C.11}$$

Integrating, we obtain

$$\begin{aligned}\left(L_2^T\right)^2 = {} & \overline{e^T}^2 S^T - \overline{e^T}\frac{S^T}{9}\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f - \frac{1}{135}S^{T\,2}S_{pq}^T \\ & + \frac{S^T}{36}\sum_{i=1}^{3}\left(\Psi_x^T\Delta x_i + \Psi_y^T\Delta y_i\right)^2 + \frac{S^T}{2160}\left\{\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f\right\}^2 \\ & + \frac{S^T}{324}\left\{\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f\right\}^2 \\ & + \frac{S^T}{30}\sum_{i=1}^{3}\left(\Psi_x^T\Delta x_{ic} + \Psi_y^T\Delta y_{ic}\right)\left(\Delta x_{kc}\frac{\partial}{\partial x} + \Delta y_{kc}\frac{\partial}{\partial y}\right)\left(\Delta x_{lc}\frac{\partial}{\partial x} + \Delta y_{lc}\frac{\partial}{\partial y}\right) f \\ & - \frac{S^T}{648}\left\{\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f\right\}^2\end{aligned}$$

where

$$S_{pq}^T = \frac{1}{2}\sum_{i=1}^{3} p_i\Delta q_i. \tag{C.12}$$

The result can be written as

$$\begin{aligned}\left(L_2^T\right)^2 = {} & \overline{e^T}^2 S^T - \overline{e^T}\frac{S^T}{9}\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f - \frac{1}{135}S^{T\,2}S_{pq}^T \\ & + \frac{S^T}{36}\sum_{i=1}^{3}\left(\Psi_x^T\Delta x_i + \Psi_y^T\Delta y_i\right)^2 + \frac{13}{6480}\left\{\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f\right\}^2 S^T \\ & + \frac{S^T}{30}\sum_{i=1}^{3}\left(\Psi_x^T\Delta x_{ic} + \Psi_y^T\Delta y_{ic}\right)\left(\Delta x_{kc}\frac{\partial}{\partial x} + \Delta y_{kc}\frac{\partial}{\partial y}\right)\left(\Delta x_{lc}\frac{\partial}{\partial x} + \Delta y_{lc}\frac{\partial}{\partial y}\right) f\end{aligned}$$

where the subscripts k and l take 1,2,3, and are permuted cyclically for i. To simplify this further, we rewrite the curvature term as follows.

$$\sum_{i=1}^{3}\left(\Delta x_i\frac{\partial}{\partial x} + \Delta y_i\frac{\partial}{\partial y}\right)^2 f = \sum_{i=1}^{3}\Delta x_i^2\frac{\partial^2 f}{\partial x^2} + 2\sum_{i=1}^{3}\Delta x_i\Delta y_i\frac{\partial^2 f}{\partial x\partial y} + \sum_{i=1}^{3}\Delta y_i^2\frac{\partial^2 f}{\partial y^2}$$

$$
\begin{aligned}
&= \sum_{i=1}^{3} \Delta x_i^2 \frac{\partial p}{\partial x} + \sum_{i=1}^{3} \Delta x_i \Delta y_i \frac{\partial p}{\partial y} + \sum_{i=1}^{3} \Delta x_i \Delta y_i \frac{\partial q}{\partial x} + \sum_{i=1}^{3} \Delta y_i^2 \frac{\partial q}{\partial y} \\
&= \sum_{i=1}^{3} \Delta x_i \left(\frac{\partial p}{\partial x} \Delta x_i + \frac{\partial p}{\partial y} \Delta y_i \right) + \sum_{i=1}^{3} \Delta y_i \left(\frac{\partial q}{\partial x} \Delta x_i + \frac{\partial q}{\partial y} \Delta y_i \right).
\end{aligned}
$$

Now, for quadratic functions the following is an identity.

$$
\Delta p_i = \frac{\partial p}{\partial x} \Delta x_i + \frac{\partial p}{\partial y} \Delta y_i \tag{C.13}
$$

and similarly for q. Hence we have

$$
\sum_{i=1}^{3} \left(\Delta x_i \frac{\partial}{\partial x} + \Delta y_i \frac{\partial}{\partial y} \right)^2 f = \sum_{i=1}^{3} \left(\Delta p_i \Delta x_i + \Delta q_i \Delta y_i \right). \tag{C.14}
$$

Using this exact relation, we obtain

$$
\begin{aligned}
\frac{1}{S^T} \left(L_2^T \right)^2 = \quad & \overline{e^T}^2 - \frac{\overline{e^T}}{9} \sum_{i=1}^{3} \left(\Delta p_i \Delta x_i + \Delta q_i \Delta y_i \right) + \frac{1}{36} \sum_{i=1}^{3} \left(\Psi_x^T \Delta x_i + \Psi_y^T \Delta y_i \right)^2 \\
+ \quad & \frac{1}{90} \sum_{i=1}^{3} \left(\Psi_x^T \Delta x_{ic} + \Psi_y^T \Delta y_{ic} \right) \left\{ \Delta x_{kc} \left(\Delta p_k - \Delta p_i \right) + \Delta y_{kc} \left(\Delta q_k - \Delta q_i \right) \right\} \\
+ \quad & \frac{13}{6480} \left\{ \sum_{i=1}^{3} \left(\Delta p_i \Delta x_i + \Delta q_i \Delta y_i \right) \right\}^2 - \frac{1}{135} S^T S_{pq}^T.
\end{aligned} \tag{C.15}
$$

We first consider the interpolation error, i.e. exact nodal values. In this case, it can be shown that this simplifies to

$$
\left(L_2^T \right)^2 = \frac{C_T^2 S^T}{2160} \tag{C.16}
$$

where

$$
C_T = \sqrt{\sum_{i=1}^{3} K_i^2 + 4 \left\{ \sum_{i=1}^{3} K_i \right\}^2 - 16\, S^T S_{pq}^T}, \tag{C.17}
$$

and

$$
K_i = \Delta p_i \Delta x_i + \Delta q_i \Delta y_i. \tag{C.18}
$$

Adding this over the set of triangles $\{T\}$ and taking square root, we obtain the following L_2 interpolation error

$$
L_2(\Omega) = \sqrt{\sum_{T \in \{T\}} \frac{C_T^2 S^T}{2160}}. \tag{C.19}
$$

This is exact for quadratic functions and second-order accurate for nonquadratic functions.

Next let us consider the case that the nodal values are not exact. In particular, we are interested in the numerical solutions of some partial differential equation. It can be easily deduced from (C.15) that the interpolation error (C.19) estimates the L_2 error with first-order accuracy for third-order accurate nodal values, and second-order for fourth-order accurate nodal values. Even if such high accuracy cannot be attained, the interpolation error is still an important quantity. It represents the error components that are irrelevant to numerical schemes. In other words, it can be controlled only by the grid configuration. But the precise way to use this quantity for grid adaptation remains open.

C.2 Numerical Tests

Results are shown that verify the analysis in the last section. We use the following exponential function.

$$f(x, y) = \exp\{-100(x^2 + y^2)\} \tag{C.20}$$

The first derivatives are computed by a second-order difference approximation with $h = 1.0E - 06$. The domain is a triangle, which contains the peak of the function, and it is systematically refined by connecting the mid points of the sides. The results are shown in Figures C.1 and C.2 where L is the arithmetic average of the sides of the triangle, and the relative error is the absolute value of the difference between the error estimate computed by (C.19) and the actual L_2 error computed by the seven-point Gaussian quadrature. Figure C.1 shows the result for the domain of the equilateral triangle with the centroid placed at the origin and the side length 0.1. Figure C.2 shows the result for the domain of a skew triangle defined by the

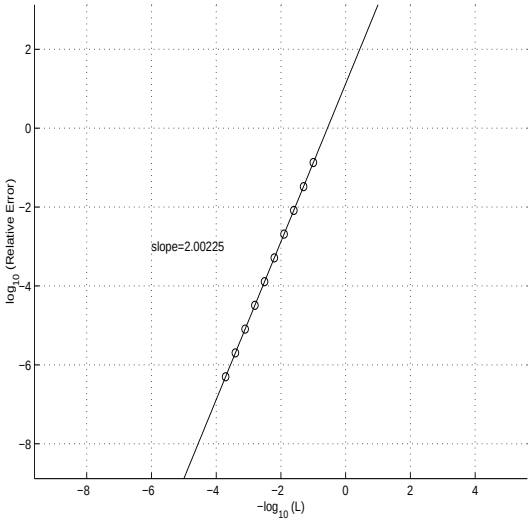

Figure C.1: $-log_{10}(L)$ vs. log_{10} (Relative Error) and the linear least-squares fit. Equilateral Triangles.

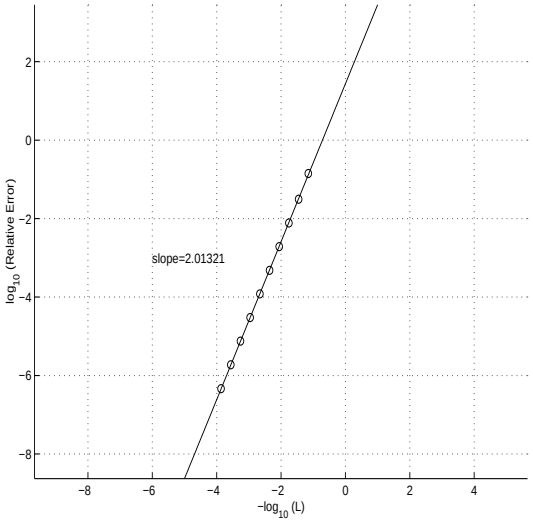

Figure C.2: $-log_{10}(L)$ vs. log_{10} (Relative Error) and the linear least-squares fit. Skew Triangles.

vertices $(0,0)$, $(0.01,0)$ and $(0,0.1)$. In both cases, the second-order convergence can be observed clearly.

APPENDIX D

A Geometrical View in Two Dimensions

We consider a geometrical viewpoint for two-dimensional problems: the least-squares moving grid method is attempting to approximate a geometrical object which can be regarded as the solution of differential equations, as in the one-dimensional problems.

In general, a solution of a set of differential equations, with $(n - p)$ dependent variables and p independent variables, can be regarded as forming a p-dimensional submanifold in an n-dimensional manifold. In the theory of exterior differential forms, a set differential equations are replaced by a set differential forms, and the solution to the set of forms are then understood as a submanifold (or integral manifold) whose tangent vectors annul the forms everywhere on it, in contrast with a solution to differential equations as a function of independent variables.

In particular, we shall consider a numerical approximation to a solution manifold, typically by triangulation in which each triangular element is thought of as forming a two-dimensional tangent space. Approximation errors are obtained on each triangle by requiring a set of differential forms to vanish when they are contracted with approximate tangent vectors defined by two sides of the triangular element. We shall show that the resulting expressions are identical to residuals, thus proving the geometric interpretation of residuals, and that this viewpoint can be useful in inter-

preting the least-squares moving grid method, especially for hyperbolic problems.

D.1 Hyperbolic Equations

D.1.1 Linear Advection

We first consider a linear advection equation

$$au_x + bu_y = 0 \tag{D.1}$$

for which the solution u is a function of x and y. Geometrical statement of the problem is: Find a two-dimensional submanifold in the manifold $\mathbf{R}^3$ with coordinates (u, x, y) that annuls a two-form α defined by

$$\alpha = a\,\tilde{d}u \wedge \tilde{d}y - b\,\tilde{d}u \wedge \tilde{d}x. \tag{D.2}$$

Note that we now treat all the variables as independent. To get back to the original equation, we impose the independence of (x, y) by $\tilde{d}u = u_x\tilde{d}x + u_y\tilde{d}y$, substituting this into the two-form, we obtain

$$\alpha = (au_x + bu_y)\tilde{d}x \wedge \tilde{d}y, \tag{D.3}$$

and requiring the coefficient to vanish leads to (D.1), thus verifying the equivalence to the original problem. We can obtain also other governing equations by choosing other pairs of independent variables: the choice of the independent variables are completely at our disposal. For example, imposing the independence of u and x, by $\tilde{d}y = y_u\tilde{d}u + y_x\tilde{d}x$, substituting this into the two-form, we obtain

$$\alpha = (ay_x - b)\tilde{d}u \wedge \tilde{d}x. \tag{D.4}$$

Vanishing coefficient gives the governing equation for y in u-x plane, $y_x = b/a$, which is just an ordinary differential equation for y with respect to x. This is quite natural for advection problems.

With the two-form, the problem now becomes finding a two-dimensional submanifold, whose tangent vectors annul the two-form. Mathematically stated, the contraction of the tangent vectors with α vanish at every point on the submanifold: $\alpha(U, V) = 0$ where U and V denote tangent vectors that define the solution surface. Note that such tangent vectors form a vector space of dimension two at a point on the integral surface, whose elements are tangent to the surface at that point. A simple solution is a characteristic solution, which can be found by rearranging the two-form as follows.

$$-\tilde{d}u \wedge \tilde{d}(bx - ay) = 0. \tag{D.5}$$

Clearly, the solution submanifold is given by

$$u = const. \text{ and } bx - ay = const. \tag{D.6}$$

These together form lines starting from initial values of u which in turn mesh together to form a solution submanifold of dimension two. Now, we consider numerical approximation of the solution submanifold. We shall construct an approximate submanifold by triangulation defined by a set $\{T\}$ of triangles in which each triangle is considered as approximating the tangent space of dimension two at a nearby point. Consider a typical triangular element $T \in \{T\}$ with vertices 1, 2 and 3. The tangent space being two-dimensional, we may define the two tangent vectors that represent the tangent space by two of the three sides of the triangle,

$$V_1 = (\Delta u_2, \Delta x_2, \Delta y_2)^T \tag{D.7}$$

$$V_2 = (\Delta u_3, \Delta x_3, \Delta y_3)^T \tag{D.8}$$

where $\Delta u_2 = (u_2 - u_1)$, $\Delta u_3 = (u_3 - u_1)$ and similarly for x and y. Contracting these vectors with the differential forms, we get

$$\alpha(V_1, V_2) = a\,\tilde{d}u \wedge \tilde{d}y(V_1, V_2) - b\,\tilde{d}u \wedge \tilde{d}x(V_1, V_2) \tag{D.9}$$

Using the formula $\omega_1 \wedge \omega_2(V_1, V_2) = \frac{1}{2!}\{\omega_1(V_1)\omega_2(V_2) - \omega_1(V_2)\omega_2(V_1)\}$ where ω_1 and ω_2 are one-forms, we obtain, after some rearrangement,

$$\Phi_T = \frac{1}{2} \sum_{i \in j_T} u_i \left(a\Delta y_i - b\Delta x_i\right) \tag{D.10}$$

where $j_T = \{1, 2, 3\}$ and $\Delta(\ )_i$ denotes a difference taken counterclockwise along the side opposite to j. This Φ_T is nothing but the residual which is obtained by integrating the differential equations on a triangular element in $x - y$ plane. But our viewpoint here is that Φ_T represents an error in aligning the tangent vectors along the solution surface: a geometrical interpretation of residual. So, by minimizing this error in a least-squares norm, we are in effect attempting to align the triangular element along the solution surface in $\mathbf{R}^3$, which means that the unknowns are the positions of the vertices in $\mathbf{R}^3$, and therefore it is natural to include x and y as additional unknowns.

Note that

$$\Phi_T = a\,\tilde{d}u \wedge \tilde{d}y(V_1, V_2) - b\,\tilde{d}u \wedge \tilde{d}x(V_1, V_2) = aS_{uy} - bS_{ux} \tag{D.11}$$

where S_{uy} is the area of the triangular element projected onto $u - y$ plane, and analogously S_{ux} is that projected onto $u - x$ plane. This is a fundamental property of differential forms. We will make use of this fact in the rest of the section. It is interesting that by (D.5) we have

$$\Phi_T = -\tilde{d}u \wedge \tilde{d}(bx - ay)(V_1, V_2) = -S_{uc} \tag{D.12}$$

where S_{uc} is the area of the triangle projected onto $u - c$ plane and $c = bx - ay$. Indeed we can write (D.10)

$$\Phi_T = \frac{1}{2} \sum_{i \in j_T} u_i \Delta \left(ay - bx\right)_i \tag{D.13}$$

Then, it is obvious that the residual will vanish if the characteristic relations are satisfied along one of the edges of the triangle, implying collapse of the edge into a point, and thus vanishing area. Therefore, by minimizing this, the least-squares method attempts to minimize the area S_{uc} so that the characteristic relation is satisfied.

D.1.2 Linearized Aerodynamics

We consider

$$\beta^2 u_x - v_y = 0, \qquad v_x - u_y = 0 \tag{D.14}$$

where $\beta^2 = M^2 - 1$ and $M > 1$, the supersonic case. On a manifold $\mathbf{R}^4$ with coordinates (u, v, x, y), we define two-forms

$$\delta = \beta^2 \, \tilde{d}u \wedge \tilde{d}y + \tilde{d}v \wedge \tilde{d}x \tag{D.15}$$

$$\omega = \tilde{d}v \wedge \tilde{d}y + \tilde{d}u \wedge \tilde{d}x. \tag{D.16}$$

The equivalence with the original problem can be shown as before. Imposing the independence of x and y, $\tilde{d}u = u_x \tilde{d}x + u_y \tilde{d}y$ and $\tilde{d}v = v_x \tilde{d}x + v_y \tilde{d}y$,

$$\delta = \beta^2 u_x \, \tilde{d}x \wedge \tilde{d}y + v_y \, \tilde{d}y \wedge \tilde{d}x = (\beta^2 u_x - v_y) \tilde{d}x \wedge \tilde{d}y \tag{D.17}$$

$$\omega = v_x \, \tilde{d}x \wedge \tilde{d}y + u_y \, \tilde{d}y \wedge \tilde{d}x = (v_x - u_y) \tilde{d}x \wedge \tilde{d}y \tag{D.18}$$

and requiring that the coefficients vanish, we obtain the original system (D.14). On the other hand, imposing the independence of u and v, $\tilde{d}x = x_u \tilde{d}u + x_v \tilde{d}v$ and $\tilde{d}y = y_u \tilde{d}u + y_v \tilde{d}v$, we obtain

$$\delta = \beta^2 y_v \, \tilde{d}u \wedge \tilde{d}v + x_u \, \tilde{d}v \wedge \tilde{d}u = (\beta^2 y_v - x_u) \, \tilde{d}u \wedge \tilde{d}v \tag{D.19}$$

$$\omega = y_u \, \tilde{d}v \wedge \tilde{d}u + x_v \, \tilde{d}u \wedge \tilde{d}v = (x_v - y_u) \tilde{d}u \wedge \tilde{d}v. \tag{D.20}$$

Vanishing coefficients implies the hodograph equations

$$\beta^2 y_v - x_u = 0, \qquad x_v - y_u = 0. \tag{D.21}$$

The two-forms are therefore an unified representation of both physical and hodograph equations. In this case also, we can find the characteristic solution. Consider a linear combination of the two-forms

$$\delta + k\omega = \beta^2\, \tilde{d}u \wedge \tilde{d}y + \tilde{d}v \wedge \tilde{d}x + k\, \tilde{d}v \wedge \tilde{d}y + k\, \tilde{d}u \wedge \tilde{d}x. \tag{D.22}$$

We will require that this can be written

$$\delta + k\omega = \tilde{d}(\lambda u + v) \wedge \tilde{d}(x + \lambda y) \tag{D.23}$$

which leads to

$$\lambda^2 = \beta^2, \quad \lambda = k. \tag{D.24}$$

This gives, for $M > 1$, $\lambda = k = \pm\beta$, and therefore we have

$$\delta \pm \beta\omega = \tilde{d}(v \pm \beta u) \wedge \tilde{d}(x \pm \beta y) \tag{D.25}$$

which imply the characteristic representation of the solution submanifold

$$u + \beta v = \text{const. and } x + \beta y = \text{const.} \tag{D.26}$$

$$u - \beta v = \text{const. and } x - \beta y = \text{const..} \tag{D.27}$$

Now, we consider approximating the solution submanifold by a triangulation. As before, we have discrete tangent vectors on each triangular element.

$$V_1 = (\Delta u_2, \Delta v_2, \Delta x_2, \Delta y_2)^T \tag{D.28}$$

$$V_2 = (\Delta u_3, \Delta v_3, \Delta x_3, \Delta y_3)^T \tag{D.29}$$

Contraction with the two-forms α_1 and α_2 results

$$\Delta_T = \lambda(V_1, V_2) = \beta^2 S_{uy} + S_{vx} = \beta^2 \frac{1}{2} \sum_{i \in j_T} u_i \Delta y_i + \frac{1}{2} \sum_{i \in j_T} v_i \Delta x_i \tag{D.30}$$

$$\Omega_T = \omega(V_1, V_2) = \quad S_{vy} + S_{ux} = \quad \frac{1}{2} \sum_{i \in j_T} v_i \Delta y_i + \frac{1}{2} \sum_{i \in j_T} u_i \Delta x_i. \tag{D.31}$$

Again, we have obtained residuals. Consider now instead defining characteristic fluctuations

$$C_T = \tilde{d}(v+\beta u) \wedge \tilde{d}(x+\beta y)(V_1, V_2) \tag{D.32}$$

$$D_T = \tilde{d}(v-\beta u) \wedge \tilde{d}(x-\beta y)(V_1, V_2) \tag{D.33}$$

and minimizing this in a least-squares norm

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{4} \sum_{T\in\{T\}} \left[C_T^2 + D_T^2\right] \tag{D.34}$$

where the factor $1/4$ has been introduced for convenience. This is a norm that recognizes characteristic relations. Hence, for each element, F_T vanishes when a characteristic relation is satisfied along one of its sides, and the other is satisfied along another side. If it is desired to write this using the original fluctuations, one can make the following substitutions, due to (D.25),

$$C_T = (\delta + \beta\omega)(V_1, V_2) = \Delta_T + \beta\Omega_T \tag{D.35}$$

$$D_T = (\delta - \beta\omega)(V_1, V_2) = \Delta_T - \beta\Omega_T. \tag{D.36}$$

Then, the norm becomes

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2} \sum_{T\in\{T\}} \left[\Delta_T^2 + \beta^2\Omega_T^2\right], \tag{D.37}$$

which suggests the choice of weighting matrix Q_T for $\Phi_T = [\Delta_T, \Omega_T]^t$

$$Q_T = \begin{bmatrix} 1 & 0 \\ 0 & \beta^2 \end{bmatrix}. \tag{D.38}$$

This norm does recognizes characteristic relations as discussed in Section 4.1.2.

D.1.3 Burgers' Equation

We consider

$$\partial_y u + u\partial_x u = 0. \tag{D.39}$$

On a manifold $\mathbf{R}^3$ with coordinates (u, x, y), we define a two-form α

$$\alpha = -\tilde{d}u \wedge \tilde{d}x + u\,\tilde{d}u \wedge \tilde{d}y. \tag{D.40}$$

Note that this has a nonconstant coefficient u. We may write

$$\alpha = \tilde{d}u \wedge (u\tilde{d}y - \tilde{d}x) \tag{D.41}$$

from which we find a characteristic relation

$$u = const. \text{ and } \frac{dx}{dy} = u. \tag{D.42}$$

Again, we consider constructing an approximate submanifold by triangulation. The discrete tangent vectors are

$$V_1 = (\Delta u_2, \Delta x_2, \Delta y_2)^T \tag{D.43}$$

$$V_2 = (\Delta u_3, \Delta x_3, \Delta y_3)^T \quad . \tag{D.44}$$

The discrete error is then given by

$$\Phi_T = \alpha(V_1, V_2) = -\tilde{d}u \wedge \tilde{d}x(V_1, V_2) + u\,\tilde{d}u \wedge \tilde{d}y(V_1, V_2) = -\frac{1}{2}\sum_{i \in j_T} u_i \left(\Delta x_i - u \Delta y_i\right). \tag{D.45}$$

Now the question remains as to how to evaluate the nonconstant coefficient u, or where to evaluate the two-form, in other words. The simplest choice would be $\overline{u}_T = (u_1 + u_2 + u_3)/3$ which corresponds to evaluating the two-form at the centroid of the triangular element or the so-called conservative linearization.

$$\Phi_T = -\frac{1}{2}\sum_{i \in j_T} u_i \left(\Delta x_i - \overline{u}_T \Delta y_i\right). \tag{D.46}$$

Obviously, this vanishes when the linearized characteristic relations, (D.42) with $dx/dy = \overline{u}_T$ are satisfied along one of the sides.

It is obvious that there are infinitely many choices as to where to evaluate the form on a triangular element. Then, it would be reasonable to define the error by integration of the two-form over the triangular element, which represents a sort of average error.

$$\Phi_T = \int_T \alpha = \int_T (-\tilde{d}u \wedge \tilde{d}x + u\,\tilde{d}u \wedge \tilde{d}y). \tag{D.47}$$

Note that the integrand can be written as $-\tilde{d}(u\tilde{d}x + (u^2/2)\tilde{d}y)$, and therefore by Stokes' theorem we have

$$\Phi_T = \int_T \alpha = -\oint_{\partial T} u\tilde{d}x + (u^2/2)\tilde{d}y \tag{D.48}$$

in which u may be considered as a function of x and y since $\tilde{d}u$ is absent. This is of course equivalent to the usual definition of the fluctuation. But here it is interpreted as an integral error in aligning the discrete tangent vector to the solution surface represented by the nonconstant two-form.

D.2 Cauchy-Riemann equations

We consider the Cauchy-Riemann system

$$u_x + v_y = 0, \qquad v_x - u_y = 0 \tag{D.49}$$

where u and v denote velocity components. On a manifold $\mathbf{R}^4$ with coordinates (u, v, x, y), we define the two-forms α_1 and α_2 as

$$\alpha_1 = \tilde{d}u \wedge \tilde{d}y - \tilde{d}v \wedge \tilde{d}x \tag{D.50}$$

$$\alpha_2 = \tilde{d}v \wedge \tilde{d}y + \tilde{d}u \wedge \tilde{d}x \quad . \tag{D.51}$$

The equivalence with the original equations may be verified by imposing the independence of x and y. On substituting $\tilde{d}u = u_x\,\tilde{d}x + u_y\,\tilde{d}y$ and $\tilde{d}v = v_x\,\tilde{d}x + v_y\,\tilde{d}y$, we

have

$$\alpha_1 = (u_x + v_y)\,\tilde{d}x \wedge \tilde{d}y \tag{D.52}$$

$$\alpha_2 = (v_x - u_y)\,\tilde{d}x \wedge \tilde{d}y \tag{D.53}$$

and retrieve the original equations by requiring the coefficients to vanish. Note that integrating these forms, assuming the linear variation of u and v, over a triangle in the physical plane (x, y), we obtain the fluctuation as usual, i.e. integrating the differential equations. The hodograph equations are derived by imposing the independence of u and v as before,

$$\alpha_1 = (y_v + x_u)\,\tilde{d}u \wedge \tilde{d}v \tag{D.54}$$

$$\alpha_2 = (-y_u + x_v)\,\tilde{d}u \wedge \tilde{d}v \tag{D.55}$$

Integrating these forms, assuming the linear variation of x and y, over a triangle in the hodograph plane (u, v) yields exactly the same fluctuations as shown in section 123. Here, the fluctuations are obtained, as in the previous sections, from the two-forms which is the unified representation, by applying the two tangent vectors as the two of the three sides of the triangle.

$$V_1 = (\Delta u_2, \Delta v_2, \Delta x_2, \Delta y_2)^T \tag{D.56}$$

$$V_2 = (\Delta u_3, \Delta v_3, \Delta x_3, \Delta y_3)^T \tag{D.57}$$

The results are

$$U_T = \alpha_1(V_1, V_2) = \frac{1}{2}\sum_{i \in j_T} u_i \Delta y_i - \frac{1}{2}\sum_{i \in j_T} v_i \Delta x_i \tag{D.58}$$

$$V_T = \alpha_2(V_1, V_2) = \frac{1}{2}\sum_{i \in j_T} v_i \Delta y_i + \frac{1}{2}\sum_{i \in j_T} u_i \Delta x_i. \tag{D.59}$$

Alternatively, we can define them by integration

$$U_T = \int_T \alpha_1, \quad V_T = \int_T \alpha_2 \tag{D.60}$$

which can be written as line integrals by Stokes' theorem

$$U_T = \oint_{\partial T} u\tilde{dy} - v\tilde{dx}, \quad V_T = \oint_{\partial T} v\tilde{dy} + u\tilde{dx} \tag{D.61}$$

in which u and v may be considered as functions of x and y. And assuming the linear variation, we obtain U_T and V_T.

APPENDIX E

A Least-Squares Norm for the Euler Equations

E.1 Introduction

We consider numerically solving the Euler equations in two dimensions by minimizing a norm of the form

$$\mathcal{F} = \sum_{T \in \{T\}} F_T = \frac{1}{2} \sum_{T \in \{T\}} \mathbf{\Phi}_T^t \mathbf{Q}_T \mathbf{\Phi}_T \tag{E.1}$$

over a set $\{T\}$ of triangles where $\mathbf{\Phi}_T$ is a vector of residuals for the Euler equations and Q_T is a positive definite symmetric matrix that assigns relative weight to the different equations. The most important part of the least-squares formulation is the definition of the norm to be minimized which greatly affects the numerical solutions. In the class of norms defined above, it is the choice of the matrix Q_T that determines the properties of the numerical solutions.

Assuming that we evaluate the residual based on its conservative from, we derive the matrix Q_T that gives a certain number of properties to the minimization scheme. Keep in mind that we retain time dependent terms in the analysis, but our primary concern is steady state solutions.

E.2 A Dimensionally Consistent Set of Variables and Its Equations

We begin with the Euler equations in the conservative form.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial x} = 0 \tag{E.2}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{bmatrix} \tag{E.3}$$

where ρ is the density, u and v are the velocity components in the x and y direction, respectively, and p is the static pressure. The specific energy and enthalpy are given by

$$E = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}(u^2 + v^2) \tag{E.4}$$

$$H = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}(u^2 + v^2). \tag{E.5}$$

One important consideration on defining a least-squares norm is the dimensional consistency. Clearly, the set of equations are not dimensionally consistent in the conservative form. To make it consistent, we introduce a set of variables that is dimensionally consistent.

$$\partial \mathbf{V} = \begin{bmatrix} \partial p \\ \rho q^2 \partial \theta \\ \partial p - a^2 \partial \rho \\ \partial p + \rho q \partial q \end{bmatrix} \tag{E.6}$$

where θ is the flow angle, q is the flow speed, a is the speed of sound, and the third and fourth components represent the entropy and the enthalpy respectively.

Note that the components now have the common physical dimension. To transform the conservative variables into the consistent variables, it is convenient to use the primitive variables.

$$\mathbf{W} = \begin{bmatrix} \rho \\ u \\ v \\ p \end{bmatrix} \tag{E.7}$$

which is linked with the consistent and conservative variables through the transformations

$$\partial \mathbf{V} = \mathbf{T_v} \partial \mathbf{W}, \quad \partial \mathbf{U} = \mathbf{T_u} \partial \mathbf{W} \tag{E.8}$$

respectively where

$$\mathbf{T_v} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & -\rho\, v & \rho\, u & 0 \\ -a^2 & 0 & 0 & 1 \\ 0 & \rho\, u & \rho\, v & 1 \end{bmatrix}, \quad \mathbf{T_u} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ \frac{1}{2} q^2 & \rho\, u & \rho\, v & \frac{1}{\gamma - 1} \end{bmatrix}. \tag{E.9}$$

It follows that

$$\partial \mathbf{V} = \mathbf{T_v} \mathbf{T_u^{-1}} \partial \mathbf{U} \equiv \mathbf{T} \partial \mathbf{U}. \tag{E.10}$$

The transformation matrix T is thus given by

$$\mathbf{T} = \begin{bmatrix} \frac{1}{2}(\gamma - 1) q^2 & -(\gamma - 1) u & -(\gamma - 1) v & \gamma - 1 \\ 0 & -v & u & 0 \\ \frac{1}{2}(\gamma - 1) q^2 - a^2 & -(\gamma - 1) u & -(\gamma - 1) v & \gamma - 1 \\ \frac{1}{2}(\gamma - 3) q^2 & -(\gamma - 2) u & -(\gamma - 2) v & \gamma - 1 \end{bmatrix}. \tag{E.11}$$

Now, we can transform the conservation form into a dimensionally consistent form by multiplying (E.2) by $\mathbf{T}$ from the left.

$$\mathbf{T}\frac{\partial \mathbf{U}}{\partial t} + \mathbf{T}\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial x}\right) = 0 \tag{E.12}$$

$$\mathbf{T}\frac{\partial \mathbf{U}}{\partial t} + \mathbf{T}\left(\mathbf{A}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial x}\right) = 0 \tag{E.13}$$

$$\mathbf{T}\frac{\partial \mathbf{U}}{\partial t} + \mathbf{T}\left(\mathbf{A}\mathbf{T}^{-1}\mathbf{T}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\mathbf{T}^{-1}\mathbf{T}\frac{\partial \mathbf{U}}{\partial x}\right) = 0 \tag{E.14}$$

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{T}\mathbf{A}\mathbf{T}^{-1}\frac{\partial \mathbf{V}}{\partial x} + \mathbf{T}\mathbf{B}\mathbf{T}^{-1}\frac{\partial \mathbf{V}}{\partial x} = 0 \tag{E.15}$$

where $\mathbf{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$, $\mathbf{B} = \frac{\partial \mathbf{G}}{\partial \mathbf{U}}$, and we thus find

$$\mathbf{T}\mathbf{A}\mathbf{T}^{-1} = \begin{bmatrix} u\frac{M^2-1}{M^2} & -\frac{v}{M^2} & 0 & \frac{u}{M^2} \\ -v & u & 0 & 0 \\ 0 & 0 & u & 0 \\ u\frac{M^2-1}{M^2} & -\frac{v}{M^2} & 0 & u\frac{M^2+1}{M^2} \end{bmatrix} \tag{E.16}$$

$$\mathbf{T}\mathbf{B}\mathbf{T}^{-1} = \begin{bmatrix} v\frac{M^2-1}{M^2} & \frac{u}{M^2} & 0 & \frac{v}{M^2} \\ u & v & 0 & 0 \\ 0 & 0 & v & 0 \\ v\frac{M^2-1}{M^2} & \frac{u}{M^2} & 0 & v\frac{M^2+1}{M^2} \end{bmatrix} \tag{E.17}$$

where $M = q/a$ is the Mach number. This is the form of the Euler equations that is dimensionally consistent in terms of the variable $\mathbf{V}$. The equations already being dimensionally consistent, in the least-squares method, we may therefore define the norm as

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \frac{1}{2}\sum_{T\in\{T\}} (\mathbf{T}\mathbf{\Phi}_T)^t\mathbf{T}\mathbf{\Phi}_T = \frac{1}{2}\sum_{T\in\{T\}} \mathbf{\Phi}_T^t\mathbf{T}^t\mathbf{T}\mathbf{\Phi}_T \tag{E.18}$$

which suggests

$$\mathbf{Q}_T = \mathbf{T}^t\mathbf{T}. \tag{E.19}$$

Note that

$$\mathbf{\Phi}_T = -\int_T \frac{\partial \mathbf{U}}{\partial t}\, dx\, dy = \int_T \left[\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial x}\right]\, dx\, dy \tag{E.20}$$

which is evaluated by some quadrature rule. There are formulas that endow the scheme with a certain property such as exact shock capturing as discussed in Section 4.2..

E.3 A Decomposition Matrix

For simplicity, we write the dimensionally consistent Euler equations (E.15) in terms of the natural coordinates: the streamline and its normal.

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{A_v}\frac{\partial \mathbf{V}}{\partial s} + \mathbf{B_v}\frac{\partial \mathbf{V}}{\partial n} = 0 \tag{E.21}$$

where

$$\mathbf{A_v} = \mathbf{TAT}^{-1}\cos\theta + \mathbf{TBT}^{-1}\sin\theta = \begin{bmatrix} q\frac{M^2-1}{M^2} & 0 & 0 & \frac{q}{M^2} \\ 0 & q & 0 & 0 \\ 0 & 0 & q & 0 \\ q\frac{M^2-1}{M^2} & 0 & 0 & q\frac{M^2+1}{M^2} \end{bmatrix} \tag{E.22}$$

$$\mathbf{B_v} = \mathbf{TBT}^{-1}\cos\theta - \mathbf{TAT}^{-1}\sin\theta = \begin{bmatrix} 0 & \frac{q}{M^2} & 0 & 0 \\ q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{q}{M^2} & 0 & 0 \end{bmatrix}. \tag{E.23}$$

Note that the equation for the entropy has been decoupled, but not for the enthalpy. Now, we seek a matrix $\mathbf{P}_d$ that decouples the enthalpy as well by following the technique of preconditioning. We consider an altered system, preconditioned by $\mathbf{P}_d$.

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{P}_d\left(\mathbf{A_v}\frac{\partial \mathbf{V}}{\partial s} + \mathbf{B_v}\frac{\partial \mathbf{V}}{\partial n}\right) = 0 \tag{E.24}$$

and require that

$$\mathbf{P}_d\mathbf{A}_\mathbf{v} = \begin{bmatrix} q(M^2-1) & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & q \end{bmatrix}, \quad \mathbf{P}_d\mathbf{B}_\mathbf{v} = \begin{bmatrix} 0 & q & 0 & 0 \\ q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{E.25}$$

which imply that we have two advection equations for the entropy and the enthalpy and a 2×2 acoustic subsystem in the resulting system. The matrix $\mathbf{P}_d$ can be determined from the first equation. The result is

$$\mathbf{P}_d = \begin{bmatrix} M^2+1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \tag{E.26}$$

and an *a posteriori* check confirms that it satisfies the other requirement for $\mathbf{B}_\mathbf{v}$. Therefore, the system (E.24) becomes

$$\frac{\partial p}{\partial t} + q(M^2-1)\frac{\partial p}{\partial s} + \rho q^3 \frac{\partial \theta}{\partial n} = 0 \tag{E.27}$$

$$\rho q^2 \frac{\partial \theta}{\partial t} + \rho q^3 \frac{\partial \theta}{\partial s} + q\frac{\partial p}{\partial n} = 0 \tag{E.28}$$

$$\frac{\partial S}{\partial t} + q\frac{\partial S}{\partial s} = 0 \tag{E.29}$$

$$\frac{\partial h}{\partial t} + q\frac{\partial h}{\partial s} = 0 \tag{E.30}$$

where the first two compose the 2×2 acoustic subsystem and the others are completely decoupled advection equations for the entropy $\partial S = \partial p - a^2 \partial \rho$ and the enthalpy $\partial h = \partial p + \rho q \partial q$. In the least-squares method, we may therefore multiply the residual by the matrix $\mathbf{P}_d$ to decompose the residual, and then minimize the

decomposed residual in the least-squares sense.

$$\mathcal{F} = \frac{1}{2} \sum_{T \in \{T\}} (\mathbf{P}_d \mathbf{T} \mathbf{\Phi}_T)^t \mathbf{P}_d \mathbf{T} \mathbf{\Phi}_T = \frac{1}{2} \sum_{T \in \{T\}} \mathbf{\Phi}_T^t \mathbf{T}^t \mathbf{P}_d^t \mathbf{P}_d \mathbf{T} \mathbf{\Phi}_T \tag{E.31}$$

which suggests

$$\mathbf{Q}_T = \mathbf{T}^t \mathbf{P}_d^t \mathbf{P}_d \mathbf{T}. \tag{E.32}$$

E.4 A Matrix for the Acoustic Subsystem

We consider the subsystem in its steady state form.

$$q(M^2 - 1)\frac{\partial p}{\partial s} + \rho q^3 \frac{\partial \theta}{\partial n} = 0 \tag{E.33}$$

$$\rho q^3 \frac{\partial \theta}{\partial s} + q\frac{\partial p}{\partial n} = 0 \tag{E.34}$$

The system is then hyperbolic in supersonic flow ($M > 1$) and elliptic in subsonic flow ($M < 1$). Let us write the system in the matrix form.

$$\mathbf{A}_s \frac{\partial \mathbf{V}_s}{\partial s} + \mathbf{B}_s \frac{\partial \mathbf{V}_s}{\partial n} = 0 \tag{E.35}$$

where $\mathbf{V}_s = [p, \theta]^t$ and

$$\mathbf{A}_s = \begin{bmatrix} q(M^2-1) & 0 \\ 0 & \rho q^3 \end{bmatrix}, \quad \mathbf{B}_s = \begin{bmatrix} 0 & \rho q^3 \\ q & 0 \end{bmatrix}. \tag{E.36}$$

In the hyperbolic case, we can diagonalize the system as follows. Multiplying $\mathbf{A}_s^{-1}$ from the left, we have

$$\frac{\partial \mathbf{V}_s}{\partial s} + \mathbf{A}_s^{-1}\mathbf{B}_s \frac{\partial \mathbf{V}_s}{\partial n} = 0. \tag{E.37}$$

The matrix $\mathbf{A}_s^{-1}\mathbf{B}_s$ has the eigenvalues $\pm 1/\beta$ where $\beta = \sqrt{M^2 - 1}$ and the eigenvectors which are arranged into the columns of a matrix $\mathbf{R}_s$

$$\mathbf{R}_s = \begin{bmatrix} -\frac{1}{q\beta^2} & \frac{1}{q\beta^2} \\ \frac{1}{\rho q^3 \beta} & \frac{1}{\rho q^3 \beta} \end{bmatrix}. \tag{E.38}$$

Then, the characteristic form is obtained by multiplying the system by $\mathbf{R}_s^{-1}$.

$$\mathbf{R}_s^{-1}\frac{\partial \mathbf{V}_s}{\partial s} + \mathbf{R}_s^{-1}\mathbf{A}_s^{-1}\mathbf{B}_s\mathbf{R}_s\mathbf{R}_s^{-1}\frac{\partial \mathbf{V}_s}{\partial n} = 0 \tag{E.39}$$

$$\frac{\partial \mathbf{W}_s}{\partial s} + \mathbf{\Lambda}\frac{\partial \mathbf{W}_s}{\partial n} = 0 \tag{E.40}$$

where $\partial \mathbf{W}_s = \mathbf{R}_s^{-1}\partial \mathbf{V}_s$ is the characteristic variables and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the eigenvalues. Now, note that we obtained the characteristic system by multiplying the original system (E.34) by $\mathbf{A}_s^{-1}$ and then $\mathbf{R}_s^{-1}$, or the matrix $\mathbf{P}_s \equiv \mathbf{R}_s^{-1}\mathbf{A}_s^{-1}$ from the left. Then, denoting the residual corresponding to the original system by $\mathbf{\Psi}_T$, we obtain the residual for the characteristic system as

$$\mathbf{P}_s\mathbf{\Psi}_T. \tag{E.41}$$

We then minimize this characteristic residual in the lease-squares norm.

$$\mathcal{F}_s = \sum_{T\in\{T\}} (\mathbf{P}_s\mathbf{\Psi}_T)^t\mathbf{P}_s\mathbf{\Psi}_T = \sum_{T\in\{T\}} \mathbf{\Psi}_T^t\mathbf{P}_s^t\mathbf{P}_s\mathbf{\Psi}_T. \tag{E.42}$$

where

$$\mathbf{P}_s = \frac{1}{2}\begin{bmatrix} -1 & \beta \\ 1 & \beta \end{bmatrix}, \quad \mathbf{D}_s \equiv \mathbf{P}_s^t\mathbf{P}_s = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & \beta^2 \end{bmatrix}. \tag{E.43}$$

With this weighting matrix, the minimization scheme recognizes the characteristic equations: the norm vanishes if the characteristic equations are satisfied for simple as well as non-simple waves just as in the linear case. To deal with subsonic cases, we define, following the analysis of Roe[82],

$$\mathbf{P}_s = \frac{1}{2}\begin{bmatrix} -1 & \sqrt{|\beta^2|} \\ 1 & \sqrt{|\beta^2|} \end{bmatrix}, \quad \mathbf{D}_s \equiv \mathbf{P}_s^t\mathbf{P}_s = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & |\beta^2| \end{bmatrix}. \tag{E.44}$$

Then, in subsonic case, with area weighting, the discretization becomes identical to the standard finite element discretization for the equivalent second-order partial differential equations.

E.5 The Norm for the Euler Equations

We now put everything together. Assume that we have evaluated the residual from the conservative form, $\mathbf{\Phi}_T$. First, for dimensional consistency, we multiply the residual by $\mathbf{T}$.

$$\mathbf{T}\mathbf{\Phi}_T \tag{E.45}$$

Second, to decompose the system, we multiply this by $\mathbf{T}$.

$$\mathbf{P}_d\mathbf{T}\mathbf{\Phi}_T \tag{E.46}$$

Finally, for the acoustic system, we multiply this by $\mathbf{P}_s$.

$$\mathbf{P}_s\mathbf{P}_d\mathbf{T}\mathbf{\Phi}_T \tag{E.47}$$

where $\mathbf{P}_s$ is now 4×4 matrix defined by

$$\mathbf{P}_s = \begin{bmatrix} -1/2 & \sqrt{|\beta^2|}/2 & 0 & 0 \\ 1/2 & \sqrt{|\beta^2|}/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{E.48}$$

We then minimize this residual in the least-squares sense, i.e. minimize

$$\mathcal{F} = \sum_{T\in\{T\}} (\mathbf{P}_s\mathbf{P}_d\mathbf{T}\mathbf{\Phi}_T)^t\mathbf{P}_s\mathbf{P}_d\mathbf{T}\mathbf{\Phi}_T \tag{E.49}$$

or

$$\mathcal{F} = \sum_{T\in\{T\}} \mathbf{\Phi}_T^t\mathbf{Q}_T\mathbf{\Phi}_T \tag{E.50}$$

where

$$\mathbf{Q}_T = \mathbf{T}^t\mathbf{P}_d^t\mathbf{P}_s^t\mathbf{P}_s\mathbf{P}_d\mathbf{T}. \tag{E.51}$$

E.6 Implementation

Since the matrix $\mathbf{Q}_T$ is complicated, it would be simpler to implement the weight by computing the transformed residual vector.

$$\mathbf{\Phi}_T^p = \mathbf{P}\mathbf{\Phi}_T = \mathbf{P}_s\mathbf{P}_d\mathbf{T}\mathbf{\Phi}_T \tag{E.52}$$

where

$$\mathbf{P} = \begin{bmatrix} -\frac{q^2}{4}(1+R) & \frac{uR}{2} - \frac{v}{2}\sqrt{|\beta^2|} & \frac{vR}{2} + \frac{u}{2}\sqrt{|\beta^2|} & -\frac{\gamma-1}{2}M^2 \\ \frac{q^2}{4}(1+R) & -\frac{uR}{2} - \frac{v}{2}\sqrt{|\beta^2|} & -\frac{vR}{2} + \frac{u}{2}\sqrt{|\beta^2|} & \frac{\gamma-1}{2}M^2 \\ \frac{\gamma-1}{2}q^2 - a^2 & -(\gamma-1)u & -(\gamma-1)v & \gamma-1 \\ -q^2 & u & v & 0 \end{bmatrix} \tag{E.53}$$

where $R = 1 + (\gamma-1)M^2$. This matrix is evaluated within each triangle by the value in accordance with the linearization of the conservation form. Then we minimize

$$\mathcal{F} = \sum_{T\in\{T\}} F_T = \sum_{T\in\{T\}} (\mathbf{\Phi}_T^p)^t \mathbf{\Phi}_T^p \tag{E.54}$$

by a steepest descent method

$$\delta\mathbf{Z}_j = -\omega\frac{\partial\mathcal{F}}{\partial\mathbf{Z}_j} = -\omega\sum_{T\in\{T_j\}}\frac{\partial F_T}{\partial\mathbf{Z}_j} \tag{E.55}$$

where ω is a small constant and $\mathbf{Z}_j$ is a set of variables with respect to which the norm is minimized. The gradient in the cell T is given by

$$\frac{\partial F_T}{\partial\mathbf{Z}_j} = (\mathbf{\Phi}_T^p)^t\frac{\partial\mathbf{\Phi}_T^p}{\partial\mathbf{Z}_j} = (\mathbf{\Phi}_T^p)^t\mathbf{P}\frac{\partial\mathbf{\Phi}_T}{\partial\mathbf{Z}_j}. \tag{E.56}$$

E.7 Summary

The norm derived here has the following properties.

1. It is dimensionally consistent.

2. It is equivalent to a least-squares norm for the decomposed Euler equations.

3. In supersonic flows, it is equivalent to a least-squares norm for the characteristic equations, and thus the norm can vanish for simple or non-simple wave solutions.

4. In subsonic case, with area weighting, the method becomes identical to the standard finite element discretization for the second-order partial differential equations equivalent to the elliptic subsystem.

Shock capturing capability is given by a particular linearization of the conservative form via a parameter α as discussed in Section 4.2.

APPENDIX F

Computing the Exact Solutions for Airfoil Problems

Our concern here is to compute the exact solutions (the velocity components) of the incompressible irrotational flow around a Joukowsky airfoil *at a specified location in the space where the airfoil resides.* A conventional method to obtain the exact solution to the flow around an airfoil is based on a *direct* transformation: from a flow around a cylinder onto a flow around an airfoil. This means that we choose a point in the plane of the cylinder, compute the solution there, and transform the location as well as the solution onto a point in the plane of the airfoil to obtain the exact solution. In this manner, however, it is difficult to obtain the solution at a desired location in the airfoil plane which is exactly what we wish to do in order to obtain the solution on a given grid point for the purpose of computing the numerical error. To achieve this, we need to reverse the transformation, which, however, must be performed carefully.

We begin by outlining the direct procedure. Let $\zeta = \xi + i\eta$ and $Z = x + iy$ denote the complex planes of the cylinder and the airfoil respectively. Consider the Joukowsky transformation

$$Z = \zeta + \frac{l^2}{\zeta} \tag{F.1}$$

by which the circle $|\zeta| = l$ in ζ-plane is transformed onto the line segment joining the points $Z = \pm 2l$ in Z-plane and the exterior of the circle is transformed onto the exterior of the line segment. To generate an airfoil, we choose a circle that passes through the point $\zeta = l$, which corresponds to the trailing edge, but whose center is shifted by a small amount $\mu = -\epsilon l + i\kappa l$ where ϵ and κ are small positive constants that control the thickness and camber of the resulting airfoil respectively. The circle is thus defined by

$$\zeta_c = \mu + ae^{i\theta} \tag{F.2}$$

where

$$a = l\sqrt{(1+\epsilon)^2 + \kappa^2}, \tag{F.3}$$

and $0 \leq \theta < 2\pi$. We define an angle, β, by

$$\zeta_{TE} = l = \mu + ae^{-i\beta}. \tag{F.4}$$

It is easy to show that this angle is computable by

$$\beta = \sin^{-1}\left(\frac{\kappa}{a}\right). \tag{F.5}$$

The generating circle is now completely defined and the airfoil can be obtained simply by mapping the points on the circle by (F.1). The exact solution is available for a flow around a cylinder defined by (F.2) in the form of the complex velocity

$$\frac{W(\zeta)}{V_\infty} = \frac{d(F(\zeta)/V_\infty)}{d\zeta} = \frac{u - i\,v}{V_\infty} = e^{-i\alpha} + \frac{2ai\,\sin(\alpha+\beta)}{\zeta - \mu} - \frac{a^2 e^{i\alpha}}{(\zeta-\mu)^2} \tag{F.6}$$

where $F(\zeta)$ denotes the corresponding complex potential, an angle of attack is denoted by α, and the circulation has been determined by the Kutta condition $W(\zeta_{TE}) = 0$. To obtain the complex velocity in Z-plane, we use the chain rule.

$$\frac{W(Z)}{V_\infty} = \frac{d(F(\zeta(Z))/V_\infty)}{dZ} = \frac{d(F(\zeta)/V_\infty)}{d\zeta}\frac{d\zeta}{dZ} = \frac{d(F(\zeta)/V_\infty)}{d\zeta}\left(\frac{dZ}{d\zeta}\right)^{-1}. \tag{F.7}$$

Note that the derivative $dZ/d\zeta$ becomes zero at ζ_{TE} and therefore $W(\zeta_{TE}) = 0$ is required to obtain a finite velocity at the trailing edge. The velocity there can be found by using L'Hospital's rule. The result is

$$\frac{W(Z_{TE})}{V_\infty} = \frac{l}{a}\, e^{2i\beta}\cos(\alpha+\beta). \tag{F.8}$$

Now, everything is set up and we can do the following: pick up a point in ζ-plane; transform it into a point in Z-plane by (F.1); compute the velocity at that point by (F.7) or (F.8). It is important to note that the velocity is a function of ζ alone and therefore can be computed without knowing the corresponding point in Z-plane. Therefore, to compute the velocity at a given point in Z-plane which is what we want to do, we just need to find the corresponding point in ζ-plane, thus leading to the inverse transformation. Consequently, the discussion that follows will be purely geometrical.

In order to invert the transformation (F.1), we first write it in an alternative form. Note that adding $\pm 2l$ to the both sides of (F.1) gives

$$Z \pm 2l = \frac{\zeta^2 + l^2 \pm 2l\zeta}{\zeta} = \frac{(\zeta \pm l)^2}{\zeta}, \tag{F.9}$$

and division yields

$$\frac{Z-2l}{Z+2l} = \left(\frac{\zeta - l}{\zeta + l}\right)^2 \tag{F.10}$$

which is the desired result. It then follows immediately that

$$\arg\left(\frac{Z-2l}{Z+2l}\right) = 2\,\arg\left(\frac{\zeta-l}{\zeta+l}\right). \tag{F.11}$$

This shows that the angle formed by two line segments, the one joining the point of interest and $Z = -2l$ and the other joining the point and $Z = 2l$ is always twice the angle formed by the segment joining the corresponding point and $\zeta = -l$ and the one joining that point and $\zeta = l$. It follows from this that the Joukowsky transformation

maps the circle $|\zeta| = l$ in ζ-plane onto the line segment joining the points $Z = \pm 2l$ in Z-plane and the exterior of the circle onto the exterior of the line segment. The map is bijective (one-to-one and onto), and therefore the inverse transformation exists. It is obtained from

$$\left(\frac{Z-2l}{Z+2l}\right)^{\frac{1}{2}} = \frac{\zeta - l}{\zeta + l}, \tag{F.12}$$

and solving for ζ gives

$$\zeta = \frac{(Z+2l)^{\frac{1}{2}} + (Z-2l)^{\frac{1}{2}}}{(Z+2l)^{\frac{1}{2}} - (Z-2l)^{\frac{1}{2}}}. \tag{F.13}$$

It is now possible to compute the exact solution at a given point in the airfoil plane: pick any point Z outside the airfoil; transform back to the corresponding point in ζ-plane by (F.13); compute the velocity by (F.7) or (F.8).

However, the inverse transformation fails for cambered airfoils. The reason is that the transformation is not injective (one-to-one) if the inside of the circle $|\zeta| = l$ is included in its domain. In fact, it is easy to show from (F.11) that the lower-half region of the circle is mapped onto the upper-half of Z-plane while the upper-half region of the circle is mapped onto the lower-half of Z-plane. The cambered airfoils are generated by a circle part of which goes inside of the core circle $|\zeta| = l$, due to nonzero κ, typically near but less than $\theta = -\beta$. Therefore the region inside of the core circle, which is below the real axis, is now transformed a region that is above the real axis in Z-plane, thus creating a camber. It can be shown that this happens if a point $\zeta = re^{i\theta}$ lies in the region defined by

$$2\pi - (2\sin^{-1}(l\kappa/|\mu|) - \beta) < \theta < 2\pi - \beta \text{ and } |\zeta_c| < r < l. \tag{F.14}$$

This shows conversely that it never happen only if $\kappa = 0$, i.e. symmetrical airfoils. If κ is nonzero and the point in consideration happens to be a map from a point inside of the core circle, the inverse transformation (F.13) fails, transforming the point into

the one outside of the circle. A correct transformation is obtained by putting the minus sign on the left side of in (F.12), and solving for ζ, thus giving

$$\zeta = \frac{(Z+2l)^{\frac{1}{2}} - (Z-2l)^{\frac{1}{2}}}{(Z+2l)^{\frac{1}{2}} + (Z-2l)^{\frac{1}{2}}}. \tag{F.15}$$

It is however not easy in practice to tell if the point in Z-plane belongs to this domain to switch the inverse transformation. A more practical way to overcome this difficulty is to employ von Kármán-Trefftz airfoils generated by

$$\frac{Z-nl}{Z+nl} = \left(\frac{\zeta - l}{\zeta + l}\right)^n \tag{F.16}$$

where

$$n = 2 - \frac{\tau}{\pi}. \tag{F.17}$$

This is a generalization of the Joukowsky transformation which replaces the cusp at the trailing edge by a point with two distinct tangents creating a finite interior angle of the trailing edge that is specified by τ. It can be shown that the von Kármán-Trefftz transformation is bijective in the domain that includes a bit of inside of the core circle. Therefore the inverse transformation, (F.13) with 2 replaced by n, exists even for cambered airfoils although the camber must be small.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Ariffin. *Grid Equidistribution via Various Algorithmic Approaches.* Master's Thesis, Department of mathematics at the University of Reading, September 1999.

[2] G. Ashford. An Unstructured Grid Generation and Adaptive Solution Technique For High-Reynolds-Number Compressible Flows. PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1996.

[3] M. J. Baines. *Moving Finite Elements.* Oxford Science Publications, 1994.

[4] M. J. Baines. Properties of a Grid Movement Algorithm. Numerical Analysis Report 8/95, The University of Reading, Department of Mathematics. 1995.

[5] M. J. Baines. Algorithms for Optimal Discontinuous Piecewise Linear and Constant L_2 Fits to Continuous Functions with Adjustable Nodes in One and Two Dimensions. *Mathematics of Computation*, 62:645-669, 1994.

[6] M. J. Baines. Least Squares and Approximate Equidistribution in Multidimensions. *Numerical Methods for Partial Differential Equations*, 15:605-615, 1999.

[7] M. J. Baines and S. J. Leary. Fluctuation Distribution Schemes on Adjustable Meshes for Scalar Hyperbolic Equations. Numerical Analysis Report 6/98, The University of Reading, Department of Mathematics. 1998.

[8] M. J. Baines, S. J. Leary, M. E. Hubbard. A Finite Volume Method for Steady Hyperbolic Equations. *Finite Volumes for Complex Applications II, problems and perspectives, 2nd International Symposium at Duisburg, Germany, July 19–22, 1999*, ed. by R. Vilsmeier, F. Benkhaldoun, D. Hanel (Hermes Science Publications, Paris 1999) pp. 787–794.

[9] T. J. Barth. Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations. *Special Course on Unstructured Grid Methods for Advection Dominated Flows.* AGARD Report 787. pp. (6-1) – (6-61).

[10] J. T. Batina. Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes. *AIAA Journal*, 28:1381-1388, 1990.

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

[12] A. Borzi, K. W. Morton, E. Suli, and M. Vanmaele. Multilevel Solution of Cell-Vertex Cauchy-Riemann Equations. *SIAM Journal on Scientific Computing*, 18(2):441-459, 1997.

[13] R. L. Burden and J. D. Faires. *Numerical Analysis*. PWS Publishing Company, 1992.

[14] F. Blom. Considerations on the Spring Analogy. *International Journal for Numerical Methods in Fluids*, 32:647-668, 2000.

[15] J. U. Brackbill and J. S. Saltzman. Adaptive Zoning for Singular Problems in Two Dimensions . *Journal of Computational Physics*, 46:342-368, 1982.

[16] J. U. Brackbill. An Adaptive Grid with Directional Control. *Journal of Computational Physics*, 108:38-50, 1993.

[17] P. B. Bochev and M. D. Gunzburger. Finite Element Methods of Least-Squares Type. *SIAM Reviews*, 40(4):789-837, 1998.

[18] Z. Cai, R. D. Lazarov, T. Manteuffel, and S. McCormick. First-Order System Least-Squares for Partial Differential Equations; Part 1. *SIAM Journal on Numerical Analysis*, 31:1785-1799, 1994.

[19] D. Caraeni, L. Fuchs. A New Compact High Order Multidimensional Upwind Discretization. *Proceedings of the 4th World CSCC Conference*, Vouliagmeni, Greece, July 10-15 2000.

[20] G. F. Carey. *Computational Grids*. Taylor & Francis Publishers, 1997.

[21] G. F. Carey, A. I. Pehlivav, Y. Shen, A. Bose, and K. C. Wang. Least-Squares Finite Elements for Fluid Flow and Transport. *International Journal for Numerical Methods in Fluids*, 27:97-107, 1998.

[22] N. N. Carlson and K. Miller. Design and Application of a Gradient-Weighted Moving Finite Element Code II: In Two Dimensions. *SIAM Journal on Scientific Computing*, 19(3):766-798, 1998.

[23] M. J. Castro-Diaz, F. Hechet, B. Mohammadi, and O. Pironneau. Anisotropic Unstructured Mesh Adaptation for Flow Simulations. *International Journal For Numerical Methods in Fluids*, 25:475-491, 1997.

[24] C-L. Chang and M. D. Bunzburger. A subdomain-Galerkin/least-squares method for first-order elliptic systems in the plane. *SIAM Journal on Numerical Analysis*, 27(5):1197-1211, 1990.

[25] M. J. Cloud and B. C. Drachman. *Inequalities with Applications to Engineering*. Springer-Verlag New York, 1998.

[26] W. Coirier. An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations. PhD thesis, The University of Michigan, Department of Aerospace Engineering. 1994.

[27] P. I. Crumpton, J. A. Mackenzie, and K. W. Morton. Cell Vertex Algorithms for the Compressible Navier-Stokes Equations.*Journal of Computational Physics*, 109:01-15, 1993.

[28] D. DeZeeuw. A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm For Solution of the Euler Equations PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1993.

[29] E. F. D'Azevedo. Are Bilinear Quadrilaterals better than Linear Triangles?. *SIAM Journal on Scientific Computing*, 22(1):198-217, 2000.

[30] H. Deconinck, P. L. Roe, and R. J. Struijs. A Multidimensional Generalisation of Roe's Flux Difference Splitter for the Euler Equations. *Computers and Fluids*, 22(2/3):215-222, 1993.

[31] H. Deconinck and G. Degrez. Multidimensional Upwind Residual Distribution Scheme and Application. *Finite Volumes for Complex Applications II, problems and perspectives, 2nd International Symposium at Duisburg, Germany, July 19–22, 1999*, ed. by R. Vilsmeier, F. Benkhaldoun, D. Hanel (Hermes Science Publications, Paris 1999) pp. 27-40.

[32] J. R. Dormand. *Numerical Methods for Differential Equations : A Computational Approach*. CRC Press, LLC, 1996.

[33] H. A. Dwyer, R. J. Kee, and B. R. Sandeers. Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer. *AIAA Journal*, 18:1205-1212, 1980.

[34] E. M. Edwards. *Advanced Calculus: A Differential Forms Approach*. Birkhäuser, 1997.

[35] A. J. Engel. Directional Control in Grid Generation. *Journal of Computational Physics*, 74:422-439, 1988.

[36] K. Eriksson, D. Step, P. Hansbo, and C. Jofnson. *Computational Differential Equations*. Cambridge University Press, 1996.

[37] A. Ferrante. Solution of the Unsteady Euler Equations using Residual Distribution and Flux Corrected Transport. VKI PR-1997-08, von Karman Institute for Fluid Dynamics, 1997.

[38] G. J. Fix and M. E. Rose. A Comparative Study of Finite Element and Finite Difference Methods for Cauchy-Riemann Type Equations. *SIAM Journal on Numerical Analysis*, 22(2):250-261, 1985.

[39] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, 1981.

[40] P. A. Gnoffo. A Finite-Volume, Adaptive Grid Algorithm applied to Planetary Entry Flow Fields. *AIAA Journal*, 21:1249-1254, 1983.

[41] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin and M. Vallet. Anisotropic Mesh Adaptation: Towards User-Independent, Mesh-Independent and Solver-Independent CFD. Part I: General Principles. *International Journal For Numerical Methods in Fluids*, 32:725-744, 2000.

[42] L. Hasdorff. *Gradient Optimization and Nonlinear Control*. Wiley-Interscience, 1976.

[43] 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. *Journal of Computational Physics*, 95:254-302, 1991.

[44] D. G. Holmes and D. D. Snyder. The Generation of Unstructured Triangular Meshes using Delaunay Triangulation. *Numerical Grid Generation in Computational Fluid Mechanics '88*, ed. by S. Sengupta et. al. (Pineridge Press, 1988) pp 643-652.

[45] M. E. Hubbard and P. L. Roe. Compact High-Resolution Algorithms for Time-Dependent Advection on Unstructured Grids. *Journal of Computational Physics*, 33:711-736, 2000.

[46] C. Ilinca, X. D. Zhang, J.-Y. Trepanier, and R. Camarero. A Comparison of Three Error Estimation Techniques for Finite-Volume Solutions of Compressible Flows. *Computer Methods in Applied Mechanics and Engineering*, 189:1277-1294, 2000.

[47] A. Jameson. Aerodynamic Design via Control Theory. *Journal of Scientific Computing*, 3:233-260, 1998.

[48] B. Jiang. *The Least-Squares Finite Element Method: Theory and Applications in Computational Fluid Dynamics and Electromagnetics*. Springer-Verlag New York, 1998.

[49] O. B. Khairullina. Method of Constructing Block Regular Optimal Grids in Two-Dimensional Multiply Connected Domains of Complex Geometries. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 11(4):343-358, 1996.

[50] L. Kania. Elliptic Adaptive Grid Generation and Area Equidistribution. *International Journal for Numerical Methods in Fluids*, 30:481-491, 1999.

[51] H. B. Keller. *Numerical Methods for Two-Point Boundary-Value Problems.* Dover, 1992.

[52] R. Ketzscher and S. Forth. A Shooting Method for the Generation of the best L_1 piecewise linear interpolation. AMOR 99/3, Cranfield University.

[53] P. Knupp and S. Steinberg. *Fundamentals of Grid Generation.* CRC Press Inc., 1993.

[54] J. D. Lambert. *Numerical Methods for Ordinary differential Equations; The Initial Value Problems.* John Wiely & Suns, Chichester, England 1991.

[55] S. J. Leary. Numerical Experiments on the Solution of a Scalar Advection Equation by Least Squares on Optimal Grids. Numerical Analysis Report 9/97, The University of Reading, Department of Mathematics. 1997.

[56] R. J. LeVeque. *Numerical Methods for Conservation Laws.* Birkhäuser, 1992.

[57] M. W. Lipschutz. *Differential Geometry*, (Schaum's Outline Series). McGraw-Hill, 1969.

[58] V. D. Liseikin. *Grid Generation Methods.* Springer-Verlag New York, 1999

[59] R. Löhner, K. Morgan, and O. C. Zienkiewics. Adaptive Grid Refinement for the Compressible Euler Equations. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, ed. by I. Babusuka et. al. (Wiley 1986) pp. 281–297.

[60] M. J. Marchant and N. P. Wheatherill. Adaptivity Techniques for Compressible Inviscid Flows.

[61] *Computer Methods in Applied Mechanics and Engineering*, 106:83-106, 1993. J. März. Improving Time Accuracy for Residual Distribution Schemes. VKI PR-1996-17, von Karman Institute for Fluid Dynamics, 1996.

[62] D. S. McRae. R-Refinement Grid Adaptation Algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 189:1161-1182, 2000.

[63] L. M. Mesaros. Multi-Dimensional Fluctuation Splitting Schemes for the Euler Equations on Unstructured Grids. PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1995.

[64] K. Miller and R. N. Miller. Moving Finite Elements I. *SIAM Journal on Numerical Analysis*, 18:1019-1032, 1981.

[65] K. Miller. Moving Finite Elements II. *SIAM Journal on Numerical Analysis*, 18:1033-1057, 1981.

[66] L. M. Milne-Thomson. *Theoretical Hydrodynamics*. Dover, 1996.

[67] K. W. Morton. *Numerical Solution of Convection-Diffusion Problems*. Chapman & Hall, 1996.

[68] K. W. Morton and P. L. Roe. Vorticity-Preserving Lax-Wendroff-Type Schemes for the System Wave Equation. *SIAM Journal on Scientific Computing*. (to appear).

[69] K. Nakahashi and G. S. Deiwert. Self-Adaptive-Grid Method with Application to Airfoil Flow. *AIAA Journal*, 25(4):513-520, 1987.

[70] R. A. Nicolaides. Direct Discretization of Planar Div-Curl Problems. *SIAM Journal on Numerical Analysis*, 29(1):32-56, 1992.

[71] H. Nishikawa, M. Rad, and P. L. Roe. Grids and Solutions from Residual Minimisation. *Proceedings of the First International Conference on Computational Fluid Dynamics*, Kyoto, Japan, July 9-14 2000.

[72] B. Palmerio and A. Dervieux. 2D and 3D Unstructured Mesh Adaptation Relying on Physical Analogy. *Proceedings 2nd International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pp. 653-663, Miami, Florida, December 1988.

[73] B. Palmerio and A. Dervieux. Application of a FEM Moving Node Adaptive Method to Accurate Shock Capturing. *Proceedings 1st International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pp. 653-663, Landshut, Germany, July 1986.

[74] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewics. Adaptive Remeshing for Compressible Flow Computations. *Journal of Computational Physics*, 72(2):449-466, 1987.

[75] J. Peraire, K. Morgan, K. Peiro, and O. C. Zienkiewics. An Adaptive Finite Element Method for High Speed Flows. AIAA Paper 87-0558, 1987.

[76] L. F. Pierce and M. B. Giles, Adjoint Recovery of Superconvergent Functionals From PDE Approximations. *SIAM Review*, 42(2):247-264, 2000.

[77] S. Z. Pizadeh. An Adaptive Unstructured Grid Method by Grid Subdivision, Local Remeshing, and Grid Movement. AIAA Paper 99-3255, 1999.

[78] M. Rad. A Residual Distribution Approach to the Euler Equations that Preserves Potential Flow. The University of Michigan, Ann Arbor, Michigan, 2001.

[79] S. Rippa. Long and Thin Triangles can be Good for Linear Interpolation. *SIAM Journal on Numerical Analysis*, 29(1):257-270, 1992.

[80] P. L. Roe. Compounded of many simples. *Barriers and Challenges in CFD*, ed. by V. Venkatakrishnan, M. D. Salas and S. R. Chakravarthy (Kluwer, 1997) pp 241-258.

[81] P. L. Roe and E. Turkel. The Quest for Diagonalization of Differential Systems. *Barriers and Challenges in CFD*, ed. by V. Venkatakrishnan, M. D. Salas and S. R. Chakravarthy (Kluwer, 1997) pp 351-369.

[82] P. L. Roe. Fluctuation Splitting Schemes on Optimal Grids. AIAA Paper 97-2032, 1997.

[83] P. L. Roe. Error Estimates for Cell Vertex Schemes. Technical Report 87-6, ICASE, 1987.

[84] P. L. Roe. Fluctuations and Signals, a Framework for Numerical Evolution Problems. *Numerical Methods for Fluid Dynamics II*, ed. by K. W. Morton and M. J. Baines (Academic Press, 1982) pp 219-257.

[85] P. L. Roe and H. Nishikawa. Adaptive Grid Generation by Minimising Residuals. *Proceedings of the ICFD Conference on Numerical Methods for Fluid Dynamics*, Oxford, UK, March 26-29 2001.

[86] B. Schutz. *Geometrical Methods of Mathematical Physics*. Cambridge, 1980.

[87] B. K. Soni, R. Koomullil, D. S. Thompson, and H. Thornburg. Solution Adaptive Grid Strategies based on Point Redistribution. *Computer Methods in Applied Mechanics and Engineering*, 189:1183-1204, 2000.

[88] F. Taghaddosi, W. G. Habashi, G. Guevremont, and D. Alit-Ali-Yahia. An Adaptive Least-Squares Method for the Compressible Euler Equations. *International Journal for Numerical Methods in Fluids*, 31:1121-1139, 1999.

[89] J. L. Thomas and M. D. Salas. Far Field Boundary Conditions for Transonic Lifting Solutions to Euler Equations. *AIAA Journal*, 24:1074-1080, 1986.

[90] J. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of Grid Generation*. CRC Press, 1999.

[91] Y. Tourigny and F. Hulsemann. A New Moving Mesh Algorithm for The Finite Element Solution of Variational Problems. *SIAM Journal on Numerical Analysis*, 35:1416-1438, 1998.

[92] S. H. Weintraub. *Differential Forms: A Complement to Vector Calculus.* Academic Press, 1997

[93] M. G. Vallet, F. Hecht, and B. Mantel. Anisotropic Control of Mesh Generation based upon a Voronoi Type Method. *Proceedings 3rd International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pp.93-103, Barcelona, Spain, June 1991.

[94] M. V. Dyke. *Perturbation methods in fluid mechanics.* Academic Press, 1964.

[95] M. Vanmaele, K. W. Morton, E. Suli, A. Borzi. Analysis of the Cell Vertex Finite Volume Method for the Cauchy-Riemann Equations. *SIAM Journal on Numerical Analysis*, 34(5):2043-2062, 1997.

[96] D. A. Venditti and D. Darmofal. Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow. *Journal of Computational Physics*, 164:204-227, 2000.

[97] G. P. Warren, W. K. Anderson, J. T. Thomas, and S. L. Krist. Grid Convergence for Adaptive Methods. AIAA Paper 91-1592, 1991.

[98] A. B. White. On the Selection of Equidistributing Meshes for Two-Point Boundary Value Problems. *SIAM Journal on Numerical Analysis*, 16:473-502, 1979.

[99] W. A. Wood, W. L. Kleb. On Multi-dimensional Unstructured Mesh Adaptation. AIAA Paper 99-3254, 1999.

[100] D. G. Zeitoun, J. P. Laible, and G. F. Pinder. A Weighted Least Squares Method for First-Order Hyperbolic Systems. *International Journal For Numerical Methods in Fluids*, 20:191-212, 1995.

[101] X. D. Zhang, J.-Y. Trepanier, and R. Camarero. An a posteriori Error Estimation Method Based on An Error Equation. AIAA Paper 97-1889, 1997.

[102] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*, Vol.1. McGraw-Hill Book Company, 1994.