Grid Adaptation for Functional Outputs of Compressible Flow Simulations

by

David Anthony Venditti

B.Eng., McGill University, Canada (1996)
M.Eng., McGill University, Canada (1998)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Author ______________________________
Department of Aeronautics and Astronautics
May 24, 2002

Certified by ______________________________
David L. Darmofal
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ______________________________
Mark Drela
Professor of Aeronautics and Astronautics

Certified by ______________________________
Anthony T. Patera
Professor of Mechanical Engineering

Certified by ______________________________
Jaime Peraire
Professor of Aeronautics and Astronautics

Accepted by ______________________________
Professor Wallace E. Vander Velde
Chair, Committee on Graduate Students

Grid Adaptation for Functional Outputs of Compressible Flow Simulations

by

David Anthony Venditti

Submitted to the Department of Aeronautics and Astronautics
on May 24, 2002, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

An error correction and grid adaptive method is presented for improving the accuracy of functional outputs of compressible flow simulations. The procedure is based on an adjoint formulation in which the estimated error in the functional can be directly related to the local residual errors of both the primal and adjoint solutions. This relationship allows local error contributions to be used as indicators in a grid adaptive method designed to produce specially tuned grids for accurately estimating the chosen functional. The method is applied to two-dimensional inviscid and viscous (laminar) flows using standard finite volume discretizations, and to scalar convection–diffusion using a Galerkin finite element discretization.

Isotropic h–refinement is used to iteratively improve the grids in a series of subsonic, transonic, and supersonic inviscid test cases. A commonly–used adaptive method that employs a curvature sensor based on measures of the local interpolation error in the solution is implemented to comparatively assess the performance of the proposed output–based procedure. In many cases, the curvature–based method fails to terminate or produces erroneous values for the functional at termination. In all test cases, the proposed output–based method succeeds in terminating once the prescribed accuracy level has been achieved for the chosen functional.

Output–based adaptive criteria are incorporated into an anisotropic grid–adaptive procedure for laminar Navier–Stokes simulations. The proposed method can be viewed as a merging of Hessian–based adaptation with output error control. A series of airfoil test cases are presented for Reynolds numbers ranging from 5,000 to 100,000. The proposed adaptive method is shown to compare very favorably in terms of output accuracy and computational efficiency relative to pure Hessian–based adaptation.

Thesis Supervisor: David L. Darmofal
Title: Associate Professor of Aeronautics and Astronautics

To Marta

The error therefore lyeth neither in the abstract nor in geometry, nor in physicks, but in the calculator, that knoweth not how to adjust his accompts.

– Galileo Galilei, 1632

Acknowledgments

I wish to express my sincere gratitude to my thesis supervisor, Professor David L. Darmofal, for his unwaivering support, guidance, and friendship throughout the course of this work. I am indebted to him for introducing me to the truly fascinating topic that is the basis of this dissertation, and for the continuous stream of insightful comments and suggestions that helped propel the research forward. I am also very grateful that he made the effort to introduce me to some of the most highly respected researchers in the field.

I would like to thank the other members of my committee, Professors Drela, Patera, and Peraire for their constructive comments and criticism. Their participation is greatly appreciated. I am grateful for the involvement of Professor Lo as my minor advisor. Thanks are also due to the external readers of the thesis: Dr. Steven Allmaras, Dr. James Thomas, and Professors Giles, Habashi, Pierce, and Willcox.

Special thanks are directed towards Mike Giles and Jim Thomas for comments that were particularly useful during the course of the research including some excellent suggestions for test cases. The papers by Mike Giles, Niles Pierce and their collaborators were invaluable references for this work. Dr. W. K. Anderson, and Dr. E. J. Nielsen deserve a special acknowledgment for their valued support in using and modifying FUN2D. I also benefited from several discussions with Mike Park at Langley.

I feel privileged to have had the opportunity to meet and befriend my fellow lab mates, faculty and staff in the ACDL (formerly, FDRL). In particular, I must thank the lab elders – Ed, Angie, Ali, Tolu, Karen, and Alex – for facilitating my arrival and integration into the lab. Linux and Unix assistance was generously provided by Ed, Angie, Tolu, and Victor more times than I can remember. I should also add Bob Haimes to that list; Bob provided color to the daily affairs of the lab; and if he was in the building, you never felt lonely. Jean deserves a wholehearted acknowledgment for keeping the lab running and for occasionally getting me out of a jam. My compliments to Tolu – Cplot came in very handy towards the end. Some of the more interesting conversations I enjoyed in this lab were with Joseph; topics ranged from flux functions to economic policy to Japanese cuisine and beyond. Larry also enjoyed a good discussion and always provided an excellent perspective on things; I am particularly glad our paths crossed in this lab and for the friendship we have developed. Anyone who has ever met Adam cannot leave him off an acknowledgments page; he is always able to brighten your day. I have benefited greatly from getting to know the newer guys in the lab: Ricardo (nutritionalist, conversationalist), Tony, Keith, Sean, Vince, Dave W., Sudeep, James (versatile mathematician), Mark, and the French guys–Guillaume and Jerome. Amongst those in the neighboring labs, I am particularly grateful to have befriended Luis, Jameel, Jorge, Simon, and Tatsuo. Thanks are due to Luis, Jorge and Adam for helping me move in and out of apartments. As for my old friends Chuk and Walter, I appreciate the honest advise and friendship they provided. Walter kept me in shape with his intense squash matches. I was very honoured that Chuk took the red–eye express just to attend my defense.

A sincere thanks goes to my mother who always provided me with an abundant supply of lasagna every time I drove home. I am grateful for her perpetual and unconditional love and support. The help that Mike and Katherine provided during the most recent move was greatly appreciated. I am fortunate to have a brother and sister–in–law of their caliber.

In a few short months, I will marry the girl of my dreams. Although we have been separated by 330 miles (528 km.) for most of the last few years, I feel closer to her than ever. I will not miss the drive along I–93 and I–89 – Lord knows I can do it in my sleep – but I will never forget the fabulous weekends we had together and the magical evenings in Montpelier. I will also cherish the memories on Marlborough and, especially, rue Poirier de Narçay. Je t'aime, mon amour. Ça, c'est pour toi.

The support of the National Science Foundation (ACI-9896341), The Boeing Company, NASA Langley (NAG1-2275), and FCAR (Fonds pour la Formation de Chercheurs et l'Aide à la Recherche) is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational simulations of complex aerodynamic flows have become integral components of the design process in the aerospace industry due to their potential to provide valuable physical data thereby reducing the need for costly wind tunnel testing. Increasing reliance on computer simulations necessitates a corresponding increase in the accuracy and reliability of the underlying algorithms. Furthermore, large-scale simulations must be made more affordable so that their potential benefits can be fully realized within the design cycle.

A well known strategy for minimizing the cost of a computational simulation while achieving a given level of accuracy is grid adaptation [1, 6, 8, 10, 15, 16, 27, 28, 30, 33, 45, 48, 50, 51, 63, 64, 66, 68, 70, 71, 72]. The basic premise is to locally enrich the computational grid in regions which most adversely affect the accuracy of the final solution while coarsening the grid in more benign regions to prevent incurring unnecessary computational costs. A major difficulty in achieving definite improvements using adaptation for Euler and Navier–Stokes calculations is the lack of reliable error indicators. For example, a common strategy is to adapt to certain physical features of the flow, such as shock waves, boundary layers, wakes, slip lines, or stagnation points, by employing indicators based on large flow gradients or undivided differences [6, 8, 50, 64]. The assumption here is that regions of larger gradients are associated with regions of larger error. Unfortunately, continuous local refinement of the dominant features of the flow does not necessarily guarantee that certain measures of the global error will simultaneously be reduced. In some cases this procedure may even lead to incorrect results[1] [64].

An alternate approach is to use adaptive indicators based on the local interpolation error in the solution [16, 27, 28, 48, 64, 67]. The error associated with piecewise–linear interpolation of a scalar field is proportional to the second derivatives of that scalar. Thus, these indicators essentially adapt to the curvature of the solution. From this perspective they share some of the qualities and potential deficiencies associated with feature–based indicators for nonlinear flow problems.

Within the context of linear elliptic problems, a relatively rigorous adaptive procedure has been outlined by Zienkiewicz and Zhu [70, 71, 72]. They propose using a recovery technique to obtain higher-order projections of the solution gradients. The local error is assessed by comparing the solution gradients with the higher-order projections. Unfortunately, these types of estimates cannot be made rigorous for convection–dominated flow problems.

Error estimates based on local residual evaluations can be used as indicators for adaptation. Aftosmis and Berger [1] employ a multilevel Richardson–type indicator for adaptive refinement within a finite volume context. Error estimates are constructed by evaluating the discrete residual on a coarse grid using a restricted solution from a fine grid. Zhang *et al.* [68] derive local adaptive indicators by evaluating the residual of the governing partial differential equations using the approximate numerical solution.

The aforementioned adaptive indicators all use local measures of the error in one form or another. However, the local error in a discrete flow solution may be dominated by remote effects in an entirely different part of the domain [13, 51]. An example of where such an occurrence might arise would be in a transonic flow. An adaptive algorithm based on derivatives in the local Mach number would tend to provide very high grid resolution near shocks. While this would result in a sharply–resolved shock, the shock location could, nevertheless, be in significant error due to a lack

[1] The term *incorrect results* refers to certain measures of the error converging to non-zero values as the refinement process proceeds indefinitely.

of grid resolution far upstream from the shock (for examples of such an occurrence, see reference [64]). This, in turn, could lead to substantial errors in other derived quantities.

Even within the setting of finite-element-discretized elliptic problems, where rigorous error estimates have been available for decades [2, 58], one can argue that a global error norm based directly on the solution and its derivatives may not be optimal within an engineering context. The issue is even less clear for Euler and Navier–Stokes computations that have multiple length scales and isolated flow features scattered throughout complicated, multidimensional domains. Adaptation based on a global error norm of this nature generally leads to a somewhat uniform consideration of all the features in a flow, which may not be desirable. For example, if one is only interested in computing the drag on an engine nacelle mounted below the wing of an aircraft, or the lift generated by a canard, it may not be necessary to resolve the wake downstream of the rudder to any appreciable extent.

An alternate approach to making error estimation more relevant for engineering applications is to assess the error made in predicting an integral quantity representing an engineering output. For example, in aerodynamic applications, outputs of interest include the lift, drag and moment coefficients on aircraft configurations. These are all expressible as surface integrals over portions of the domain boundary. Other examples include the total losses across a blade row in a gas turbine engine, the total heat flux to a high–pressure turbine blade, the acoustic noise levels at an airport terminal due to an aircraft taking off, or the rate of ice formation on an aircraft wing during adverse flight conditions.

This dissertation presents an error estimation and grid adaptive method specifically designed for improving the accuracy of functional outputs from numerical simulations. The method is applicable to general discretizations including finite volume, finite element, or finite difference approximations. This thesis focuses on finite volume implementations for inviscid and viscous (laminar) compressible flows, and a Galerkin finite element discretization for scalar convection–diffusion. The procedure invokes a duality concept in which an equivalent *dual* or *adjoint* formulation of the original *primal* problem is exploited. The primary benefit of invoking the dual problem, in the context of grid adaptation, is that the error in a chosen functional can be directly related to local residual errors of the primal solution through the adjoint variables. This property elucidates the potential for devising an optimal grid adaptive strategy designed to produce specially tuned grids for maximizing the accuracy of a particular functional.

1.1 Error Analysis for Functional Outputs

The first results related to error analysis for functional outputs likely originated in the work of Aubin and Nitsche (see reference [58]). They developed a technique (widely known as *Nitsche's trick*) for proving *a priori* finite element convergence rates for elliptic problems in certain norms other than the natural norm (or energy norm) for the particular problem. The technique makes use of a suitably defined adjoint problem that serves as an artifice in the proof.

Babuška and Miller [5] were perhaps the first to focus attention on functional outputs. In structural analysis applications, the outputs of interest are typically point quantities such as displacements, rotations and moments. In their paper, Babuška and Miller outlined a procedure whereby these point quantities could be converted into integral expressions using what they termed *extraction functions.* Using an *auxiliary* (or adjoint) problem based on the extraction functions, and under certain smoothness conditions, these integral quantities were shown to converge at the same rate as the strain energy, whereas, the point quantities obtained directly from the finite element solution converged at suboptimal rates.

More recently, there has been a significant volume of research into *a posteriori* error analysis and optimal grid adaptation for functional outputs within the context of finite element methods for fluid dynamics. Researchers in the area include Becker, Rannacher and collaborators [9, 10, 13, 51], Süli and co-workers [23, 30, 59], Larson and Barth [34], and Patera, Peraire and collaborators [38, 46, 47].

Becker, Rannacher and collaborators [9, 10, 13, 51] have developed an optimal control approach for output–based grid adaptation within a Galerkin finite element framework. Their approach exploits finite element orthogonality properties and duality concepts that involve the use of an adjoint problem. *A posteriori* error estimates are used to derive effective indicators for adaptation.

Süli and Houston [30, 59] present an *a posteriori* error analysis of the *hp*–version of the discontinuous Galerkin finite element method. Using duality arguments, computable error bounds are derived for linear functionals of scalar, first–order hyperbolic problems. These bounds are used to design an adaptive algorithm that is capable of both local mesh subdivision and local polynomial–degree enrichment.

Larson and Barth [34] have outlined an *a posteriori* error estimation procedure for specified functionals of first–order systems of conservation laws for the discontinuous Galerkin finite element method. Using duality techniques, error representations are derived for linear and nonlinear functionals given an associated bilinear or nonlinear variational form.

Patera, Peraire and collaborators [38, 46, 47] have established an implicit *a posteriori* procedure for computing upper and lower bounds on functional outputs of finite element solutions. These error bounds are measured with respect to the value of the output on a suitably refined *truth mesh* that is considered sufficiently fine that the discretization errors may be neglected. They incorporate an adaptive procedure into the bounding framework designed to produced optimized grids that meet a target bound gap.

Pierce and Giles [24, 25, 26, 49] have developed an adjoint-based error correction technique for functional outputs. Essentially, this technique extends superconvergence properties, automatically inherent in many finite element methods, to cover numerical results obtained from any numerical method, including finite difference, finite volume, or finite element without natural superconvergence. Moreover, the technique can also be used to improve the accuracy of superconvergent functionals obtained from finite element methods by constructing smoother, higher-order interpolants of the primal and dual solutions.

The output-based adaptive method outlined in this dissertation is based on an algebraic version of the Pierce and Giles correction technique. It shares the advantage of being applicable to any type of discretization method, including finite volume methods which are widely used for the simulation of compressible flows. The present methodology also borrows, from Patera and Peraire, the concept of the *truth mesh*, referred to in the present context as simply the *fine grid*, which is to be distinguished from the associated *working* or *coarse grid.* The procedure involves estimating the error in the coarse-grid functional with respect to its value on a globally refined fine grid. Solutions on the fine grid are not required. The only auxiliary computations are functional and residual evaluations on the fine grid, and the solution of a linear adjoint problem on the coarse grid. The adaptive algorithm strives to improve the quality of the aforementioned error estimate by attempting to reduce and equidistribute the remaining error in the functional after correction. At the next iteration, a new fine grid is defined with respect to the newly adapted coarse grid and the process is repeated until convergence. Convergence is defined in terms of local and global adaptation parameters and a prescribed error level for the computed functional.

1.2 Unstructured Grid Adaptation

A traditional method of discretizing the computational domain is to employ structured or block–structured grids comprised of regular arrays of quadrilateral or hexahedral cells in two or three dimensions, respectively. Generating these types of grids on regularly shaped domains is usually a straightforward matter, however, they become exceedingly difficult to construct as the domains become increasingly more complex. Unstructured grids employing triangles in two dimensions or tetrahedra in three dimensions have emerged as a viable alternative to structured grids, primarily due to their ability to discretize geometrically complex domains with relative ease while providing a natural setting for the implementation of grid–adaptive techniques.

The compressible flows considered in this work are simulated using unstructured grids composed of triangular cells or elements. While the numerical results are focused on two-dimensional test cases, the concepts, algorithms and procedures presented in this dissertation are, in principal, applicable or extendable to three dimensions. In particular, analogies to unstructured tetrahedral grids can be readily drawn in most cases [45].

Standard h–version isotropic adaptation is used for the inviscid simulations in this work. At each iteration in a typical adaptive loop, criteria are derived for determining the desired element sizes for the new grid. There is no information provided for stretching or reorienting the elements in this case. The viscous simulations employ anisotropic adaptation. At each node in the domain, information regarding the local size, stretching, and orientation of the elements is provided. Based on this information, the grid is regenerated using a generalized Delaunay–type algorithm [12, 35] and the entire process is repeated, iteratively, until the coupled grid–solution system has converged.

The output–based anisotropic grid–adaptive method proposed in this work is an extension of current anisotropic methods. Peraire *et al.* [48] proposed a directional refinement method for two-dimensional inviscid flows. In their work, grids composed of stretched triangles are constructed with biased resolution along directions of rapidly changing gradients of the density. The approach they adopt is based on reducing the interpolation error in the density by focusing on the Hessian matrix of second derivatives. The local Hessian is diagonalized and the absolute value of the eigenvalues are used to determine the local grid spacing in two orthogonal directions given by the corresponding eigenvectors. This allows stretching parameters and principal directions to be defined over the entire domain. The grid is then regenerated based on equidistributing the estimated error along the eigenvector directions.

The more recent work of Castro–Díaz *et al.* [16], Habashi *et al.* [27, 28] and Dervieux *et al.* [19] also employs anisotropic adaptive indicators based on the interpolation error in a chosen scalar or combination of scalars. They make use of a symmetric positive–definite metric obtained from the Hessian by taking the absolute value of its eigenvalues. The metric, which varies continuously over the domain, can be interpreted as a transformation matrix from the physical space to a Riemannian space. Equidistributing the error along edges in the grid can be achieved by equidistributing the length of the edges in the Riemannian space. Castro–Díaz *et al.* use an *hr*–adaptive strategy to achieve directionally–adapted unstructured grids with high–aspect–ratio elements. Habashi *et al.* generate highly stretched grids using local operations: refinement, coarsening, edge swapping and node movement, without any recourse to remeshing. The use of local grid operations enables tighter coupling with the solver.

A common scalar used in the Hessian–based approaches for viscous flows has been the Mach number [6, 27, 67]. Unfortunately, several difficulties are associated with using the Mach number Hessian alone. Xia *et al.* [67] present a summary of heuristics for modifying the Hessian in an attempt to alleviate some of its inherent limitations as an adaptive sensor. Examples include bounding the eigenvalues from below when the flow is nearly uniform and the Hessian approaches zero. Conversely, the eigenvalues are capped from above in the vicinity of shocks where the Hessian becomes virtually unbounded in the normal direction. Another difficulty can arise in boundary layers when the Mach number passes through curves of inflection or layers of very low curvature. In practice this may result in nearly isotropic elements within the boundary layer adjacent to the wall.

For systems of PDEs involving multiple dependent variables, the choice of a single scalar may not be appropriate. Combinations of scalars can be used, but the appropriate weighting is unclear. Castro–Díaz *et al.* [16] suggest using the *intersection metric* associated with all the conservative variables of the system. This amounts to approximating the metric that minimizes the maximum interpolation error over all variables. Incorporating all the conservative variables into the Hessian calculation is an improvement over using a single scalar variable. However, no justification is given for choosing the conservative variables over any other set of variables. Furthermore, their criterion still focuses on the interpolation error alone, without addressing other sources of error.

Castro–Díaz *et al.* have shown that anisotropically adapted grids for high–Reynolds–number flows often yield poor boundary layer resolution and inaccurate skin friction estimates. This is attributed to the nonuniformity in the distance of the first layer of nodes to the wall. They recommend modifying the metric near the boundaries in order to obtain a *quasi-orthogonal* grid in near-wall regions. This requires the user to specify a spacing parameter for the first layer of elements near the wall.

The output-based adaptive method used in the present work is able to alleviate some of the aforementioned shortcomings of pure Hessian-based adaptation, while maintaining a rigorous link to the functional error. The anisotropic adaptive algorithm merges Hessian-based information with adjoint-based error estimation. Metric information is decomposed into three distinct characteristics for each element: size, aspect ratio, and orientation. The Mach number Hessian is used to obtain stretching and orientation information for each element while the adjoint information is used to determine its size.

No explicit effort is made to avoid large–angle triangles for the anisotropic test cases in this work[2]. Babuška and Aziz [4] have shown that the accuracy of a finite element approximation on a triangular element degrades as the maximum angle of the element is increased. This suggests that stretched obtuse-angle triangles containing one large angle and two small angles should be avoided, whereas, triangles with one small angle and two nearly right angles are preferred. An acceptable limit on how large the largest angle may be depends on the specific application. For linear interpolation, Rippa [52] has shown that if the function being interpolated has a highly preferred direction, angles approaching π may still be acceptable provided the triangle is appropriately oriented. He presents bounds on the interpolation error in terms of the triangle shape and the second derivatives of the function.

For the directionally adapted grids in this thesis, the largest angle in a typical triangulation often approaches π to within 1%, particularly in higher Reynolds number simulations. Nevertheless, the adaptive results indicate that this has not been a significant factor, in regards to output accuracy, whenever the adjoint-based adaptive criteria have been used.

[2] The grid generator used for the viscous test cases [35] implements various grid quality measures that may influence the magnitude of the largest angles.

1.3 Thesis Overview

In Chapter 2, the output–based error correction and grid adaptive procedure is described in detail. The chapter begins with a discussion of some preliminary concepts and definitions associated with the functional correction technique. The method is described within a general framework, without reference to a particular discretization method. A brief description of the Pierce and Giles [26, 49] correction technique is provided (a more detailed treatment is given in Appendix D). The present method is essentially an application of their method within an algebraic context. In Section 2.3, the proposed adaptive methodology is described. The functional error is decomposed into two contributions: a computable term and a remaining error term. The computable term is the correction term that can be used to improve the accuracy of the chosen functional. The proposed adaptive algorithm is designed to enhance the quality of the correction term by reducing the magnitude of the remaining error term. Adaptive parameters are derived by invoking the principle of error equidistribution. In the proposed adaptive procedure, local elemental contributions to the remaining error are driven towards the same value throughout the domain, while the estimated global remaining error is reduced towards a user–specified tolerance.

In Chapter 3, the functional correction and grid–adaptive procedure is applied to two–dimensional, inviscid flows covering subsonic, transonic, and supersonic regimes. The functionals considered are the lift, drag and moment coefficients on a variety of airfoil configurations. Isotropic grid–refinement is applied to improve the quality of the grids. A curvature–based adaptive method [6, 64] is implemented to comparatively assess the performance of the output–based method. The output–based method is shown to compare favorably in terms of output accuracy, computational efficiency and reliability.

In Chapter 4, the functional correction and grid–adaptive methodology is applied to two–dimensional, laminar, subsonic Navier–Stokes flows for Reynolds numbers ranging from 5,000 to 100,000. The functionals considered are the lift and drag coefficients in a series of airfoil test cases. The output–based adaptive criteria are incorporated into an anisotropic grid–adaptive procedure. The local element size is determined using output–based criteria, whereas the local stretching and orientation is based on the computed Mach number Hessian. An anisotropic grid–adaptive method based exclusively on the Mach number Hessian [16, 19, 27, 48] is implemented for the purposes of comparison. The proposed output–based method is shown to compare very favorably in terms of output accuracy and computational efficiency relative to pure Hessian–based adaptation.

Chapter 5 concludes with a summary of the thesis, a delineation of the primary contributions, and suggestions for future work. In Appendix A, the output correction/adaptive method is applied to a Galerkin finite–element discretization of scalar convection–diffusion. The functional correction theory, specialized for a linear problem, is presented in A.4. Appendix D presents the continuous adjoint correction theory of Pierce and Giles [25, 26, 49] and draws analogies with the algebraic approach employed in the current work.

Chapter 2

Grid Adaptation for Functional Outputs

A correction technique and grid adaptive methodology for functional outputs is presented. The functional correction technique is based on a discrete adjoint formulation that relates the local residual errors in the flow solution to the corresponding global error in the output. A correction term is thus derived that can be used to improve the accuracy of the chosen output once an auxiliary adjoint problem has been solved. The proposed adaptive methodology is based on reducing the magnitude of the remaining error in the functional, after correction. Adaptive parameters are derived by invoking the principle of error equidistribution. During a typical adaptive simulation, local elemental contributions to the remaining error are driven towards a common value throughout the domain while the estimated global remaining error is reduced towards a user–specified tolerance. The correction/adaptive methodology is presented within a general framework, without reference to a particular discretization method.

2.1 Adjoint Error Correction

This section begins with some preliminary definitions related to the mechanics of the functional correction technique, which is essentially an algebraic version of the Pierce and Giles [26, 49] method. The functional correction term is derived in Section 2.1.2.

	Coarse Grid (Ω_H)	Fine Grid (Ω_h)
Characteristic Element Size	H	h
Discrete Solution	U_H	U_h
Residual Equations	$R_H(U_H) = 0$	$R_h(U_h) = 0$
Functional Output	$f_H(U_H)$	$f_h(U_h)$

Table 2.1: Nomenclature associated with coarse- and fine-grid quantities

2.1.1 Preliminary Definitions

The primary goal is to estimate an integral quantity $f(U)$ that can be written as a nonlinear functional of the solution U to a system of partial differential equations (PDEs). The PDEs are defined over a physical domain denoted by Ω. Consider two distinct computation grids that approximate the physical domain: a coarse grid Ω_H and a fine grid Ω_h. The parameters H and h $(H > h)$ represent characteristic lengths associated with each grid such as the average edge length in a finite element or finite volume triangulation, or the average grid spacing in a finite difference approximation. The nonlinear system of discrete residual equations arising from some discretization of the original PDEs on the coarse grid is denoted $R_H(U_H) = 0$, where U_H is the corresponding discrete solution on that grid. The discrete approximation of $f(U)$ on the coarse grid using a prescribed quadrature rule is denoted $f_H(U_H)$. Analogous quantities are defined for the fine grid as summarized in Table 2.1.

Consider the *coarse grid* as being representative of a typical working grid; one that might be used in a design context, for example, or some application requiring rapid simulation turnover. While the coarse grid may be regarded as affordable with respect to available computing resources and allowable solution times, it may not provide sufficient accuracy for the intended application. More precisely, it may not provide a sufficiently accurate output $f_H(U_H)$.

The *fine grid* is considered to be a grid of improved resolution relative to the coarse grid. Solving on the fine grid would generally be expensive, however, if the discrete solution were obtained on this grid, the computed output would have in-

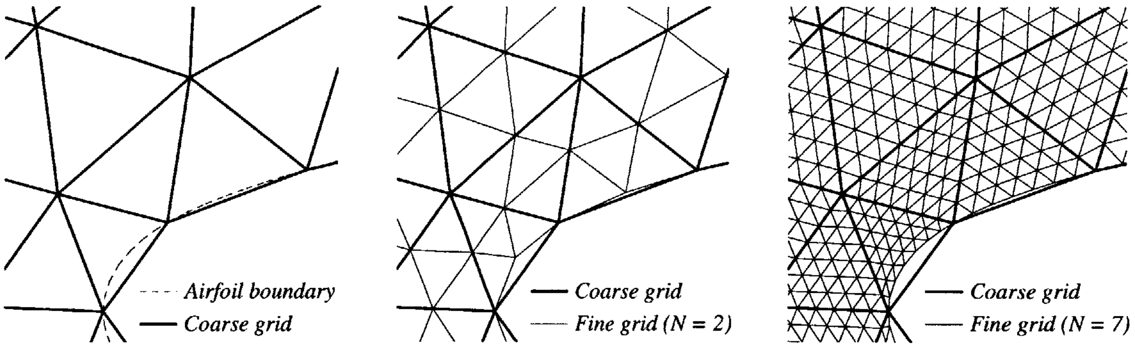


Figure 2-1: A typical coarse grid and two possible fine grids that may be used in the functional correction procedure. Left: patch of coarse-grid elements near an airfoil boundary; center: superimposed fine grid corresponding to $N = 2$; right: superimposed fine grid corresponding to $N = 7$. Note that the fine grids conform to the original airfoil boundary.

creased accuracy for the particular application. In the present work, the fine grid is taken to be a uniform refinement of the coarse grid. At the boundaries of the domain it is made to conform to the physical boundary $\partial\Omega$. For example, if Ω_H is a two-dimensional triangulation, one way of constructing Ω_h is to subdivide each of the triangles of Ω_H into an integer number, N^2, of self-similar triangles where $N = H/h$. The fine-grid boundary nodes are then repositioned so as to coincide with $\partial\Omega$. This may also require the movement of some interior nodes adjacent to the boundary in order to maintain the quality and integrity of the fine grid. Figure 2-1 shows a patch of elements in a typical coarse grid and two possible fine grids corresponding to $N = 2$ and $N = 7$. Note that a hierarchy of fine grids corresponding to $N = 2, 3, \ldots$ can be constructed in this way and that each grid in the hierarchy would be completely characterized by N with respect to the original coarse grid.

If Ω_H is a three dimensional grid composed of tetrahedra, a slightly different approach is required since it is not possible to subdivide a tetrahedron into self-similar tetrahedra. Alternatively, a 1:12 refinement ratio can be achieved by adding nodes to the mid-points of each of the edges of the original tetrahedron and by adding one more node to its centroid. The application of this systematic refinement algorithm to each tetrahedron in the coarse grid would yield a legitimate fine grid for the purposes of the error estimation procedure. Recursive repetition of this process would result in a hierarchy of grids where each successive grid would be embedded within the previous one. As in the two-dimensional case, each fine grid should be made to conform to the physical boundary, $\partial\Omega$.

The goal in the subsequent section will be to obtain an accurate estimate of $f_h(U_h)$ without ever solving on the fine grid. The only auxiliary computations will be residual evaluations and inner products on the fine grid, and the solution of a linear adjoint problem on the coarse grid.

2.1.2 Functional Correction

In this section, the focus will be on deriving a correction term for the error in the fine-grid output due to a perturbation in the fine-grid solution. In the present context, the perturbed solution will be obtained by prolongating the coarse-grid solution onto the fine grid. In general, however, it could be obtained from other sources such as an asymptotic expansion, a reduced order model, or an entirely different discretization. The impetus for this procedure is the correction technique of Pierce and Giles [26, 49].

Let δU_h represent a small perturbation in the primal solution. Each component of the vector δU_h corresponds to the local error in the perturbed solution relative to the exact solution of the primal residual equations on the fine grid. The perturbed solution is denoted $\tilde{U}_h$ and is defined by

$$\tilde{U}_h \equiv U_h + \delta U_h. \tag{2.1}$$

In practice, the fine-grid solution U_h and the error δU_h are not known, whereas, the approximate solution $\tilde{U}_h$ is assumed given. The resulting perturbations in the functional and residual operators due to the perturbation in the solution are, respectively,

$$\delta f_h \equiv f_h(\tilde{U}_h) - f_h(U_h), \tag{2.2}$$

and,

$$\begin{aligned} \delta R_h &\equiv R_h(\tilde{U}_h) - R_h(U_h), \\ &= R_h(\tilde{U}_h). \end{aligned} \tag{2.3}$$

The last equality holds due to the fact that the fine-grid solution satisfies the residual equations $R_h(U_h) = 0$. Linearizing about the perturbed solution yields,

$$\delta f_h \approx \frac{\partial f_h}{\partial U_h} \delta U_h, \tag{2.4}$$

and,

$$\delta R_h \approx \frac{\partial R_h}{\partial U_h} \delta U_h. \tag{2.5}$$

The row vector, $\partial f_h / \partial U_h$, contains the linear sensitivities of the functional with respect to the solution vector. The square matrix $\partial R_h / \partial U_h$ is the Jacobian of the residual operator. Both quantities are evaluated using the perturbed solution.

The discrete adjoint system of equations associated with the primal residual operator and functional is given by[1]

$$\frac{\partial R_h}{\partial U_h}^T \Psi_h = \frac{\partial f_h}{\partial U_h}^T, \tag{2.6}$$

where Ψ_h is the discrete adjoint solution on the fine grid. Using (2.4)–(2.6), the perturbation in the functional can be expressed as

$$\begin{aligned} \delta f_h &\approx \frac{\partial f_h}{\partial U_h} \delta U_h, \\ &= \Psi_h^T \frac{\partial R_h}{\partial U_h} \delta U_h, \\ &\approx \Psi_h^T R_h(\tilde{U}_h). \end{aligned} \tag{2.7}$$

[1] Supplemental information on adjoint equations and their uses can be found in references [21, 24, 32, 37, 42, 57].

The last expression in (2.7) approximates the error in the functional as the inner product of the adjoint solution and the primal residual error. This expression is exact for linear functions and residuals. To compute this estimate would require the solution of the adjoint problem on the fine grid, which is undesirable. Instead, it is assumed that an approximate adjoint solution $\tilde{\Psi}_h$ is available as a substitute. In a fashion analogous to (2.1), we define the adjoint perturbation $\delta\Psi_h$ by

$$\tilde{\Psi}_h \equiv \Psi_h + \delta\Psi_h. \tag{2.8}$$

Note that the perturbation in the adjoint is independent of the perturbation in the primal.

The adjoint residual operator is defined as,

$$R_h^{\Psi}(\cdot) \equiv \frac{\partial R_h}{\partial U_h}^T (\cdot) - \frac{\partial f_h}{\partial U_h}^T, \tag{2.9}$$

so that $R_h^{\Psi}(\Psi_h) = 0$, and

$$\begin{aligned} R_h^{\Psi}(\tilde{\Psi}_h) &= \frac{\partial R_h}{\partial U_h}^T (\Psi_h + \delta\Psi_h) - \frac{\partial f_h}{\partial U_h}^T, \\ &= \frac{\partial R_h}{\partial U_h}^T \delta\Psi_h. \end{aligned} \tag{2.10}$$

Using (2.7) – (2.10), the perturbation in the functional can be further expanded as

$$\begin{aligned} \delta f_h &\approx \Psi_h^T R_h(\tilde{U}_h), \\ &= \tilde{\Psi}_h^T R_h(\tilde{U}_h) - \delta\Psi_h^T R_h(\tilde{U}_h), \\ &\approx \tilde{\Psi}_h^T R_h(\tilde{U}_h) - \delta\Psi_h^T \frac{\partial R_h}{\partial U_h} \delta U_h, \\ &= \tilde{\Psi}_h^T R_h(\tilde{U}_h) - R_h^{\Psi}(\tilde{\Psi}_h)^T \delta U_h, \\ &\approx \tilde{\Psi}_h^T R_h(\tilde{U}_h) - R_h^{\Psi}(\tilde{\Psi}_h)^T \frac{\partial R_h}{\partial U_h}^{-1} R_h(\tilde{U}_h). \end{aligned} \tag{2.11}$$

The term, $\tilde{\Psi}_h^T R_h(\tilde{U}_h)$, is computable given the approximate solutions $\tilde{U}_h$ and $\tilde{\Psi}_h$. This term can be used to correct the perturbed functional yielding the following estimate for the fine-grid output:

$$f_h(U_h) \approx f_h(\tilde{U}_h) - \tilde{\Psi}_h^T R_h(\tilde{U}_h). \tag{2.12}$$

An asymptotic bound on the remaining error in the corrected functional is obtained from the last expression in (2.11) as,

$$\begin{aligned} \left|\delta f_h - \tilde{\Psi}_h^T R_h(\tilde{U}_h)\right| &\leq \left\|R_h^{\Psi}(\tilde{\Psi}_h)\right\| \left\|\frac{\partial R_h}{\partial U_h}^{-1}\right\| \left\|R_h(\tilde{U}_h)\right\| + \epsilon, \\ &= \frac{\left\|R_h^{\Psi}(\tilde{\Psi}_h)\right\| \left\|R_h(\tilde{U}_h)\right\|}{\sigma_{\min}} + \epsilon, \end{aligned} \tag{2.13}$$

where $\|\cdot\|$ denotes the 2-norm, $\epsilon \geq 0$ represents the magnitude of all higher-order terms, and $\sigma_{\min}$ is the smallest singular value of $\partial R_h / \partial U_h$ satisfying

$$\sigma_{\min} = \min_{z_h, \|z_h\|=1} \left\| \frac{\partial R_h}{\partial U_h} z_h \right\|. \tag{2.14}$$

Note that for linear functionals and residuals, $\epsilon \equiv 0$ and the bound in (2.13) is uniformly valid.

Discrete Adjoint

The functional correction procedure provides an estimate for the true value of the fine-grid output given approximations of the primal and adjoint solutions on that grid. In the present context, these approximations are obtained by prolongating coarse-grid solutions onto the fine grid using polynomial reconstruction (see Section 2.2). This imposes smoothness requirements on the discrete adjoint. For this reason, the primal residual operator $R_h(\cdot)$ is assumed to represent an integral statement such as one that would arise naturally from a finite element or finite volume formulation. A typical finite difference stencil would need to be scaled by an appropriate area term (or volume term in three dimensions) so that the residual became analogous to an integral expression. If the primal residual operator does not represent an integral quantity, the solution of the discrete adjoint equations in (2.6) will generally scale with the local grid size. Consequently, smoothness will be compromised on grids with irregular spacing. Furthermore, the discrete solution will no longer be consistent with the analytical adjoint [24].

Implementation Issues

Depending on the structure of the source code, the functional correction procedure can be implemented very rapidly into a flow solver that already computes the adjoint solution[2], particularly if the residual and functional subroutines are modular and easily accessible. Residual and functional evaluations on the fine grid can be performed by simply calling these subroutines with the fine grid coordinates and approximate solutions as input. This would require substantial memory overhead since the fine grid would need to be stored in its entirety. The alternative is to modify the machinery of the code so that residual evaluations on the fine grid can be achieved by assembling local contributions sequentially – in a loop through the coarse grid elements, for example. This would eliminate the memory overhead at the cost of increased coding work.

Pierce and Giles Correction Technique

The correction formula given by (2.12) is an algebraic analogue of the Pierce and Giles (P&G) correction term [26, 49]. The current derivation utilizes the notion of a fine grid and appeals directly to the nonlinear, discrete residual equations. The P&G derivation is cast in a continuous framework; errors are measured relative to the exact value of the functional corresponding to the solution of the PDEs under consideration. In the P&G approach, the vector inner product of the adjoint and the discrete residual in (2.12) is replaced by an integral inner product of the adjoint and the residual of the governing PDEs using approximate adjoint and primal solutions. Both the P&G correction technique and the algebraic version used in the present work have been shown to produce significant improvements in functional accuracy

[2] For example, the adjoint is used in several gradient-based aerodynamic optimization codes [21, 31, 32, 42].

(See the superconvergence results in references [26, 49, 62] and the improvements in functional accuracy in reference [63] and in Sections 3.6, 4.6, and Appendix A).

A convenience associated with the current method is that the error contribution from the domain boundary is automatically accounted for in the correction term given by (2.12). The reason for this is that the primal boundary conditions are already incorporated into the residual operator by construction. In the P&G technique, additional terms need to be accounted for explicitly whenever the approximate solutions do not satisfy inhomogeneous boundary conditions exactly [25, 26]. However, as will be seen in Section 4.4, some care must be taken in the current approach when primal boundary conditions are imposed in a strong sense.

A brief outline of the continuous adjoint correction theory of Pierce and Giles is presented in Appendix D.

2.2 Prolongation Operators

The approximate primal and adjoint solutions used in the functional correction procedure are obtained by prolongating coarse-grid solutions onto the fine grid. The prolongation operators take advantage of the fact that the fine grid elements are embedded within associated coarse-grid elements. Polynomial interpolants are constructed over each coarse-grid element and then used to inject values directly onto the embedded fine grid nodes within that coarse-grid element. Two different polynomial orders are used in the adaptive procedure: linear and quadratic.

2.2.1 Linear Operator

The linear operator L_h^H represents simple linear interpolation over each coarse-grid element. The resulting interpolant matches the data at the coarse-grid nodes exactly and is, therefore, continuous across element edges. This operator is used for computing the adaptive parameters described in Section 2.3.2.

2.2.2 Quadratic Operator

The quadratic prolongation operator Q_h^H is defined by a local least squares procedure. A quadratic profile is obtained over each coarse-grid element using a local error-minimization process involving the nodal values and slopes of the function being interpolated.

Consider a generic scalar function, ϕ, and its derivatives, ϕ_x and ϕ_y, sampled at the nodes of the coarse grid[3]. Over each coarse-grid element, k, we can establish the following linear distributions

$$\bar{\phi} = \sum_{i=1}^{3} N_i \phi^i, \tag{2.15}$$

$$\bar{\phi}_x = \sum_{i=1}^{3} N_i \phi_x^i, \tag{2.16}$$

$$\bar{\phi}_y = \sum_{i=1}^{3} N_i \phi_y^i, \tag{2.17}$$

where ϕ^i, ϕ_x^i, and ϕ_y^i are the sampled values at the three vertices of element k, and N_i are the standard shape functions for a linear triangle. A quadratic interpolant of ϕ over element k can be represented as

$$\tilde{\phi} = \sum_{j=1}^{6} \tilde{N}_j \tilde{\phi}_j, \tag{2.18}$$

where $\tilde{N}_j$ are the shape functions for a quadratic triangle [69] and $\tilde{\phi}_j$ are the unknown

[3] The function and its derivatives are assumed to have comparable accuracy at the nodes. To achieve this for a typical finite element or finite volume discretization, the gradients can be recovered from the numerical solution using gradient recovery techniques such as those described in references [3, 71]. The solver used for the flow simulations in this work is FUN2D [41]. Gradients at the nodes are recovered from the flow/adjoint solution using a local least squares procedure described in reference [3]. This algorithm computes nodal gradients as least squares averages of the gradients computed along each edge impinging upon that node.

coefficients to be determined. The least squares problem is to determine these coefficients by minimizing the $\mathcal{H}^1$ norm of the difference between the linear and quadratic profiles. That is,

$$\begin{aligned} \text{find} \quad & z^* = \{\tilde{\phi}_1^*, \tilde{\phi}_2^*, \ldots, \tilde{\phi}_6^*\}^T, \\ \text{such that} \quad & \Lambda(z^*) = \min_z \Lambda(z), \qquad\qquad (2.19) \\ \text{where} \quad & \Lambda(z) = \int_{\Omega_k} \left[\left(\tilde{\phi}(z) - \bar{\phi}\right)^2 + \left(\tilde{\phi}_x(z) - \bar{\phi}_x\right)^2 + \left(\tilde{\phi}_y(z) - \bar{\phi}_y\right)^2 \right] d\Omega. \end{aligned}$$

Once z^* has been determined for element k, it is substituted back into (2.18) yielding the desired interpolant. The procedure is repeated for all elements in the coarse grid.

This process produces a piecewise quadratic interpolant that is discontinuous across element edges. Values on edges are obtained by taking the arithmetic average of the interpolant values from the two adjacent elements associated with that edge. Values at coarse-grid nodes are similarly obtained by taking the arithmetic average of the interpolant values from the patch of elements surrounding that node.

The least squares problem defined by (2.19) can be solved efficiently by inverting the associated normal equations. Unfortunately, the normal equations become severely ill-conditioned on highly stretched elements resulting in numerical instabilities. To circumvent this issue, the least squares problem is solved using a singular value decomposition (SVD) algorithm. Solving the least squares problem via SVD is computationally more expensive, however, it provides for a highly stable algorithm [61].

The quadratic prolongation operator is used for evaluating the correction term in (2.12), and for computing the adaptive parameters described in Section 2.3.2.

2.2.3 Grid Movement

As described in Section 2.1.1, the fine grid is constructed in two steps. The first step is to uniformly refine the coarse grid by subdividing each triangle into a specified number of self-similar triangles. This results in a fine grid with boundary nodes that do not generally lie on the physical boundary. The second step is to reposition the boundary nodes, and possibly several layers of interior nodes, so that the fine grid conforms to the physical boundary.

The physical boundary is represented by cubic splines. Nonconforming boundary nodes are snapped to the nearest point on the cubic spline representation. The remaining fine–grid nodes within each coarse–grid boundary element are smoothed using the isotropic smoothing algorithm described in Appendix C. Figure 2-1 shows examples of two fine grids after their boundary nodes have been repositioned to conform to the physical boundary. In the case of the $N = 7$ fine grid, adjacent nodes have also been smoothed in order to improve grid quality.

For highly stretched grids, such as those used in the viscous test cases, repositioning of the boundary nodes may result in negative elements. If this occurs, the grid is *repaired* by moving adjacent interior nodes until consistency is restored. Laplace's equation is used to determine the $x-$ and $y-$ components of the interior–node displacements. A standard Galerkin discretization is used to approximate Laplace's equation on the nonconforming grid. The boundary–node displacements are used as Dirichlet conditions. The resulting systems of discrete equations (one for the x–component and one for the y–component of the displacement vector) are solved, approximately, yielding a consistent displacement field for the interior nodes. This is a simplified version of the node movement strategy used in reference [43].

Prolongation of the coarse–grid solution onto the fine grid is performed prior to the aforementioned node repositioning steps. The nodal values of the prolongated solution remain unchanged during the boundary conforming process. This can be viewed as a one-to-one mapping of the prolongated solution from the nonconforming grid to the boundary-conformed grid.

2.2.4 Strong Boundary Conditions

For problems where primal boundary conditions are imposed in a strong sense, some care must be taken when prolongating the coarse-grid adjoint onto the fine grid. The issue arises when Dirichlet conditions are imposed in a strong sense as is the case for the viscous test cases in Chapter 4. Modifications of the prolongation operators for the viscous adjoint are described in Section 4.4.3.

2.3 Adaptive Methodology

The grid adaptive procedure proposed in the present work is designed to compliment the functional correction procedure of Section 2.1.2. The goal of the adaptive algorithm is to improve the accuracy of the corrected output in (2.12) by reducing the remaining error after correction.

2.3.1 Remaining Error in the Functional

As shown in (2.11), the error in the functional can be expressed as the sum of two terms: a computable correction that can be evaluated given the approximate solutions $\tilde{U}_h$ and $\tilde{\Psi}_h$, and a remaining error term that generally cannot be evaluated without solving for quantities on the fine grid. The remaining error can be written in several different forms. Two forms that are particularly useful in the present context are,

$$\begin{aligned} \underbrace{\delta f_h}_{\text{Error}} - \underbrace{\tilde{\Psi}_h^T R_h(\tilde{U}_h)}_{\text{Correction}} &\approx \underbrace{-\delta\Psi_h^T R_h(\tilde{U}_h)}_{\text{Remaining Error}}, \\ &\approx \underbrace{-R_h^{\Psi}(\tilde{\Psi}_h)^T \delta U_h}_{\text{Remaining Error}}. \end{aligned} \tag{2.20}$$

The proposed adaptive strategy is based on reducing these two forms of the remaining error, thereby improving the accuracy of the computable correction. We see from (2.20) that the remaining error can be expressed as the inner product of the adjoint solution-error and the primal residual-error, or as the inner product of the adjoint residual-error and the primal solution-error. Neglecting nonlinear terms, these two inner products are equal. The magnitudes of their corresponding components are comparable (but generally not equal) and their units are identical to those of the functional $f(U)$. This illustrates the duality between the primal and adjoint residual operators. If nonlinear effects are accounted for, a duality gap, $D \neq 0$, will exist between the two inner products. By retaining nonlinear terms in the expansion of the residual operator in (2.5), one can obtain the following expression for the duality gap:

$$\begin{aligned} D &\equiv R_h^\Psi(\tilde{\Psi}_h)^T \delta U_h - \delta\Psi_h^T R_h(\tilde{U}_h), \\ &= \delta\Psi_h^T W, \end{aligned} \tag{2.21}$$

where W is a vector containing quadratic forms of the primal error. An explicit expression for W is given in Appendix B. The proposed adaptive procedure is based on reducing and equidistributing the magnitudes of the components of each of the inner products on the right-hand-side of (2.21). In addition to improving the quality of the computable correction, this will lead to a reduction in the magnitude of the duality gap, and hence, a reduction in the nonlinear contribution to the functional error.

The primal form of the remaining error, $\delta\Psi_h^T R_h(\tilde{U}_h)$, is essentially what is used by Becker and Rannacher [9, 10] in their finite-element output-based adaptive strategy. The addition of the dual term in the current approach is a natural way of incorporating the adjoint residual error into the adaptive scheme. Utilizing information from both the primal and adjoint residuals is expected to lead to a more robust adaptive algorithm.

Müller and Giles [40] have adopted a somewhat different philosophy in their adaptive strategy for functional outputs. Instead of using the remaining error in the functional as an adaptive sensor, they focus directly on the correction term $\tilde{\Psi}_h^T R_h(\tilde{U}_h)$. In principle, this leads to an adaptive algorithm for minimizing the magnitude of the correction term. However, if one is to preserve the role of this term as an accurate correction for the functional, it may not be advantageous to minimize its magnitude. Unlike the remaining error terms, the correction term is computable and generally approximates the leading order error in the functional to high accuracy (for example, see the functional convergence results in references [26, 49, 62, 63] and in Sections 3.6.2

and 4.6.1). The strategy in the present work, therefore, is to take advantage of the improved accuracy from correcting the functional, and to use the adaptive process to further enhance this accuracy by reducing the remaining error.

2.3.2 Adaptive Criteria and Parameters

The philosophy adopted for the current adaptive framework is to allow the user the freedom to specify control parameters that are most relevant within a practical engineering design context. One or more of these parameters will ultimately define the termination criteria for the adaptive procedure. The parameters that may be prescribed *a priori* in the current strategy include: 1) a desired remaining error level for the functional of interest; 2) an upper limit on the total number of degrees of freedom in the simulation; 3) an upper limit on the total computing time; and 4) an upper limit on the total number of adaptive iterations. The anisotropic adaptation procedure used for the viscous test cases requires additional information to be specified as discussed in Section 4.5.4.

In the proposed adaptive strategy, we seek to reduce and equidistribute the value of an adaptation parameter throughout the computational domain while simultaneously monitoring and reducing an upper bound on the estimated remaining error in the functional of interest. Consider the operation of computing an inner product over the fine grid, Ω_h, embedded within Ω_H. For each coarse-grid element, k, there are N^2 (in two dimensions) fine-grid elements over which a partial inner product must be computed. For each fine-grid node, $l(k)$, within element k, there are 4 subcomponents (for the two-dimensional Euler and Navier-Stokes equations) to the primal and adjoint residual vectors corresponding to the mass, x-momentum, y-momentum and energy conservation equations. Equation (2.21) suggests the following definition for the adaptation parameter, ε_k, at element k:

$$\begin{aligned}\varepsilon_k = \frac{1}{2}\sum_{l(k)}\{ \quad & \left| \, [R_h^{\Psi}(L_h^H \Psi_H)]_{l(k)}^T \, [Q_h^H U_H - L_h^H U_H]_{l(k)} \, \right| \\ + \quad & \left| \, [Q_h^H \Psi_H - L_h^H \Psi_H]_{l(k)}^T \, [R_h(L_h^H U_H)]_{l(k)} \, \right| \quad \}. \end{aligned} \tag{2.22}$$

In this last expression, a term of the form, $[V_h]_{l(k)}$, for some generic vector, V_h, on Ω_h, refers to the 4×1 subvector (component) of V_h corresponding to the fine-grid node, $l(k)$, within the coarse-grid element, k. The summation in (2.22) is over all fine-grid nodes within the k^{th} coarse element. For those nodes on the boundary of k, a partial contribution may be implied depending on how the nodal residual is defined for the particular discretization. For example, in the case of a vertex-based finite volume discretization, the fractional contribution from a node on the boundary of element k is proportional to the fraction of the associated control volume within that element. The operators L_h^H and Q_h^H are prolongation operators that map coarse-grid vectors onto the fine grid via linear and quadratic interpolation, respectively. These prolongation operators are defined in Section 2.2.

The adaptation parameter in (2.22) is a crude approximation of the magnitudes of the primal and dual forms of the remaining error in the functional. The intention here is not to obtain a quantitative estimate of these terms, but rather to establish an indication of how they are distributed throughout the domain. The residual operators in the expression for the adaptation parameter are evaluated using linear prolongations of the respective coarse-grid solutions. Linear prolongation is chosen over quadratic in this case because the magnitudes of the residuals tend to be larger, giving a more conservative bound on the remaining error terms. The perturbations δU_h and $\delta\Psi_h$ in (2.21) are replaced by measures of the local interpolation error in the primal and adjoint solutions, respectively. In particular, setting $\delta U_h \approx Q_h^H U_H - L_h^H U_H$ and $\delta\Psi_h \approx Q_h^H \Psi_H - L_h^H \Psi_H$ amounts to the assumption that the dominant component of the local error in the primal and adjoint solutions is characterized by the interpolation error. This does not preclude the notion that the *cause* of these local errors may ultimately be attributed to residual errors in remote parts of the domain.

Let ε denote the summation of ε_k over all elements in Ω_H. That is,

$$\varepsilon = \sum_k \varepsilon_k. \tag{2.23}$$

With the presence of the absolute value signs in (2.22), ε represents an upper bound on the estimated remaining error in the functional. In light of this, a global adaptation parameter can be defined as

$$\eta_g = \frac{\varepsilon}{e_o}, \tag{2.24}$$

where e_o is a user-specified desired error level. The global error criterion is satisfied if $\eta_g \leq 1$ while further refinement is required if $\eta_g > 1$. Using this measure alone would lead to uniform grid-refinement only. To complete the h-refinement strategy a local adaptation parameter must also be defined. An appropriate local error measure can be determined by invoking the principle of error equidistribution [6, 69]. In particular, an attempt is made to equidistribute ε_k over all elements in the domain. In this vein, a local adaptation parameter is defined as

$$\eta_k = \frac{\varepsilon_k}{\bar{e}_o}, \tag{2.25}$$

where $\bar{e}_o = e_o/N_e$ is the target error for each element and N_e is the total number of elements in the current grid, Ω_H. The local criterion is satisfied if $\eta_k \leq 1$ while further refinement of element k is indicated if $\eta_k > 1$. Finally, at each adaptive iteration, a new desired element size, H'_k is computed from the old one, H_k, according to

$$H'_k = H_k \left(\frac{1}{\eta_g \eta_k} \right)^{\omega}. \tag{2.26}$$

The underrelaxation parameter ω controls how aggressively each subsequent grid is refined during the iterative adaptive process. If the refinement procedure does not allow for coarsening, an inappropriately large choice for ω would lead to over–refinement, resulting in a suboptimal final grid with more elements than is necessary for the prescribed level of error. If coarsening is allowed, then a large value of ω would lead to oscillatory grid convergence, whereby the grid is continually over– and under–refined in an alternating fashion. Conversely, an inappropriately small value for ω would prolong the adaptive process by increasing the total number of adaptive iterations to convergence. An appropriate value for ω can be deduced by examining the asymptotic convergence rates of the global and local adaptation parameters. Numerical results indicate (see Section 3.6.2) that $\eta_g \sim \eta_k \sim \mathcal{O}(\bar{H}^p)$ with $2 \leq p \leq 3$ where $\bar{H}$ is an average element size in the grid. These results are obtained by monitoring the convergence of η_g and η_k on a series of uniformly refined grids for an inviscid, subsonic test case without geometric irregularities. A heuristic for determining ω is to render the new element size independent of its current size; that is, to choose ω such that $H_k^\prime \sim \mathcal{O}(1)$. In the present case we assume a convergence of $\eta_g \eta_k \sim \mathcal{O}(H_k^4)$ yielding a heuristic value of $\omega = 1/4$ which is used in all the output-based adaptive simulations in this dissertation. This simple convergence rule may not be valid near singularities, discontinuities or geometric irregularities. Nevertheless, the adaptive procedure exhibits robust grid-convergence, usually terminating after 3–4 adaptive iterations regardless of the type of refinement algorithm employed. Reference [44] provides further discussion on grid convergence for adaptive methods.

Two types of adaptive refinement are used in this work. A standard isotropic h–refinement strategy without coarsening is employed for the inviscid problems of Chapters 3. For the viscous simulations of Chapter 4, the BL2D [12] anisotropic grid generator is employed to regenerate the grids at each adaptive iteration. The proposed anisotropic grid–adaptive procedure combines Hessian–based adaptation with the output-based criteria of this section.

Chapter 3

Inviscid Flow

The output–based adaptive methodology presented in Chapter 2 is applied to a finite volume discretization of the Euler equations. The proposed method is demonstrated on a series of subsonic, transonic, and supersonic test cases involving various airfoil configurations. The outputs considered are the lift, drag and moment coefficients. An isotropic h–refinement algorithm is used to improve the grids during the adaptive simulations. A commonly used adaptive indicator based on the local flow curvature [6, 15, 64] is implemented to comparatively assess the performance of the proposed output–based adaptive procedure. The curvature–based method often fails to terminate or produces incorrect values for the functional at convergence. In all test cases, the output–based method provides superior accuracy and succeeds in terminating once the specified error tolerance has been achieved for the chosen functional.

3.1 Governing Equations

The Euler equations are expressions of the principles of mass, momentum, and energy conservation for an inviscid, compressible gas. In two dimensions these equations may be written as

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=0, \tag{3.1}$$

where the conservative variables U, and inviscid fluxes F and G, are given by

$$U=\left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho E\end{array}\right\},\quad F=\left\{\begin{array}{c}\rho u\\ \rho u^2+p\\ \rho uv\\ u(\rho E+p)\end{array}\right\},\quad G=\left\{\begin{array}{c}\rho v\\ \rho uv\\ \rho v^2+p\\ v(\rho E+p)\end{array}\right\}. \tag{3.2}$$

In these expressions, ρ is the mass density; u and v are, respectively, the $x-$ and $y-$components of the gas velocity; p is the static pressure and E is the total energy. The system is closed with the equation of state for an ideal gas:

$$p=\rho(\gamma-1)\left(E-\frac{u^2+v^2}{2}\right), \tag{3.3}$$

where γ is the ratio of specific heats. The working gas is assumed to be ideal air with specific heat ratio $\gamma=1.4$

Attention is focused on steady solutions of the Euler equation system specified by (3.1)-(3.3).

3.2 Aerodynamic Forces

The functionals chosen for the present study are those of common interest in aeronautical applications: the lift, drag and moment coefficients. These outputs are all expressible as surface integrals over parts of the domain boundary. Let $\partial\Omega'$ denote that portion of the continuum boundary over which aerodynamic forces are to be computed, such as the surface of an airfoil or the walls of a channel. The resultant force $\vec{F}_R$ acting on $\partial\Omega'$ due to the inviscid flow is given by

$$\vec{F}_R \equiv \int_{\partial\Omega'} p\,\hat{\mathbf{n}}\,ds, \tag{3.4}$$

where $\hat{\mathbf{n}}$ is the unit normal vector on $\partial\Omega'$. The lift L and drag D are defined as the components of $\vec{F}_R$ perpendicular and parallel, respectively, to the free-stream velocity, $\vec{V}_\infty$:

$$L \equiv \left|\vec{F}_R\right| \sin\theta, \tag{3.5}$$

$$D \equiv \left|\vec{F}_R\right| \cos\theta, \tag{3.6}$$

where θ is the angle between $\vec{V}_\infty$ and $\vec{F}_R$. In two dimensions, an implicit definition for θ that accounts for the proper sign convention is

$$\sin\theta \equiv \left(\frac{\vec{V}_\infty}{\left|\vec{V}_\infty\right|} \times \frac{\vec{F}_R}{\left|\vec{F}_R\right|}\right) \cdot \hat{k},$$

where $\hat{k}$ is the unit vector pointing out of the page. The total moment exerted on $\partial\Omega'$ about $\vec{r}_0 = (x_0, y_0)$ is given by

$$M \equiv \int_{\partial\Omega'} (\vec{r} - \vec{r}_0) \times p\,\hat{\mathbf{n}}\,ds \cdot \hat{k}, \tag{3.7}$$

where $\vec{r} = (x, y)$ is the position vector measured with respect to the origin. Finally, the lift, drag and moment coefficients are defined, respectively, as

$$C_L \equiv \frac{L}{q_\infty c}, \tag{3.8}$$

$$C_D \equiv \frac{D}{q_\infty c}, \tag{3.9}$$

$$C_M \equiv \frac{M}{q_\infty c^2}, \tag{3.10}$$

where $q_\infty = \rho_\infty V_\infty^2/2$ is the free-stream dynamic pressure and c is a reference length such as the airfoil chord length.

3.3 Flow Solver

The base solver used for the inviscid and viscous simulations in this investigation is FUN2D: a fully implicit, unstructured grid, vertex-based, finite volume code originally developed by Dr. W. K. Anderson, formerly of NASA Langley [41]. The flow solver has available several different methods for evaluating fluxes at cell interfaces. The Osher upwind scheme [17] is used for the inviscid test cases in the present study. A backward-Euler time-stepping method is used to drive the solution to a steady state. An approximate solution of the linear system of equations formed at each time step is obtained using several iterations of a point-iterative Gauss-Seidel-type method. Boundary conditions are imposed weakly in the inviscid calculations. Flow–tangency conditions are enforced by setting $\vec{V} \cdot \hat{\mathbf{n}} = 0$ in the flux contributions from solid–wall boundary segments. Farfield fluxes are evaluated using characteristic reconstructions. Further details can be found in reference [3].

3.4 Adjoint Solver

After each flow (primal) solution in an adaptive run, the coarse-grid adjoint Ψ_H is obtained by solving the system,

$$\frac{\partial R_H}{\partial U_H}^T \Psi_H = \frac{\partial f_H}{\partial U_H}^T . \tag{3.11}$$

Although this system can be solved directly using GMRES [54], a time-like derivative is added and the solution is obtained by marching in time, much like the flow solver:

$$\left\{\frac{A_H}{\Delta t} + \left[\frac{\partial R_H}{\partial U_H}\right]^T\right\}(\Delta\Psi_H)^n = \left(\frac{\partial f_H}{\partial U_H}\right)^T - \left[\frac{\partial R_H}{\partial U_H}\right]^T (\Psi_H)^n, \tag{3.12}$$

where $(\Psi_H)^{n+1} = (\Psi_H)^n + (\Delta\Psi_H)^n$ and A_H is a diagonal matrix containing the control-volume areas associated with each node in Ω_H. The time term can be used to increase the diagonal dominance of the system for cases in which GMRES alone would tend to stall. Generally, this results in a more robust adjoint solver. Preconditioning is achieved using a point-iterative scheme similar to that used in the flow solver. Further details concerning the adjoint solver can be found in the references [42, 43].

3.5 Grid Refinement

The original grids (prior to adaptive refinement) are generated using AFLR2: an advancing-front/local-reconnection, unstructured, triangulator developed by Dr. D. Marcum at Mississippi State University [39]. During the adaptive runs, a Delaunay node insertion and local retriangulation algorithm written by Shewchuk [55] is used for refining the grids. The algorithm attempts to satisfy a prescribed element size distribution such as the one implied by (2.26). If more then 50% of the elements in the current grid are flagged for refinement, then the desired element size distribution obtained from (2.26) is scaled so that no more than half of the elements are refined from one adaptive iteration to the next. This measure may be triggered in the early stages of an adaptive run if the prescribed error level for the functional is very low relative to the resolution of the initial grid.

Grid coarsening is not performed for the inviscid test cases in this study. Requests for larger element sizes are simply ignored by the refinement algorithm. Coarsening could be introduced into the algorithm without modification to the proposed adaptive strategy. Various grid quality measures [15] are incorporated as postprocessing steps after the initial Delaunay refinement: edge swapping is used to reduce the maximum node degree in the triangulation to 7, and a node removal algorithm is used to remove nodes of degree lower than 5. All remaining nodes that have been affected by these measures are then smoothed using several iterations of a local least–squares procedure described in Appendix C.

3.6 Numerical Results

Several inviscid test cases are presented to demonstrate the effectiveness of the proposed output–based grid adaptive procedure. Comparisons are made with a commonly used adaptive method that employs a curvature sensor based on interpolation–error estimates [6, 15, 64]. The proposed output-based scheme is shown to compare favorably in terms of accuracy, efficiency and reliability relative to the curvature-based scheme.

3.6.1 Curvature–Based Adaptive Method

A curvature–based adaptive method is implemented for the purpose of comparatively assessing the performance of the proposed output-based method. The adaptive indicator is based on the scalar sum of the local magnitudes of the second derivatives in either the computed pressure or Mach number scaled with H_k^r where H_k is the local element size and $r = 2$ or 3 [6, 15, 64]. The elemental second derivatives are estimated using the quadratic reconstruction procedure described in Section 2.2.2. The adaptive criterion requires that the value of the indicator be less than some predetermined threshold value for all elements. In the absence of any rigorous relationship to the functional error, the threshold values used in this work are chosen heuristically depending on the field quantity being adapted on and the value of r. While the threshold value influences the extent to which the grid is refined, it does not change the general refinement pattern of the final grid. Hence meaningful comparisons with the output-based adaptive patterns are possible. If the adaptive indicator for an element is greater than the predetermined threshold value, the element is flagged for refinement and the desired new element size is set to half of its original size. Grids are

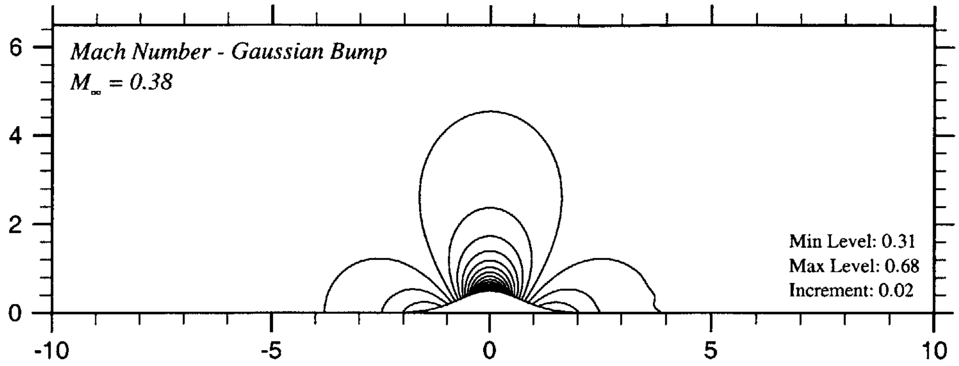


Figure 3-1: Gaussian bump test case: $M_\infty = 0.38$, $\alpha = 0^o$. Computed Mach number distribution.

refined using the Shewchuk [55] algorithm supplemented with the same grid quality measures described in Section 3.5.

3.6.2 Functional Correction Results

Subsonic flow in a channel of height 10 and length 20 is simulated with an inlet Mach number of $M_\infty = 0.38$. The shape of the lower wall is a Gaussian of amplitude 0.5 centered at the mid-length of the channel. The purpose of this test case is to examine convergence rates of various quantities associated with the base solver and functional correction procedure. No adaptation is performed for this test case. A hierarchy of 8 embedded grids is constructed for the purposes of conducting the convergence tests. These grids are generated by subdividing each of the edges in the coarsest grid into n equal segments for values of $n = 2\,,3\,,\ldots\,,8$. This corresponds to subdividing each triangle in the coarsest grid into n^2 self–similar triangles to create the nth grid. Boundary edges are made to conform to the channel geometry as described in Sections 2.1.1 and 2.2.3. Figure 3-1 shows a plot of the computed Mach number distribution on the finest grid. Three functionals are considered: the coefficients of drag, lift, and moment acting on the walls of the duct. The moment is computed with respect to the origin. Figure 3-2 shows plots of the relative error in the functionals $|\delta f_h| = \left|f_h(Q_h^H U_H) - f_h(U_h)\right|$ and the remaining error after correction $\left|\delta f_h - (Q_h^H \Psi_H)^T R_h(Q_h^H U_H)\right|$. Errors are measured with respect to fine–grid values corresponding to $(N = 2)$ (see Section 2.1.1). These quantities are plotted versus

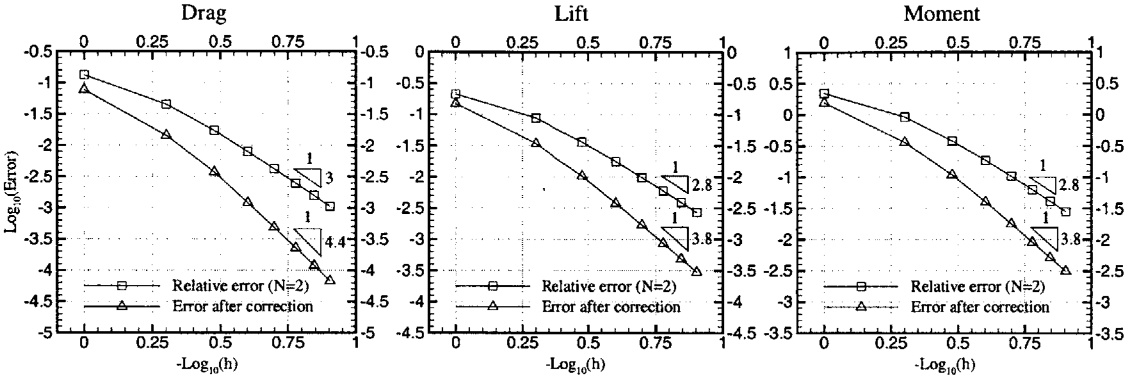


Figure 3-2: Gaussian bump test case: $M_\infty = 0.38$, $\alpha = 0^o$. Convergence plots of the error in the functional and the remaining error after correction. Errors are measured with respect to fine-grid values corresponding to $N = 2$.

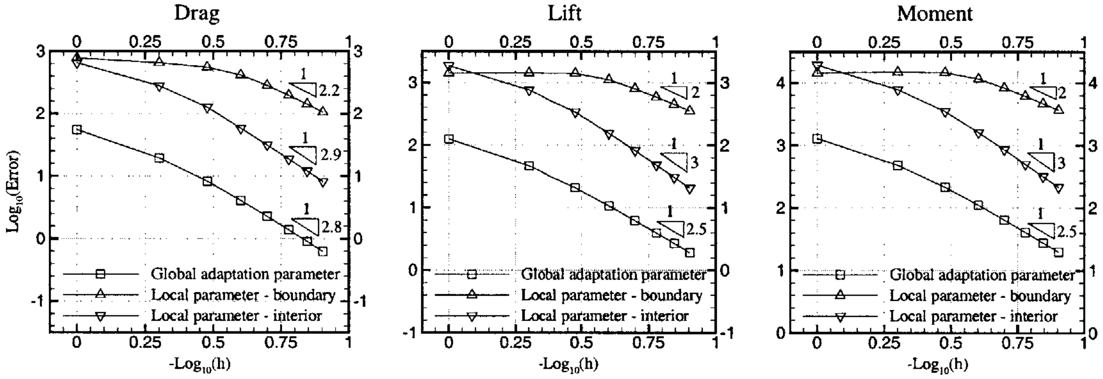


Figure 3-3: Gaussian bump test case: $M_\infty = 0.38$, $\alpha = 0^o$. Convergence plots of the global and local adaptation parameters as defined in (2.24) and (2.25), respectively.

a characteristic length parameter h associated with each grid in the hierarchy. A value of $h_1 = 1$ is arbitrarily assigned to the coarsest grid. The length parameter associated with the nth grid is given by $h_n = h_1/n$. The drag converges like $\mathcal{O}(h^3)$, whereas the lift and moment coefficients appear to converge at a slightly lower rate. In all cases, the remaining error after correction is observed to converge at a faster rate than the uncorrected error, exhibiting approximately $\mathcal{O}(h^4)$ convergence. Figure 3-3 shows convergence plots of the global and local adaptation parameters defined in (2.24) and (2.25), respectively. The local adaptation parameter η_k is evaluated over an interior element located approximately 0.5 units above the peak of the Gaussian bump, and over a boundary element bordering the peak. The local adaptation parameter on the boundary is observed to converge essentially as $\mathcal{O}(h^2)$, one order slower than in the interior. Even slower convergence would be expected near singularities, discontinuities or geometric irregularities such as sharp corners. The global adaptation parameter exhibits approximately $\mathcal{O}(h^{2.5})$ convergence. The convergence rates of these parameters are used as a guide in determining an appropriate value for the underrelaxation parameter ω that appears in (2.26) (see Section 2.3.2).

3.6.3 $M_\infty = 0.26$, $\alpha = 8^o$ Advanced Energy Efficient Transport (EET) Three-Element Airfoil

Adaptive simulations of subsonic flow past the Advanced Energy Efficient Transport (EET) three-element airfoil [36] is performed for $M_\infty = 0.26$ and $\alpha = 8^o$. This test case is representative of a high–lift wing configuration for take–off or landing. The functional of interest is the lift coefficient for the entire airfoil (i.e. all three elements). Figure 3-4 shows the computed Mach number contours and streamlines for this test case. Results from the output-based adaptive simulation are shown in Figure 3-5 for a specified error level of $e_o = 0.05$. For comparison, the lift convergence on a series of uniformly refined grids is also plotted (lower-most curve). These grids are generated by subdividing each of the edges in the original grid into n equal segments for values of $n = 2, 3, \ldots, 5$. The original grid in the hierarchy of uniformly refined grids is identical to the starting grid in the adaptive run. The extrapolated value in the figure is obtained using a Richardson extrapolation of the three finest, uniformly refined grids. The output–based algorithm surpasses the requested error level and terminates at a final grid of less than 9,000 nodes. In comparison, the uniformly refined grids achieve the same level of accuracy at approximately 70,000 nodes.

Performance of an interpolation–error–based adaptive indicator is compared with that of the proposed output-based approach. Implementation details are given in Section 3.6.1 and reference [64]. Figure 3-6 shows adaptive results using a curvature sensor applied to the static pressure field. The adaptive indicator is based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the

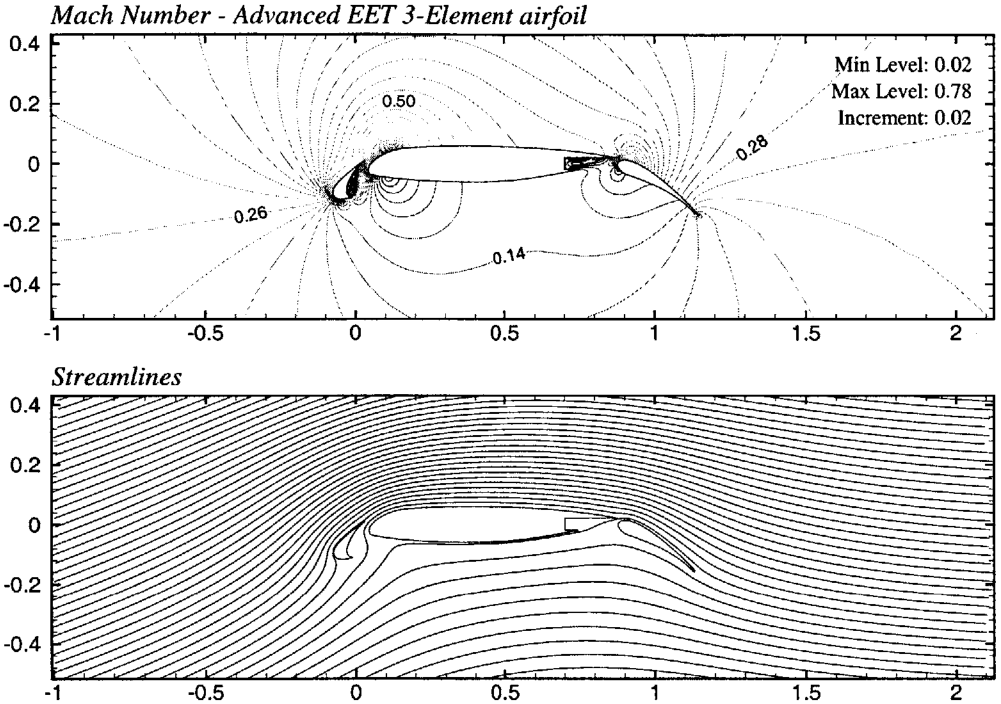


Figure 3-4: Advanced EET 3-element airfoil test case: $M_\infty = 0.26$, $\alpha = 8^o$. Computed Mach number distribution and streamlines.

local element size. Even though the flow field is smoothly varying (i.e. no shocks), the (uncorrected) lift appears to be converging to an erroneous value and never achieves the level of accuracy of the final output-based grid. Note that the error estimation procedure dramatically improves the lift error but still does not quite match the apparent asymptotic lift value of the uniform-grid results. Based on visual inspections of the grids in Figures 3-5 and 3-6, we attribute the lack of convergence in the lift using the pressure-based scheme to a lack of grid resolution in the following regions: the rear portion of the suction side of the main element, the pressure side of the main element, and the pressure side of the flap. During the last 4 pressure-based adaptive iterations there is little or no grid refinement in these regions. Most of the nodes are being added near the geometric irregularities of the three elements (i.e. at the sharp corners).

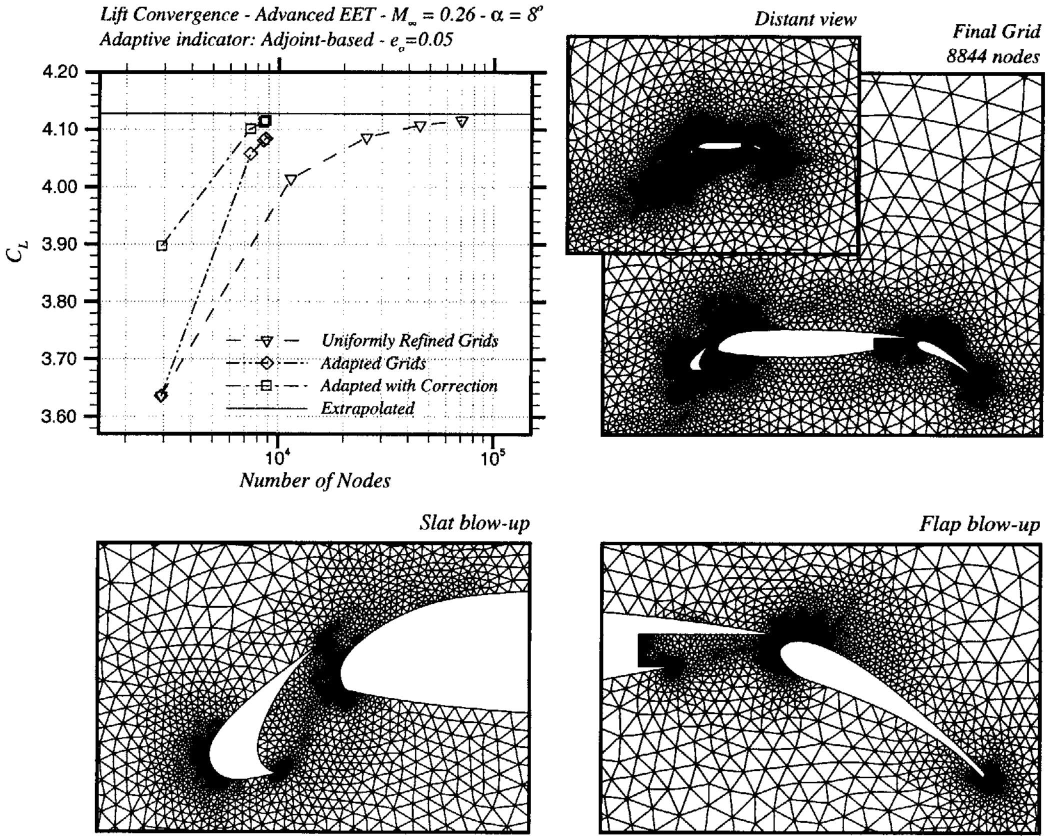


Figure 3-5: Advanced EET 3-element airfoil test case: $M_\infty = 0.26$, $\alpha = 8^o$. Upper left: error convergence in the computed lift during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 0.05$ is prescribed. Upper right: final adapted grid. Lower: blow-up near slat and flap.

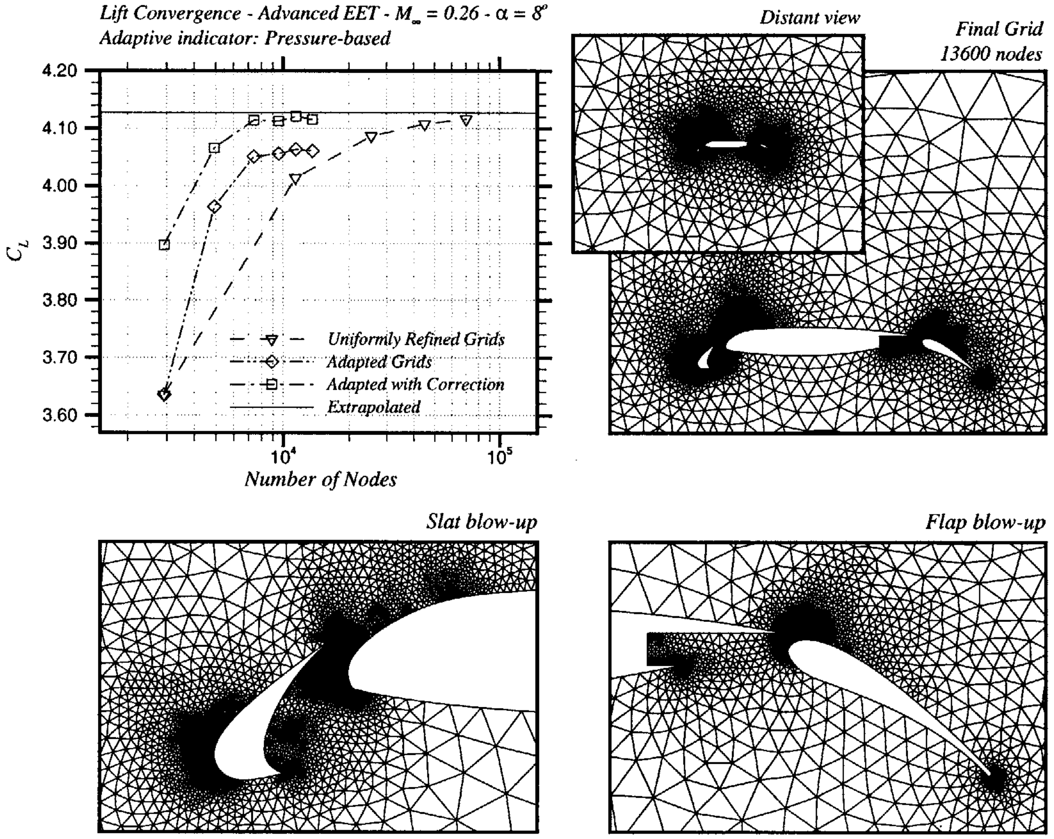


Figure 3-6: Advanced EET 3-element airfoil test case: $M_\infty = 0.26, \alpha = 8^o$. Upper left: error convergence in the computed lift during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Upper right: final adapted grid. Lower: blow-up near slat and flap

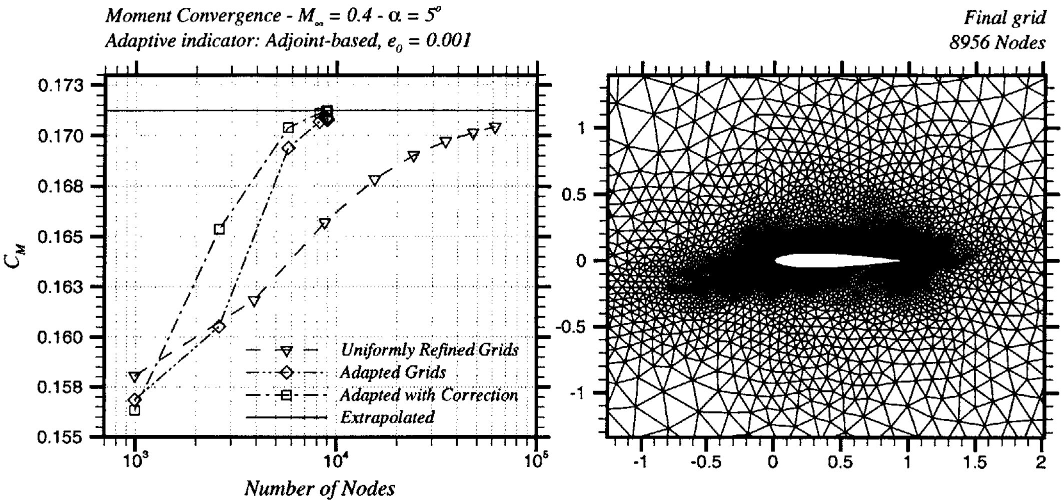


Figure 3-7: NACA 0012 test case: $M_\infty = 0.4$, $\alpha = 5^o$. Left: error convergence in the computed leading-edge moment during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 0.001$ is prescribed. Right: final adapted grid.

3.6.4 $M_\infty = 0.4$, $\alpha = 5^o$ NACA 0012 Airfoil

Subsonic flow past a NACA 0012 airfoil is simulated with a free-stream Mach number of $M_\infty = 0.4$, and angle of attack, $\alpha = 5^o$. The functional of interest is chosen as the moment coefficient taken about the leading edge. In this test case, proper flow resolution on the suction side of the airfoil near the leading edge is particularly important for predicting the aerodynamic forces accurately since the flow accelerates rapidly in this region and essentially sets the pressure distribution over the rest of the airfoil. In the case of the moment coefficient, however, resolution further downstream is also important due to the increasing influence of the moment arm on the magnitude of the output as one proceeds away from the leading edge.

The output–based adaptive results are shown in Figure 3-7. Convergence of the moment to the desired error tolerance of $e_0 = 0.001$ is achieved in three to four adaptive iterations with a final converged grid containing approximately 9,000 nodes. The output–based algorithm tends to refine the grid along the entire length of the airfoil as anticipated. Pressure-based adaptive results are shown in Figure 3-8. The computed moment coefficient fluctuates erratically during the pressure-based adaptive process and does not appear to be converging to any particular value. Furthermore,

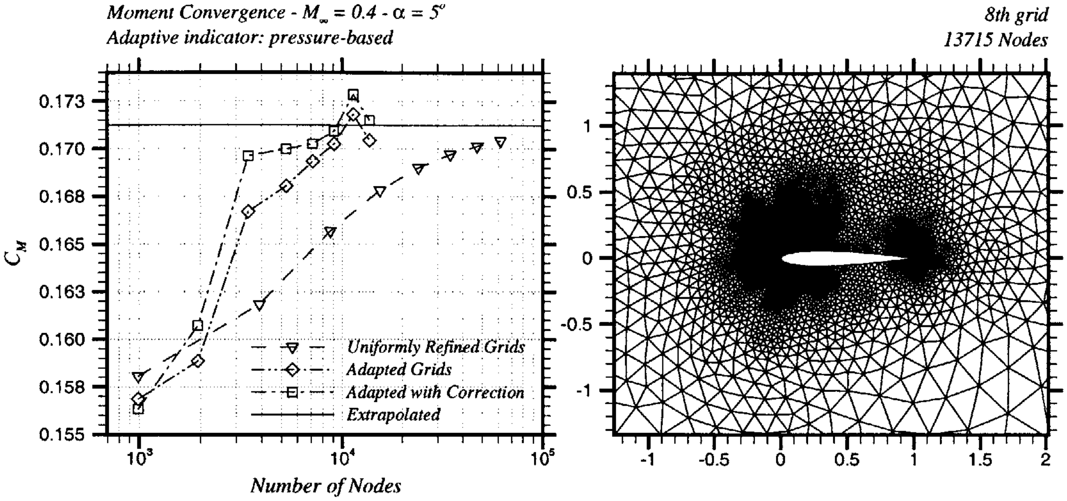


Figure 3-8: NACA 0012 test case: $M_\infty = 0.4$, $\alpha = 5^o$. Left: error convergence in the computed leading-edge moment during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Right: final adapted grid.

the adaptive process does not show signs of terminating after the eighth iteration. In contrast, the adjoint-based adaptive scheme provides a stable, monotonic convergence of the moment to the extrapolated value and self-terminates once the requested error level has been surpassed. The erratic convergence and poorer accuracy of the pressure-based scheme is primarily attributed to a lack of grid resolution on both the upper and lower airfoil surfaces downstream of the mid–chord position.

3.6.5 $M_\infty = 0.8$, $\alpha = 1.25^o$ NACA 0012 Airfoil

Transonic flow past a NACA 0012 airfoil is simulated with a free-stream Mach number of $M_\infty = 0.8$, and angle of attack, $\alpha = 1.25^o$. This test case is representative of cruise conditions for a commercial airliner. The flow exhibits a strong shock on the upper surface of the airfoil and a relatively weaker shock on the lower surface. The functional of interest is chosen to be the lift coefficient. Output-based adaptive results are shown in Figures 3-9 and 3-10 for prescribed error levels of $e_o = 5\%\ C_L$ and $e_o = 2.5\%\ C_L$, respectively. Pressure-based adaptive results are presented in Figure 3-11. The lift converges poorly on the uniformly refined grids for this test case. This is attributed

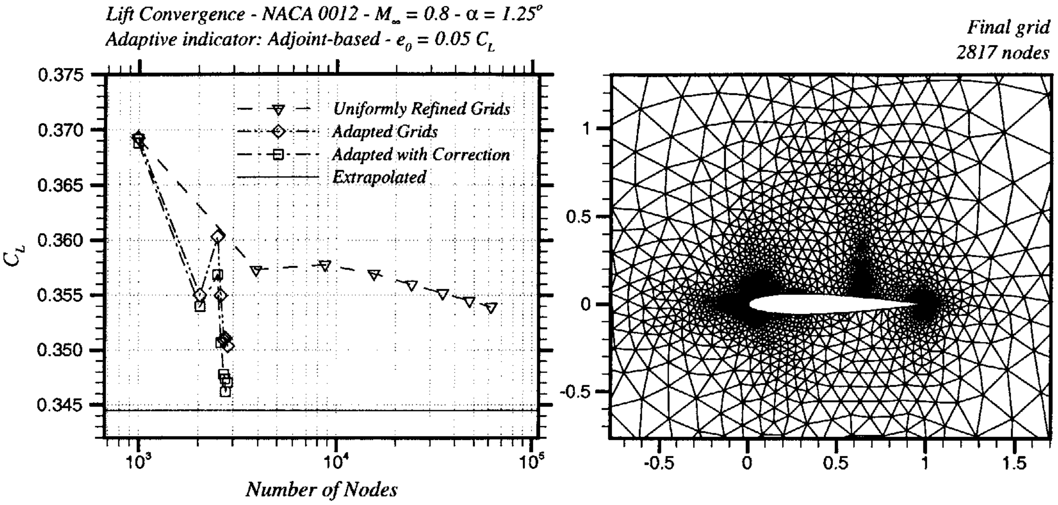


Figure 3-9: NACA 0012 test case: $M_\infty = 0.8$, $\alpha = 1.25^o$. Left: error convergence in the computed lift coefficient during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 5\%\ C_L$ is prescribed. Right: final adapted grid.

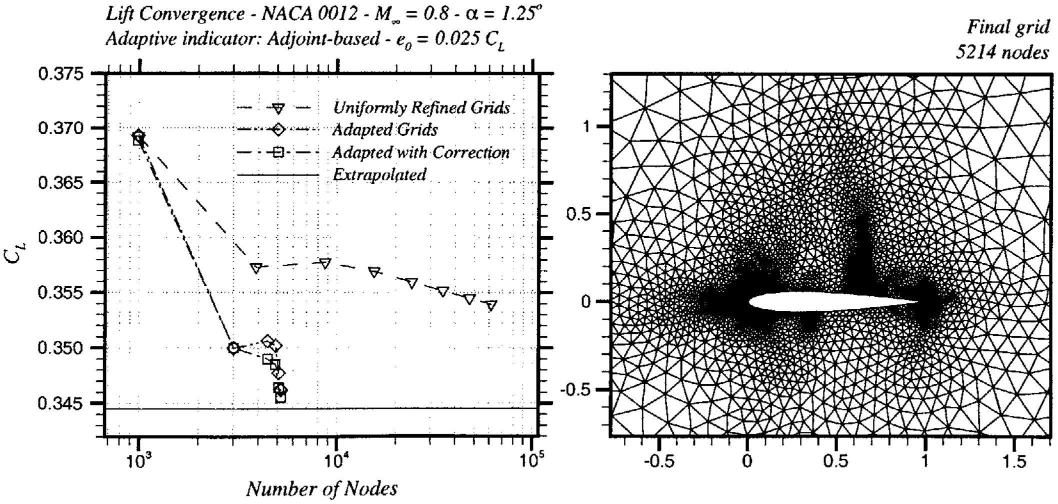


Figure 3-10: NACA 0012 test case: $M_\infty = 0.8$, $\alpha = 1.25^o$. Left: error convergence in the computed lift coefficient during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 2.5\%\ C_L$ is prescribed. Right: final adapted grid.

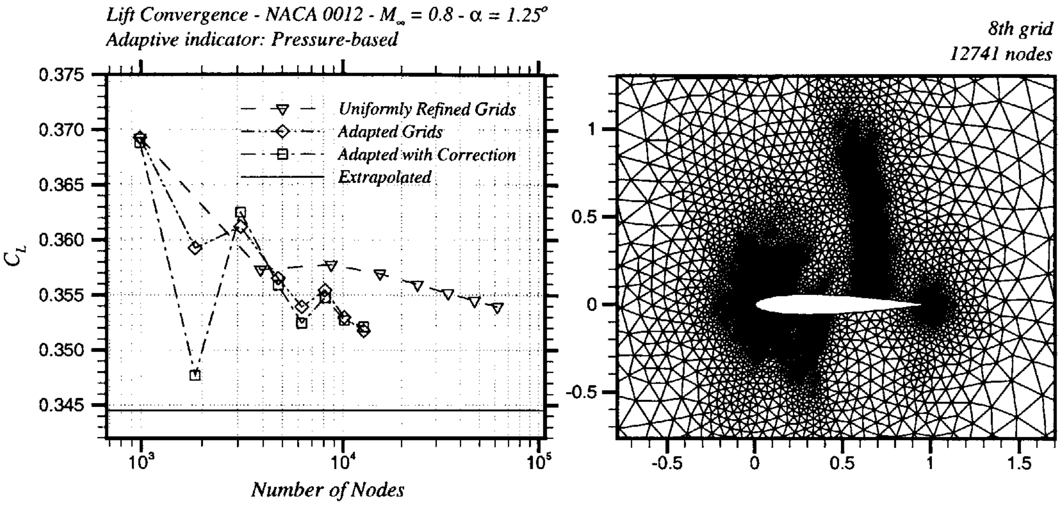


Figure 3-11: NACA 0012 test case: $M_\infty = 0.8$, $\alpha = 1.25^o$. Left: error convergence in the computed lift coefficient during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Right: final adapted grid.

to inadequate grid resolution in the trailing edge region of the seed grid. Note that this seed grid is also the starting grid in each of the adaptive runs. For this test case only, the extrapolated lift value in the convergence plots of Figures 3-9 - 3-11 is based on a different set of uniformly refined grids. The alternate seed grid is chosen as the final adapted grid in the $e_o = 2.5\%\ C_L$ output-based run; it contains 5214 nodes and is shown in Figure 3-10. This grid is globally refined twice yielding two additional fine grids corresponding to $n = 2$ and $n = 4$. The $n = 4$ grid contains 81840 nodes. The final lift value is obtained by performing a Richardson extrapolation on the three grids in the series. The implied convergence rate from the extrapolation is first order.

It is apparent from Figures 3-9 and 3-10 that the output-based algorithm terminated well within the prescribed error limits for the lift. Note that the weak shock on the lower surface is only slightly refined in the $e_o = 5\%\ C_L$ case. Evidently, refining the lower shock is not as important as refining the leading edge, upper shock region, and trailing edge for the given error tolerance. When the tolerance is lowered to $e_o = 2.5\%\ C_L$, however, it becomes beneficial to resolve the weaker shock as well.

We see from Figure 3-11 that the pressure-based adaptive method tends to refine

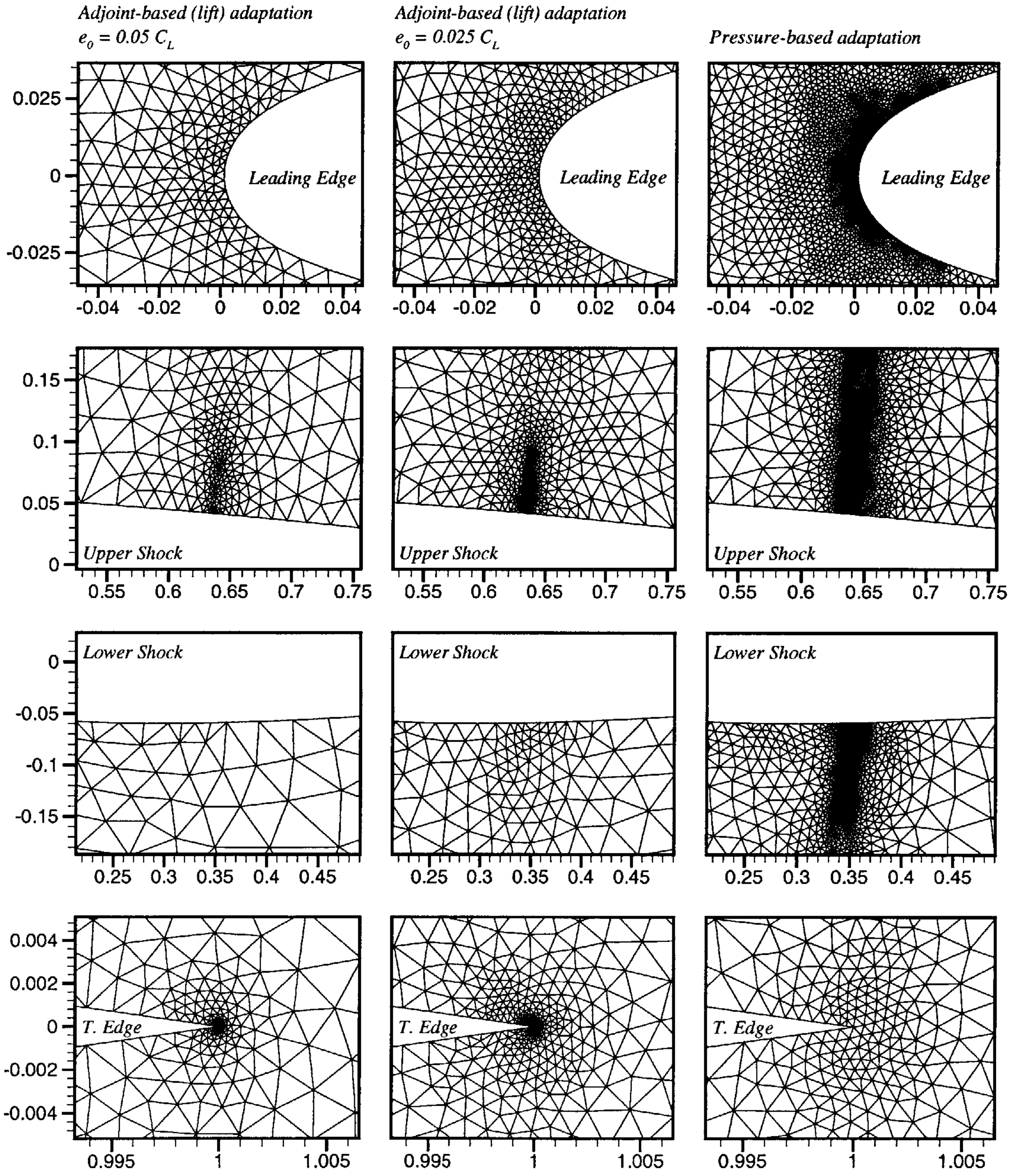


Figure 3-12: NACA 0012 test case: $M_\infty = 0.8$, $\alpha = 1.25^o$. Comparison of final adapted grids using the proposed output-based method with $e_o = 5\%\ C_L$ (left), $e_o = 2.5\%\ C_L$ (middle), and the pressure-based method (right). Top row: blow-up of leading edge region; 2nd row: shock region on suction side; 3rd row: shock region on pressure side; bottom row: trailing edge region.

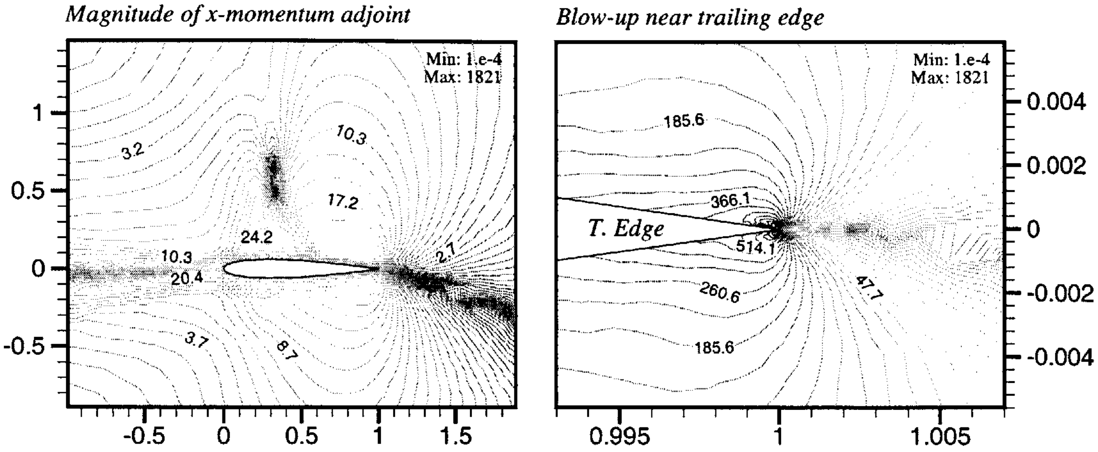
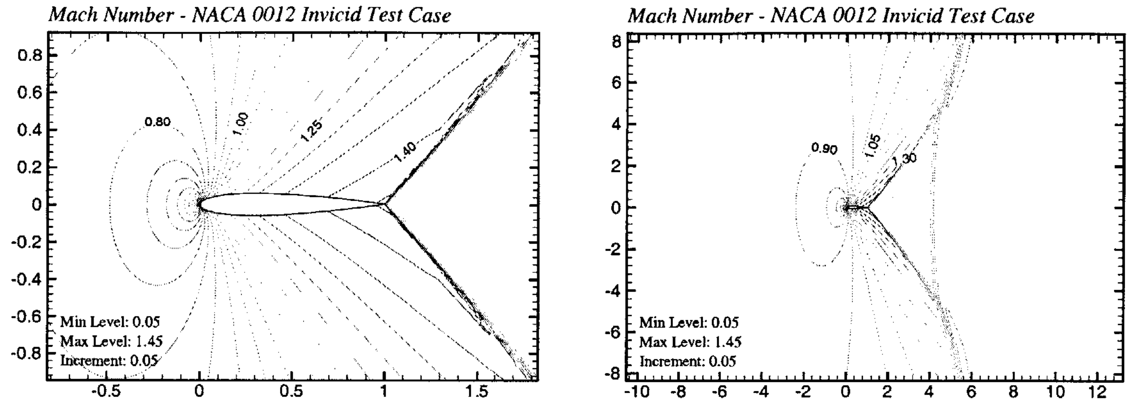


Figure 3-13: NACA 0012 test case: $M_\infty = 0.8$, $\alpha = 1.25^o$. Adaptive adjoint solution using the proposed output-based method with $e_o = 2.5\%\ C_L$. Left: magnitude of the computed x-momentum adjoint variable based on the lift. Right: blow-up near trailing edge.

the shock further away from the airfoil surface, where it is less important to have high resolution with respect to the lift.

Figure 3-12 shows blow-ups of the final refined grids for each of the adaptive runs. The pressure-based scheme refines the leading edge and shock regions significantly more than does the output-based method, however, it comparatively underrefines the trailing edge. In light of the convergence results, we conclude that the resolution of the trailing edge region is critical for computing an accurate lift in this test case.

This is further elucidated in Figure 3-13 where plots of the x-momentum adjoint variable are shown. The magnitude of the adjoint is plotted using a logarithmic contour distribution. It is evident from the right-hand plot that the magnitude of the adjoint is very large near the trailing edge, implying that the functional error is extremely sensitive to local residual errors in the trailing edge region. Other features of interest include what appears to be a singularity in the adjoint along the stagnation streamline and a weak discontinuity upstream of the primal shock on the upper surface. An analysis of the analytical adjoint for the quasi–one–dimensional and two–dimensional Euler equations is provided in reference [24]. Their analysis shows that the adjoint variables have an inverse square–root singularity along the up-



Mach Number - NACA 0012 Invicid Test Case

Mach Number - NACA 0012 Invicid Test Case

Figure 3-14: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Computed Mach number distributions. Left: blow up near the airfoil; right: more distant view.

stream stagnation streamline in the two-dimensional case, but are continuous across primal shocks. The adjoint plot in Figure 3-13 appears to be consistent with their results.

3.6.6 $M_\infty = 0.95$ NACA 0012 Airfoil

Transonic flow past a NACA 0012 airfoil [64] is simulated with a free-stream Mach number of $M_\infty = 0.95$, and angle of attack, $\alpha = 0^o$. The drag coefficient is chosen as the functional of interest in this case. Figure 3-14 shows plots of the computed Mach number distribution for a well-resolved solution. The left plot is a blow-up near the airfoil; the right plot illustrates some of the flow features further away from the airfoil. In this test case, the flow accelerates as it passes over the leading edge section through the sonic line to supersonic speeds. The flow then encounters an oblique shock at the trailing edge after which it remains at a low supersonic Mach number. Further downstream of the airfoil, a weak normal shock occurs lowering the Mach number to subsonic conditions. An adaptive scheme based on the local curvature of the Mach number or pressure field would tend to refine the shocks indefinitely owing to the unbounded derivatives across the discontinuity. However, if the primary goal is to accurately compute the airfoil drag, it is not necessary to refine the shocks downstream of the trailing edge to any appreciable extent. This test case demonstrates that the

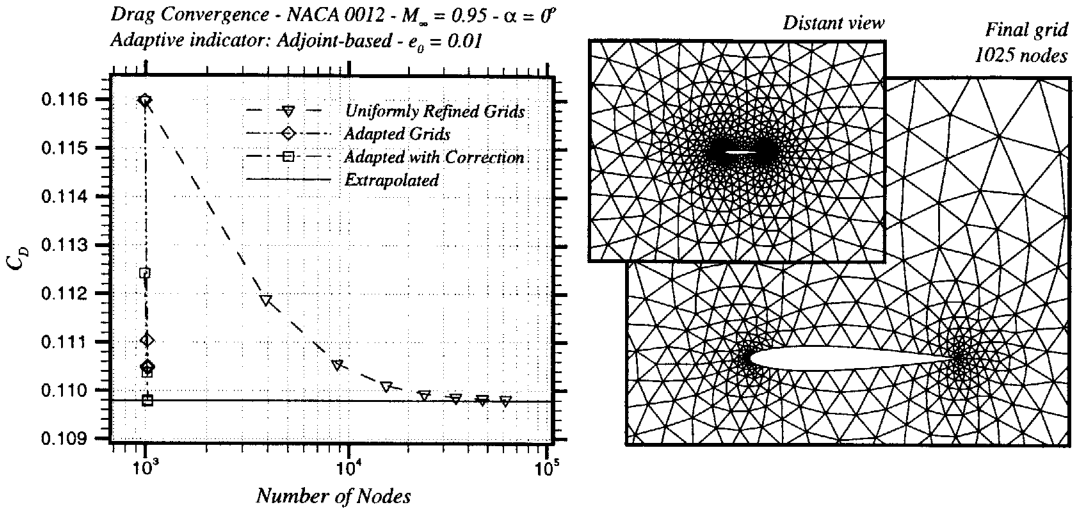


Figure 3-15: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Left: error convergence in the computed drag during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 0.01$ is prescribed. Right: final adapted grid.

output–based adaptive algorithm, unlike the curvature–based scheme, avoids refining the shocks and instead focuses on regions that are more important for computing the drag, such as the leading edge and trailing edge regions.

Figures 3-15 and 3-16 show adaptive results using the proposed output-based adaptive procedure. Desired error levels of $e_o = 0.01$ and $e_o = 0.0005$ are specified, respectively. The drag is plotted versus the total number of nodes in the corresponding grid. Also shown is the corrected drag on each grid after using the proposed error estimation procedure to correct the base value. For comparison, the drag convergence on a series of uniformly refined grids is also plotted.

In both adaptive cases the desired error levels are surpassed, the drag predictions converge to the correct value, and the adaptive process terminates after only two or three iterations. Note that the grid is refined near the leading edge region where the flow is rapidly accelerating. There is also some modest refinement near the trailing edge of the airfoil. Conversely, the oblique shock and the normal shock downstream of the airfoil are not resolved by the output–based adaptive algorithm.

Also of interest is the manner in which the adaptive procedure converges for different requested error levels, e_0. Comparing Figures 3-15 and 3-16, we see that

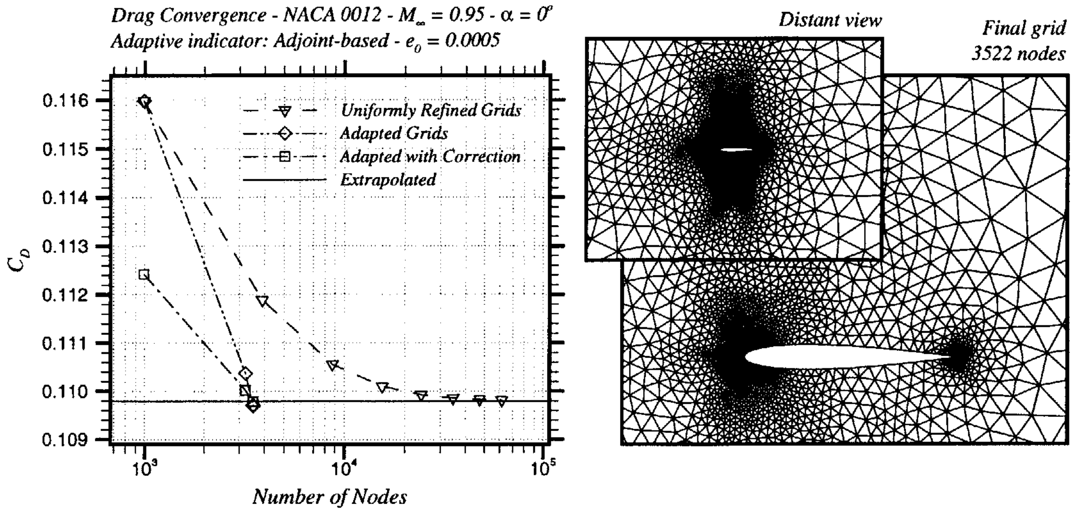


Figure 3-16: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Left: error convergence in the computed drag during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 0.0005$ is prescribed. Right: final adapted grid.

when e_0 is relatively large (e.g. $e_0 = 0.01$), the adaptive algorithm adds only a limited number of nodes (less then 50 new nodes are added) to quickly raise the desired accuracy. However, when higher accuracy is requested (i.e. lower values of e_0), the adaptive procedure tends toward more global refinement. For $e_0 = 0.0005$, the initial refinement more than triples the grid size, adding over 2000 new nodes.

Figure 3-17 shows plots of the initial (top left) and final (top right) grids in a typical adaptive run using the proposed output-based scheme. Below each of these are the corresponding distributions of the adaptation parameter ε_k. The adaptation parameter is effectively reduced during the adaptive process to a level below the minimum contour level on the plot.

The left-hand plot in Figure 3-18 shows the convergence of the computed drag on each grid during a typical adaptive run. The adaptive indicator is based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. It is evident that both the corrected and uncorrected values of the drag are converging to an erroneous value somewhat less than the extrapolated value. Furthermore, the adaptive process does not show signs of self-terminating and is, therefore, halted after 8 adaptive iterations. The right plot in Figure 3-18 shows

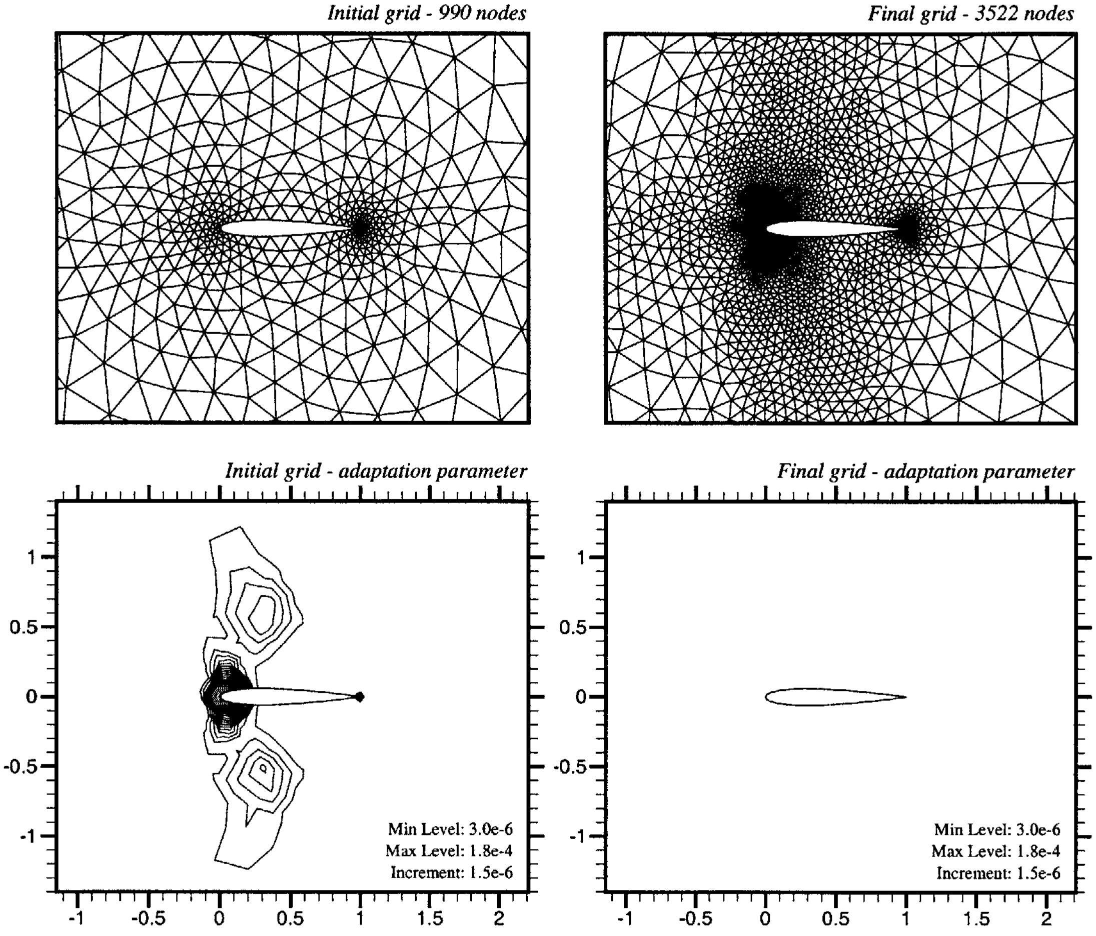


Figure 3-17: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Top left: original grid. Bottom left: adaptation parameter, ε_k, on the original grid. Top right: final adapted grid using the proposed output-based adaptive method with a prescribed error level of $e_o = 0.0005$. Bottom right: adaptation parameter on the final grid.

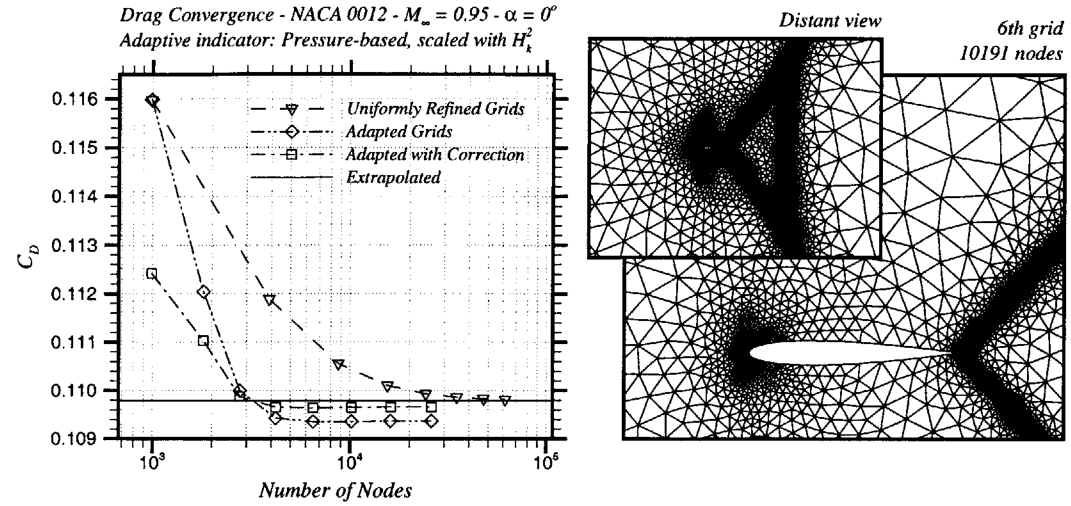


Figure 3-18: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Left: error convergence in the computed drag during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Right: intermediate grid in the adaptive run.

an intermediate grid in the adaptive sequence. The pressure-based indicator, being extremely sensitive to discontinuities, continues to refine the grid near the shocks despite the lack of convergence in the drag.

Figure 3-19 is analogous to Figure 3-18 in almost every regard except that the adaptive indicator used in this run is based on the magnitude of the second derivatives in the pressure scaled with H_k^3 instead of H_k^2. This is recommended as an improvement in reference [64]. This time, the predicted values for the drag are converging to an erroneous value significantly larger than the extrapolated value, however, it appears as though the adaptive process, itself, would have eventually terminated had it not been stopped after 8 iterations. Note that the functional corrections in Figures 3-18 and 3-19 are providing a significant improvement in accuracy. Nevertheless, the corrected values are still not converging to the extrapolated value of the drag.

Finally, Figure 3-20 shows analogous results using an indicator based on the magnitude of the second derivatives in the Mach number scaled with H_k^2. As with the pressure-based scheme, the predicted drag converges to an erroneous value slightly less than the extrapolated value. Furthermore, the adaptive process does not appear

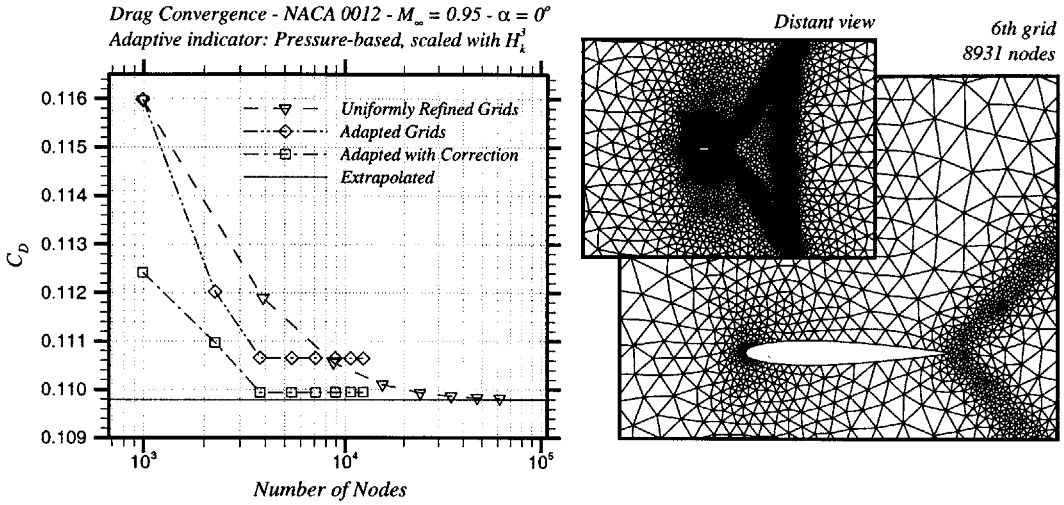


Figure 3-19: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Left: error convergence in the computed drag during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^3 where H_k is the local element size. Right: intermediate grid in the adaptive run.

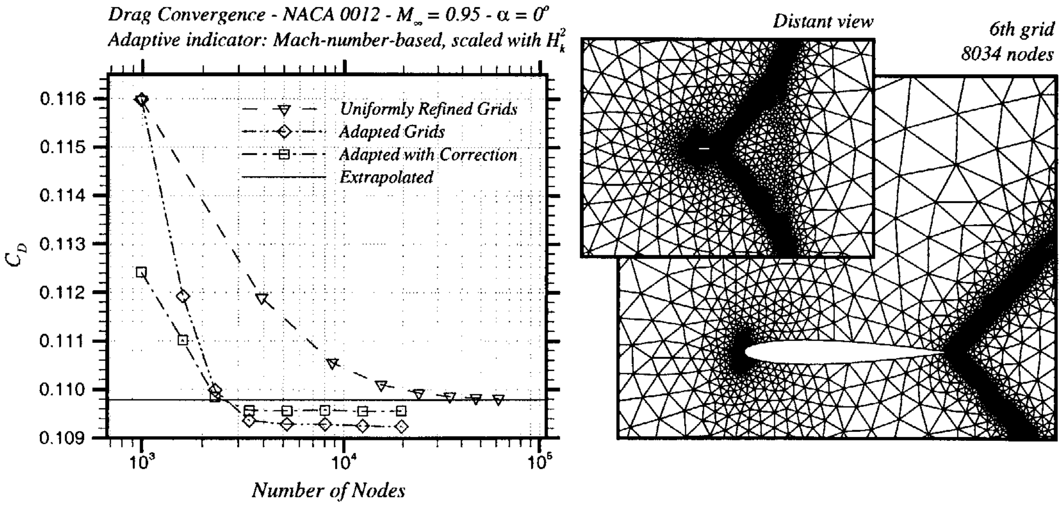


Figure 3-20: NACA 0012 test case: $M_\infty = 0.95$, $\alpha = 0^o$. Left: error convergence in the computed drag during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the Mach number scaled with H_k^2 where H_k is the local element size. Right: intermediate grid in the adaptive run.

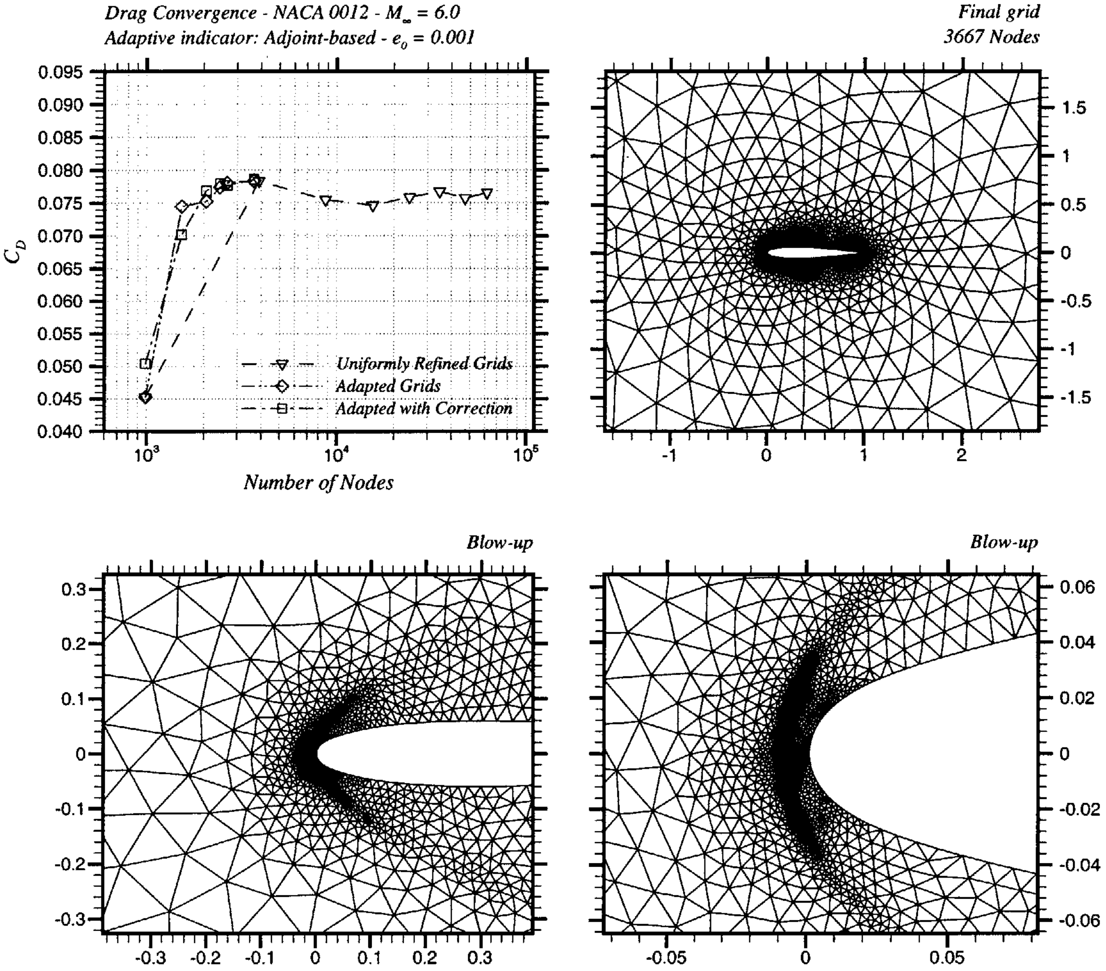


Figure 3-21: NACA 0012 test case: $M_\infty = 6.00$, $\alpha = 0^o$. Upper left: error convergence in the computed drag during a typical adaptive run. The proposed output-based adaptive algorithm is used. A requested error level of $e_o = 0.001$ is prescribed. Upper right: final adapted grid. Lower: blow-up of airfoil leading edge region.

to be terminating. The Mach number indicator continues to refine the grid near the shocks without any corresponding improvement in the accuracy of the drag.

3.6.7 $M_\infty = 6.0$ NACA 0012 Airfoil

Supersonic flow past a NACA 0012 airfoil is simulated with a free-stream Mach number of $M_\infty = 6.0$, and angle of attack, $\alpha = 0^o$. The functional of interest is the drag coefficient. This test case evaluates the performance of the proposed output–based adaptive algorithm in the presence of a strong shock.

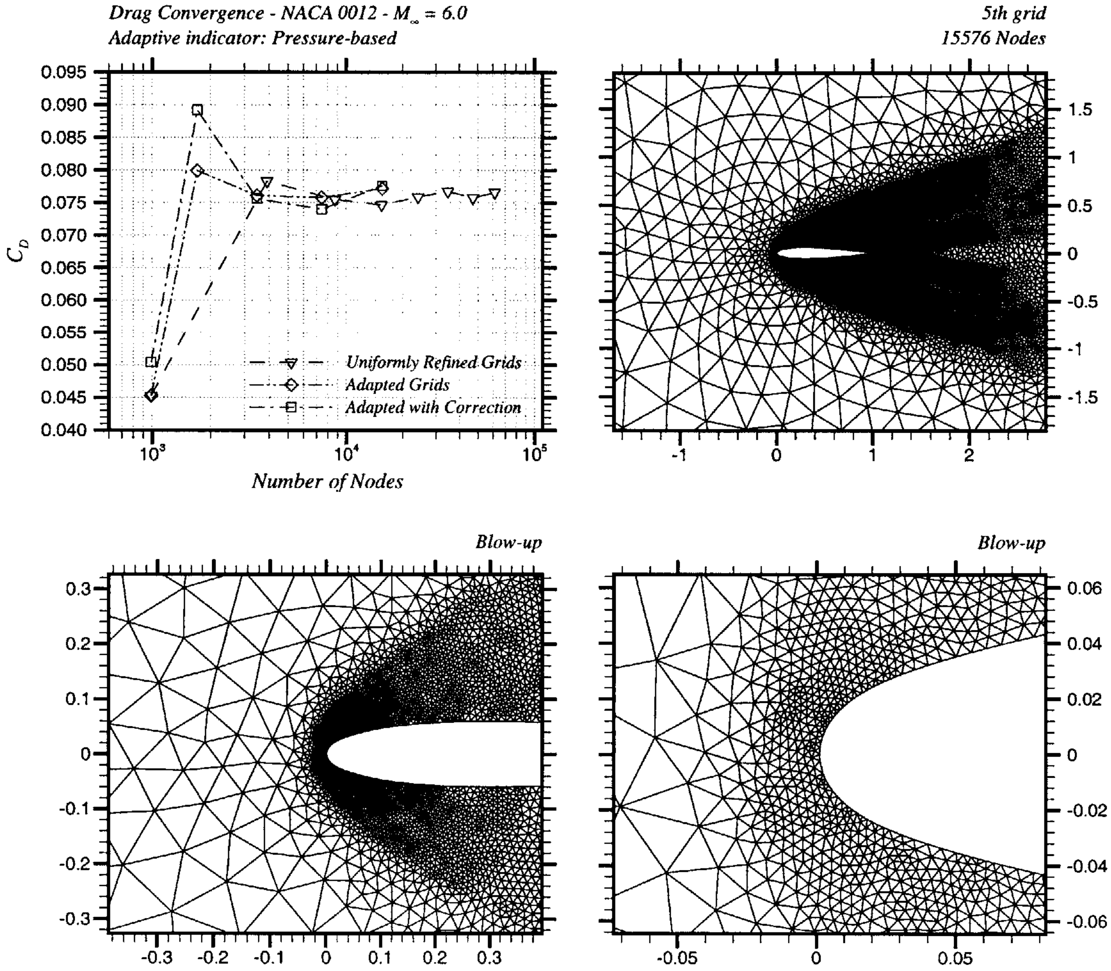


Figure 3-22: NACA 0012 test case: $M_\infty = 6.00$, $\alpha = 0^o$. Upper left: error convergence in the computed drag during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Upper right: final adapted grid. Lower: blow-up of airfoil leading edge region.

Output-based results are shown in Figure 3-21. Convergence of the drag to the desired error limit (in this case $e_0 = 0.001$) is achieved with under 4,000 nodes. By comparison, the uniform refinement cases have still not converged even after 60,000 nodes. Output–based adaptation occurs around the body in general but is most significant at the leading edge and at the bow shock in the leading edge region. Pressure-based adaptive results are shown in Figure 3-22. The adaptive algorithm does not appear to be self-terminating after the 5th adaptive iteration and the drag is exhibiting oscillatory convergence. Refinement for the pressure-based scheme is much more uniform between the bow shock and the airfoil, in contrast to the output-based method which favors leading-edge refinement. As one would expect, the leading-edge is a dominant region for determining aerodynamic performance.

3.6.8 $M_\infty = 3.0$ Two–element Airfoil

Two airfoils in a $M_\infty = 3.0$ free–stream are simulated. The left plot of Figure 3-23 shows the Mach number distribution for this flow. The airfoils are positioned such that the bow shock of the upstream airfoil interacts with the upper surface of the downstream airfoil. The functional is chosen to be the drag coefficient on the downstream airfoil. This test case is included to demonstrate that the proposed output–based procedure is capable of providing appropriate grid resolution for the shocks that influence the lower–airfoil drag while avoiding unnecessary refinement of the shocks that play no role whatsoever.

The right plot of Figure 3-23 shows a contour plot of the adjoint x-momentum variable based on the lower-airfoil drag. The results for the output-based and pressure-based algorithms are shown in Figures 3-24 and 3-25, respectively. The output-based approach (with an error tolerance of $e_0 = 0.001$) converges the lower airfoil drag in less than 4,000 nodes while the pressure-based approach requires over 37,000 nodes to achieve the same accuracy. The differences in adaptation are striking with the pressure-based approach excessively refining all shock waves including those which do not play any role in determining the lower airfoil drag (e.g. the upper surface shocks on the upstream airfoil). Also, the pressure-based refinement of the leading-edge of

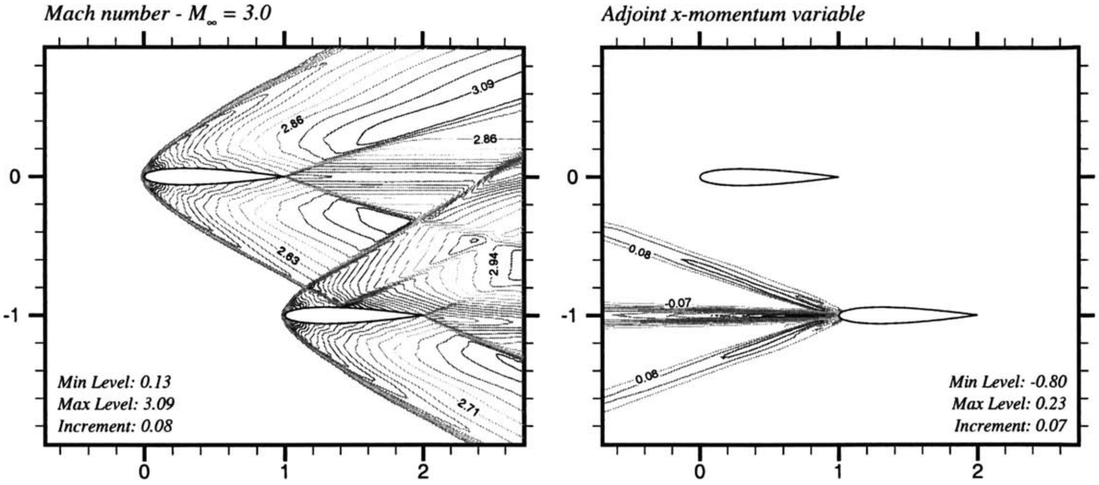


Figure 3-23: Two airfoil (NACA 0012) test case: $M_\infty = 3.00$, $\alpha = 0^o$. Left: computed Mach number distribution. Right: adjoint x-momentum variable based on the lower-airfoil drag.

the two airfoils is nearly identical whereas the output-based approach refines only those features which dominate the lower airfoil drag. Specifically, the leading-edge region (including the bow shock) of the lower airfoil, the oblique shock from the upper airfoil, and the leading-edge region of the upper airfoil are adapted. None of the trailing edge shocks are refined as they play little or no role in the drag.

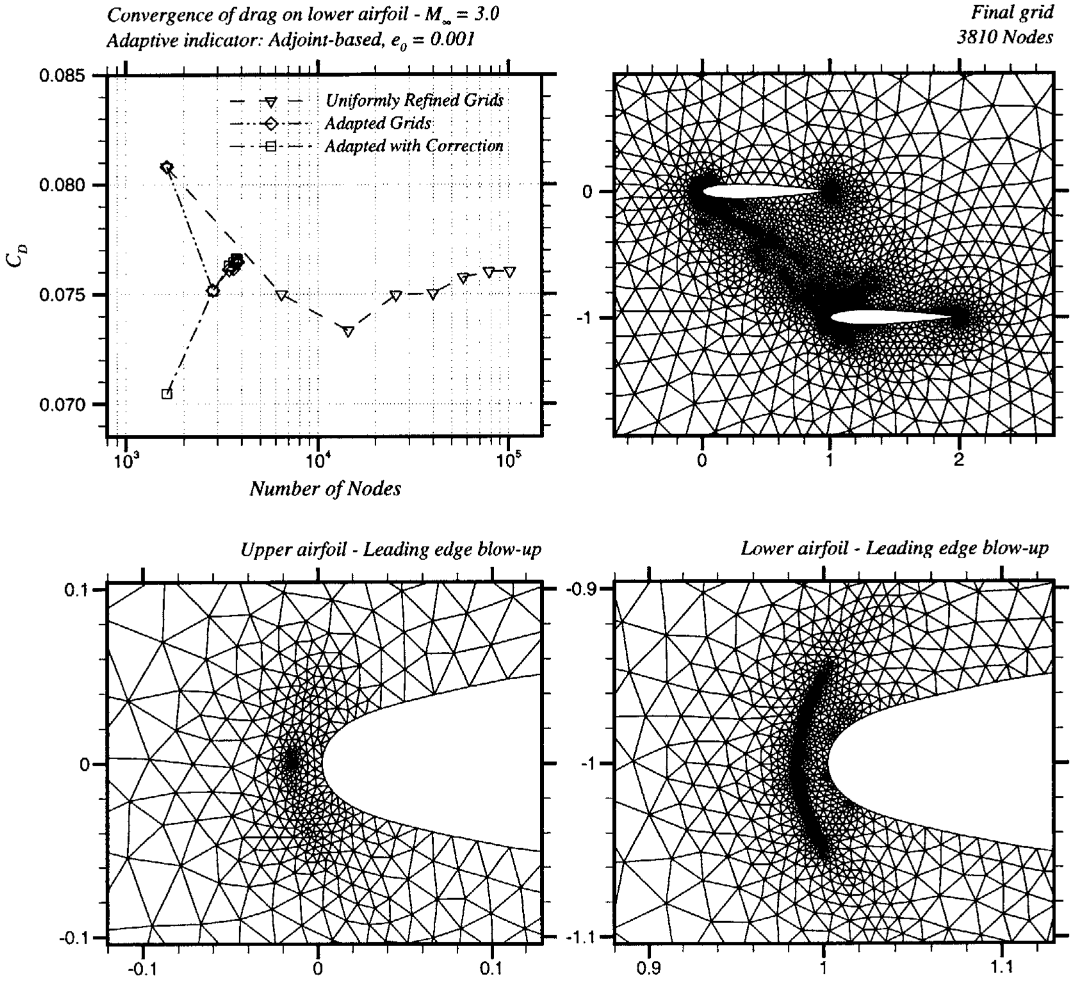


Figure 3-24: Two airfoil (NACA 0012) test case: $M_\infty = 3.00$, $\alpha = 0^o$. Upper left: error convergence in the computed drag on the *downstream* (i.e. lower) airfoil during a typical adaptive run. The proposed output-based adaptive algorithm is used. The adjoint is based on the lower-airfoil drag. A requested error level of $e_o = 0.001$ is prescribed. Upper right: final adapted grid. Lower: blow-up of leading edge regions.

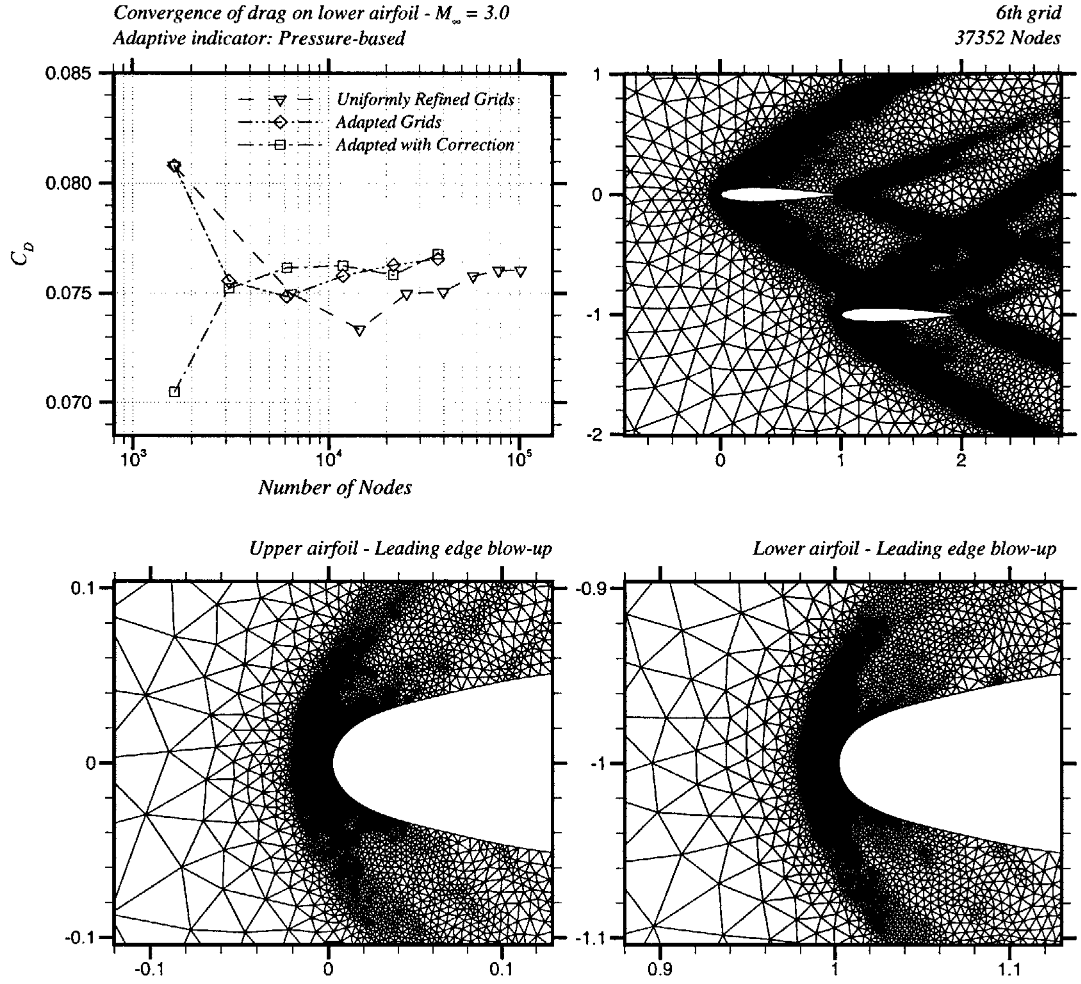


Figure 3-25: Two airfoil (NACA 0012) test case: $M_\infty = 3.00$, $\alpha = 0^o$. Upper left: error convergence in the computed drag on the *downstream* (i.e. lower) airfoil during a typical adaptive run. A curvature-based adaptive strategy is used [64] with an indicator based on the magnitude of the second derivatives in the pressure scaled with H_k^2 where H_k is the local element size. Upper right: final adapted grid. Lower: blow-up of leading edge regions.

Chapter 4

Viscous Flow

The output–based adaptive criteria presented in Chapter 2 are incorporated into a anisotropic grid–adaptive procedure for laminar Navier–Stokes simulations. The proposed method is demonstrated on a series of airfoil test cases for Reynolds numbers ranging from 5,000 to 100,000. The outputs considered are the lift and drag coefficients. The proposed anisotropic grid–adaptive method can be viewed as a merging of output error control with Hessian–based adaptation. Stretching and orientation of the elements in the grid are determined using information from the local Hessian matrix of second derivatives in the Mach number, whereas the local element size is determined using the output–based adaptive criteria. Use of the Mach number Hessian in the proposed output–based procedure is completely arbitrary; it is chosen in the present context in order to facilitate comparisons with a commonly used anisotropic adaptive method based exclusively on the Mach number Hessian [27, 28, 67]. The proposed output–based method is shown to compare very favorably in terms of output accuracy, reliability, and computational efficiency relative to pure Hessian–based adaptation.

4.1 Governing Equations

The Navier–Stokes equations are expressions of the principles of mass, momentum, and energy conservation for a viscous, compressible fluid. In two–dimensions these equations my be written as

$$\frac{\partial U}{\partial t}+\frac{\partial (F-F_v)}{\partial x}+\frac{\partial (G-G_v)}{\partial y}=0, \tag{4.1}$$

where U is the vector of conservative variables, F and G are the inviscid fluxes, and F_v and G_v are the viscous fluxes given, respectively, by

$$U=\left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho E\end{array}\right\}, \tag{4.2}$$

$$F=\left\{\begin{array}{c}\rho u\\ \rho u^2+p\\ \rho uv\\ u(\rho E+p)\end{array}\right\},\quad G=\left\{\begin{array}{c}\rho v\\ \rho uv\\ \rho v^2+p\\ v(\rho E+p)\end{array}\right\}, \tag{4.3}$$

and,

$$F_v=\left\{\begin{array}{c}0\\ \tau_{xx}\\ \tau_{xy}\\ u\tau_{xx}+v\tau_{xy}-q_x\end{array}\right\},\quad G_v=\left\{\begin{array}{c}0\\ \tau_{xy}\\ \tau_{yy}\\ u\tau_{xy}+v\tau_{yy}-q_y\end{array}\right\}. \tag{4.4}$$

In these expressions, ρ is the mass density; u and v are, respectively, the $x-$ and $y-$components of the fluid velocity; p is the static pressure and E is the total energy. The equations are nondimensionalized by the free stream density $\tilde{\rho}_\infty$, speed of sound $\tilde{a}_\infty$, temperature $\tilde{T}_\infty$, viscosity $\tilde{\mu}_\infty$, thermal conductivity $\tilde{k}_\infty$, and a reference length $\tilde{c}$. From this point onward, expressions are considered nondimensional with respect to these quantities unless accented by the tilde symbol. If the nondimensional conservative variables and flux vectors are of the form (4.1)–(4.4), then the corresponding nondimensional shear stress and heat conduction terms are given by

$$\tau_{xx} = \frac{2}{3}\frac{M_\infty}{\mathrm{Re}}\mu\left(2u_x - v_y\right), \tag{4.5}$$

$$\tau_{xy} = \frac{M_\infty}{\mathrm{Re}}\mu\left(u_y + v_x\right), \tag{4.6}$$

$$\tau_{yy} = \frac{2}{3}\frac{M_\infty}{\mathrm{Re}}\mu\left(2v_y - u_x\right), \tag{4.7}$$

$$q_x = \frac{-1}{\gamma - 1}\frac{M_\infty}{\mathrm{Re}}\frac{\mu}{\mathrm{Pr}}\frac{\partial a^2}{\partial x}, \tag{4.8}$$

$$q_y = \frac{-1}{\gamma - 1}\frac{M_\infty}{\mathrm{Re}}\frac{\mu}{\mathrm{Pr}}\frac{\partial a^2}{\partial y}, \tag{4.9}$$

where $\mathrm{Re} = \tilde{\rho}_\infty \tilde{V}_\infty \tilde{c}/\tilde{\mu}_\infty$ is the Reynolds number, $\tilde{V}_\infty = \left(\tilde{u}_\infty^2 + \tilde{v}_\infty^2\right)^{1/2}$ is the free–stream gas speed, and $M_\infty = \tilde{V}_\infty/\tilde{a}_\infty$ is the free–stream Mach number. In the present context, the working fluid is assumed to be ideal air with constant specific heat ratio $\gamma = 1.4$ and Prandtl number $\mathrm{Pr} = 0.72$. The dynamic viscosity is determined using Sutherland's law [65]

$$\mu = \frac{1 + C^*}{T + C^*}T^{3/2}, \tag{4.10}$$

with $C^* = \tilde{S}/\tilde{T}_\infty$, where $\tilde{S} = 198.6\,^\circ\mathrm{R}$ is Sutherland's constant and $\tilde{T}_\infty = 460.0\,^\circ\mathrm{R}$ is the assumed value for the free-stream temperature. The system is closed with the equation of state for an ideal gas

$$p = \rho(\gamma - 1)\left(E - \frac{u^2 + v^2}{2}\right). \tag{4.11}$$

Attention is focused on steady, laminar solutions of the Navier–Stokes equation system specified by (4.1)–(4.11).

4.2 Functional Outputs

The functional outputs considered are the lift and drag coefficients, denoted C_L and C_D, respectively. These coefficients are nondimensional forces expressible as surface integrals over portions of the domain boundary. The lift $\tilde{L}$ and drag $\tilde{D}$ forces acting on a given body are defined as the components of the net aerodynamic force on this body acting perpendicular and parallel, respectively, to the free–stream gas velocity $\vec{V}_\infty$. The lift and drag coefficients evaluated over the closed contour $\partial\Omega'$ can be expressed, respectively, as

$$\begin{aligned} C_L \equiv \frac{\tilde{L}}{\tilde{q}_\infty \tilde{c}} = \oint_{\partial\Omega'} \Big\{ & C_p \left(-n_x \sin\alpha + n_y \cos\alpha\right) \\ & + \frac{2}{M_\infty^2} \Big[\sin\alpha \left(\tau_{xx}\, n_x + \tau_{xy}\, n_y\right) \\ & \qquad - \cos\alpha \left(\tau_{xy}\, n_x + \tau_{yy}\, n_y\right) \Big] \Big\}\, ds, \end{aligned} \tag{4.12}$$

and,

$$\begin{aligned} C_D \equiv \frac{\tilde{D}}{\tilde{q}_\infty \tilde{c}} = \oint_{\partial\Omega'} \Big\{ & C_p \left(n_x \cos\alpha + n_y \sin\alpha\right) \\ & - \frac{2}{M_\infty^2} \Big[\cos\alpha \left(\tau_{xx}\, n_x + \tau_{xy}\, n_y\right) \\ & \qquad + \sin\alpha \left(\tau_{xy}\, n_x + \tau_{yy}\, n_y\right) \Big] \Big\}\, ds, \end{aligned} \tag{4.13}$$

where $\tilde{q}_\infty = \tilde{\rho}_\infty \tilde{V}_\infty^2/2$ is the free-stream dynamic pressure, and $\hat{\mathbf{n}} = n_x\hat{\imath} + n_y\hat{\jmath}$ is the unit normal vector on $\partial\Omega'$. The angle of attack α is defined by

$$\cos\alpha \equiv \frac{\vec{V}_\infty}{\left|\vec{V}_\infty\right|} \cdot \hat{\imath}, \tag{4.14}$$

implying that the chord of the airfoil is aligned with the x-axis. The pressure coefficient C_p is defined as

$$C_p \equiv \frac{\tilde{p} - \tilde{p}_\infty}{\tilde{q}_\infty} = \frac{2}{M_\infty^2} \left(p - p_\infty \right). \tag{4.15}$$

4.3 Flow and Adjoint Solvers

The flow and adjoint solvers used for the viscous simulations are part of the FUN2D suite of codes [41]. The FUN2D flow solver is a two-dimensional, implicit finite volume scheme that employs unstructured grids composed of triangular elements. Non-overlapping control volumes are constructed around each node in the grid by connecting the centroid of each triangle to the midpoints of its edges. The discrete residual equations associated with each interior node are obtained by numerically integrating the steady form of (4.1) over the control volume surrounding that node. These residual equations, therefore, represent approximations to the conservation laws embodied by (4.1). The Roe upwind scheme [53] is used to evaluate the inviscid fluxes at control-volume interfaces and a Galerkin-type formulation is employed to approximate the viscous terms. A brief description of the solution procedure used in the flow solver is given in Section 3.3.

The adjoint solver utilizes an exact linearization of the flow (primal) residual operator described above. The adjoint code is discussed further in Section 3.4. Additional details pertaining to the solvers can be found in the references [3, 42, 43].

4.4 Strong Boundary Conditions

The imposition of boundary conditions in a *strong sense* refers to the explicit modification of the form of the residual equations at boundary nodes relative to their counterparts in the interior. This has implications on the character of the discrete adjoint at the corresponding nodes. For Navier-Stokes simulations, the issue arises when the momentum and energy residuals associated with nodes on solid boundaries are discarded in favor of imposing no-slip and specified temperature conditions, respectively. Given a perturbation δU_h, the residual on the boundary represents the error in the Dirichlet data, whereas in the remainder of the domain it represents lack of momentum and energy conservation. Correspondingly, the discrete adjoint takes on a different character on the boundary relative to the interior. This will be illustrated in the following development, which follows closely that found in reference [22].

Consider the projection matrix B_h which is the zero matrix except for unit diagonal entries on the rows where Dirichlet conditions will be imposed strongly. It has the properties $(B_h)^n = B_h$ and $B_h^T = B_h$. We also have the complimentary projector $I_h - B_h$ where I_h is the identity matrix. Note that $B_h(I_h - B_h) = 0$. The primal system with Dirichlet conditions can be written as

$$\begin{aligned} (I_h - B_h)\, R_h(U_h) &= 0, \\ D_h\, U_h &= 0. \end{aligned} \tag{4.16}$$

The first expression represents the residual equations at interior nodes. These equations represent discrete conservation statements over the control volumes surrounding each interior node. In the second expression, the block diagonal matrix D_h sets the Dirichlet conditions in a strong sense. For example, in FUN2D, the Dirichlet conditions at solid boundary nodes are

$$\begin{aligned} u &= 0, \\ v &= 0, \\ \rho E - \frac{T_{wall}}{\gamma(\gamma - 1)}\rho &= 0. \end{aligned} \tag{4.17}$$

Thus, the 4×4 diagonal block of $D_h\, U_h = 0$ corresponding to a boundary node is given by

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -K & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{4.18}$$

where $K = T_{wall}/\gamma(\gamma - 1)$.

For convenience, we consider a different set of primal variables $\bar{U}_h$ through an affine transformation $U_h = A_h\bar{U}_h$. We choose A_h such that $B_h = D_h A_h$. This allows the Dirichlet conditions on the new variables to be set using the projection matrix B_h since $D_h\, U_h = D_h\, A_h\, \bar{U}_h = B_h\, \bar{U}_h$. In FUN2D, the appropriate choice for A_h is

$$A_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ K & 0 & 0 & 1 \end{bmatrix}, \tag{4.19}$$

yielding the new variables

$$\begin{aligned} \bar{U}_h &= A_h^{-1}\, U_h, \\ &= (\rho,\ \rho u,\ \rho v,\ \rho E - \rho K)^T . \end{aligned} \tag{4.20}$$

Thus, the primal system can be rewritten as,

$$\begin{aligned} (I_h - B_h)\, R_h(U_h) &= 0, \\ B_h\, \bar{U}_h &= 0, \end{aligned} \tag{4.21}$$

or, combining these,

$$(I_h - B_h)\, R_h(U_h) + B_h\, \bar{U}_h = 0. \tag{4.22}$$

The perturbation in the primal solution $\delta U_h = A_h \delta \bar{U}_h$ satisfies, approximately,

$$\begin{aligned} (I_h - B_h) \frac{\partial R_h}{\partial U_h} A_h \, \delta \bar{U}_h &\approx (I_h - B_h) \, \delta R_h, \\ B_h \delta \bar{U}_h &= D_h \, \delta U_h. \end{aligned} \tag{4.23}$$

This can be combined into

$$\left((I_h - B_h) \frac{\partial R_h}{\partial U_h} A_h + B_h \right) \delta \bar{U}_h \approx (I_h - B_h) \, \delta R_h + D_h \, \delta U_h. \tag{4.24}$$

The discrete adjoint system is, therefore, given by

$$\left(A_h^T \frac{\partial R_h}{\partial U_h}^T (I_h - B_h) + B_h \right) \Psi_h = A_h^T \frac{\partial f_h}{\partial U_h}^T . \tag{4.25}$$

Note that Equation (2.6) is recovered if we left-multiply the last expression by A_h^{-T}.

Using (2.4), (4.24) and (4.25), the perturbation in the functional can be expressed as

$$\begin{aligned} \delta f_h &\approx \frac{\partial f_h}{\partial U_h} \, \delta U_h, \\ &= \frac{\partial f_h}{\partial U_h} \, A_h \, \delta \bar{U}_h, \\ &= \left[\left(A_h^T \frac{\partial R_h}{\partial U_h}^T (I_h - B_h) + B_h \right) \Psi_h \right]^T \delta \bar{U}_h, \\ &= \Psi_h^T \left((I_h - B_h) \frac{\partial R_h}{\partial U_h} A_h + B_h \right) \delta \bar{U}_h, \\ &\approx \Psi_h^T \, \left[(I_h - B_h) \, \delta R_h + D_h \, \delta U_h \right]. \end{aligned} \tag{4.26}$$

Now consider a decomposition of the adjoint into two orthogonal components,

$$\Psi_h = \Psi_h^{(i)} + \Psi_h^{(b)}, \tag{4.27}$$

where

$$\begin{aligned}\Psi_h^{(i)} &= (I_h - B_h)\Psi_h,\\ \Psi_h^{(b)} &= B_h\Psi_h. \end{aligned} \tag{4.28}$$

The superscripts (b) and (i) correspond to strong boundary nodes and interior nodes, respectively. Using this decomposition, the last expression for the functional error in (4.26) can be rewritten as

$$\delta f_h \approx \left(\Psi_h^{(i)}\right)^T \delta R_h + \left(\Psi_h^{(b)}\right)^T \delta \bar{U}_h, \tag{4.29}$$

showing that the adjoint takes on a different role on the boundary relative to its function in the interior: the boundary adjoint weighs the solution error whereas the interior adjoint weighs the residual error.

Figure 4-1 shows a contour plot of the discrete adjoint x–momentum variable for a Navier-Stokes simulation of flow past a cylinder. The corresponding grid is plotted on the left. In this case, Re $= 10$, $M_\infty = 0.38$ and the adjoint is based on the drag. The distinction between the boundary adjoint and the interior adjoint is clear from the clustering of contour lines near the boundary. The contour levels in this plot are distributed linearly within each element, however, the sharp variations within the elements adjacent to the boundary are meaningless since two entirely different quantities are being interpolated within these elements.

4.4.1 Recovering the Boundary Adjoint

As illustrated in the previous section, using interpolation as a means of transferring the coarse-grid adjoint onto the fine grid is not appropriate for elements adjacent to the boundary. A viable alternative is to extrapolate the interior adjoint to the boundary of the coarse grid and prolongate the extrapolant onto the fine grid, as described in Section 4.4.3. Once this is done, the only remaining task is to postprocess the boundary adjoint on the fine grid. The formula for recovering the boundary

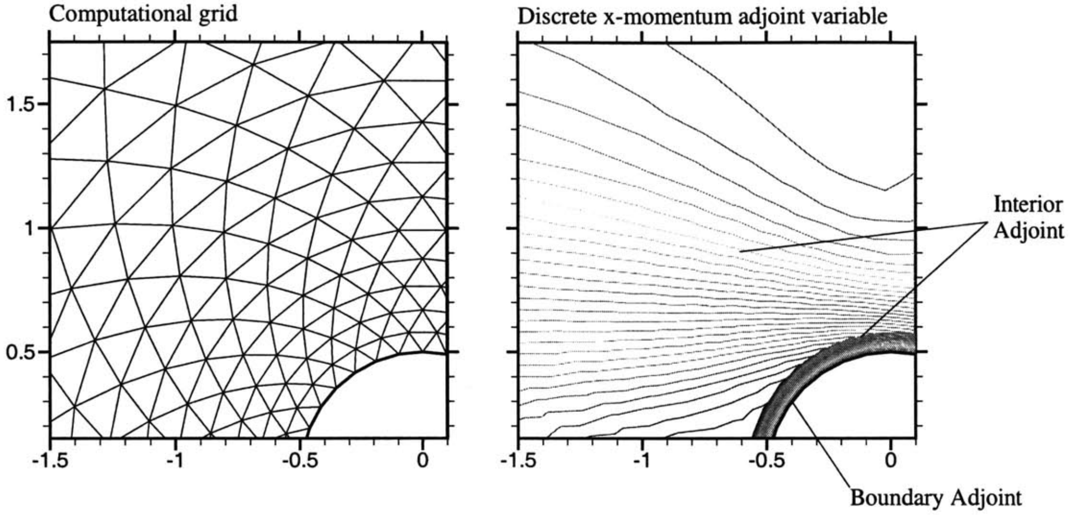


Figure 4-1: Discrete x–momentum adjoint variable based on the the drag (right), and corresponding grid (left) for viscous flow past a circular cylinder (Re = 10, $M_\infty = 0.38$).

adjoint given an approximation of the interior adjoint is derived below.

Substituting (4.27) into (4.25) and premultiplying by $(I_h - B_h)$ yields

$$(I_h - B_h)A_h^T \frac{\partial R_h}{\partial U_h}^T \Psi_h^{(i)} + (I_h - B_h)B_h\Psi_h = (I_h - B_h)A_h^T \frac{\partial f_h}{\partial U_h}^T . \tag{4.30}$$

The second term is zero and the interior adjoint satisfies

$$\begin{aligned}(I_h - B_h)A_h^T \frac{\partial R_h}{\partial U_h}^T \Psi_h^{(i)} &= (I_h - B_h)A_h^T \frac{\partial f_h}{\partial U_h}^T, \\ B_h\Psi_h^{(i)} &= 0, \end{aligned} \tag{4.31}$$

which can be combined into

$$\left((I_h - B_h)A_h^T \frac{\partial R_h}{\partial U_h}^T + B_h\right)\Psi_h^{(i)} = (I_h - B_h)A_h^T \frac{\partial f_h}{\partial U_h}^T, \tag{4.32}$$

proving that $\Psi_h^{(i)}$ is not dependent on $\Psi_h^{(b)}$. The expression for the adjoint on the strong boundary may be calculated in a postprocessing step once $\Psi_h^{(i)}$ has been obtained from (4.32). The desired formula is obtained directly from (4.25) as,

$$\begin{aligned}\Psi_h^{(b)} &= A_h^T \left(\frac{\partial f_h}{\partial U_h}^T - \frac{\partial R_h}{\partial U_h}^T \Psi_h^{(i)} \right), \\ &= -A_h^T R_h^{\Psi}(\Psi_h^{(i)}). \end{aligned} \tag{4.33}$$

Similarly, if $\tilde{\Psi}_h$ is an approximation to the interior adjoint on the fine grid, then the discrete boundary adjoint may be approximated as

$$\Psi_h^{(b)} \approx -B_h A_h^T R_h^{\Psi}\Big((I_h - B_h) \tilde{\Psi}_h \Big) . \tag{4.34}$$

4.4.2 Residual-Based Functionals

Engineering outputs often involve boundary integrals of diffusive-type fluxes. These fluxes involve gradients of the solution normal to the boundary. In Navier-Stokes applications, the skin friction contribution to the aerodynamic forces on a body can be expressed as a contour integral involving the gradients of the velocity components at the surface of the body. One method of computing these forces is to differentiate the solution within the elements adjacent to the body, and then integrate the result. Evaluating the forces in this manner generally results in poor convergence of the output. In the finite element community it is commonly referred to as an unbounded form of the output. Finite volume practitioners often labeled it as a non-conservative evaluation of the output. This type of functional definition may also lead to irregularities in the adjoint variables near the corresponding boundaries resulting in compromised accuracy of the corrected functionals and diminished effectiveness of the adaptive algorithm. A more natural and accurate method of evaluating the aerodynamic forces, which is both bounded and conservative, is to define them as residual balances as described below.

Let $\bar{R}_h(U_h)$ represent the vector of residual equations prior to imposing no-slip conditions and specified temperature in a strong sense at the solid boundaries. These equations are discrete conservation statements defined over control volumes surrounding each node in the grid. At the boundary nodes, the momentum and energy residuals are missing viscous flux contributions from the wall. In FUN2D, $\bar{R}_h(U_h)$ includes pressure contributions from integrating the inviscid fluxes along the boundary, but is missing flux contributions from the viscous terms. In the case of the momentum residuals, these missing fluxes correspond to the shear forces that would be needed in order to maintain momentum conservation over the control volumes surrounding the boundary nodes. The sum of these forces over all boundary nodes corresponds to the total skin friction force on the body. In this vein [22], the forces can be defined as

$$f_h(U_h) = f_h^p(U_h) + c_h^T B_h \bar{R}_h(U_h), \tag{4.35}$$

where $f_h^p(U_h)$ is the pressure contribution[1]. The vector c_h takes the appropriate component of the momentum residuals into the selected force direction (whether it be lift or drag, for example). The right hand side for the discrete adjoint system is given by

$$\frac{\partial f_h}{\partial U_h}^T = \frac{\partial f_h^p}{\partial U_h}^T + \frac{\partial \bar{R}_h}{\partial U_h}^T B_h c_h. \tag{4.36}$$

Fortunately, the last term in this expression can be evaluated using existing subroutines from the discrete adjoint code.

Figure 4-2 compares the computed x–momentum adjoint variable near the boundary of a circular cylinder for a non-conservative evaluation (center plot) and a conservative evaluation (right plot) of the drag. The flow conditions for this simulation are Re $= 10$ and $M_\infty = 0.38$. The computational grid is plotted on the left. In the center plot, irregularities in the adjoint are observed in the second layer of elements away from the boundary. These irregularities are not present in the right plot, illustrating that the conservative evaluation of the drag leads to an interior adjoint with better smoothness properties. The clustering of contour lines in the first layer of elements reflects the presence of the boundary adjoint, which is a separate issue as discussed earlier in the section.

[1] If $\bar{R}_h(U_h)$ is missing both inviscid and viscous flux contributions from the wall, then $f_h^p(U_h)$ should be omitted from (4.35) since it would already be incorporated into $\bar{R}_h(U_h)$.

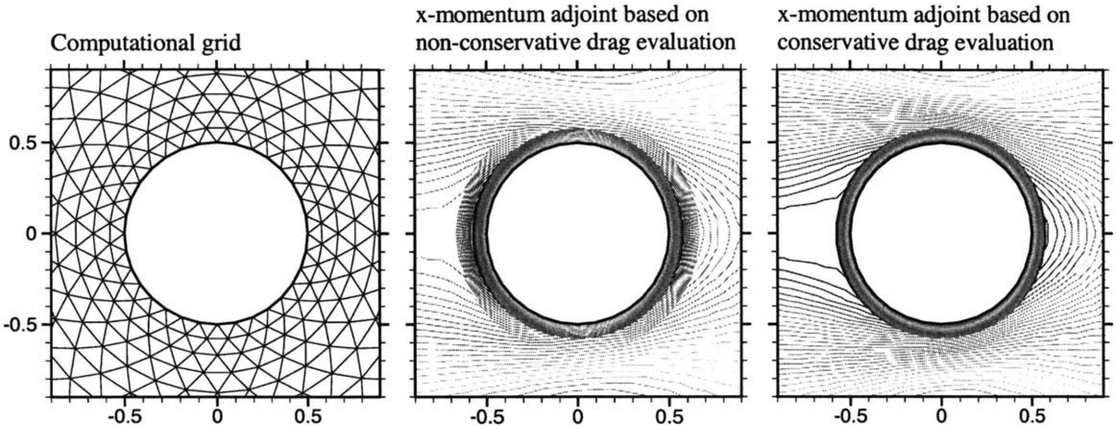


Figure 4-2: Comparison of the x–momentum adjoint variable based on a non-conservative (center) and a conservative (right) evaluation of the drag for viscous flow past a circular cylinder (Re $= 10$, $M_\infty = 0.38$). The computation grid is plotted on the left.

Figure 4-3 compares lift and drag convergence plots for the same test case using both types of functional evaluations. These values are obtained from simulations on a series of uniformly refined grids. The farfield boundary is placed at 22.5 diameters. The coarsest grid in the series contains 800 nodes and is shown in Figure 4-2. The nth grid in the series is constructed by subdividing each triangle in the coarsest grid into n^2 self–similar triangles for values of $n = 2$, 4, and 8. Boundary edges are made to conform to the circular cylinder as described in Sections 2.1.1 and 2.2.3. The lift and drag errors are plotted versus a characteristic length parameter h associated with each grid in the series. A value of $h_1 = 1$ is arbitrarily assigned to the coarsest grid. The length parameter associated with the nth grid is given by $h_n = h_1/n$. The true lift and drag values used to measure the errors in Figure 4-3 are taken as $C_L = 0$ and $C_D = 2.99165$, respectively. The true drag value is obtained as a Richardson extrapolation of the conservative drag values from the three finest grids. The implied convergence rate from the extrapolation is approximately third order. We see from Figure 4-3 that the non-conservative evaluation produces essentially first-order-accurate values for the forces whereas the conservative evaluation is essentially third order.

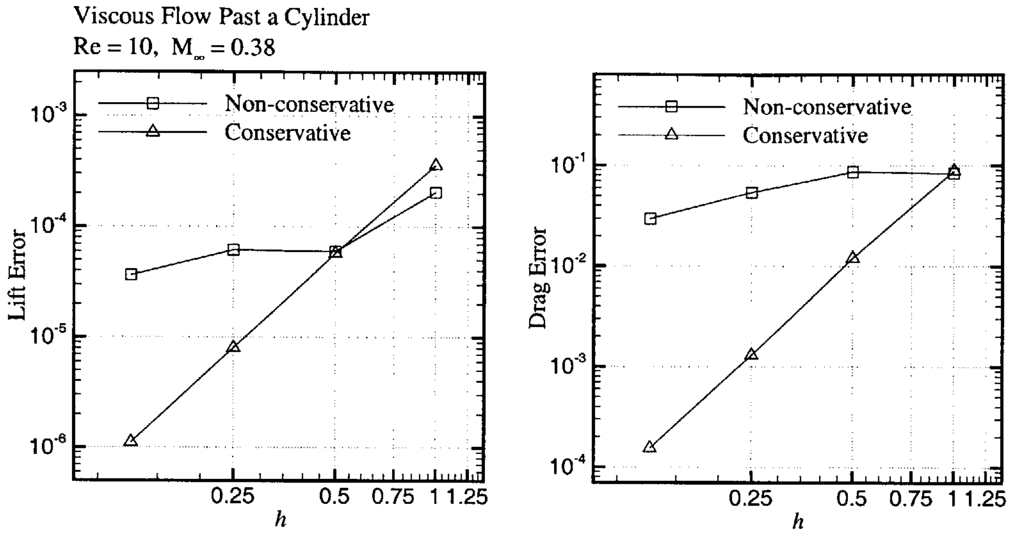


Figure 4-3: Convergence of the error in the lift and drag on a series of uniformly refined grids for viscous flow past a circular cylinder (Re $= 10$, $M_\infty = 0.38$). The conservative and non-conservative evaluations of the forces are compared.

4.4.3 Extrapolation of the Interior Adjoint to the Boundary

The prolongation operators L_h^H and Q_h^H, defined in Section 2.2, should only be applied to coarse-grid functions that are uniform in character throughout the domain. If the primal problem involves strongly imposed boundary conditions (as is the case for the convection-diffusion and viscous test cases in the present work), the discrete adjoint takes on a fundamentally different character on the boundary relative to the interior as discussed earlier in the section. Interpolation of the adjoint across boundary and interior nodes is not appropriate in this case. An alternative approach, used in the present work, is to extrapolate the interior adjoint to the boundary of the coarse grid thereby establishing polynomial extrapolants over each coarse-grid element adjacent to the boundary. The value of the interior adjoint at the fine-grid nodes within these elements is obtained by evaluating the corresponding extrapolant. The interior coarse-grid elements are handled by the standard interpolation-based prolongation operations associated with L_h^H and Q_h^H. Once the entire prolongation of the interior adjoint is completed, the final task is to postprocess the boundary adjoint on the fine grid using (4.34).

The linear and quadratic prolongation operators modified for extrapolation at the boundary are denoted $\bar{L}_h^H$ and $\bar{Q}_h^H$, respectively. The corresponding algorithm that defines their operation is summarized here:

1. Loop through all the coarse-grid nodes and flag those nodes where primal boundary conditions are imposed in a strong sense (hereafter referred to as strong nodes). If a node that is initially flagged as an interior (non-strong) node has less than two interior nodes as its immediate neighbors, then that node is re-designated a strong node.

2. Flag all coarse-grid elements with at least one strong node as a vertex (hereafter referred to as strong elements).

3. Flag all coarse-grid edges with at least one strong node as an endpoint (hereafter referred to as strong edges).

4. In the case of the quadratic operator $\bar{Q}_h^H$, compute the gradients of the interior adjoint at the interior nodes of the coarse grid using a modification of the least-squares algorithm described in reference [3]. This algorithm computes gradients at a node as least squares averages of the gradients along each edge impinging upon that node. The algorithm is modified by using only those edges that are not strong edges. This allows gradients of the interior adjoint to be computed without contamination from the boundary adjoint.

5. For all interior (non-strong) elements, construct polynomial interpolants using operations associated with the operators L_h^H and Q_h^H described in Section 2.2.

6. For all strong elements construct polynomial extrapolants by extending the polynomial interpolants of adjacent interior elements. If there is more than one adjacent interior element take the arithmetic average of the extended interpolants. If there are no adjacent interior elements use the interpolant of the closest (in terms of Euclidean distance between centroids) interior element.

7. Compute the prolongated nodal values on the embedded fine grid by evaluating the appropriate interpolants/extrapolants from the coarse grid. Values

for fine-grid nodes on coarse-grid edges are obtained by taking the arithmetic average of the interpolant/extrapolant values from the two adjacent elements associated with that edge. Values at fine-grid nodes that are colocated with coarse-grid nodes are similarly obtained by taking the arithmetic average of the interpolant/extrapolant values from the patch of elements surrounding that node.

8. Postprocess the boundary adjoint on the fine grid using (4.34).

4.5 Anisotropic Adaptation

The adjoint–based adaptive criteria derived in Section 2.3.2 are incorporated into a framework for anisotropic adaptation for functional outputs. The goal is to ensure that the estimated remaining error in the functional, after correction, is less than a specified tolerance prescribed *a priori*. In Section 4.5.1 the theory and procedure is laid out for standard interpolation–error–based adaptation using the Hessian matrix of second derivatives of a chosen scalar [27, 48]. In Section 4.5.2 a geometric analogy is made between the interpolation error along an edge in the grid and the generalized length of this edge in a Riemannian space [12, 16, 19, 27]. The transformation from the physical space to the Riemannian space is governed by a symmetric positive–definite metric that can be obtained from the Hessian matrix by taking the absolute value of the eigenvalues of the symmetric Hessian. In a typical Hessian-based adaptive procedure, the size, stretching, and orientation information for the elements in the grid are obtained exclusively from the local Hessian matrix. In the proposed output–based procedure, only the stretching and orientation characteristics are obtained from the Hessian. The local element size is determined from the adjoint–based adaptive parameters derived in Section 2.3.2. This allows for effective control of the functional error while simultaneously achieving more efficient resolution of the flow field than would otherwise be possible with isotropic adaptation alone.

4.5.1 Interpolation Error

Error indicators for driving anisotropic adaptation of unstructured triangular and tetrahedral grids are typically based on the interpolation error incurred by approximating continuous functions using piecewise–linear interpolation [16, 27, 48, 67].

Consider the one–dimensional function $u(x)$ and a piecewise linear approximation $u_h^L(x)$ of this function on a grid with node spacing h. The error at the nodes is zero, by construction. It can be shown through simple Taylor–series arguments (see, for example [58]) that the maximum interpolation error over a typical interval $x \in \Delta_j = [jh, (j+1)h]$ is given by

$$\max_{x\in\Delta_j} \left| u(x) - u_h^L(x) \right| \leq \frac{1}{8} h^2 \max_{x\in\Delta_j} \left| u_{xx} \right| . \tag{4.37}$$

This error estimate can be directly extended to higher dimensions by restricting attention to the interpolation error along a line. For example, consider using a piecewise–linear interpolant $u_h^L(x,y)$ over a two–dimensional triangulation to approximate the continuous function $u(x,y)$. Assuming zero error at the nodes, the maximum interpolation error over an edge E in the triangulation, with unit tangent vector $\hat{s}$ and length h, is given by

$$\max_{(x,y)\in E} \left| u(x,y) - u_h^L(x,y) \right| \leq \frac{1}{8} h^2 \max_{(x,y)\in E} \left| u_{ss} \right| , \tag{4.38}$$

In this last expression u_{ss} is the directional second derivative of u in the direction $\hat{s}$, which can be expressed as

$$u_{ss} = \hat{s}^T \mathcal{H} \hat{s}, \tag{4.39}$$

where

$$\mathcal{H} = \begin{bmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{bmatrix} , \tag{4.40}$$

is the Hessian matrix of second derivatives.

4.5.2 Riemannian Metric

A geometric interpretation of the interpolation error along an edge can be made by considering the length of that edge in a Riemannian space [12, 16, 19, 27]. The transformation from the physical space to the Riemannian space is specified by a symmetric positive–definite matrix $\mathcal{M}$,

$$\mathcal{M} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} . \tag{4.41}$$

The generalized length $l_{\mathcal{M}}$ of edge E in the transformed space is given by

$$l_{\mathcal{M}}^2 = \hat{s}^T \mathcal{M} \hat{s}\, h^2, \tag{4.42}$$

Note that if $\mathcal{M}$ is the identity matrix, the physical (Euclidean) length $l_E^2 = \hat{s}^T I \hat{s}\, h^2 = h^2$ is recovered.

To illustrate the directional properties of the metric $\mathcal{M}$, consider the unit phasor $\hat{e} = (x, y)^T$ stemming from the origin in the metric space. The square of its metric length is given by

$$\hat{e}^T \mathcal{M} \hat{e} = ax^2 + 2bxy + cy^2 = 1. \tag{4.43}$$

This is the equation of an ellipse centered at the origin in the physical space. The ellipse is rotated by an angle θ to the x–axis, and has major and minor principal lengths h_1 and h_2, respectively, as shown in Figure 4-4. These parameters are related to the metric components (a, b, c) as follows:

$$\mathcal{M} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = R \begin{bmatrix} 1/h_1^2 & 0 \\ 0 & 1/h_2^2 \end{bmatrix} R^T , \tag{4.44}$$

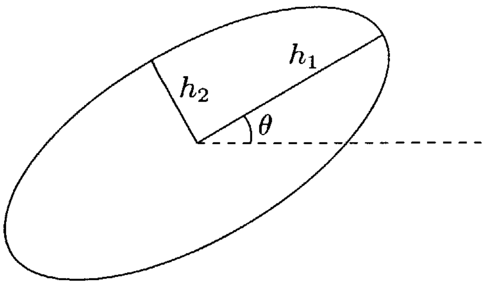


Figure 4-4: Ellipse defined by the metric parameters θ, h_1 and h_2.

where,

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \tag{4.45}$$

Comparing (4.38)–(4.39) with (4.42), the interpolation error along edge E is analogous to the square of the metric length of that edge if $\mathcal{M}$ is chosen to be a modification of the Hessian matrix. In particular, the Hessian is modified by taking the absolute value of its eigenvalues, resulting in a symmetric positive–definite matrix. Let R be the orthonormal matrix containing the eigenvectors of $\mathcal{H}$ as its columns, and let Λ be the corresponding diagonal matrix containing its eigenvalues. The symmetric Hessian is diagonalized by

$$\mathcal{H} = R\,\Lambda\,R^T, \tag{4.46}$$

and the Hessian–based metric can be defined as

$$\mathcal{M}_{\mathcal{H}} = R\,|\Lambda|\,R^T. \tag{4.47}$$

One possible strategy for anisotropic adaptation is to equidistribute the error along edges in the grid by equidistributing the length of the edges in the Riemannian space governed by $\mathcal{M}_{\mathcal{H}}$ [12, 16, 19, 27].

The definition for the metric length in (4.42) assumes that the metric is constant over the edge. In general, the metric may vary continuously over the physical domain. Consider the parameterization of $\mathcal{M}$ using $t \in [0,1]$ on edge E. The metric length can be generalized as

$$l_{\mathcal{M}} = h \int_0^1 \left(\hat{s}^T \mathcal{M}(t)\, \hat{s}\right)^{1/2} dt, \tag{4.48}$$

In practice, the metric is interpolated over a background grid and the integral in (4.48) is approximated using a prescribed quadrature rule [27].

4.5.3 Pure Hessian–Based Adaptation

In pure Hessian–based adaptation, the element size, aspect ratio, and orientation are determined using the Hessian–based metric $\mathcal{M}_{\mathcal{H}}$. As seen in (4.44) and (4.45), the eigenvalues $\sigma_{\min} = 1/h_1^2$ and $\sigma_{\max} = 1/h_2^2$ are related to lengths in two orthogonal directions and the eigenvectors yield the orientation of these directions with respect to the coordinate axis.

A Hessian-based scheme is implemented for the purposes of comparing the effectiveness of the output-based strategy described in Section 4.5.4. The algorithm attempts to equidistribute the interpolation error in the computed Mach number over each edge in the grid. Different error tolerances are achieved by scaling the metrics associated with each element by a constant multiplicative factor κ. Larger values of κ correspond to more stringent tolerances on the estimated interpolation error. The Mach–number Hessian is obtained using the quadratic reconstruction procedure described in Section 2.2.2. Piecewise quadratic Mach number profiles are constructed over each element and differentiated twice yielding piecewise constant second derivatives. Nodal values of the second derivatives are obtained using an area–weighted average of the piecewise–constant values over each element surrounding the node. To avoid excessive element sizes in the farfield, the eigenvalues of the metric are bounded from below so that $h_2 \leq h_1 \leq h_{\max}$, where $h_{\max}$ is a specified maximum element length for the domain. A value of $h_{\max} = 10$ chord–lengths is used in the present simulations. A typical adaptive simulation requires several adaptive iterations before grid convergence is achieved. At each iteration, a flow solution is obtained, the Mach–number Hessian and corresponding metric are computed at the nodes, and the grid is regenerated using the BL2D grid generator (see Section 4.5.5), which takes the nodal metrics as input.

Adaptation based on the Mach–number Hessian alone is known to have several deficiencies. For example, curves of inflection in the Mach–number distribution may lead to inappropriate grid stretching in certain regions of the domain, and inadequate resolution of the flow field may occur in regions where the magnitude of the Hessian is close to zero. Heuristic remedies for these deficiencies have been proposed in the literature [16, 67], however, no explicit steps are taken in the current implementation to deal with them. It will be shown in Section 4.6 that the output–based scheme is able to mitigate the effects of these shortcomings while maintaining effective control of the functional error.

4.5.4 Output Error Control

The output–based adaptive parameters derived in Section 2.3.2 can be incorporated into an anisotropic grid–adaptive framework for functional outputs. The method proposed in this thesis is a modification of a Hessian–based adaptive procedure to use adjoint information for controlling the output error.

Metric Parameters

The Riemannian metric introduced in Section 4.5.2 contains three independent pieces of information that can be used for anisotropic grid adaptation. In its original form the metric is specified by the components (a, b, c). Through diagonalization, the metric can also be characterized by the parameters (h_1, h_2, θ), as shown in (4.44)–(4.45). In the present context it is convenient to decompose the metric into the parameters (H, β, θ), which represent the local size, stretching, and orientation of the elements, respectively. These parameters are defined by

$$(H, \beta, \theta) \equiv (h_2, h_1/h_2, \theta) \tag{4.49}$$

In the proposed output–based adaptive method, the stretching and orientation parameters, β and θ, are obtained from the Mach number Hessian whereas the local size parameter H is determined from the adjoint criteria.

Grid–Implied Metric

In a typical output–based adaptive simulation, several adaptive iterations are required in order to achieve grid convergence. Each iteration involves a flow and adjoint solution on the current grid, and an adaptive step in which a new grid is generated. For each element in the current grid the parameters (H', β', θ') are computed to indicate how the local size, stretching and orientation should change in the new grid. The stretching and orientation parameters for the new grid, β' and θ', are obtained directly from the Hessian–based metric. The new size parameter H', however, is obtained in a more implicit manner. It is not computed exclusively from the adjoint criteria, but rather as the product of an adjoint–based factor and the current element–size parameter H.

The following describes a method for determining the size parameter H associated with an element in the current grid. If it is assumed that the current grid is optimal in some metric, then the metric lengths of all the edges in the grid are constant. Correspondingly, the local metric value for an element can be approximated using (4.43). Specifically, consider the triangle shown in Figure 4-5. For each of its edges $k \in [1, 3]$, the following holds, approximately[2]:

$$a(\Delta x_k)^2 + 2b\Delta x_k \Delta y_k + c(\Delta y_k)^2 = 1. \tag{4.50}$$

[2] The appropriate constant on the right–hand–side of (4.50) is dependent on the standard employed by the particular grid generator. In BL2D, physical lengths are mapped onto a metric length of unity [35].

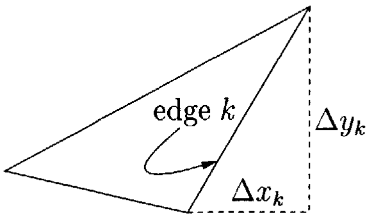


Figure 4-5: Typical triangle in the current grid.

This results in the system of equations

$$\begin{bmatrix} (\Delta x_1)^2 & 2\Delta x_1 \Delta y_1 & (\Delta y_1)^2 \\ (\Delta x_2)^2 & 2\Delta x_2 \Delta y_2 & (\Delta y_2)^2 \\ (\Delta x_3)^2 & 2\Delta x_3 \Delta y_3 & (\Delta y_3)^2 \end{bmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \tag{4.51}$$

which can be solved for the metric components (a, b, c). These components represent an average metric for the triangle. The size parameter H can be obtained from the largest singular value $\sigma_{\max}$ of the implied metric via[3]

$$H = \frac{1}{\sigma_{\max}^{1/2}} \tag{4.52}$$

The approximate grid–implied metric for a tetrahedron can be obtained in an analogous fashion. In three dimensions, the Riemannian metric is a symmetric positive–definite 3×3 matrix specified by 6 components. Tetrahedra have 6 edges over which 6 equations analogous to (4.50) can be written and solved for the 6 unknown components.

Adaptive Procedure

Prior to an output–based adaptive simulation, one must specify the functional of interest $f(U)$, a desired tolerance on the functional error e_o, and a scalar field for the

[3] The true metric is symmetric positive–definite; its eigenvalues are positive and equal to its singular values. The implied metric is not guaranteed to be positive definite, therefore, the size parameter H is computed in terms of the largest singular value, which is equal to the absolute value of the largest eigenvalue in magnitude.

Hessian calculations. For the viscous test cases in this dissertation, the Hessians are computed from the Mach number distribution.

A single iteration in the output–based procedure is comprised of the following steps:

1. Obtain the flow and adjoint solutions on the current grid.

2. For each element, compute the piecewise–constant Mach number Hessians using the quadratic reconstruction procedure outlined in Section 2.2.2.

3. For each element, compute the Hessian–based metric $\mathcal{M}_{\mathcal{H}}$ using (4.47), by diagonalizing the Hessian and taking the absolute value of its eigenvalues. Compute the desired new stretching and orientation parameters, β' and θ', using (4.44), (4.45) and (4.49).

4. For each element, estimate the grid–implied metric as described in Section 4.5.4, and compute the current element–size parameter H.

5. Compute the adjoint–based adaptation parameters η_g and η_k derived in Section 2.3.2. Specifically, $\eta_g = \varepsilon/e_o$ and $\eta_k = \varepsilon_k/\bar{e}_o$ for each element k, where $\bar{e}_o = e_o/N_e$ is the element target error, N_e is the total number of elements in the current grid, $\varepsilon = \sum_k \varepsilon_k$, and

$$\begin{aligned}\varepsilon_k = \frac{1}{2}\sum_{l(k)}\{ \quad & \left| \, [R_h^\Psi(\bar{L}_h^H \Psi_H)]_{l(k)}^T \, [Q_h^H U_H - L_h^H U_H]_{l(k)} \, \right| \\ + \quad & \left| \, [\bar{Q}_h^H \Psi_H - \bar{L}_h^H \Psi_H]_{l(k)}^T \, [R_h(L_h^H U_H)]_{l(k)} \, \right| \quad \}. \end{aligned} \tag{4.53}$$

   The prolongation operators $\bar{L}_h^H$ and $\bar{Q}_h^H$, modified for the viscous adjoint, are defined in Section 4.4.3.

6. For each element, use (2.26) to compute the desired new element size parameter H' from the current one H:

$$H' = H\left(\frac{1}{\eta_g \eta_k}\right)^{\omega}. \tag{4.54}$$

A value of $\omega = 1/4$ is chosen for the underrelaxation parameter for all the test cases considered. This value is obtained using a heuristic described in Section 2.3.2.

7. For each element, construct the new output–based metric $\mathcal{M}'$ for the next grid using the parameters (H', β', θ'). Transfer the elemental metric components to the nodes using area–weighted averages of the piecewise–constant values over the elements surrounding each node.
8. Input the nodal metrics into the anisotropic grid generator and regenerate the grid.

The iterative process is considered converged when the change in the total number of nodes from one iteration to the next is on the order of 1% or less ($\sim$ 5% for very coarse grids). If the desired error tolerance e_o is very low relative to the resolution of the initial grid, excessive overrefinement may result in the early stages of the adaptive process. To overcome this difficulty, a modest (large) value for e_o can be prescribed initially and then gradually ramped down to the desired error level over the course of several iterations.

Functional Correction

Once the adaptive procedure has converged, (2.12) can be used to compute the corrected functional. Specifically,

$$f_h(U_h) \approx f_h(Q_h^H U_H) - (\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H). \tag{4.55}$$

4.5.5 Grid Generator

The BL2D grid generator [35] is used to anisotropically adapt the grids for the viscous test cases in this work. The BL2D software package is a two–dimensional, adaptive, anisotropic grid generator developed at INRIA–Rocquencourt, France. The domain grid generator accepts metric values at the nodes of the current grid, as input, and generates a new grid using a generalized Delaunay–type algorithm [12]. Domain boundaries are represented by cubic splines. BL2D is incorporated into the current adaptive framework using shell scripts.

4.6 Results

Several viscous test cases are presented to demonstrate the performance of the proposed anisotropic grid–adaptive method for functional outputs. An adaptive method based exclusively on the Mach–number Hessian (see Section 4.5.3) is also implemented for the purposes of comparatively assessing the proposed output-based scheme. Pure Hessian–based adaptation is essentially what is used in references [12, 16, 19, 27, 48, 67] although their methodologies and implementations may differ to varying degrees. The proposed output-based method is shown to provide substantial improvement in terms of output accuracy and computational efficiency relative to pure Hessian–based adaptation.

4.6.1 Functional Correction Results

Two test cases are presented to demonstrate the output correction procedure for low–to–moderate Reynolds number (laminar) Navier–Stokes simulations. Hierarchies of embedded grids are constructed to conduct error convergence tests. No adaptation is performed for these test cases.

Re = 10, $M_\infty = 0.38$ Cylinder

Flow over a cylinder (Re = 10, $M_\infty = 0.38$) is simulated on 4 embedded grids. The coarsest grid contains 800 nodes and serves as the seed grid in the hierarchy. Subsequent grids are constructed by subdividing each of the triangles in the previous grid into 4 self–similar triangles. The farfield boundary is placed at 22.5 diameters. Figure 4-6 shows plots of the computable correction $\left|(\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)\right|$, the true error $|\delta f_h| = \left|f_h(Q_h^H U_H) - f_h(U_h)\right|$, and the remaining error after correction $\left|\delta f_h - (\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)\right|$, for each grid. Errors are measured with respect to the

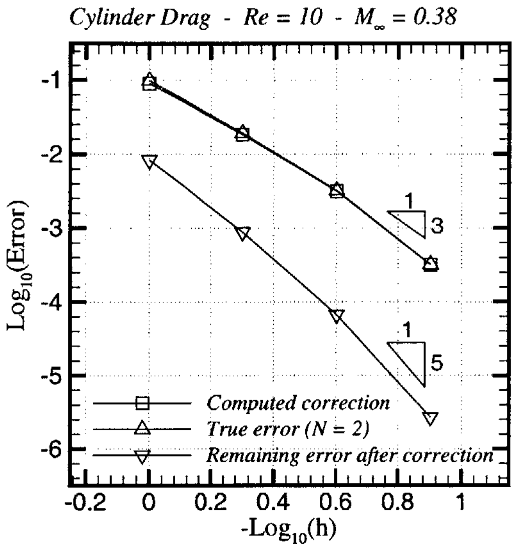


Figure 4-6: Cylinder test case: Re = 10, $M_\infty = 0.38$. Convergence of the error in the computed drag, the correction term, and the remaining error after correction. Errors are measured with respect to fine–grid values corresponding to $N = 2$.

fine grids ($N = 2$) associated with each base grid in the hierarchy. The computable correction is shown to approximate the true error to high accuracy. The $\sim \mathcal{O}(h^5)$ convergence of the remaining error demonstrates that the corrected functionals converge faster than the corresponding perturbed functionals on the fine grid, which are converging at approximately $\mathcal{O}(h^3)$.

Re $= 10,000$, $M_\infty = 0.4$ NACA 0005 Airfoil

Analogous results for flow over a NACA 0005 airfoil (Re $= 10,000$, $M_\infty = 0.4$, $\alpha = 0^o$) are presented. A hierarchy of 8 grids is constructed and used to conduct error convergence tests. The coarsest grid contains 507 nodes and serves as the seed grid ($n = 1$) in the hierarchy. The nth grid ($n \in [2,8]$) is constructed by subdividing each of the triangles in the original grid into n^2 self–similar triangles. The farfield boundary is placed at 25 chords. The plots on the right side of Figure 4-7 show the first ($n = 1$) and fifth ($n = 5$) grid in the hierarchy. The left side of the figure shows a plot of the computable correction $\left|(\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)\right|$, the true error $|\delta f_h| = \left|f_h(Q_h^H U_H) - f_h(U_h)\right|$, and the remaining error after correction $\left|\delta f_h - (\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)\right|$, for each grid. Errors are measured with respect to the

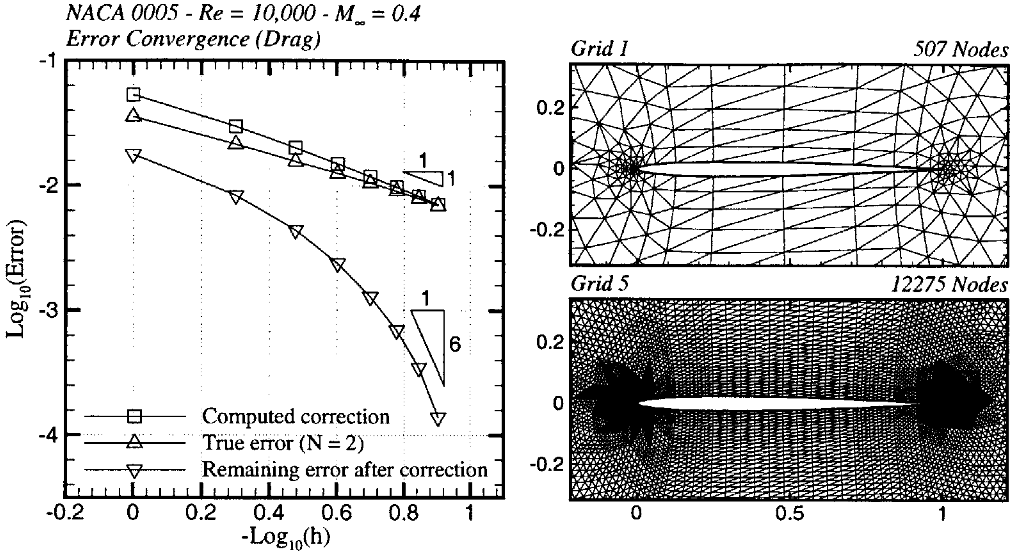


Figure 4-7: NACA 0005 Airfoil test case: Re = 10,000, M_∞ = 0.4, α = 0^o. Left plot: convergence of the error in the computed drag, the correction term, and the remaining error after correction. Errors are measured with respect to fine–grid values corresponding to $N = 2$. Right plot: the first and fifth grid used in the convergence study.

fine grids ($N = 2$) associated with each base grid in the hierarchy. The corrected functionals are shown to converge faster than the associated perturbed functionals. It is likely, however, that the finest grids in the hierarchy are not yet in the asymptotic region for this problem, and that the remaining error curve would eventually flatten to a lower rate of convergence if finer grids were added to the hierarchy. Nevertheless, the convergence plot demonstrates that the adjoint correction procedure is capable of yielding substantial improvements in functional accuracy.

4.6.2 Re $= 5,000$, $M_\infty = 0.5$, $\alpha = 0^o$ NACA 0012 Airfoil

A series of adaptive simulations are performed for flow over a NACA 0012 airfoil with free stream conditions Re $= 5,000$, $M_\infty = 0.5$ and $\alpha = 0^o$. The functional of interest is the drag coefficient. The initial grid in the adaptive simulations is shown in Figure 4-8. The farfield boundary is placed at 15 chords. This test case is representative of 'cruise'–type conditions (i.e. no significant boundary layer separation).

Output–based simulations are performed for five different error tolerances: $e_0 =$ 0.005, 0.0025, 0.00125, 0.000625, and 0.0003125. The final adapted grids range in size

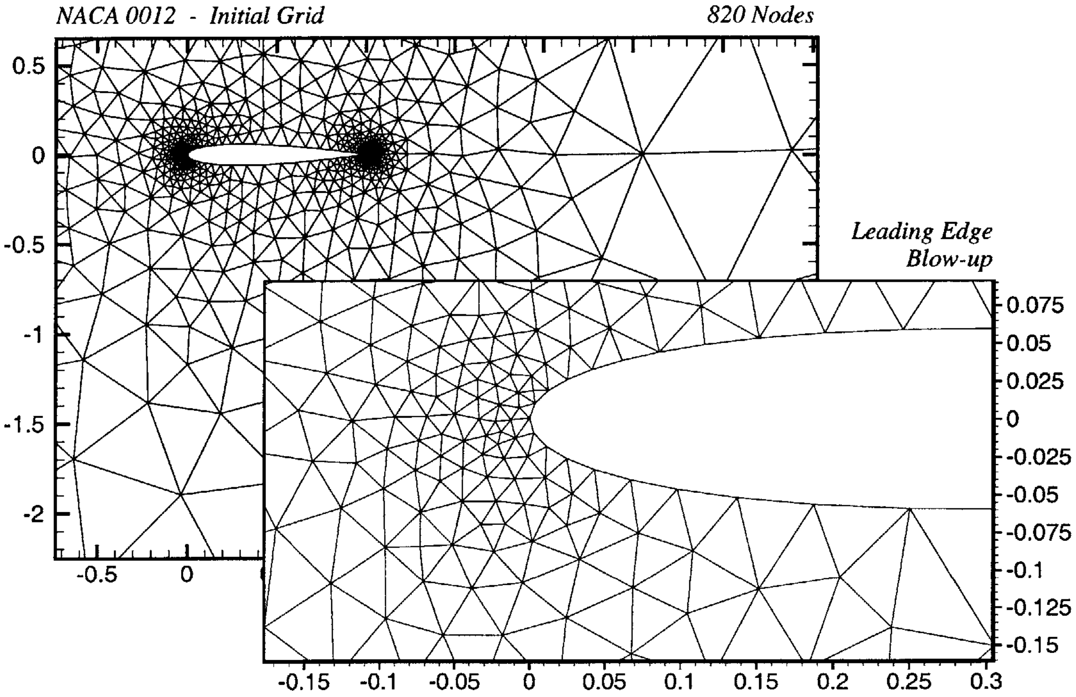


Figure 4-8: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 0^o$. Initial grid in the adaptive runs for this test case.

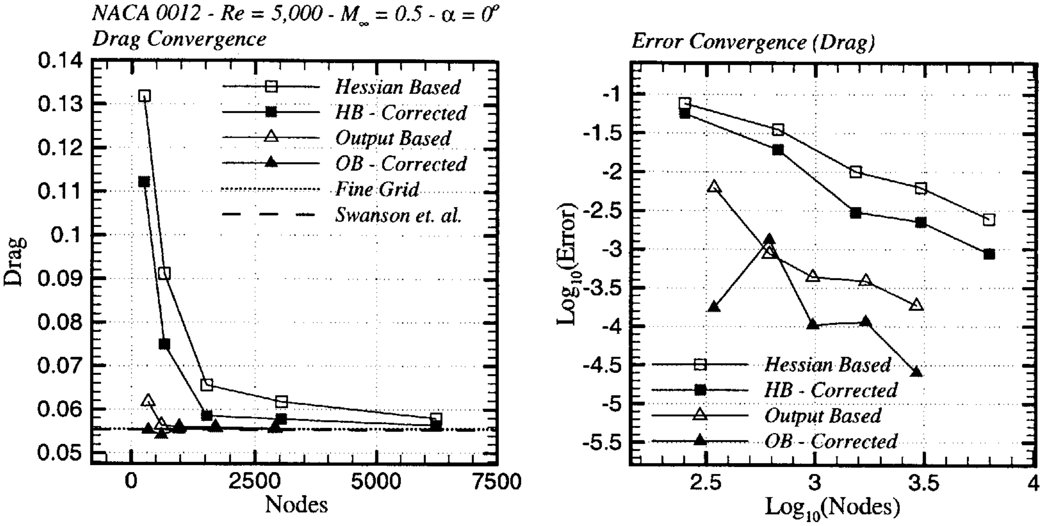


Figure 4-9: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 0^o$. Plots of the base and corrected drag (left), and corresponding errors (right) from a series of adaptive simulations. Adaptive runs are performed using the proposed output–based scheme and pure Hessian–based adaptation. Five different error tolerances are specified for the output–based simulations ranging from $e_0 = 0.005$ to $e_0 = 0.0003125$. Five different multiplicative factors are prescribed for the pure Hessian–based simulations ranging from $\kappa = 25$ to $\kappa = 400$. The fine–grid value shown in the left plot is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the output–based simulations. Errors in the right plot are measured with respect to this fine–grid value. The Swanson *et al.* value is obtained from reference [60].

from 343 nodes ($e_0 = 0.005$) to 2904 nodes ($e_0 = 0.0003125$). Pure Hessian–based adaptation is performed for five multiplicative factors: $\kappa = 25$, 50, 100, 200, and 400. The final grids range in size from 251 nodes ($\kappa = 25$) to 6235 nodes ($\kappa = 400$).

The left plot in Figure 4-9 shows the base drag $f_H(U_H)$ and corrected drag $f_h(Q_h^H U_H) - (\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)$ for each of the final adapted grids. The fine–grid value shown in the plot (hereafter denoted $f_h(U_h)|_{e_0=0.0003125}$) is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the output–based simulations. This fine grid contains 11435 nodes. The right plot shows the corresponding errors in the drag measured with respect to the fine–grid value $f_h(U_h)|_{e_0=0.0003125}$. The Swanson *et al.* value shown in the plot is obtained from reference [60]. Swanson *et al.* use a finite volume discretization that employs a structured C–type grid with 512 × 128 cells and farfield boundary at 10 chords.

The plots indicate that the output–based scheme offers a considerable improvement in output accuracy over pure Hessian–based adaptation for grids of comparable size. Furthermore, the adjoint–corrected functionals are consistently more accurate than their corresponding base values for both types of adaptation.

Figure 4-10 compares the final adapted grids from the $e_0 = 0.0003125$ output-based simulation (top) and the $\kappa = 200$ Hessian–based simulation (bottom). The inaccurate drag prediction from the Hessian–based simulation is attributed to insufficient grid resolution in the inviscid regions of the flow. Moreover, the boundary layer and wake regions are comparatively overrefined relative to the output-based grid. Note that in the output-based grid, the wake is not significantly refined beyond one chord–length downstream of the trailing edge. Discretization errors downstream of that point do not appreciably influence the error in the computed drag. Figure 4-11 presents plots of the Mach number distributions on the same grids. It is again evident that the Hessian–based solution is not well–resolved outside the boundary layer and wake regions due to insufficient grid resolution.

The near–field views in Figure 4-10 show grid anisotropy a small distance away from the airfoil boundary whereas the triangles immediately adjacent to the wall are essentially isotropic. This effect, which occurs in both the output–based and pure Hessian–based grids, is caused by layers of very small curvature or by curves of inflection in the Mach number close to the wall. Xia *et al.* [67] observed the same effect using their Hessian–based scheme prior to implementing a heuristic for improving the grid resolution at the wall. No measures are taken to prevent this effect in the present implementations of the output–based and pure Hessian–based adaptive procedures. Nevertheless, results indicate that this effect has not been a significant factor, in regards to output accuracy, whenever the output-based adaptive method has been used.

4.6.3 Re $= 5,000$, $M_\infty = 0.5$, $\alpha = 3^o$ NACA 0012 Airfoil

Adaptive simulations are performed for flow over a NACA 0012 airfoil with free stream conditions Re $= 5,000$, $M_\infty = 0.5$ and $\alpha = 3^o$. Under these conditions, the

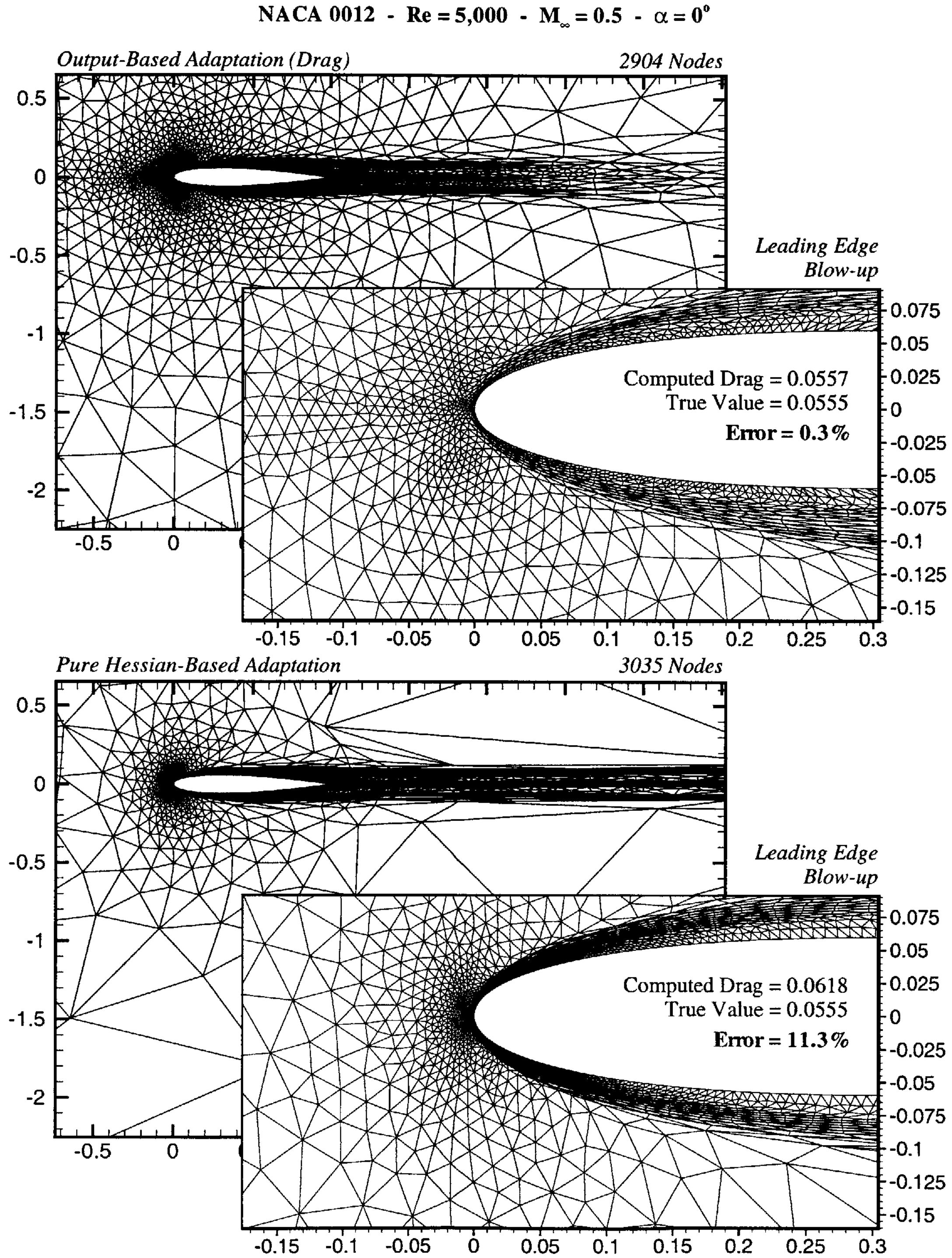


Figure 4-10: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 0^o$. Top: final adapted grid using output–based adaptation with a requested error level of $e_0 = 0.0003125$ for the drag. Bottom: final adapted grid using pure Hessian–based adaptation with $\kappa = 200$. The computed drag values shown in the plots correspond to base values without correction. The true value is the base drag computed on the fine grid ($N = 2$) associated with the top grid.

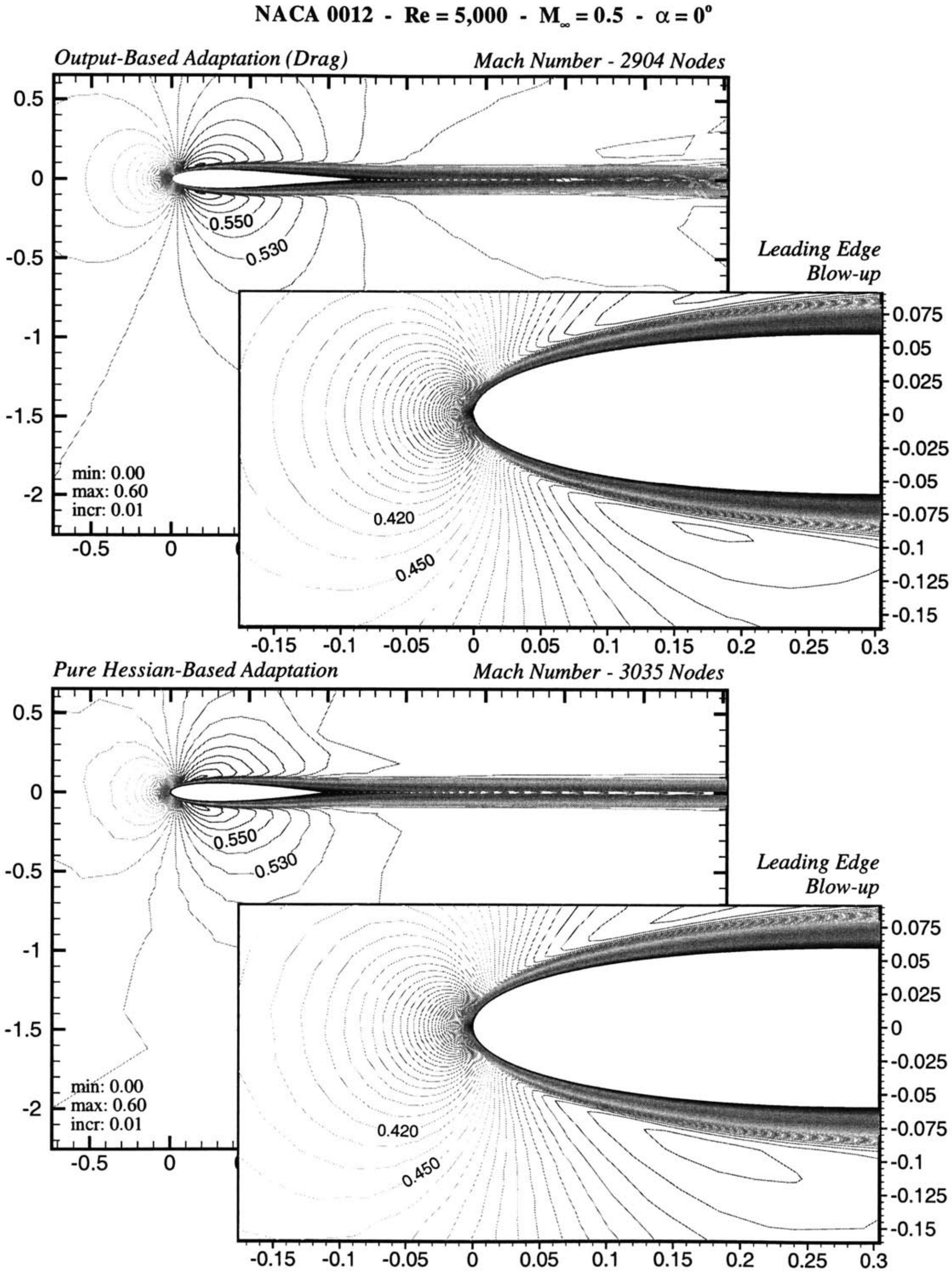


Figure 4-11: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 0^o$. Top: plot of the Mach number distribution on the final adapted grid using output–based adaptation. A requested error level of $e_0 = 0.0003125$ is prescribed for the drag. Bottom: Mach number distribution on the final adapted grid using pure Hessian–based adaptation with $\kappa = 200$.

flow separates from the suction side of the airfoil at approximately the 50% chord position while remaining attached on the lower surface. The functionals considered for this test case are the lift and drag coefficients. The initial grid in the adaptive simulations is shown in Figure 4-8. The farfield boundary is placed at 15 chords. This test case is included to demonstrate whether the proposed output–based scheme is able to resolve the separation bubble and correctly predict the aerodynamic forces.

Lift–based simulations are performed for five different error tolerances: $e_0 = 0.05$, 0.025, 0.0125, 0.00625, and 0.003125. The final adapted grids range in size from 392 nodes ($e_0 = 0.05$) to 4745 nodes ($e_0 = 0.003125$). Drag–based simulations are performed for $e_0 = 0.005$, 0.0025, 0.00125, 0.000625, and 0.0003125 with final grids ranging in size from 358 nodes ($e_0 = 0.005$) to 3479 nodes ($e_0 = 0.0003125$). Pure Hessian–based adaptation is performed for five multiplicative factors: $\kappa = 25$, 50, 100, 200, and 400. The final grids range in size from 263 nodes ($\kappa = 25$) to 6395 nodes ($\kappa = 400$). The left plot in Figure 4-12 shows the base and corrected lift values from the final adapted grids using lift–based adaptation and Hessian–based adaptation. The lift is also computed on a series of adapted grids using *drag*–based adaptation. The fine–grid value shown in the plot is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the lift–based simulations. This fine grid contains 18746 nodes. The right plot shows the corresponding errors in the lift measured with respect to the aforementioned fine–grid value. The Swanson *et al.* value shown in the plot is obtained from reference [60]. Swanson *et al.* use a finite volume discretization that employs a structured C–type grid with 512 × 128 cells and farfield boundary at 10 chords.

It is evident from Figure 4-12 that the lift–based scheme offers a considerable improvement in lift accuracy over pure Hessian–based adaptation for grids of comparable size. As expected, lift accuracy on the drag–based grids is inferior to that on the lift–based grids. Nevertheless, the drag–based grids generally yield superior lift accuracy relative to the Hessian–based grids.

Figure 4-12 is analogous to Figure 4-13 except that the drag is plotted in the latter figure. The fine–grid value shown in the left–hand plot of Figure 4-13 is computed on

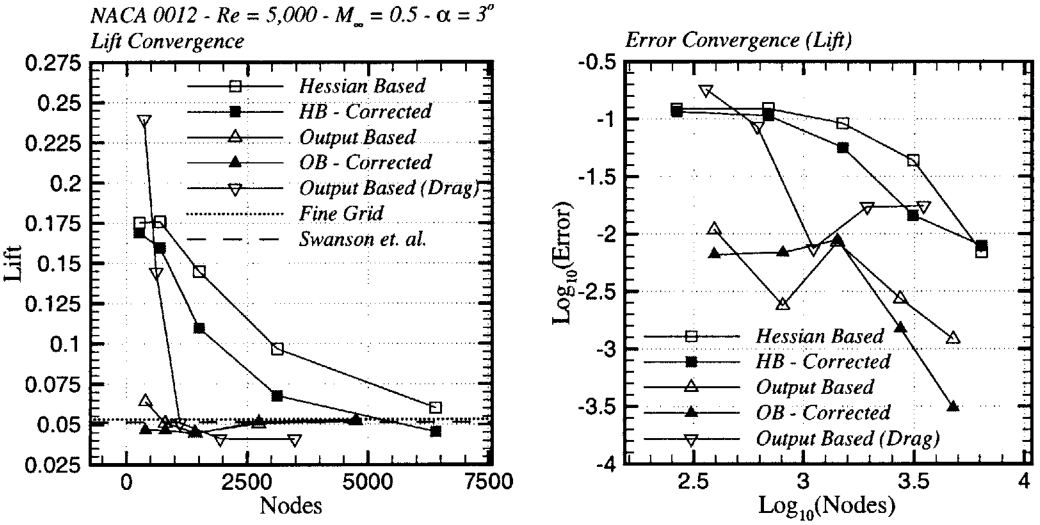


Figure 4-12: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 3^o$. Plots of the base and corrected lift (left), and corresponding errors (right) from a series of adaptive simulations. Five different error tolerances on the lift are specified for the output–based simulations ranging from $e_0 = 0.05$ to $e_0 = 0.003125$. Five different multiplicative factors are prescribed for the Hessian–based simulations ranging from $\kappa = 25$ to $\kappa = 400$. Also shown is the base lift computed on a series of adapted grids using output–based adaptation on the *drag*. The fine–grid value shown in the left plot is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the output–based simulations on the lift. Errors in the right plot are measured with respect to this fine–grid value. The Swanson *et al.* value is obtained from reference [60].

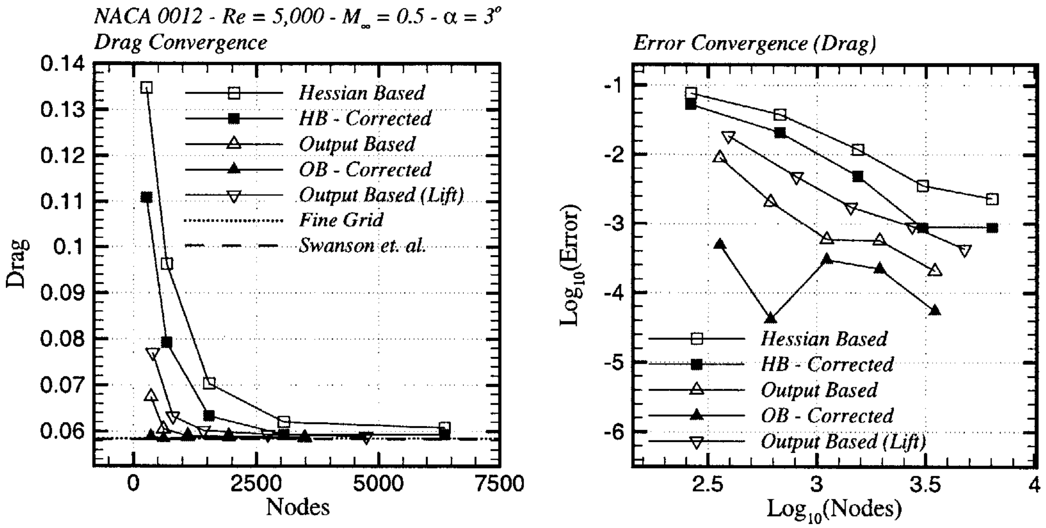


Figure 4-13: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 3^o$. Plots of the base and corrected drag (left), and corresponding errors (right) from a series of adaptive simulations. Five different error tolerances on the drag are specified for the output–based simulations ranging from $e_0 = 0.005$ to $e_0 = 0.0003125$. Five different multiplicative factors are prescribed for the Hessian–based simulations ranging from $\kappa = 25$ to $\kappa = 400$. Also shown is the base drag computed on a series of adapted grids using output–based adaptation on the *lift*. The fine–grid value shown in the left plot is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the output–based simulations on the drag. Errors in the right plot are measured with respect to this fine–grid value. The Swanson *et al.* value is obtained from reference [60].

a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the drag–based simulations. This fine grid contains 13716 nodes. Once again, the drag–based simulations are dramatically more accurate than the Hessian–based simulations with respect to the computed drag. Even the lift–based grids yield superior drag estimates relative to the Hessian–based grids.

Figure 4-14 compares the final adapted grids from the $e_0 = 0.00625$ lift-based simulation (top) and the $\kappa = 200$ Hessian–based simulation (bottom). The inaccurate lift prediction from the Hessian–based simulation is attributed to insufficient grid resolution in the inviscid regions of the flow, and to insufficient resolution of the separation zone on the suction side of the airfoil. The elements adjacent to the upper surface near the trailing edge are inappropriately large, resulting in unacceptable discretization errors in both the flow solution and in the geometric representation of the airfoil. The lack of resolution in the separation zone is further illustrated in Figure 4-15, which plots the Mach number distributions for the same grids. It is evident that the flow solution is poorly resolved in this portion of the Hessian–based grid relative to the corresponding region in the output–based grid. Moreover, comparison of the contour lines outside the boundary layer give further indication that the inviscid regions are relatively underresolved in the Hessian–based grid.

Figure 4-16 shows the final adapted grid using drag–based adaptation with $e_0 = 0.000625$. The grid is similar in appearance to the lift–based grids of comparable size.

4.6.4 Re $= 5,000$, $M_\infty = 0.5$, $\alpha = 3^o$ Two Element Airfoil

Adaptive simulations of flow past two NACA 0012 airfoil elements are presented. The leading edge of the downstream element is positioned four chord–lengths aft of the trailing edge of the upstream element. The flow conditions are Re $= 5000$ (based on the chord of a single element), $M_\infty = 0.5$ and $\alpha = 3^o$. At this angle of attack, the wake from the upstream element passes directly over the suction side of the downstream element.

If the grid possesses insufficient resolution downstream of the left element, discretization errors will cause its wake to diffuse prematurely and impinge upon the

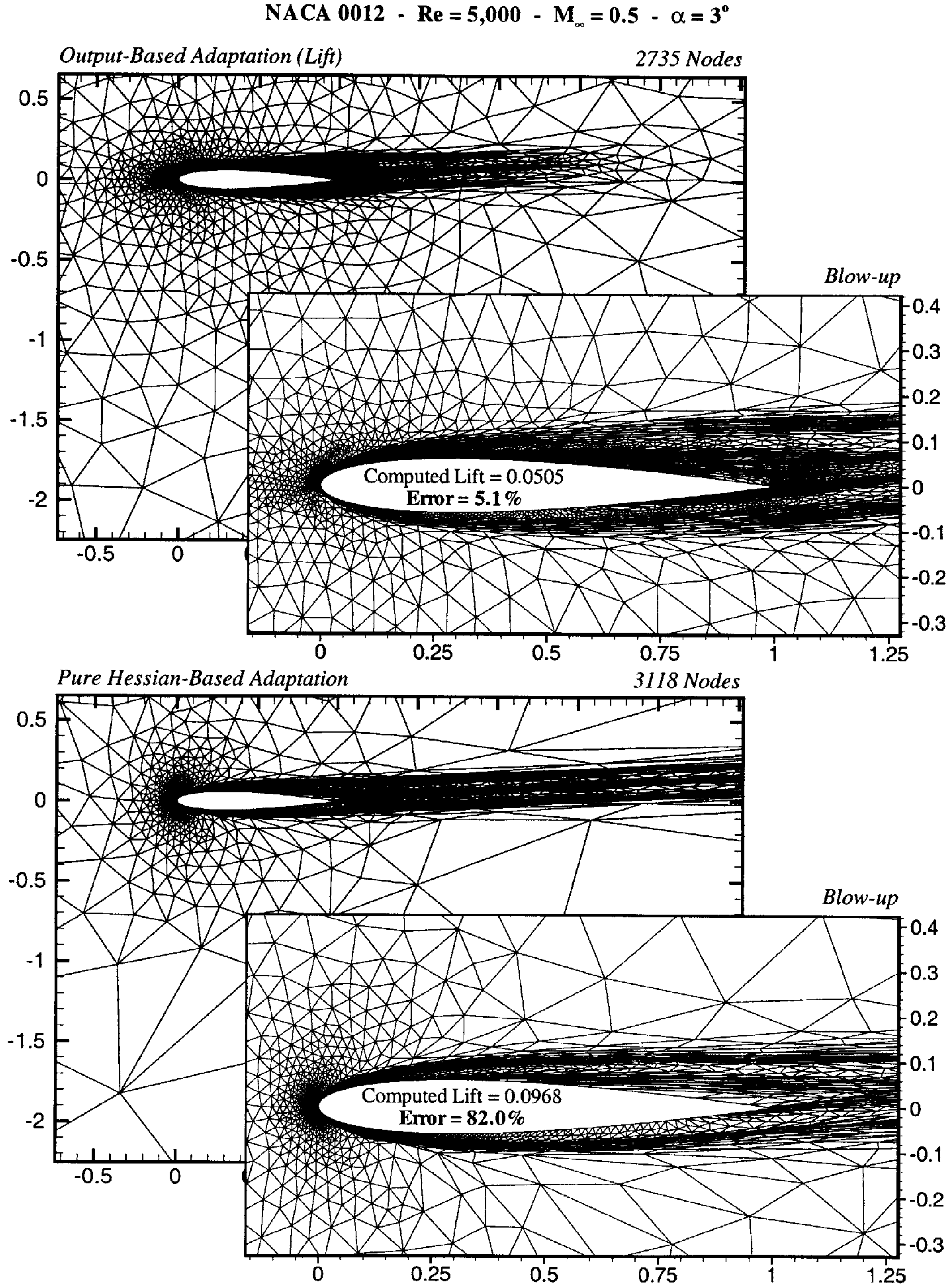
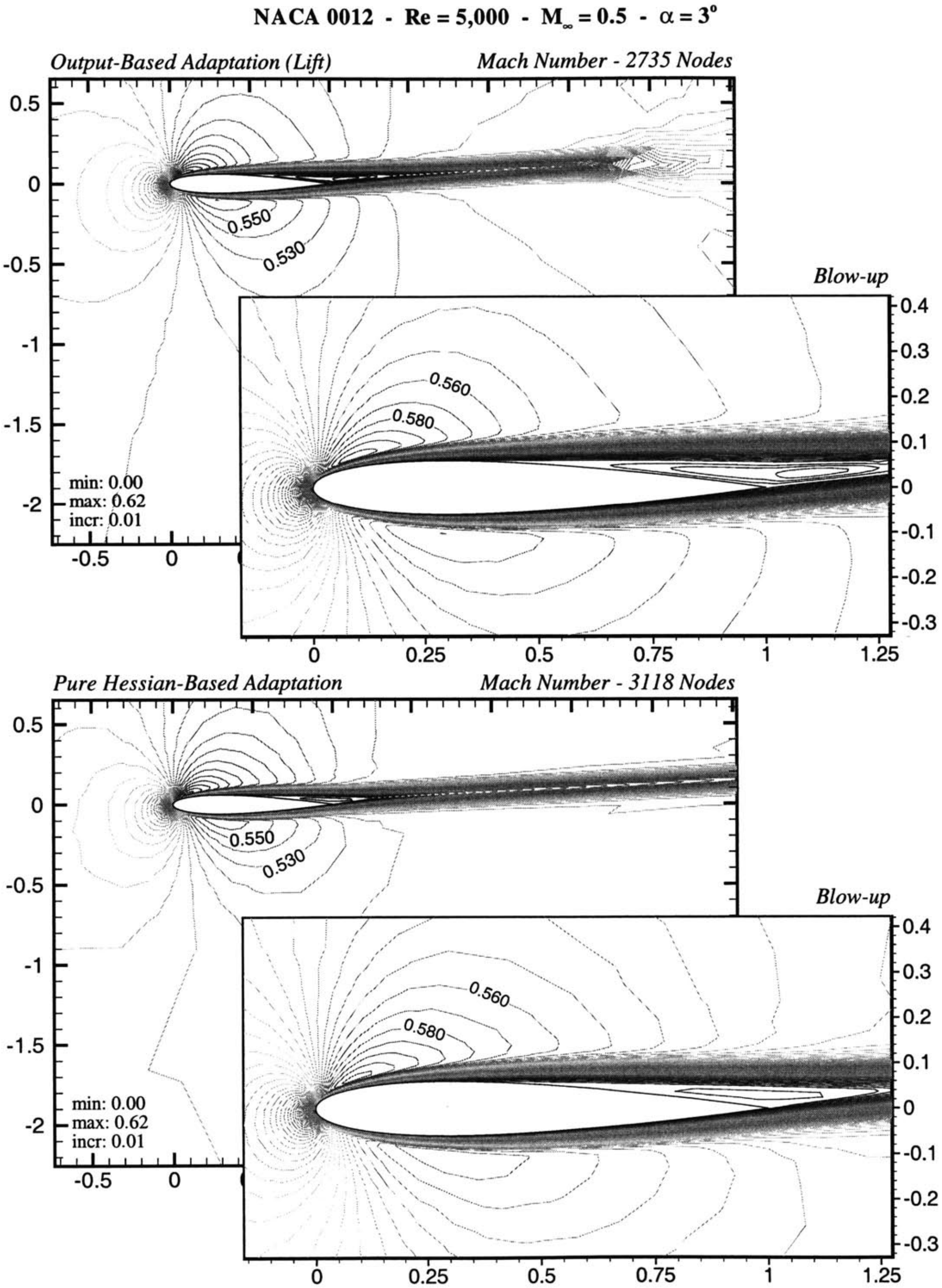


Figure 4-14: NACA 0012 Airfoil test case: Re = 5,000, $M_\infty = 0.5$, $\alpha = 3^o$. Top: final adapted grid using output–based adaptation. A requested error level of $e_0 = 0.00625$ is prescribed for the lift. Bottom: final adapted grid using pure Hessian–based adaptation with $\kappa = 200$. The computed lift values shown in the plots correspond to base values without correction. The error is measured with respect to the base lift computed on the fine grid ($N = 2$) associated with the finest grid in the output–based simulations for the lift.



NACA 0012 - Re = 5,000 - M_∞ = 0.5 - α = 3°

Output-Based Adaptation (Lift) — Mach Number - 2735 Nodes

Pure Hessian-Based Adaptation — Mach Number - 3118 Nodes

Figure 4-15: NACA 0012 Airfoil test case: Re $= 5,000$, $M_\infty = 0.5$, $\alpha = 3^o$. Top: plot of the Mach number distribution on the final adapted grid using output–based adaptation. A requested error level of $e_0 = 0.00625$ is prescribed for the lift. Bottom: Mach number distribution on the final adapted grid using pure Hessian–based adaptation with $\kappa = 200$.

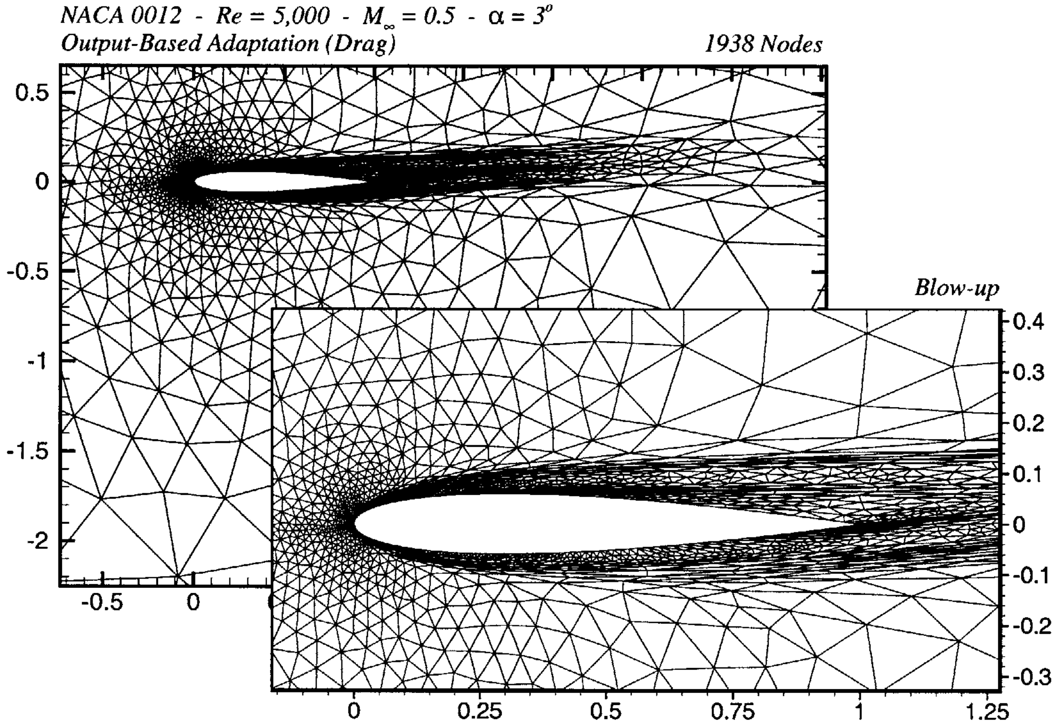

NACA 0012 - Re = 5,000 - M_∞ = 0.5 - α = 3°
Output-Based Adaptation (Drag)
1938 Nodes
0.5
0
-0.5
-1
-1.5
-2
-0.5
0
Blow-up
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.25
0.5
0.75
1
1.25

Figure 4-16: NACA 0005 Airfoil test case: Re = 5,000, M_∞ = 0.5, $\alpha = 3^o$. Final adapted grid using output–based adaptation. A requested error level of $e_0 = 0.000625$ is prescribed for the drag.

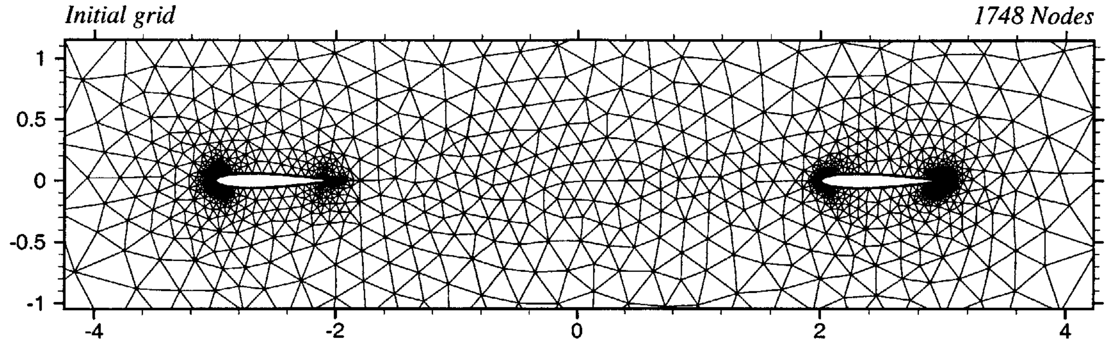


Figure 4-17: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Initial grid in the adaptive runs for this test case.

downstream element. This, in turn, will adversely effect the accuracy of certain derived quantities associated with the downstream element, such as the computed lift or drag. If the output of interest is the drag on the *upstream* element only, then the wake structure several chord–lengths downstream of the left element is of little concern since it does not influence the left–element drag to any appreciable extent. Similarly, discretization errors near the downstream element will have little impact on the accuracy of the upstream–element drag. Alternatively, if the drag on the *downstream* element is important, it will be necessary to resolve the flow around the upstream element and to resolve its wake up to and beyond the downstream element.

Adaptive simulations are performed to demonstrate the ability of the output–based scheme to provide appropriate grid resolution for three different outputs. The output–based scheme is applied, independently, to the left–element drag, the right–element drag, and the total drag (both elements). In each case, the prescribed error tolerance on the output is $e_o = 0.0005$. For comparison, an additional simulation is performed using pure Hessian–based adaptation.

The initial grid for all four adaptive runs is shown in Figure 4-17. The farfield boundary is placed at 15 chords. Figure 4-18 shows the final adapted grids for each case and Figure 4-19 presents near–field views of the same grids. Figure 4-20 shows the corresponding Mach number distributions computed on these grids.

Output–based adaptation on the left–element drag alone produces a final grid with limited wake resolution beyond two chord–lengths downstream of the left element. As eluded to previously, the left–element drag is relatively insensitive to discretization

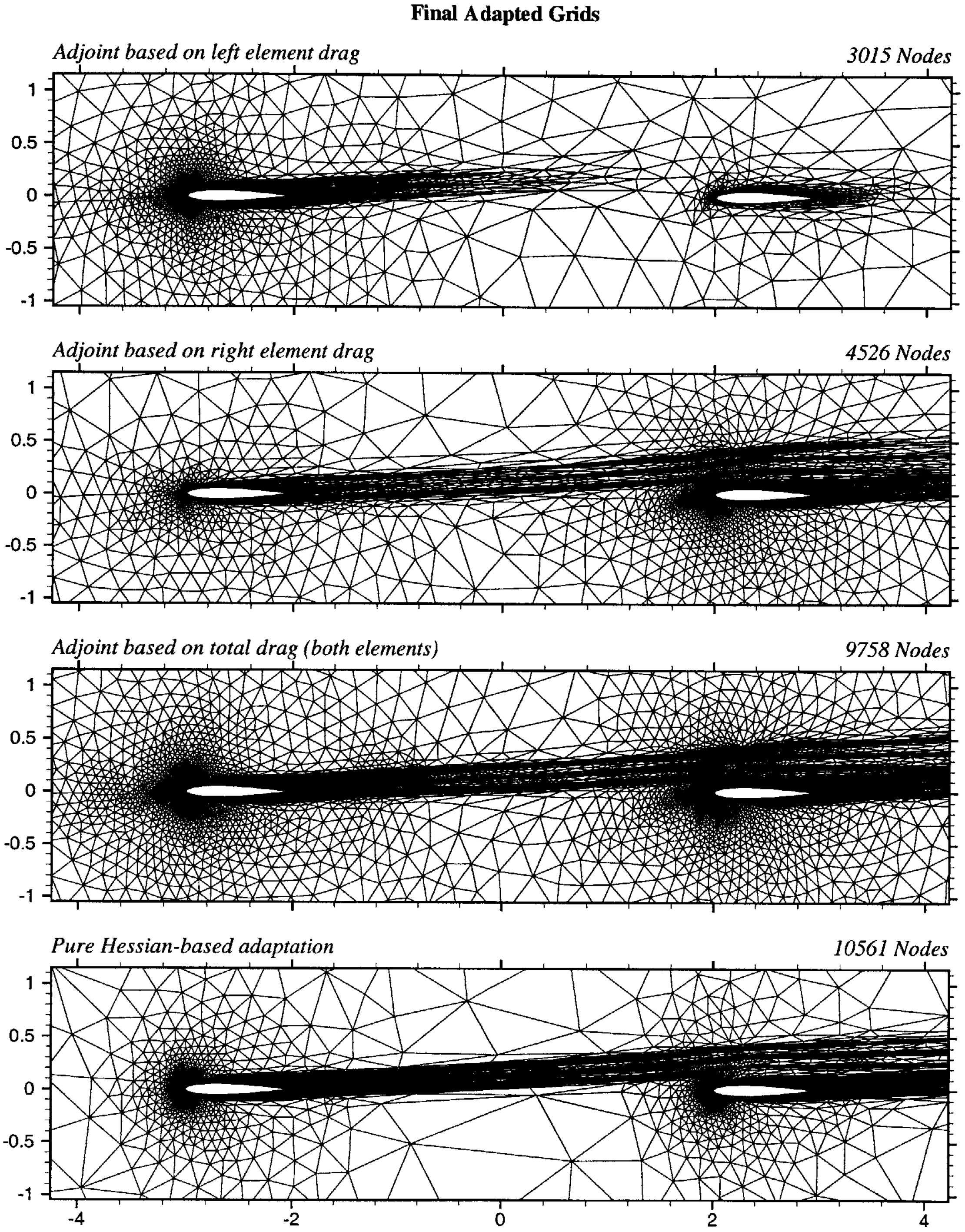

Final Adapted Grids
Adjoint based on left element drag
3015 Nodes
Adjoint based on right element drag
4526 Nodes
Adjoint based on total drag (both elements)
9758 Nodes
Pure Hessian-based adaptation
10561 Nodes
1
0.5
0
-0.5
-1
-4
-2
0
2
4

Figure 4-18: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Final adapted grids from four different adaptive runs. The proposed output-based method is applied to the left-element drag (top), the right-element drag (second from top), and the total drag (third from top). The bottom plot corresponds to pure Hessian–based adaptation.

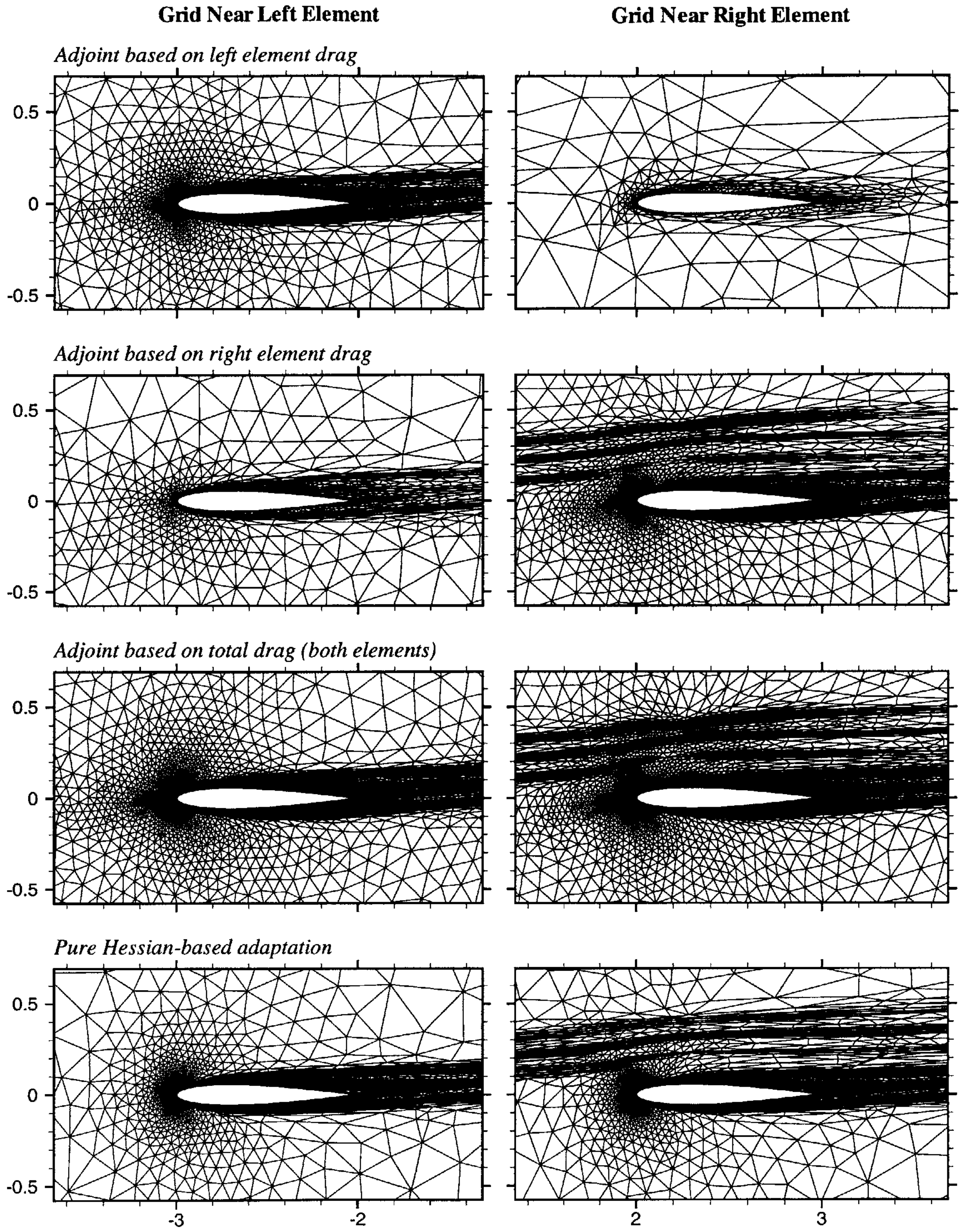

Grid Near Left Element
Grid Near Right Element
Adjoint based on left element drag
Adjoint based on right element drag
Adjoint based on total drag (both elements)
Pure Hessian-based adaptation
0.5
0
-0.5
-3
-2
2
3

Figure 4-19: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Near–field view of final adapted grids from four different adaptive runs. The left column (respectively, right column) of plots shows the grid near the upstream element (respectively, downstream element). The proposed output–based method is applied to the left–element drag (top row), the right–element drag (second from top), and the total drag (third from top). The bottom row corresponds to pure Hessian-based adaptation.

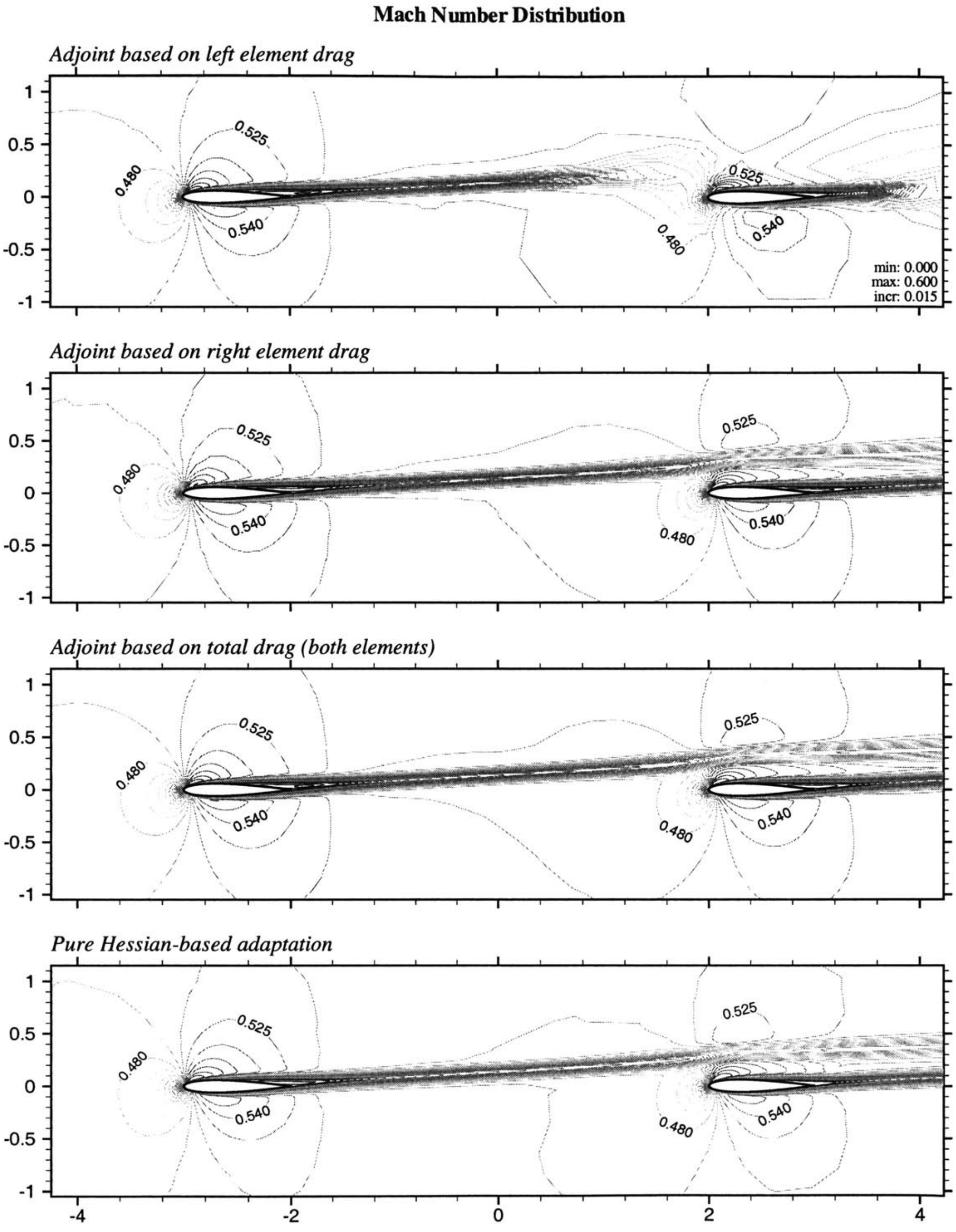


Figure 4-20: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Contour plots of the Mach number distribution from four different adaptive runs. The proposed output–based method is applied to the left-element drag (top), the right-element drag (second from top), and the total drag (third from top). The bottom plot corresponds to pure Hessian–based adaptation.

errors downstream of that point. Correspondingly, the grid near the downstream element is only marginally refined.

Output–based adaptation on the right–element drag produces a grid with significant wake resolution over the entire length of the inter–element gap, and beyond. The drag on the downstream element is more strongly effected by discretization errors in the vicinity of the upstream element. The adaptive algorithm responds accordingly by providing moderate grid resolution around the upstream element.

Output–based adaptation applied to the total drag produces significant grid resolution near both elements. The wake from the upstream element is resolved well past the downstream element.

Adaptation based on the Mach number Hessian yields a final grid with higher resolution in the leading edge and boundary layer regions relative to any of the output–based simulations. However, as seen in previous test cases, the inviscid regions further away from the elements are relatively underresolved without the adjoint error control. The multiplicative factor for this case is $\kappa = 325$.

The left–element, right–element, and total drag computed from each adaptive simulation is presented in Table 4.1. Numbers in brackets correspond to corrected values using the associated adjoint correction term (see Section 4.5.4). The drag values from the adaptive simulations are compared with the drag computed on a single fine grid obtained by uniformly refining ($N = 2$) the adapted grid corresponding to output–based adaptation on both airfoil elements.

In each case, the output–based method is effective at ensuring that the computed output (after correction) from the final adapted grid meets the imposed error tolerance. The adaptive algorithm resolves only those regions of the flow that are crucial for accurately computing the prescribed output. For the case where the output–based method is applied to the left–element drag alone, accuracy in the right–element drag is sacrificed in favor of a considerable reduction in grid size. This effect is not as severe in the opposite case, when the output–based method is applied to the right–element drag alone, for reasons outlined earlier. It is evident that for grids of comparable size, output–based adaptation on both elements yields drag estimates that are dra-

Grid			Drag (Error)		
Adaptive Criteria		Nodes	Left Element	Right Element	Total
Pure Hessian-Based		10561	0.06049 (4.2%)	0.06148 (7.5%)	0.1220 (5.8%)
Output Based	Left-Element Adjoint	3015	0.05847 (0.7%) **0.05821 (0.3%)**	0.07265 (27.0%)	0.1311 (13.7%)
	Right-Element Adjoint	4526	0.06018 (3.7%)	0.05800 (1.4%) **0.05741 (0.3%)**	0.1182 (2.6%)
	Based on Both Elements	9758	0.05820 (0.3%)	0.05756 (0.6%)	0.1158 (0.5%) **0.1153 (0.0%)**
Fine Grid		38574	0.05804	0.05722	0.1153

Table 4.1: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Computed drag from four different adaptive runs: output–based adaptation on the left-element drag, right-element drag, and total drag (both elements), and pure Hessian–based adaptation. Numbers in bold face correspond to corrected drag values using the associated adjoint correction term. Numbers in brackets are the percentage error in the drag measured with respect to the fine grid value. The fine grid is obtained by uniformly refining ($N = 2$) the adapted grid corresponding to output–based adaptation on both airfoil elements.

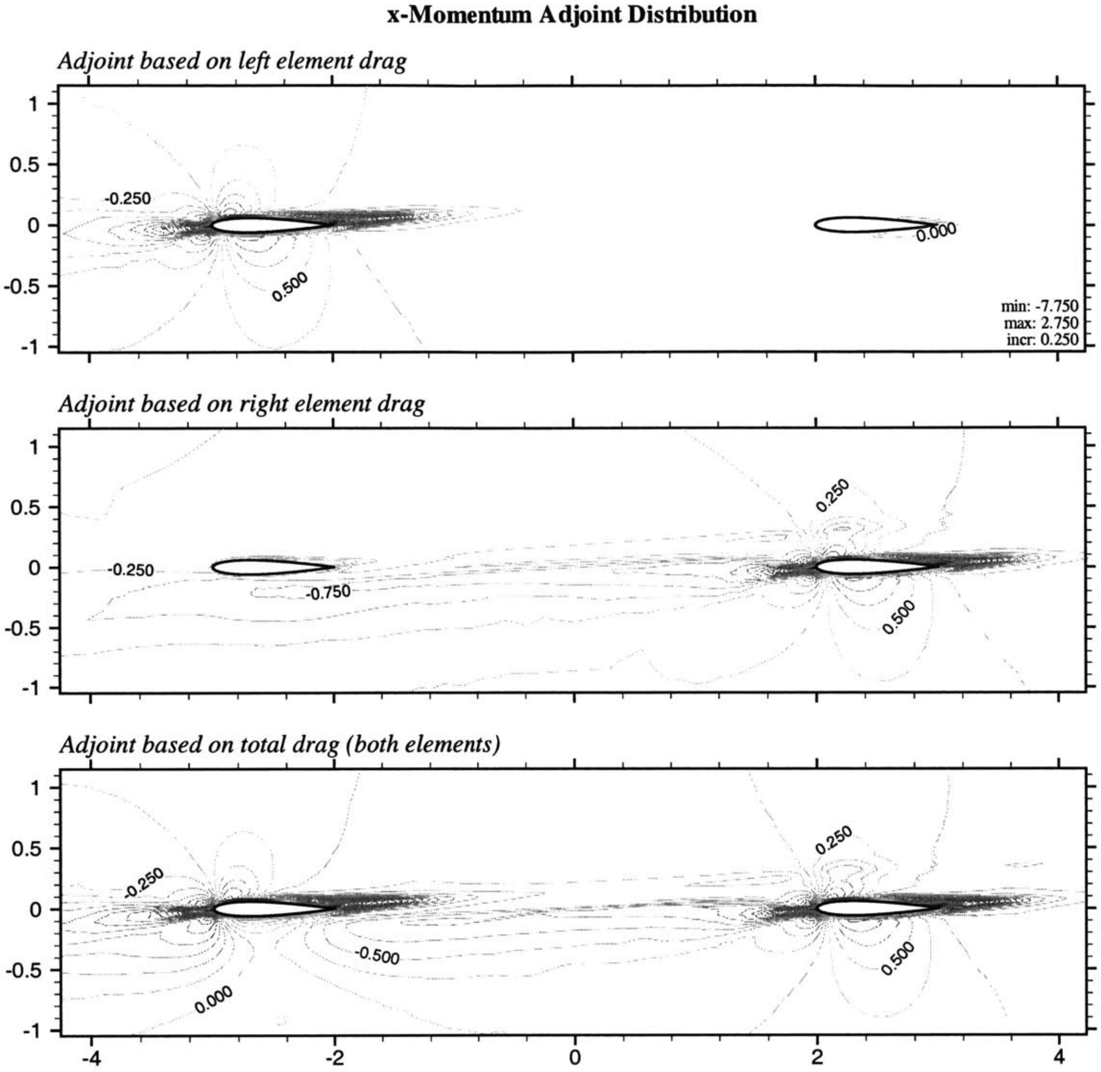


Figure 4-21: Two Element (NACA 0012) Airfoil test case: Re = 5000, $M_\infty = 0.5$, $\alpha = 3^o$. Contour plots of the x–momentum adjoint variable from three different adaptive runs. Output–based adaptation is on the left-element drag (top), the right-element drag (middle), and the total drag (bottom).

matically more accurate than those obtained from pure Hessian–based adaptation.

Figure 4-21 shows plots of the x–momentum adjoint distribution for each of the output–based adaptive simulations. With adaptation on the left element alone, the magnitude of the adjoint is close to zero near the downstream element. This is consistent with the fact that the accuracy of the left–element drag is essentially insensitive to discretization errors near the downstream element. However, with output adaptation on the right element alone, the adjoint takes on non–zero values near the upstream

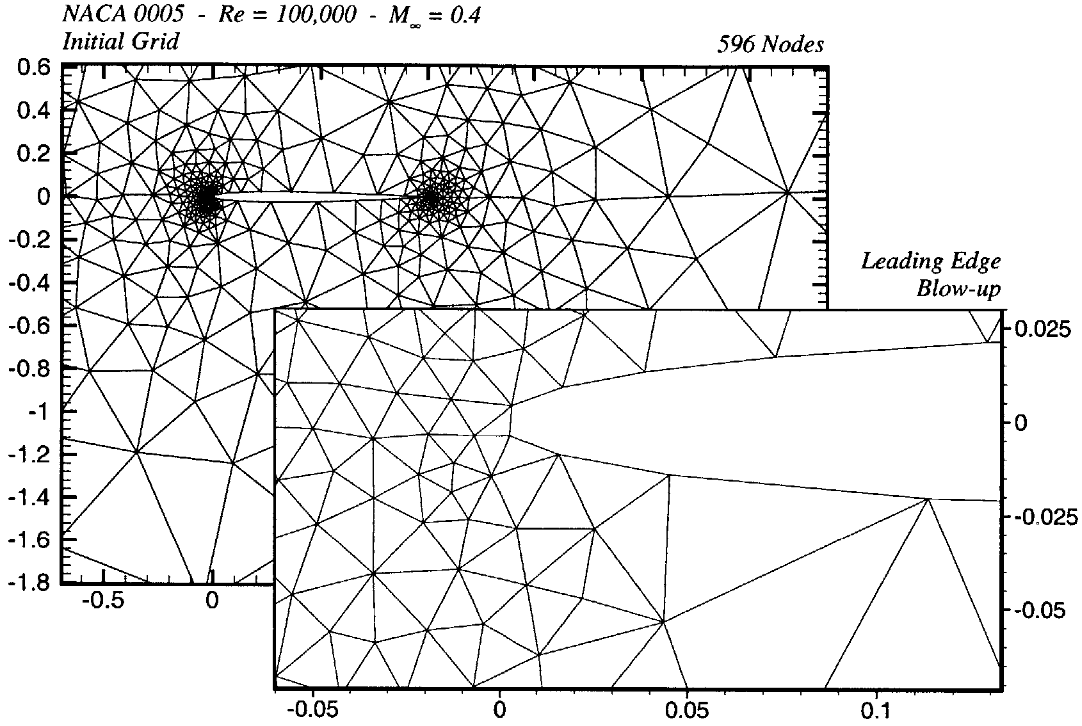


Figure 4-22: NACA 0005 Airfoil test case: Re = 100,000, $M_\infty = 0.4$, $\alpha = 0^o$. Initial grid in the adaptive runs for this test case.

element illustrating that the right–element drag is indeed sensitive to discretization errors near the upstream element.

4.6.5 Re = 100,000, $M_\infty = 0.4$ NACA 0005 Airfoil

Adaptive simulations are performed for flow over a NACA 0005 airfoil with free stream conditions Re = 100,000, $M_\infty = 0.4$ and $\alpha = 0^o$. This test case is included to assess the performance of the output–based method on a higher–Reynolds–number flow. In comparison to previous test cases, this flow is characterized by the presence of thinner boundary layers, resulting in increased stretching requirements on the grid in order to achieve comparable flow resolution.

The functional of interest is the drag coefficient. The initial grid in the adaptive simulations is shown in Figure 4-22. The farfield boundary is placed at 25 chords. Output–based simulations are performed for error tolerances of e_0 = 0.001, 0.0005,

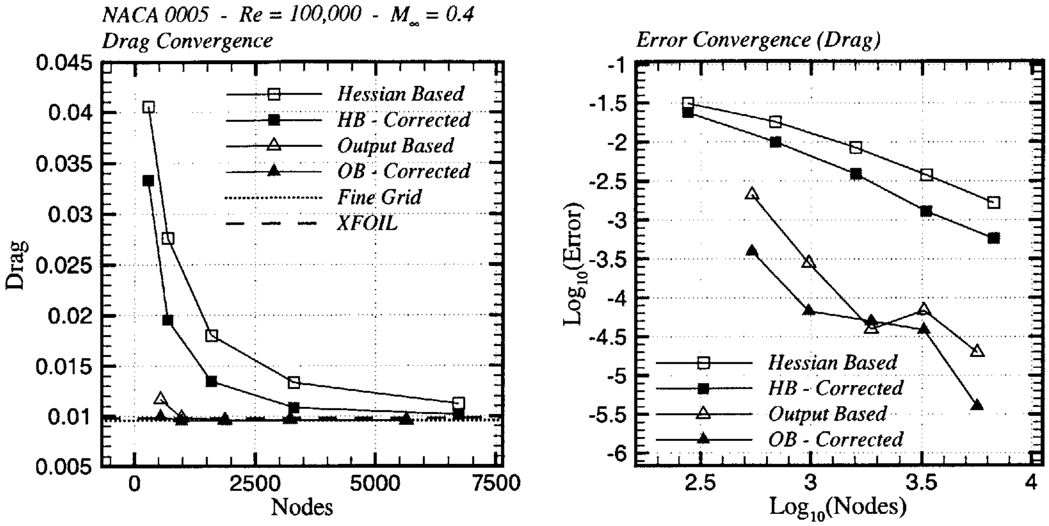


Figure 4-23: NACA 0005 Airfoil test case: Re = 100,000, M_∞ = 0.4, $\alpha = 0^o$. Plots of the base and corrected drag (left), and corresponding errors (right) from a series of adaptive simulations. Five different error tolerances are specified for the output–based simulations ranging from $e_0 = 0.001$ to $e_0 = 0.0000625$. Five different multiplicative factors are prescribed for the Hessian–based simulations ranging from $\kappa = 25$ to $\kappa = 400$. Errors in the right plot are measured with respect to a single fine–grid ($N = 2$) value corresponding to the finest adapted grid from the output–based simulations. Also shown is the drag computed using XFOIL [20].

0.00025, 0.000125, and 0.0000625. The final adapted grids range in size from 539 nodes ($e_0 = 0.001$) to 5641 nodes ($e_0 = 0.0000625$). Pure Hessian–based adaptation is performed for multiplicative factors of κ = 25, 50, 100, 200, and 400. The final grids range in size from 275 nodes ($\kappa = 25$) to 6709 nodes ($\kappa = 400$).

The left plot in Figure 4-23 shows the base drag $f_H(U_H)$ and corrected drag $f_h(Q_h^H U_H) - (\bar{Q}_h^H \Psi_H)^T R_h(Q_h^H U_H)$ for each of the final adapted grids. The fine–grid value shown in the plot (hereafter denoted $f_h(U_h)|_{e_0=0.0000625}$) is computed on a single fine–grid ($N = 2$) corresponding to the finest adapted grid from the output–based simulations. This fine grid contains 22123 nodes. The right plot shows the corresponding errors in the drag measured with respect to the fine–grid value $f_h(U_h)|_{e_0=0.0000625}$. The left plot also shows the computed drag using XFOIL [20], an interactive program for the design and analysis of isolated airfoils. XFOIL uses a high–order panel method for incompressible potential flow interacted with an integral boundary layer formulation. A Karman–Tsien compressibility correction is incorporated.

The output–based method provides corrected and uncorrected drag estimates that are considerably more accurate than those predicted from the Hessian–based simulations. As in previous test cases, the poor performance of the Hessian–based method is attributed to insufficient grid resolution outside the boundary layer and wake regions. This is illustrated in Figures 4-24 and 4-25 which present plots of the final adapted grids and corresponding Mach number distributions for the $e_0 = 0.000125$ output-based simulation and the $\kappa = 200$ Hessian–based simulation.

Figure 4-26 shows plots of the final adapted grids from the $\kappa = 50$ Hessian–based simulation (left) and the $e_0 = 0.001$ output–based simulation (right). Below these are corresponding plots of the local adaptation parameter η_k defined by (2.25). A value of $e_0 = 0.001$ is used to compute η_k on each grid. The value of η_k for a given element k quantifies the magnitude of the local contribution to the estimated remaining error in the functional, expressed as a multiple of the target error for that element. In the Hessian–based grid, η_k reaches values greater than 150 indicating that the local error contribution is 150 times greater than the elemental target error. In the output–based grid, values of η_k are generally less than the smallest contour level in the plot, which is 10. Typical values of η_k in a fully converged output–based grid are $\mathcal{O}(1)$.

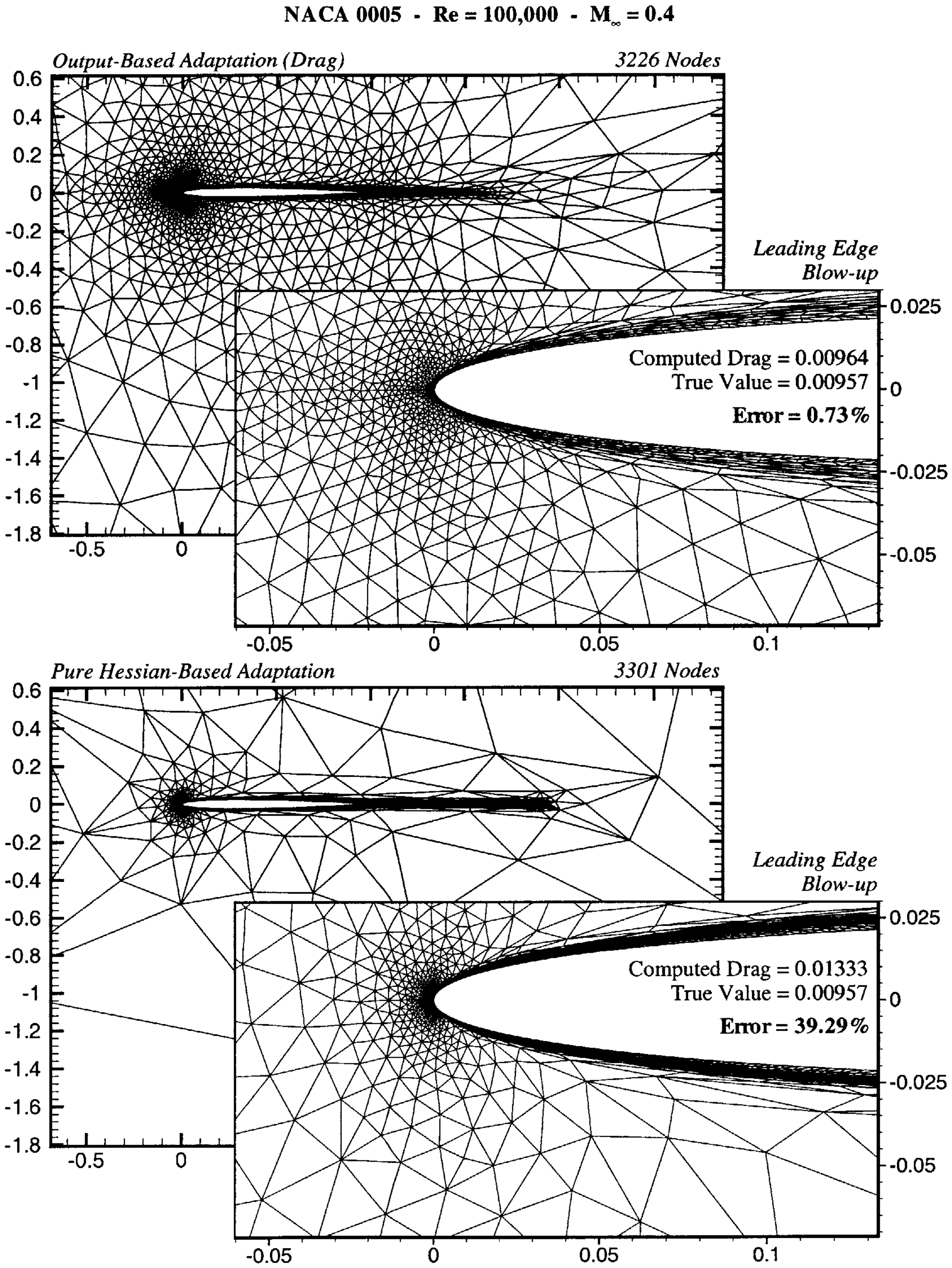


Figure 4-24: NACA 0005 Airfoil test case: Re = 100,000, $M_\infty = 0.4$, $\alpha = 0^o$. Top: final adapted grid using output–based adaptation with a requested error level of $e_0 = 0.000125$ for the drag. Bottom: final adapted grid using pure Hessian–based adaptation with $\kappa = 200$. The computed drag values shown in the plots correspond to base values without correction. The true value is the base drag computed on the fine grid ($N = 2$) associated with the finest grid in the output–based simulations for this test case.

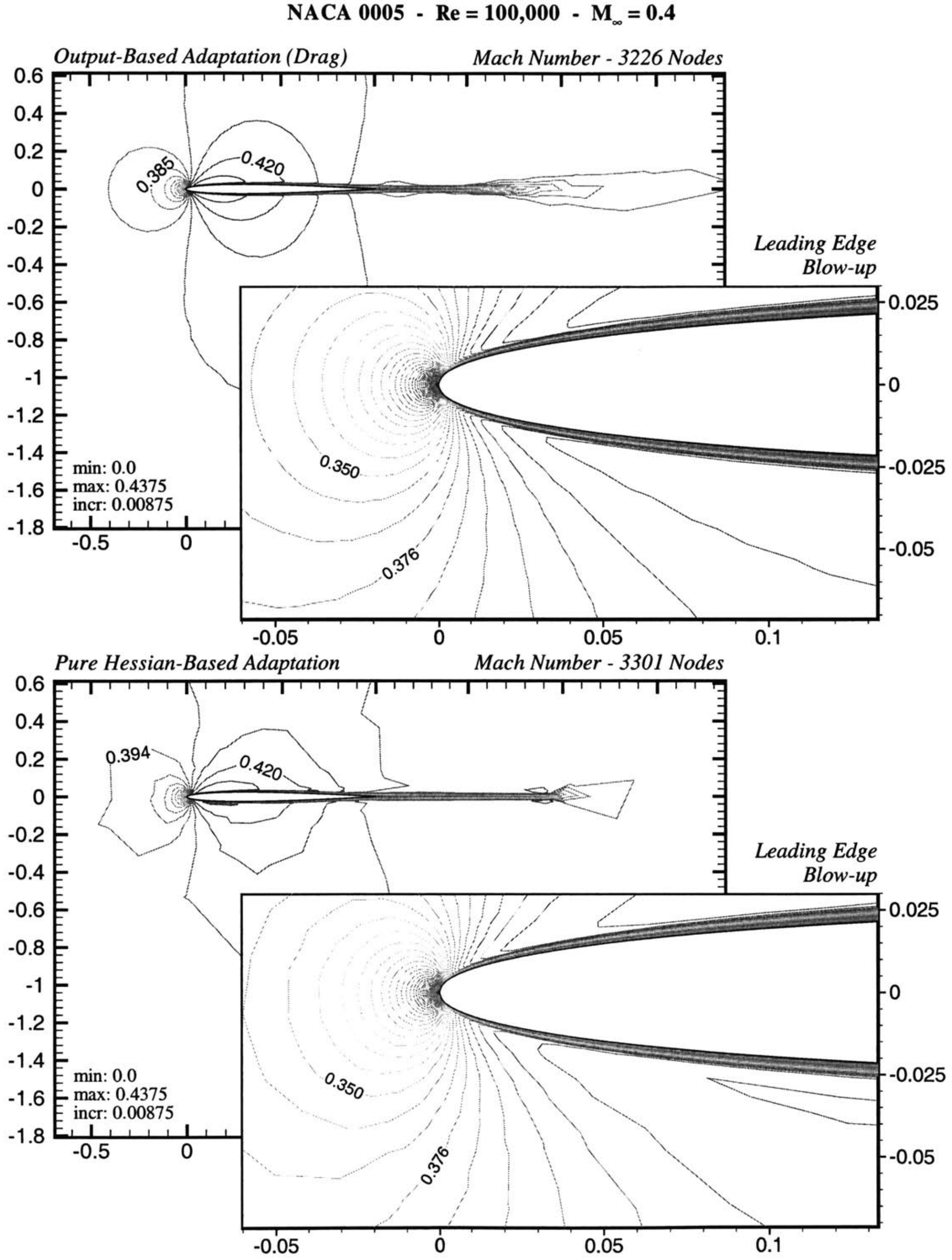


Figure 4-25: NACA 0005 Airfoil test case: Re $= 100,000$, $M_\infty = 0.4$, $\alpha = 0^o$. Top: plot of the Mach number distribution on the final adapted grid using output–based adaptation. A requested error level of $e_0 = 0.000125$ is prescribed for the drag. Bottom: Mach number distribution on the final adapted grid using pure Hessian–based adaptation with $\kappa = 200$.

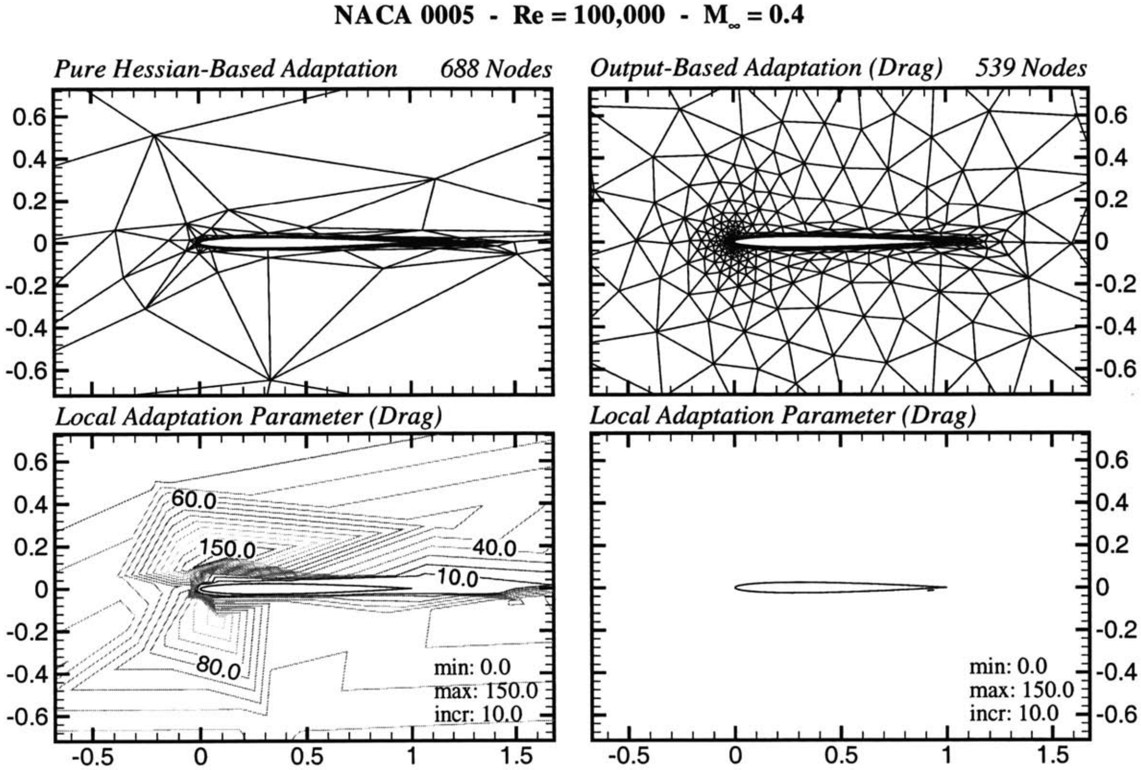


Figure 4-26: NACA 0005 Airfoil test case: Re $= 100,000$, $M_\infty = 0.4$, $\alpha = 0^o$. Top row: final adapted grids; bottom row: corresponding distributions of the local adaptation parameter η_k as defined by (2.25). The grid on the left is obtained using pure Hessian–based adaptation with $\kappa = 50$. The grid on the right is obtained using output–based adaptation on the drag with $e_0 = 0.001$.

Chapter 5

Conclusion

5.1 Summary

An error correction and grid adaptive procedure was presented for improving the accuracy of functional outputs from numerical simulations of compressible flows. The procedure is based on a discrete adjoint formulation in which the estimated error in the functional can be directly related to the local residual errors of both the primal and adjoint solutions. The adaptation/correction procedure was demonstrated by application to two–dimensional inviscid and viscous (laminar) test cases using standard finite volume discretizations. The procedure was also applied to a Galerkin finite element discretization of scalar convection–diffusion.

An isotropic grid refinement procedure was outlined and applied to a series of subsonic, transonic, and supersonic inviscid test cases. Comparisons were made with a commonly–used adaptive scheme [6, 64] that employs indicators based on the local magnitude of the second derivatives of the static pressure or Mach number. In many cases, this scheme either failed to self-terminate or produced erroneous values for the predicted functional at termination. For every test case considered, the proposed output-based method succeeded in self-terminating and surpassed the prescribed accuracy level for the chosen functional.

An anisotropic grid–adaptive procedure for functional outputs was presented and applied to a series of laminar Navier–Stokes test cases for Reynolds numbers ranging from 5,000 to 100,000. The output–based procedure can be viewed as a merging of Hessian–based adaptation with output error control. The Hessian of a prescribed scalar is used to determine the local stretching and orientation characteristics of the grid, while adjoint–based criteria are used to establish the local element size. The proposed output–based method was shown to compare very favorably in terms of output accuracy and computational efficiency relative to pure Hessian–based adaptation.

5.2 Contributions

1. **Output-based adaptation for finite volume methods.**

   Adaptive methods for functional outputs have been successfully implemented within the framework of the finite element method [9, 10, 13, 23, 30, 38, 46, 47, 51]. The current work is perhaps the first implementation of an output-based adaptive method for compressible flow simulations within a finite volume framework. The proposed output–based method has been implemented for quasi-one-dimensional inviscid flows [62] and two-dimensional inviscid flows [63]. This dissertation presents the Navier–Stokes implementation for the first time. Müller and Giles [40] have recently implemented an output-based adaptive method for two-dimensional Euler flows using a finite volume discretization. Their adaptive indicators differ from those used in the present method (see Section 2.3.1 for further details).

   The adaptive criteria used in the present work are based on reducing and equidistributing the remaining error in the functional after correction. The adaptive parameters are comprised of two inner products representing measures of the remaining error: a primal term and a dual term (see Section 2.3.2). These inner products can be expressed as sums of local contributions from each element in the domain. While the two inner products are approximately equal (exactly equal for linear problems), the local contributions to these products from any given element are generally not equal. The primal term is essentially what is used by Becker, Rannacher and collaborators [9, 10, 13, 51] in their finite-element-based adaptive procedure for functional outputs. To the author's knowledge, the addition of the dual term in the present procedure is novel.

2. **Anisotropic adaptation for functional outputs.**

   Much of the previous work in anisotropic grid adaptation has utilized adaptive sensors based on the error incurred by linear interpolation of a prescribed scalar field [16, 18, 19, 27, 48, 67]. To the author's knowledge, the viscous implementation in this work represents the first time output-based adaptive criteria have been incorporated into an anisotropic grid–adaptive procedure for either the finite volume or finite element methods. Output error control is incorporated into a metric that governs three characteristics: the size, stretching, and orientation of the elements in the grid. Output–based criteria are used to control the local element size whereas the Mach number Hessian controls the stretching and orientation. To the author's knowledge, the manner in which these three metric characteristics are decomposed, to allow for the introduction of the adjoint information, is novel. In principal, this decomposition technique can be used to specify adaptive information for each of the three metric characteristics individually.

3. **Application of adjoint error correction to functional outputs of Navier–Stokes simulations and two–dimensional transonic Euler simulations.**

   The functional correction technique used to improve the accuracy of the computed outputs in the present work is essentially an algebraic version of the Pierce and Giles method [26, 49]. The primary difference being that the current approach is cast in a multilevel framework in which the functional value on a fine grid is used as the accuracy benchmark instead of the exact functional corresponding to the solution of the original PDEs. In this sense, the fine grid used here plays a similar role to that of the *truth mesh* in the bounds procedure of Patera and Peraire [38, 46, 47].

   Pierce and Giles obtained superconvergent functional estimates for the Poisson equation in one and two dimensions, the quasi-one-dimensional Euler equations, the one-dimensional Helmholtz equation, and for nonlinear diffusion in two dimensions using their adjoint–based correction technique [25, 26, 49]. Superconvergent results for the quasi-one-dimensional Euler equations were also obtained in reference [62] using the current algebraic implementation. Improvements in functional accuracy are exhibited in virtually all the numerical results presented in this dissertation. Accuracy improvements in the lift, drag and/or moment coefficients from the current implementation of the adjoint correction technique have been obtained for subsonic, transonic, and supersonic Euler flows (see Section 3.6), and for subsonic Navier–Stokes simulations with Reynolds numbers ranging from 5,000 to 100,000 (see Section 4.6). Improvements in the rates of convergence of the functional error were demonstrated for the Gaussian bump inviscid test case in Section 3.6.2, and the Re $= 10$, $M_\infty = 0.38$ viscous cylinder and Re $= 10,000$, $M_\infty = 0.4$ NACA 0005 airfoil test cases in Section 4.6.1. In addition, improvements in the convergence rate of a diffusive flux integral was obtained for modest–Peclet–number scalar convection–diffusion in two dimensions using a Galerkin finite element method (See Section A.5.1).

5.3 Potential Impact

1. **Improved reliability of CFD simulations.**

   In industry, complex CFD simulations often require hours or even days of CPU time to converge. The reliability and accuracy of these simulations is imperative owing to the substantial costs involved. An inaccurate drag prediction due to inappropriate grid resolution in a obscure part of a complex, three–dimensional domain would require that the grid be adjusted, and the simulation rerun, which is generally unacceptable. The possibility of such an occurrence is what may preclude the use of CFD in some applications. Furthermore, even for the most experienced practitioners of CFD, the manner in which a grid should be adapted in order to ensure the accuracy of the simulation while maintaining computational efficiency is not always clear. This is particularly true if the accuracy of the simulation is measured by the accuracy of specific outputs such as lift or drag, which tend to be the most important quantities needed from the simulation. As demonstrated in this thesis, some of the most commonly used feature-based or interpolation–error–based adaptive algorithms cannot achieve this reliably. The potential benefit of a robust output-based adaptive scheme is that these outputs can be predicted accurately, efficiently and in an automated manner, saving valuable time and resources. Most importantly, however, is the added reliability such an adaptive scheme could provide for the overall simulation process.

2. **Enhanced applicability of anisotropic adaptation.**

   Numerical results indicate that anisotropic adaptation based exclusively on the Hessian of the Mach number often provides unsatisfactory estimates for the lift and drag coefficients. This has also been observed in the literature [16, 67]. Castro–Díaz *et al.* [16] state that the poor performance of Hessian–based adaptation is due to incorrect boundary layer resolution; in particular, they claim that the cause is due to the nonuniformity in distance of the first layer of nodes to the wall. Other explanations are related to the general shapes of the triangles in the grid. According to Rippa [52], a broad consensus exists in the literature that good triangulations should contain as few long and thin triangles as possible, particularly when the largest angle in the triangle approaches π radians[1].

   Despite these past results, the proposed output–based adaptive method was able to provide dramatic improvements in the accuracy of the lift and drag over pure Hessian–based adaptation for all the viscous test cases considered. No special modifications were needed near the walls in regards to the distance to the first layer of nodes, and no explicit effort was made to limit the largest angle in the triangulation. Typical values for the maximum angle in the triangulation

[1] Rippa goes on to show that thin triangles with angles approaching π may still be acceptable for interpolating functions with highly biased directionality.

were well within 1% of π.

Results indicate that the Hessian–based scheme was unable to provide sufficient grid resolution in key parts of the domain, where discretization errors had significant impact on the accuracy of the aerodynamic forces. In particular, boundary layers and wakes tended to be overrefined relative to adjacent inviscid regions where the grid was often severely underrefined.

The fact that slight modifications of the grid in localized regions of the domain can have such a dramatic impact on the accuracy of certain outputs is even more relevant in three dimensions. This is where a reliable adaptive algorithm can make the biggest impact on the scope of CFD usage in practical design applications.

5.4 Future Work

1. **Implementation for Reynolds–Averaged–Navier–Stokes (RANS) computations.**

   In principle, the output–based anisotropic grid–adaptive method outlined in Chapter 4 can be applied without modification to the simulation of the RANS equations using turbulence models such as the Baldwin–Barth [7] or Spalart–Allmaras [56] one–equation models, for example. Output–based adaptation is expected to be particularly effective for turbulent flows. High-Reynolds–number RANS simulations are characterized by very thin boundary layers that require highly stretched elements for efficient resolution. Furthermore, achieving grid convergence for turbulent flows is generally more difficult than for laminar computations due to the additional transport equation for the turbulent viscosity. The use of an effective output-based adaptive scheme could greatly facilitate the process, particularly when key outputs are of primary importance.

2. **Extension to three dimensions.**

   The algorithms and procedures outlined in this dissertation are all extendible to three dimensions. In many cases the details that define their operation for triangular grids can be extended in a natural manner to tetrahedral grids. Park [45] has implemented an extension of the proposed output-based adaptive procedure for three–dimensional Euler simulations. Application to three–dimensional high–Reynolds-number Navier–Stokes simulations could potentially be the most important extension of this work (see Section 5.3).

3. **Output–based adaptive criteria for element stretching and orientation.**

   In the output–based adaptive method described in Section 4.5.4, adjoint criteria are used to determine the local element size, and the Mach number Hessian is used to obtain local stretching and orientation information. The use of the Mach number distribution for the local Hessian calculations is completely arbitrary. It was chosen in the present context to facilitate comparisons with pure Hessian–based adaptation on the same scalar; in particular, it highlighted the fact that the dramatic improvements in accuracy obtained by the output–based adaptive method were entirely due to the use of adjoint error control on the functional of interest, and not on the choice of scalar for the Hessian calculations. In principle, the current framework does not even require that the Hessian of a scalar be used to provide the stretching and orientation information. A useful extension to the current method would be to introduce size, stretching and orientation information based exclusively on the adjoint criteria.

4. **Extension to unsteady flows.**

   Many practical flows are characterized by unsteady phenomena such as vortex shedding or oscillatory shock motion. In such cases, the outputs of interest may be time-averaged quantities, or perhaps time histories of instantaneous quantities. During a typical unsteady simulation, the local resolution requirements of the grid may change significantly from one time step to the next. An efficient time–accurate simulation would require frequent modifications of the grid to account for the changing flow patterns while maintaining output accuracy.

An output–based adaptive scheme might be an ideal candidate to achieve this. The most obvious implementation (in the general nonlinear case) requires that the unsteady adjoint equations be marched backward in time *after* the full unsteady primal problem has been solved to completion. In general, this is not a practical way to proceed. An alternate possibility, for appropriately defined outputs, might be to solve a steady adjoint problem at each primal time step. The adjoint solution from the previous time step could be used to initialize the solution process at the current time step. In principle, the adjoint might also be used to provide information on the optimal time step.

Bibliography

[1] M. J. Aftosmis and M. J. Berger. Multilevel error estimation and adaptive *h*-refinement for cartesian meshes with embedded boundaries. AIAA paper 2002-0863, 2002.

[2] M. Ainsworth and J. T. Oden. *A posteriori* error estimation in finite element analysis. *Comput. Meth. Appl. Mech. Eng.*, 142:1–88, 1997.

[3] W. K. Anderson and D. L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers Fluids*, 23:1–21, 1994.

[4] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Anal.*, 13:214–226, 1976.

[5] I. Babuška and A. Miller. The post-processing approach in the finite element method - Part 1: Calculation of displacements, stresses and other higher order derivatives of the displacements. *Int. J. Num. Meth. Eng.*, 20:1085–1109, 1984.

[6] T. J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements Anal. Design*, 25:243–273, 1997.

[7] B. S. Baldwin and T. J. Barth. A one–equation turbulence transport model for high reynolds number wall–bounded flows. AIAA paper 91-0610, 1991.

[8] L. Baskett and R. Haimes. Feature extraction of shear layers. AIAA paper 2001-2665, 2001.

[9] R. Becker and R. Rannacher. Weighted *a posteriori* error control in finite element methods, in *Proceedings of ENUMATH-97, Heidelberg*, (World Scientific, Singapore). pages 621–637, 1998.

[10] R. Becker and R. Rannacher. An optimal control approach to *a posteriori* error estimation in finite element methods, in *Acta Numerica 2001* (Ed. A. Iserles), Cambridge University Press. 2001.

[11] M. Berzins. An introduction to mesh quality. VKI Lecture Series 2000–05, 2000.

[12] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel. Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes. Research Report No. 2741, INRIA–Rocquencourt, France, 1995.

[13] M. Braack and R. Rannacher. Adaptive finite element methods for low–Mach–number flows with chemical reactions. VKI Lecture Series 1999–03, 1999.

[14] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov–Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Comput. Meth. Appl. Mech. Eng.*, 32:199–259, 1982.

[15] J. C. Carette. *Adaptive Unstructured Mesh Algorithms and SUPG Finite Element Method for Compressible High Reynolds Number Flows.* PhD thesis, von Karman Institute for Fluid Dynamics, Belgium, 1997.

[16] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *Int. J. Numer. Meth. Fluids*, 25:475–491, 1997.

[17] S. R. Chakravarthy and S. Osher. Numerical experiments with the Osher upwind scheme for the Euler equations. *AIAA J.*, 21(9):1241–1248, 1983.

[18] E. F. D'Azevedo and R. B. Simpson. On optimal triangular meshes for minimizing the gradient error. *Numer. Math.*, 59:321–348, 1991.

[19] A. Dervieux, D. Leservoisier, P. L. George, and Y. Coudiere. About theoretical and practical impact of mesh adaptation on approximation of functions and of solutions of PDE. ECCOMAS CFD Conference, Swansea, UK, 2001.

[20] M. Drela and H. Youngren. XFOIL 6.94 User Guide. M.I.T. Department of Aeronautics and Astronautics, Cambridge, MA, 2001, `http://raphael.mit.edu/xfoil`.

[21] J. Elliott and J. Peraire. Practical 3D aerodynamic design and optimization using unstructured meshes. *AIAA J.*, 35:1479–1486, 1997.

[22] M. B. Giles, M. C. Duta, and J. D. Müller. Adjoint code developments using the exact discrete approach. AIAA paper 2001-2596, 2001.

[23] M. B. Giles, M. G. Larson, M. Levenstam, and E. Süli. Adaptive error control for finite element approximations of the lift and drag in a viscous flow. Technical Report NA 97/06, Oxford Computing Laboratory, Oxford, 1997.

[24] M. B. Giles and N. A. Pierce. Adjoint equations in CFD: Duality, boundary conditions and solution behavior. AIAA paper 97-1850, 1997.

[25] M. B. Giles and N. A. Pierce. Improved lift and drag estimates using adjoint Euler equations. AIAA paper 99-3293, 1999.

[26] M. B. Giles and N. A. Pierce. Adjoint error correction for integral outputs. 2002. To appear in Lecture Notes in Computer Science. Springer–Verlag.

[27] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, and M. G. Vallet. Anisotropic mesh adaptation: Towards user-independent, mesh-independent and solver-independent CFD. Part I: General principles. *Int. J. Numer. Meth. Fluids*, 32:725–744, 2000.

[28] W. G. Habashi, M. Fortin, J. Dompierre, M. G. Vallet, and Y. Bourgault. Anisotropic mesh adaptation: A step towards a mesh-independent and user-independent CFD. In *Barriers and Challenges in Computational Fluid Dynam-*

ics (Ed. V. Venkatakrishnan *et al.*), Kluwer Academic Publishers, Netherlands, 1998.

[29] C. Hirsch. *Numerical Computation of Internal and External Flows. Vol. 1: Fundamentals of Numerical Discretization.* John Wiley & Sons Ltd., Chichester, England, 1992.

[30] P. Houston and E. Süli. *hp*–adaptive discontinuous Galerkin finite element methods for first–order hyperbolic problems. *SIAM J. Sci. Comput.*, 23(4):1226–1252, 2001.

[31] A. Jameson. Aerodynamic design via control theory. *J. Sci. Comput.*, 3:233–260, 1988.

[32] A. Jameson, N. Pierce, and L. Martinelli. Optimum aerodynamic design using the Navier–Stokes equations. *J. Theor. Comp. Fluid Mech.*, 10:213–237, 1998.

[33] P. R. Lahur and Y. Nakamura. Anisotropic Cartesian grid adaptation. AIAA paper 2000-2243, 2000.

[34] M. G. Larson and T. J. Barth. *A posteriori* error estimation for discontinuous Galerkin approximations of hyperbolic systems, NAS Technical Report NAS-99-010. 1999.

[35] P. Laug and H. Borouchaki. The BL2D Mesh Generator: Beginner's Guide, User's and Programmer's Manual. Technical Report No. 0194, INRIA–Rocquencourt, France, 1996, `http://www-rocq.inria.fr/gamma/cdrom/www/bl2d/eng.htm`.

[36] J. C. Lin and C. J. Dominick. Optimization of an advanced design three-element airfoil at high Reynolds numbers. AIAA paper 95-1858, 1995.

[37] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer–Verlag. Translated by S. K. Mitter, 1971.

[38] L. Machiels, J. Peraire, and A. T. Patera. *A posteriori* finite element output bounds for the incompressible Navier-Stokes equations: Application to a natural convection problem. *J. Comput. Phys.*, 172:401–425, 2001.

[39] D. L. Marcum and N. P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA J.*, 33(9):1619–1625, 1995.

[40] J. D. Müller and M. B. Giles. Solution adaptive mesh refinement using adjoint error analysis. AIAA paper 2001-2550, 2001.

[41] E. Nielsen. FUN2D/3D Fully Unstructured Navier–Stokes User Manual. NASA Langley Research Center, Computational Modeling and Simulation Branch, Virginia, 2002, `http://fun3d.larc.nasa.gov`.

[42] E. J. Nielsen and W. K. Anderson. Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. AIAA paper 98-4809, 1998.

[43] E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic optimization on unstructured meshes. AIAA paper 2001-0596, 2001.

[44] E. Onate and G. Bugeda. Mesh optimality criteria for adaptive finite element computations. Ch. 7, in *The Mathematics of Finite Elements and Applications* (Ed. J. R. Whiteman), pages 121–135, Wiley & Sons, Chichester, England, 1994.

[45] M. A. Park. Adjoint–based, three–dimensional error prediction and grid adaptation. AIAA paper 2002-3286, 2002.

[46] J. Peraire and A. T. Patera. Bounds for linear–functional outputs of coercive partial differential equations: Local indicators and adaptive refinement. In *Advances in Adaptive Computational Methods in Mechanics* (Ed. P. Ladevèze and J. T. Oden), Elsevier Science Ltd., 1998.

[47] J. Peraire and A. T. Patera. Asymptotic *a posteriori* finite element bounds for the outputs of non–coercive problems: The Helmholtz and Burgers equations. *Comp. Meth. Appl. Mech. Engrg.*, 171:77–86, 1999.

[48] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.*, 72:449–466, 1987.

[49] N. A. Pierce and M. B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Rev.*, 42(2):247–264, 2000.

[50] S. Z. Pirzadeh. An adaptive unstructured grid method by grid subdivision, local remeshing, and grid movement. AIAA paper 99-3255, 1999.

[51] R. Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128:205–233, 2001.

[52] S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29:257–270, 1992.

[53] P. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[54] Y. Saad and M. H. Schultz. GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[55] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delauney triangulator. School of Computer Science, Carnegie Mellon University, Pittsburg, PA, 1996.

[56] P. R. Spalart and S. R. Allmaras. A one–equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, 1:5–21, 1994.

[57] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1986.

[58] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Wellesley-Cambridge Press, Wellesley, MA, 1988.

[59] E. Süli, P. Houston, and B. Senior. *hp*–discontinuous Galerkin finite element methods for hyperbolic problems: error analysis and adaptivity. In *Numerical Methods for Fluid Dynamics VII* (Ed. M. J. Baines), ICFD Oxford, 2001.

[60] R. C. Swanson and E. Turkel. Multistage schemes with multigrid for Euler and Navier–Stokes equations – applications. Unpublished NASA report, 2002.

[61] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[62] D. A. Venditti and D. L. Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *J. Comput. Phys.*, 164:204–227, 2000.

[63] D. A. Venditti and D. L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *J. Comput. Phys.*, 176:40–69, 2002.

[64] G. P. Warren, W. K. Anderson, J. T. Thomas, and S. L. Krist. Grid convergence for adaptive methods. AIAA paper 91-1592, 1991.

[65] F. M. White. *Viscous Fluid Flow.* McGraw–Hill, New York, 1991.

[66] W. A. Wood and W. L. Kleb. On multi–dimensional unstructured mesh adaption. AIAA paper 99-3254, 1999.

[67] G. Xia, D. Li, and C. L. Merkle. Anisotropic grid adaptation on unstructured meshes. AIAA paper 2001-0443, 2001.

[68] X. D. Zhang, M.-G. Vallet, J. Dompierre, P. Labbé, D. Pelletier, J.-Y. Trépanier, R. Camarero, J. V. Lassaline, L. M. Manzano, and D. W. Zingg. Mesh adaptation using different error indicator for the Euler equations. AIAA paper 2001-2549, 2001.

[69] O. C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, London, 3rd edition, 1977.

[70] O. C. Zienkiewicz and J. Z. Zhu. A simple error estimation and adaptive procedure for practical engineering analysis. *Int. J. Num. Meth. Eng.*, 24:337–357, 1987.

[71] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and *a posteriori* error estimates. Part 1: The recovery technique. *Int. J. Num. Meth. Eng.*, 33:1331–1364, 1992.

[72] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and *a posteriori* error estimates. Part 2: Error estimates and adaptivity. *Int. J. Num. Meth. Eng.*, 33:1365–1382, 1992.

Appendix A

Convection–Diffusion

The adjoint error correction and grid adaptive methodology presented in Chapter 2 is applied to steady, scalar, linear convection–diffusion using a Galerkin finite element discretization. In Section A.4 the general nonlinear correction theory is specialized for a linear problem. In Section A.5 functional correction and adaptive results are presented. The correction technique is shown to provide an improvement in the convergence rate of a diffusive flux integral. Adaptive results demonstrate effective error control for a similar functional using the proposed output–based adaptive procedure.

A.1 Governing Equation

The steady, two-dimensional convection-diffusion equation may be written as:

$$\mathcal{L}\left(u\right) = \vec{\nabla} \cdot (\vec{\mathbf{V}}u) - \frac{1}{\mathrm{Pe}}\nabla^2 u = f, \tag{A.1}$$

where $\mathcal{L}$ represents the linear, homogeneous, differential operator, u is the scalar dependent variable, $\vec{\mathbf{V}}$ is a prescribed, divergence-free velocity field, f is the volumetric generation term, and Pe is the Peclet number which quantifies the relative strengths of advective to diffusive transport of the scalar u.

The most common boundary conditions encountered for this equation are Dirichlet (specified value) and Neumann (specified normal derivative) conditions. If $\partial\Omega_D$

and $\partial\Omega_N$ represent portions of the domain boundary where Dirichlet and Neumann conditions are specified, respectively, then these conditions may be written as

$$\begin{aligned} u &= u_{sp} && \text{on} \quad \partial\Omega_D, \\ \vec{\nabla} u \cdot \hat{\mathbf{n}} &= \left(\frac{\partial u}{\partial n}\right)_{sp} && \text{on} \quad \partial\Omega_N, \end{aligned} \tag{A.2}$$

where u_{sp} and $(\partial u / \partial n)_{sp}$ are specified distributions over their respective boundaries. Another common boundary condition is the Robin condition (also known as the convective boundary condition) in which a linear combination of the scalar and its normal derivative is specified.

Two physical processes that are described by the convection-diffusion equation are heat and mass transfer. In the case of heat transfer u represents the internal energy or temperature of a fluid being transported due to overall fluid motion (advection) and molecular interaction (diffusion). In the case of mass transfer u represents the concentration of a particular species being transported by similar mechanisms. Convection-diffusion equations also result from a particular linearization and simplification of the incompressible Navier-Stokes equations. In this case u represents a component of the specific momentum vector (that is, a velocity component). In the Navier-Stokes equations, the velocity, itself, is an unknown giving rise to the nonlinear terms in the equations.

A.2 Galerkin Finite Element Method

The finite element method is a numerical discretization method for approximating solutions to partial differential equations (PDEs) with associated boundary conditions. The method may be summarized as follows: find $u_h \in S_h$ such that $\forall\, w_h \in S_h'$

$$\int_{\Omega_h} w_h \, \mathcal{R}\left(u_h\right) d\Omega = 0, \tag{A.3}$$

where S_h is a prescribed function space of trial (or interpolation) polynomials and S'_h is an associated space of weighting functions. The piecewise polynomial function u_h is the resulting approximation to the PDE. The residual $\mathcal{R}$ is defined as

$$\mathcal{R}(u) \equiv \mathcal{L}(u) - f, \tag{A.4}$$

where $\mathcal{L}$ is the homogeneous differential operator associated with the PDE under consideration and f is the corresponding inhomogeneous term as defined in (A.1), for example. The computational domain, Ω_h, is a partition of the original domain (upon which the continuum PDE is defined) into N_e non-overlapping triangular elements, Ω_e, such that

$$\Omega_h = \bigcup_{e=1}^{N_e} \Omega_e. \tag{A.5}$$

In the present implementation, S_h is chosen to be the space of piecewise linear polynomials over each element in the domain. The Galerkin method is obtained when the space of weighting functions, S'_h, is also chosen to be piecewise linear except at the Dirichlet boundaries, where w_h is set to 0 and the Dirichlet conditions are imposed strongly. The function u_h is usually written as a linear combination of nodal basis (or shape) functions $N_i(x, y)$

$$u_h(x, y) = \sum_i u_i N_i(x, y), \tag{A.6}$$

where the coefficients u_i are nodal values $u_h(x_i, y_i)$ which constitute the degrees of freedom or unknowns of the discrete problem. The shape functions $N_i(x, y)$ are linear "tent-shaped" functions associated with each node i such that

$$N_i(x_j, y_j) = \delta_{ij}, \tag{A.7}$$

for any grid node j where δ_{ij} is the Kronecker delta symbol. Figure A-1 shows an example of the shape function $N_i(x, y)$ for a typical node i and a piecewise linear rep-

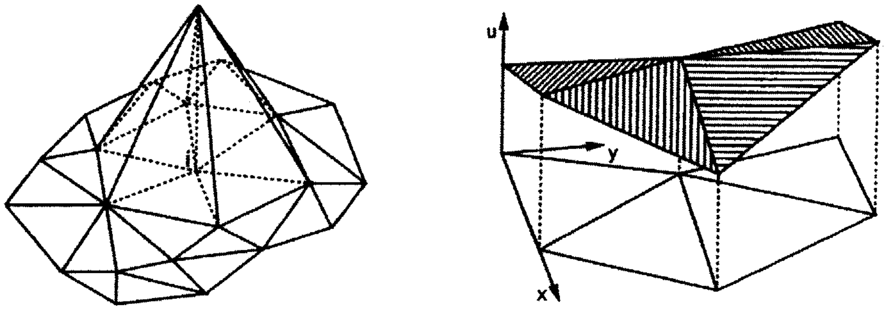


Figure A-1: Left: Galerkin weighting function ("tent function") for node i; right: schematic of a piecewise linear representation of the solution u over a typical patch of elements. *Figure obtained from reference [15].*

resentation of the solution over a typical patch of elements. Further implementation details can be found in reference [29].

A.3 Stability

When the Peclet number is large (that is, $\text{Pe} \gg 1$) advection dominates diffusion. In this case the exact solution often exhibits narrow internal and/or boundary layers characterized by strong variations in the transport variable. Under these conditions, the Galerkin method and other central-difference-type methods are known to produce solutions that are corrupted with spurious node-to-node oscillations. One way to eliminate this is to severely refine the mesh to the point where advection no longer dominates diffusion at an elemental level. In quantitative terms, if one defines an elemental Peclet number as $\text{Pe}_h = \|\vec{\mathbf{V}}\|h/\kappa$ where h is a measure of the element size (such as the diameter) and κ is the physical diffusivity (appearing in the dimensional form of the convection-diffusion equation) then the Galerkin method may produce stable solutions if $\text{Pe}_h \sim \mathcal{O}(1)$ throughout the domain [29]. For higher Peclet-number problems it may become prohibitive to refine the mesh to the extent that is needed for stability. An alternate approach is to use some type of upwind differencing on the advective term. In the finite element context, the Streamline Upwind/Petrov-Galerkin Method (SUPG) [14] has proven to be quite robust and stable at higher

Peclet numbers without requiring such extreme limits on grid spacing. In the present implementation, however, only modest Peclet numbers (≤ 100) are considered, for which the Galerkin method performs adequately.

A.4 Adjoint Error Correction: Linear Theory

The functional correction procedure presented in Chapter 2 is specialized for a linear problem. This procedure is essentially an algebraic implementation of the Pierce and Giles [26, 49] correction method. From this point onward u_h represents the discrete vector containing the finite element solution at the nodes.

A.4.1 Duality

We are interested in computing the linear functional output J_h

$$J_h = g_h^T u_h + \alpha_h, \tag{A.8}$$

where u_h is the primal variable satisfying the discrete finite–element system of equations

$$L_h u_h = f_h, \tag{A.9}$$

and α_h is a constant vector. The dual problem is to determine

$$J_h = v_h^T f_h + \alpha_h, \tag{A.10}$$

where v_h is the adjoint variable satisfying

$$L_h^T v_h = g_h. \tag{A.11}$$

Equivalence of the formulation is seen from

$$\begin{aligned} J_h &= g_h^T u_h + \alpha_h, \\ &= v_h^T L_h u_h + \alpha_h, \\ &= v_h^T f_h + \alpha_h. \end{aligned} \tag{A.12}$$

A.4.2 Error Correction

The primal and dual residual operators are defined as

$$\begin{aligned} R_h(\phi_h) &\equiv L_h \phi_h - f_h, \\ R_h^v(\phi_h) &\equiv L_h^T \phi_h - g_h. \end{aligned} \tag{A.13}$$

where $R_h(u_h) = 0$ and $R_h^v(v_h) = 0$. Note that these are discrete operators that represent integral quantities such as those that would arise from the finite element approximation (A.3).

Consider a perturbation δu_h. We have,

$$\begin{aligned} R_h(u_h + \delta u_h) &= L_h(u_h + \delta u_h) - f_h, \\ &= L_h \delta u_h. \end{aligned} \tag{A.14}$$

The corresponding error in the functional will be

$$\begin{aligned} \delta J_h &= g_h^T \delta u_h, \\ &= v_h^T L_h \delta u_h, \\ &= v_h^T R_h(u_h + \delta u_h). \end{aligned} \tag{A.15}$$

Let $\tilde{u}_h = u_h + \delta u_h$ and $\tilde{J}_h = J_h + \delta J_h$ denote the perturbed primal solution and functional, respectively. Also consider an independent perturbation in the adjoint,

$\tilde{v}_h = v_h + \delta v_h$. We have,

$$\begin{aligned}
\delta J_h &= v_h^T R_h(\tilde{u}_h), \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) + (v_h - \tilde{v}_h)^T R_h(\tilde{u}_h), \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) - \delta v_h^T R_h(\tilde{u}_h), \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) - \delta v_h^T L_h \delta u_h, \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) - R_h^v(\tilde{v}_h)^T \delta u_h, \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) - R_h^v(\tilde{v}_h)^T L_h^{-1} R_h(\tilde{u}_h). \qquad \text{(A.16)}
\end{aligned}$$

Note that the perturbation in the adjoint will also lead to an independent perturbation in the functional, $\tilde{J}_h^v = J_h + \delta J_h^v$, which can be expressed as,

$$\begin{aligned}
\delta J_h^v &= \delta v_h^T f_h, \\
&= \delta v_h^T L_h u_h, \\
&= R_h^v(\tilde{v}_h)^T u_h, \\
&= R_h^v(\tilde{v}_h)^T \tilde{u}_h - R_h^v(\tilde{v}_h)^T \delta u_h. \qquad \text{(A.17)}
\end{aligned}$$

The remaining error term is identical to that which was derived for δJ_h leading to the following identity:

$$\begin{aligned}
\delta J_h - \delta J_h^v &= \tilde{J}_h - \tilde{J}_h^v, \\
&= \tilde{v}_h^T R_h(\tilde{u}_h) - R_h^v(\tilde{v}_h)^T \tilde{u}_h. \qquad \text{(A.18)}
\end{aligned}$$

A.4.3 Strong Boundaries

The following development follows the derivation in Section 4.4 and reference [22]. Let $\bar{L}_h$ denote the discrete operator without boundary integrals, and let B_h represent the projection matrix that sets the boundary conditions in a strong sense (see Section 4.4).

We have,

$$\begin{aligned} (I_h - B_h)\bar{L}_h u_h &= (I_h - B_h)\bar{f}_h, \\ B_h u_h &= B_h c_h, \end{aligned} \tag{A.19}$$

or,

$$\left[(I_h - B_h)\bar{L}_h + B_h\right] u_h = (I_h - B_h)\bar{f}_h + B_h c_h. \tag{A.20}$$

The relation to L_h is given by

$$\begin{aligned} L_h &\equiv (I_h - B_h)\bar{L}_h + B_h, \\ f_h &\equiv (I_h - B_h)\bar{f}_h + B_h c_h, \end{aligned} \tag{A.21}$$

so that we recover (A.9)

$$L_h u_h = f_h. \tag{A.22}$$

The adjoint can be decomposed into two orthogonal components:

$$\begin{aligned} v_h^{(b)} &= B_h v_h, \\ v_h^{(i)} &= (I_h - B_h) v_h. \end{aligned} \tag{A.23}$$

where (b) denotes the strong boundary nodes and (i) the remaining interior nodes.

Expressing the functional error in terms of these components,

$$\begin{aligned}
\delta J_h &= g_h^T \delta u_h, \\
&= v_h^T L_h \delta u_h, \\
&= v_h^T R_h(\tilde{u}_h), \\
&= \left(B_h v_h + (I_h - B_h) v_h\right)^T R_h(\tilde{u}_h), \\
&= \left(B_h^2 v_h + (I_h - B_h)^2 v_h\right)^T R_h(\tilde{u}_h), \\
&= \left(B_h v_h^{(b)} + (I_h - B_h) v_h^{(i)}\right)^T R_h(\tilde{u}_h), \\
&= \left(v_h^{(b)}\right)^T B_h R_h(\tilde{u}_h) + \left(v_h^{(i)}\right)^T (I_h - B_h) R_h(\tilde{u}_h), \\
&= \left(v_h^{(b)}\right)^T B_h L_h \delta u_h + \left(v_h^{(i)}\right)^T (I_h - B_h) L_h \delta u_h, \\
&= \left(v_h^{(b)}\right)^T B_h \left[(I_h - B_h)\bar{L}_h + B_h\right] \delta u_h + \left(v_h^{(i)}\right)^T (I_h - B_h) \left[(I_h - B_h)\bar{L}_h\right. , \\
&\quad \left. + B_h\right] \delta u_h, \\
&= \left(v_h^{(b)}\right)^T B_h \delta u_h + \left(v_h^{(i)}\right)^T (I_h - B_h) \bar{L}_h \delta u_h, \\
&= \left(v_h^{(b)}\right)^T B_h (\tilde{u}_h - c_h) + \left(v_h^{(i)}\right)^T (I_h - B_h)(\bar{L}_h \tilde{u}_h - \bar{f}_h), \\
&= \left(v_h^{(b)}\right)^T B_h (\tilde{u}_h - c_h) + \left(v_h^{(i)}\right)^T (I_h - B_h) \bar{R}_h(\tilde{u}_h), \qquad \text{(A.24)}
\end{aligned}$$

shows that the boundary adjoint weighs the explicit error in the solution at the strong boundary whereas the interior adjoint weighs the residual error in the interior. This elucidates the difference in the character of the adjoint on the strong boundaries. In general, the magnitude of the adjoint on the boundary can be quite different than that in the interior.

A.4.4 Postprocessing the Boundary Adjoint

The following is based on reference [22] (see also Section 4.4.1). Examining the adjoint system we have,

$$\begin{aligned} L_h^T v_h &= g_h, \\ \left[(I_h - B_h)\bar{L}_h + B_h\right]^T v_h &= g_h, \\ \bar{L}_h^T (I_h - B_h) v_h + B_h v_h &= g_h. \end{aligned} \tag{A.25}$$

Left multiplying by $(I_h - B_h)$ yields,

$$(I_h - B_h)\bar{L}_h^T v_h^{(i)} = (I_h - B_h) g_h. \tag{A.26}$$

Thus $v_h^{(i)}$ satisfies

$$\begin{aligned} (I_h - B_h)\bar{L}_h^T v_h^{(i)} &= (I_h - B_h) g_h, \\ B_h v_h^{(i)} &= 0, \end{aligned} \tag{A.27}$$

which can be combined into

$$\left[(I_h - B_h)\bar{L}_h^T + B_h\right] v_h^{(i)} = (I_h - B_h) g_h. \tag{A.28}$$

Once $v_h^{(i)}$ is obtained, the boundary adjoint can be postprocessed from

$$\begin{aligned} v_h^{(b)} &= g_h - \bar{L}_h^T v_h^{(i)}, \\ &= B_h \left[g_h - \bar{L}_h^T v_h^{(i)} \right]. \end{aligned} \tag{A.29}$$

A.5 Results

The error correction technique and grid adaptive procedure are tested within the framework of the Galerkin finite element method outlined in Section A.2.

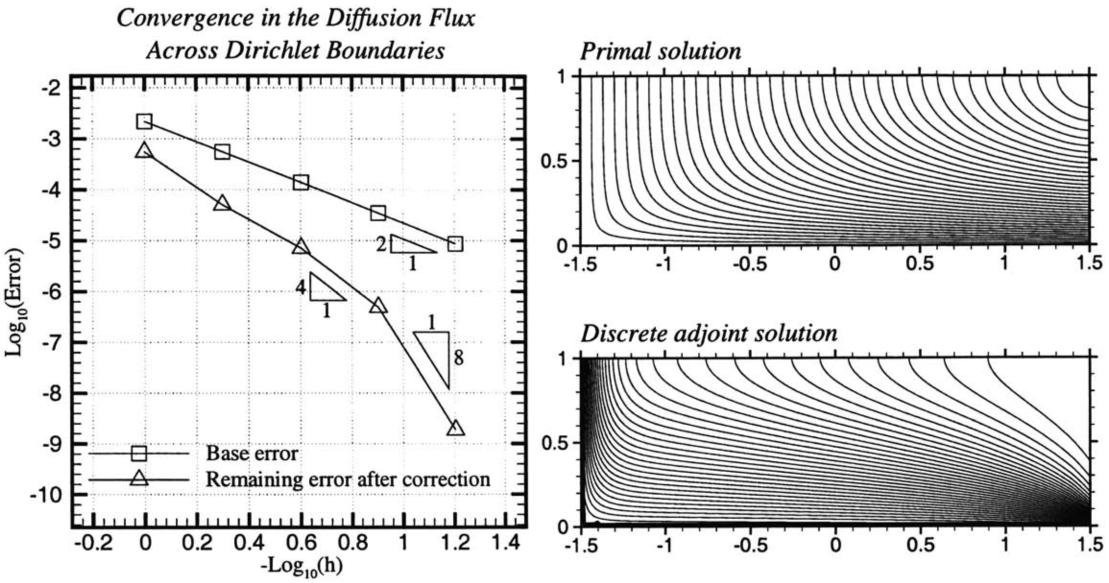


Figure A-2: Convergence results (left); Primal and adjoint solutions (right) for Pe = 10.

A.5.1 Functional Correction Results

Scalar convection–diffusion is simulated on the rectangular domain shown in Figure A-2. The velocity field is set to $\vec{\mathbf{V}} = \hat{\imath}$. The Dirichlet condition $u = 1$ is prescribed on the left and bottom sides of the rectangle, denoted $\partial\Omega_D$. Homogeneous Neumann conditions are prescribed on the top and right boundaries $\partial\Omega_N$. The source term is $f = 1$ and the Peclet number is 10. The functional of interest is the diffusion flux across Dirichlet boundaries,

$$J(u) = \int_{\partial\Omega_D} \vec{\nabla} u \cdot \vec{\mathbf{n}}\, ds, \tag{A.30}$$

which is approximated using residual balances as described in Section 4.4.2. Figure A-2 shows plots of the base error $|\delta J_h| = \left|\tilde{J}_h - J_h\right|$ and the remaining error after correction $\left|\delta J_h - (\bar{Q}_h^H v_H)^T R_h(Q_h^H u_H)\right|$ computed on a hierarchy of 5 embedded grids. Starting with the coarsest grid, each subsequent grid is constructed by subdividing the triangles of the previous grid into 4 self–similar triangles. Errors are measured with respect to the fine grids ($N = 2$) associated with each base grid in the hierarchy.

The prolongation operators Q_h^H and $\bar{Q}_h^H$ are defined in Sections 2.2.2 and 4.4.3, respectively. The convergence results demonstrate that the corrected functionals converge faster than the associated base values.

A.5.2 Adaptive Results

A convection-diffusion test case is contrived for testing the proposed output–based adaptive procedure outlined in Section 2.3.2. Grid refinement is achieved using the isotropic h–refinement algorithm described in Section 3.5. The domain is contained within a rectangular outer boundary. The inner boundary is geometrically complex; shaped like a letter of the alphabet. Figure A-3 shows the preliminary grid used (prior to adaptation) with a blow-up near the object.

Velocity Field

Prior to solving the convection-diffusion equation, a divergence-free velocity field $\vec{\mathbf{V}}$ must be specified. One way of obtaining a velocity field that satisfies $\vec{\nabla} \cdot \vec{\mathbf{V}} = 0$ everywhere and $\vec{\mathbf{V}} \cdot \vec{\mathbf{n}} = 0$ on solid walls is to solve the Laplace equation for the velocity potential ϕ with appropriate boundary conditions. By defining $\vec{\mathbf{V}} = \vec{\nabla}\phi$ we obtain

$$\vec{\nabla} \cdot \vec{\mathbf{V}} = \nabla^2 \phi = 0. \tag{A.31}$$

To prevent flow through solid walls we impose homogeneous Neumann conditions there,

$$\vec{\mathbf{V}} \cdot \vec{\mathbf{n}} = \vec{\nabla}\phi \cdot \vec{\mathbf{n}} = 0. \tag{A.32}$$

Lastly, a potential difference is imposed across the inflow and outflow boundaries by prescribing different values for the potential at each boundary. A greater value for the potential is assigned at the outflow boundary (relative to the inflow boundary) so that the flow does indeed leave the domain there. Fortunately, the Laplace equation is

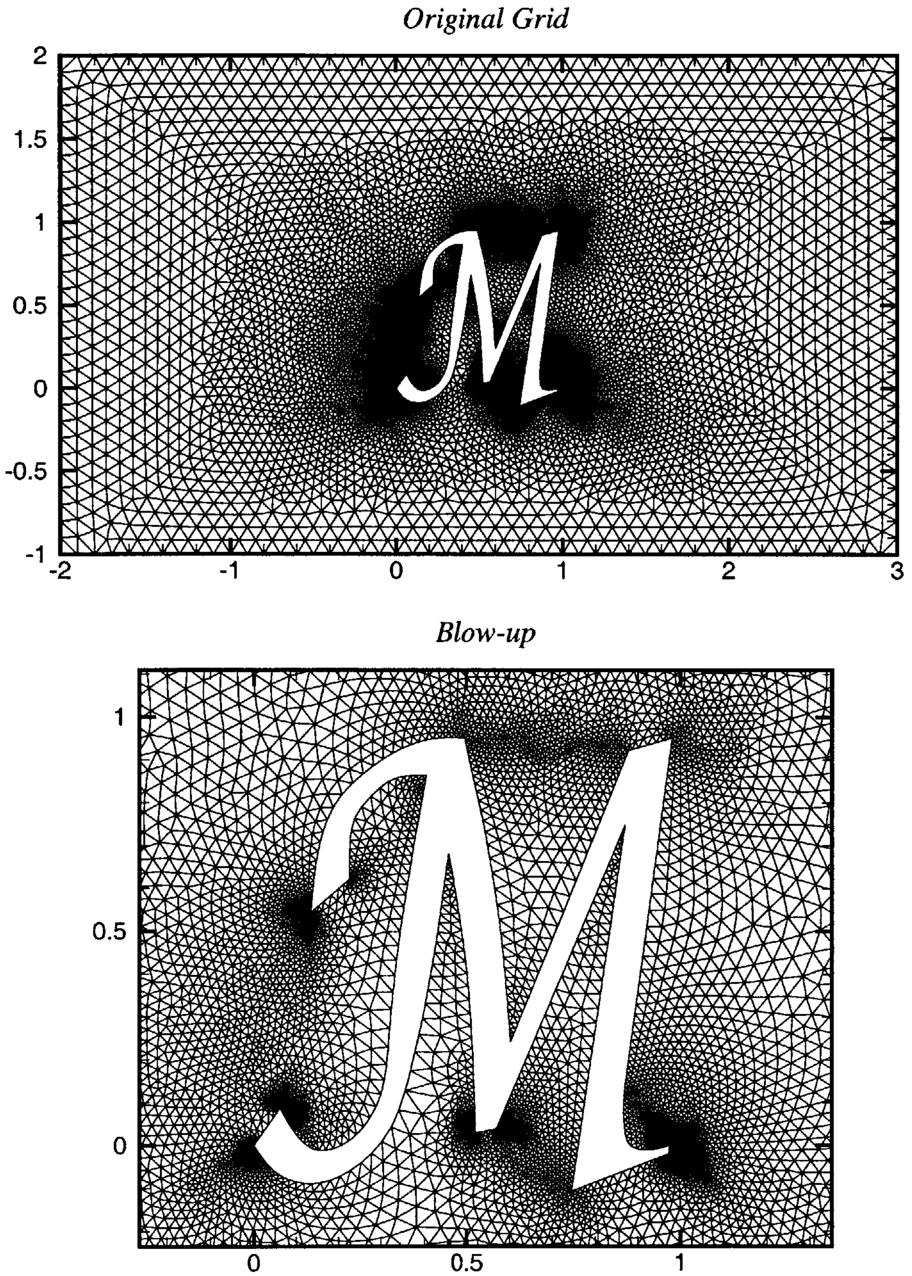


Figure A-3: Initial computation grid prior to adaptation (top); blow-up of the same grid near the obstacle (bottom).

a special case of the convection-diffusion equation; it is obtained when Pe $= 0$. Thus, the very same finite element method described in Section A.2 can be used without modification to solve for the velocity potential. Once ϕ is obtained it is a trivial matter to differentiate the solution to obtain the velocity components. Figure A-4 shows the computed velocity potential (top) and the corresponding velocity field (bottom) in the vicinity of the object.

Boundary Conditions

As is evident from Figure A-4, the inflow boundary is chosen to be the left-most segment of the outer boundary and the outflow is prescribed at the right-most segment. The upper and lower segments as well as the entire surface of the interior object are chosen as solid walls. At the inflow boundary, the value of the transport variable u is held at 0. Homogeneous Neumann conditions are prescribed at the upper and lower walls as well as on the outflow boundary. Homogeneous Dirichlet conditions are applied on the surface of the object except on the upper-left-most segment where a Gaussian distribution for u is prescribed. The distribution varies from machine-zero at the edges to a maximum of 1 at the middle of the segment.

Functional

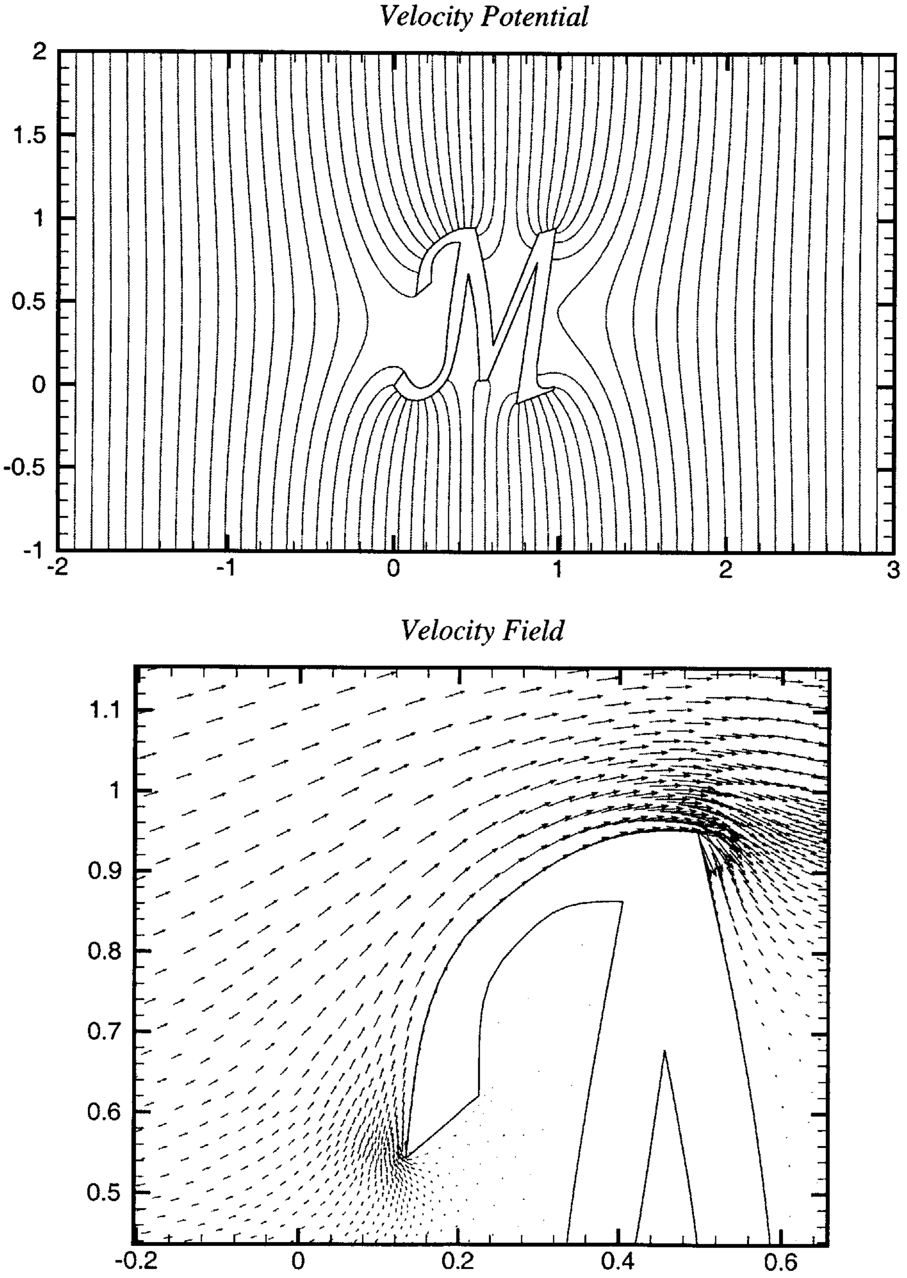
The functional of interest is chosen to be the diffusive flux across the boundary segment Ω_{D_1},

$$J(u) = \int_{\partial\Omega_{D_1}} \vec{\nabla} u \cdot \vec{\mathbf{n}}\, ds, \tag{A.33}$$

where Ω_{D_1} is the upper-left-most segment of the object.

Primal and Adjoint Solutions

Figure A-5 shows plots of the primal and adjoint solutions for different Peclet numbers. In each case, the adjoint solution corresponds to the functional defined in (A.33). All solutions presented in this figure were solved on the mesh shown in Figure A-3.


Velocity Potential

Velocity Field

Figure A-4: Velocity potential (top); corresponding velocity field near the upper-left portion of the obstacle (bottom).

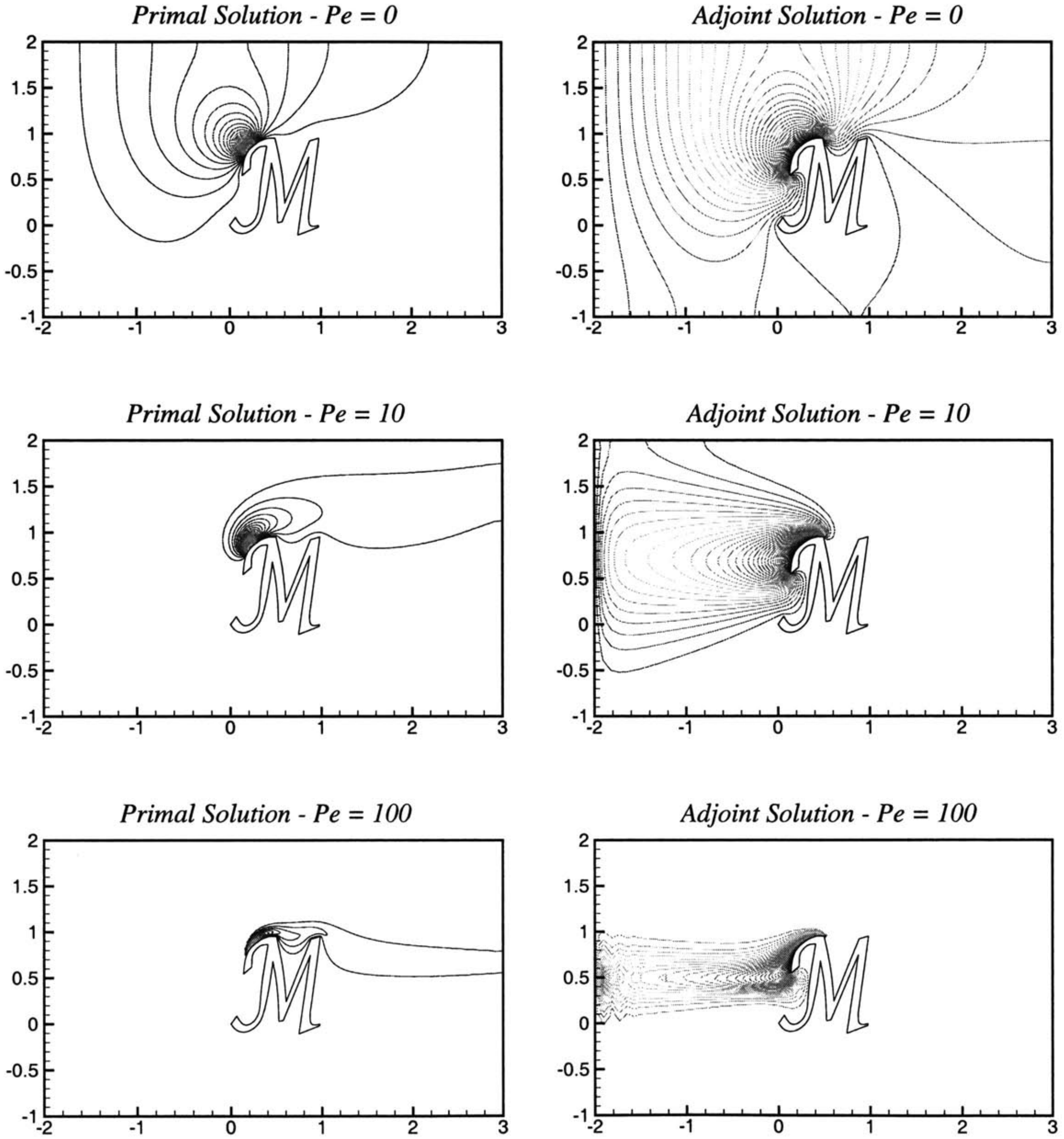


Figure A-5: Computed solutions of the primal (left column) and adjoint (right column) convection-diffusion equations for different Peclet numbers (Pe).

For Pe = 0 the primal and adjoint equations simplify to the Laplace equation. In both cases the transport scalar diffuses out from the upper-left segment of the object in an isotropic manner without any advective bias. For Pe = 10 the primal scalar is diffused and transported downstream by advection whereas the adjoint variable is advected upstream. At Pe = 100 advective effects are even more pronounced. Note the slight instability in the adjoint solution at the inflow boundary. This is attributed to the inadequacy of the Galerkin method to handle the moderate to large advection associated with the higher Peclet number. See Section A.2 and [14] for a more elaborate discussion of this.

Adaptation

Figure A-6 shows adaptive results performed for Pe = 10. The original and final grids and corresponding primal solutions are shown in this figure. The adapted solution is better resolved in the vicinity of the upper-left surface of the object upon which the functional is defined. Figure A-7 shows the convergence of the functional during the iterative adaptive process versus the total number of nodes in the respective grid. The total number of nodes is plotted on a logarithmic scale. Also shown is the convergence of the corrected functional. The functional convergence on uniformly refined meshes is also provided for comparison. The two uniformly refined grids were obtained by subdividing each of the elements in the original mesh into 4 and 16 elements, respectively. The solid line is a Richardson extrapolation of the functional values computed on the uniformly refined grids.

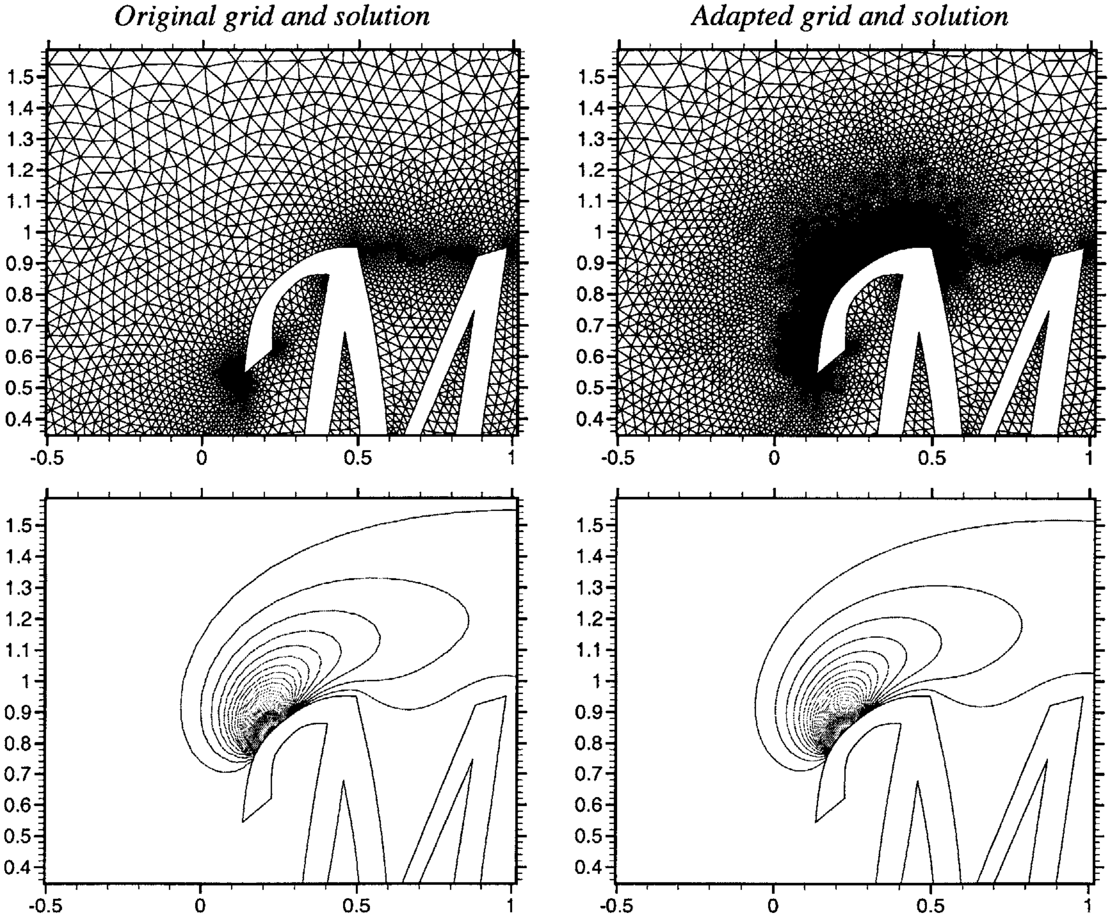


Figure A-6: Plots of the original grid and primal solution (left) and the final refined grid and corresponding primal solution (right) for Pe = 10.

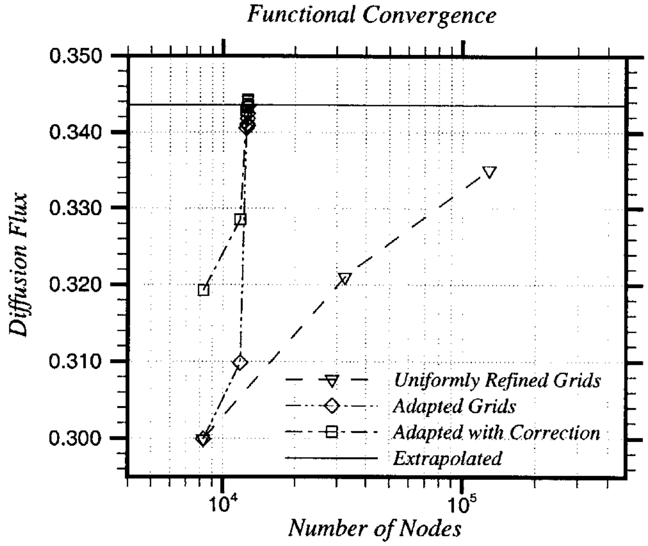


Figure A-7: Convergence of the computed functional for Pe = 10. The value of the functional is plotted versus the total number of degrees of freedom in the corresponding grid.

Appendix B

Error Analysis

In this section, nonlinear terms are retained in the analysis of the functional error. A derivation of the duality gap D, encountered in Section 2.3.1, is also presented.

B.1 Functional Error

The exact, truncated, second order, Taylor series expansion of $f_h(U_h)$ about the perturbed solution can be written as

$$f_h(U_h) = f_h(\tilde{U}_h) + \left.\frac{\partial f_h}{\partial U_h}\right|_{\tilde{U}_h} (U_h - \tilde{U}_h) + \frac{1}{2}(U_h - \tilde{U}_h)^T \left.\frac{\partial^2 f_h}{\partial U_h^2}\right|_{\tilde{U}'_h} (U_h - \tilde{U}_h), \qquad \text{(B.1)}$$

where $f_h(U_h)$ is assumed to be twice differentiable with respect to the components of U_h. In this expression, $[\partial^2 f_h / \partial U_h^2]|_{\tilde{U}'_h}$ is the Hessian of the functional evaluated at the intermediate state $\tilde{U}'_h$, where $\left|U_h - \tilde{U}'_h\right| \leq \left|U_h - \tilde{U}_h\right|$ componentwise. Similarly, the residual operator, $R_h(U_h)$ can be expanded as

$$R_h(U_h) = R_h(\tilde{U}_h) + \left.\frac{\partial R_h}{\partial U_h}\right|_{\tilde{U}_h} (U_h - \tilde{U}_h) \; + \; W, \qquad \text{(B.2)}$$

where

$$(W)_i = \frac{1}{2}(U_h - \tilde{U}_h)^T \left.\frac{\partial^2 (R_h)_i}{\partial U_h^2}\right|_{\tilde{U}''_h} (U_h - \tilde{U}_h), \qquad \text{(B.3)}$$

for each component i of the vector W. The term, $[\partial^2 (R_h)_i/\partial U_h^2]|_{\tilde{U}_h''}$ represents the Hessian of the i^{th} component of the residual vector evaluated at the intermediate state $\tilde{U}_h''$, where $\left|U_h - \tilde{U}_h''\right| \leq \left|U_h - \tilde{U}_h\right|$ componentwise. In general, the intermediate state $\tilde{U}_h''$ is different for every component i of the residual expansion in (B.2). From (B.2), the solution error is obtained, symbolically, as

$$(U_h - \tilde{U}_h) = - \left.\frac{\partial R_h}{\partial U_h}\right|_{\tilde{U}_h}^{-1} \left(R_h(\tilde{U}_h) + W\right), \tag{B.4}$$

where the Jacobian is assumed nonsingular due to wellposedness. Using (B.1), (B.4), and the definition of the fine-grid adjoint given by (2.6), we arrive at the following expression for the functional error:

$$\begin{aligned} f_h(\tilde{U}_h) - f_h(U_h) \quad &= \quad \underbrace{\tilde{\Psi}_h^T R_h(\tilde{U}_h)}_{\textit{Computable correction}} \\ &+ \quad \underbrace{\left(\Psi_h - \tilde{\Psi}_h\right)^T R_h(\tilde{U}_h)}_{\textit{Remaining error neglecting nonlinear effects}} \\ &+ \quad \underbrace{\Psi_h^T W - \frac{1}{2}(U_h - \tilde{U}_h)^T \left.\frac{\partial^2 f_h}{\partial U_h^2}\right|_{\tilde{U}_h'} (U_h - \tilde{U}_h)}_{\textit{Error due to nonlinear effects}}. \end{aligned} \tag{B.5}$$

B.2 Duality Gap

Right-multiplying the transpose of (2.10) by $\left[\partial R_h/\partial U_h|_{\tilde{U}_h}\right]^{-1} R_h(\tilde{U}_h)$, and using (B.4),

$$\begin{aligned} \left(\Psi_h - \tilde{\Psi}_h\right)^T R_h(\tilde{U}_h) \quad &= \quad -R_h^{\Psi}\left(\tilde{\Psi}_h\right)^T \left[\left.\frac{\partial R_h}{\partial U_h}\right|_{\tilde{U}_h}\right]^{-1} R_h(\tilde{U}_h), \\ &= \quad R_h^{\Psi}\left(\tilde{\Psi}_h\right)^T \left[(U_h - \tilde{U}_h) + \left[\left.\frac{\partial R_h}{\partial U_h}\right|_{\tilde{U}_h}\right]^{-1} W\right], \\ &= \quad R_h^{\Psi}\left(\tilde{\Psi}_h\right)^T (U_h - \tilde{U}_h) - \left(\Psi_h - \tilde{\Psi}_h\right)^T W, \end{aligned} \tag{B.6}$$

from which the duality gap D is obtained as

$$\begin{aligned} D &\equiv \left(\Psi_h - \tilde{\Psi}_h\right)^T R_h(\tilde{U}_h) - R_h^{\Psi}\left(\tilde{\Psi}_h\right)^T \left(U_h - \tilde{U}_h\right), \\ &= -\left(\Psi_h - \tilde{\Psi}_h\right)^T W, \\ &= \delta\Psi_h^T W. \end{aligned} \tag{B.7}$$

Appendix C

Isotropic Grid Smoothing

A local least squares procedure for isotropic grid smoothing is presented. This procedure is used to improve the quality of the grids during the inviscid simulations of Chapter 3. Referring to triangle k in Figure C-1, the isotropic quality measure [11]

$$q_k = \frac{h_1^2 + h_2^2 + h_3^2 + (h_1 + h_2 + h_3)^2}{16\sqrt{3}A}, \tag{C.1}$$

has value 1 for an equilateral triangle and tends to infinity as the area A of the triangle tends to zero holding at least one of its sides constant. Note that for a positive triangle, $q_k \in [1, \infty)$. Let N_i be the number of triangles surrounding node i (in Figure C-1, $N_i = 6$). The least squares problem is to determine the coordinates (x_i', y_i') of node i such that the sum of the squares of the quality measures q_k for the

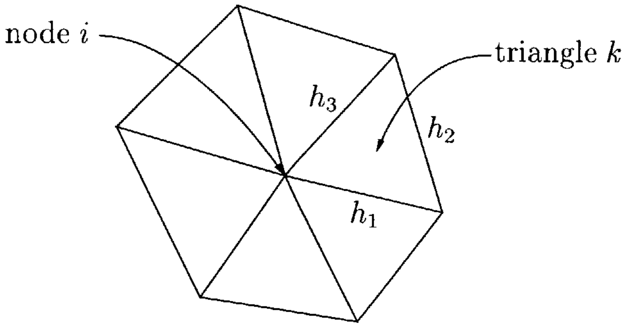


Figure C-1: Patch of triangles surrounding node i.

triangles $k \in [1, N_i]$ surrounding node i is minimized. The surrounding nodes are held fixed during the minimization process. More specifically, the optimal location of node i is given by

$$(x_i', y_i') = \arg \min_{(x_i, y_i)} \sum_{k=1}^{N_i} q_k^2 \tag{C.2}$$

Thus the procedure positions node i such that the surrounding triangles are as close as possible, in a least squares sense, to equilateral triangles. This procedure is applied sequentially to selected nodes in the grid (see Section 3.5).

Appendix D

Continuous Adjoint Correction Theory

A brief outline of the continuous adjoint correction theory of Pierce and Giles is presented. The subsequent treatment focuses on the linear theory and follows the presentation given in references [25, 26, 49]. The reader is referred to the same references for the nonlinear continuous theory. Analogies are made with the linear and nonlinear algebraic theory of Sections A.4 and 2.1.2, respectively.

D.1 Theory Without Boundary Terms

Let u be the solution of the linear partial differential equation

$$\mathcal{L}u = f, \tag{D.1}$$

in the domain Ω, subject to homogeneous boundary conditions. The adjoint differential operator $\mathcal{L}^*$ and associated homogeneous boundary conditions are defined by the identity

$$(v, \mathcal{L}u) = (\mathcal{L}^*v, u)\,, \tag{D.2}$$

for all u and v satisfying the respective boundary conditions. The notation $(\cdot\,,\cdot)$ represents an integral inner product over the domain Ω.

We are concerned with the functional output

$$J = (g, u)\,, \tag{D.3}$$

where g is a given function. An equivalent dual formulation of the problem is to determine the functional

$$J = (v, f)\,, \tag{D.4}$$

where v satisfies the adjoint equation

$$\mathcal{L}^* v = g, \tag{D.5}$$

subject to homogeneous boundary conditions. Equivalence of the two forms of the problem follows from

$$(v, f) = (v, \mathcal{L}u) = (\mathcal{L}^* v, u) = (g, u)\,. \tag{D.6}$$

Let u_H and v_H be approximations to u and v, respectively, that satisfy the homogeneous boundary conditions. We assume that these approximations are derived from a numerical discretization such as a finite element or finite volume approximation on a grid with average element size H. When using a finite volume method, u_H and v_H might be created by interpolation through computed values at the nodes. With finite element methods, one might use the finite element solutions themselves, or one could again use interpolation through nodal values to obtain approximate solutions that are smoother than the finite element solutions. Substituting u_H and v_H into the

	Linear–Continuous	Linear–Algebraic	Nonlinear–Algebraic
Exact Functional	(g,u)	$g_h^T u_h$	$f_h(U_h)$
Approximate Functional	(g,u_H)	$g_h^T \tilde{u}_h$	$f_h(\tilde{U}_h)$
Computable Correction	$(v_H,\mathcal{L}u_H - f)$	$\tilde{v}_h^T(L_h\tilde{u}_h - f_h)$	$\tilde{\Psi}_h^T R_h(\tilde{U}_h)$
Remaining Error	$(v_H - v,\mathcal{L}u_H - f)$	$(\tilde{v}_h - v_h)^T(L_h\tilde{u}_h - f_h)$	$(\tilde{\Psi}_h - \Psi_h)^T R_h(\tilde{U}_h)$
Remaining Error	$(\mathcal{L}^*v_H - g,u_H - u)$	$(L_h^T\tilde{v}_h - g_h)^T(\tilde{u}_h - u_h)$	$R_h^{\Psi}(\tilde{\Psi}_h)^T(\tilde{U}_h - U_h)$

Table D.1: Terms associated with the algebraic linear correction theory of Section A.4 and the algebraic nonlinear correction theory of Section 2.1.2 that are analogous to the terms in the continuous linear theory of Pierce and Giles [25, 26, 49].

primal and adjoint partial differential equations yields

$$\begin{aligned}\mathcal{L}u_H &= f_H,\\ \mathcal{L}^*v_H &= g_H.\end{aligned}\tag{D.7}$$

Using the definitions and identities above, the functional can be expressed as

$$\begin{aligned}J &= (g,u),\\ &= (g,u_H) - (g_H,u_H - u) + (g_H - g,u_H - u),\\ &= (g,u_H) - (\mathcal{L}^*v_H,u_H - u) + (\mathcal{L}^*v_H - g,u_H - u),\\ &= (g,u_H) - (v_H,\mathcal{L}u_H - f) + (v_H - v,\mathcal{L}u_H - f),\end{aligned}\tag{D.8}$$

where (g,u_H) is the functional estimate computed using the numerical solution u_H, $(v_H,\mathcal{L}u_H - f)$ is the computable adjoint correction, and $(\mathcal{L}^*v_H - g,u_H - u)$ and $(v_H - v,\mathcal{L}u_H - f)$ are two equivalent forms of the remaining error after correction. Table D.1 shows analogous terms associated with the algebraic linear correction theory of Section A.4 and the algebraic nonlinear correction theory of Section 2.1.2. One difference between the algebraic and continuous approaches is that the algebraic expression for the functional, $g_h^T u_h$ or $f_h(U_h)$, already contains boundary integral contributions whereas the continuous expression, (g,u), does not. Boundary integrals are added as separate terms in the continuous case as will be seen in the next section.

D.2 Theory With Boundary Terms

The theory is extended to include inhomogeneous boundary conditions for the primal and dual problems, and boundary integrals in the output functional. The subsequent presentation follows references [25, 26].

Let u be the solution of the linear partial differential equation

$$\mathcal{L}u = f, \tag{D.9}$$

in the domain Ω, subject to the linear boundary conditions

$$\mathcal{B}u = e, \tag{D.10}$$

on the boundary $\partial\Omega$. The form of the operator $\mathcal{B}$ may differ on different parts of the boundary (e.g. inflow and outflow boundaries for convection).

The output of interest is taken to be

$$J = (g, u) + (h, \mathcal{C}u)_{\partial\Omega}\,, \tag{D.11}$$

where $(\cdot\,,\cdot)_{\partial\Omega}$ represents an integral inner product over the boundary. The boundary operator $\mathcal{C}$ may be algebraic (e.g. $\mathcal{C}u \equiv u$) or differential (e.g. $\mathcal{C}u \equiv \partial u/\partial n$), but must have the same dimension as the adjoint boundary operator $\mathcal{B}^*$ to be defined below.

The corresponding linear adjoint problem is

$$\mathcal{L}^*v = g, \tag{D.12}$$

in Ω, subject to the boundary conditions

$$\mathcal{B}^* v = h, \tag{D.13}$$

on $\partial\Omega$. The fundamental identity defining $\mathcal{L}^*$, $\mathcal{B}^*$, and the boundary operator $\mathcal{C}^*$ is

$$(v, \mathcal{L}u) + (\mathcal{C}^* v, \mathcal{B}u)_{\partial\Omega} = (\mathcal{L}^* v, u) + (\mathcal{B}^* v, \mathcal{C}u)_{\partial\Omega}, \tag{D.14}$$

for all u and v. This identity is obtained by integration by parts [24]. The equivalent dual form of the output functional is obtained immediately from D.14 as

$$J = (v, f) + (\mathcal{C}^* v, e)_{\partial\Omega}, \tag{D.15}$$

Given the approximate numerical solutions u_H and v_H, the functional can be expressed as

$$\begin{aligned}
J &= (g, u) + (h, \mathcal{C}u)_{\partial\Omega}, \\
&= (g, u_H) + (h, \mathcal{C}u_H)_{\partial\Omega} \\
&\quad - (g_H, u_H - u) - (\mathcal{B}^* v_H, \mathcal{C}(u_H - u))_{\partial\Omega} \\
&\quad + (g_H - g, u_H - u) + (\mathcal{B}^* v_H - h, \mathcal{C}(u_H - u))_{\partial\Omega}, \\
&= (g, u_H) + (h, \mathcal{C}u_H)_{\partial\Omega} \\
&\quad - (\mathcal{L}^* v_H, u_H - u) - (\mathcal{B}^* v_H, \mathcal{C}(u_H - u))_{\partial\Omega} \\
&\quad + (\mathcal{L}^* v_H - g, u_H - u) + (\mathcal{B}^* v_H - h, \mathcal{C}(u_H - u))_{\partial\Omega}, \\
&= (g, u_H) + (h, \mathcal{C}u_H)_{\partial\Omega} \\
&\quad - (v_H, \mathcal{L}u_H - f) - (\mathcal{C}^* v_H, \mathcal{B}u_H - e)_{\partial\Omega} \\
&\quad + (v_H - v, \mathcal{L}u_H - f) + (\mathcal{C}^*(v_H - v), \mathcal{B}u_H - e)_{\partial\Omega}.
\end{aligned} \tag{D.16}$$

In this last expression, $(g, u_H) + (h, \mathcal{C}u_H)_{\partial\Omega}$ is the functional estimate computed using the approximate solution u_H, and $(v_H, \mathcal{L}u_H - f) + (\mathcal{C}^* v_H, \mathcal{B}u_H - e)_{\partial\Omega}$ is the computable adjoint correction.

We see from (D.16) that the computable correction term in the continuous framework includes an extra boundary term related to the residual error in satisfying the primal boundary conditions. This boundary contribution to the functional error is already incorporated into the linear algebraic correction term $\tilde{v}_h^T(L_h\tilde{u}_h - f_h)$ (see Section A.4) and the nonlinear algebraic correction term $\tilde{\Psi}_h^T R_h(\tilde{U}_h)$ (see Section 2.1.2) due to the fact that primal boundary conditions are already incorporated into the primal residual operator $L_h\tilde{u}_h - f_h$ or $R_h(\tilde{U}_h)$, and that boundary integrals are already incorporated into the expression for the functional $g_h^T\tilde{u}_h$ or $f_h(\tilde{U}_h)$. In the linear algebraic theory, if the discrete adjoint is decomposed into a boundary component $\tilde{v}_h^{(b)}$ and an interior component $\tilde{v}_h^{(i)}$ as defined in (A.23), then a somewhat analogous form for the algebraic correction term can be expressed as

$$\delta J_h \approx \left(\tilde{v}_h^{(i)}\right)^T (L_h\tilde{u}_h - f_h) + \left(\tilde{v}_h^{(b)}\right)^T B_h(\tilde{u}_h - e_h), \tag{D.17}$$

where B_h is a projection matrix that sets the boundary conditions in a strong sense as defined in Section A.4.3. Note the similarity in form between (D.17) and the second–to–last line in (D.16). In the nonlinear algebraic context, the form of the correction term analogous to (D.17) is

$$\delta f_h \approx \left(\tilde{\Psi}_h^{(i)}\right)^T R_h(\tilde{U}_h) + \left(\tilde{\Psi}_h^{(b)}\right)^T \delta\bar{U}_h, \tag{D.18}$$

where $\bar{U}$ is a vector of transformed primal variables as defined in Section 4.4.