To The Graduate School:

The members of the Committee approve the dissertation of Karthik Mani presented on March 27, 2009.

Dimitri J. Mavriplis, Chairperson

Frederico Furtado, Graduate Faculty Representative

Jonathan W. Naughton

Douglas R. Smith

Paul A. Dellenback

Lakshmi N. Sankar

APPROVED:

Paul A. Dellenback, Head, Department of Mechanical Engineering

Don Roth, Dean, The Graduate School

Mani, Karthik, Application of the Discrete Adjoint Method to Coupled Multidisciplinary Unsteady Flow Problems for Error Estimation and Optimization, Ph.D., Department of Mechanical Engineering, May, 2009.

Adjoint methods have found applications in several key areas of computational fluid dynamics (CFD), namely, shape optimization and goal based adaptive solutions. CFD has become an essential tool in the design process by enabling the rapid testing of multiple designs, and currently it is normal practice to use CFD in conjunction with optimization algorithms for design improvement. In the context of shape optimization problems based on CFD, adjoint methods offer the significant advantage of computing sensitivity derivatives of the optimization cost function with respect to the set of design parameters, at a cost that is essentially independent of the number of design parameters. Adjoint methods reduce the cost of obtaining the complete gradient vector at any point in the design space equivalent to that of a single flow solution at the same point in the design space. This immediately enables the use of all gradient based optimization algorithms and lifts any restrictions on the number of design parameters required for the adequate definition of the optimization problem.

Adaptive techniques in CFD constitute the other aspect where adjoint methods have have made great inroads. Typical adaptive solutions of the governing flow equations rely on estimating the local error in an evolving solution to target regions of the computational mesh for increased discrete resolution. The main goal of any adaptive solution method is the overall increase in solution accuracy with minimal increase in computational cost. However, targeting local error in the solution does not translate into efficient use of computational resources, since ultimately it is the accurate estimation of boundary integrated functional quantities such as load coefficients that are of importance to the user. Contrary to local error-based methods, adjoint methods allow the adaptation of the computational mesh specifically for the improvement of functionals such as load coefficients. This is achieved by mathematically establishing a clear relationship between the functional of interest and the regions of the computational mesh that are most relevant to it.

The current work extends the use of adjoint methods to multiple governing disciplines that are tightly coupled, and more importantly unsteady in nature. The adjoint method is derived in a very general form for the purpose of computing the gradient vector for use in shape optimization in the context of coupled multidisciplinary unsteady equations. It is shown that computing the gradient vector in unsteady problems involves solving the analysis problem forward in time and then solving the adjoint problem backward in time. While adjoint methods have been used successively to drive spatial mesh adaptation, the current work extends the use of the computed unsteady adjoint variables for estimating temporal discretization error, which is then applied to temporal mesh adaptation. Additionally, the computed adjoint variables are also used for the estimation of algebraic error in the solution arising due to intentional or nonintentional partial convergence of the governing equations. Results indicate that the adaptation of the temporal resolution and convergence tolerance limits using adjoint-based error estimates is able to outperform traditional adaptation methods such as uniform refinement and those based on local error estimates. All of the development is carried out in a fully unstructured mesh framework with dynamic deformation of the computational spatial mesh.

APPLICATION OF THE DISCRETE ADJOINT METHOD TO COUPLED MULTIDISCIPLINARY UNSTEADY FLOW PROBLEMS FOR ERROR ESTIMATION AND OPTIMIZATION

by

Karthik Mani

A dissertation submitted to the Department of Mechanical Engineering
and The Graduate School of The University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
MECHANICAL ENGINEERING

Laramie, Wyoming
May, 2009

To the advancement of Science

Contents

List of Figures

List of Tables

Acknowledgments

I would like express my sincere gratitude to my doctoral advisor, Professor Dimitri J. Mavriplis. He has been a tremendous force in shaping my attitude toward research over the past three and a half years. He has also unconditionally supported me the entire time and I am greatly indebted to him for all of the knowledge I have gained in the field. I would also like to thank the other members of my committee for their support in various forms throughout my time here.

I am particularly grateful to David Vaughn, currently at CD-adapco, for supporting and believing in me at a time when I was faced with difficult decisions in my professional life. I would also like to extend my thanks to my Masters degree advisor, Professor Lakshmi N. Sankar at the Georgia Institute of Technology for all of his support, most notably his willingness to serve on my doctoral thesis committee.

My life in Wyoming as a graduate student would have been very uninteresting had it not been for the company of my fellow students and postdocs within and beyond the CFD lab. Sincere thanks to all of you for all the conversations, parties, lunches, dinners and travels together.

Karthik Mani

The University of Wyoming

May 2009

Chapter 1

Introduction

Advances in numerical simulations of complex fluid flows over the past two decades have firmly established the field of computational fluid dynamics (CFD) as an integral part of aerodynamic analysis and design. In the early years, the lack of computational resources and robust algorithms limited the overall impact of CFD on the design process. For a while, challenges in accurately predicting fluid dynamic phenomena within reasonable amounts of time were considered to be the foremost shortcoming hindering the widespread use of numerical simulations. However, with computational capability increasing by leaps and bounds, some of these issues present less of a problem nowadays. It is now possible to numerically solve the steady-state Reynolds-Averaged-Navier-Stokes (RANS) equations around complex two-dimensional geometries within a couple of hours on an inexpensive mainstream desktop computer. While large three-dimensional steady-state solutions of the RANS equations may take a couple of days on a desktop computer running a single threaded process, parallelization of the same problem on a small compute-cluster reduces this time to a matter of hours.

Unsteady or time-dependent solutions are inherently more expensive than steady-state solutions. Much work has been done on improving the efficiency of time-dependent simulations of the RANS equations, particularly for high Reynolds number aerodynamic flows. High Reynolds number aerodynamic flows have large differences between the spatial and temporal scales of the flow features of interest. Such problems are typically solved implicitly at each time step with the total cost of a single unsteady simulation being the cost of the nonlinear solution at each time step multiplied with the number of time steps chosen to represent the time domain of interest. The advent of convergence acceleration methods such as multigrid [1,2], and the parallelization of larger problems on compute-clusters has addressed the first component of the total cost by reducing the computational expense associated with the nonlinear solution at each implicit time step. Reducing the temporal resolution (i.e. the number of required time steps) without loss in overall temporal accuracy is another necessary step for the reduction of the total unsteady simulation cost. To this end, stable and robust higher-order time-integration schemes [3,4] have been investigated that permit similar temporal error levels with fewer time steps to represent the time domain. The high cost of unsteady simulations has also prompted the development of frequency-domain approaches [5,6], which have been shown to be more economical for problems with periodic behavior. Although time domain methods may be more expensive for such problems, they can be expected to be more suitable in the general case for problems with no dominant periodic behavior, and the two approaches should be considered complimentary to one another.

1.1 Background

1.1.1 Dynamic Meshes

While unsteady fluid flow phenomena such as vortex shedding from blunt bodies may not involve motion of the geometry, most interesting problems involve motion and/or deformation of the geometric boundaries. Rigid body motion of geometries can theoretically be modeled using rigid computational mesh rotations and translations or by the appropriate variation of the boundary conditions. However, geometric deformations can be handled only via the deformation of the computational mesh. Good examples of such problems include control surface deflections on aircraft undergoing maneuvers, relative motion problems such as armament detachment, and forward flight in rotorcraft. Dynamic mesh algorithms [7–11], which deform the computational mesh based on prescribed displacements of geometric boundaries become necessary for such problems. The governing flow equations, which are typically solved in an Eulerian framework also have to be modified into a Lagrangian-Eulerian framework to handle mesh motion and include mesh velocity terms. It is also important that the mesh velocities be determined in accordance with the Geometric-Conservation-Law [12] so as to not introduce conservation errors into the flow solution, which in simpler terms implies the preservation of a uniform flow in the presence of mesh deformation. Another aspect is the recovery of the formal temporal accuracy of the chosen temporal discretization in the presence of mesh deformation [12, 13]. This is particularly important in the context of retaining computational savings achieved by the use of higher-order time-integration schemes when dynamic meshes are involved.

1.1.2 Unstructured Methodology

Numerical simulations performed on structured meshes rely on a body conforming mapping of a uniformly spaced cartesian mesh onto the geometry of interest. Most of the computational methods such as higher-order finite-difference based spatial discretizations [14] were developed assuming solutions were to be obtained on structured meshes with the ordered availability of large stencils. The complexity of geometries that can be represented by structured meshes is however somewhat limited. Multi-block and overset methodologies alleviate the problem but require sophisticated data management and coding practices. Unstructured meshes by definition do not require any blocking strategy and offer the flexibility of meshing virtually any type of geometry beginning with a surface triangulation of the geometry. However, unstructured meshes do not have implied connectivity between neighboring points and require the explicit storage of such information. While they may offer ease of meshing complex geometries, they have higher memory requirements.

1.1.3 Coupled Multidisciplinary Equations

With improvements to unsteady simulation capabilities over complex geometries along with the inclusion of dynamic mesh effects, analysis involving the coupling between disciplines such as that in aeroelasticity between fluid flow and structures is a natural extension. Large scale aeroelastic simulations in CFD are becoming more and more commonplace [15]. Aeroelastic simulations involve the simultaneous solution of the flow, mesh motion and structural equations in a fully coupled manner. The solution of the flow equations results in aerodynamic loads acting on the geometry, which is then deformed or displaced as a consequence. The coupling arises from the fact that any displacements or deformations of the geometry in turn affects the flow field around it thus changing the loads. The solution to the overall coupled system must therefore be obtained iteratively going back-and-forth between the coupled disciplines until the solutions of all disciplines have converged. Aeroelastic simulations tend to be more expensive than purely unsteady simulations of uncoupled disciplines because of the added computational expense of a fully-coupled nonlinear solution at each time step. Aeroelastic solutions may be steady or unsteady in nature. Steady-state aeroelasticity problems involve the deformation or displacement of geometries from some neutral position to a strained position as a reaction to the loads generated by the flow field. Such situations arise in jetliners operating at cruise conditions, where the wings attain a steady deflected state from their neutral position. Unsteady aeroelasticity typically involves a periodic response of the geometry to aerodynamic loads acting on it. Common situations for unsteady aeroelasticity include flutter in aircraft wings, and periodic rotor blade deformations in helicopters.

1.1.4 Adaptive Solvers

Adaptive methods for the solution of the governing flow equations are a means of reducing the overall cost of simulations. The basic principle behind adaptive solutions is the efficient deployment of computational resources. Adaptation of the computational domain takes on several different forms such as h, p, or r adaptation. The most common form is h-adaptation [16–18] where the discretization resolution is increased through the addition of points or elements to the computational mesh wherever necessary. p-adaptation [19, 20] is more commonly used in variational frameworks such as Discontinuous-Galerkin discretizations, which use a high-order polynomial representation of the solution within each element. In this method, the degree of the polynomial used within each element is allowed to vary throughout the computational domain based on some criteria. r-adaptation [21] involves the moving of mesh points in order to change the clustering without the addition of new points or elements.

In the case of finite-volume or finite-difference discretizations of the governing flow equations, the computational expense of obtaining a solution is directly proportional to the resolution of the mesh. However for two different meshes that have the same number of points, it is possible to obtain different error levels in the overall solution due to different distributions of the mesh points. Non-adaptive solutions rely on user judgment for the clustering of mesh points in various regions of the computational domain in order to capture flow phenomena. The normal practice consists of using some predetermined clustering of points in high gradient regions such as boundary layers, wakes, stagnation points, shock waves and contact discontinuities. Since the flow field being solved for is unknown, the user has to rely on engineering judgment for the determination of regions where clustering may be required. Adaptive solvers approach this problem by starting with a coarse resolution mesh and then add points on-the-fly during the solution process in relevant regions of the computational domain. The core of any adaptation algorithm is the determination of relevant regions to target for increased resolution. A common adaptation method is to use the spatial gradients of the evolving flow solution to refine regions of high gradients in the hopes that the overall solution error level is reduced. While the approach may capture interesting flow phenomena such as vortices and shock waves, there is no guarantee that the solution obtained in this manner necessarily results in improved estimates for specific objective values of interest such as lift or drag. To this end, goal-based methods for the adaptation of the computational domain have been developed to specifically target the error in objective scalars of interest. Goal-based methods use *a posteriori* estimates of the error in a functional computed using a flow solution to identify regions in the computational domain that are most relevant to the accuracy of the functional. For different scalars of interest computed using the same flow solution (i.e. identical flow conditions and geometry), it is entirely possible that goal-based adaptation produces significantly different adaptations.

1.2 Design Optimization

One of the fundamental purposes of aerodynamic simulations is that of shape design. Before the advent of numerical simulations, aerodynamic design solely relied on engineering judgment and experimental investigation. Numerically solving the governing equations however opens up the means to study multiple designs without expensive experiments. In the early days of CFD, design methodologies utilized in conjunction with experiments were simply carried over to the computational world, with CFD supplementing or substituting for the wind tunnel. Over time, design methodologies specific to computational methods have been developed and used quite effectively in aerodynamic design. Currently there is a whole field devoted to design optimization algorithms and software packages [22, 23].

1.2.1 Gradients for Optimization

Typical optimization problems involve the determination of the minimum of a given function. The goal is to identify the values for parameters that are inputs for the function such that the output of the function is a minimum. If the slope (or gradient) of the function at any point is known, it becomes possible to move toward the minimum by moving in the direction of negative slope. The gradient can also be interpreted as the sensitivity of the functional output to the parameters that control the function. In the context of aerodynamic shape design, functional quantities of interest are usually surface integrated values such as the lift, drag, or moment. Design parameters usually are the variables that control the shape of the geometry and the governing flow equations form the function relating the design variables to the output functional. If gradient-based optimization is to be employed for the purpose of reducing the output functional, the goal then is to determine the gradient or sensitivity of the output with respect to the design variables.

The obvious choice for determining such sensitivities is to perturb the design variables individually and run the numerical simulation (evaluation of the function) for each perturbation in order to determine the effect of the perturbation on the quantity of interest. The method, known as finite-differencing, becomes inefficient as the number of design variables increases since the number of times the simulation has to be performed is directly dependent on the number of design variables. The sensitivity or the gradient determined in this manner is also highly dependent on the magnitude of the perturbation, since for large perturbations the wrong slope may be predicted due to nonlinear effects, while very small perturbations are susceptible to machine precision related issues. In a perfect world using a computer with infinite precision, an infinitesimally small perturbation would recover the analytically exact value of the gradient.

Alternative optimization methods such as the genetic algorithm (GA) [24] that do not require gradient information have been developed and partially address such problems. GA type methods also offer the benefit of being able to navigate through highly nonlinear, noisy and discontinuous design environments. However, such methods usually require very large numbers of function evaluations and present their own set of difficulties. For situations involving aerodynamic simulations where the cost of evaluating the function is very high, gradient-based optimization is usually the only feasible approach. Also, for problems with smooth design spaces and a single extremum or where only local optima are important, gradient-based optimization is the ideal choice. The primary problem then becomes the rapid determination of the gradient vector and not the optimization process itself.

1.2.2 Adjoint Method

In the recent past, the use of adjoint equations has become a popular approach for solving aerodynamic design optimization problems based on CFD [11,25–29]. Adjoint equations are a very powerful tool in the sense that they enable the computation of sensitivity derivatives of an objective functional to a set of given inputs at a cost that is essentially independent of the number of inputs. Adjoint methods rely on direct differentiation of the governing equations in order to obtain an analytic form of the sensitivity derivative to avoid the accuracy issues presented by the perturbation in the finite-difference method. Also, the sensitivity equations are transposed and solved for the gradient by introducing an adjoint variable in order to remove dependence on the number of input variables. It is possible to differentiate the continuous form of the governing flow equations to obtain a continuous form of the sensitivity derivative which may then be discretized appropriately and solved to obtain the gradient. This is the continuous linearization approach and results in a discrete approximation to the exact analytic derivative of the continuous governing equations. It is also possible to differentiate the discretized form of the governing equations in order to directly obtain the discrete version of the sensitivity derivative. For consistent discretizations, the gradient computed using both methods asymptotically converge to the same answer as the discretization resolution is increased. Both methods possess their own set of advantages and disadvantages. The continuous linearization approach offers flexibility in the discretization stage while the discrete linearization approach traditionally follows the discretization of the governing equations. The determination of boundary conditions for the analytically differentiated continuous governing equations may pose challenges and rely more on the fundamental mathematical behavior of the governing PDEs, whereas in the discrete approach they simply are discrete derivatives of the boundary conditions of the governing equations. The discrete approach also provides exact sensitivities for the discretized governing problem and is more relevant in the context of using CFD for design optimization. This is because it is the discretely evaluated objective functional that is always the target of the optimization. Figure 1.1 compares the paths of the two approaches beginning with the continuous governing equations to the final computed gradient.

Solutions of the steady-state aerodynamic optimization problem utilizing adjoint methods [11, 30–33] are now fairly well established. However, relatively little work has been done in applying these methods for unsteady time-dependent problems [34–36]. More recently, unsteady time-domain adjoint methods have been investigated by Rumpfkeil et al. [36], although this work is in the context of structured meshes using relatively simple mesh deformation strategies. While only recently adjoint methods have begun gaining ground in the aerodynamics community especially for time-dependent problems [37, 38], they have been well established in the field of structural optimization for some time now [39]. The coupling between the fluid and structure equations and the use of sensitivity analysis on such a system has been addressed but again primarily from a steady-state standpoint [40–42]. Unsteady aeroelastic optimization has been attempted but only using reduced order [43] or other simplistic models for the flow equations [44, 45]. Optimization based on the fully-coupled unsteady aeroelastic equations with no simplifying assumptions has not been tackled thus far mainly due to prohibitive cost and complexities in the linearization

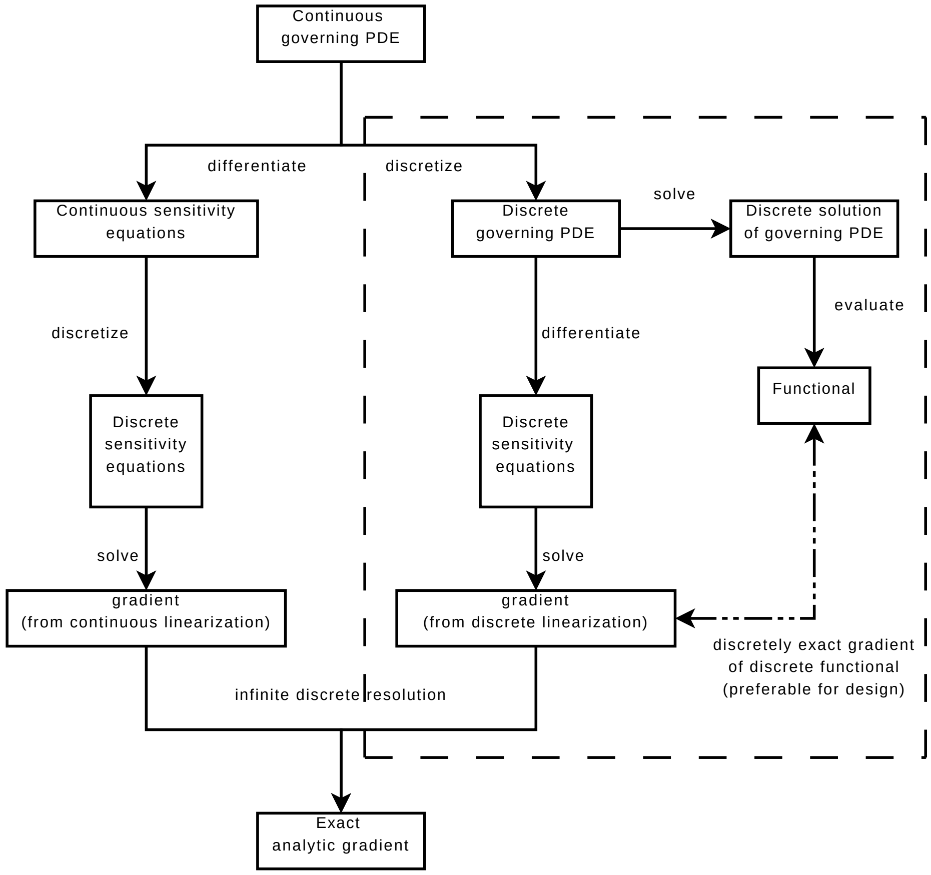

Continuous governing PDE
differentiate
discretize
Continuous sensitivity equations
Discrete governing PDE
solve
Discrete solution of governing PDE
discretize
differentiate
evaluate
Functional
Discrete sensitivity equations
Discrete sensitivity equations
solve
solve
gradient (from continuous linearization)
gradient (from discrete linearization)
discretely exact gradient of discrete functional (preferable for design)
infinite discrete resolution
Exact analytic gradient

Figure 1.1: Comparison of continuous and discrete linearization approaches and the advantage of the discrete method for CFD based design.

of coupled time-dependent systems.

1.3 Error Estimation and Adaptation

The purpose of adaptation is to reduce the overall error in the solution without incurring severe increases in computational expense. The contributions to the overall error in a solution include the discretization error (which can be tied to the error due to lack of resolution) and algebraic error. Algebraic error arises from the incomplete solution of the discretized governing equations irrespective of the discretization error. Given a computational mesh with a certain number of elements with some arbitrary distribution of element sizes, the accuracy of the solution may be improved by increasing the order-of-accuracy of the discretization of the governing equations. Conversely, it is also possible to improve the solution accuracy by increasing the resolution of the computational mesh while utilizing some fixed order-of-accuracy for the discretization of the governing equations. In the case of a finite-volume method applied on unstructured meshes, it is often difficult to construct higher-order discretizations of the governing equations due to the lack of a simple data structure for the large computational stencils required. In most cases the equations are discretized for only second-order accuracy. Combined with the inherent challenge of clustering points in unstructured mesh generation, this necessitates the adaptation of the computational mesh.

The concept of adaptation is easier to visualize in the context of steady-state simulations since only the spatial domain exists. However the arguments presented above regarding error apply to the temporal domain also. In addition to the spatial discretization error and the algebraic error, unsteady simulations are also affected by the temporal discretization error. For iterative solution methods, the algebraic error can be reduced by performing more iterations during the solution process.

1.3.1 Adaptation for the Reduction of Discretization Error

Adaptation of the spatial domain has existed in various forms for well over a decade. The most popular adaptation methods such as those based on spatial gradients are still local methods in the sense that they refine the spatial resolution in regions of the computational domain where the local discretization error is high. Considering that output functionals such as surface integrated loads are most often the goal of an aerodynamic simulation, such local methods cannot guarantee improved estimates for the functional. There may be little or no correlation between the local error distribution and the functional of interest. However, goal-based or adjoint-based adaptation directly targets the discretization error that is relevant to the functional of interest by applying the adjoint variable as a functional relevant weight on the local error in the solution. The functional relevance in the adjoint variable is introduced through its definition in terms of the Taylor expansion of the functional between two spatial meshes of different resolutions. The overall method is *a posteriori* since the solution to the flow problem on a spatial mesh of some arbitrary resolution is required before the error can be estimated. For steady-state problems, adjoint methods have been used quite successfully to drive adaptive spatial mesh refinement for reducing the spatial discretization error relevant to output functionals [16, 17, 46–51]. The method has matured to the point where it is now employed in a production level cartesian mesh based steady-state solver [52, 53].

For unsteady flow problems, the time-integration process is typically carried out using a uniform time step which is applied throughout the time domain of interest. The limitation on the size of the time step is mainly governed by the temporal resolution required to capture essential unsteady flow features and deliver acceptably low temporal error. The traditional approach is to use a uniform time-step of a size that by engineering knowledge is known to be sufficient to capture essential flow features. However, the inherent temporal nonlinearity of the flow equations makes it difficult to determine the temporal scales of the flow features that are relevant to the functional output of interest from the unsteady simulation. There are no obvious high gradient features analogous to boundary layers and shock waves in the temporal domain, and the user must rely on the computed temporal gradient of the solution for identification of such features. However, using a uniform time-step due to the lack of such knowledge results in unnecessarily high computational expense. Uniform time-steps result in unnecessarily high resolution in certain regions of the time domain and low resolution in other regions that may be important.

Much work has been done on temporal adaptation and error control in the absence of spatial adaptation for ordinary differential equations [54, 55]. The most common approach consists of using local temporal error estimation to drive time-step size selection, where the temporal error at a given time-step is estimated by discretizing the time derivative with different orders-of-accuracy and comparing the corresponding results [56]. Also, such methods assume that any solutions obtained at each implicit time step are numerically exact, or at least converged to levels below the local temporal discretization error, and do not consider the effect of partially converged solutions. While local error estimation has been used successfully for temporal refinement, a more appealing approach would be one that targets the global temporal error for a functional of interest directly. As in the case of the purely spatial problem, there is no guarantee that a gradient-based or any other local error-based adaptation of the time domain would necessarily lead to an improved estimate of the functional. Analogous to the spatial argument, there may be little or no correlation between the flow solution gradients in the time domain and the functional of interest. While adjoint methods have been used in the spatial domain for goal oriented error estimation and control, their use in the time domain has mostly been confined to simpler problems such as those involving ordinary differential equations [57,58], although their use for partial differential equations has recently been demonstrated [59–61].

The total error incurred in time-dependent simulations can be broken down into spatial error, temporal error, and algebraic error. For time-dependent solutions, the use of adaptive spatial mesh refinement techniques requires the construction of a new refined mesh at each time-step, which results in discontinuous processes in time, as new grid points are added and deleted between time-steps. The full error reduction and control problem for time-dependent problems must involve simultaneous adaptation in time and in space.

1.3.2 Algebraic Error for Unsteady Problems

While the overall cost of a simulation may be correlated against the resolution of the computational domain (space and/or time), the primary cost of obtaining the solution of the flow equations arises from the nonlinear iterative solution procedure. In the case of steady-state solutions only a single nonlinear solution is required, while unsteady solutions necessitate a nonlinear solution at each time-step for an implicit time discretization. Typically the equations are never solved to machine precision unless the spatial resolution of the problem under consideration is relatively coarse. Convergence to small tolerance levels can become time consuming for stiff problems particularly in the presence of anisotropic mesh effects. A sensible guideline for determining suitable implicit system convergence levels is that the algebraic error resulting from incomplete system convergence at each time-step be reduced to levels below the local temporal discretization error at each time-step [62, 63]. However, this requires a good measure of the local temporal error as well as some estimate of the algebraic error due to incomplete system convergence, and is seldom enforced. The end result is that, as in the case of temporal resolution, some predetermined constant convergence limit is set based purely on engineering judgment such that the resulting solution has acceptably low overall error at practical computational expense. It is quite possible that the equations converged to limits set by such ad-hoc methods could result in unnecessarily restrictive convergence in some regions of the time domain and insufficient convergence in others. Under most circumstances the effect or error due to insufficient convergence becomes difficult to quantify and even harder to remedy.

1.3.3 Separation of Error Components

An additional advantage of adjoint-based approaches is that they also permit the distinct identification of the contributions to the estimated error as arising from the two primary sources, namely the discretization error and the algebraic error due to partial convergence of the equations. The method also permits the separation of the discretization and algebraic errors into disciplinary components and further breaks it down into a distribution in time. This provides the criteria necessary to target specifically the set of governing equations and the time-steps that influence the functional the most. This is uniquely different from common adaptation or time-step control schemes which are based on estimates of the local temporal error at each time-step, in that the global temporal error in the functional can be estimated and adapted on.

1.4 Overview of Thesis

1.4.1 Objectives

The main goals of this work can be summarized as follows:

- Develop a unified framework for the computation of sensitivity derivatives in the context of steady, unsteady and multidisciplinary coupled unsteady governing equations using adjoint variables.

- Demonstrate the use of the adjoint-based sensitivity derivatives for shape optimization in steady, unsteady and multidisciplinary coupled unsteady flow problems.

- Extend the utility of the computed adjoint variables for the purpose of estimating functional relevant temporal discretization error and algebraic error.

- Demonstrate adaptation of the temporal domain and adaptation of convergence tolerances for temporal and algebraic error reduction in functional outputs using the adjoint-based error estimates.

1.4.2 Approach

All of this is accomplished using a two-dimensional finite-volume discretization of the governing equations on fully unstructured meshes with the inclusion of dynamic mesh effects. The unsteady inviscid Euler equations are used to simulate the flow component of the problem, while the aeroelastic response of an airfoil with two degrees-of-freedom (pitch and plunge) is used to demonstrate coupling between disciplines. Direct differentiation of the discretized governing equations is used to construct the equations whose solution yield the adjoint variables.

Optimization cost considerations

In general, the total cost of an optimization is equal to the number of design iterations required to reach an optimum solution multiplied by the cost of a single design iteration. The cost of a single design iteration can be broken down into the cost of an analysis solution and the cost of obtaining the gradient vector. Reduction of the total number of design cycles to obtain an optimum solution can be achieved by utilizing sophisticated optimization algorithms. In this work the gradient-based reduced Hessian LBFGS-B optimization algorithm with bounds developed in references [64, 65] is used. Considering the fact that unsteady analysis solutions themselves are inherently expensive, all possible methods to improve efficiency both in the case of the flow solver and the sensitivity solver must be exploited to the fullest extent. This necessitates the use of multigrid methods to reduce dependency on mesh resolution and also the use of high-order time-integration schemes to obtain high quality unsteady solutions with a minimal number of time-steps [66]. Conversely, it is important that adjoint methods developed be general enough such that they are applicable to any order of time-integration scheme. Also, convergence acceleration methods such as multigrid must be easily extendable to the developed algorithms. The adjoint formulation presented here partially addresses both of these concerns. The algorithm takes into account higher-order BDF (Backward-Difference-Formula) time-integration schemes and can be extended to higher-order implicit Runge-Kutta schemes without severe additional cost. The algorithm also utilizes the linear multigrid method [1] for convergence acceleration.

Note regarding adjoint-based error estimates

An adjoint-based approach permits the estimation of error relevant to a functional, and the corresponding distribution of this error in the time domain. The method relies on applying time-dependent discrete adjoint equations on a Taylor expansion of the functional. A linear approximation of the error in the functional between temporal meshes of two different resolutions can be estimated by solving the flow problem and the adjoint problem on the coarser temporal mesh, and then projecting the flow and adjoint solutions to the finer temporal mesh. It should be noted that in this particular formulation, only the change in the functional between two different temporal resolutions is predicted by the adjoint equations, and not the total error defined as the difference between the current and exact analytic value of the functional. The current work is restricted to adaptation in the time domain, in order to reduce temporal discretization error, with no attempt to reduce spatial discretization error. However, algebraic error due to incomplete convergence of the governing equations at each time-step is also targeted in this work.

Modularity

Modularity is a key aspect in the development of the framework to compute adjoint variables for coupled governing equations. The method is modular in the sense that multiple sets of governing equations from various disciplines that are coupled may be addressed in a unified manner independent of the number of disciplines. Also, the choice of solution technique employed for the individual disciplines should have little impact on the overall framework. This is particularly important from a cost standpoint since efficient solution techniques for the purpose of reducing overall costs inherent in coupled unsteady aeroelastic simulations should carry over to the adjoint computations. It should also be noted that no approximations in the process of linearization have to be made and all components contributing to the solution of the multidisciplinary analysis problem are taken into account.

Chapter 2

Formulation of the Analysis Problem

2.1 Governing Equations

2.1.1 ALE Integral Form of the Unsteady Euler Equations

The conservative form of the Euler equations is used in solving the flow problem. The current work is limited to inviscid flow problems since the primary focus is the development and verification of the adjoint method for computation of functional sensitivity to design variables and the estimation of functional relevant temporal and algebraic error. Extension of the algorithm to viscous flow problems with the inclusion of turbulence models should prove to be relatively straightforward. The differences arise only in the linearization of the additional flux contributions and not in the base formulation presented.

In vectorial form the conservative form of the Euler equations may be written as

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \tag{2.1}$$

where the state vector $\mathbf{U}$ of conserved variables and the cartesian inviscid flux vector

$\mathbf{F} = (\mathbf{F}^x, \mathbf{F}^y)$ are defined as

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E_t \end{pmatrix}, \quad \mathbf{F}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E_t + p) \end{pmatrix}, \quad \mathbf{F}^y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E_t + p) \end{pmatrix}. \tag{2.2}$$

Here ρ is the fluid density, (u, v) are the cartesian fluid velocity components, p is the pressure and E_t is the total energy. For an ideal gas, the equation of state relates the pressure to total energy by

$$p = (\gamma - 1)\left[E_t - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{2.3}$$

where $\gamma = 1.4$ is the ratio of specific heats. Applying the divergence theorem and integrating over a moving control volume $A(t)$ that is bounded by the control surface $B(t)$ yields:

$$\int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{B(t)} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n} dB = 0 \tag{2.4}$$

Using the differential identity

$$\frac{\partial}{\partial t} \int_{A(t)} \mathbf{U} dA = \int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{B(t)} (\dot{\mathbf{x}} \cdot \mathbf{n}) dB \tag{2.5}$$

equation (2.4) is rewritten as

$$\frac{\partial}{\partial t} \int_{A(t)} \mathbf{U} dA + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{2.6}$$

or when considering cell-averaged values for the state $\mathbf{U}$ as

$$\frac{dA\mathbf{U}}{dt} + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{2.7}$$

This is the Arbitrary-Lagrangian-Eulerian (ALE) finite-volume form of the Euler equations. The equations are required in ALE form since the presented work involves deforming meshes where mesh elements change in shape and size at each time-step. Here A refers to the area or volume of the element, $\dot{\mathbf{x}}$ is the vector of mesh face or edge velocities, $\mathbf{n}$ is the unit normal of the face or edge, and B refers to the area or length of the bounding surface or edge.

The method presented for the estimation of global temporal error involves projections of the flow and adjoint solutions from a given temporal mesh onto a temporal mesh of higher resolution as will be discussed later. In order to remove the dependence of the order-of-magnitude of the computed adjoint variables on the temporal mesh resolution the time-integrated form of equation (2.7) is utilized as

$$\int_T \left\{ \frac{dA\mathbf{U}}{dt} + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB \right\} dt = 0 \tag{2.8}$$

or in a discrete sense when considering a temporal element of size Δt as

$$\left\{ \frac{dA\mathbf{U}}{dt} \right\} \Delta t + \left\{ \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB \right\} \Delta t = 0 \tag{2.9}$$

2.1.2 Linear Mesh Deformation Equations

Deformation of the mesh is achieved through the linear tension spring analogy [7,30], which approximates the mesh as a network of interconnected springs. The spring coefficient κ is assumed to be inversely proportional to the square of the edge length. Two independent force balance equations are formulated for each node based on displacements of neighbors. This results in a nearest neighbor stencil for the final linear system to be solved. Referring to Figure 2.1, the displacement of an arbitrary node i in the mesh can be computed based on displacements of neighbors j as

$$\delta \mathbf{x}_i = \frac{\sum_{j=1}^{j_{max}} (\kappa_{ij} \delta \mathbf{x}_j)}{\sum_{j=1}^{j_{max}} \kappa_{ij}} \tag{2.10}$$

The linear system of equations for the overall computational mesh that relates the interior node displacements to known displacements on the boundaries is given as:

$$[\mathbf{K}] \delta \mathbf{x}_{int} = \delta \mathbf{x}_{surf} \tag{2.11}$$

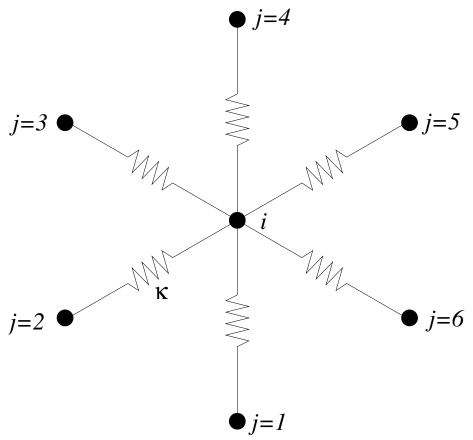


Figure 2.1: Approximation of the computational mesh as a network of springs.

where $[\mathbf{K}]$ is the stiffness matrix assembled using the spring coefficients of each of the edges in the computational mesh. The spring coefficient for each edge is precomputed using the baseline computational mesh and remains constant throughout the solution process. As in the case of the governing flow equations, the mesh motion and the mesh adjoint solutions have to be projected between temporal meshes of different resolutions, and in order to avoid projection related scaling issues the mesh motion equations are redefined in the form of a discrete time-integrated mesh residual as:

$$\mathbf{G}(\mathbf{x}) = \Delta t \left\{ [\mathbf{K}]\delta\mathbf{x}_{int} - \delta\mathbf{x}_{surf} \right\} = 0 \tag{2.12}$$

2.1.3 Nonlinear Aeroelastic Structural Equations

For the purpose of demonstrating the discrete adjoint method on multidisciplinary coupled unsteady governing equations, a two-dimensional aeroelastic model based on the response of an airfoil with two degrees-of-freedom, namely pitch and plunge is used. Referring to Figure 2.2, the equations of motion for such a system can be summarized as

$$\begin{aligned} m\ddot{h} + S_\alpha \ddot{\alpha} + K_h h &= -L \\ S_\alpha \ddot{h} + I_\alpha \ddot{\alpha} + K_\alpha \alpha &= M_{ea} \end{aligned} \tag{2.13}$$

where

m:	mass of airfoil
S_α:	static imbalance, positive for c.g. aft of mid-chord, $S_\alpha = mbx_\alpha$
I_α:	sectional moment of inertia of airfoil
K_h:	plunging spring coefficient
K_α:	pitching spring coefficient
h:	vertical displacement (positive downward)
α:	angle-of-attack
L:	sectional lift of airfoil
M_{ea}:	sectional moment of airfoil about elastic axis (positive nose up)

Non-dimensionalizing equations (2.13) and (2.14) yields

$$[M]\{\ddot{q}\} + [K]\{q\} = \{F\} \tag{2.14}$$

where

$$\begin{aligned} &[M] = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \quad [K] = \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix}, \\ &\{F\} = \frac{1}{\pi \mu k_c^2} \begin{Bmatrix} -C_l \\ 2C_m \end{Bmatrix}, \quad \{q\} = \begin{Bmatrix} \frac{h}{b} \\ \alpha \end{Bmatrix} \end{aligned} \tag{2.15}$$

are the non-dimensional mass matrix, stiffness matrix, load vector and displacement vector and

b:	semichord of airfoil
k_c:	reduced frequency of pitch, $k_c = \frac{\omega c}{2U_\infty}$
μ:	airfoil mass ratio, $\mu = \frac{m}{\pi \rho b^2}$
ω_h, ω_α:	uncoupled natural frequencies of plunge and pitch
x_α:	structural parameter defined as $\frac{S_\alpha}{mb}$
r_α^2:	structural parameter defined as $\frac{I_\alpha}{mb^2}$
C_l, C_m:	section lift coefficient and section moment coefficient about the elastic axis

The reduced frequency k_c is typically written in terms of the flutter velocity V_f as:

$$k_c = \frac{\omega_\alpha c}{2U_\infty} = \frac{1}{V_f \sqrt{\mu}} \tag{2.16}$$

where c is the chord length of the airfoil, U_∞ is the freestream velocity, and the flutter velocity V_f is defined as

$$V_f = \frac{U_\infty}{\omega_\alpha b \sqrt{\mu}} \tag{2.17}$$

2.2 Spatial Discretization

The flow solver uses a cell-centered finite-volume formulation where the inviscid flux integral S around a closed control volume is discretized as

$$S = \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = \sum_{i=1}^{n_{edge}} \mathbf{F}_{e_i}^{\perp}(V_{e_i}, \mathbf{U}, \mathbf{n}_{e_i}) B_{e_i} \tag{2.18}$$

where B_e is the edge length, $V_e = \dot{\mathbf{x}} \cdot \mathbf{n}$ is the normal edge velocity, $\mathbf{n}_e$ is the unit normal of the edge, and $F_e^\perp$ is the normal flux across the edge. The normal flux across the edge (Figure 2.3) is computed using the second-order accurate matrix dissipation scheme [67] as the sum of a central difference and an artificial dissipation term as

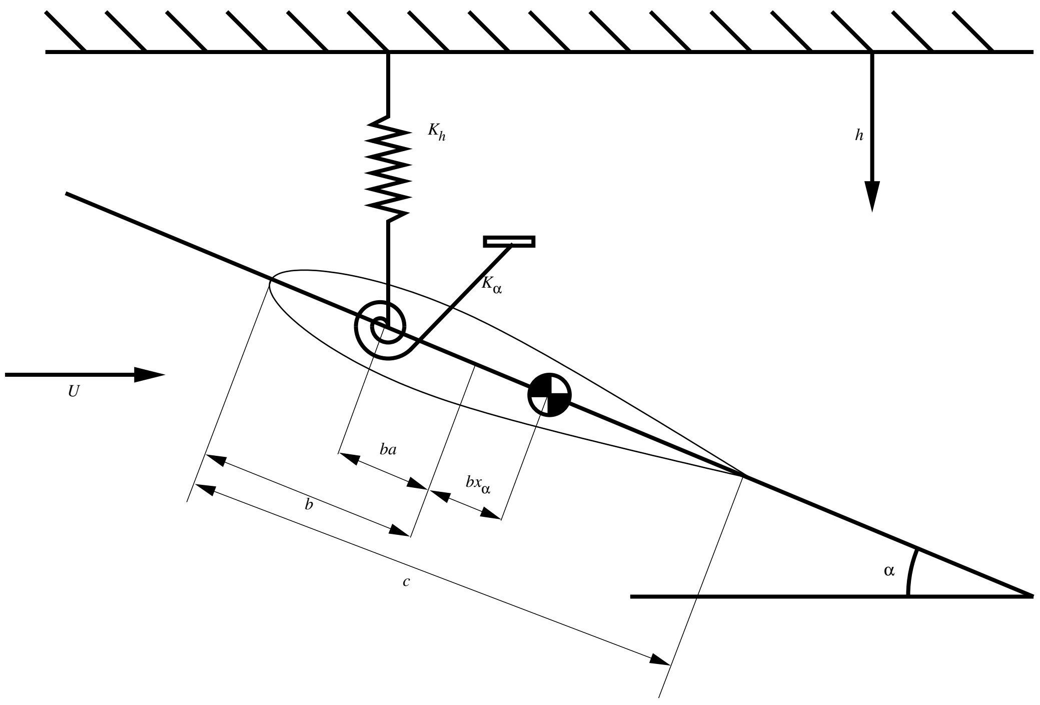


Figure 2.2: Two-dimensional aeroelastic model of an airfoil with pitch and plunge degrees-of-freedom.

shown below.

$$\mathbf{F}_e^{\perp} = \frac{1}{2}\Big\{\mathbf{F}_L^{\perp}(\mathbf{U}_L, V_e, \mathbf{n}_e) + \mathbf{F}_R^{\perp}(\mathbf{U}_R, V_e, \mathbf{n}_e) + \kappa^{(4)}[\mathbf{X}]|[\lambda]|[\mathbf{X}]^{-1}\left\{(\nabla^2\mathbf{U})_L - (\nabla^2\mathbf{U})_R\right\}\Big\} \tag{2.19}$$

where $\mathbf{U}_L$, $\mathbf{U}_R$ are the left and right state vectors and $(\nabla^2\mathbf{U})_L$, $(\nabla^2\mathbf{U})_R$ are the left and right undivided Laplacians computed for any element i as a sum over neighbors as

$$(\nabla^2\mathbf{U})_i = \sum_{j=1}^{neighbors} (\mathbf{U}_j - \mathbf{U}_i) \tag{2.20}$$

The matrix $[\lambda]$ is diagonal and consists of the eigenvalues (adjusted by normal edge velocity V_e) of the flux Jacobian matrix $\frac{\partial \mathbf{F}^{\perp}}{\partial \mathbf{U}}$, and the matrix $[\mathbf{X}]$ consists of the corresponding eigenvectors. The scalar parameter $\kappa^{(4)}$ is empirically determined and controls the amount of artificial dissipation added to the centrally differenced flux. For transonic problems this is usually taken as 0.1. The advantage of using the

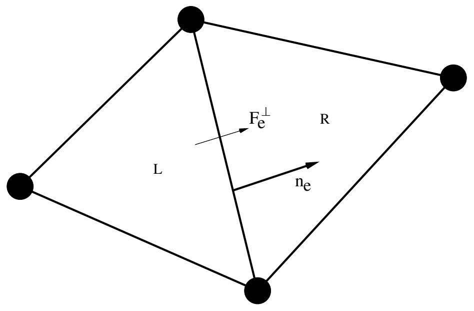


Figure 2.3: Convention for the computation of convective flux across an edge between two triangular elements in an unstructured mesh

difference of the undivided Laplacians in the construction of the convective flux is that it offers second-order spatial accuracy without the need for mesh metrics and state reconstruction required by MUSCL type schemes [68]. The normal native flux vector is computed as

$$\mathbf{F}^\perp = \begin{pmatrix} \rho(V^\perp - V_e) \\ \rho(V^\perp - V_e)u + \hat{n}_x p \\ \rho(V^\perp - V_e)v + \hat{n}_y p \\ E_t(V^\perp - V_e) + pV^\perp \end{pmatrix} \tag{2.21}$$

where the fluid velocity normal to the edge $V^\perp$ is defined as $u\hat{n}_x + v\hat{n}_y$, with $\hat{n}_x$ and $\hat{n}_y$ representing the components of the unit edge normal vector, and V_e is the physical velocity of edge itself in the normal direction.

2.3 Temporal Discretization

2.3.1 Uniform Time-Step Size

The time derivative term in both the flow and structural governing equations are discretized using a second-order accurate BDF2 (Backward-Difference-Formula-2) scheme. The standard BDF2 scheme for the flow equations using a uniform time-step size of Δt is given as

$$\frac{dA\mathbf{U}}{dt} = \frac{\frac{3}{2}A^n\mathbf{U}^n - 2A^{n-1}\mathbf{U}^{n-1} + \frac{1}{2}A^{n-2}\mathbf{U}^{n-2}}{\Delta t} \tag{2.22}$$

2.3.2 Non-Uniform Time-Step Size

One of the objectives of the current work is to adapt the temporal mesh using the estimated functional relevant error. Adaptation of the temporal mesh leads to different time-step sizes in different regions of the temporal domain and therefore the base scheme requires modification to handle non-uniform temporal mesh spacing. This can be done by deriving a second-order accurate backward finite-difference formula using Taylor expansions of the solution over a three point stencil and relaxing the assumption of uniform mesh spacing. The modified BDF2 scheme for non-uniform temporal meshes is given as

$$\begin{aligned}
\frac{dA\mathbf{U}}{dt} &= c_0 A^n\mathbf{U}^n + c_1 A^{n-1}\mathbf{U}^{n-1} + c_2 A^{n-2}\mathbf{U}^{n-2} \qquad (2.23)\\
c_0 &= \frac{2\Delta t_1 \Delta t_2 + \Delta t_2^2}{\Delta t_1 \Delta t_2 (\Delta t_1 + \Delta t_2)}\\
c_1 &= \frac{-(\Delta t_1 + \Delta t_2)^2}{\Delta t_1 \Delta t_2 (\Delta t_1 + \Delta t_2)}\\
c_2 &= \frac{\Delta t_1^2}{\Delta t_1 \Delta t_2 (\Delta t_1 + \Delta t_2)}\\
\Delta t_1 &= T^n - T^{n-1}\\
\Delta t_2 &= T^{n-1} - T^{n-2}
\end{aligned}$$

Although not a sufficient test, it can be seen that substituting ($\Delta t = \Delta t_1 = \Delta t_2$) recovers a standard second-order accurate backward difference with uniform mesh spacing.

2.4 Geometric Conservation Law

The discrete geometric conservation law (GCL) requires that a uniform flow field be preserved when equation (2.7) is integrated in time. In other words, the deformation of the computational mesh should not introduce conservation errors in the solution of the flow problem. This translates into $\mathbf{U} = constant$ being an exact solution of equation (2.7). For a conservative scheme, the integral of the inviscid fluxes around a closed contour goes to zero when $\mathbf{U} = constant$. Applying these conditions to equation (2.7) results in the mathematical description of the GCL as stated below:

$$\frac{\partial A}{\partial t} - \int_{dB(t)} \dot{\mathbf{x}} \cdot \mathbf{n} dB = 0 \tag{2.24}$$

For the second-order BDF2 time-integration scheme utilized in this work, this can be discretely represented using equations (2.18) and (2.23) as:

$$\left(c_0 A^n + c_1 A^{n-1} + c_2 A^{n-2}\right) - \sum_{i=1}^{n_{edge}} V_{e_i} B_i = 0 \tag{2.25}$$

where as described earlier, the integral of the convective flux over a closed contour goes to zero for a uniform flow. The edge velocities V_{e_i} for each of the edges encompassing the element must be chosen such that equation (2.25) is satisfied. While various methods for computing the edge velocities satisfying the GCL have been developed for different temporal discretizations [12, 66], a unifying approach applicable to first, second and third-order backwards differencing schemes (BDF) as well as higher-order accurate implicit Runge-Kutta (IRK) schemes has been developed in reference [13]. Following the procedure outlined in reference [13], the edge velocity $V_{e_i}^n$ at time-step n can be written as a linear combination of the area swept by the edge between three temporal locations n,$n-1$ and $n-2$ as:

$$V_{e_i}^n = k_1(\Delta\Omega^n) + k_2(\Delta\Omega^{n-1}) \tag{2.26}$$

Here $\Delta\Omega^n$ refers to the area swept by the edge between time-steps n and $n-1$, while $\Delta\Omega^{n-1}$ is the area swept between $n-1$ and $n-2$ as shown in Figure 2.4.

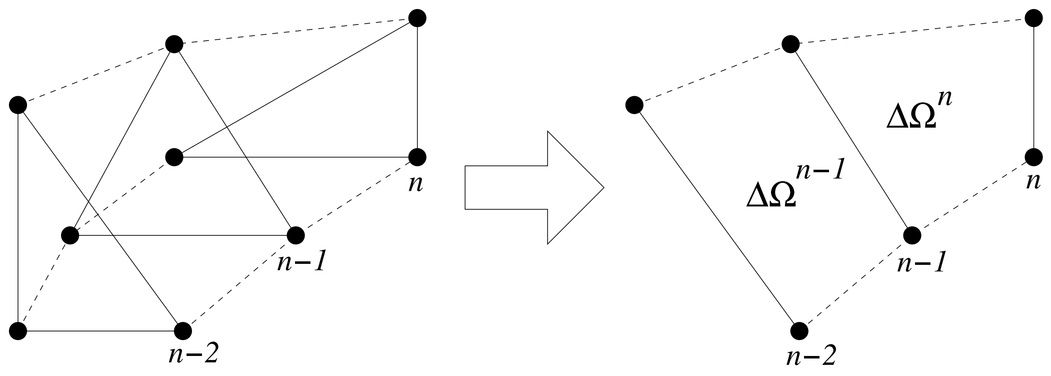


Figure 2.4: A deforming triangular element at three temporal locations n, $n-1$, $n-2$, and the area swept between those locations by a single edge belonging to the element.

The coordinates of the nodes belonging to an edge are known quantities at all three temporal locations, and therefore the areas $\Delta\Omega^n$ and $\Delta\Omega^{n-1}$ swept by the edge can be computed using Green-Gauss integration. The coefficients k_1 and k_2 are functions of the time-step sizes Δt_1 and Δt_2 and are given by:

$$k_1 = \frac{2\Delta t_1 \Delta t_2 + \Delta t_2^2}{\Delta t_1 \Delta t_2 (\Delta t_1 + \Delta t_2)} \tag{2.27}$$

$$k_2 = \frac{-\Delta t_1^2}{\Delta t_1 \Delta t_2 (\Delta t_1 + \Delta t_2)} \tag{2.28}$$

The GCL compliant edge velocity V_e^n at an arbitrary time-step n for any edge in the computational domain is therefore a function of the mesh coordinates belonging to the edge from time indices n, $n-1$ and $n-2$.

2.5 Parametrization of the Geometry

Modification of the baseline geometry is achieved through displacements of the surface nodes defining the geometry. In order to ensure smooth geometry shapes, any displacement of a surface node is controlled by a bump function which influences neighboring nodes with an effect that diminishes moving away from the displaced node. The bump function used for the work presented here is the Hicks-Henne Sine bump function [69]. The design variables or inputs for the optimization examples pre-

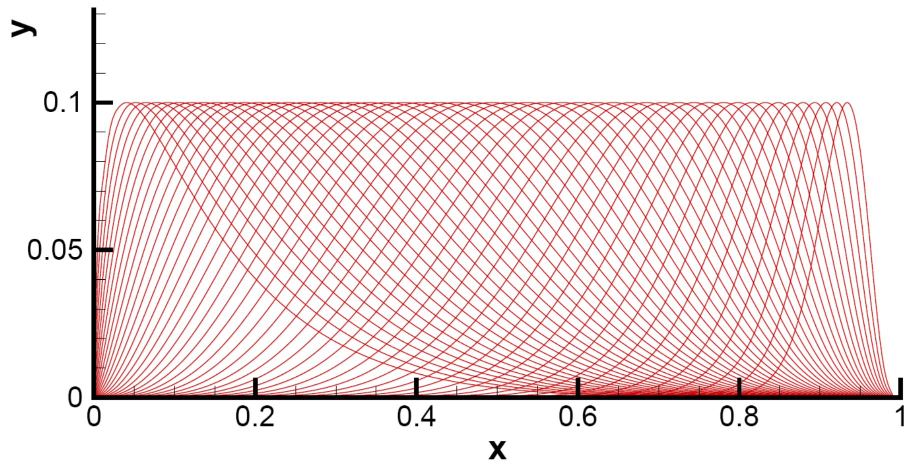


Figure 2.5: Hick-Henne Sine bump functions.

sented form a vector of weights controlling the magnitude of bump functions placed at various chordwise locations. The bump function, although a function of x, does not have any influence in the x direction. The Hicks-Henne bump functions are defined as

$$b_i(x) = a_{max} \cdot \sin^{t_b}(\pi x^{m_i}) \tag{2.29}$$

$$m_i = \ln(0.5)/\ln(x_{M_i}) \tag{2.30}$$

Here x_{M_i} refers to the location where the bump is to be placed. $b_i(x)$ are the displacements of points with x coordinates ranging from 0 to 1 as a result of the bump placed at x_{M_i} in accordance with the sine bump function. a_{max} is the magnitude of the bump placed at x_{M_i} and t_b is a parameter that controls the width of the bump. In most situations a value of 4 is used for t_b. The bump function is valid in the range of $0 \leq x, x_{M_i} < 1$, and is ideally suited for airfoil problems. For geometries extending beyond this range, the bump function may be mapped locally over some predetermined radius about the surface node of interest. The superposition of bump functions is used to augment the existing shape of the baseline airfoil in order to deform its surface. Figure 2.5 shows a series of bumps with a magnitude of 0.1 placed between $0 \leq x < 1$.

2.6 Transformation of the Structural Equations

The aeroelastic equations as shown in Equation (2.14) are second-order partial differential equations. These are transformed into two first-order linear equations in order to facilitate a direct solution procedure. The transformation is given as [70]

$$\begin{aligned} \mathbf{r}_1 &= \{q\} \\ \mathbf{r}_2 &= \dot{\mathbf{r}_1} \end{aligned} \tag{2.31}$$

The resulting first-order equations are then

$$\begin{aligned} \dot{\mathbf{r}_1} &= \mathbf{r}_2 \\ \dot{\mathbf{r}_2} &= -[M]^{-1}[K]\mathbf{r}_1 + [M]^{-1}\{F\} \end{aligned} \tag{2.32}$$

which in matrix notation can be expressed as

$$\left\{ \begin{array}{c} \dot{\mathbf{r}_1} \\ \dot{\mathbf{r}_2} \end{array} \right\} = \left[\begin{array}{cc} 0 & [I] \\ -[M]^{-1}[K] & 0 \end{array} \right] \left\{ \begin{array}{c} \mathbf{r}_1 \\ \mathbf{r}_2 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ [M]^{-1}\{F\} \end{array} \right\} \tag{2.33}$$

or in general form as

$$\dot{\mathbf{r}} = [\Psi]\mathbf{r} + \{\Phi\} \tag{2.34}$$

The matrix $[\Psi]$ is a constant and can be precomputed and stored for use during the time-integration process. The time derivative term $\dot{\mathbf{r}}$ is discretized using the second-order accurate BDF2 scheme as in the case of the time derivative term in the flow equations. It should be noted that the time-step $\Delta\tau$ appearing in the denominator of the discretized version of the structural equations is different from the time-step of the flow equations, and their relation is $\Delta\tau = 2k_c\Delta t$, where Δt is the non-dimensional time-step used for the flow equations. This is due to the non-dimensionalization of time in the structural equations by the uncoupled natural frequency in pitch ω_α. Once the vector $\mathbf{r}^n$ at a time-level n has been solved for, the displacement vector $\{q\}$ is known and the configuration of the newly deformed mesh (i.e. $\mathbf{x}^n$) can be computed using the mesh deformation equations (2.11) where $\mathbf{x}_{surf}$ is now a function of $\mathbf{r}$.

2.7 Solution of the Analysis Problem and General Implementation Details

The analysis solver consists of three specific parts namely, the steady-state flow solver, the unsteady flow solver, and the coupled unsteady aeroelastic solver. The solver requires the baseline computational mesh and the flow parameters such as the freestream Mach number and the angle-of-attack of the airfoil in order to begin the solution process. The solution sequence begins by obtaining a steady-state solution of the Euler equations for the given flow parameters and then utilizes the steady-state solution as the initial condition for the unsteady solution process. Aeroelastic solutions typically require a period of forced unsteady motion of the airfoil before the free aeroelastic response to loads is allowed to evolve. The prescribed unsteady motion in the second part of the solver provides the necessary initial condition for the third component which is the aeroelastic solver. The flow and structural equations are coupled through the mesh deformation equations and the aeroelastic solver obtains the solution using a fully coupled iterative strategy.

2.7.1 Implicit Flow Solution Procedure

The flow equations are solved implicitly by defining an implicit flow residual as

$$\mathbf{R}(\mathbf{U},\mathbf{x}) = \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{2.35}$$

for the steady-state problem and as

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = \frac{dA\mathbf{U}}{dt} + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{2.36}$$

for the unsteady problem based on the BDF2 time-integration scheme at an arbitrary implicit time-step n. The dependence of the residual on flow solutions at time indices $n-1$ and $n-2$ appear due to the time derivative term discretized using the BDF2 scheme and the dependence on the mesh coordinates at previous time indices appears through the GCL compliant mesh edge velocities and the element volume terms in the time derivative. The implicit residual is then linearized with respect to the unknown solution $\mathbf{U}$ in order to construct a Newton scheme as

$$\left[\frac{\partial \mathbf{R}(\mathbf{U}^j)}{\partial \mathbf{U}^j}\right] \delta\mathbf{U}^j = -\mathbf{R}(\mathbf{U}^j) \tag{2.37}$$

$$\mathbf{U}^{j+1} = \mathbf{U}^j + \delta\mathbf{U}^j \tag{2.38}$$

where once the update $\delta\mathbf{U} \rightarrow 0$, the unknown steady-state solution $\mathbf{U}$ or unknown unsteady solution $\mathbf{U}^n$ is available and satisfies the corresponding implicit residual equation. The intermediate linear systems in the Newton scheme are solved using either Gauss-Seidel iterations or the linear agglomeration multigrid [1] approach.

2.7.2 Mesh Deformation Solution Procedure

For both steady and unsteady problems the displacement of the geometry of interest is a prescribed quantity. In the case of steady-state problems, the angle-of-attack is given, while in unsteady problems the angle-of-attack is a time-dependent quantity. In either case, surface mesh coordinates defining the airfoil or geometry of interest have to be rotated and/or translated from the baseline computational mesh configuration. These prescribed displacements are used to construct the right-hand-side of the mesh motion equations shown in equation (2.11). The system is linear and elliptic in nature and is solved using Gauss-Seidel iterations or by an agglomeration multigrid [1] technique employing Gauss-Seidel iterations as an error smoother. In the case of aeroelasticity, the displacement of the geometry is provided by the structural response to aerodynamic loads, which can then be used to drive the interior mesh deformation.

2.7.3 Coupled Aeroelastic Solution Procedure

The governing equations in the case of aeroelasticity are fully coupled. The flow equations predict the loads acting on the geometry, while the structural equations deform or displace the geometry based on the loads. The mesh motion equations are used to deform the mesh based on the displacements predicted by the structural equations. The deformed mesh however changes the flow solution thus resulting in changes to the loads. The equations have to be solved in a coupled iterative manner, where each discipline is solved partially while the source terms coupled with the remaining disciplines are constantly updated. As in the case of the flow equations, the structural equations are solved implicitly by defining an implicit structural residual at any time-step n as

$$\mathbf{J}^n(\mathbf{r}^n, \mathbf{r}^{n-1}, \mathbf{r}^{n-2}, \mathbf{U}^n, \mathbf{x}^n) = \frac{\partial \mathbf{r}}{\partial \tau} - [\Psi]\,\mathbf{r}^n - \{\Phi\} = 0 \tag{2.39}$$

which when linearized with respect to the structural state $\mathbf{r}^n$ enables the construction of a Newton scheme as

$$\left[\frac{\partial \mathbf{J}(\mathbf{r}^j)}{\partial \mathbf{r}^j}\right] \delta \mathbf{r}^j = -\mathbf{J}(\mathbf{r}^j) \tag{2.40}$$

Although derived from a Newton scheme, the above equation can be solved linearly since the Jacobian matrix is a constant. The coupling between the three disciplines, i.e. flow, structure and mesh becomes apparent when summarizing the respective residual equations as shown below.

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0 \tag{2.41}$$

$$\mathbf{J}^n(\mathbf{r}^n, \mathbf{r}^{n-1}, \mathbf{r}^{n-2}, \mathbf{U}^n, \mathbf{x}^n) = 0 \tag{2.42}$$

$$\mathbf{G}^n(\mathbf{x}^n, \mathbf{x}_{surf}(\mathbf{r}^n, \mathbf{x}^d_{surf})) = 0 \tag{2.43}$$

Here $\mathbf{x}^d_{surf}$ refers to the surface coordinates of the baseline airfoil deformed using the shape functions. The flow equations depend on the mesh coordinates, but the mesh coordinates depend on the structural state, which in turn depends on both the flow solution and the mesh coordinates. Figure 2.6 shows the coupled iterative strategy employed in the aeroelastic solver to advance from time-step n to time-step $n+1$. Only partial solutions of each discipline are required during the coupled iterations. Although converging each discipline completely within the coupled iterative strategy

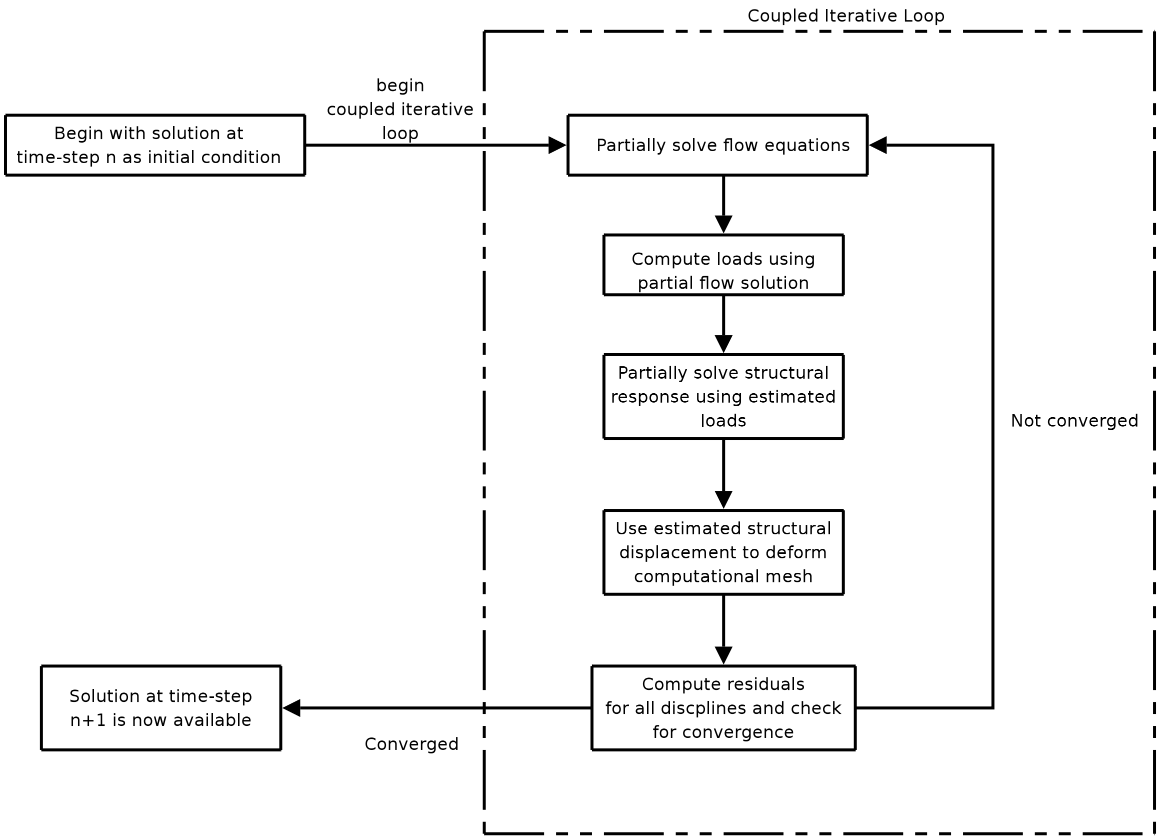

Coupled Iterative Loop
begin coupled iterative loop
Begin with solution at time-step n as initial condition
Partially solve flow equations
Compute loads using partial flow solution
Partially solve structural response using estimated loads
Use estimated structural displacement to deform computational mesh
Compute residuals for all discplines and check for convergence
Not converged
Converged
Solution at time-step n+1 is now available

Figure 2.6: Coupled iterative loop for advancing from time-step n to time-step $n+1$ in the aeroelastic solver.

does not change the final answer, this is unnecessary and imposes severe additional cost. Figure 2.7 compares the typical convergence of the flow, mesh and structural equations when employing the coupled iterative strategy.

2.8 Damped Newton Solver

In many cases, the Jacobian matrix used to drive the nonlinear Newton method for the flow equations is ill conditioned causing difficulties in the solution of the linear system. For this reason, damping is added to the Jacobian matrix by augmenting the diagonal terms with a damping factor as

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]_{damped} = \frac{A}{k_{damp}}[\mathbf{I}] + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right] \tag{2.44}$$

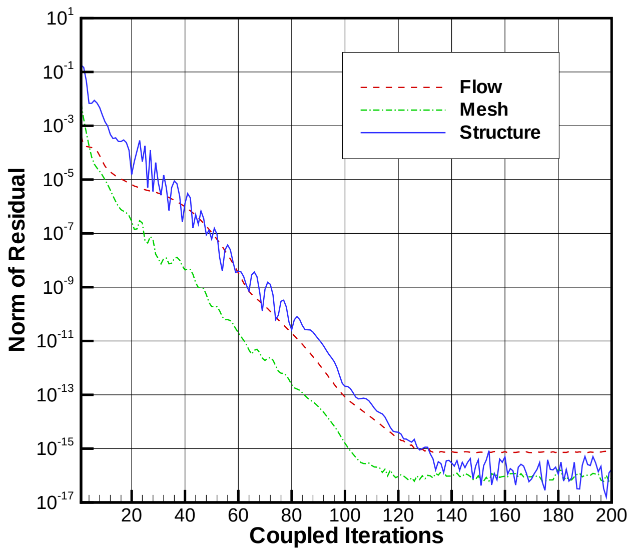


Figure 2.7: Typical convergence of the coupled aeroelastic equations. For this particular example, each coupled iteration consists of 1 Newton iteration for the flow, 200 Gauss-Jacobi iterations for the mesh, and 3 Gauss-Seidel iterations for the structure. A single Newton iteration for the flow consists of 2 multigrid cycles over 4 levels with 10 Gauss-Jacobi smoothing cycles at each multigrid level.

where A refers to the local element area and k_{damp} refers to an empirically determined damping parameter. Smaller values of k_{damp} add more damping to the Jacobian matrix, slowing down the overall nonlinear convergence but increases robustness in the linear solutions. As the damping parameter goes to infinity, no damping is added to the Jacobian matrix and the nonlinear solver recovers the exact Newton method. Typically the solution process is started with a small empirically determined value for the damping parameter and is progressively increased at every nonlinear update to the solution.

2.9 Overall Solution Procedure

The overall solution procedure involves starting with a steady-state solution, followed by an unsteady solution with prescribed motion of the geometry, and then initiating a coupled aeroelastic solution with the prescribed unsteady solution. The process of obtaining a solution to the described set of governing equations can be broken down as follows:

1. Read in baseline computational mesh of airfoil and freestream flow parameters.
2. Deform surface coordinates defining airfoil to change its shape as per current design variables and construct right-hand-side of mesh motion equations.
3. Solve linear mesh motion equations to obtain distribution of interior mesh coordinates.
4. Rotate surface coordinates $\mathbf{x}^d_{surf}$ to the prescribed steady-state angle of attack.
5. Solve mesh motion equations to obtain interior mesh coordinate distribution.
6. Initialize all elements in the spatial mesh with freestream conditions.
7. Obtain steady-state solution using Newton iterations operating on the steady-state residual $\mathbf{R}(\mathbf{U}, \mathbf{x}) = 0$.

8. If only a steady-state solution is required, stop at this point and compute functionals based on $\mathbf{U}$ and $\mathbf{x}$.

9. Begin time-integration process by using steady-state solution as the initial condition.

10. Rotate surface coordinates of airfoil $\mathbf{x}^d_{surf}$ to the correct angle-of-attack for the current time-step n to determine $\mathbf{x}^n_{surf}$.

11. Solve mesh motion equations to obtain interior mesh coordinate distribution $\mathbf{x}^n$ for time-step n.

12. Compute GCL compliant edge velocities for current time-step.

13. Obtain flow solution $\mathbf{U}^n$ for time-step n using Newton iterations operating on the residual $\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0$.

14. Keep integrating forward in time until solution at all time-steps representing the time domain of interest has been obtained.

15. If only an unsteady solution with prescribed motion of the airfoil is required, stop at this point and compute functionals based on the unsteady flow solution set $(\mathbf{U}^1, \mathbf{U}^2, \cdots, \mathbf{U}^n)$ and the unsteady set of mesh coordinates $(\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n)$.

16. Begin coupled unsteady aeroelastic solution procedure using the unsteady solution of the prescribed motion as the initial condition.

17. Partially solve flow equations using Newton iterations and obtain an estimate for the loads on the airfoil.

18. Partially solve structural equations using Newton iterations and estimate a displacement vector for the airfoil.

19. Construct right-hand-side of mesh motion equations using the estimated displacement vector and partially solve mesh motion equations.

20. Return to flow equations and continue solving with updated computational mesh.

21. Keep iterating until residuals for all disciplines have converged, indicating that the solution set satisfying all disciplines has been obtained. The solution has advanced one time-step at this point.

22. Continue integration in time until aeroelastic solution set spanning all time-steps representing the time domain of interest has been obtained.

23. Compute functionals based on aeroelastic solution set $(\mathbf{U}, \mathbf{r}, \mathbf{x})$, spanning the spatial and temporal domains.

Chapter 3

Linearization for Functional Sensitivity Computation

3.1 Generalized Sensitivity Analysis

Consider a nonlinear function F operating on set of arbitrary independent variables $\mathbf{D}$ resulting in a scalar functional L. In the context of computational fluid dynamics, for a steady-state inviscid flow problem, the nonlinear operator F represents the Euler equations and the set of independent variables $\mathbf{D}$ represents design input parameters that control the shape of the geometry under consideration. The output functional for such problems are typically surface integrated quantities such as the lift or drag. If the shape of the geometry is to be modified in order to achieve some target value of the functional, the vector of sensitivity derivatives of the functional with respect to the set of design inputs is required. The sensitivity derivatives can be obtained via finite-differencing where each of the design input variables in the vector $\mathbf{D}$ is perturbed by some appropriate value $\Delta\mathbf{D}$ and the nonlinear function F is evaluated in order to determine the corresponding change in the functional ΔL. The sensitivity of the functional to the design input $\mathbf{D}$ can then be approximated as $\frac{\Delta L}{\Delta \mathbf{D}}$. In this procedure the number of nonlinear function evaluations is directly proportional to the number of design inputs in the vector $\mathbf{D}$. Additionally, the accuracy of the estimated sensitivity derivative is dependent on the size of the perturbation ΔD. From theory it is known that the analytically exact derivative is obtained as $\Delta \mathbf{D} \rightarrow 0$. Discrete computational operations however require a finite value for the perturbation and choosing the appropriate value poses certain challenges. Too small of a perturbation can result in inaccuracies arising from machine precision limitations, while larger values could potentially estimate the wrong slope if the function is highly nonlinear.

The other approach to estimate the sensitivity derivative is to directly differentiate the analytic nonlinear function F (Euler equations) with respect to the design inputs $\mathbf{D}$ in order to obtain an analytic form for the sensitivity derivative. The resulting analytic or continuous form of the sensitivity derivative may be discretized and solved for numerically. This is the continuous linearization method and results in a discrete approximation to the exact analytic sensitivity derivatives of interest. Alternatively, each discrete operation used to evaluate $F(\mathbf{D})$ numerically may be sequentially differentiated with respect to the design inputs $\mathbf{D}$. In this work, this involves the direct differentiation of the code used to solve the discretized Euler equations which takes in $\mathbf{D}$ as the input and computes L as the functional output. This represents the discrete linearization method, which results in an exact linearization of the discrete equations and forms the core of the presented work.

While the computation of the functional L may be expressed in general form as $L = F(\mathbf{D})$, the nonlinear function F may involve several sub functions sequentially operating on one another. Also, it may be possible that the nonlinear operator F produces multiple functionals $\mathbf{L}$. A good example in the context of this work is the computation of multiple load coefficients such lift, drag and moment coefficients using the solution of the Euler equations. Expressing the overall evaluation procedure with all intermediate dependencies, the functionals $\mathbf{L}$ may be expressed as

$$\mathbf{L} = F_i(F_{i-1}(F_{i-2}(\cdots F_2(F_1(F_0(\mathbf{D})))))) \tag{3.1}$$

Note that the bold font used for $\mathbf{L}$ now indicates a vector of functionals and not a scalar. The sensitivity of the functionals $\mathbf{L}$ to the set of design inputs $\mathbf{D}$ now forms a matrix and can be determined by differentiating using the chain rule as:

$$\frac{d\mathbf{L}}{d\mathbf{D}} = \frac{\partial \mathbf{L}}{\partial F_i}\frac{\partial F_i}{\partial F_{i-1}}\frac{\partial F_{i-1}}{\partial F_{i-2}}\cdots\frac{\partial F_2}{\partial F_1}\frac{\partial F_1}{\partial F_0}\frac{\partial F_0}{\partial \mathbf{D}} \tag{3.2}$$

Assuming that the derivative of a function with respect to another function forms a matrix, for multiple design inputs $n_{\mathbf{D}}$ the product at the right extreme of equation (3.2) represents a matrix-matrix product operation. As a general rule in computational work, matrix-matrix products are to be avoided since they are $\mathcal{O}(n^2)$ operations. However matrix-vector products are considered acceptable since they result in vectors and require $\mathcal{O}(n)$ operations. For the case of a single design input ($n_{\mathbf{D}} = 1$) and single or multiple functional outputs $\mathbf{L}$, the total sensitivity $\frac{d\mathbf{L}}{d\mathbf{D}}$ can be evaluated by a series of efficient matrix-vector or vector-vector products. This is a direct consequence of the rightmost term in equation (3.2) being a vector for the case of a single design input. Equation (3.2) represents the forward or tangent linearization of the functional(s) $\mathbf{L}$ with respect to the design input(s) $\mathbf{D}$. Practical design problems in aerodynamics however typically involve multiple design inputs and a single functional of interest. The forward linearization becomes inefficient for such cases since for multiple inputs this results in matrix-matrix product operations. This problem can be elegantly circumvented by simply transposing equation (3.2) to obtain the reverse or adjoint linearization of the functional with respect to the design inputs. The transpose of the forward linearization reverses the multiplication order and can be expressed as

$$\frac{d\mathbf{L}}{d\mathbf{D}}^T = \frac{\partial F_0}{\partial \mathbf{D}}^T\frac{\partial F_1}{\partial F_0}^T\frac{\partial F_2}{\partial F_1}^T\cdots\frac{\partial F_{i-1}}{\partial F_{i-2}}^T\frac{\partial F_i}{\partial F_{i-1}}^T\frac{\partial \mathbf{L}}{\partial F_i}^T \tag{3.3}$$

For a single functional of interest, the last term in equation (3.3) is now a vector, and the resulting sequence of product operations again consists of either matrix-vector or vector-vector products. To summarize, it is efficient to invoke the forward linearization for cases with multiple outputs **L** and a single or few inputs **D**, while the adjoint or reverse linearization is more economical for cases with a single or few outputs **L** and multiple inputs **D**. Although the forward linearization of the governing equations is presented in the current work, it is only for the purpose of clarifying the derivation of the corresponding adjoint linearization. The adjoint linearization is used exclusively for applications presented in this work.

3.2 Generalized Gradient Formulation for Multidisciplinary Coupled Unsteady Equations

3.2.1 Forward Linearization

Consider an arbitrary number of m coupled unsteady disciplines, where the solution of each discipline is given by the state variable $\mathcal{U}$. The state variables for each of the disciplines span both the spatial and temporal domains since it is assumed that the governing equations of every discipline are unsteady in nature. For the general case, a scalar functional L evaluated using the multidisciplinary solution set can be expressed as

$$L = L(\mathbf{D},\mathcal{U}_1,\mathcal{U}_2,\cdots,\mathcal{U}_m) \tag{3.4}$$

where in addition to an explicit dependence on the design inputs **D**, there exists an implicit dependence through the state variables $\mathcal{U}$ of each discipline. Differentiating with respect to the design inputs using the chain rule, the total sensitivity or gradient vector $\frac{dL}{d\mathbf{D}}$ can be expressed as

$$\frac{dL}{d\mathbf{D}} = \frac{\partial L}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathcal{U}_1}\frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathcal{U}_2}\frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} + \cdots + \frac{\partial L}{\partial \mathcal{U}_m}\frac{\partial \mathcal{U}_m}{\partial \mathbf{D}} \tag{3.5}$$

which can also be expressed in terms of the inner product between the vector of functional sensitivities to state variables and the vector of state variable sensitivities to the design inputs as

$$\frac{dL}{d\mathbf{D}} = \frac{\partial L}{\partial \mathbf{D}} + \begin{bmatrix} \frac{\partial L}{\partial \mathcal{U}_1} & \frac{\partial L}{\partial \mathcal{U}_2} & \cdots & \frac{\partial L}{\partial \mathcal{U}_m} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} \\ \frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} \\ \vdots \\ \frac{\partial \mathcal{U}_m}{\partial \mathbf{D}} \end{bmatrix} \tag{3.6}$$

Now assume that the spatial domain is represented by n_{cells} discrete elements while the temporal domain consists of n_{steps} discrete time-steps. The linearization of the functional with respect to the disciplinary state variables (i.e $\frac{\partial L}{\partial \mathcal{U}_i}$) then forms vectors of dimension $[1 \times n_{steps}]$, where each element in the vector is a vector by itself of dimension $[1 \times n_{cells}]$. These vectors are easily computable since the functional L is a scalar quantity. The linearization of the disciplinary state variables with respect to the design inputs (i.e. $\frac{\partial \mathcal{U}_i}{\partial \mathbf{D}}$) form matrices of dimension $[n_{ncells} \times n_{\mathbf{D}}]$ at each time-step for the case of $n_{\mathbf{D}}$ design inputs. For each discipline, there are n_{steps} number of such matrices. The state variable sensitivity matrices are unknown quantities at this point and an expression for their evaluation must be determined. The governing nonlinear equations of each discipline may be cast in residual form as

$$\begin{aligned} \mathcal{R}_1(\mathbf{D}, \mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m) &= 0 \\ \mathcal{R}_2(\mathbf{D}, \mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m) &= 0 \\ &\vdots \\ \mathcal{R}_m(\mathbf{D}, \mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m) &= 0 \end{aligned} \tag{3.7}$$

where it is important to note that the residual equation of each discipline is dependent not only on its own state but also on the states of the remaining disciplines due to the assumption of coupling between disciplines. For the general case, the residual equations also have explicit dependence on the design inputs $\mathbf{D}$ and an implicit dependence through the state variables $\mathcal{U}$. For a fully converged solution set $\mathcal{U}$, the residual equations have to evaluate identically to zero at all spatial and temporal points in the domain within which the solution was obtained. For this reason the derivatives of the residual for a fully converged solution set also have to evaluate to zero at all spatial and temporal points within the domain. Differentiating the disciplinary residual equations with respect to the set of design inputs $\mathbf{D}$ then yields:

$$\begin{aligned}
\frac{\partial \mathcal{R}_1}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}\frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}\frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} + \cdots + \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}\frac{\partial \mathcal{U}_m}{\partial \mathbf{D}} &= 0 \\
\frac{\partial \mathcal{R}_2}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}\frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}\frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} + \cdots + \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}\frac{\partial \mathcal{U}_m}{\partial \mathbf{D}} &= 0 \\
&\vdots \\
\frac{\partial \mathcal{R}_m}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}\frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} + \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}\frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} + \cdots + \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}\frac{\partial \mathcal{U}_m}{\partial \mathbf{D}} &= 0
\end{aligned} \tag{3.8}$$

which can further be combined into matrix form by extracting the vector of unknown state variable sensitivity matrices as:

$$\begin{bmatrix}
\frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m} \\
\frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m} \\
\vdots & & & \\
\frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial \mathcal{U}_1}{\partial \mathbf{D}} \\
\frac{\partial \mathcal{U}_2}{\partial \mathbf{D}} \\
\vdots \\
\frac{\partial \mathcal{U}_m}{\partial \mathbf{D}}
\end{bmatrix}
=
\begin{bmatrix}
-\frac{\partial \mathcal{R}_1}{\partial \mathbf{D}} \\
-\frac{\partial \mathcal{R}_2}{\partial \mathbf{D}} \\
\vdots \\
-\frac{\partial \mathcal{R}_m}{\partial \mathbf{D}}
\end{bmatrix} \tag{3.9}$$

The Jacobian matrices $\frac{\partial \mathcal{R}_i}{\partial \mathcal{U}_j}(i = 1, m : j = 1, m)$ span both spatial and temporal domains, however time being purely hyperbolic in nature implies that the discrete residual $\mathcal{R}^n$ at an arbitrary time-step n can only depend on quantities at time indices $\leq n$. Consequently, for the general case, any Jacobian matrix when discretely expanded in time for a temporal domain consisting of n time-steps is block lower triangular as shown below, where each of the blocks is a matrix spanning the spatial domain.

$$\frac{\partial \mathcal{R}}{\partial \mathcal{U}} = \begin{bmatrix} \ddots & 0 & 0 & 0 \\ \cdots & \dfrac{\partial \mathcal{R}^{n-2}}{\partial \mathcal{U}^{n-2}} & 0 & 0 \\ \cdots & \dfrac{\partial \mathcal{R}^{n-1}}{\partial \mathcal{U}^{n-2}} & \dfrac{\partial \mathcal{R}^{n-1}}{\partial \mathcal{U}^{n-1}} & 0 \\ \cdots & \dfrac{\partial \mathcal{R}^{n}}{\partial \mathcal{U}^{n-2}} & \dfrac{\partial \mathcal{R}^{n}}{\partial \mathcal{U}^{n-1}} & \dfrac{\partial \mathcal{R}^{n}}{\partial \mathcal{U}^{n}} \end{bmatrix} \tag{3.10}$$

As an example, for two coupled disciplines ($m = 1, 2$) and a temporal domain consisting of two time-steps n and $n-1$, equation (3.9) can be rewritten using the discrete temporally expanded form of the Jacobian matrices as

$$\left[\begin{array}{cc|cc} \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_1^{n-1}} & 0 & \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_2^{n-1}} & 0 \\ \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n-1}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n}} \\ \hline \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_1^{n-1}} & 0 & \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_2^{n-1}} & 0 \\ \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n-1}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n}} \end{array}\right] \left[\begin{array}{c} \dfrac{\partial \mathcal{U}_1^{n-1}}{\partial \mathbf{D}} \\ \dfrac{\partial \mathcal{U}_1^{n}}{\partial \mathbf{D}} \\ \hline \dfrac{\partial \mathcal{U}_2^{n-1}}{\partial \mathbf{D}} \\ \dfrac{\partial \mathcal{U}_2^{n}}{\partial \mathbf{D}} \end{array}\right] = \left[\begin{array}{c} -\dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathbf{D}} \\ -\dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathbf{D}} \\ \hline -\dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathbf{D}} \\ -\dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathbf{D}} \end{array}\right] \tag{3.11}$$

Swapping rows and columns, the overall system may be rearranged as a block lower triangular matrix of the form shown below, where each diagonal block represents a single time-step.

$$\left[\begin{array}{cc|cc} \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_2^{n-1}} & 0 & 0 \\ \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_2^{n-1}} & 0 & 0 \\ \hline \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n-1}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n}} & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n}} \\ \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n-1}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n-1}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n}} & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n}} \end{array}\right] \left[\begin{array}{c} \dfrac{\partial \mathcal{U}_1^{n-1}}{\partial \mathbf{D}} \\ \dfrac{\partial \mathcal{U}_2^{n-1}}{\partial \mathbf{D}} \\ \hline \dfrac{\partial \mathcal{U}_1^{n}}{\partial \mathbf{D}} \\ \dfrac{\partial \mathcal{U}_2^{n}}{\partial \mathbf{D}} \end{array}\right] = \left[\begin{array}{c} -\dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathbf{D}} \\ -\dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathbf{D}} \\ \hline -\dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathbf{D}} \\ -\dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathbf{D}} \end{array}\right] \tag{3.12}$$

For the limiting case of a single design input $\mathbf{D}$, this forms a block lower triangular linear system where the unknown state variable sensitivities and the right-hand-side form vectors rather than matrices. The system can be solved using forward substitution, i.e. a forward sweep in time beginning at time-step $n-1$ and progressing to time-step n. At each time-step, a fully coupled linear system is solved iteratively to obtain the sensitivity of the state variables at that time-step to the single design input. For the case of multiple design inputs $\mathbf{D}$, the forward sweep in time along with the coupled linear solution at each time-step has to be performed for each input in order to construct the state variable sensitivity matrix for each discipline one column at a time. Once the state variable sensitivity matrices are available, they can substituted into equation (3.6) where the inner product can be evaluated and the complete gradient vector $\frac{dL}{d\mathbf{D}}$ becomes available.

3.2.2 Adjoint Linearization

In the case of the forward linearization, the number of solutions of equation (3.12) that are required is directly dependent on the number of design inputs $\mathbf{D}$. This issue can be circumvented by switching to the adjoint or reverse linearization of the same problem by transposing equation (3.6). The transpose of the forward linearization can be expressed as

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial L}{\partial \mathbf{D}}^T + \begin{bmatrix} \frac{\partial \mathcal{U}_1}{\partial \mathbf{D}}^T & \frac{\partial \mathcal{U}_2}{\partial \mathbf{D}}^T & \cdots & \frac{\partial \mathcal{U}_m}{\partial \mathbf{D}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \mathcal{U}_1}^T \\ \frac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \frac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix} \tag{3.13}$$

where the goal now is to avoid explicit computation of the unknown state variable sensitivity matrices. A convenient expression for the vector of state variable sensitivity matrices can be obtained by transposing and rearranging equation (3.9) as

$$\begin{bmatrix} \dfrac{\partial \mathcal{U}_1}{\partial \mathbf{D}}^T & \dfrac{\partial \mathcal{U}_2}{\partial \mathbf{D}}^T & \cdots & \dfrac{\partial \mathcal{U}_m}{\partial \mathbf{D}}^T \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathbf{D}}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathbf{D}}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathbf{D}}^T \end{bmatrix} \begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}^{-1} \tag{3.14}$$

Substituting into the transposed total sensitivity equation (3.13) yields,

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial L}{\partial \mathbf{D}}^T + \begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathbf{D}}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathbf{D}}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathbf{D}}^T \end{bmatrix} \begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}^{-1} \begin{bmatrix} \dfrac{\partial L}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \dfrac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix} \tag{3.15}$$

Now defining a vector of adjoint variables per discipline spanning the spatial and temporal domains as

$$\begin{bmatrix} \Lambda_{\mathcal{U}_1} \\ \Lambda_{\mathcal{U}_2} \\ \vdots \\ \Lambda_{\mathcal{U}_m} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}^{-1} \begin{bmatrix} \dfrac{\partial L}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \dfrac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix} \tag{3.16}$$

a block coupled linear adjoint system of equations can be recovered as

$$\begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix} \begin{bmatrix} \Lambda_{\mathcal{U}_1} \\ \Lambda_{\mathcal{U}_2} \\ \vdots \\ \Lambda_{\mathcal{U}_m} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial L}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \dfrac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix} \tag{3.17}$$

Following the same example presented in the derivation of the forward linearization, for a problem with two disciplines ($m = 2$) and two time-steps n and $n-1$ representing the temporal domain, the adjoint system can be discretely expanded in time and expressed as

$$\left[\begin{array}{cc|cc} \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_1^{n-1}}^T & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n-1}}^T & \dfrac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_2^{n-1}}^T & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n-1}}^T \\ 0 & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n}}^T & 0 & \dfrac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n}}^T \\ \hline \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_1^{n-1}}^T & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n-1}}^T & \dfrac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_2^{n-1}}^T & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n-1}}^T \\ 0 & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n}}^T & 0 & \dfrac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n}}^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathcal{U}_1}^{n-1} \\ \Lambda_{\mathcal{U}_1}^{n} \\ \hline \Lambda_{\mathcal{U}_2}^{n-1} \\ \Lambda_{\mathcal{U}_2}^{n} \end{array}\right] = \left[\begin{array}{c} \dfrac{\partial L}{\partial \mathcal{U}_1^{n-1}}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_1^{n}}^T \\ \hline \dfrac{\partial L}{\partial \mathcal{U}_2^{n-1}}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_2^{n}}^T \end{array}\right] \tag{3.18}$$

which can be rearranged into block upper triangular form as shown below by swapping rows and columns.

$$\left[\begin{array}{cc|cc} \frac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_1^{n-1}}^T & \frac{\partial \mathcal{R}_1^{n-1}}{\partial \mathcal{U}_2^{n-1}}^T & \frac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n-1}}^T & \frac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n-1}}^T \\ \frac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_1^{n-1}}^T & \frac{\partial \mathcal{R}_2^{n-1}}{\partial \mathcal{U}_2^{n-1}}^T & \frac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n-1}}^T & \frac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n-1}}^T \\ \hline 0 & 0 & \frac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_1^{n}}^T & \frac{\partial \mathcal{R}_1^{n}}{\partial \mathcal{U}_2^{n}}^T \\ 0 & 0 & \frac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_1^{n}}^T & \frac{\partial \mathcal{R}_2^{n}}{\partial \mathcal{U}_2^{n}}^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathcal{U}_1}^{n-1} \\ \Lambda_{\mathcal{U}_2}^{n-1} \\ \hline \Lambda_{\mathcal{U}_1}^{n} \\ \Lambda_{\mathcal{U}_2}^{n} \end{array}\right] = \left[\begin{array}{c} \frac{\partial L}{\partial \mathcal{U}_1^{n-1}}^T \\ \frac{\partial L}{\partial \mathcal{U}_2^{n-1}}^T \\ \hline \frac{\partial L}{\partial \mathcal{U}_1^{n}}^T \\ \frac{\partial L}{\partial \mathcal{U}_2^{n}}^T \end{array}\right] \tag{3.19}$$

The system can be solved through back substitution, i.e. a backward sweep in time, where a coupled linear system is solved at time-step n before progressing backward to time-step $n-1$. At each time-step a coupled iterative linear solution is required in order to determine the vector of unknown disciplinary adjoint variables at that time-step. Once the vector of disciplinary adjoint variables spanning the spatial and temporal domains is available, it may be substituted into the total sensitivity equation as

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial L}{\partial \mathbf{D}}^T + \left[\begin{array}{cccc} \frac{\partial \mathcal{R}_1}{\partial \mathbf{D}}^T & \frac{\partial \mathcal{R}_2}{\partial \mathbf{D}}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathbf{D}}^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathcal{U}_1} \\ \Lambda_{\mathcal{U}_2} \\ \cdots \\ \Lambda_{\mathcal{U}_m} \end{array}\right] \tag{3.20}$$

where the effect of the number of design inputs $\mathbf{D}$ has been confined to a series of matrix-vector products at the end of the computation. The elegance of the method lies in the fact that only a single backward sweep in time with coupled linear solutions at each time step is required in order to compute the total gradient vector, contrary to the forward linearization. The determination of the gradient vector therefore involves a single forward integration in time to obtain the solution to the coupled unsteady analysis problem, and a single backward sweep in time to obtain the necessary adjoint variables.

3.3 Gradients for the Unsteady Aeroelasticity Problem

It is now possible to apply the general formulation specifically to the unsteady aeroelasticity analysis problem. The three state variables for this case are the flow $\mathbf{U}$, mesh $\mathbf{x}$ and structure $\mathbf{r}$. All three disciplines are coupled and additionally unsteady in nature, implying that the state variables for each discipline span the spatial and temporal domains. Following the derivation presented in the previous section, a general time-integrated functional L evaluated using the unsteady aeroelastic solution set can be defined as

$$L = L\left(\mathbf{U}, \mathbf{r}, \mathbf{x}\right) \tag{3.21}$$

where there is only an implicit dependence on the design inputs through the state variables and no explicit dependence. Linearizing the functional with respect to the set of design inputs $\mathbf{D}$ and transposing the result, the total sensitivity vector $\frac{dL}{d\mathbf{D}}$ can be written as the inner product of the vector of state sensitivities to design inputs and the vector of functional sensitivities to state variables as

$$\begin{aligned} \frac{dL}{d\mathbf{D}}^T &= \frac{\partial \mathbf{U}}{\partial \mathbf{D}}^T \frac{\partial L}{\partial \mathbf{U}}^T + \frac{\partial \mathbf{r}}{\partial \mathbf{D}}^T \frac{\partial L}{\partial \mathbf{r}}^T + \frac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \frac{\partial L}{\partial \mathbf{x}}^T \\ &= \begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{r}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}}^T \\ \frac{\partial L}{\partial \mathbf{r}}^T \\ \frac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \end{aligned} \tag{3.22}$$

The three disciplinary residual equations with state variables for each spanning both the spatial and temporal domains can be summarized as

$$\mathbf{R}\left(\mathbf{U}, \mathbf{x}\right) = 0 \tag{3.23}$$

$$\mathbf{J}\left(\mathbf{r}, \mathbf{U}, \mathbf{x}\right) = 0 \tag{3.24}$$

$$\mathbf{G}\left(\mathbf{x}, \mathbf{x}_{surf}\left(\mathbf{r}, \mathbf{x}^d_{surf}\right)\right) = 0 \tag{3.25}$$

where **R**, **J** and **G** refer to the flow, structure and mesh residual equations respectively. These may now be linearized with respect to the design inputs using the chain rule yielding

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{D}} = 0 \tag{3.26}$$

$$\frac{\partial \mathbf{J}}{\partial \mathbf{r}}\frac{\partial \mathbf{r}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{D}} = 0 \tag{3.27}$$

$$\frac{\partial \mathbf{G}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{D}} + \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}\left(\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{r}}\frac{\partial \mathbf{r}}{\partial \mathbf{D}} + \frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{x}^d_{surf}}\frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}\right) = 0 \tag{3.28}$$

Now combining the linearized residual equations into matrix form as

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}} & 0 & \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{J}}{\partial \mathbf{U}} & \frac{\partial \mathbf{J}}{\partial \mathbf{r}} & \frac{\partial \mathbf{J}}{\partial \mathbf{x}} \\ 0 & \left(\frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{r}}\right) & \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{r}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}}{\partial \mathbf{D}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\left(\frac{\partial \mathbf{G}}{\partial \mathbf{x}^d_{surf}}\frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}\right) \end{bmatrix} \tag{3.29}$$

and transposing, yields an expression for the vector of state variable sensitivity matrices as:

$$\begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{r}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & -\left(\frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}^d_{surf}}^T\right) \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{U}}^T & 0 \\ 0 & \frac{\partial \mathbf{J}}{\partial \mathbf{r}}^T & \left(\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{r}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T\right) \\ \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \end{bmatrix}^{-1} \tag{3.30}$$

Substituting the above expression into equation (3.22), the total sensitivity equation becomes

$$\frac{dL}{d\mathbf{D}}^T =$$

$$\begin{bmatrix} 0 & 0 & -\left(\frac{\partial \mathbf{x}_{surf}^d}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}^d}^T\right) \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{U}}^T & 0 \\ 0 & \frac{\partial \mathbf{J}}{\partial \mathbf{r}}^T & \left(\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{r}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T\right) \\ \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}}^T \\ \frac{\partial L}{\partial \mathbf{r}}^T \\ \frac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \tag{3.31}$$

Now introducing a vector of adjoint variables per discipline, the recovered coupled adjoint system can be expressed as

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{U}}^T & 0 \\ 0 & \frac{\partial \mathbf{J}}{\partial \mathbf{r}}^T & \left(\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{r}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T\right) \\ \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{J}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{G}}{\partial \mathbf{x}}^T \end{bmatrix} \begin{bmatrix} \Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{r}} \\ \Lambda_{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}}^T \\ \frac{\partial L}{\partial \mathbf{r}}^T \\ \frac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \tag{3.32}$$

where $\Lambda_{\mathbf{U}}$, $\Lambda_{\mathbf{x}}$ and $\Lambda_{\mathbf{r}}$ are the flow, mesh and structural adjoint variables spanning both the spatial and temporal domains. The next step is to determine the forms of the various Jacobian matrices appearing in the adjoint system. The implicit flow residual equation based on the BDF2 temporal discretization at an arbitrary time-step n is given as

$$\mathbf{R}^n\left(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}\right) = 0 \tag{3.33}$$

indicating that the linearization of the flow residual with respect to the flow variables $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$, is a block tridiagonal matrix in time of the form

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right] = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 \\ \ddots & \ddots & 0 & 0 & 0 \\ \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-4}} & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-3}} & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-3}} & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}} & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}} & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}} & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}} \end{bmatrix} \tag{3.34}$$

Similarly, the linearization of the flow residual $\mathbf{R}$ with respect to the mesh coordinates in time also forms a block tridiagonal matrix as shown below.

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right] = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 \\ \ddots & \ddots & 0 & 0 & 0 \\ \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-4}} & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-3}} & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-3}} & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}} & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}} & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}} & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}} \end{bmatrix} \tag{3.35}$$

The mesh residual equation for the aeroelastic problem at an arbitrary time-step n is given as

$$\mathbf{G}^n \left(\mathbf{x}^n, \mathbf{x}^n_{surf}\left(\mathbf{r}^n, \mathbf{x}^d_{surf}\right)\right) = 0 \tag{3.36}$$

The surface coordinates $\mathbf{x}^n_{surf}$ at an arbitrary time-step n are computed using the displacement vector $\mathbf{r}^n$ at that time-step provided by the solution of the structural equations and the baseline surface mesh coordinates deformed by the shape functions $\mathbf{x}^d_{surf}$. It can be seen from the mesh residual equation that, at any time-step n, there is dependence on quantities only within the same time-step. Consequently the linearization of the mesh residual with any of the other terms with the exception of $\mathbf{x}^d_{surf}$, results in block diagonal matrices in time. Referring to equation (2.12), it can be seen that the linearization of the mesh residual $\mathbf{G}$ with respect to the mesh coordinates $\mathbf{x}$ results in the constant mesh stiffness matrix $[\mathbf{K}]$, while the linearization of the same with respect to the surface mesh coordinates $\mathbf{x}_{surf}$ results in the negative identity matrix $-[\mathbf{I}]$.

Lastly, examining the structure residual based on the BDF2 temporal discretization at an arbitrary time-step n given as

$$\mathbf{J}^n\left(\mathbf{r}^n, \mathbf{r}^{n-1}, \mathbf{r}^{n-2}, \mathbf{U}^n, \mathbf{x}^n\right) = 0 \tag{3.37}$$

it is clear that the Jacobian matrix $\frac{\partial \mathbf{J}}{\partial \mathbf{r}}$ also forms a block tridiagonal matrix in time as:

$$\left[\frac{\partial \mathbf{J}}{\partial \mathbf{r}}\right] = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 \\ \ddots & \ddots & 0 & 0 & 0 \\ \frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{r}^{n-4}} & \frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{r}^{n-3}} & \frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{r}^{n-2}} & 0 & 0 \\ 0 & \frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-3}} & \frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-2}} & \frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-1}} & 0 \\ 0 & 0 & \frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-2}} & \frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-1}} & \frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n}} \end{bmatrix} \tag{3.38}$$

The structure residual at time-step n depends on the flow and mesh states only at time-step n and therefore the linearization produces block diagonal matrices in time. Substituting these definitions into equation (3.32) the adjoint system can be written in the temporally expanded form as (shown for three points in time)

$$\left[\begin{array}{ccc|ccc|ccc}
{\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}}^T & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}}^T & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{U}^{n-2}}}^T & 0 & 0 & 0 & 0 & 0 \\
0 & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}}^T & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}}^T & 0 & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{U}^{n-1}}}^T & 0 & 0 & 0 & 0 \\
0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}}^T & 0 & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{U}^{n}}}^T & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{r}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-2}}}^T & \frac{\partial \mathbf{x}^{n-2}_{surf}}{\partial \mathbf{r}^{n-2}} & 0 & 0 \\
0 & 0 & 0 & 0 & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-1}}}^T & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-1}}}^T & 0 & \frac{\partial \mathbf{x}^{n-2}_{surf}}{\partial \mathbf{r}^{n-2}} & 0 \\
0 & 0 & 0 & 0 & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n}}}^T & 0 & 0 & \frac{\partial \mathbf{x}^{n-2}_{surf}}{\partial \mathbf{r}^{n-2}} \\
\hline
{\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}}^T & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}}^T & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{x}^{n-2}}}^T & 0 & 0 & [\mathbf{K}]^T & 0 & 0 \\
0 & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}}^T & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}}^T & 0 & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{x}^{n-1}}}^T & 0 & 0 & [\mathbf{K}]^T & 0 \\
0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}}^T & 0 & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{x}^{n}}}^T & 0 & 0 & [\mathbf{K}]^T
\end{array}\right]
\left[\begin{array}{c}
\Lambda_{\mathbf{U}}^{n-2} \\ \Lambda_{\mathbf{U}}^{n-1} \\ \Lambda_{\mathbf{U}}^{n} \\ \hline \Lambda_{\mathbf{r}}^{n-2} \\ \Lambda_{\mathbf{r}}^{n-1} \\ \Lambda_{\mathbf{r}}^{n} \\ \hline \Lambda_{\mathbf{x}}^{n-2} \\ \Lambda_{\mathbf{x}}^{n-1} \\ \Lambda_{\mathbf{x}}^{n}
\end{array}\right]
=
\left[\begin{array}{c}
{\frac{\partial L}{\partial \mathbf{U}^{n-2}}}^T \\ {\frac{\partial L}{\partial \mathbf{U}^{n-1}}}^T \\ {\frac{\partial L}{\partial \mathbf{U}^{n}}}^T \\ \hline {\frac{\partial L}{\partial \mathbf{r}^{n-2}}}^T \\ {\frac{\partial L}{\partial \mathbf{r}^{n-1}}}^T \\ {\frac{\partial L}{\partial \mathbf{r}^{n}}}^T \\ \hline {\frac{\partial L}{\partial \mathbf{x}^{n-2}}}^T \\ {\frac{\partial L}{\partial \mathbf{x}^{n-1}}}^T \\ {\frac{\partial L}{\partial \mathbf{x}^{n}}}^T
\end{array}\right]
\tag{3.39}$$

which when rearranged through row and column swapping into upper triangular form yields

$$\left[\begin{array}{ccc|ccc|ccc}
{\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{U}^{n-2}}}^T & 0 & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}}^T & 0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}}^T & 0 & 0 \\
0 & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{r}^{n-2}}}^T & -\frac{\partial \mathbf{x}^{n-2}_{surf}}{\partial \mathbf{r}^{n-2}} & 0 & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-2}}}^T & 0 & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-2}}}^T & 0 \\
{\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}}^T & {\frac{\partial \mathbf{J}^{n-2}}{\partial \mathbf{x}^{n-2}}}^T & [\mathbf{K}]^T & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}}^T & 0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}}^T & 0 & 0 \\
\hline
0 & 0 & 0 & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}}^T & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{U}^{n-1}}}^T & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}}^T & 0 & 0 \\
0 & 0 & 0 & 0 & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{r}^{n-1}}}^T & -\frac{\partial \mathbf{x}^{n-1}_{surf}}{\partial \mathbf{r}^{n-1}} & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n-1}}}^T & 0 \\
0 & 0 & 0 & {\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}}^T & {\frac{\partial \mathbf{J}^{n-1}}{\partial \mathbf{x}^{n-1}}}^T & [\mathbf{K}]^T & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}}^T & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}}^T & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{U}^{n}}}^T & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{r}^{n}}}^T & -\frac{\partial \mathbf{x}^{n}_{surf}}{\partial \mathbf{r}^{n}} \\
0 & 0 & 0 & 0 & 0 & 0 & {\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}}^T & {\frac{\partial \mathbf{J}^{n}}{\partial \mathbf{x}^{n}}}^T & [\mathbf{K}]^T
\end{array}\right]
\left[\begin{array}{c}
\Lambda_{\mathbf{U}}^{n-2} \\ \Lambda_{\mathbf{r}}^{n-2} \\ \Lambda_{\mathbf{x}}^{n-2} \\ \hline \Lambda_{\mathbf{U}}^{n-1} \\ \Lambda_{\mathbf{r}}^{n-1} \\ \Lambda_{\mathbf{x}}^{n-1} \\ \hline \Lambda_{\mathbf{U}}^{n} \\ \Lambda_{\mathbf{r}}^{n} \\ \Lambda_{\mathbf{x}}^{n}
\end{array}\right]
=
\left[\begin{array}{c}
{\frac{\partial L}{\partial \mathbf{U}^{n-2}}}^T \\ {\frac{\partial L}{\partial \mathbf{r}^{n-2}}}^T \\ {\frac{\partial L}{\partial \mathbf{x}^{n-2}}}^T \\ \hline {\frac{\partial L}{\partial \mathbf{U}^{n-1}}}^T \\ {\frac{\partial L}{\partial \mathbf{r}^{n-1}}}^T \\ {\frac{\partial L}{\partial \mathbf{x}^{n-1}}}^T \\ \hline {\frac{\partial L}{\partial \mathbf{U}^{n}}}^T \\ {\frac{\partial L}{\partial \mathbf{r}^{n}}}^T \\ {\frac{\partial L}{\partial \mathbf{x}^{n}}}^T
\end{array}\right]
\tag{3.40}$$

The solution of the adjoint system now involves a backward sweep in time beginning at the final time-step n. At each time-step in the backward sweep a coupled linear solution is required in order to determine the vector of unknown disciplinary adjoint variables at that time-step. The coupled linear adjoint system at an arbitrary time-step n (except the final two time-steps) can be expressed as

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda_{\mathbf{U}}^n = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{r}}^n - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{r}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{r}}^{n+2} \quad (3.41)$$

$$\left[\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\right]^T \Lambda_{\mathbf{r}}^n = \frac{\partial L}{\partial \mathbf{r}^n}^T + \frac{\partial \mathbf{x}_{surf}^n}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{x}}^n - \frac{\partial \mathbf{J}^{n+1}}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{r}}^{n+1} - \frac{\partial \mathbf{J}^{n+2}}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{r}}^{n+2} \quad (3.42)$$

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}}^n = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^n - \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{r}}^n - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+2} \quad (3.43)$$

where the coupling between the disciplinary adjoint equations is clear since adjoint variables from the same time-step of each discipline appear in the form of source terms in the adjoint equations of the remaining disciplines.

Comparing the disciplinary adjoint equations above with equations (2.37), (2.40) and (2.11), it can be seen that linear systems arising in the analysis problem and the adjoint problem are very similar. The coefficient matrices operating on the vector of unknowns in both cases are the same with the exception of the transpose. In the analysis problem, the Jacobian matrices are computed in order to drive the nonlinear Newton solver and these are readily available for use in the solution of the adjoint problem. Transpose operations do not change the eigenvalues of the coefficient matrix and consequently the convergence of the linear adjoint systems are similar to those appearing as intermediate linear systems in the analysis problem. The solution of the coupled adjoint system at each time-step follows the same method outlined for the analysis problem. Partial solutions of the adjoint equations of each discipline are required while the source terms due to coupling are constantly updated, until all of the disciplinary adjoint equations converge. Figure 3.1 compares the typical convergence of the coupled adjoint system at an arbitrary time-step for the coupled aeroelasticity problem. Once the vector of disciplinary adjoint variables spanning both the spatial and temporal domains has been solved for in this manner, they can

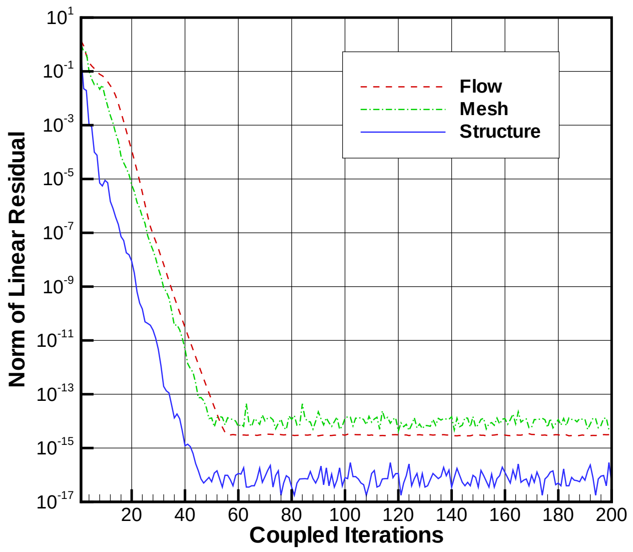


Figure 3.1: Typical convergence of the coupled aeroelastic adjoint equations.

then be substituted into the total sensitivity shown in equation (3.31) to obtain

$$\frac{dL}{d\mathbf{D}}^T = \left[\begin{array}{ccc} 0 & 0 & -\left(\frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}^d_{surf}}\right)^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{r}} \\ \Lambda_{\mathbf{x}} \end{array}\right] = -\frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}^d_{surf}}^T \Lambda_{\mathbf{x}} \quad (3.44)$$

Referring to equation (3.36), it can be seen that the mesh residual $\mathbf{G}$ at each time-step depends on the deformed baseline surface coordinates $\mathbf{x}^d_{surf}$ through the surface coordinates $\mathbf{x}_{surf}$ at that time-step. The total sensitivity can therefore be further split as

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{x}^d_{surf}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T \Lambda_{\mathbf{x}} \quad (3.45)$$

The derivative of the surface mesh coordinates $\mathbf{x}_{surf}$ at any time-step with respect to the deformed baseline surface coordinates $\mathbf{x}^d_{surf}$ can be defined as the transformation matrix $[\beta]$ for that time-step. The transformation matrix is used to rotate and/or translate the baseline surface coordinates defining the airfoil to the correct orientation at the current time-step. For the aeroelastic case, the transformation matrix is easily computable once the displacement of the airfoil is known from the solution of the structural equations. Based on this definition of the transformation matrix and that the linearization of the mesh residual $\mathbf{G}^n$ with respect to the surface coordinates $\mathbf{x}^n_{surf}$ is the negative identity matrix, the total sensitivity then becomes

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T [\beta]^T \Lambda_{\mathbf{x}} \tag{3.46}$$

Now expanding the transformation matrix $[\beta]$ and the vector of mesh adjoint variables discretely in time, the complete gradient vector can be computed as the matrix-vector product between the sensitivity of the deformed baseline surface coordinates and the sum in time of the transformed mesh adjoint variable as

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \left\{ \sum_{n=1}^{n_{steps}} [\beta^n]^T \Lambda^n_{\mathbf{x}} \right\} \tag{3.47}$$

3.4 Gradients for the Uncoupled Unsteady Flow Problem

The uncoupled unsteady problem can be viewed as a subset of the coupled unsteady aeroelastic problem, where now the functional is dependent only on two states namely the flow and mesh solutions as:

$$L = L(\mathbf{U}, \mathbf{x}) \tag{3.48}$$

The state variables still span both spatial and temporal domains and the corresponding adjoint form of the total sensitivity can be expressed as

$$\frac{dL}{d\mathbf{D}}^T = \left[\begin{array}{cc} \dfrac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \dfrac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{array} \right] \left[\begin{array}{c} \dfrac{\partial L}{\partial \mathbf{U}}^T \\ \dfrac{\partial L}{\partial \mathbf{x}}^T \end{array} \right] \tag{3.49}$$

The set of residual equations for the non-aeroelastic unsteady case can be summarized as

$$\mathbf{R}(\mathbf{U},\mathbf{x}) = 0 \tag{3.50}$$

$$\mathbf{G}(\mathbf{x},\mathbf{x}_{surf}(\mathbf{x}^d_{surf})) = 0 \tag{3.51}$$

which when linearized, combined into matrix form and transposed following the same procedure as explained in the previous section yields an expression for the vector of state variable sensitivity matrices as

$$\begin{bmatrix} \dfrac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \dfrac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{bmatrix} = \begin{bmatrix} 0 & -\left(\dfrac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \dfrac{\partial \mathbf{G}}{\partial \mathbf{x}^d_{surf}}^T \right) \end{bmatrix} \begin{bmatrix} \dfrac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & 0 \\ \dfrac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \dfrac{\partial \mathbf{G}}{\partial \mathbf{x}}^T \end{bmatrix}^{-1} \tag{3.52}$$

The corresponding unsteady adjoint system is then

$$\begin{bmatrix} \dfrac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & 0 \\ \dfrac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \dfrac{\partial \mathbf{G}}{\partial \mathbf{x}}^T \end{bmatrix} \begin{bmatrix} \Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial L}{\partial \mathbf{U}}^T \\ \dfrac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \tag{3.53}$$

The Jacobian matrices appearing in the system of adjoint equations take up the same form as those in the case of aeroelasticity since the respective residual equations are nearly identical. The flow residual equation for this case remains identical to that shown in equation (3.33), while the mesh residual at an arbitrary time-step n now does not have any dependence on the structural state and is given as

$$\mathbf{G}^n\left(\mathbf{x}^n,\mathbf{x}^n_{surf}(\mathbf{x}^d_{surf})\right) = 0 \tag{3.54}$$

Expanding the adjoint system discretely in time, equation (3.53) can be expressed as (shown for a time domain with three steps)

$$\left[\begin{array}{ccc|ccc} \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}^T & 0 & 0 & 0 \\ 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}^T & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}^T & 0 & 0 & 0 \\ \hline \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}^T & [\mathbf{K}]^T & 0 & 0 \\ 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}^T & 0 & [\mathbf{K}]^T & 0 \\ 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}^T & 0 & 0 & [\mathbf{K}]^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathbf{U}}^{n-2} \\ \Lambda_{\mathbf{U}}^{n-1} \\ \Lambda_{\mathbf{U}}^{n} \\ \hline \Lambda_{\mathbf{x}}^{n-2} \\ \Lambda_{\mathbf{x}}^{n-1} \\ \Lambda_{\mathbf{x}}^{n} \end{array}\right] = \left[\begin{array}{c} \frac{\partial L}{\partial \mathbf{U}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{x}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{x}^{n}}^T \end{array}\right] \tag{3.55}$$

which when rearranged through row and column swapping results in an upper triangular system of linear equations as:

$$\left[\begin{array}{cc|cc|cc} [\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}^T \\ 0 & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}^T \\ \hline 0 & 0 & [\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}^T \\ 0 & 0 & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}^T \\ \hline 0 & 0 & 0 & 0 & [\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}^T \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}^T \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathbf{x}}^{n-2} \\ \Lambda_{\mathbf{U}}^{n-2} \\ \hline \Lambda_{\mathbf{x}}^{n-1} \\ \Lambda_{\mathbf{U}}^{n-1} \\ \hline \Lambda_{\mathbf{x}}^{n} \\ \Lambda_{\mathbf{U}}^{n} \end{array}\right] = \left[\begin{array}{c} \frac{\partial L}{\partial \mathbf{x}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-2}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n}}^T \end{array}\right] \tag{3.56}$$

The solution is obtained again through a backward sweep in time beginning at the final time-step n, but the difference is that the temporal diagonal blocks are themselves upper triangular for this case. This indicates that there is no coupling between the disciplinary adjoint equations, namely the flow and mesh, and is expected since there is no coupling between the two in the analysis problem. The vector of disciplinary adjoint variables can be obtained by sweeping back in time while at each time-step, first the unknown flow adjoint is solved for before the mesh adjoint solution is obtained. At any arbitrary time-step n (except the final two time-steps), the linear flow and mesh adjoint equations take up the form shown below.

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda_{\mathbf{U}}^n = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{U}}^{n+2} \tag{3.57}$$

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}}^n = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+2} - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+2} \tag{3.58}$$

The total sensitivity equation now becomes

$$\frac{dL}{d\mathbf{D}}^T = \begin{bmatrix} 0 & -\left(\frac{\partial \mathbf{x}_{surf}^d}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}^d}^T\right) \end{bmatrix} \begin{bmatrix} \Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{x}} \end{bmatrix} = \frac{\partial \mathbf{x}_{surf}^d}{\partial \mathbf{D}}^T [\beta]^T \Lambda_{\mathbf{x}} \tag{3.59}$$

The transformation matrix is constructed using the prescribed motion in time of the surface coordinates defining the airfoil. As in the case of the aeroelastic problem, the complete gradient vector simply becomes a matrix-vector product between the sensitivity of the deformed airfoil to the design inputs and the sum in time of the transformed mesh adjoint variables as shown in equation (3.47).

3.5 Steady-State Flow Problem

The steady-state problem is a subset of the unsteady problem in general, with the only difference being the span of the solution. For the steady-state case, the solutions span only the spatial domain, however the problem itself may be coupled or uncoupled in nature. The discrete temporal expansion of the system of adjoint equations in the case of the unsteady flow problem or the coupled unsteady aeroelastic problem is absent for steady-state problems. However, the formulation of the adjoint system remains identical. For the case of a steady-state flow problem with no coupling between disciplines (non aeroelastic solution), referring to equation (3.53), the steady-state flow adjoint variable can be obtained through a single spatial solution of the form

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^T \Lambda_{\mathbf{U}} = \frac{\partial L}{\partial \mathbf{U}}^T \tag{3.60}$$

followed by the solution for the mesh adjoint variable as

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}} = \frac{\partial L}{\partial \mathbf{x}}^T - \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T \Lambda_{\mathbf{U}} \tag{3.61}$$

The mesh residual for the steady-state problem is simply

$$\mathbf{G}(\mathbf{x}, \mathbf{x}^d_{surf}) = 0 \tag{3.62}$$

indicating that the complete gradient can be determined through the matrix-vector product of deformed surface coordinates sensitivity to design inputs directly with the mesh adjoint variable as:

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial \mathbf{x}^d_{surf}}{\partial \mathbf{D}}^T \Lambda_{\mathbf{x}} \tag{3.63}$$

3.6 Linearization for Second-Order Spatial Accuracy

Computation of the flow adjoint variable requires the linearization of the flow residual equation with respect to the flow state variables. Such a linearization results in the flow Jacobian matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$ and is also used in the Newton solver employed for the flow solution. In the case of a spatially first-order accurate discretization, the flow Jacobian matrix is constructed using a nearest neighbor stencil where, for any element in the mesh, the flow residual $\mathbf{R}$ is a function of its own state variables and the state variables of its immediate neighbors. On triangular unstructured meshes this translates into the flow Jacobian matrix being sparse with each row consisting of a dominant diagonal block element and three off-diagonal block elements. A simple edge-based data structure is sufficient for the purpose of constructing and storing the elements of the Jacobian matrix. In the case of second-order spatial accuracy, the flow residual of each element is no longer restricted to a nearest neighbor stencil since the residual $\mathbf{R}$ depends not only on the state variables but also their undivided Laplacian. This is can be seen in equation (2.19) describing the matrix dissipation scheme used to construct the fluxes with second-order accuracy, where $(\nabla^2\mathbf{U})$ is the undivided Laplacian of the state vector $\mathbf{U}$. Since the residual is a now a function of element states extending beyond the nearest neighbor stencil, constructing and storing an exact flow Jacobian matrix quickly becomes impractical due to large memory requirements and the lack of a simple data structure. For the flow solver, it is not required that an exact linearization of the flow residual $\mathbf{R}$ be available since only the solution of $\mathbf{R}(\mathbf{U}) = 0$ is necessary. In order to achieve second-order accuracy, it is only required that the flow residual itself be constructed for second-order accuracy of $\mathbf{U}$. An inexact Newton's method is employed where the first-order accurate flow Jacobian matrix is used in solving second-order accurate residual equations $\mathbf{R}_2(\mathbf{U})$. The method being inexact no longer exhibits the quadratic rate of convergence typical of a Newton solver but greatly simplifies the solution procedure.

In the case of the adjoint solver, it has been shown that approximating a second-order accurate flow Jacobian matrix with a first-order accurate matrix leads to significant errors in the computed sensitivities [71]. The linear systems that involve the flow Jacobian matrix are equations (3.41), (3.57), and (3.60). Although computing and storing the exact second-order Jacobian is impractical, it is quite straightforward to construct the product of the second-order Jacobian and a vector using a two-pass approach [30,31]. This property is taken advantage of and a defect correction method is used to solve the linear system involving the second-order flow Jacobian matrix. For example, consider the left-hand-side of the linear system shown in equation (3.60).

In the case where the flow Jacobian matrix is second-order accurate, this can be split into the sum of two matrix-vector products as follows:

$$\begin{aligned}\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]_2^T \Lambda_{\mathbf{U}} &= \left\{\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]_1 + \left[\frac{\partial \mathbf{R}}{\partial (\nabla^2 \mathbf{U})}\right]\left[\frac{\partial (\nabla^2 \mathbf{U})}{\partial \mathbf{U}}\right]\right\}^T \Lambda_{\mathbf{U}} \\ &= \underbrace{\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]_1^T \Lambda_{\mathbf{U}}}_{(1)} + \underbrace{\left[\frac{\partial (\nabla^2 \mathbf{U})}{\partial \mathbf{U}}\right]^T \left[\frac{\partial \mathbf{R}}{\partial (\nabla^2 \mathbf{U})}\right]^T \Lambda_{\mathbf{U}}}_{(2)} \end{aligned} \tag{3.64}$$

where $[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}]_1$ is the first-order Jacobian. Term (1) can be easily computed since the first-order Jacobian is already stored for the solution of the analysis problem. Term (2) can be computed via a series of matrix-vector products on the fly if $\Lambda_{\mathbf{U}}$ is available. The defect correction method is a pseudo nonlinear solution method and for equation (3.60) can be expressed as

$$\begin{aligned}[\mathbf{P}]\delta\Lambda_{\mathbf{U}}^k &= -\left\{\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]_2^T \Lambda_{\mathbf{U}}^k - \left[\frac{\partial L}{\partial \mathbf{U}}\right]^T\right\} \\ \Lambda_{\mathbf{U}}^{k+1} &= \Lambda_{\mathbf{U}}^k + \delta\Lambda_{\mathbf{U}}^k\end{aligned} \tag{3.65}$$

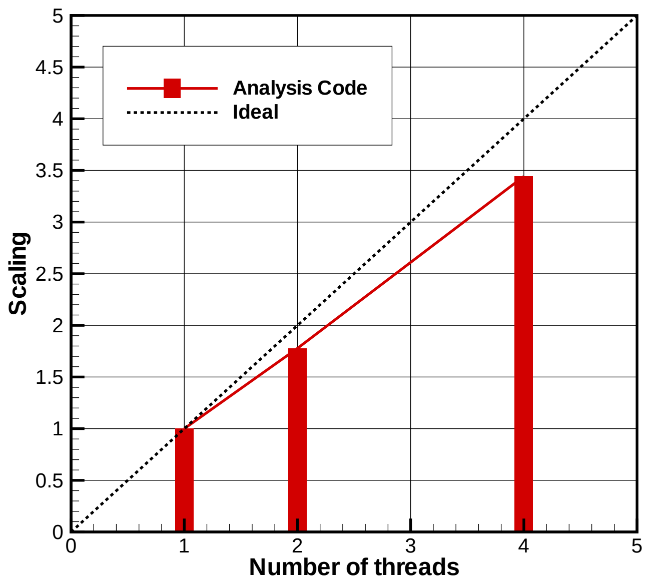
Here $[\mathbf{P}]$ refers to the preconditioner and $\delta\Lambda_{\mathbf{U}}$ the correction for $\Lambda_{\mathbf{U}}$. When the correct value for $\Lambda_{\mathbf{U}}$ has been achieved, the right-hand-side of equation (3.65) goes to zero. In order for the system to converge, the preconditioner $[\mathbf{P}]$ must be closely related to the second-order flow Jacobian matrix. Typically the preconditioner is taken to be the transpose of the first-order flow Jacobian matrix [31, 72]. The right-hand-side of equation (3.65) includes the effect of the second-order flow Jacobian matrix and is evaluated as described by equation (3.64).

3.7 Optimization Algorithm

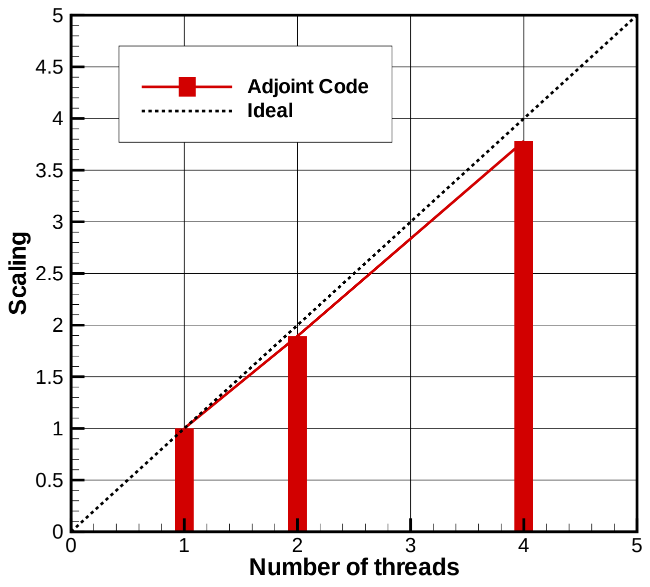
The simplest approach for solving the optimization problem is to use steepest-descent or some form of a line-search algorithm, but it has been determined in previous work [37] that these methods perform poorly on stiff nonlinear optimization problems particularly when the number of design parameters is large. The optimization examples presented use the LBFGS-B bounded reduced Hessian algorithm developed in reference [64]. The algorithm not only uses gradient information but also builds an approximate Hessian matrix (2^{nd} derivative of the functional with respect to design parameters) on-the-fly using the optimization history in order to speed up the overall convergence to minimum. The bounded rather than the unbounded LBFGS algorithm was chosen in order to prevent the generation of invalid or non-physical geometries during the optimization process. The actual bounds themselves are established *a priori* by manually exploring the effect of the design variables on the shape of the airfoil and limiting them to regions which do not collapse the geometry.

3.8 Sensitivity Solver Implementation Details

The analysis solver and the sensitivity solver form two separate codes that share several subroutines in the implementation. The analysis code is second-order accurate both spatially and temporally, while the adjoint code is a discretely exact linearization of all aspects of the analysis code. Both codes run in parallel on multi-core hardware architectures with commonly addressable memory using OpenMP parallelization. All results presented were run on a quad-core Intel desktop with 4 gigabytes of memory using 4 OpenMP threads. Both codes exhibit good scaling between 1,2 and 4 cores, with the adjoint code scaling slightly better than the analysis code. This is most likely due to the linear nature of the adjoint code where, once the coefficient matrices have been setup, only a single linear solution of a coupled system per time-step is required. On the contrary, the nonlinear nature of the analysis problem requires repeated construction of coefficient matrices for multiple linear solutions at each time-step. Figures 3.2(a) and 3.2(b) compare the performance scaling between 1,2 and 4 cores for the analysis and adjoint problems respectively. It should also be noted that the solution of the adjoint problem typically consumes less time than the analysis problem. Again, this is attributed to the fact that only linear solutions are required in the adjoint problem while the analysis problem requires both nonlinear and linear solutions. The analysis code performs the forward integration in time and writes the solution of the coupled system at each time-step to the hard-drive, which is then read by the adjoint solver sweeping backward in time. It is necessary that the unsteady solution set be written to disk and read-in by the adjoint solver since the adjoint code involves a backward sweep in time. Holding the entire solution set in memory would quickly become impractical for large problems. However, disk I/O operations consume trivial computational resources especially when done in a piecewise manner (i.e. at each time-step) as in this case.



(a) Analysis Code



(b) Adjoint Code

Figure 3.2: Performance scaling of analysis and adjoint codes using OpenMP parallelization.

Chapter 4

Functional Relevant Error Estimation

4.1 Generalized Error Formulation for Coupled Multidisciplinary Unsteady Equations

In this section, an adjoint based approach for estimating functional relevant temporal discretization error and algebraic error due to partial convergence is presented. It is shown that the adjoint variables obtained in the process of computing the sensitivity of a functional with respect to design inputs can be used for the additional purpose of estimating functional relevant error, provided the functional in both cases is the same.

4.1.1 Discipline Specific Temporal Discretization Error

As in the case of the derivation of the gradient, consider an arbitrary number of m coupled unsteady disciplines, with the solution represented by the set of state variables $\mathcal{U}$ spanning the spatial and temporal domains. For a solution obtained on a temporal domain of arbitrary resolution h, a functional computed using the coupled solution set can be represented as

$$L_h = L_h(\mathcal{U}_h) = L_h \left\{ (\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m)_h \right\} \tag{4.1}$$

It is possible to express the functional as a Taylor expansion about the same functional evaluated using a fully converged solution obtained on a coarser temporal domain of resolution H projected onto the finer resolution temporal domain h as

$$\begin{aligned} L_h(\mathcal{U}_h) = L_h(\mathcal{U}_h^H) &+ \left[\frac{\partial L}{\partial \mathcal{U}_1}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{1h} - \mathcal{U}_{1h}^H) \\ &+ \left[\frac{\partial L}{\partial \mathcal{U}_2}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{2h} - \mathcal{U}_{2h}^H) \\ &\vdots \\ &+ \left[\frac{\partial L}{\partial \mathcal{U}_m}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{mh} - \mathcal{U}_{mh}^H) \\ &+ \mathcal{O}\left(\mathcal{U}_h - \mathcal{U}_h^H\right)^2 + \mathcal{O}\left(\mathcal{U}_h - \mathcal{U}_h^H\right)^3 \cdots \end{aligned} \tag{4.2}$$

where $\mathcal{U}_h^H$ refers to the state variables (all disciplines) from the coarse resolution temporal solution projected onto the finer temporal domain. Ignoring higher-order terms in the Taylor expansion, the linear approximation of the error in the functional evaluated on the fine temporal domain using projected solutions from the coarse temporal domain can be written as the inner product between the vector of functional sensitivities to states and the vector of error in the states as

$$L_h(\mathcal{U}_h) - L_h(\mathcal{U}_h^H) = \begin{bmatrix} \dfrac{\partial L}{\partial \mathcal{U}_1} & \dfrac{\partial L}{\partial \mathcal{U}_2} & \cdots & \dfrac{\partial L}{\partial \mathcal{U}_m} \end{bmatrix}_{\mathcal{U}_h^H} \begin{bmatrix} (\mathcal{U}_{1h} - \mathcal{U}_{1h}^H) \\ (\mathcal{U}_{2h} - \mathcal{U}_{2h}^H) \\ \vdots \\ (\mathcal{U}_{mh} - \mathcal{U}_{mh}^H) \end{bmatrix} \tag{4.3}$$

where it is important to note that the error is relative to the functional evaluated using a fully converged solution obtained on the fine temporal domain h and not the exact analytic solution of the governing equations. The vector of differences in the disciplinary state variables between the fine temporal domain solution and the projection of the coarse temporal domain solution onto the fine temporal domain can be computed if both the fine and coarse temporal domain solutions were available. However, the goal is to to estimate the fine temporal domain functional using only coarse level information so as to avoid any computations directly on the fine temporal domain.

In order to determine the vector of differences between the fine level solutions and the coarse level solutions projected onto the fine level, the property that a fully converged fine temporal domain solution must satisfy the fine temporal domain residuals exactly is exploited. The disciplinary residual equations have to evaluate to zero if the true fine level solution were available, however no solutions are to be obtained directly on the fine temporal domain. It is possible to write the fine level disciplinary residuals as Taylor expansions about the fine level residuals evaluated on the fine temporal domain, using solutions projected from the coarse temporal domain. In general, for any discipline j, the Taylor expansion of the residual up to a linear approximation can be written as

$$
\begin{aligned}
\mathcal{R}_{j_h}(\mathcal{U}_h) = \mathcal{R}_{j_h}(\mathcal{U}_h^H) &+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_1}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{1h} - \mathcal{U}_{1h}^{\ H}) \\
&+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_2}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{2h} - \mathcal{U}_{2h}^{\ H}) \\
&\vdots \\
&+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_m}\right]_{\mathcal{U}_h^H} (\mathcal{U}_{mh} - \mathcal{U}_{mh}^{\ H}) \cdots = 0
\end{aligned}
\tag{4.4}
$$

The Taylor expansions for all m disciplines can be expanded in the same manner and be written in a unified matrix form as

$$\begin{bmatrix} \mathcal{R}_{1h}(\mathcal{U}_h) \\ \mathcal{R}_{2h}(\mathcal{U}_h) \\ \vdots \\ \mathcal{R}_{mh}(\mathcal{U}_h) \end{bmatrix} \approx \begin{bmatrix} \mathcal{R}_{1h}(\mathcal{U}_h^H) \\ \mathcal{R}_{2h}(\mathcal{U}_h^H) \\ \vdots \\ \mathcal{R}_{mh}(\mathcal{U}_h^H) \end{bmatrix} + \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m} \\ \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m} \\ \vdots & & & \\ \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m} \end{bmatrix}_{\mathcal{U}_h^H} \begin{bmatrix} (\mathcal{U}_{1h} - \mathcal{U}_{1h}^H) \\ (\mathcal{U}_{2h} - \mathcal{U}_{2h}^H) \\ \vdots \\ (\mathcal{U}_{mh} - \mathcal{U}_{mh}^H) \end{bmatrix} = 0 \tag{4.5}$$

which can then be rearranged to obtain an expression for the vector of differences between the true fine temporal domain solution set represented by the state variables $\mathcal{U}_h$, and the projection of the state variables from the coarse temporal domain solution onto the fine temporal domain $\mathcal{U}_h^H$ as

$$\begin{bmatrix} (\mathcal{U}_{1h} - \mathcal{U}_{1h}^H) \\ (\mathcal{U}_{2h} - \mathcal{U}_{2h}^H) \\ \vdots \\ (\mathcal{U}_{mh} - \mathcal{U}_{mh}^H) \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m} \\ \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m} \\ \vdots & & & \\ \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m} \end{bmatrix}_{\mathcal{U}_h^H}^{-1} \begin{bmatrix} \mathcal{R}_{1h}(\mathcal{U}_h^H) \\ \mathcal{R}_{2h}(\mathcal{U}_h^H) \\ \vdots \\ \mathcal{R}_{mh}(\mathcal{U}_h^H) \end{bmatrix} \tag{4.6}$$

The expression may now be substituted into the functional error equation (4.3) yielding

$$L_h(\mathcal{U}_h) - L_h(\mathcal{U}_h^H) =$$

$$\underbrace{-\begin{bmatrix} \frac{\partial L}{\partial \mathcal{U}_1} & \frac{\partial L}{\partial \mathcal{U}_2} & \cdots & \frac{\partial L}{\partial \mathcal{U}_m} \end{bmatrix}_{\mathcal{U}_h^H} \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m} \\ \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m} \\ \vdots & & & \\ \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m} \end{bmatrix}^{-1}_{\mathcal{U}_h^H}}_{\Lambda^T_{\mathcal{U}_h}} \begin{bmatrix} \mathcal{R}_{1h}(\mathcal{U}_h^H) \\ \mathcal{R}_{2h}(\mathcal{U}_h^H) \\ \vdots \\ \mathcal{R}_{mh}(\mathcal{U}_h^H) \end{bmatrix} \tag{4.7}$$

where a vector of disciplinary adjoint variables $\Lambda^T_{\mathcal{U}_h}$ on the fine temporal domain is introduced in order to recover a system of coupled adjoint equations as

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\mathcal{U}_h^H} \begin{bmatrix} \Lambda^T_{\mathcal{U}_{1h}} \\ \Lambda^T_{\mathcal{U}_{2h}} \\ \vdots \\ \Lambda^T_{\mathcal{U}_{mh}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \mathcal{U}_1}^T \\ \frac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \frac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\mathcal{U}_h^H} \tag{4.8}$$

This is a coupled linear system of adjoint equations nearly identical to equation (3.17) with the only difference being that it is to be solved on the fine level temporal domain using the solution from the coarse temporal domain projected onto the fine domain. The procedure to solve the system of adjoint equations is identical to that presented in the derivation of the adjoint based gradient. However, direct solutions on the fine temporal domain are to be avoided since the goal is to predict the fine domain functional without actually solving for anything on the fine domain. For this reason, an approximation is made where the system of adjoint equations is recast on the coarse temporal domain as

$$\begin{bmatrix} \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \dfrac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \dfrac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \dfrac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\mathcal{U}_H} \begin{bmatrix} \Lambda^T_{\mathcal{U}_{1H}} \\ \Lambda^T_{\mathcal{U}_{2H}} \\ \vdots \\ \Lambda^T_{\mathcal{U}_{mH}} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial L}{\partial \mathcal{U}_1}^T \\ \dfrac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \dfrac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\mathcal{U}_H} \tag{4.9}$$

where $\mathcal{U}_H$ refers to the fully converged solution set on the coarse temporal domain, and $\Lambda_{\mathcal{U}_H}$ refers to adjoint variables on the coarse temporal domain. Once the adjoint variables have been determined on the coarse temporal domain, the fine temporal domain adjoint variables are estimated through some appropriate projection operator I_h^H as

$$\Lambda_{\mathcal{U}_h} = I_h^H \Lambda_{\mathcal{U}_H} \tag{4.10}$$

The vector of disciplinary adjoint variables can then be substituted back into the functional error equation to obtain the temporal discretization error as

$$\varepsilon_{disc} = L_h(\mathcal{U}_h) - L_h(\mathcal{U}_h^H) = -\begin{bmatrix} \Lambda^T_{\mathcal{U}_{1h}} & \Lambda^T_{\mathcal{U}_{2h}} & \cdots & \Lambda^T_{\mathcal{U}_{mh}} \end{bmatrix} \begin{bmatrix} \mathcal{R}_{1h}(\mathcal{U}_h^H) \\ \mathcal{R}_{2h}(\mathcal{U}_h^H) \\ \vdots \\ \mathcal{R}_{mh}(\mathcal{U}_h^H) \end{bmatrix} \tag{4.11}$$

which can be interpreted as the sum of the inner products of the disciplinary adjoint variables and their corresponding residuals as:

$$\begin{aligned} \varepsilon_{disc} = L_h(\mathcal{U}_h) - & L_h(\mathcal{U}_h^H) = \\ & -\left\{\Lambda^T_{\mathcal{U}_{1h}}\mathcal{R}_{1h}(\mathcal{U}_h^H) + \Lambda^T_{\mathcal{U}_{2h}}\mathcal{R}_{2h}(\mathcal{U}_h^H) + \cdots + \Lambda^T_{\mathcal{U}_{mh}}\mathcal{R}_{mh}(\mathcal{U}_h^H)\right\} \end{aligned} \tag{4.12}$$

The residuals are nonzero since they are evaluated on the fine temporal domain using solutions projected from the coarse temporal domain. It can be seen that the difference in the functional L evaluated using a solution obtained directly on the fine temporal domain h and the functional evaluated using the projection of the solution set from the coarse temporal domain H, has contributions from each discipline. The contribution from any discipline is equal to the negative of the inner product between the nonzero residual of that discipline and the disciplinary adjoint variable. While the nonzero residuals constructed on the fine temporal domain are an estimate of the local discretization error between the two temporal meshes, the application of the adjoint variable as a weight acting on the local error results in the conversion of the local error into a functional relevant error. The inner product within each discipline yielding a contribution to the total functional relevant error can further be discretely expanded in time to obtain a temporal distribution of the error. As an example, for any particular discipline j, on a fine level temporal domain represented by n time-steps, the inner product may be expressed as the sum of inner products at each time-step as

$$\Lambda^T_{\mathcal{U}_j h}\mathcal{R}_{j h}(\mathcal{U}^H_h) = {\Lambda^T_{\mathcal{U}_j h}}^1 \mathcal{R}_{j h}^1(\mathcal{U}^H_h) + {\Lambda^T_{\mathcal{U}_j h}}^2 \mathcal{R}_{j h}^2(\mathcal{U}^H_h) \cdots + {\Lambda^T_{\mathcal{U}_j h}}^n \mathcal{R}_{j h}^n(\mathcal{U}^H_h) \tag{4.13}$$

The distribution in time of the temporal discretization error provides the criteria necessary for the adaptation of the temporal domain (i.e. refinement of the time-steps) specifically to improve the functional L. The method presents two advantages, namely the estimation of a linear error correction term for the functional evaluated on the fine temporal domain using coarse level solutions and also the disciplinary and temporal distribution of the error which can be used for adaptation.

4.1.2 Discipline Specific Algebraic Error

The next step is to formulate a similar error estimation procedure to target the algebraic error in the solution due to partial convergence of the governing equations. Consider a temporal domain of some arbitrary resolution H. It is possible to write a functional evaluated using a fully converged solution on this mesh as a Taylor expansion about the same functional evaluated using a partially converged solution on the same mesh as

$$\begin{aligned} L_H(\mathcal{U}_H) = L_H(\overline{\mathcal{U}}_H) &+ \left[\frac{\partial L}{\partial \mathcal{U}_1}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{1H} - \overline{\mathcal{U}}_{1H}) \\ &+ \left[\frac{\partial L}{\partial \mathcal{U}_2}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{2H} - \overline{\mathcal{U}}_{2H}) \\ &\vdots \\ &+ \left[\frac{\partial L}{\partial \mathcal{U}_m}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{mH} - \overline{\mathcal{U}}_{mH}) \\ &+ \mathcal{O}\left(\mathcal{U}_H - \overline{\mathcal{U}}_H\right)^2 + \mathcal{O}\left(\mathcal{U}_H - \overline{\mathcal{U}}_H\right)^3 \cdots \end{aligned} \tag{4.14}$$

where $\mathcal{U}_H$ refers to the fully converged solution set spanning all disciplines on the temporal domain of resolution H, and $\overline{\mathcal{U}}_H$ refers to the partially converged version of the same solution. The error due to partial convergence of the governing equations is the algebraic error in the solution, and the goal is to obtain an estimate of this error without actually fully converging the system of coupled governing equations. Similar to the case of temporal discretization error, the expression may be rewritten in terms of the inner product between the vector of functional sensitivities to partially converged disciplinary solutions and the vector of differences between fully converged and partially converged disciplinary solutions as

$$L_H(\mathcal{U}_H) - L_H(\overline{\mathcal{U}}_H) = \left[\begin{array}{cccc} \frac{\partial L}{\partial \mathcal{U}_1} & \frac{\partial L}{\partial \mathcal{U}_2} & \cdots & \frac{\partial L}{\partial \mathcal{U}_m} \end{array}\right]_{\overline{\mathcal{U}}_H} \left[\begin{array}{c} (\mathcal{U}_{1H} - \overline{\mathcal{U}}_{1H}) \\ (\mathcal{U}_{2H} - \overline{\mathcal{U}}_{2H}) \\ \vdots \\ (\mathcal{U}_{mH} - \overline{\mathcal{U}}_{mH}) \end{array}\right] \tag{4.15}$$

As in the case of temporal discretization error, the disciplinary residual equations are tapped to obtain an expression for the vector of differences between fully converged and partially converged disciplinary solutions. In the case of temporal discretization error, the argument regarding the residuals was that they would evaluate to zero on the fine temporal domain if the solution was directly obtained on the fine temporal domain. For the case of partial convergence on the temporal domain with resolution H, the equivalent argument is that the disciplinary residuals evaluate to zero on H if the solution is fully converged. Expressing the fully converged residual for any discipline j as the Taylor expansion about the partially converged residual yields

$$\begin{aligned} \mathcal{R}_{jH}(\mathcal{U}_H) = \mathcal{R}_{jH}(\overline{\mathcal{U}}_H) &+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_1}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{1H} - \overline{\mathcal{U}}_{1H}) \\ &+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_2}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{2H} - \overline{\mathcal{U}}_{2H}) \\ &\vdots \\ &+ \left[\frac{\partial \mathcal{R}_j}{\partial \mathcal{U}_m}\right]_{\overline{\mathcal{U}}_H} (\mathcal{U}_{mH} - \overline{\mathcal{U}}_{mH}) \cdots = 0 \end{aligned} \tag{4.16}$$

The Taylor expansion when applied to all disciplines and simplified into combined matrix form yields an expression for the unknown difference vector as

$$\begin{bmatrix} (\mathcal{U}_{1H} - \overline{\mathcal{U}}_{1H}) \\ (\mathcal{U}_{2H} - \overline{\mathcal{U}}_{2H}) \\ \vdots \\ (\mathcal{U}_{mH} - \overline{\mathcal{U}}_{mH}) \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m} \\ \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m} \\ \vdots & & & \\ \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1} & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2} & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m} \end{bmatrix}_{\overline{\mathcal{U}}_H}^{-1} \begin{bmatrix} \mathcal{R}_{1H}(\overline{\mathcal{U}}_H) \\ \mathcal{R}_{2H}(\overline{\mathcal{U}}_H) \\ \vdots \\ \mathcal{R}_{mH}(\overline{\mathcal{U}}_H) \end{bmatrix} \tag{4.17}$$

Substituting the above expression for the vector of solution differences into the functional error equation (4.15) and defining a vector of disciplinary adjoint variables, a coupled adjoint system is recovered as

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_1}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_1}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_1}^T \\ \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_2}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_2}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_2}^T \\ \vdots & & & \\ \frac{\partial \mathcal{R}_1}{\partial \mathcal{U}_m}^T & \frac{\partial \mathcal{R}_2}{\partial \mathcal{U}_m}^T & \cdots & \frac{\partial \mathcal{R}_m}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\overline{\mathcal{U}}_H} \begin{bmatrix} \Lambda_{\mathcal{U}_{1H}}^T \\ \Lambda_{\mathcal{U}_{2H}}^T \\ \vdots \\ \Lambda_{\mathcal{U}_{mH}}^T \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \mathcal{U}_1}^T \\ \frac{\partial L}{\partial \mathcal{U}_2}^T \\ \vdots \\ \frac{\partial L}{\partial \mathcal{U}_m}^T \end{bmatrix}_{\overline{\mathcal{U}}_H} \tag{4.18}$$

The system of adjoint equations shown above is nearly identical to the adjoint system shown in equation (4.9), with the only difference being that all terms appearing in the system are evaluated using the partially rather than fully converged solution set on the coarse level temporal mesh H. Substituting the vector of adjoint variables back into the functional error equation (4.15), the algebraic error in the functional can be evaluated as

$$\begin{aligned}\varepsilon_{alg} = L_H(\mathcal{U}_H) - L_H(\overline{\mathcal{U}}_H) = \\ - \left\{\Lambda^T_{\mathcal{U}_{1H}}\mathcal{R}_{1H}(\overline{\mathcal{U}}_H) + \Lambda^T_{\mathcal{U}_{2H}}\mathcal{R}_{2H}(\overline{\mathcal{U}}_H) + \cdots \Lambda^T_{\mathcal{U}_{mH}}\mathcal{R}_{mH}(\overline{\mathcal{U}}_H)\right\}\end{aligned} \tag{4.19}$$

where the disciplinary contributions to the total algebraic takes the form of the sum of inner products between the disciplinary adjoint variables and the corresponding nonzero residuals on the coarse temporal domain. The disciplinary residuals are nonzero in the case of algebraic error due to the partial convergence of the governing equations. As in the case of the temporal discretization error, the algebraic error can be further split into the sum of inner products between the adjoint variable and the nonzero residual at each time-step in the temporal domain, thus providing a distribution of the error which can be used for adaptation.

4.2 Total Error and Separation of Error Components

At this point, the derivation makes it possible to independently compute the temporal discretization error and the algebraic error for multidisciplinary coupled equations. Additionally, each type of error can further be separated into components from each discipline and within each discipline separated into contributions from each time-step. It should also be noted that, in addition to breaking down the error components all the way to each time-step, one further possibility exists. Since both the adjoint variable and the residual span the temporal and spatial domains, it is possible to expand the inner product between the adjoint variable and the residual at each time-step in the spatial domain. That is to say, the inner product between the adjoint and the residual at each time-step is actually a sum of inner products between the same over all of the elements representing the spatial domain. This provides a spatial distribution of the temporal discretization error at each time-step in the time domain. It is in theory possible to adapt the time-step size for each spatial element based on the distribution of the estimated temporal error in the spatial domain. This however creates discontinuities where neighboring spatial elements may be at different temporal locations thus creating difficulties in computing discretely conservative fluxes. This is a problem that is yet to be addressed and remains beyond the scope of this thesis. The current work is limited to adaptation of the temporal domain equally for all spatial elements.

In the case of the error due to temporal discretization, the adjoint variables were obtained on the coarse temporal domain using the fully converged coarse time domain solution $\mathcal{U}_H$. The coarse adjoint variables and the fully converged coarse solution set were then projected onto the fine temporal domain, where the projected adjoint variables were multiplied with the corresponding nonzero residuals. The residuals on the fine temporal domain were nonzero since the coarse temporal domain solutions projected onto the fine time domain do not satisfy the fine temporal domain residual equations. In the case of algebraic error, there are no projections involved and all operations are performed on the coarse temporal domain. The coarse residuals are nonzero since a partial solution of the governing equations on the coarse temporal domain do not satisfy the coarse residual equations. Relaxing the assumption of a fully converged coarse time domain solution for the temporal discretization error does not affect the error estimated by that procedure. The important realization is that the error computed as temporal discretization error by relaxing this assumption includes the error due to the partial convergence of the governing equations on the coarse temporal domain. The temporal discretization error in reality is blind to what solution is projected from the coarse time domain, since the only assumption in the derivation is that $\mathcal{R}_h(\mathcal{U}_h^H) \neq 0$, which may also occur as $\mathcal{R}_h(\overline{\mathcal{U}}_h^H) \neq 0$. Projecting a partially converged solution from the coarse to the fine temporal domain includes the error due to the projection and the error due to partial convergence. The error due to temporal discretization can be separated from the total error by subtraction of the error due to partial convergence computed on the coarse time domain as:

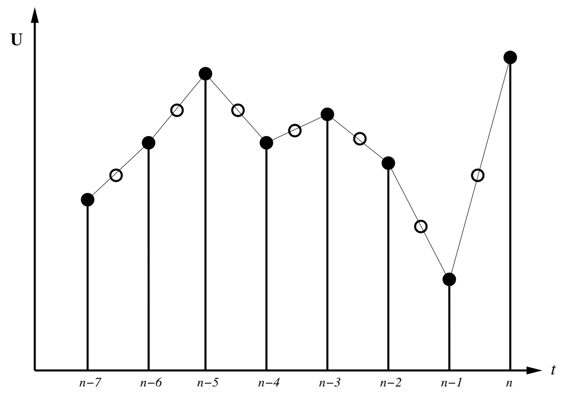
$$\begin{aligned} \varepsilon_{disc} &= \varepsilon_{total} - \varepsilon_{alg} \\ &= \left(I_h^H \Lambda_{\overline{\mathcal{U}}_H}^T \right) \mathcal{R}_h(\overline{\mathcal{U}}_h^H) - \Lambda_{\overline{\mathcal{U}}_H}^T \mathcal{R}_H(\overline{\mathcal{U}}_H) \end{aligned} \tag{4.20}$$

4.3 Error Estimates in the Unsteady Aeroelastic and Unsteady Uncoupled Flow Problems

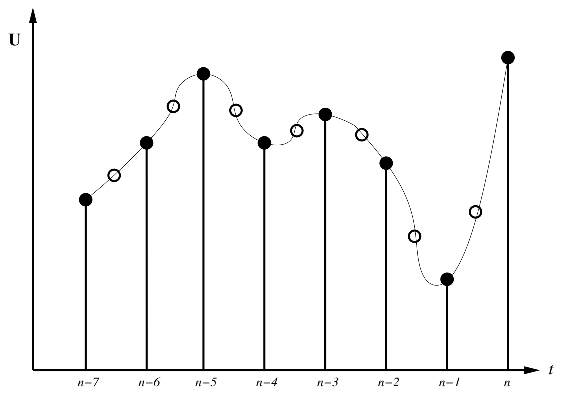
The estimation of temporal discretization and algebraic errors in the unsteady aeroelastic or the uncoupled unsteady problem follows the same procedure outlined up to this point. The main component is the computation of the adjoint variables in either problem for the functional of interest. Once the adjoint variables are known, the necessary residuals, either due to solution projection onto a finer temporal mesh or due to partial convergence are computed and weighted with the adjoint variable to determine the relevant error components.

4.4 Error Estimation Solver Implementation Details

The adaptive solver is essentially the same as the optimization solver consisting of two parts namely the forward flow solver and backward adjoint solver. The projection and error computation routines are built into the adjoint solver. The flow solver performs the forward integration in time and writes out the flow solution, mesh solution and partially converged residual values to the hard drive in a piecewise manner (i.e. at each time-step). The adjoint solver reads in the flow and mesh solutions and performs the backward sweep in time to compute the flow and adjoint variables at each time level. The adjoint solver also reads in the nonzero partially converged flow and mesh residuals written out by the flow solver and multiplies them with the corresponding adjoint variables to determine the algebraic partial convergence error at each time-step. A simple linear interpolation of the flow solution, mesh coordinates and the adjoint variables on the coarse time domain is used as the projection operator to determine unknown values on the fine time domain. A refinement ratio of 2-to-1 is used where each temporal element in the coarse level temporal mesh is divided into two. Although more advanced interpolation techniques such as third- and fifth-order splines may be used as the projection operator, it has been determined through testing that they do not offer any significant advantages in terms of gain in accuracy compared to a simpler linear interpolation. This is a useful finding since a linear interpolation is much easier to construct and there are fewer file I/O operations and flop operations resulting in faster code. Figure 4.1 clarifies the projection of coarse temporal domain variables onto the fine temporal domain using either a linear or cubic-spline interpolant. Further discussion regarding different projection operators and the advantage of using linear interpolation is presented toward the end of the following chapter. The adjoint solver also constructs the nonzero residuals resulting from the projected solutions from the coarse to the fine time domains and multiplies them with the corresponding projected adjoint variables to determine the total error (discretization+algebraic) at each time level. The algebraic or partial convergence error at each time-step is then subtracted out of the total error in order to obtain the temporal discretization error.



(a) Linear



(b) Spline

Figure 4.1: Linear and 3^{rd}-order spline interpolation of temporal data from coarse to fine temporal domains. The solid points indicate coarse level temporal elements, while the clear points indicate the division of the coarse level elements with a ratio of 2-to-1 in order to create a fine temporal domain with twice the resolution.

4.5 Adaptation Strategy

The adjoint based adaptation method involves the forward integration in time to determine the solution to the analysis problem and a backward sweep in time to compute the time-dependent adjoint variables and the corresponding necessary error distributions (discretization and algebraic). The sum of the individual error distributions provides an estimate of the correction from each type of error. The temporal elements are first sorted in decreasing order by their contribution to the total error of each type. Then parsing down each of the lists, elements are flagged for refinement or tolerance tightening until 99% of the total error of the type under consideration is reached. The method assures that only the most offending elements are targeted in the case of an extremely non-uniform error distribution. The method also assures that near uniform refinement or tolerance tightening is requested once the error in the temporal domain has been equidistributed. Elements flagged for refinement are subdivided into two, while elements flagged for algebraic error have their convergence tolerances tightened by a factor of three.

Chapter 5

Validation

5.1 Validation of the Analysis Solver

The analysis solver consists of the steady-state solver, the unsteady solver with prescribed motion and the aeroelastic solver with coupling between the flow, mesh and structural equations. The validation of the overall analysis solver must therefore be performed in steps beginning with the steady-state solver. The steady-state solver only depends on the spatial discretization of the governing equations, and therefore can be used to validate all spatial aspects of the code such as spatial order-of-accuracy and predicted steady-state loads. Once the spatial discretization has been validated, the unsteady solver can be utilized to assure that the temporal discretization follows expected trends and that the predicted time-dependent loads match results from other well established codes. Since the current work is based on the inviscid Euler equations, it is not possible to validate predicted loads directly with experimental data. The final step is to validate the aeroelastic solver following the same principle of comparing against other established numerical results.

5.1.1 Steady-State Solver Validation

Design Accuracy of the Spatial Discretization

The spatial fluxes constructed using the matrix dissipation scheme are free of mesh metric terms since they use the undivided Laplacians of the flow state variables. Although it was mentioned that this offers second-order spatial accuracy, this is strictly true only on uniform meshes. However, for a high quality unstructured inviscid mesh, there usually are no abrupt changes in the sizes of neighboring spatial elements, and the matrix dissipation scheme should offer near design accuracy. The accuracy convergence with respect to uniform spatial refinement was studied using the problem of inviscid flow over a Gaussian bump with a freestream Mach number of 0.2. It is known from theory that such a problem is shock free and that there is no other mechanism for the generation of entropy with the exception of spatial discretization error. Measurement of the global entropy L_2 norm defined as

$$L_2|ds| = \| \, s - s_\infty \, \|_2 \tag{5.1}$$

offers a convenient measure of the spatial discretization error in the solution. An initial mesh consisting of 1215 elements shown in Figure 5.1 was used as the starting point for the convergence study, where subsequent higher resolution spatial meshes were constructed through nested subdivision of the coarse elements. A total of five meshes with 1215, 4860, 19440, 77760 and 311040 elements were used in the study. Figures 5.2 and 5.3 show the computed density and change in entropy contours for problem solved on a mesh with 77760 elements. The appearance of a boundary layer and a wake in the entropy plot indicates the presence of spatial discretization error. Figure 5.4 shows the reduction in the global entropy norm with increase in spatial resolution for both a first-order and second-order spatial discretization. Observing the asymptotic slopes of the convergence curves in the figure, it can be seen that near design accuracy is achieved in both cases.

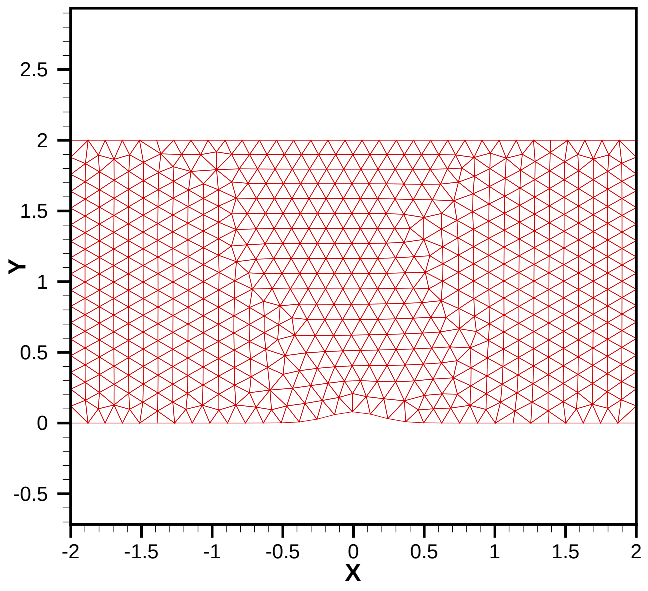


Figure 5.1: Computational mesh consisting of 1215 elements for the Gaussian bump flow problem used to validate spatial accuracy of the analysis solver.

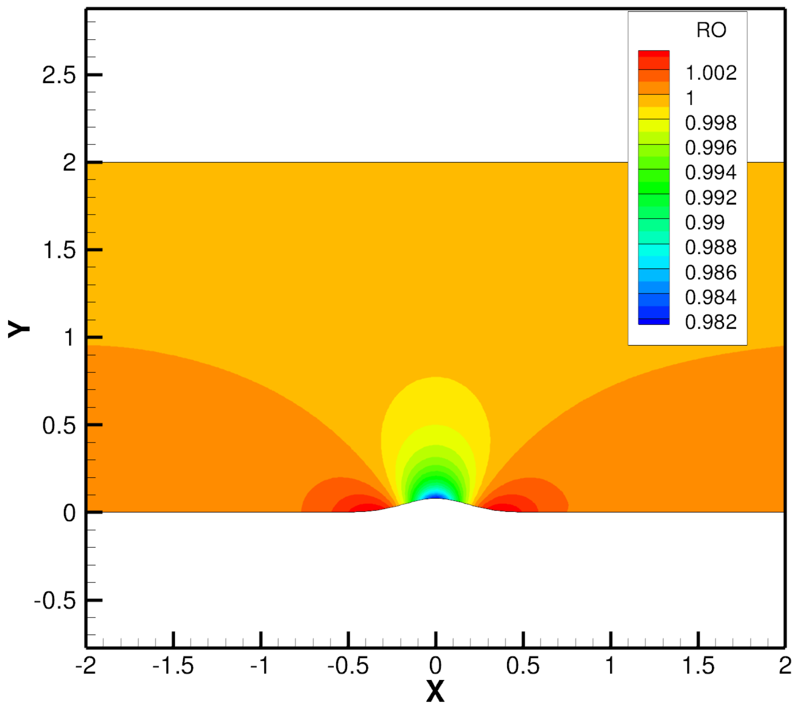


Figure 5.2: Density contours for the Gaussian bump problem at a Mach number of 0.2 solved on a mesh consisting of 77760 elements with a second-order accurate spatial discretization.

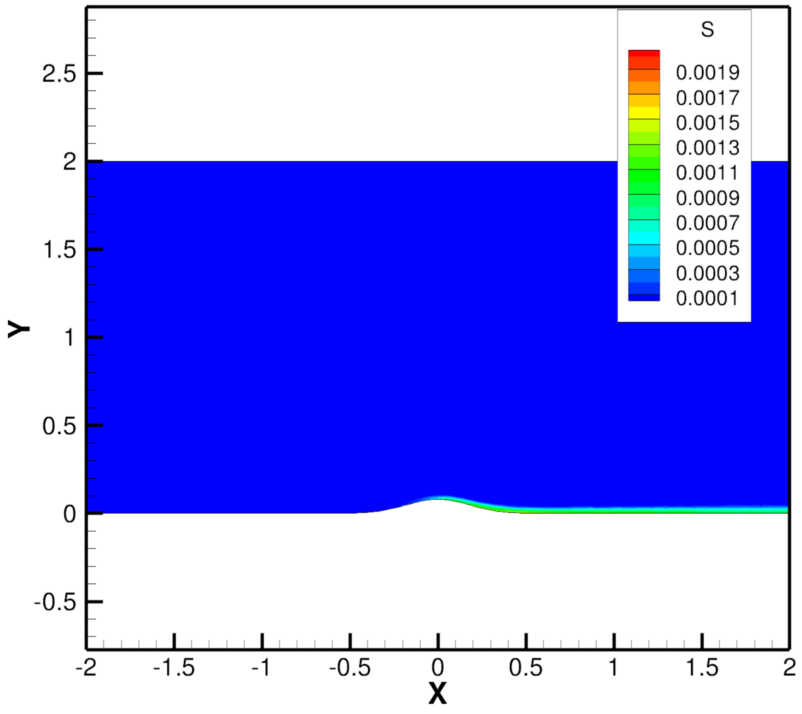


Figure 5.3: Contours of entropy generated in the Gaussian bump problem at a Mach number of 0.2 solved on a mesh consisting of 77760 elements with a second-order accurate spatial discretization.

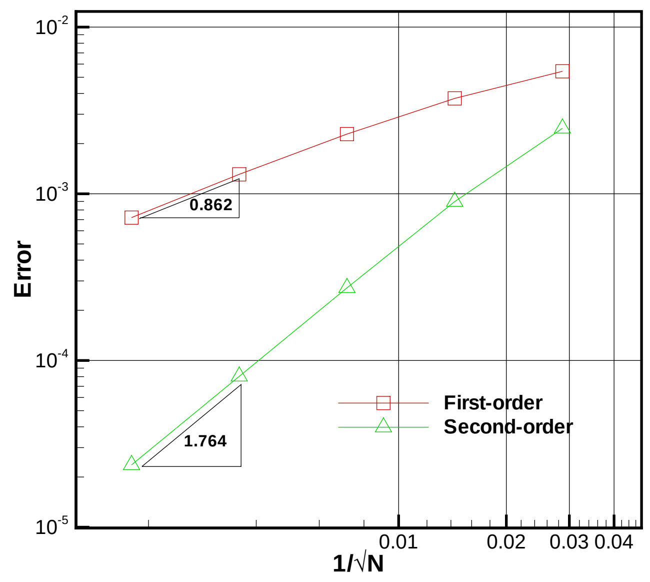


Figure 5.4: Verification of second-order spatial accuracy of the matrix dissipation scheme using the Gaussian bump entropy test problem.

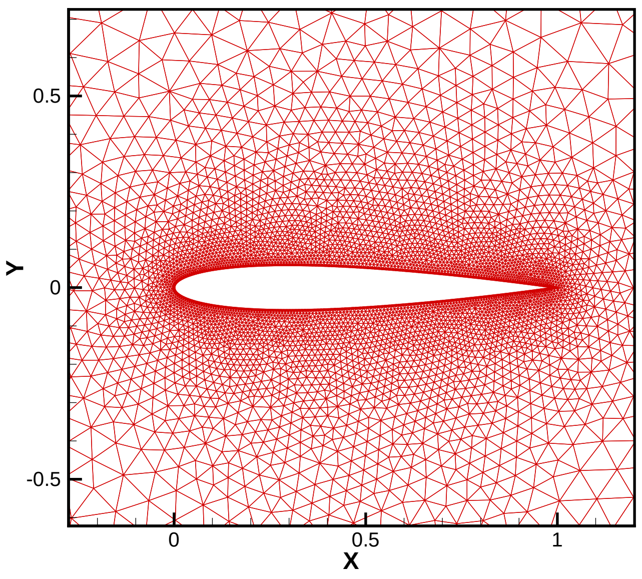


Figure 5.5: NACA0012 mesh consisting of approximately 10,000 elements used for the validation of steady-state loads, temporal discretization accuracy, and unsteady time-dependent loads.

Predicted Steady-State Loads

The case of transonic flow over a NACA0012 airfoil operating at a Mach number of 0.8 and an angle-of-attack of 1.25^o was chosen as the test case for validating predicted steady-state loads. The resulting force coefficients (lift, drag and moment) were compared against results from the existing in-house solver, NSU2D [73], a well established CFD code routinely used by the aircraft industry. Figure 5.5 shows the computational mesh consisting of approximately 10,000 elements used for the steady-state and subsequent unsteady load validation cases. Table 5.1 compares the results from the developed solver against values from NSU2D and indicates excellent agreement in the integrated coefficients. Figure 5.6 compares the pressure coefficient distributions from both solvers, showing good agreement.

Load Coefficient	**Developed Solver**	**NSU2D**
C_L	0.3385	0.3375
C_D	0.0217	0.0218
C_M	-0.0357	-0.0358

Table 5.1: Comparison of steady-state loads computed by current solver and NSU2D for transonic flow over a NACA0012 airfoil at a Mach number of 0.8 and an angle-of-attack of 1.25^o. The computational mesh consisted of approximately 10,000 elements.

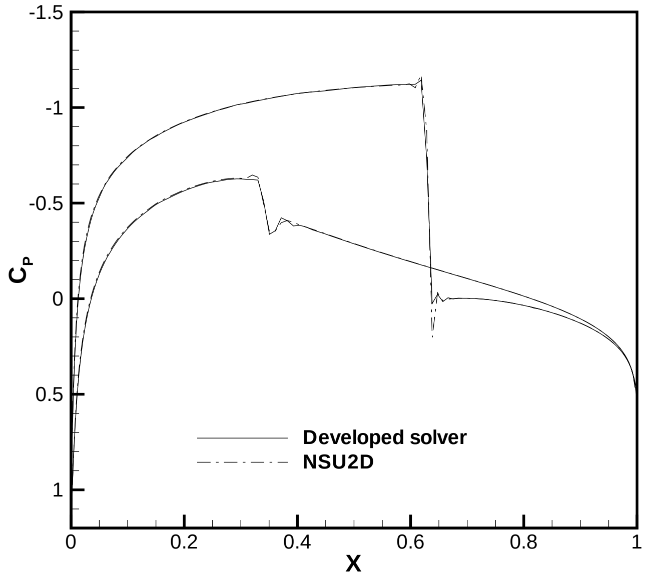


Figure 5.6: Comparison of computed pressure coefficient distribution between the developed solver and NSU2D. Both solutions were obtained on the same computational mesh around a NACA0012 airfoil with a freestream Mach number of 0.8 and an angle-of-attack of 1.25^o.

5.1.2 Unsteady Flow Solver Validation

Having validated the spatial discretization accuracy and the predicted steady-state loads, the next step is to validate the temporal discretization accuracy and the predicted time-dependent loads.

Design Accuracy of the Temporal Discretization

The method for validating the temporal accuracy of the solver involves obtaining time-dependent solutions on temporal meshes of different resolutions (i.e. different number of time-steps to represent the same time domain) and comparing the solution at the end of the time-integration process against a reference solution. The reference solution is obtained by solving the same time dependent problem on a temporal mesh with at least one order-of-magnitude more number of time-steps than the finest temporal mesh used in the study. Beginning with a coarse temporal mesh consisting of 16 time-steps, successive higher resolution meshes were constructed by dividing each time-step by two. The L_2 norm of the difference between computed and reference solutions at the end of the time-integration was used as the measure of the temporal error. The BDF2 temporal discretization was modified for nonuniform mesh spacing, and therefore the temporal meshes used in the study must have nonuniform spacing. For this reason, the coarsest temporal mesh was constructed using an arbitrarily chosen uniform stretching factor of 1.01 and a total of 16 time-steps. Higher resolution meshes were constructed by uniformly refining the coarsest mesh using a ratio of 2-to-1. The time domain under consideration ranged from 0 to 10 in non-dimensional time and the time dependent flow problem involved the sinusoidal pitching of a NACA0012 airfoil operating in a freestream Mach number of $M_\infty = 0.6$. The sinusoidal time variation of the angle-of-attack is given by the formula

$$\alpha = \alpha_0 + \alpha_{max} sin(2k_c t) \tag{5.2}$$

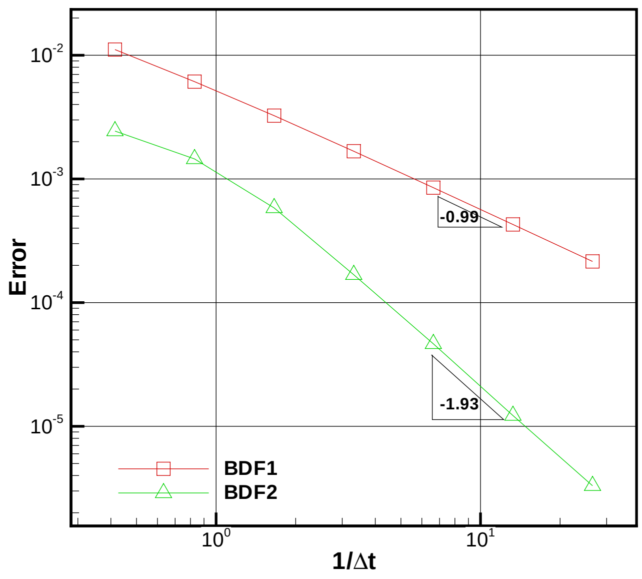


Figure 5.7: Verification of second-order temporal accuracy of the BDF2 scheme modified for nonuniform temporal mesh spacing in the presence of mesh deformation.

where the pitch parameters consisted of a reduced frequency of pitch of $k_c = 0.05$, an amplitude of pitch of $\alpha_{max} = 5^o$, and a mean angle-of-attack of $\alpha_0 = 0^o$ with the center of pitch located at the quarter-chord of the airfoil. Figure 5.7 shows the result of the study and indicates near design accuracy of the temporal discretization on nonuniform meshes with mesh deformation.

Time-Dependent Load Predictions

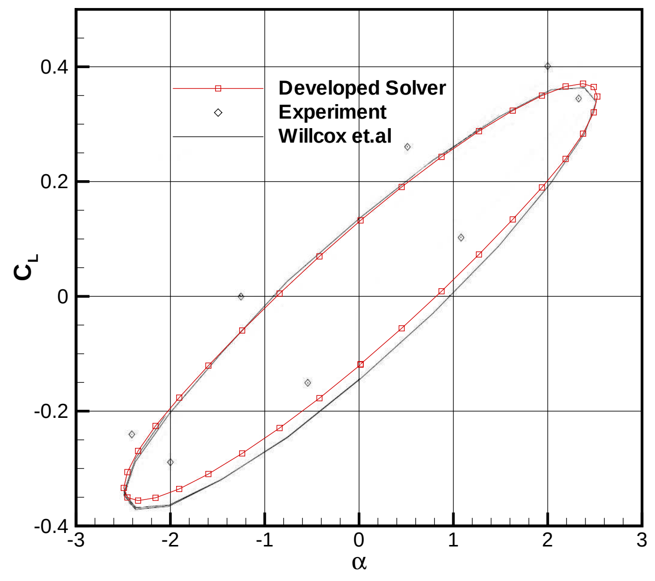
The problem chosen for the verification of time dependent loads is the AGARD test case No.5 [74]. The problem involves a NACA0012 airfoil sinusoidally pitching at a transonic Mach number of $M_\infty = 0.755$. For AGARD test case No.5, the airfoil pitches around its quarter-chord with a mean angle-of-attack $\alpha_0 = 0.016^o$, pitch amplitude of $\alpha_{max} = 2.51^o$ at a reduced frequency of $k_c = 0.0814$. The computational mesh consisting of approximately 10,000 elements, shown in Figure 5.5, was used for the unsteady load validation. Figures 5.8(a) and 5.8(b) show the time varying lift and moment coefficients computed by the solver which match well with results from references [7, 70, 74, 75], thus establishing confidence in the non-aeroelastic unsteady aspect of the solver.

5.1.3 Coupled Aeroelastic Solver Validation

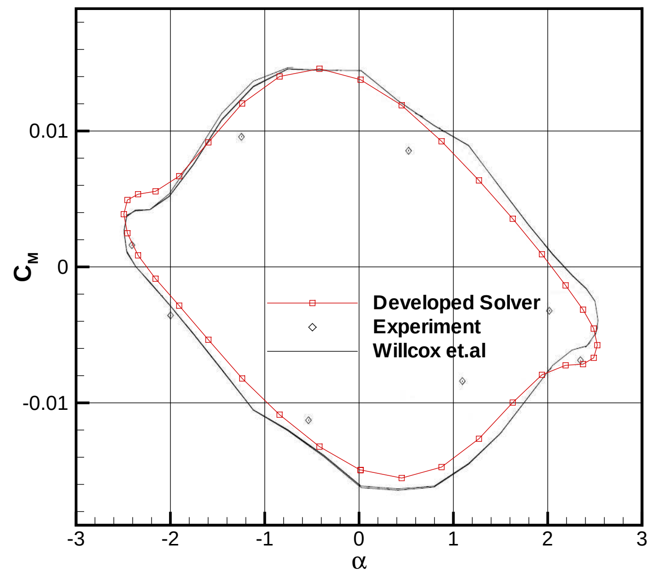
The final step is the validation of the aeroelastic aspect of the solver for which the two-dimensional swept wing model exhibiting the transonic dip phenomenon suggested by Isogai [76] was chosen as the test case. The structural parameters for this case are

$$\begin{aligned} x_\alpha &= 1.8 \\ r_\alpha^2 &= 3.48 \\ \omega_h, \omega_\alpha &= 100 \text{ rad/s} \\ \mu &= 60 \\ a &= -2.0 \end{aligned}$$

The quantity a is the non-dimensional elastic axis location along the chord of the airfoil measured from the midchord of the airfoil when it is in the neutral position. Since it is non-dimensionalized by the semi-chord of the airfoil, the elastic axis is located half a chord length ahead of the leading edge of the airfoil in this particular case. The airfoil under consideration is the NACA64A010 (Ames) airfoil operating with a mean angle-of-attack α_0 of zero and an amplitude of forced pitching α_{max} of 1^o. The computational mesh used for this case consists of approximately 6,600 elements and is shown in Figure 5.9. The solution process involves obtaining a steady-state solution at the mean angle-of-attack and then forcing the airfoil in pitch for three periods at the natural frequency of pitch before allowing it to respond aeroelastically. The goal of the validation procedure is to compute and compare a flutter boundary against existing data. Computation of the flutter boundary is performed by manually modifying the flutter velocity at various Mach numbers with the objective of



(a) Time variation of lift coefficient for unsteady solver validation



(b) Time variation of moment coefficient for unsteady solver validation

Figure 5.8: Unsteady load coefficients computed for the transonic pitching NACA0012 airfoil described in AGARD test case No.5.

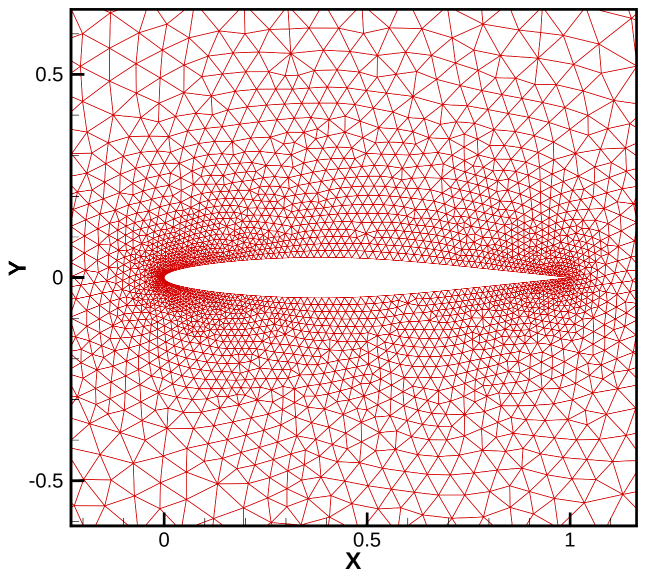


Figure 5.9: NACA64A010 mesh consisting of approximately 6,600 elements used for the aeroelastic solver validation. The same mesh is also used in the first aeroelastic optimization example and the first temporal adaptation examples.

obtaining a neutral aeroelastic response. Although the aeroelastic solver was not directly validated against experimental data, the predicted flutter boundary matches well with computational results from references [70, 75] as indicated by the flutter diagram in Figure 5.10. The aeroelastic response of the airfoil in both pitch and plunge degrees-of-freedom at various locations (stable and unstable) in the flutter diagram are shown in Figure 5.11 and follow expected trends.

5.2 Validation of Adjoint-Based Gradients

The procedure to validate the adjoint based sensitivity is twofold. The forward linearization is typically more intuitive to construct since it follows a framework very similar to that of the analysis code (i.e. forward in time). Issues with the discrete derivatives of the different pieces of the code contributing to the total linearization can

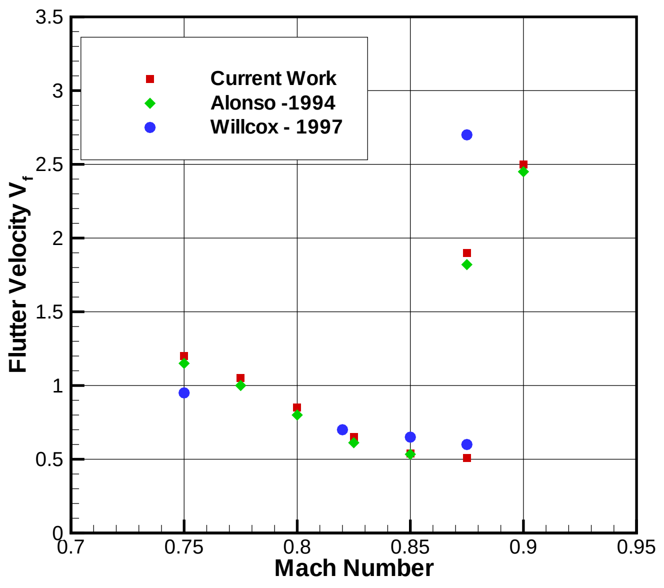


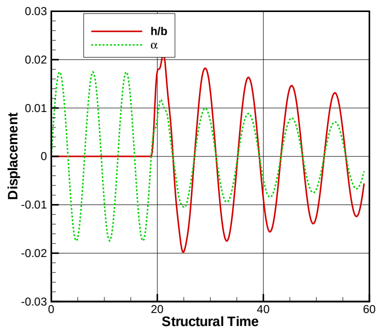
Figure 5.10: Comparison of the predicted flutter boundary against data from other references.

be tracked down far more easily using the forward linearization than with the adjoint or backward linearization. For this reason, the forward linearization is first verified against finite-difference values, and then the adjoint linearization is validated by verifying the property of dual consistency [29] with respect to the forward linearization. The principle of duality in the case of transpose operations can be summarized by the following example. Consider a primal operation of a matrix operating on a vector and the transposed dual of the same operation, both given as

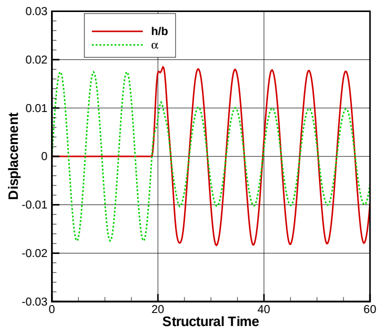
$$\begin{aligned}[\mathbf{A}]\xi &= \mathbf{f} \\ [\mathbf{A}]^T\eta &= \mathbf{g}\end{aligned} \tag{5.3}$$

where the vectors ξ and η are arbitrary. The principle of duality establishes equivalence between the primal and dual operations as

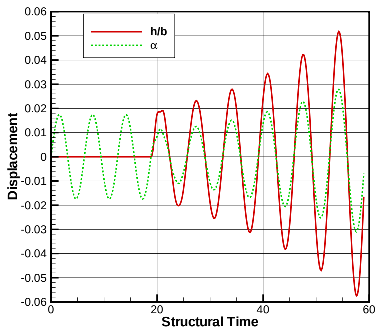
$$\xi^T\mathbf{g} = \eta^T\mathbf{f} \tag{5.4}$$



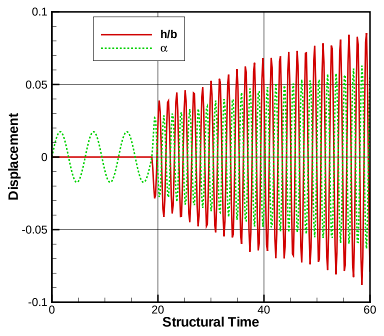
(a) Damped response ($M_\infty = 0.82, V_f = 0.4$)



(b) Neutral response ($M_\infty = 0.82, V_f = 0.71$)



(c) Divergent response ($M_\infty = 0.85, V_f = 0.8$)



(d) 2^{nd} mode response ($M_\infty = 0.875, V_f = 2.6$)

Figure 5.11: Aeroelastic response of NACA64A010 airfoil at various points in the flutter diagram for Isogai's test case exhibiting the transonic dip phenomenon.

Sections of the code where transpose operations are performed are validated by using arbitrary input vectors ξ and η to determine the corresponding outputs $\mathbf{f}$ and $\mathbf{g}$ and establishing that equation (5.4) is satisfied. For correctly transposed operations, equation (5.4) should match to machine precision. Although duality using a particular set of input vectors is a necessary test to ensure accuracy of the transpose operation, it is by no means sufficient. As a consequence the duality test is performed using several random input vectors in order to validate the operation.

Once the issues with the general framework of the adjoint linearization have been worked out in this manner, minor modifications to the analysis code can typically be directly carried over to the adjoint code and verified directly against finite-difference. The physical aspects of the test problem are not relevant for validating the adjoint-based gradient. It is more important that all aspects of the analysis solver which have been linearized to develop the sensitivity solver be verified for accuracy. For this reason, the aeroelastic test case presented earlier, but using only 5 uniform time-steps of size 0.1 for the forced pitching and 5 additional time-steps of the same size for the aeroelastic response, was used. The design variables for the test case formed a vector of weights controlling the magnitude of bump functions placed at every single surface node location. Table 5.2 compares adjoint-based and finite-differenced sensitivity values for a few design variables, while Figure 5.12 shows the comparison over all design variables. The adjoint-based sensitivity values typically match to a minimum of 4 digits against finite-difference values as indicated by Table 5.2, and are virtually indistinguishable from one another when overlaid on a plot as shown in Figure 5.12.

5.3 Validation of Adjoint-Based Error Estimates

The final step in the validation process before the methodology can be applied to example problems is that of verifying the accuracy of adjoint-based error estimates. The case of uncoupled unsteady disciplines with only the flow and mesh equations

Design Variable ID	Adjoint Sensitivity	Finite-difference
183	**6.368**45285306436	**6.368**25832067700
184	**6.79732**2288068724	**6.79734**654118747
185	**7.490**93700072817	**7.490**87240015101
186	**8.831**56670890565	**8.831**24061656915
187	**12.261**2224955801	**12.261**3669883975
188	**19.422**2465094799	**19.422**4766714157

Table 5.2: Comparison of the computed adjoint sensitivities against values obtained through finite-difference. Matching digits between the two are shown in bold font.

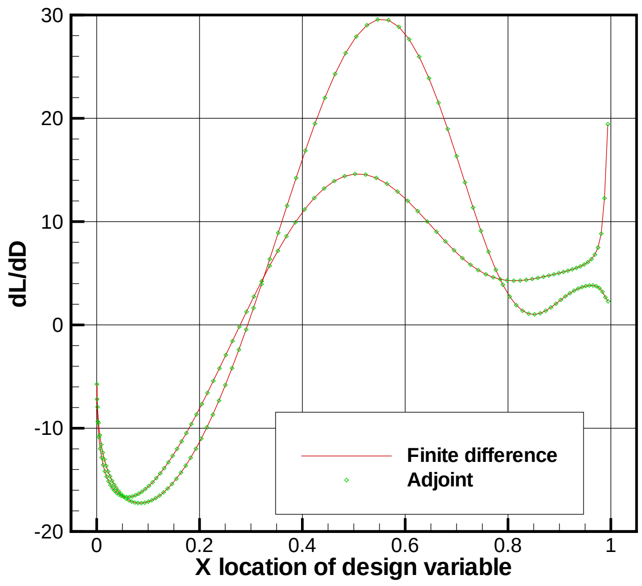


Figure 5.12: Comparison of gradients computed by the adjoint linearization against finite-difference based values.

along with prescribed motion in time of an airfoil is used for this purpose.

5.3.1 Validation of Temporal Discretization Error Estimate

Since the method developed in the current work computes a linear approximation of the error between temporal meshes successively refined using a ratio of 2-to-1, it would be wise to choose a test problem that exhibits minimum nonlinear behavior with respect to increasing temporal resolution for the purpose of validation. The approach is to consider a sinusoidally pitching NACA64A010 airfoil and zoom into a small temporal window in the pitch cycle to use as the time domain of interest. The purpose of this is to reduce the effect of nonlinear behavior as the temporal resolution is increased. The conditions chosen for the problem include a freestream Mach number of 0.8, a zero mean angle-of-attack, an amplitude of pitch of 5^o, a reduced frequency of 0.1 and a pitch center located two chord lengths ahead of the leading edge of the airfoil. The time domain considered is the first quarter of the first pitch period starting from a steady-state solution. The functional of interest is the time-integrated lift coefficient given as

$$L = \sum_{n=1}^{n_{steps}} \Delta t^n {C_L}^n \tag{5.5}$$

The coarsest temporal mesh consists of 8 time-steps of uniform size and fine meshes are constructed by successive nested subdivision of these 8 elements. The finest mesh consists of 128 elements. In order to remove the effect of partially converged solutions and to validate only the error due to temporal resolution, all of the equations (governing and corresponding adjoint) are converged to machine precision. At each temporal resolution the comparison is made against the exact functional computed directly on the next temporal mesh that has twice the resolution (i.e. twice the number of time-steps with half the time-step size). Table 5.3 shows the results from this study and indicates the expected trends. As the temporal resolution is increased, the ratio between the predicted and exact errors asymptotically approaches unity.

Flow/Mesh Convergence Tolerances	**H/h**	$L_h(\mathbf{U}_h, \mathbf{x}_h)$	$L_h(\mathbf{U}_h^H, \mathbf{x}_h^H)$	**Exact Error**	**Predicted Error**	**Ratio**
2e-14/1e-15	8/16	4.7627748	4.7419952	0.02077960	0.02222993	1.06979621
2e-14/1e-15	16/32	4.6769170	4.6602314	0.01668557	0.01616314	0.96868998
2e-14/1e-15	32/64	4.6352466	4.6257924	0.00945421	0.00945039	0.99959626
2e-14/1e-15	64/128	4.6149705	4.6097293	0.00524124	0.00524024	0.99980920

Table 5.3: Validation of the predicted temporal discretization error without the effect of algebraic error

Flow/Mesh Convergence Tolerance	**H**	$L_H(\mathbf{U}_H, \mathbf{x}_H)$	$L_H(\bar{\mathbf{U}}_H, \mathbf{x}_H)$	**Exact Error**	**Predicted Error**	**Ratio**
1e-5/1e-15	8	4.9477907	4.9335132	0.0142775	0.0143015	0.9983222

Table 5.4: Validation of the algebraic error due to the partial convergence of the flow equations. All computations are performed with the same temporal mesh resolution along with full convergence of the mesh equations.

5.3.2 Validation of Algebraic Error Estimate

Flow Partial Convergence Error

In order to validate the error only due to partial convergence of the flow, the mesh motion equations are fully converged and the flow equations partially converged (to a tolerance of $1e^{-5}$) while performing all computations on a single temporal mesh of some arbitrary resolution. This is done in order to remove the effect of mesh partial convergence and temporal resolution. The chosen temporal mesh resolution consists of 8 uniform time steps. The comparison is made against a functional computed on the same temporal mesh by fully converging both the flow and mesh equations. Table 5.4 shows the results from the study. The predicted error due to partial convergence is very close to the exact error as indicated by the near unity of the ratio between the two.

Flow/Mesh Convergence Tolerance	**H**	$L_H(\mathbf{U}_H, \mathbf{x}_H)$	$L_H(\mathbf{U}_H, \bar{\mathbf{x}}_H)$	**Exact Error**	**Predicted Error**	**Ratio**
2e-14/1e-5	8	4.9477907	4.9477703	2.0380075e-5	2.0631192e-5	1.0123217

Table 5.5: Validation of the algebraic error due to the partial convergence of the mesh equations. All computations are performed with the same temporal mesh resolution along with full convergence of the flow equations.

Mesh Partial Convergence Error

The method for validating the mesh partial convergence error is nearly identical to that of validating the flow partial convergence error, except now the flow equations are fully converged while the mesh equations are partially converged to a tolerance of $1e^{-5}$. Table 5.5 shows the results from the study and once again the near unity of the ratio between the predicted and exact errors establishes confidence in the method.

5.3.3 Validation of Combined Total Error Estimate

In this study the temporal resolution is successively refined and the convergence tolerances are tightened from loosely set values in order to validate the total predicted error. The coarsest temporal mesh consists of 2 equally sized temporal elements while the finest mesh consists of 128 elements. The convergence tolerances for the flow equations and mesh equations begin at $1e^{-5}$ and $1e^{-4}$ respectively on the coarsest temporal mesh and are tightened by an order-of-magnitude during each refinement of the temporal resolution. These starting limits were chosen such that the algebraic and resolution errors are roughly the same order-of-magnitude so as to not mask any discrepancies from either. The comparison at each temporal resolution is made against the functional computed using full convergence of flow and mesh equations on the next temporal mesh of double the resolution. It is important that the method be validated for the total combined error prediction since it is to be applied in this form to adaptation problems in general. Table 5.6 shows the results from this study

Flow/Mesh Convergence Tolerance	**H/h**	$L_h(\mathbf{U}_h, \mathbf{x}_h)$	$L_h(\bar{\mathbf{U}}_h^H, \bar{\mathbf{x}}_h^H)$	**Exact Error**	**Predicted Error**	**Ratio**
1e-5/1e-4	2/4	5.3287125	5.2361864	0.0925261	0.0897410	0.9699000
1e-6/1e-5	4/8	4.9477907	4.9142574	0.0335332	0.0364249	1.0862345
1e-7/1e-6	8/16	4.7627748	4.7419952	0.0207796	0.0222299	1.0697962
1e-8/1e-7	16/32	4.6769170	4.6602314	0.0166855	0.0161631	0.9686899
1e-9/1e-8	32/64	4.6352466	4.6257924	0.0094542	0.0094503	0.9995962
1e-10/1e-9	64/128	4.6149042	4.6097293	0.0051749	0.0051738	0.9997898

Table 5.6: Validation of the predicted total combined error which includes temporal discretization error and the algebraic error due to partial convergence of the governing equations.

and indicates the expected trend of more accurate predictions as the resolution is increased while simultaneously convergence tolerances are tightened.

5.4 Effect of Initial Condition and Change of Order During Startup

An unsteady solution procedure that involves a higher-order multi-step time-integration scheme typically starts up with a lower-order scheme. In the case of second-order temporal accuracy, the first time-step is normally treated using a first-order accurate scheme. This is due to the unavailability of a stencil large enough to accommodate the BDF2 scheme at the first time-step. Since the adjoint computation is based simply on the linearization of the governing flow equations, this change in order during startup gets carried over to the adjoint solution process. In most circumstances, adjoint variables vary smoothly throughout the entire time domain irrespective of the temporal resolution. The exception to this is when there is a change in the order-of-accuracy of the temporal discretization between consecutive temporal elements. Since this does happen at the very first time-step for the BDF2 scheme, there is a discontinuity in the adjoint variable distribution at the first time-step. The method for error estimation used in this work relies on computing the adjoint variables on a coarse temporal temporal mesh and then projecting these onto a finer temporal mesh rather than directly computing the adjoint variables on the finer temporal mesh. Although the Taylor expansion requires that the adjoint variables be computed directly on the fine temporal mesh, it is assumed that computing and projecting the adjoint variables from the coarser level will suffice for the purpose of error estimation. However, the discontinuity in the time variation of the adjoint variables due to change in discretization order occurs at different temporal locations in the coarse and fine meshes. In the case of 2-to-1 refinement between fine and coarse temporal meshes, the fine level discontinuity occurs at the midpoint of the first temporal element on the coarse level. Figure 5.13 clearly shows the difference in the location of the discontinuity when comparing adjoint variables computed on the fine and coarse meshes. The implication is that using a coarse level adjoint projected onto the fine level leads to significant inaccuracies in the predicted error. It is virtually impossible for a linear or any higher-order projection operator to shift the location of this spike when going from the coarse to the fine levels. As per theory, the correct location for the spike is that captured when the adjoint variables are computed directly on the fine level. Such a trend has also been observed in other work on spatial adjoint-based error estimates where the boundaries are to be treated with care due to fluctuations in the computed coarse mesh adjoint spatial distribution [77].

Two different methods, namely the application of a second-order accurate initial condition and the direct solution of the fine level adjoint at only the startup location were investigated to remedy this problem. The second-order initial condition is analogous to a second-order boundary condition in space. A ghost temporal point is introduced with a spacing equal to that of the first time-step on the coarse level and the explicit first-order backward Euler formula is used to determine the state at this point. Referring to Figure 5.14, the solution state vector at the ghost point can be

		Ratio against exact
Exact functional - 32 steps BDF2	0.0182487	-
Exact functional - 16 steps BDF2	0.0182499	-
Exact error between 32 and 16 time-steps	-1.1969778e-06	-
Using coarse level adjoint	-5.2181473e-07	0.43594
Using coarse level adjoint and 2nd-order initial condition	-1.1694744e-06	0.97702
Using coarse level adjoint + fine level adjoint for first step	-1.2120836e-06	1.01262
Using fine level adjoint at all time steps	-1.1949466e-06	0.99830

Table 5.7: Error due to change in temporal discretization order of accuracy during startup and mitigation using various treatment methods.

determined starting with a BDF1 temporal discretization at steady-state ($n = 0$) as

$$A^{(0)} \left(\frac{\mathbf{U}^{(0)} - \mathbf{U}^{(-1)}}{\Delta t_1} \right) + S = 0 \tag{5.6}$$

where the surface integral of spatial fluxes S, the element area at steady-state $A^{(0)}$ and the steady-state solution state $\mathbf{U}^{(0)}$ are known quantities, allowing the factorization of the ghost state $\mathbf{U}^{(-1)}$ as

$$\mathbf{U}^{(-1)} = \mathbf{U}^{(0)} + \frac{\Delta t_1}{A^{(0)}} S \tag{5.7}$$

Although the integral of the spatial fluxes S is zero at steady-state conditions ($n = 0$), the incorporation of mesh velocities due to the presence of the ghost temporal point converts this to a nonzero quantity. A change in the order of the temporal discretization within the time domain of interest is avoided since a stencil for the BDF2 scheme is now available at time-step $n = 1$. Table 5.7 shows the inaccuracy in the predicted error between two levels due to this issue and the gains in accuracy using the two different methods proposed. It is also evident from the table that both methods perform reasonably well. For the current work, the adjoint variable is computed directly on the fine level at only the first time-step. Although this is a solution on the fine level, the cost incurred in doing so corresponds exactly to that of two additional time-steps in the coarse level temporal mesh. As the temporal resolution increases, the cost of this additional computation becomes negligible.

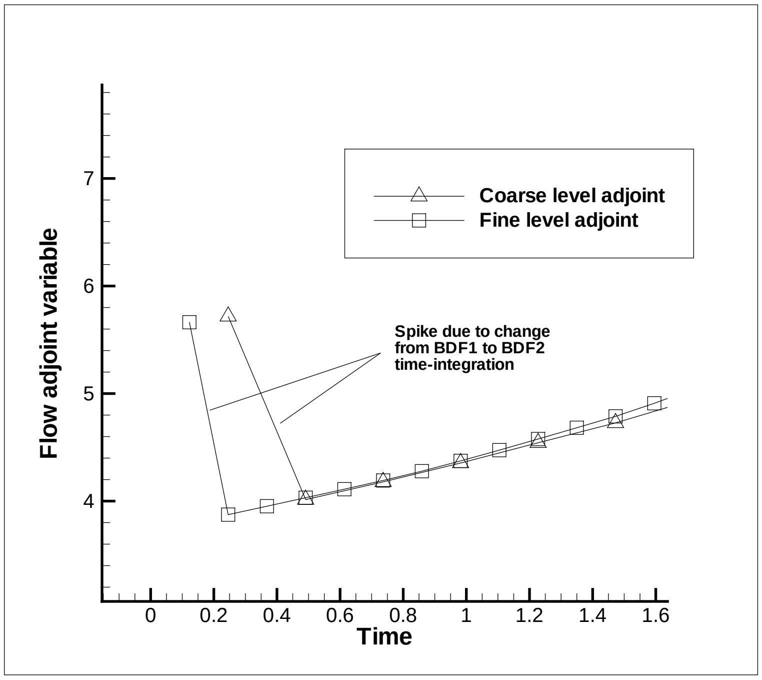


Figure 5.13: Comparison of the spike locations due to the change in the temporal discretization order of accuracy during startup between the adjoint variable computed on the coarse temporal domain and adjoint variable computed on the fine temporal domain of twice the resolution. (Shown here for an arbitrary spatial location).

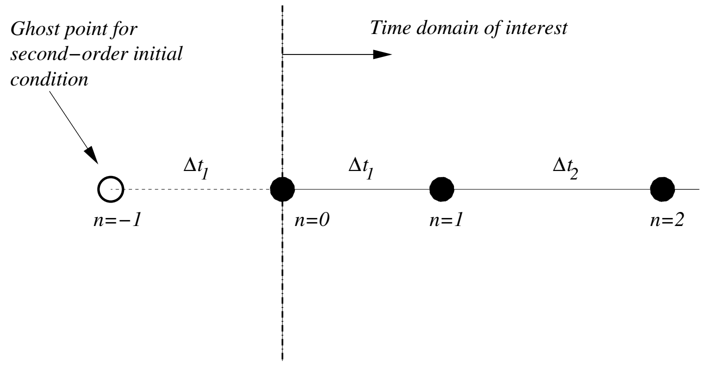


Figure 5.14: Introduction of a ghost temporal point outside the time domain of interest for the purpose of avoiding a change of temporal discretization order of accuracy within the time domain.

5.5 Shortcomings of Adjoint Variable Projection

5.5.1 Profile Dissimilarity between Coarse and Fine Adjoint Distributions

In the case of a first-order BDF1 time-integration scheme, the difference between coarse and fine level adjoint solutions is minimal allowing a linear interpolant to capture the details of the fine level using the coarse level adjoint time history. Increasing the order-of-accuracy of the temporal discretization however may create differences in the adjoint variable distribution between the two levels that cannot be captured effectively by any interpolating operator due to the lack of profile similarity between the coarse and fine level adjoint distributions. For highly nonlinear analysis problems, the profile dissimilarity is further exacerbated as the resolution of the coarse level temporal mesh increases. Figure 5.15 shows the difference in the time variation of the fine temporal domain adjoint and the coarse temporal domain projected (third-order spline interpolant) adjoint for an arbitrary spatial location between temporal resolutions of 32 and 16 time-steps. The test problem was based on the unsteady prescribed pitching of a NACA0012 airfoil, where the functional of interest was the lift coefficient at the end of the time-integration process. While the details of this particular test problem are not important, it should be noted that this problem has a tendency to surface mostly when dealing with non-time-integrated functionals. Two different methods were investigated to improve the accuracy of the adjoint variable as described below.

Utilizing Fine Level Information in Coarse Adjoint Solution

From a cost standpoint, it is important that fine level solutions of the adjoint equations be avoided, but solving only the coarse level adjoint equations and projecting the adjoint variable to the fine level leads to significant inaccuracies in the predicted

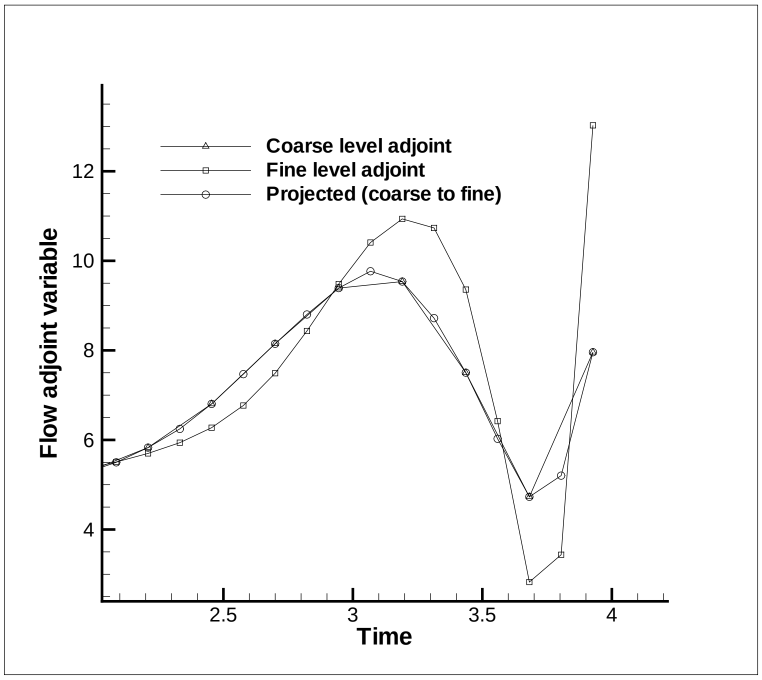


Figure 5.15: Difference in the time variation of the fine level, coarse level, and projected adjoint variables (arbitrary spatial location).

error under certain conditions. Accuracy gains while keeping costs similar to that of solving the coarse level adjoint can be achieved by attempting a solution of the fine level adjoint equations but only at points on the fine level temporal mesh that coincide with the coarse level mesh. The main difficulty is the construction of source terms for the adjoint equations from fine level temporal mesh points that do not coincide with those on the coarse level, since no adjoint variable is available at these locations. This problem can be addressed by combining the projection operation with the adjoint solution process. The unknown adjoint variables on the fine level temporal mesh are written as functions of known adjoint variables located at points that coincide with the coarse level mesh and solved simultaneously. As an example, when solving equation (3.57) on the fine temporal domain, but only at coarse level coincident points, the adjoint variable $\Lambda_{\mathbf{U}}^{n+1}$ is unknown. Referring to Figure 5.16, the unknown adjoint variable is written in terms of known adjoint variables as

$$\Lambda_{\mathbf{U}}^{n+1} = C_1\Lambda_{\mathbf{U}}^{n} + C_2\Lambda_{\mathbf{U}}^{n+2} + C_3\Lambda_{\mathbf{U}}^{n+3} + C_4\Lambda_{\mathbf{U}}^{n+4} \tag{5.8}$$

where the coefficients $C1, C2, C3$,and $C4$ are determined based on a cubic spline interpolation. Substituting the above expression for $\Lambda_{\mathbf{U}}^{n+1}$ into equation (3.57), the fine level adjoint variable $\Lambda_{\mathbf{U}}^{n}$ can be solved for without the explicit presence of $\Lambda_{\mathbf{U}}^{n+1}$. Both linear and third-order operators chosen to represent the unknown adjoint variables indicate improvements although the third-order spline operator fairs better. However, implementation of this method involves coding complexities that result in increased computational cost thus rendering it only marginally cheaper than actually solving the adjoint variable distribution directly at all fine level temporal mesh points. This method does significantly reduce the difference between a coarse level adjoint projected to the fine level and the true fine level adjoint solution since it does not completely ignore fine level information during the solution process on the coarse level. Figure 5.17 indicates significant improvement at least in the qualitative sense.

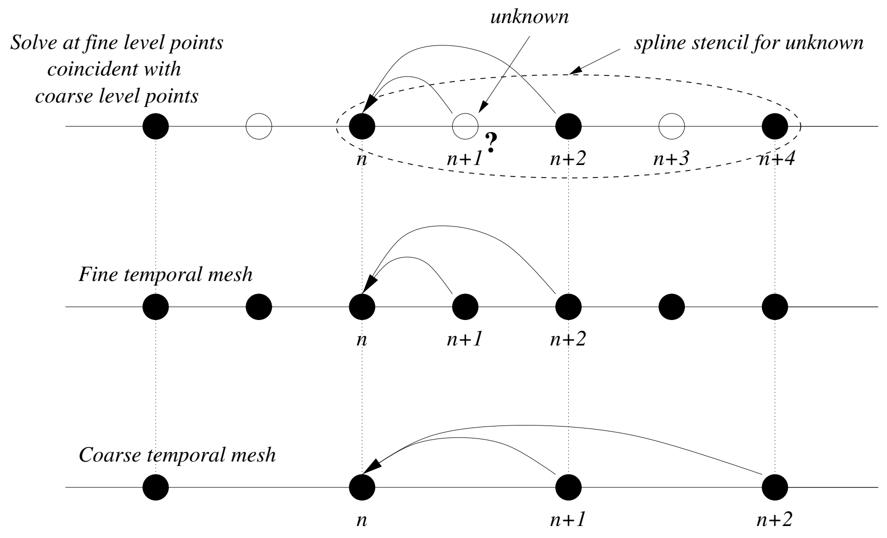

unknown
Solve at fine level points coincident with coarse level points
spline stencil for unknown
?
n
n+1
n+2
n+3
n+4
Fine temporal mesh
n
n+1
n+2
Coarse temporal mesh
n
n+1
n+2

Figure 5.16: Combination of projection and adjoint solution processes in order include fine level information during the adjoint solution process. The arrows indicate the time-steps from which source terms arise for the adjoint equation at time-step n.

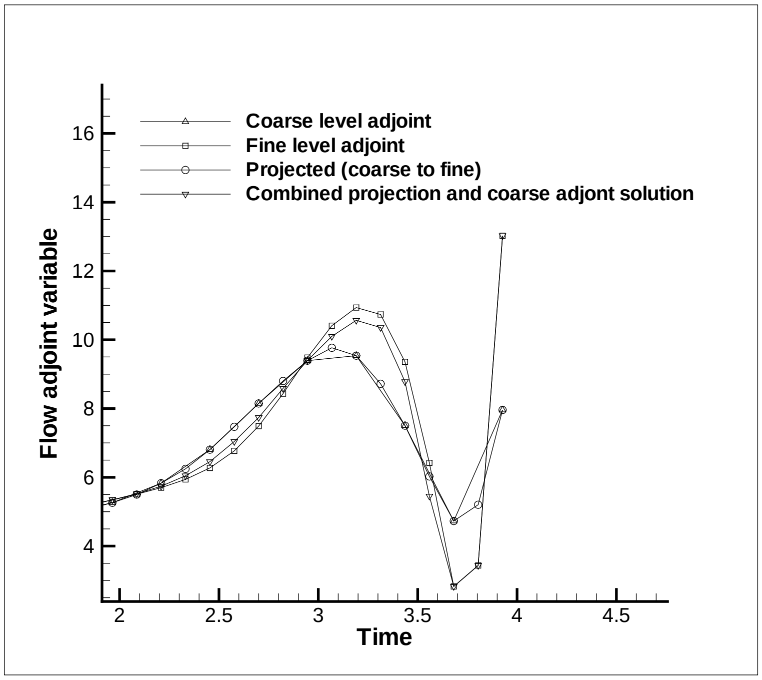


Figure 5.17: Simultaneous projection and solution of coarse level adjoint variable (arbitrary spatial location).

Direct Smoothing of Error in Adjoint Variable on Fine Level

While fine level solutions are to be avoided, the projection of the coarse level adjoint variable onto the fine level temporal mesh provides a good initial guess for the fine level adjoint equations. If the error between the coarse and fine level adjoint solutions are in the high frequency range in the context of the fine level temporal mesh, solution of the fine level adjoint equations starting with the coarse level estimates should in theory converge very rapidly using conventional smoothing techniques such as Gauss-Seidel iterations. Although this is an increase in the overall cost of the process and is actually a partial solution directly on the fine level temporal mesh, the gain in accuracy somewhat justifies the increase in cost. Current work indicates a typical increase anywhere between 20% to 30% in cost compared to solving only the coarse level equations.

General Note Regarding Projected Adjoint Variables

In general it is not necessary that the computed adjoint distribution match exactly with the true fine level adjoint distribution. Since the overall error estimation approach relies on a linearization, exact corrections between successively refined temporal meshes cannot be guaranteed. While it is desirable to obtain accurate corrections to the functional, for the method to be useful it is important that it be able to at least provide a good estimate of the error distribution relevant to our functional of interest, even if the magnitude of the total error is not accurate. Based on the proposed adaptation strategy, it is the relative error distribution amongst temporal elements in the domain that is used to drive adaptation rather than the magnitude of the error itself. The magnitude of the error manifests its importance when summed over all time-steps in the domain to obtain an estimate for the functional correction.

5.5.2 Projections of Discontinuous Solutions

Discontinuities in the solution present problems in the accurate estimation of errors. It has been shown that reconstruction of spatial solutions from coarse to fine spatial meshes while estimating spatial discretization error in the presence of shock waves or other discontinuities results in oscillations that contaminate the error estimate [78]. Discontinuities in the temporal variation of the solution can occur in problems involving moving shock waves or impulsive motion of geometries. The choice of the projection operator in such scenarios greatly influences the accuracy of the error estimates. It is important that monotone reconstruction procedures be employed in the reconstruction of the fine temporal domain solution using the coarse temporal domain solution regardless of the problem being solved. While the coarse temporal domain solution may exhibit smooth behavior, there is always the possibility that an increase in temporal resolution will lead to capturing of smaller scale temporal features that may exhibit highly nonlinear variations between the two levels. Figure 5.18 compares the linear and 3^{rd}-order spline based projections of a hypothetical discontinuous solution in time. It can be seen that the spline interpolant produces oscillations ahead and behind the discontinuity while the linear operator remains monotone.

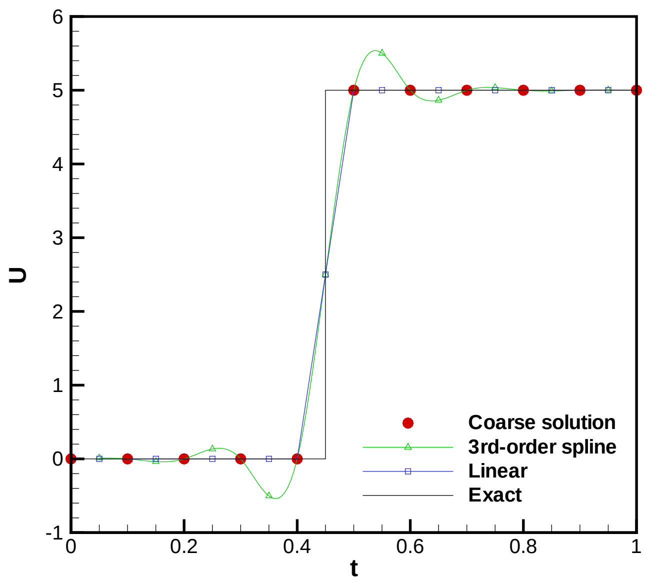


Figure 5.18: Comparison of projecting a discontinuous solution from the coarse temporal domain to a fine temporal domain with twice the resolution using a linear and a 3^{rd}-order spline interpolation operators.

Chapter 6

Results

The results of applying the adjoint-based gradients and the adjoint-based error estimates to examples problems are presented in this chapter. Firstly, the adjoint-based gradients are demonstrated for the purpose of shape optimization in the context of steady-state, uncoupled unsteady and coupled unsteady aeroelasticity problems. The second set of examples demonstrate the use of adjoint-based error estimates for adapting the temporal resolution and the convergence tolerances to reduce temporal discretization and algebraic errors.

6.1 Steady-State Lift Constrained Drag Minimization

The first optimization example is that of shape optimization using adjoint-based gradients for the reduction of drag on a transonic airfoil with a lift constraint. The airfoil under consideration for this problem is the NACA0012 and the flow parameters consist of a freestream Mach number of $M_\infty = 0.8$ and an angle-of-attack of $\alpha = 1.25^o$. Under these flow conditions, the NACA0012 airfoil exhibits a strong transonic shock wave on the upper surface at roughly 0.65 chord and a weak transonic shock wave on the lower surface at roughly 0.35 chord. The objective functional for the optimization problem is defined as

$$L = (C_L - C_{Ltarget})^2 + 10C_D^2 \tag{6.1}$$

where the target lift coefficient is that of the NACA0012 airfoil at the same flow conditions. The weight on the drag coefficient is introduced for the purpose of equalizing the typical order-of-magnitude difference between the lift and drag coefficients. The functional is formulated such that any decrease in the functional is due principally to reductions in the value of the drag coefficient. The lift coefficient term provides an excellent constraint since any deviations from the target lift coefficient lead to an increase in the value of the functional. A total of 32 bump functions, 16 on the upper surface and 16 on the lower surface with equal spacing in the chordwise direction of the airfoil and the corresponding weights controlling the magnitudes of these bumps form the set of design variables for the problem. The LBFGS-B optimization algorithm described earlier is used for the current and subsequent optimization examples.

Figure 6.1 shows the computational mesh around the NACA0012 airfoil consisting of approximately 20,000 elements used in this optimization example. Figure 6.2 shows the convergence of the optimization problem. There is at least an order-of-magnitude reduction in the functional and in the norm of the gradient vector after just eight design iterations, however the convergence flattens out beyond this. It is not possible for the functional to converge to zero even if the shock wave is eliminated due to dissipative errors in the solution. The gradient vector however should in theory reach zero when the optimization reaches a local minimum. This however is not the case for the optimization example presented here, since the design space is bounded *a priori* in order to prevent the occurrence of degenerate geometries. Figure 6.3 illustrates nonzero gradients at the determined optimum when bounds on the design space exist in the optimization problem. It was determined at the termination of the optimization that several design variables were indeed on the established boundary of the design space. Figure 6.4 shows the convergence of the lift and drag coefficients

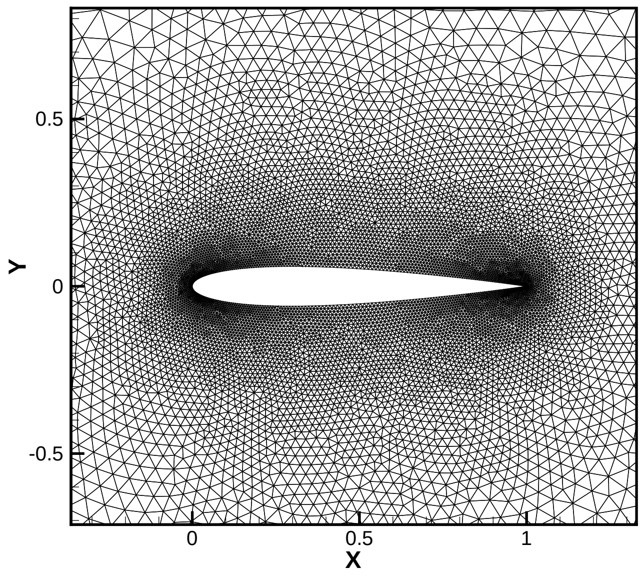


Figure 6.1: Computational mesh of approximately 20,000 elements used in steady-state and unsteady optimization examples.

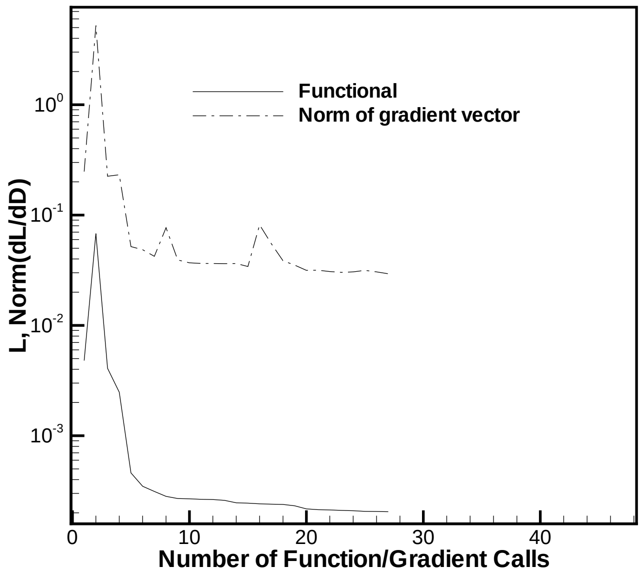


Figure 6.2: Convergence of the steady-state lift constrained drag minimization optimization example.

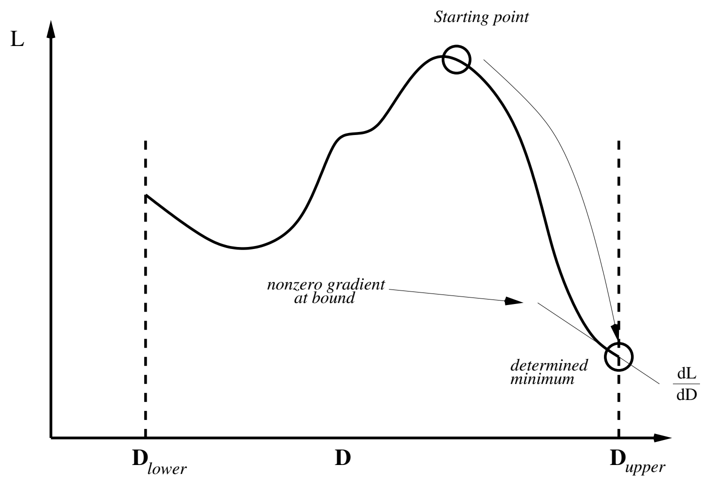


Figure 6.3: The existence of a nonzero gradient at the optimum design point determined by the optimization algorithm.

during the course of the optimization. An approximate reduction of 160 counts in the drag coefficient is seen while no change is seen in the lift coefficient. Figure 6.5 compares the baseline NACA0012 airfoil with the final optimized airfoil and it can be seen that the optimization has significantly flattened the upper surface in an attempt to weaken the upper surface shock wave. This is evident from the comparison of the C_p distribution between the NACA0012 and the optimized airfoils shown in Figure 6.6. The lower surface shock wave has been eliminated while the pressure jump across the upper shock wave is significantly smaller. The wave drag is the most significant component of the drag coefficient in this example, and any reduction of the shock strength leads to significant reduction in drag. Figures 6.7(a) and 6.7(b) compare the entropy generated by the shock waves occurring on the NACA0012 and the optimized airfoils.

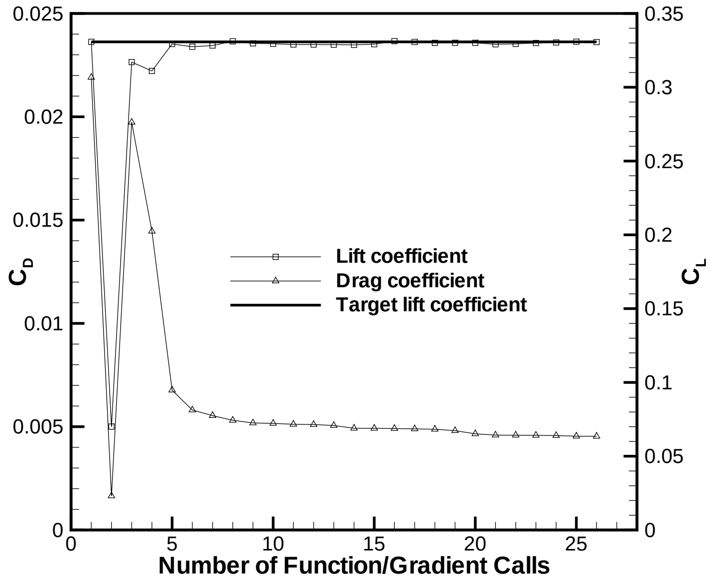


Figure 6.4: Convergence of the lift and drag coefficients for the case of steady-state lift constrained drag minimization.

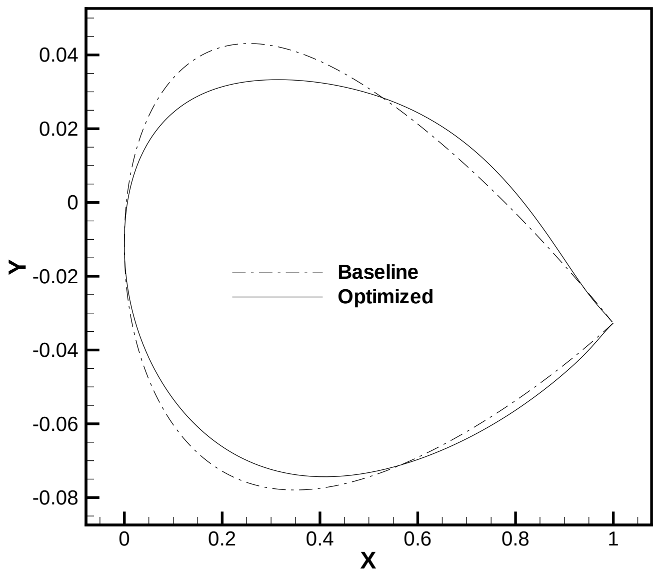


Figure 6.5: Comparison of the NACA0012 airfoil used as the starting point in the case of steady-state lift constrained drag minimization and the final optimized airfoil.

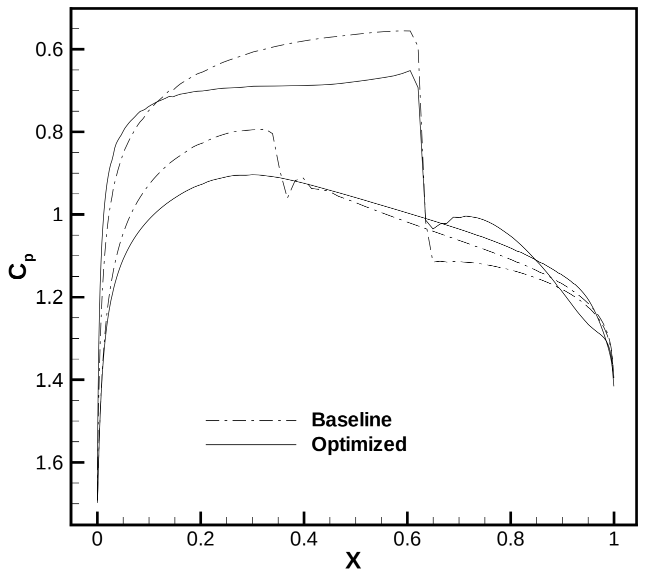
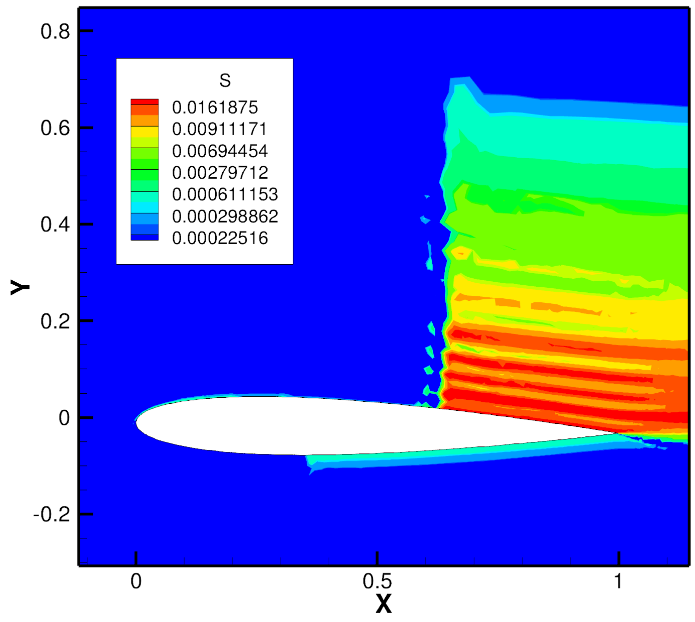
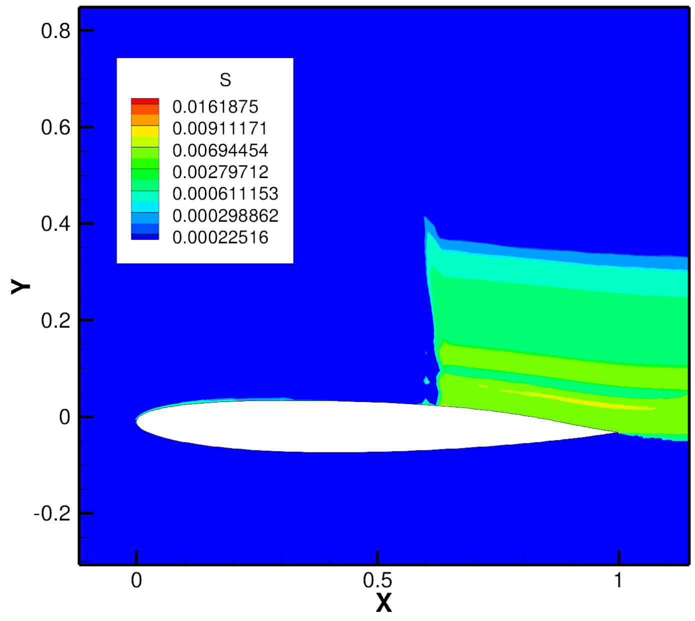


Figure 6.6: Comparison of the pressure coefficient distributions between the NACA0012 airfoil and the optimized airfoil for the steady-state lift constrained drag minimization example.



(a)



(b)

Figure 6.7: Comparison of the entropy generated in the flow field for the baseline NACA0012 airfoil and the optimized airfoils for the case of lift constrained drag minimization.

6.2 Shape Optimization in Unsteady Flow Problems

Three example cases are presented to demonstrate the use of adjoint-based gradients for shape optimization in unsteady problems with prescribed motion of the airfoil. All three involve the sinusoidal pitching of a NACA0012 airfoil about its quarter-chord location. The flow and pitch conditions are identical for all cases with a free-stream Mach number of 0.755 combined with a mean angle-of-attack of 0.016^o. The time dependent pitch criteria consists of an amplitude of 2.51^o and a reduced frequency of 0.0814. These flow parameters result in the generation of a moving shock wave on the upper surface during the upward pitching motion of the airfoil and a moving shock wave on the lower surface during the downward pitching motion. The computational mesh used in the steady-state optimization example was used for the unsteady cases as well. The first $1\frac{1}{4}$ pitch cycles of the airfoil beginning with a steady-state solution at the mean angle-of-attack was taken to be the time domain of interest for all of the examples. Although frequency domain methods may be better suited for highly periodic problems, a periodic problem with initial transient behavior was chosen in this case for demonstration purposes. As described earlier, the design variables form a vector of weights that control the magnitude of bump functions placed at various chordwise locations of the airfoil, which is to be deformed. In order to demonstrate the advantage of using the adjoint method for optimization problems with a large numbers of design variables, bump functions were placed at x-coordinate locations corresponding to all points defining the surface of the airfoil. The computational mesh used for this example consisted of 290 surface mesh points, thus translating into a equal number of design variables.

6.2.1 Time-Dependent Load Matching

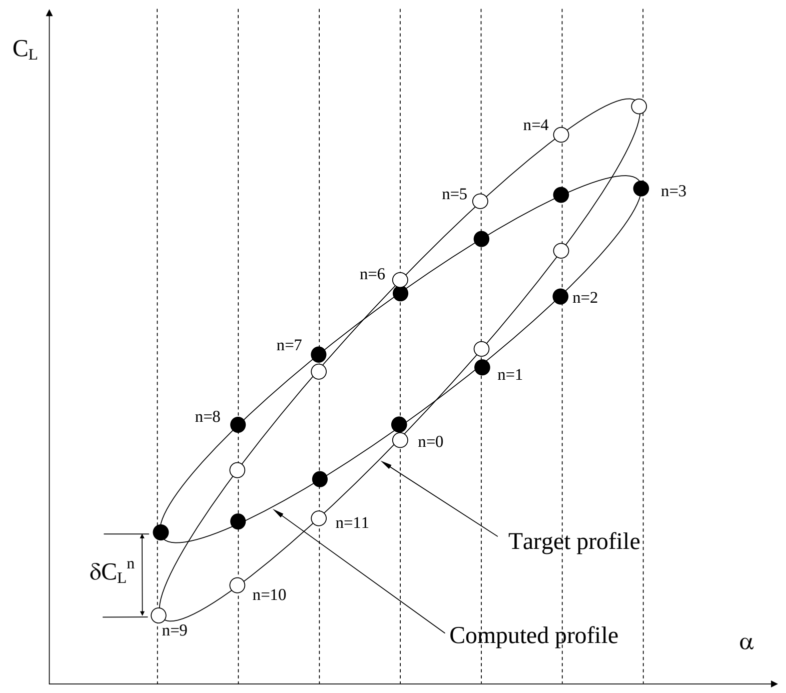
The goal of this test case is to optimize the shape of a NACA0012 airfoil such that the time-dependent load profile matches that of a sinusoidally pitching NACA64210 airfoil. This is an inverse design problem but different from the normal approach of matching some target pressure distribution in that the time dependent loads are targeted. The time integration for this example is carried out from a steady-state solution to a non-dimensional time of 48.2431, which again corresponds to the first $1\frac{1}{4}$ periods of pitch for the airfoil. The temporal resolution consists of 40 time-steps and the time-integrated "global" objective functional is defined as the mean of local functionals evaluated at each time-step as

$$L^g = \left(\frac{1}{n_f + 1}\right) \sum_{n=0}^{n=n_f} L^n \tag{6.2}$$

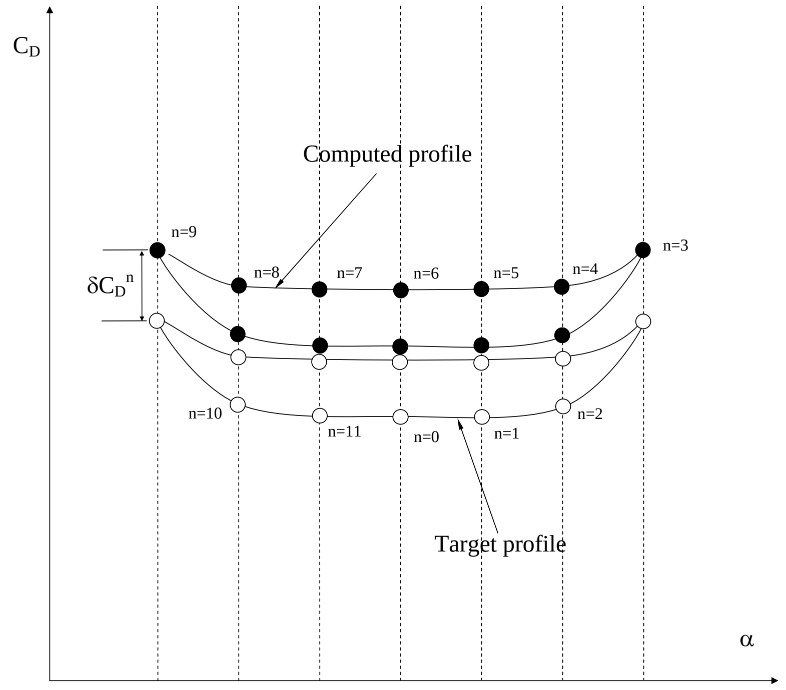
where n_f refers to the total number of time-steps chosen to represent the time domain. The local objective functional at each time-step is the linear combination of the differences between target and computed loads defined as:

$$L^n = \frac{1}{2}\left(C_L^n - C_{L target}^n\right)^2 + \frac{10}{2}\left(C_D^n - C_{D target}^n\right)^2 \tag{6.3}$$

where the factor of 10 for the drag coefficient is introduced to equalize the difference in order-of-magnitude between the lift and drag coefficients. Figures 6.8(a) and 6.8(b) clarify how the functional is constructed as differences between the computed and target time dependent loads at each time-step. Note that there is no constraint in the functional that controls the shape of the optimized airfoil. The only guarantee based on the formulation is that, when the functional goes to zero, the target and computed time dependent load profiles match. The shape is constrained in a loose sense only through the bounds placed on the design variables such that degenerate geometries do not result during the course of the optimization. As in the case of the steady-state problem, these are computed *a priori* and are read-in by the optimization routine at the onset of the optimization process. It is not necessary that the optimized airfoil



(a)



(b)

Figure 6.8: Illustration of the difference between computed and target time-dependent loads in the objective functional used for the time-dependent load matching optimization example.

match the shape of the NACA64210 airfoil although it is one of the possible solutions to the optimization problem. The convergence of the time-integrated functional and the norm of the gradient vector are shown in Figure 6.9. Figure 6.10 shows the optimized airfoil for this case. A pronounced hook shape close to the trailing edge of the optimized airfoil is visible from the figure. Since the target airfoil has finite lift at zero angle-of-attack, the baseline symmetric airfoil must be deformed such that it gains some degree of camber. The trailing-edge flow exit angle being the most influential on turning the flow from the freestream direction (i.e. camber) has pushed the optimization in that direction. In reality, a severely curved trailing edge will typically lead to flow separation and loss of lift with increase in drag. This is not the case for the presented example since the flow and adjoint solvers are based on the Euler equations and do not include the effect of viscosity. It should be interesting to observe the results of a shape-unconstrained optimization based on the Navier-Stokes equations in comparison with those presented here. Figures 6.11 and 6.12 compare the computed and targeted time-dependent lift and drag profiles at various stages in the optimization. The drag comparison figure indicates what appears to be a slightly larger error between the optimized and target values than the lift curves at the final design iteration due to the different scales of these coefficients. There is approximately a two order-of-magnitude reduction in the functional and gradient within the first five design iterations, beyond which the convergence rate greatly diminishes. No convergence specific stopping criteria was used except for total computational time in order to terminate the optimization. For this particular example, the optimization was terminated after 15 wall-clock hours and this corresponded to 58 calls of the flow solver and the adjoint code.

6.2.2 Time-Dependent Pressure Distribution Matching

The second optimization example is that of inverse design from a NACA0012 airfoil to a NACA64210 airfoil in the context of unsteady flows. The target time-dependent

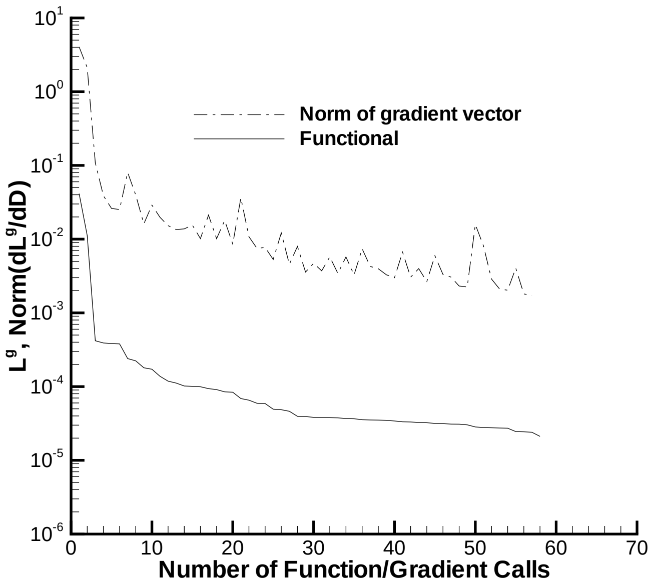


Figure 6.9: Convergence of the objective functional and the norm of the gradient vector for the case of time-dependent load matching.

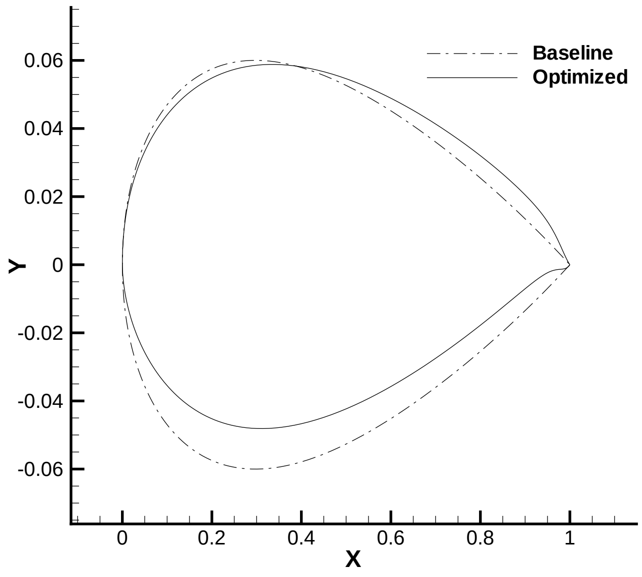
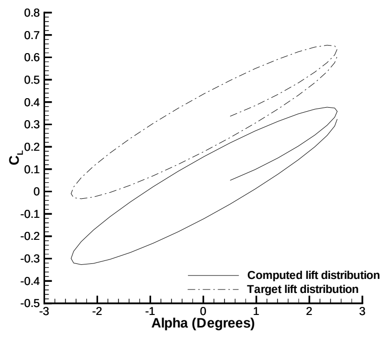
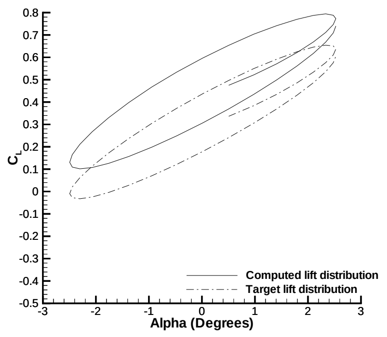


Figure 6.10: Optimized airfoil for the case of time-dependent load matching (exaggerated scale).

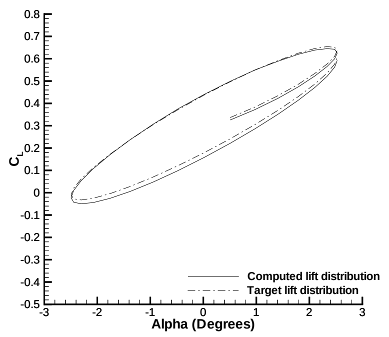
pressure profile for this example is that of a sinusoidally pitching NACA64210 type airfoil. Since inverse design is shape constrained, it is important to know *a priori* whether the geometric parametrization permits the target airfoil to lie within the design space of the problem. An optimization utilizing only the shape parametrization independent of the flow equations was performed in order to check if this were true. It was determined that an exact NACA64210 was not achievable based on the current parametrization. As a consequence the closest possible match from the result of this exploratory geometric optimization was used as the target since this guarantees an achievable shape. Figure 6.13 compares the airfoil that was used as the target against the true NACA64210 airfoil. The objective functional for this example is similar to that of the previous example, with the difference in computed and target loads replaced by the difference in computed and target time-dependent pressure profiles.



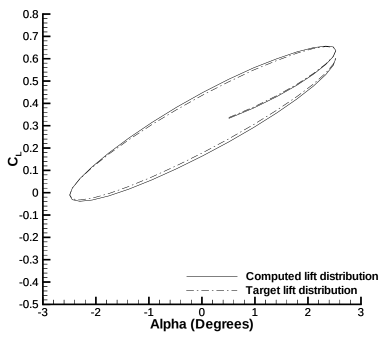
(a) 1st Design Iteration



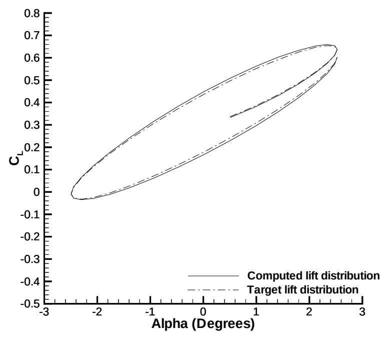
(b) 2nd Design Iteration



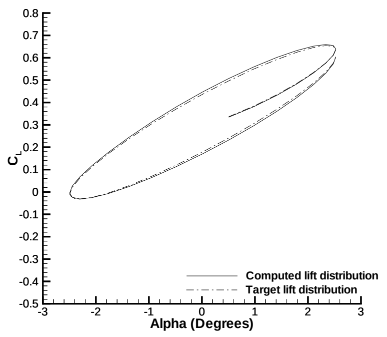
(c) 3rd Design Iteration



(d) 15th Design Iteration

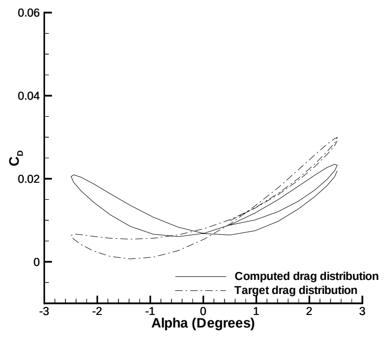


(e) 30th Design Iteration

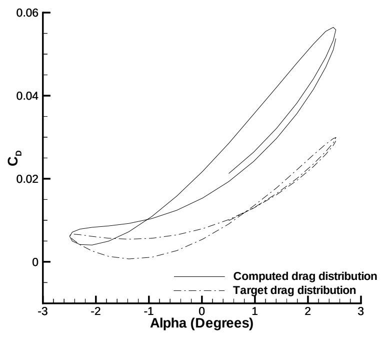


(f) 58th Design Iteration

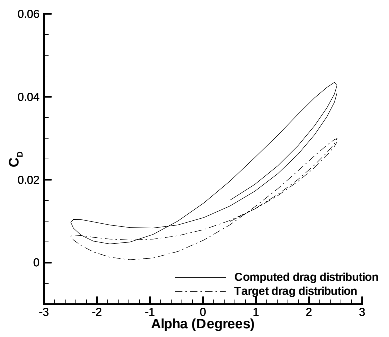
Figure 6.11: Comparison of the computed and target time dependent lift coefficient at various stages in the optimization for the case of time-dependent load matching.



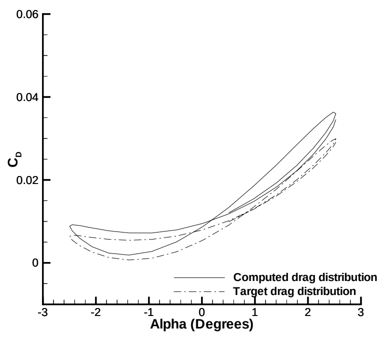
(a) 1st Design Iteration



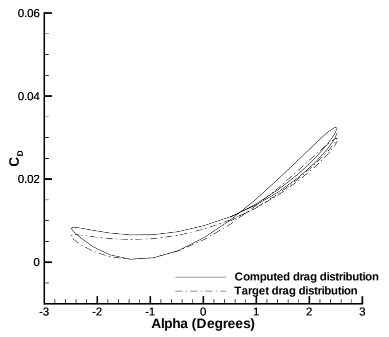
(b) 2nd Design Iteration



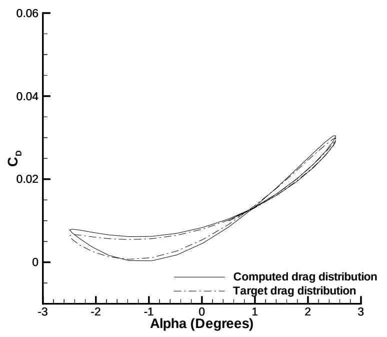
(c) 3rd Design Iteration



(d) 15th Design Iteration



(e) 30th Design Iteration



(f) 58th Design Iteration

Figure 6.12: Comparison of the computed and target time dependent drag coefficient at various stages in the optimization for the case of time-dependent load matching.

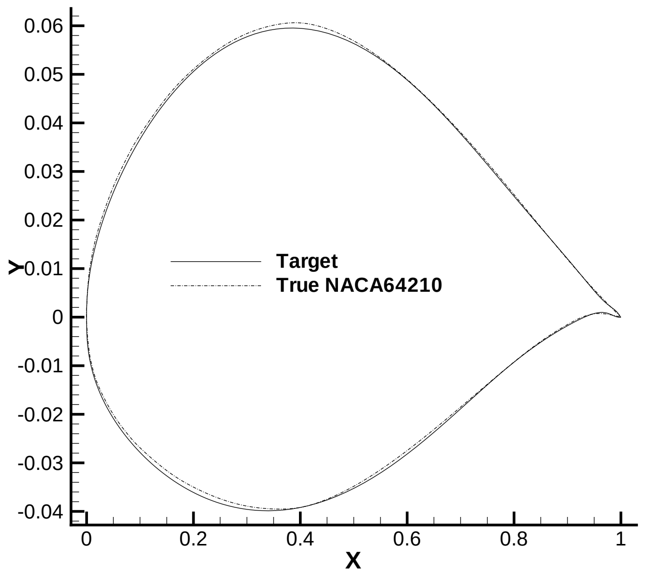


Figure 6.13: Comparison of target airfoil and true NACA64210 airfoil for the case of time-dependent pressure distribution matching.

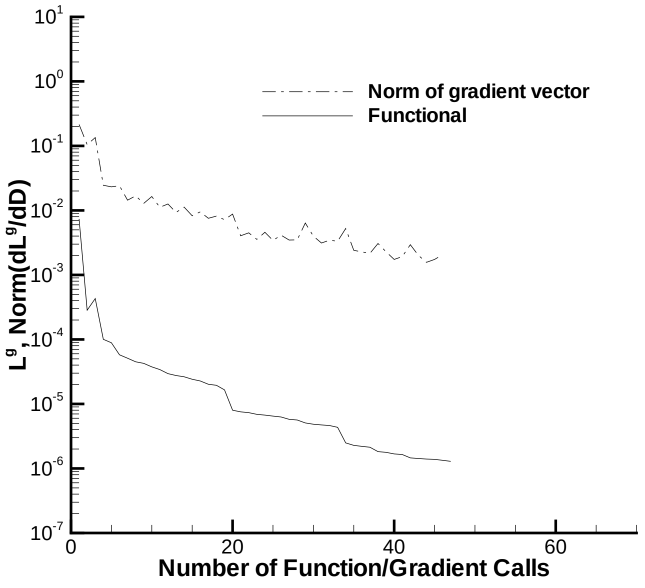


Figure 6.14: Convergence of the objective functional and the norm of the gradient vector for the case of time-dependent pressure distribution matching using 290 design variables.

The local objective functional at each time-level for this example is defined as

$$L^n = \sum_{i=1}^{n_{surf}} \left\{ \frac{1}{2} \left(p_i^n - p_{i\,target}^n \right)^2 \right\} \tag{6.4}$$

Figure 6.14 shows the convergence of the objective functional L^g and the norm of the gradient for this case. As in the previous case, the functional drops by about four orders-of-magnitude before the convergence begins to stall. This is not an acceptable scenario since it is known that the target airfoil lies within the design space in this case. Theoretically the functional should be able to reach machine zero while the gradient attains the squareroot of machine zero. The problem was suspected to be the complexity of the design space and the inability of the LBFGS-B [64] algorithm to navigate through it. Since the optimization consists of 290 design variables, the problem was simplified by reducing the total number of design variables to 14. As previously described, a new target airfoil was chosen using the parametrization based

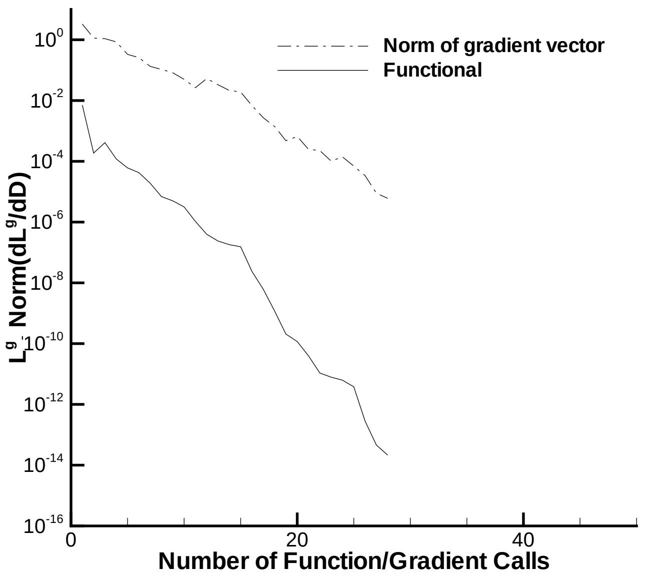
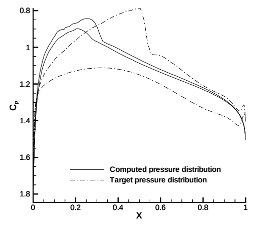
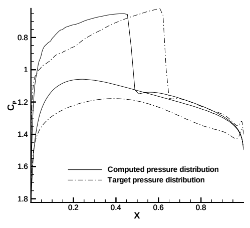


Figure 6.15: Convergence of the objective functional and the norm of the gradient vector for the case of time-dependent pressure distribution matching using 14 design variables.

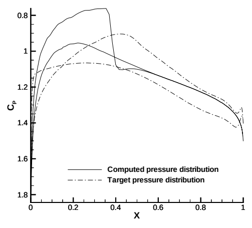
on fewer design variables and a second unsteady inverse design optimization was performed. Figure 6.15 shows the convergence for this case. The results verify the suspected cause for stalling in the previous case, since the reduction of the total number of design variables results in the expected trend. The functional reaches machine zero in 28 design iterations while the gradient approximately reaches square-root of machine precision. The termination criteria used in this case was a value of $1e^{-14}$ for the functional. Figures 6.16 and 6.17 compare the target and computed pressure profiles at different design iterations for three temporal locations for both the 290 and 14 design variables cases. Although qualitatively the results look impressive in both cases, it is clear from the convergence plots that the true optimum has been achieved only in the 14 design variable case.



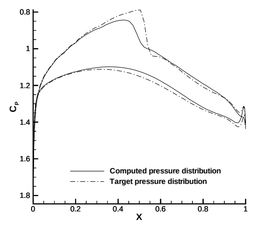
(a) ID=1, n=1



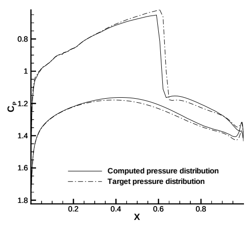
(b) ID=1, n=11



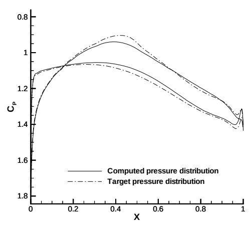
(c) ID=1, n=32



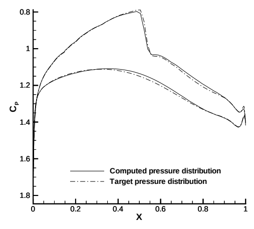
(d) ID=3, n=1



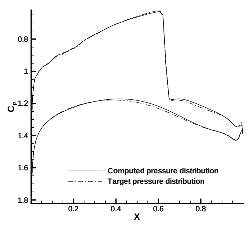
(e) ID=3, n=11



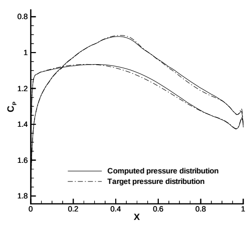
(f) ID=3, n=32



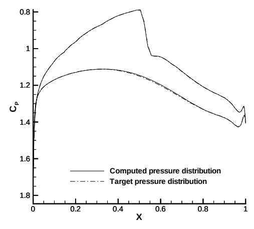
(g) ID=15, n=1



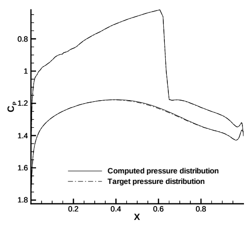
(h) ID=15, n=11



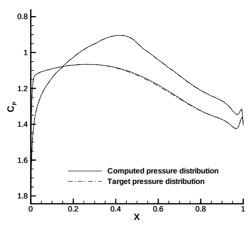
(i) ID=15, n=32



(j) ID=47, n=1

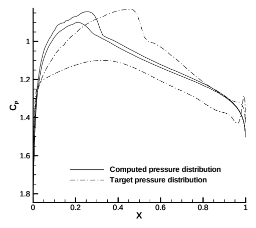


(k) ID=47, n=11

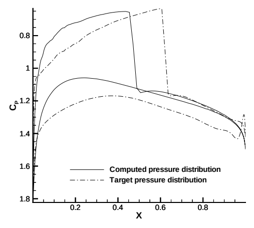


(l) ID=47, n=32

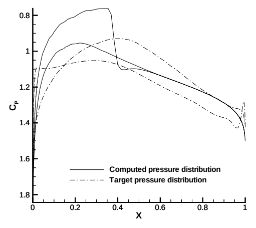
Figure 6.16: Comparison of computed and target pressure profiles at different design iterations for three time-steps $n = 1$, $n = 11$, and $n = 32$ for the time-dependent pressure distribution matching optimization example with 290 design variables. (Legend: ID=design iteration number)



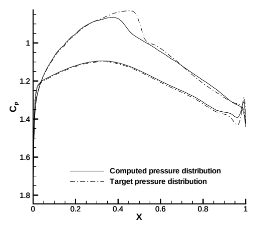
(a) ID=1, n=1)



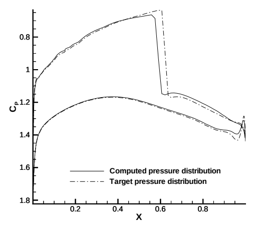
(b) ID=1, n=11



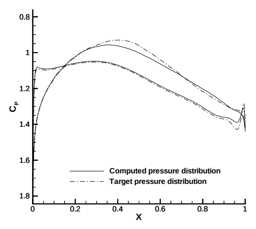
(c) ID=1, n=32



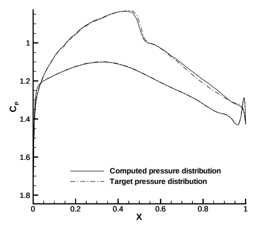
(d) ID=3, n=1



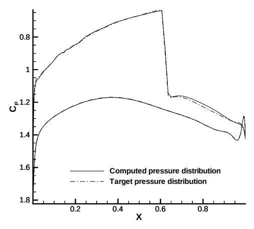
(e) ID=3, n=11



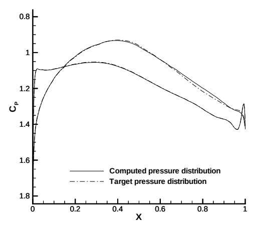
(f) ID=3, n=32



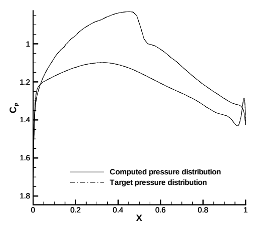
(g) ID=7, n=1



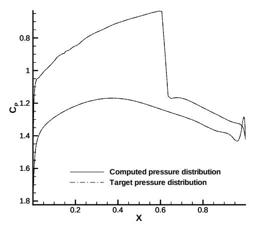
(h) ID=7, n=11



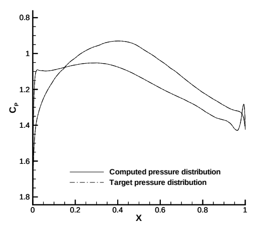
(i) ID=7, n=32



(j) ID=28, n=1



(k) ID=28, n=11



(l) ID=28, n=32

Figure 6.17: Comparison of computed and target pressure profiles at different design iterations for three time-levels $n = 1$, $n = 13$, and $n = 27$ for the time-dependent pressure distribution matching optimization example with 14 design variables. (Legend: ID=design iteration number)

6.2.3 Time-Dependent Lift Constrained Drag Minimization

The last optimization example in this section is that of lift-constrained time-dependent drag minimization. The objective formulation for this case is the same as that used for the time-dependent load matching problem. The difference lies in the target loads used in the formulation. Since this a lift-constrained drag minimization problem, the target time-dependent lift is set to be that of the baseline NACA0012 airfoil itself, while the target drag is set to be zero at all temporal locations. The flow conditions and pitch criteria remain identical to the previous two examples. The objective functional cannot reach zero in this case since the drag acting on a pitching airfoil is always nonzero due to unsteady effects, the entropy generated by the moving shock waves and due to dissipative errors in the solution. Figure 6.18 shows the convergence of the functional and the gradient where the optimization progress is seen to level out after approximately 20 cycles. Figure 6.19 shows the optimized airfoil in comparison with the baseline NACA0012 airfoil. Figure 6.20 shows the convergence of the time-averaged and maximum drag coefficients during the course of the optimization. An approximate reduction of 85 counts of drag in the peak drag coefficient and 50 counts in the time-averaged drag coefficient is observed. Figures 6.21 and 6.22 compare the baseline and optimized time-dependent lift and drag profiles.

6.3 Shape Optimization for Flutter Control

In this section, the adjoint-based gradients obtained in the unsteady coupled aeroelasticity problem are used to optimize the shape of an airfoil for the purpose of suppressing flutter.

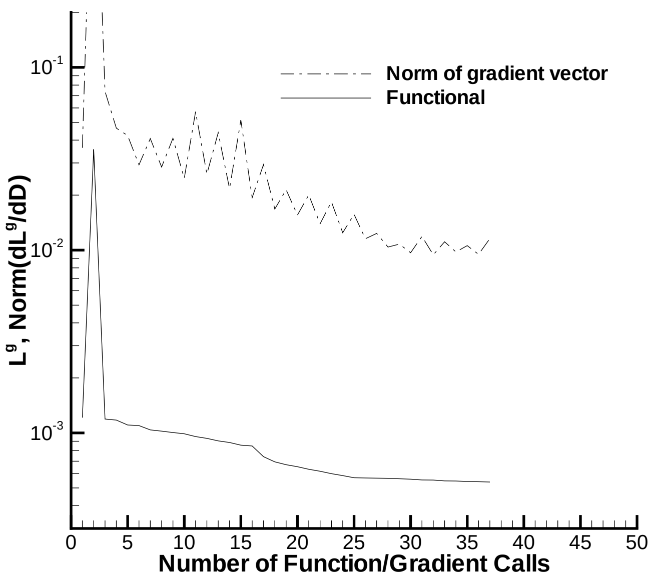


Figure 6.18: Convergence of the objective functional and the norm of the gradient vector for the case of time-dependent lift constrained drag minimization.

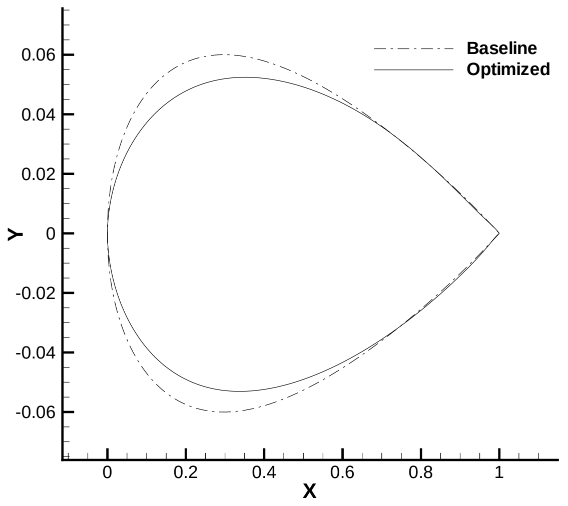


Figure 6.19: Comparison of the NACA0012 airfoil used as the starting point and the final optimized airfoil for the case of time-dependent lift constrained drag minimization.

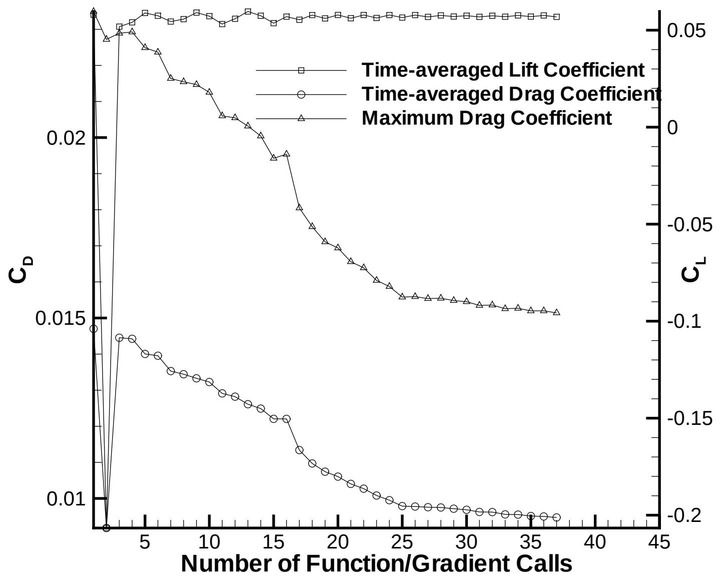


Figure 6.20: Convergence of the time-averaged and peak drag coefficients for the time-dependent lift constrained drag minimization optimization example.

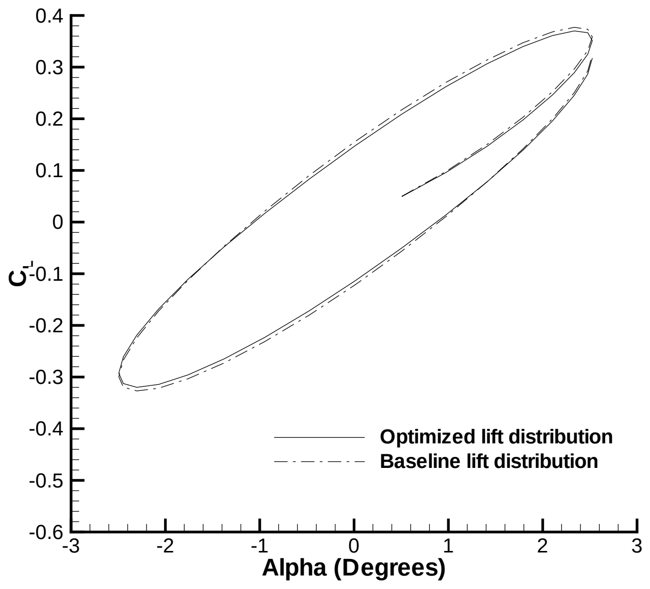


Figure 6.21: Comparison of baseline and optimized time-dependent lift profiles for the time-dependent lift constrained drag minimization optimization example.

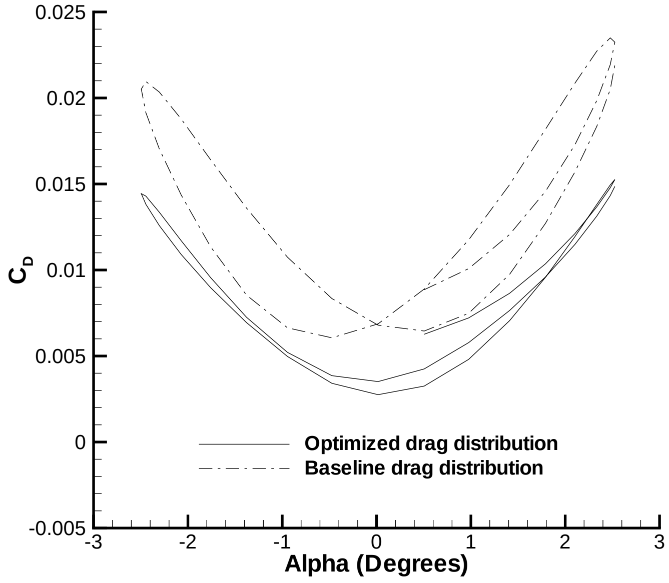


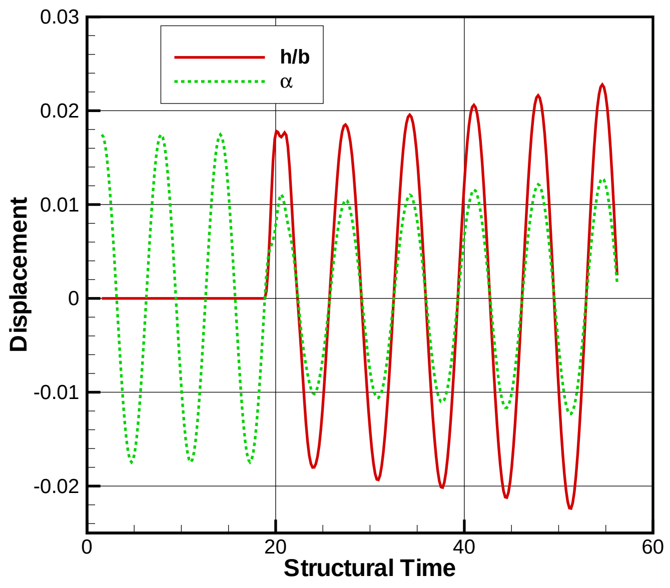
Figure 6.22: Comparison of baseline and optimized time-dependent drag profiles for the time-dependent lift constrained drag minimization optimization example.

6.3.1 Single Element Airfoil

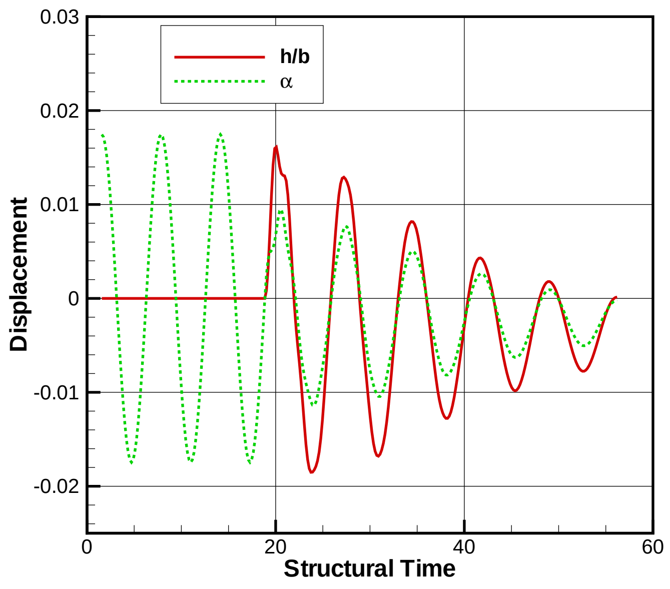
The goal of this optimization example is to consider a point in the flutter diagram of the described aeroelastic validation test case where the airfoil exhibits divergent behavior, and to change the shape of the airfoil such that it produces a damped aeroelastic response while satisfying a constraint on the steady-state lift of the airfoil. The time-integration process consisted of 36 uniform time-steps per forced pitch cycle combined with 3 pitch cycles and 214 time-steps of the same size for the aeroelastic response to evolve. The chosen number of time-steps for the aeroelastic response evolution correspond to approximately $5\frac{1}{2}$ periods in both the pitch and plunge axes at the starting design point of the optimization. The objective functional for the optimization was chosen to be

$$L = \mathbf{r}^{f^T}[W_1]\mathbf{r}^f + W_2(C_{Ls} - C_{Ls_{target}}) \tag{6.5}$$

where $\mathbf{r}^f$ refers to the structural state vector at the end of the time-integration process, $[W_1]$ is a diagonal weighting matrix with equal weighting parameters of 100, W_2 is a weight on the constraint, chosen to be unity, and C_{Ls} is the steady-state lift coefficient of the NACA64A010 airfoil. Since the airfoil is symmetric and has zero lift at zero angle-of-attack, the steady-state solution is computed at the maximum pitch angle of the forced pitch cycles. The forced unsteady pitching of the airfoil around the elastic axis begins at this point and stops at the neutral position of zero angle-of-attack (after 3 pitch cycles) at which point the aeroelastic response is allowed to evolve. The target value for the steady-state lift coefficient was taken to be that of the baseline NACA64A010 airfoil at the maximum pitch angle of 1^o. The objective formulation is such that a vanishing functional results in zero displacements and zero velocities in both degrees-of-freedom of pitch and plunge. Since the aeroelastic time-integration ends fairly quickly ($5\frac{1}{2}$ periods), this in fact creates a stiff objective formulation. However, it is not necessary to drive the objective to zero and the optimization may be stopped once a damped response is observed and the constraint on the steady-state lift is satisfied. For the described test case, the airfoil produces a neutral response for a flutter velocity $V_f = 0.65$ at a Mach number of $M_\infty = 0.825$. An unstable point of $V_f = 0.75$ at this Mach number was chosen to be the starting design point for the optimization. Figure 6.23(a) shows the aeroelastic response of the airfoil at the starting design point. The divergent behavior is clearly visible in both pitch and plunge axes. Prior to running the optimization, the time-integration for the aeroelastic response was carried out over a temporal domain that was ten times the size of the chosen domain (214 time-steps) in order to ensure that the divergent response did not lead to limit-cycle behavior. The shape of the airfoil was controlled by an equal distribution in the chordwise direction of 16 bump functions on the upper surface and 16 bump functions on the lower surface resulting in a total of 32 design variables. As mentioned earlier, the design variables are the weights controlling the magnitudes of each one of the bump functions. Although in practice any modification of the shape of the airfoil would result in changes to the structural parameters controlling the aeroelastic response, it is assumed that the structural parameters are constant and independent of the geometry for the optimization example presented here. The optimization was terminated after 41 design iterations and Figure 6.23(b) shows the aeroelastic response of the optimized airfoil. While the displacements and the velocities have not been driven to zero, the plot clearly indicates a rapidly decaying response. Figure 6.24 shows the convergence of the objective functional and the L_2 norm of the gradient vector. During the course of the optimization, a decaying response was observed after only 11 design iterations, but the optimization was not terminated at this point since the tolerance on the constraint was not satisfied. The tolerance was set to be a maximum of 0.25% deviation from the target constraint over at least two consecutive design iterations. Figure 6.25 indicates that the constraint satisfies the set tolerance only at 41 design iterations at which point the optimization was terminated. Figure 6.26 compares the optimized and the baseline airfoil geometries using 1:1 and exaggerated scales.



(a) Diverging aeroelastic response of baseline airfoil



(b) Decaying aeroelastic response of optimized airfoil

Figure 6.23: Aeroelastic response of baseline and optimized airfoils for the single element airfoil flutter suppression optimization example.

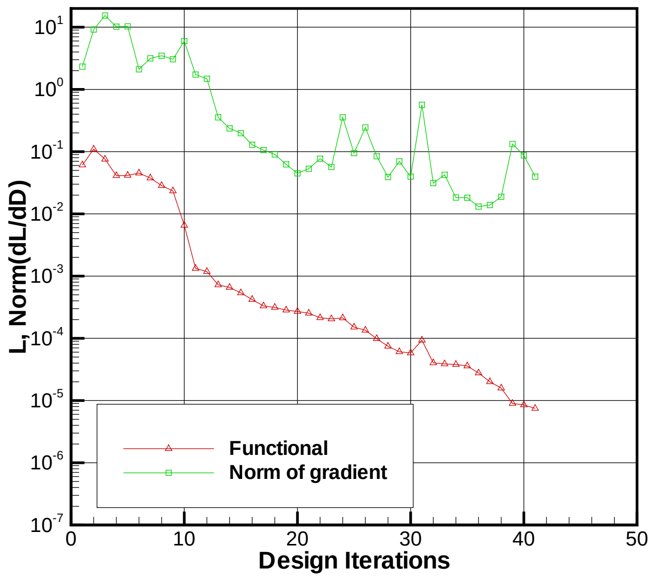


Figure 6.24: Convergence of objective functional and the norm of gradient vector for the single element airfoil flutter suppression optimization example.

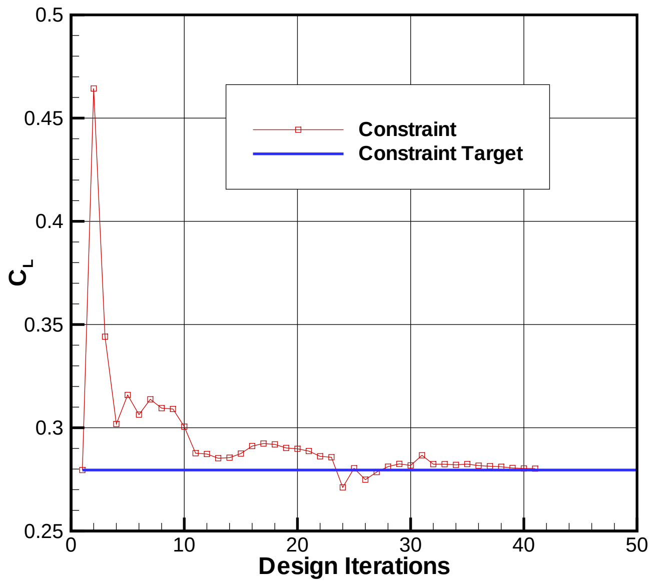
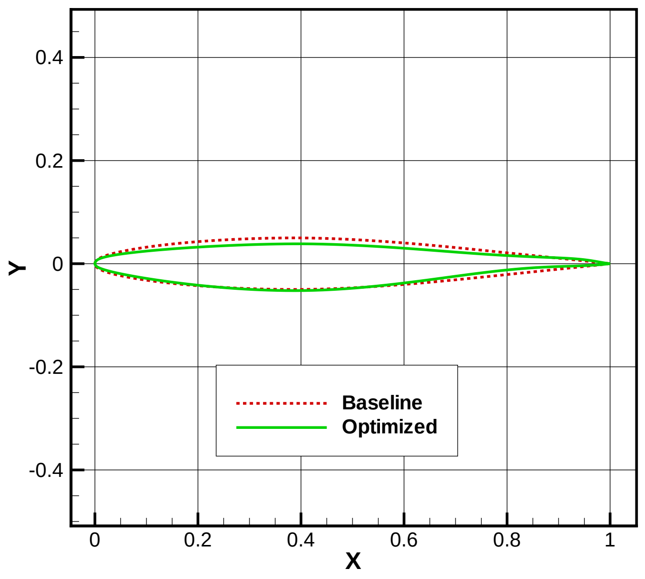


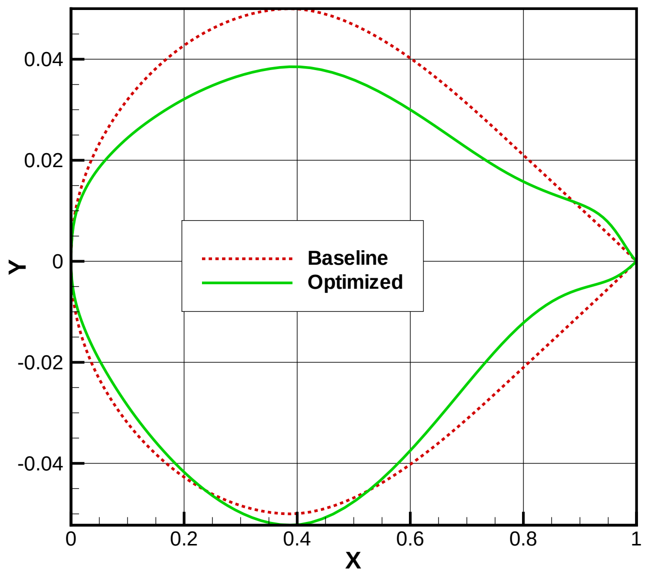
Figure 6.25: Convergence of the constraint on the objective functional.

6.3.2 Two-Element Slotted Airfoil

The purpose of the second aeroelastic optimization example is to demonstrate the advantage of using an unstructured mesh framework. The problem remains identical to the first optimization example in that the goal is to suppress divergent flutter behavior of an airfoil using gradient based optimization. However, the airfoil under consideration contains two elements separated by a slot and is thus easier to treat using unstructured methods. Figure 6.27 shows the unstructured computational mesh consisting of approximately 7,800 elements around the slotted airfoil for this problem. Since the primary focus of this optimization example is the demonstration of the advantage of using unstructured meshes for complex geometries in conjunction with the derived linearization, the physical aspects of the aeroelastic problem itself are not very important. With this in mind, the structural parameters from the aeroelastic validation test case used in the previous optimization example are carried over to this



(a) 1:1 scale



(b) Exaggerated scale

Figure 6.26: Comparison of the baseline and the optimized airfoils for the single element airfoil flutter suppression optimization example.

example. A suitable starting point where the airfoil exhibits a divergent aeroelastic response was chosen by trial and error to be a flutter velocity of 1.85 at a Mach number of 0.6. The response of the slotted airfoil at this point in the flutter diagram is shown in Figure 6.28(a). The offsets in the mean of the oscillations of both the pitch and plunge axes from zero indicate that for the given structural and aerodynamic parameters there exists a nontrivial steady-state aeroelastic solution. The objective functional for this example was chosen to be

$$L = \sum_{n=n_f-10}^{n_f} \mathbf{r}^{nT}[W_1]\mathbf{r}^n + W_2(C_{Ls} - C_{Ls_{target}}) \tag{6.6}$$

In contrast to the first optimization example, the aeroelastic term in the objective functional has been converted to a summation over the last ten time-steps. The reasoning being that, in addition to the case of flutter free behavior, zero velocities and zero displacements can occur at the peaks of the oscillations. Considering the offset in the oscillatory mean values from zero, this change in the objective formulation effectively constrains the optimization from moving to a shape that produces an oscillatory peak at zero once the time-integration process terminates. The weights in the objective remain identical to the first example and the target steady-state lift coefficient again is that of the baseline slotted airfoil at the maximum pitch angle of the forced pitching cycles. The forced pitch parameters are also identical to the first example, with the exception of the number of forced pitch cycles. For this example the forced pitching occurs only over three-quarters of a period before the aeroelastic response is allowed to evolve. Since the overall size of the time domain remains identical to the previous example, this modification to the forced pitching cycles allows more time for the aeroelastic response evolution. Four sets of eight bump functions corresponding to the upper and lower surfaces of both the main airfoil element and the flap result in a total of 32 design variables that control the overall shape of the airfoil. Figure 6.28(b) shows the aeroelastic response of the optimized airfoil and clearly indicates neutral behavior. Figures 6.29 and 6.30 show

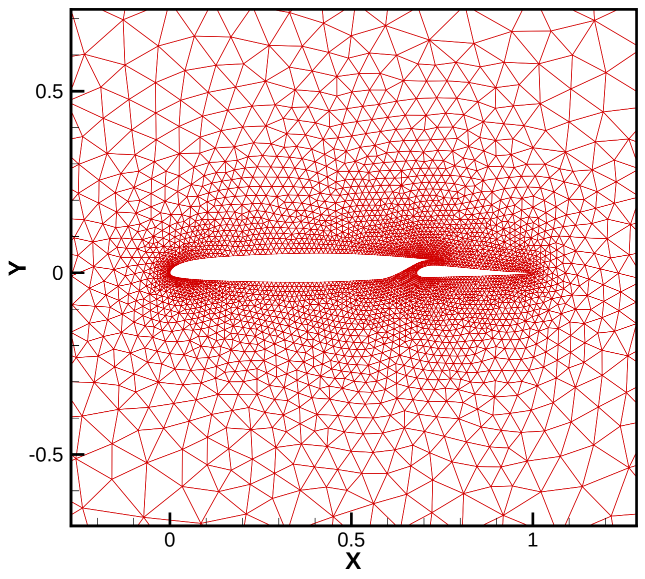
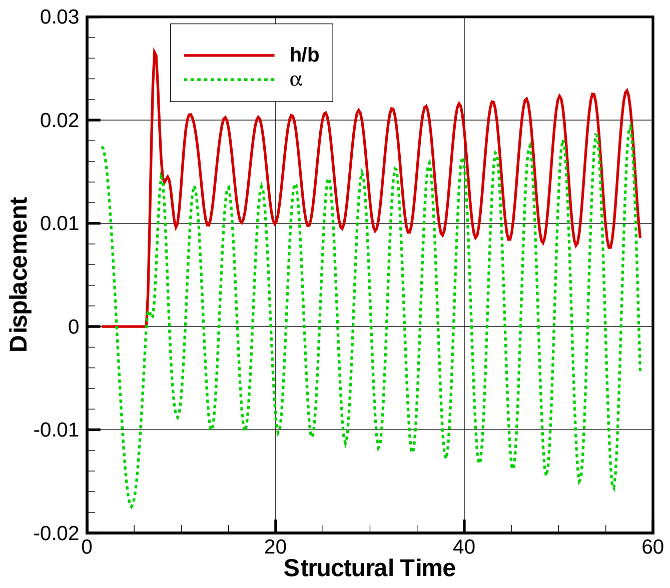


Figure 6.27: Slotted airfoil mesh consisting of approximately 7,800 elements used for the two-element slotted airfoil flutter suppression optimization example.

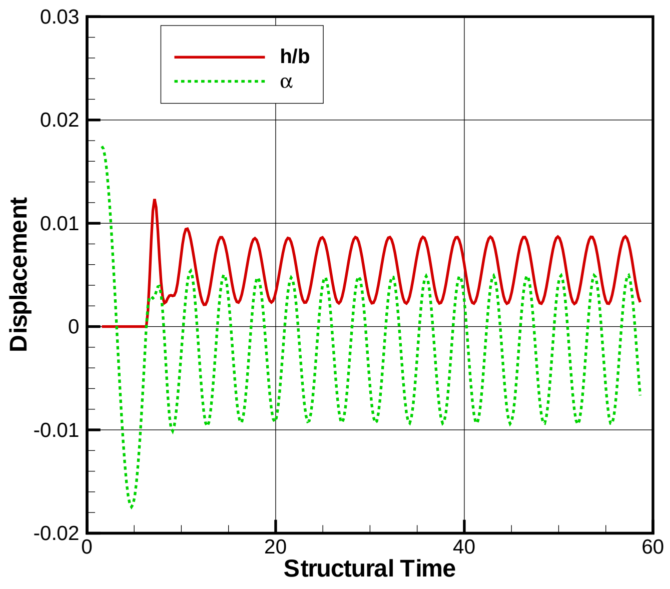
the convergence of the objective, the norm of the gradient and the constraint. The optimization was terminated after 21 design iterations. Figure 6.31 compares the baseline and optimized airfoil shapes.

6.4 Error Estimation and Adaptation of the Time Domain

This section presents the second set of example problems where the adjoint-based error estimates are used for adaptation. The distribution of the temporal discretization error is used to drive the adaptation of the temporal resolution, while the temporal distribution of the algebraic error from each discipline is used to adapt the convergence tolerances for each set of governing equations. The adjoint variables computed for the uncoupled unsteady problem with prescribed motion of the airfoil are used for



(a) Diverging aeroelastic response of baseline slotted airfoil



(b) Neutral aeroelastic response of optimized slotted airfoil

Figure 6.28: Aeroelastic response of the baseline airfoil and the optimized airfoil for the two-element slotted airfoil flutter suppression optimization example.

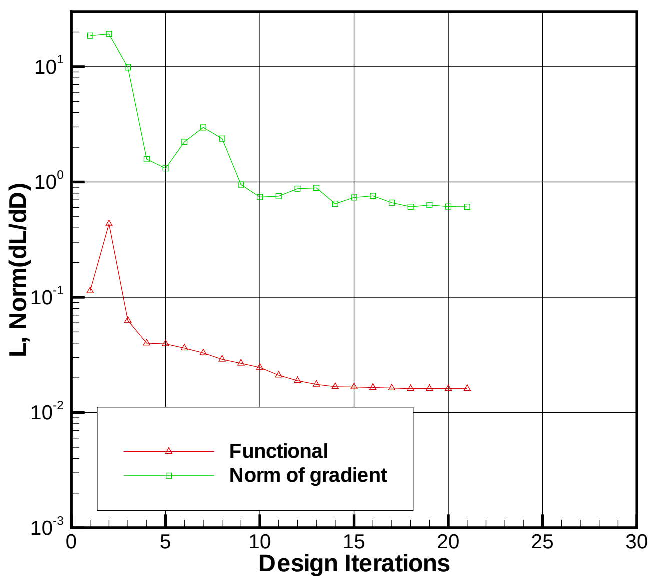


Figure 6.29: Convergence of objective and the norm of the gradient for the two-element slotted airfoil flutter suppression optimization example.

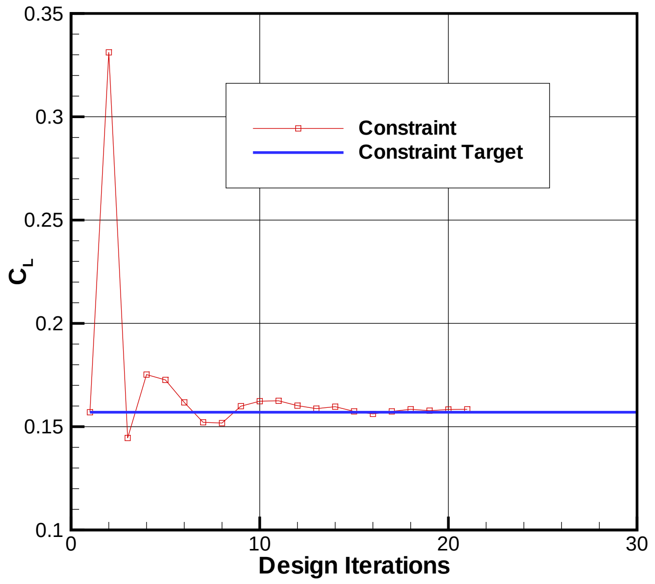
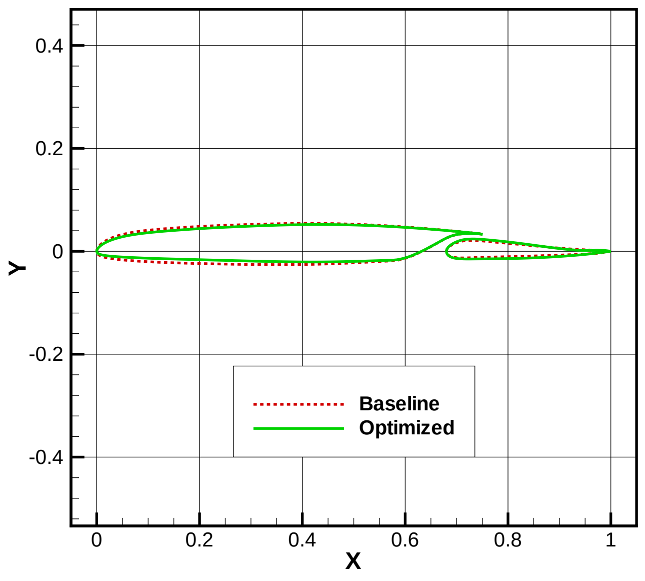
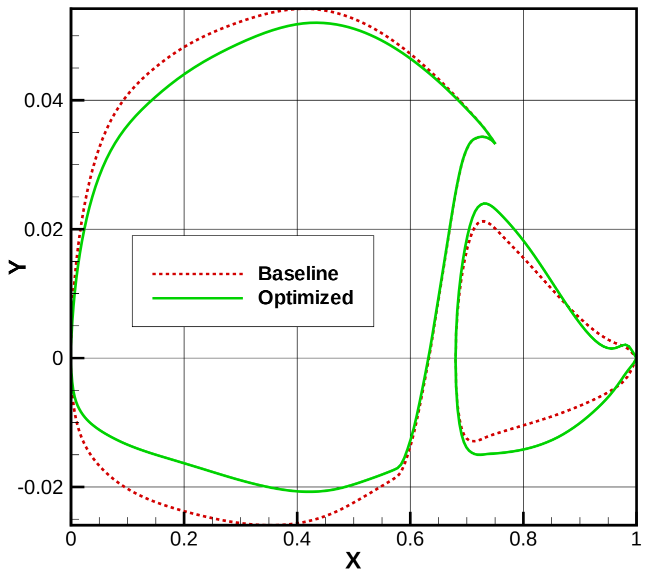


Figure 6.30: Convergence of the constraint on the objective functional for the two-element slotted airfoil flutter suppression optimization example.



(a) 1:1 scale



(b) Exaggerated scale

Figure 6.31: Comparison of baseline and optimized airfoils for the two-element slotted airfoil flutter suppression optimization example.

both examples.

6.4.1 Description of Comparative Methods

Two example problems, one where the functional of interest is a non-time-integrated quantity measured at the end of the time-integration process and the other where the functional is a time-integrated quantity are presented. The purpose of the examples presented is to demonstrate the advantage of using goal/adjoint-based adaptation of the temporal domain over traditional adaptation methods. To this end the results are compared against the more commonly used but basic method of uniform temporal refinement, and the more sophisticated method of utilizing temporal discretizations of different orders-of-accuracy for estimating local temporal error and adaptively refining the time-step.

The uniform refinement method employed involves the doubling of the temporal mesh resolution at each adaptation cycle using a 2-to-1 nested subdivision of temporal elements, while simultaneously tightening the convergence tolerances by a factor of three for all disciplines at all time-steps. As for the estimation of local temporal error, the method proposed in reference [79] is used. The solution to the analysis problem obtained using the second-order accurate BDF2 scheme is used to determine the temporal derivative term discretized based on a third-order accurate BDF3 time-integration scheme. The estimate of the local error at each time-step is then computed as

$$e_{local} = \left\| \left[\frac{dA\mathbf{U}}{dt} \right]_{BDF3} - \left[\frac{dA\mathbf{U}}{dt} \right]_{BDF2} \right\|_2 \tag{6.7}$$

Details of the BDF3 discretization of the time derivative on nonuniform temporal meshes are shown in Appendix A. The adaptation for this method is based on refining temporal elements that have local error higher than the computed mean of the local error distribution in the time domain. The convergence tolerances at each time-step are set to be equal to that of the local temporal error in order to ensure that the algebraic error is less than or equal to the temporal discretization error. Computational cost is measured using the *systemclock* intrinsic function built into FORTRAN95. As a convention, the cost shown in the comparative plots at any particular adaptation cycle regardless of the adaptation method employed includes the cost of all preceding adaptation cycles. All adaptation methods are compared in reference to numerically exact results computed using a uniform time domain of resolution at least two orders-of-magnitude higher than those achieved through adaptation.

6.4.2 Non-Time-Integrated Functional

The first example consists of a pitching NACA64A010 airfoil, where the angle-of-attack as a function of time is prescribed. The functional of interest is the lift coefficient of the airfoil evaluated at the end of the time-integration process. The computational mesh for this case is the same one used for the aeroelastic validation test case and is shown in Figure 5.9. The airfoil operates at a Mach number of 0.3 and Figure 6.32(a) shows the prescribed displacement in time. The corresponding time variation of the lift coefficient of the airfoil is shown in Figure 6.32(b). The prescribed motion begins at an angle-of-attack of 0.5^o and remains undisturbed for a short period before a spike (albeit continuous) of 0.5^o is introduced. The angle-of-attack then returns to the steady-state value of 0.5^o where it remains for an extended period before a shallow pitch down by 0.5^o is followed by a rapid pitch up to 5^o. Once the angle-of-attack reaches the maximum of 5^o, the motion of the airfoil is stopped and only transient effects remain in the flow. The initial temporal mesh consists of 50 uniform time steps and the starting convergence tolerances for the flow and mesh equations are set at $1e^{-6}$ and $1e^{-5}$ respectively. The error tolerance used as the termination criteria for adaptation is set as $1e^{-6}$. The problem is well suited for the demonstration of the advantage in using goal-based adaptation over local error-based adaptation.

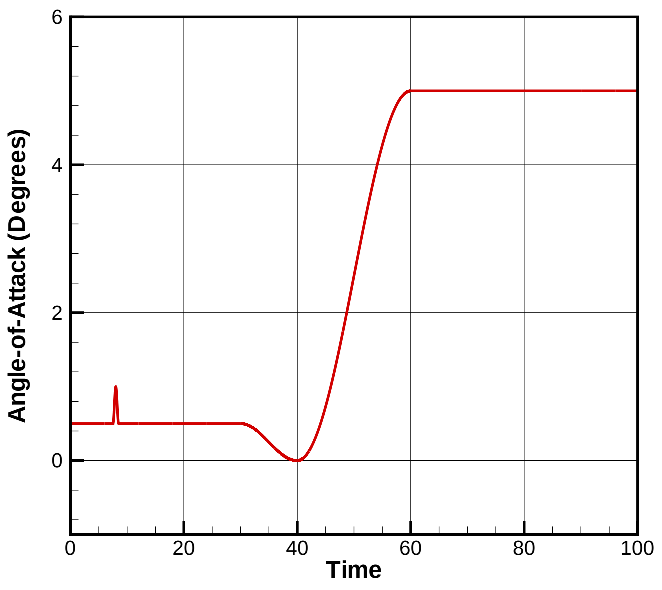
Figures 6.33(a) and 6.33(b) show the convergence of the functional and error in the functional using the three different adaptation methods. The plots indicate that adjoint-based adaptation is able to achieve similar error levels to that of unguided uniform refinement with fewer time-steps. On average, the method appears to provide a factor of saving anywhere between 1.6 and 2 when considering only adaptation. The method however also provides a correction term, which is the sum of all the error distributions and this can be used to improve the value of the functional computed using the adapted domain. When including the correction term in the functional, the method far outperforms uniform refinement at least in terms of the temporal resolution required for the same error levels. Figures 6.34(a) and 6.34(b) show the convergence with respect to computational cost. When considering the adjoint-based adapted curve in the plots, the costs appear similar to that of uniform refinement for the same error levels. However, the method is competitive when the correction term is included in the functional. It should be noted that the adjoint-based error distribution provides the correction term and no additional work is necessary in order to compute it. Therefore, a fair comparison is that between the corrected functional convergence and uniform refinement.

It is interesting to note from the plots that local error-based adaptation performs very poorly for this particular problem. This is expected since the problem was setup specifically to demonstrate the weakness of local error-based adaptation. Theoretically it is known that the spike occurring close to the start of the time-integration process has no effect on the lift coefficient at the end of the time-integration since there is sufficient time after the spike for the airfoil to recover its steady-state lift. Local error-based adaptation methods however are blind to this effect and will target the spike for adaptation since that would be the location of maximum temporal error. This is quite obvious from Figure 6.35 where the distribution of local error shows significant differences compared to the adjoint-based error, both evaluated at the first adaptation cycle. Nearly all of the refinement during the first six adaptation cycles based on local temporal error occurs at the location of the spike. It is only after the local temporal error at this location has been brought down to the same levels as those closer to the end of the time-integration does the method actually start attacking the correct regions relevant to the functional. As a consequence of this, local error-based adaptation performs less efficiently than even uniform refinement of the time domain. Figure 6.36 compares the distribution of the time-step sizes between adjoint adaptation and local error-based adaptation after the final adaptation cycle. It is clear from this plot that local error-based adaptation has refined the temporal mesh significantly in the vicinity of the spike, while the adjoint-based adaptation has almost completely ignored the spike.

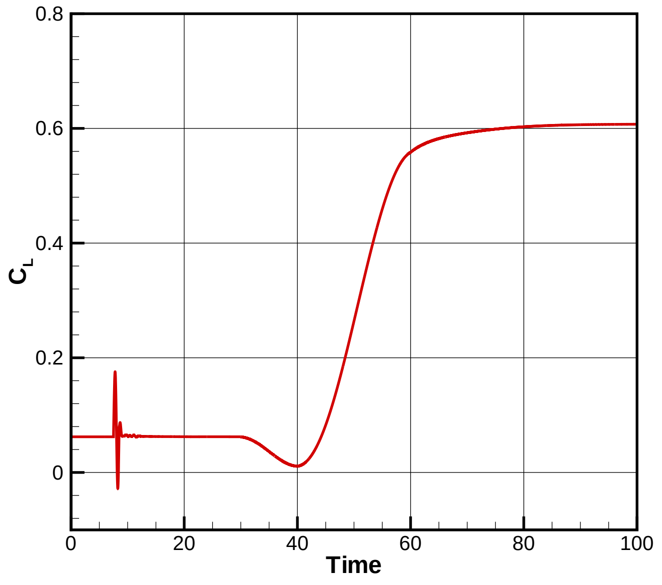
The other aspect of the problem that is tackled by the adaptation procedure is the algebraic error at each time-step due to partial convergence. Figures 6.37 and 6.38 show the distribution of the error due to partial convergence of the flow and mesh equations at various adaptation cycles. The algebraic error is dominant in the region where the airfoil is moving and also close to where the functional is evaluated. As in the case of temporal resolution, the temporal location where the spike in the angle-of-attack occurs is irrelevant to the functional. Figure 6.39 shows the final distribution of the convergence tolerances for the flow and mesh equations after termination of the adaptation. The tolerances progressively become tighter for both sets of equations as the time-integration approaches the point of the functional evaluation and this trend matches closely with intuition.

6.4.3 Time-Integrated Functional

The second example demonstrates the method when the functional of interest is a time-integrated quantity. The problem considers the interaction of a vortex with a pitching airfoil. The airfoil is a NACA0012 undergoing very slow pitching at a reduced frequency of 0.001 in a flow with a freestream Mach number of 0.4225. The amplitude of pitch is 5^o with the pitch center located at the quarter chord of the airfoil. Figures 6.40(a) and 6.40(b) show the computational spatial mesh and a zoomed in

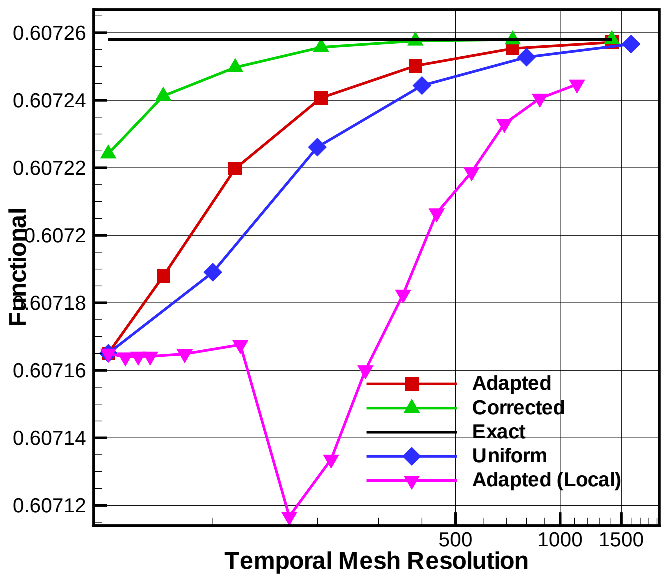


(a) Prescribed variation of airfoil angle-of-attack

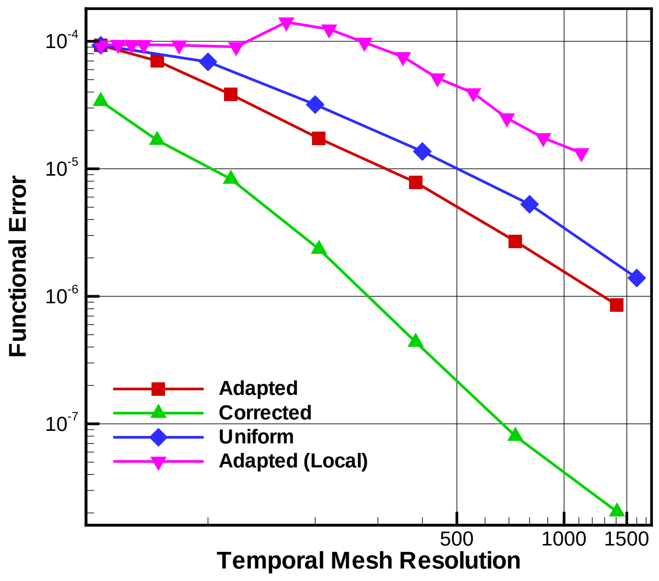


(b) Time varying lift coefficient for prescribed displacement

Figure 6.32: Prescribed displacement and the corresponding variation of the lift coefficient of the NACA64A010 airfoil used in the non-time-integrated functional example of temporal adaptation.

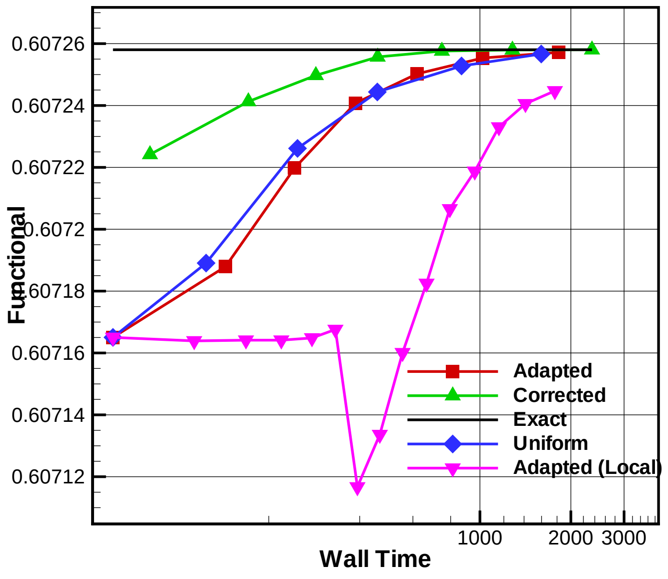


(a) Functional convergence

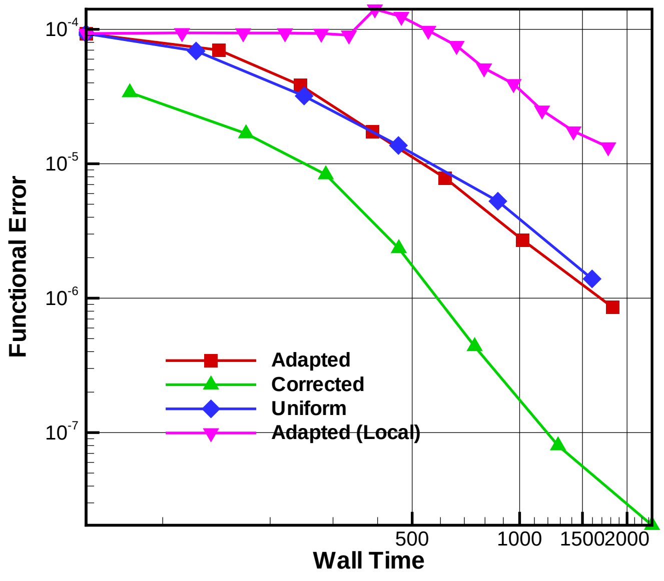


(b) Functional error convergence

Figure 6.33: Convergence of the functional and the error with respect to the temporal mesh resolution for the non-time-integrated functional example of temporal adaptation.

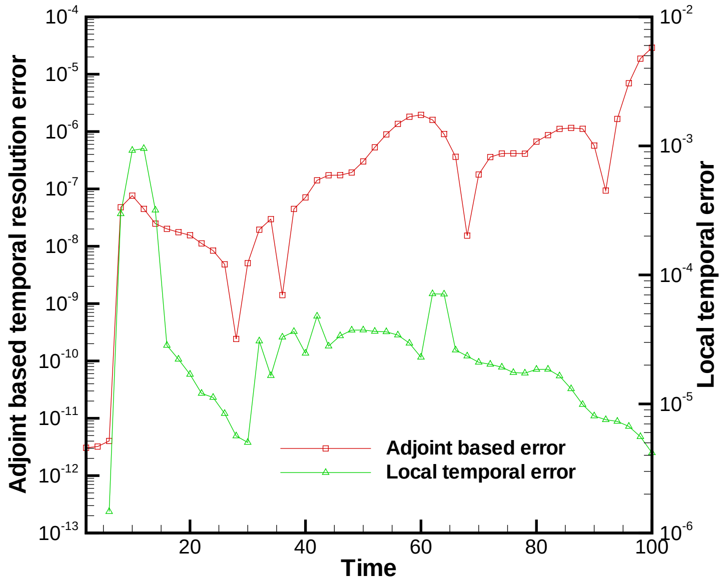


(a) Functional convergence

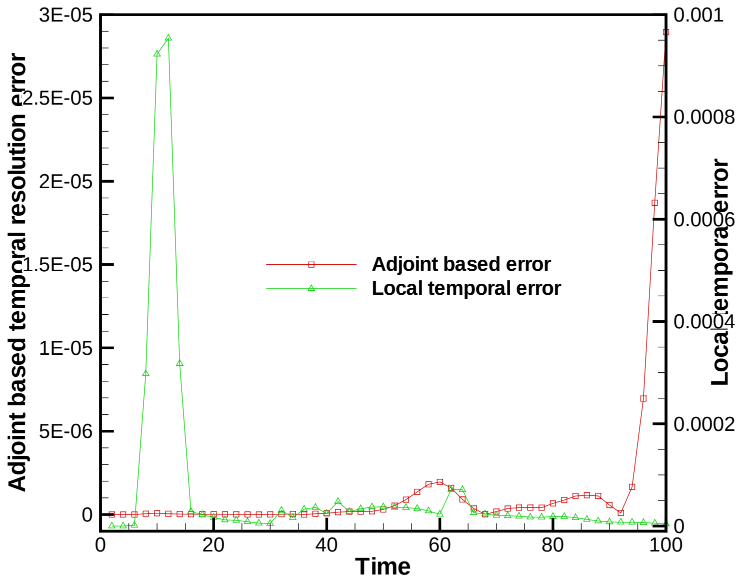


(b) Functional error convergence

Figure 6.34: Convergence of the functional and the error with respect to cost in terms of wall time for the non-time-integrated functional example of temporal adaptation.



(a) Log scale



(b) Linear scale

Figure 6.35: Comparison of the adjoint-based error distribution and the local temporal error distribution for the non-time-integrated functional example of temporal adaptation. Shown here on the coarsest temporal mesh consisting of 50 uniform time-steps.

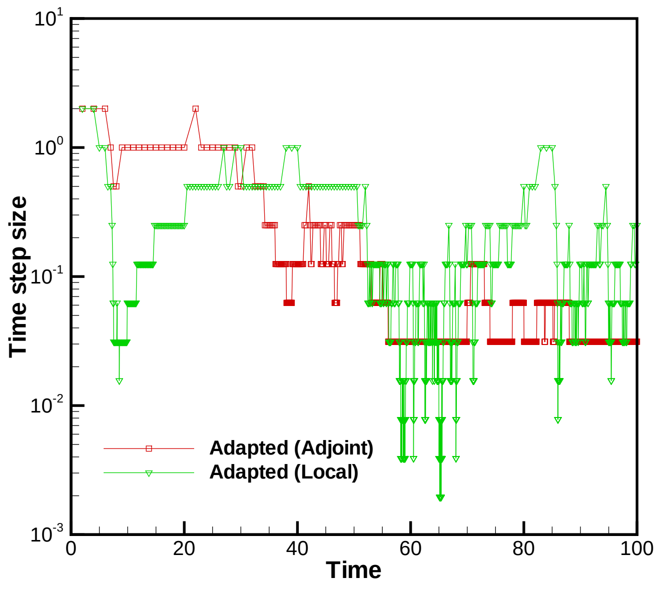
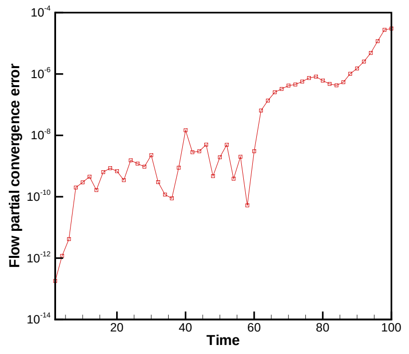
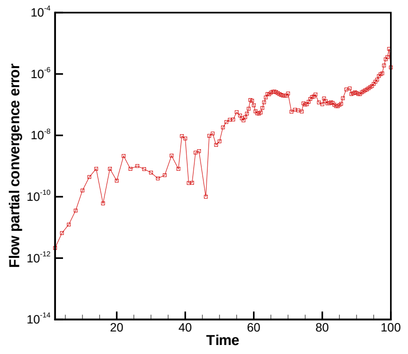


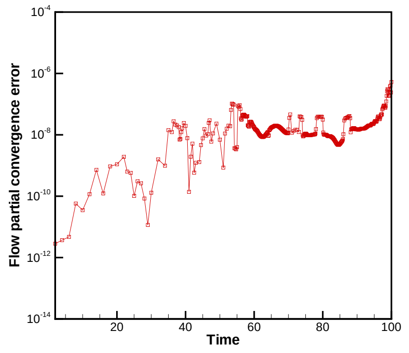
Figure 6.36: Comparison of the time-step size distribution for adjoint-based and local error-based adaptation after the final adaptation cycle for the non-time-integrated functional example of temporal adaptation.



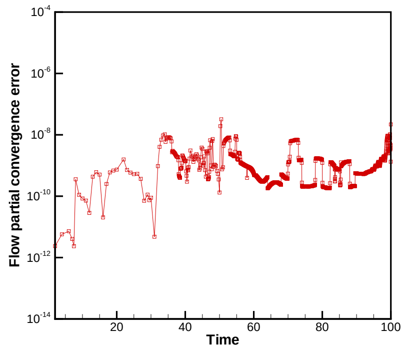
(a) 1st adaptation Cycle



(b) 3rd adaptation Cycle

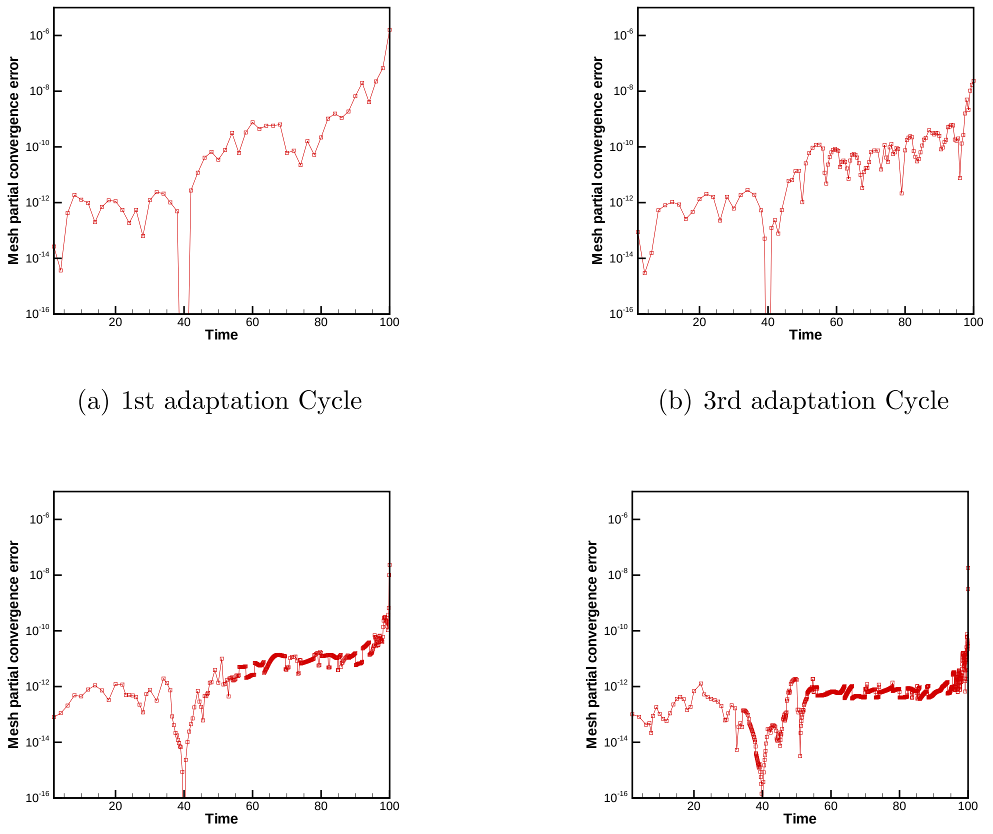


(c) 5th adaptation Cycle



(d) 7th adaptation Cycle

Figure 6.37: Flow partial convergence error distribution for the non-time-integrated functional example of temporal adaptation.



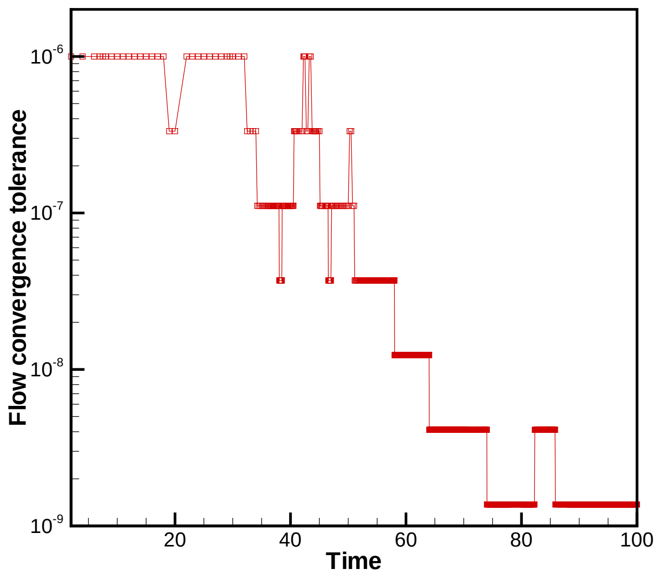
(a) 1st adaptation Cycle

(b) 3rd adaptation Cycle

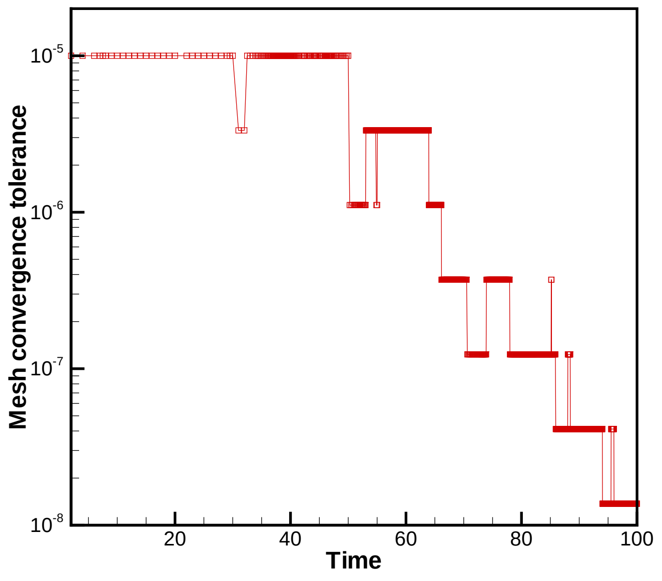
(c) 5th adaptation Cycle

(d) 7th adaptation Cycle

Figure 6.38: Mesh partial convergence error distribution for the non-time-integrated functional example of temporal adaptation.



(a) Flow convergence tolerance



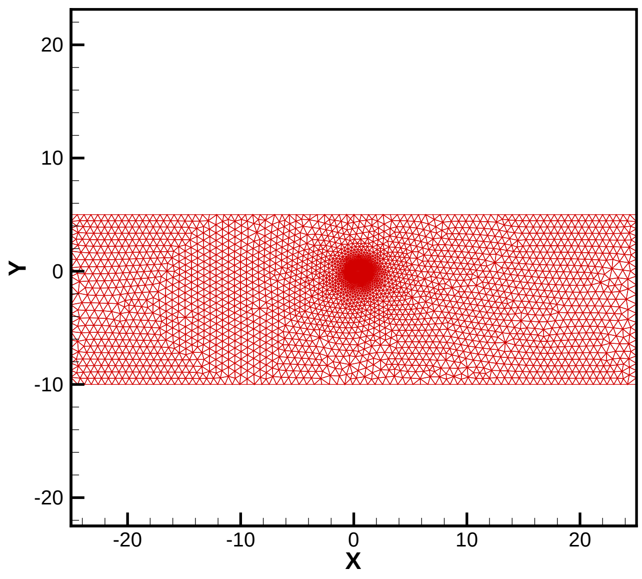
(b) Mesh convergence tolerance

Figure 6.39: Flow and mesh convergence tolerances after the final adaptation cycle for the non-time-integrated functional example of temporal adaptation.

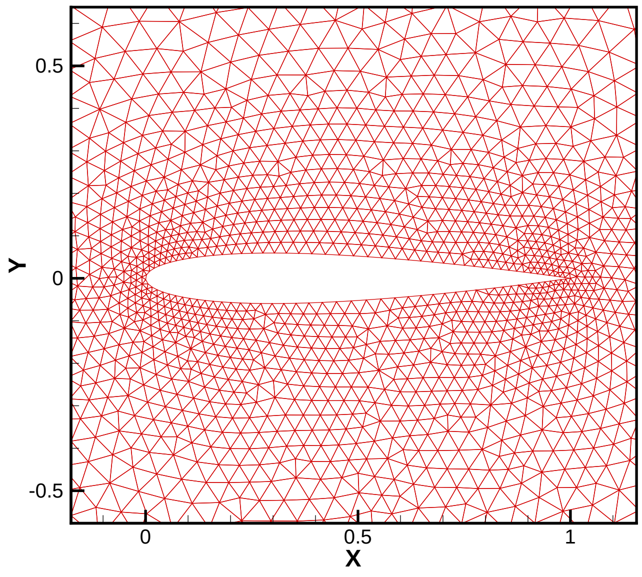
view of the airfoil within the mesh used for this example. The mesh consists of approximately 8,600 elements. The vortex is seeded in the flow 15 chord lengths ahead and 2 chord lengths below the airfoil, from where it is allowed to convect with the freestream and interact with the airfoil. Figure 6.41 shows the density contours of the initial condition for the flow in the spatial domain. Figure 6.42 shows the time variation of the lift coefficient of the airfoil with and without vortex interaction. The size of the time domain was chosen such that at the end of the time-integration process the vortex is located approximately 15 chord lengths downstream of the airfoil. The initial temporal mesh consists of 100 uniform time-steps and the starting convergence tolerances for the flow and mesh equations are set at $1e^{-6}$ and $1e^{-5}$ respectively. The functional of interest is the time-integrated value of the lift coefficient as described in equation (5.5). The effect of the vortex on the airfoil should in theory increase as it approaches the airfoil and diminish rapidly once it passes the airfoil. Engineering judgment tells us that, in order to correctly predict the effect of the vortex on the airfoil load, it must be convected to the region of interaction with the least amount of dissipation. Both temporal and spatial errors contribute toward dissipating the vortex as it convects through the domain. The sources of spatial error include the spatial discretization order-of-accuracy and the resolution of the spatial mesh. However, since all comparisons are made on the same spatial mesh with identical spatial discretization, the spatial error can be considered to be independent of the temporal error. The goal is to identify and reduce the effect of temporal error on the functional. The temporal error relevant to the functional most likely occurs in the region of the temporal domain when the vortex is still ahead of the airfoil, and also in the region when it interacts with the airfoil. It is likely that the temporal error once the vortex passes the airfoil is not relevant to the functional. The adaptive time-integration therefore can be expected to use smaller time-steps while the vortex is still ahead of the airfoil in order to ensure minimal dissipation of the vortex, and smaller time-steps have to be used during the interaction phase in order to ensure that all of the details in the perturbation of the airfoil lift are captured. The adjoint method being goal based should predict the error distribution in accordance with this expectation, while local error-based adaptation will likely request refinement of the entire time domain. This happens as a consequence of the vortex continuing to convect and dissipate downstream of the airfoil, although its effect is not relevant to the load on the airfoil. Therefore local error-based adaptation is likely to perform no better than uniform refinement of the whole time domain. The error tolerance was set at $1e^{-4}$ for the termination of adaptation for this case.

Figures 6.43(a) and 6.43(b) show the convergence of the functional and functional error with respect to the temporal mesh resolution, while Figures 6.44(a) and 6.44(b) compare these same quantities with the computational expense. For this particular case, adjoint-based adaptation is able to achieve similar error levels as uniform refinement and local error-based adaptation with fewer than half the number of time-steps. From a cost perspective, the plots indicate that initially the expense is similar between all methods but after two to three adaptation cycles, adjoint-based adaptation is able to outperform uniform refinement and local error-based adaptation. The plots also indicate the expected trend of local error-based adaptation performing similarly to uniform refinement. Figure 6.45 compares the distribution of the temporal error computed using the adjoint and the local error method after three adaptation cycles. It is quite clear that the local error distribution is fairly uniform throughout the time domain in contrast to the adjoint-based error where dominance is obvious prior to the vortex passing the airfoil. Figure 6.46 compares the time-step size distribution between adjoint-based and the local error-based method after termination of the adaptation. The adjoint-based method has not touched the region in the time domain when the vortex has passed the airfoil, while local error-based adaptation has requested fairly small time-step sizes in this region.

The algebraic error distribution due to partial convergence of the governing equations is particularly interesting for this problem. From Figures 6.47 and 6.48 it can be seen that the maximum algebraic error due to the flow equations at the start of the adaptation occurs when the vortex interacts with the airfoil, while the maximum algebraic error due to the mesh equations occurs not only during the interaction of the vortex with the airfoil but even after the vortex passes the airfoil. By far the most counterintuitive results are the algebraic error distributions for both the flow and mesh equations after the termination of adaptation. Both sets of equations continue to show higher error when the vortex is already downstream of the airfoil although adaptation of the convergence tolerances has occurred mostly in this region as shown in Figure 6.49. This is a good example where engineering judgment in lieu of a mathematically rigorous procedure to determine convergence tolerances could potentially lead to poor results.



(a) Complete domain



(b) Zoomed in view of airfoil within domain

Figure 6.40: Computational mesh consisting of approximately 8,600 elements used for the time-integrated functional example of temporal adaptation.

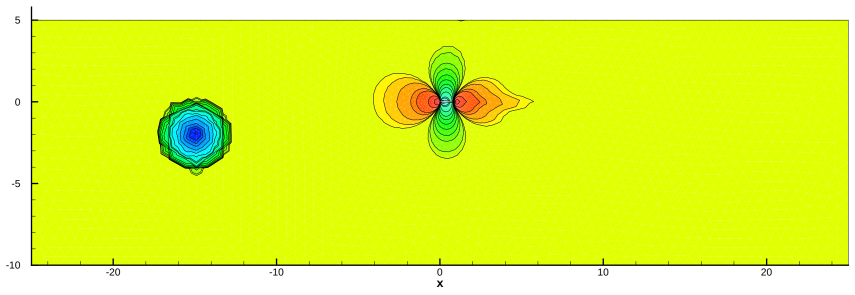


Figure 6.41: Density contours of initial condition for the time-integrated functional example of temporal adaptation.

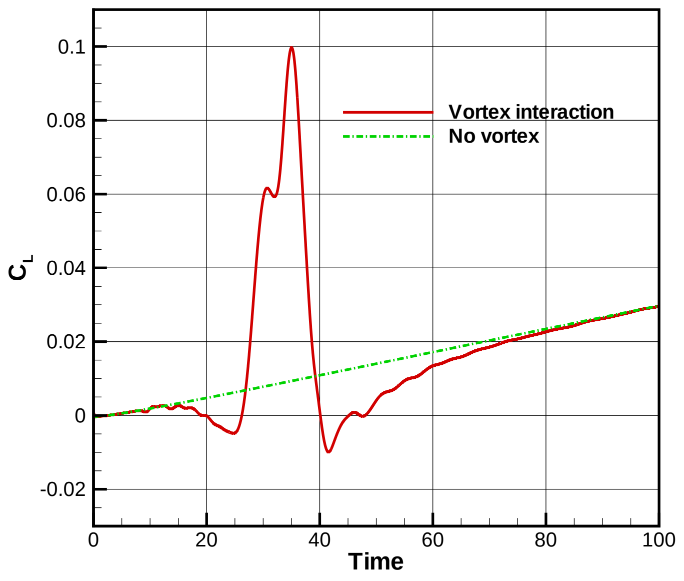
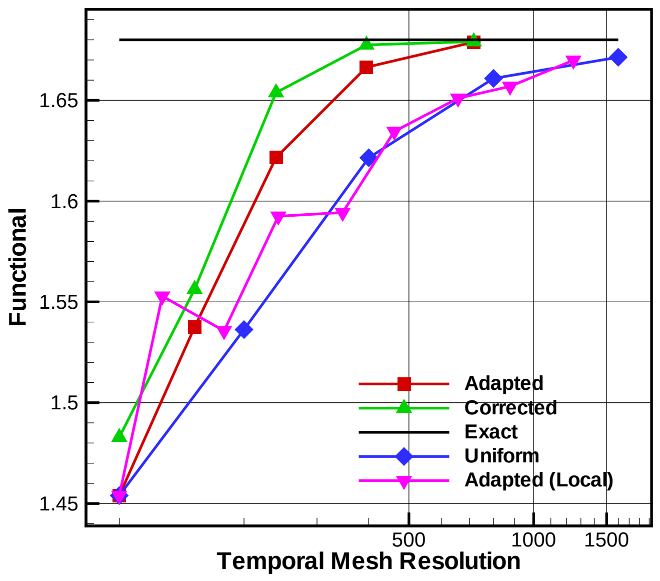
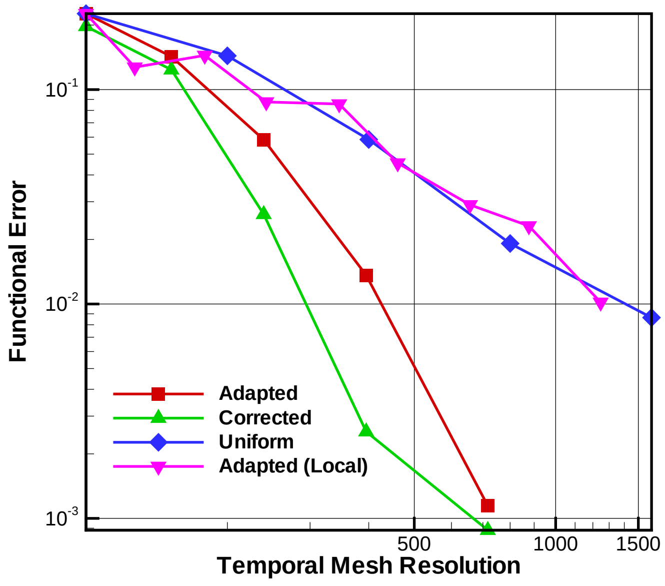


Figure 6.42: Time variation of the airfoil lift coefficient for the time-integrated functional example of temporal adaptation. The variation of the lift with and without the effect of the convecting vortex are shown.

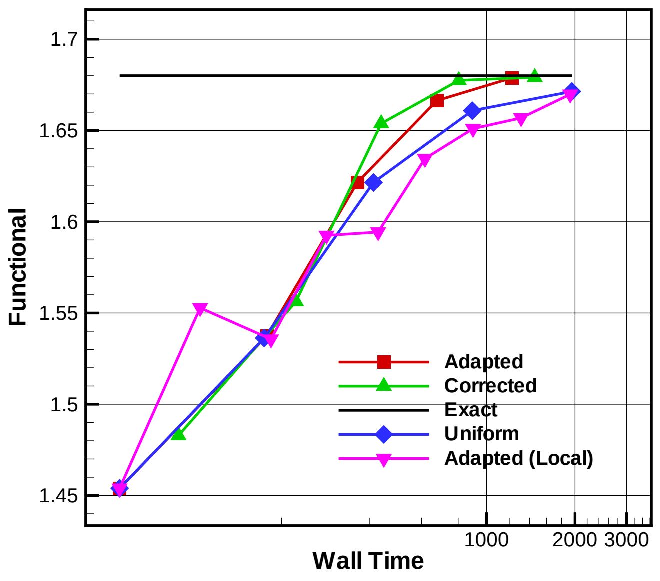


(a) Functional convergence

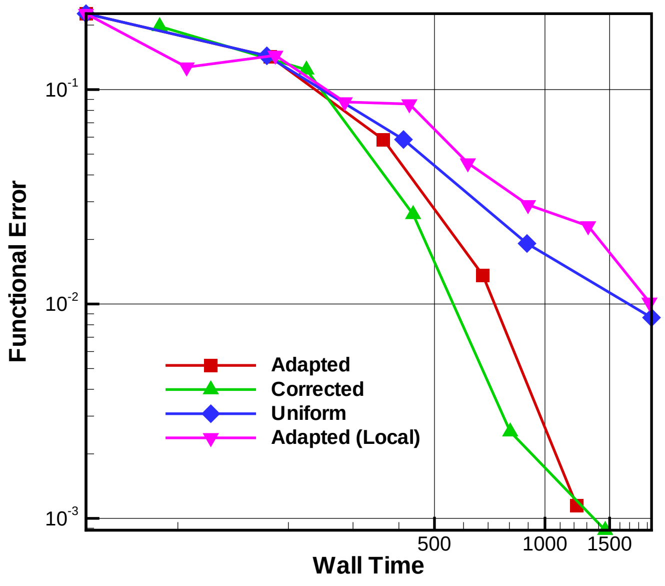


(b) Functional error convergence

Figure 6.43: Convergence of the functional and the error with respect to the temporal mesh resolution for the time-integrated functional example of temporal adaptation.

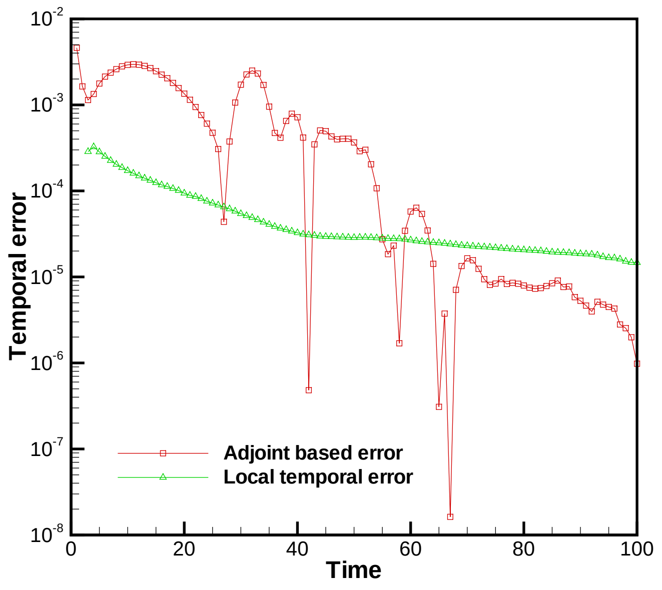


(a) Functional convergence

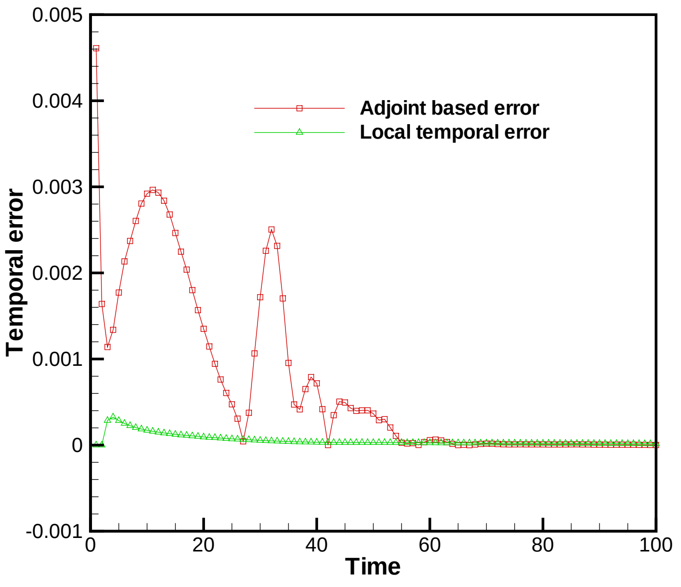


(b) Functional error convergence

Figure 6.44: Convergence of the functional and the error with respect to cost in terms of wall time for the time-integrated functional example of temporal adaptation.



(a) Log scale



(b) Linear scale

Figure 6.45: Comparison of the adjoint-based error distribution and the local temporal error distribution for the time-integrated functional example of temporal adaptation. Shown here on the coarsest temporal mesh consisting of 100 uniform time-steps.

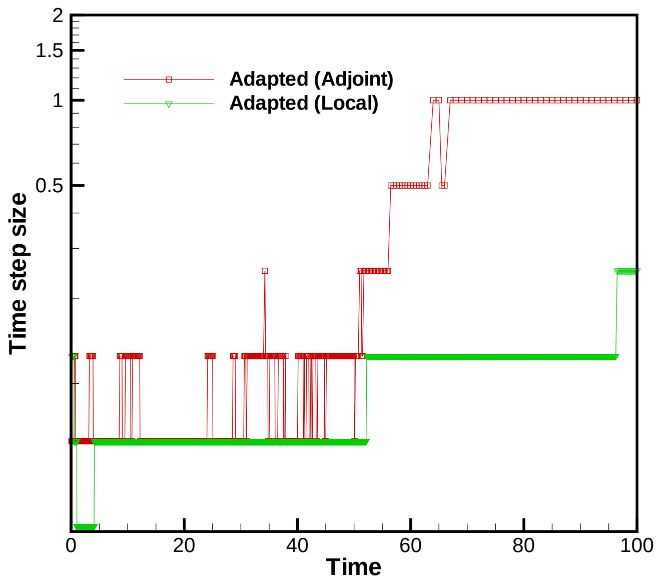
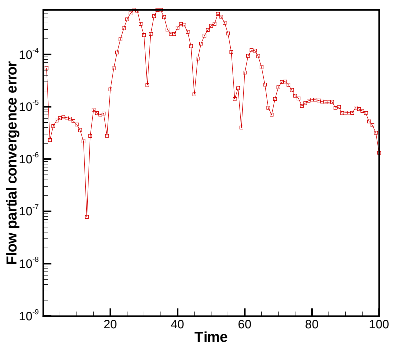
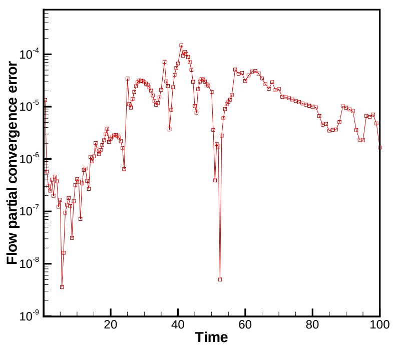


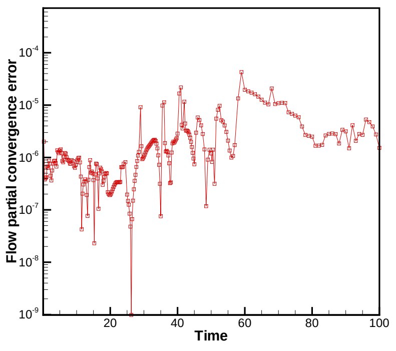
Figure 6.46: Comparison of the time-step size distribution for adjoint-based and local error-based adaptation after the final adaptation cycle for the time-integrated functional example of temporal adaptation.



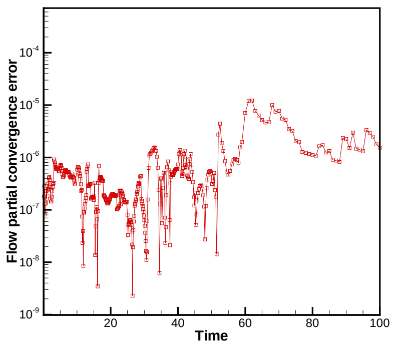
(a) 1st adaptation Cycle



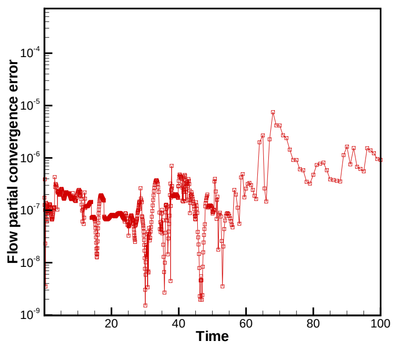
(b) 2nd adaptation Cycle



(c) 3rd adaptation Cycle

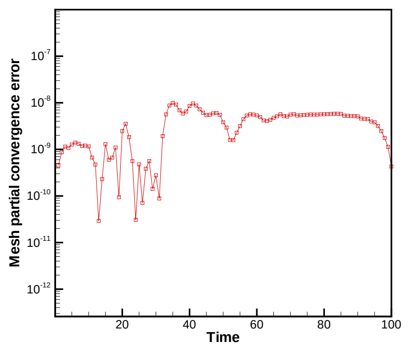


(d) 4th adaptation Cycle

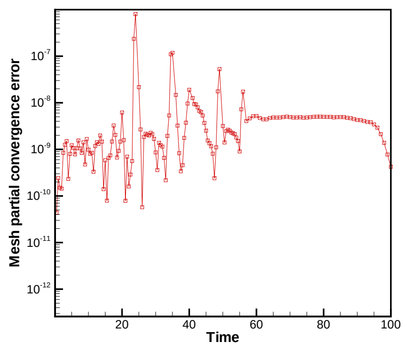


(e) 5th adaptation Cycle

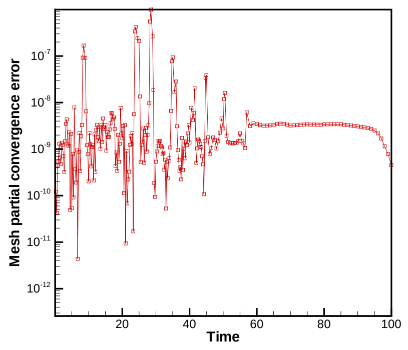
Figure 6.47: Flow partial convergence error distribution for the time-integrated functional example of temporal adaptation.



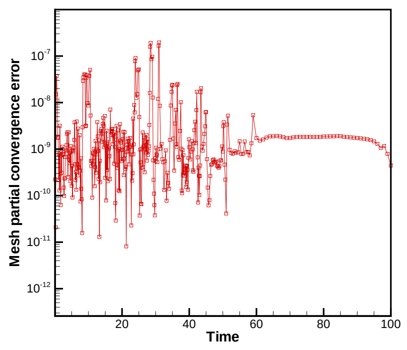
(a) 1st adaptation Cycle



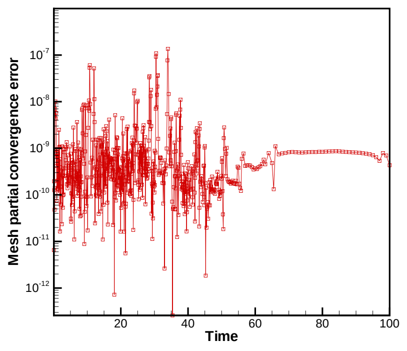
(b) 2nd adaptation Cycle



(c) 3rd adaptation Cycle

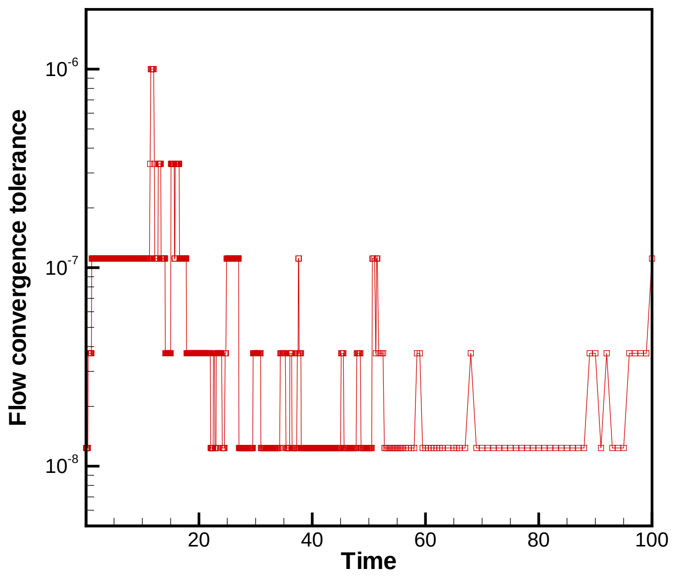


(d) 4th adaptation Cycle

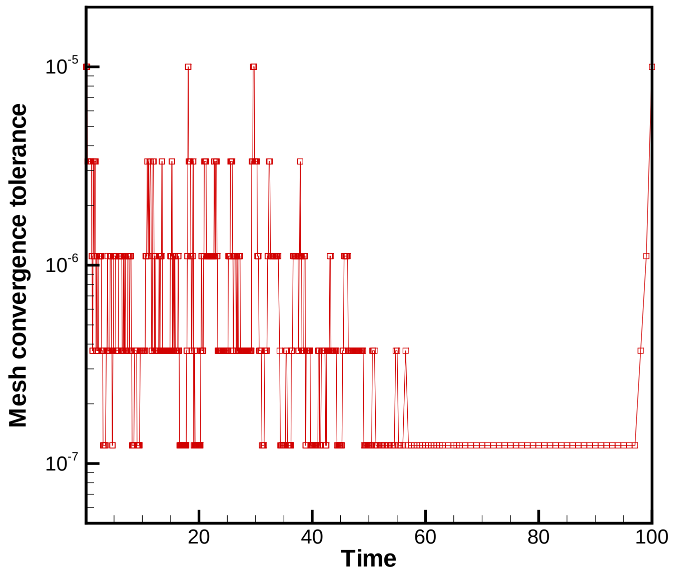


(e) 5th adaptation Cycle

Figure 6.48: Mesh partial convergence error distribution for the time-integrated functional example of temporal adaptation.



(a) Flow convergence tolerance



(b) Mesh convergence tolerance

Figure 6.49: Flow and mesh convergence tolerances after the final adaptation cycle for the time-integrated functional example of temporal adaptation.

Chapter 7

Conclusions

7.1 Summary

This thesis presented a unified approach to computing adjoint variables in the context of aerodynamic simulations. The method was shown to be general enough such that it may be applied to multiple coupled or uncoupled disciplines, which may be steady or unsteady in nature. The computed adjoint variables were used for the dual purpose of computing functional gradients with respect to design variables for use in shape optimization problems and also for the estimation of temporal discretization and algebraic errors in the numerical solution of the governing equations.

The use of adjoint-based gradients for aerodynamic shape optimization was demonstrated in the context of coupled unsteady aeroelasticity, uncoupled prescribed unsteady, and steady-state flow problems. Additionally, the adjoint variables computed using the presented approach were applied as weights on the local residual errors for the purpose of estimating functional relevant temporal discretization and algebraic errors. Adaptation of the time-step sizes, the time-step distribution, and adaptation of the convergence tolerances for all disciplines at each time-step was then demonstrated for unsteady flow problems with prescribed geometric displacements. The method was shown to outperform traditional methods in terms of computational expense and the number of time-steps by reducing the error in the functional of interest for both cases of a time-integrated and a non-time-integrated functional.

7.2 Contributions to the Field and Potential Applications

The contributions to the field made by the current work can be summarized as follows:

- **Rapid determination of the gradient vector for multiple disciplines**

  Although the adjoint method has been used quite successfully to determine the gradient vector for use in shape optimization especially for steady-state problems, the work presented in the thesis is likely the first presentation of a single treatment for all types of problems. The method is general enough such that coupled or uncoupled and steady or unsteady nonlinear disciplines can be addressed in a unified manner regardless of the number of disciplines or the number of design variables. The method taps into the advantage that adjoint variables reduce the cost of obtaining the complete gradient vector to approximately that of obtaining the solution to the nonlinear analysis problem.

- **Applications in aerodynamic shape design**

  The presented work is the first attempt to determine gradients using the adjoint method for coupled unsteady aeroelasticity problems with no simplifying assumptions. Although demonstrated in the context of a simple two-dimensional flutter model using inviscid flow, this sets the stage for design optimization in more sophisticated problems such as flutter in aircraft wings, blade deformations in helicopters and wind turbines, and rotor-stator aeroelastic interactions in turbo-machines.

- **Applications in model parameter sensitivities**

While the general push is toward eliminating the use of simplifying models in numerical simulations, several multiphysics problems such as reactor simulations and hypersonic aerodynamic flows still rely on such models to reduce the overall computational cost. Even with simpler problems such as high Reynolds number aerodynamic flows, only the RANS equations are solved and a turbulence model is used to simulate the turbulent characteristics of the boundary layer. There always remains the question of how much of an effect empirically determined parameters in such models have on functionals of interest. The computation of adjoint variables in such problems permits the determination of functional sensitivity to model parameters and may be used for fine tuning of the model.

- **Improved unsteady simulation capability**

  Much of the previous work in adapting the temporal domain resolution has been done using local error estimates. Goal-based adaptation of the temporal resolution has thus far been demonstrated only in the context of simple ODEs and PDEs. To the author's knowledge, the estimation of the functional relevant temporal discretization error, and the decomposition of the error into components arising from each discipline at each time-step is novel. This is perhaps the first attempt at adapting the temporal domain for realistic flow problems. In many time-dependent simulations, the temporal resolution required for adequately capturing flow details is unknown and the user has to rely on engineering judgment and repeated restarts with increased resolution in certain regions of the time domain. Such an approach requires intensive user involvement in the simulation process while providing very little improvement to functionals of interest. Automatic adaptation of the temporal domain using the adjoint-based error not only directly targets the functional accuracy, but also reduces the user involvement in the overall simulation process. The current work provides the necessary framework to develop "push-button" type unsteady solvers which require minimum user interaction.

- **Compensation for algebraic error in the solution**

  Other than the acknowledgment of the existence of algebraic error in numerical solutions, there has been little effort in developing methods for its quantification and reduction. For simpler problems, where solution procedure stiffness and computational expense are not issues, algebraic error can be avoided by solving the equations to machine accuracy. However, in several cases this may either not be feasible or possible. The solution process may be terminated early due to computational cost, or due to stalling of the convergence rate. The thesis presents a uniquely rigorous procedure to quantify (up to a linear approximation) the algebraic error and reduce it by adapting the convergence tolerances, or compensate for it by providing a functional correction.

7.3 Recommendations for Future Work

- **Extension to three-dimensional problems**

  The use of adjoint equations for steady-state shape optimization has been explored in three-dimensional problems. However, unsteady and coupled unsteady optimization in three dimensions remains unpopular due to cost and storage requirements. For unsteady problems, the time history has to be stored and read in by the adjoint solver since it performs a backward sweep in time.Typical three-dimensional unsteady flow problems that run in parallel with approximately 200,000 elements per processor for about 2,000 time steps would require roughly 75 gigabytes of hard-drive space per processor to hold the solution history. Such space requirements can be easily accommodated at low cost on todays computational clusters. Also, since the solution set is written piece-by-piece at the conclusion of each time-step in the time-integration process, the speed of the I/O operation itself has little impact on the overall speed of computation. The other factor weighing in on feasibility when extended to large cases is the number of design iterations required to reach an optimum design. To this end, future work will have to explore the use of advanced optimization algorithms. Similarly, adjoint-based adaptation of the spatial domain has been performed in three dimensions, but adaptation of the temporal domain in unsteady simulations has not been addressed for realistic three-dimensional flow problems. Considering that adjoint-based adaptation of the temporal domain offers reduction in the overall cost of unsteady simulations, this is an avenue worth exploring. Also, extension of the aeroelastic sensitivity problem to three dimensions will require the additional step of linearizing the transfer of forces and displacements between fluids and structures grids, which is trivial for the two-dimensional case.

- **Analysis of error due to coupling between disciplines**

  Generally, fully coupled solutions are more expensive than uncoupled simulations due to the iterative strategy employed for such systems. While it may be known that disciplines are coupled and require a coupled iterative solution procedure, it is often not known to what extent the coupling is relevant to functionals of interest. As an example, considering the aeroelasticity problem presented in this thesis, it may be possible to employ a faster loosely coupled solution procedure over a tightly coupled iterative method, to solve the flow, mesh and structure equations at each time step without significantly degrading the functional accuracy, provided the error due to coupling can be quantified. While a procedure to estimate such coupling error has not been presented in this thesis, it is possible to build on the current framework for this purpose. Preliminary research indicates that the definitions of the adjoint equations require minimal modifications to reflect the functional relevant coupling error. It is expected that the error will continue to take up the form of an inner product between a nonzero residual and the corresponding adjoint variable.

- **Spatially-local temporal domain adaptation**

  The derivation presented in this thesis for the computation of temporal discretization error, permits the decomposition of such error into a spatial distribution at each time step. However, the results presented made no effort to use such an error distribution to identify regions in the spatial domain where refinement of the temporal domain could possibly be avoided. That is to say that, all spatial elements were advanced at each time-step by the same time-step size, irrespective of the spatially local contribution to the overall temporal discretization error in the functional. Preliminary research indicates significant non-uniformity in the the spatial distribution of the temporal discretization error for the example problems presented. Additional reduction of computational expense over spatially uniform temporal domain adaptation may be realized by developing techniques to perform spatially non-uniform temporal domain adaptation. In essence this amounts to local time-stepping in unsteady problems, where spatial elements in the computational mesh are advanced in time at different rates. The key issue to be addressed in order to enable such a method is the computation of the spatial convective flux at the interface between neighboring spatial elements that are at different temporal locations.

- **Combined spatial and temporal domain adaptation**

  Another area of interest concerns the combination of spatial and temporal adaptation for functional outputs. Considering the results demonstrated in the thesis combined with existing work that has been done on spatial adaptation, the machinery now exists to adapt both the spatial and time domains independently. The obvious path from this point would be to address issues that arise from the coupling of spatial and temporal adaptation. The full error control problem involves the simultaneous targeting of spatial discretization, temporal discretization, algebraic, and coupling errors. If simultaneous goal-based adaptation using such error estimates is possible, it would ultimately lead to highly efficient solvers, which could begin with coarse meshes, adapt on-the-fly and provide an accurate functional to within some error tolerance with almost no human interaction.

Appendix A

Third-order accurate BDF3 temporal discretization with uniform and non-uniform temporal mesh spacing

A.1 BDF3 with Uniform Spacing

The discretization of the time derivative term in the governing flow equations using a third-order accurate BDF3 time-integration scheme requires a four-point stencil in time and is given as:

$$\left(\frac{dA\mathbf{U}}{dt}\right)^n = \left(\frac{1}{\Delta t}\right)\left[\frac{11}{6}(A\mathbf{U})^n - 3(A\mathbf{U})^{n-1} + \frac{3}{2}(A\mathbf{U})^{n-2} - \frac{1}{3}(A\mathbf{U})^{n-3}\right]$$

A.2 BDF3 with Non-Uniform Spacing

The third-order accurate BDF3 time-integration scheme for non-uniform time-step sizes can be derived starting with different Taylor expansions of the solution at time-step index n and relaxing the assumption of uniform mesh spacing. Using a four-point stencil in time, separated by three time-steps of sizes Δt_1,Δt_2, and Δt_3, the discretization is given as:

$$\begin{aligned}
\left(\frac{dA\mathbf{U}}{dt}\right)^n &= -\left(\frac{C_2}{C_4}\right)(A\mathbf{U})^n - \left(\frac{C_3}{C_4}\right)(A\mathbf{U})^{n-1} + \left(\frac{1}{C_4}\right)(A\mathbf{U})^{n-2} - \left(\frac{C_1}{C_4}\right)(A\mathbf{U})^{n-3} \\
C_1 &= \frac{k_2^2(k_1-k_2)}{k_3^2(k_1-k_3)} \\
C_2 &= (1-C_1) + \frac{C_1 k_3^3 - k_2^3}{k_1^3} \\
C_3 &= \frac{k_2^3 - C_1 k_3^3}{k_1^3} \\
C_4 &= \frac{k_2(k_2^2-k_1^2) + C_1 k_3(k_1^2-k_3^2)}{k_1^2} \\
k_1 &= \Delta t_1 \\
k_2 &= \Delta t_1 + \Delta t_2 \\
k_3 &= \Delta t_1 + \Delta t_2 + \Delta t_3 \\
\Delta t_1 &= T^n - T^{n-1} \\
\Delta t_2 &= T^{n-1} - T^{n-2} \\
\Delta t_3 &= T^{n-2} - T^{n-3} \\
T &= Time
\end{aligned}$$

Appendix B

Linear Multigrid Methodology

A linear geometric agglomeration multigrid [1] strategy is used to accelerate convergence of all linear systems encountered in the solution process, be either, the flow, mesh motion, flow adjoint or mesh adjoint equations. The linear multigrid method uses a correction scheme where coarser levels recursively provide corrections to the fine grid solution. The coarse level meshes are built by repeated agglomeration or merging of neighboring control volumes to form a single control volume. Figures B.1(a) through B.1(f) show an unstructured triangular mesh around a four-element airfoil and the agglomerated coarser levels.

Consider a linear system developed by discretizing and linearizing a non-linear equation on the fine level mesh.

$$[A]_h \mathbf{x}_h = \mathbf{b}_h \tag{B.1}$$

where as notation for this section of the appendix, the subscripts h and H refer to a spatial mesh of some arbitrary resolution and an agglomerated coarser spatial mesh. An approximate solution to the linear system can be obtained by using conventional iterative methods such as Jacobi or Gauss-Seidel iterations. The residual of the linear system can then be written as

$$[A]_h \bar{\mathbf{x}}_h - \mathbf{b}_h = \mathbf{r}_h \tag{B.2}$$

where $\bar{\mathbf{x}}_h$ is the approximate solution to the linear system. A correction $\mathbf{x}'_h$ to the approximate solution is defined such that the residual of the linear system on the fine level mesh goes to zero.

$$\mathbf{x}_h = \bar{\mathbf{x}}_h + \mathbf{x}'_h \tag{B.3}$$

By taking the difference between the correction equation and the residual equation on the fine mesh, a new linear system is defined as

$$[A]_h \mathbf{x}'_h = -\mathbf{r}_h \tag{B.4}$$

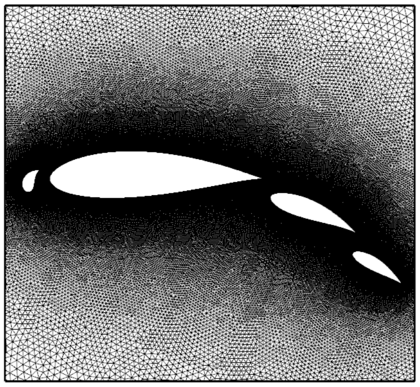
Transferring this equation to a coarse level to form a coarse level correction equation yields

$$[A]_H \mathbf{x}'_h = -I_h^H \mathbf{r}_h \tag{B.5}$$

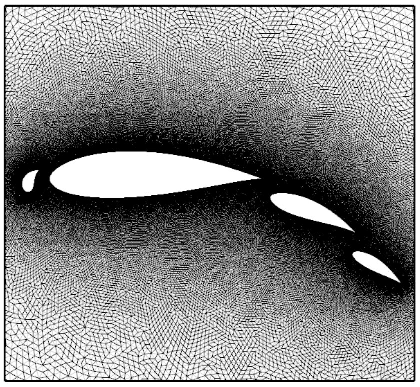
where the subscript H refers to the coarse level and I_h^H is the fine-to-coarse level restriction operator. This system can now be iteratively solved approximately or exactly if H is coarse enough. Alternatively, if H is not sufficiently coarse, a new approximate solution is obtained and the residual of this system is then transferred to a coarser level. This process is repeated until H is sufficiently coarse to obtain a numerically exact solution to the system using an iterative method. The exact or approximate solutions obtained from the coarse levels are used to correct the fine grid solution as

$$\bar{\mathbf{x}}_h^{new} = \bar{\mathbf{x}}_h + I_H^h \mathbf{x}'_H \tag{B.6}$$

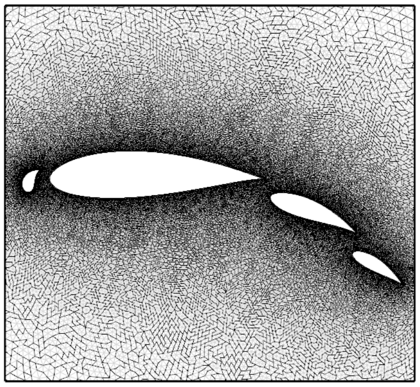
where I_H^h refers to the coarse-to-fine grid prolongation operator. In the analysis problem of the current work, summation of parent residuals is used as the restriction operator for the right-hand-side of the linear system. For the flow problem, the coefficient matrix $[A]_H$ (i.e. flow Jacobian matrix) for the coarse levels are constructed by first restricting the fine level flow solution through a parent element area weighted approach, and then using the restricted flow solution to build the flow Jacobian matrix. For all cases, direct injection of coarse level corrections into parent elements is used as the prolongation operator.



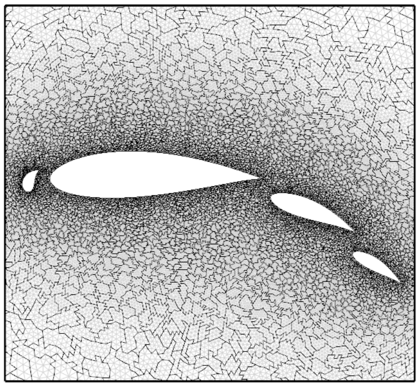
(a) Level 0



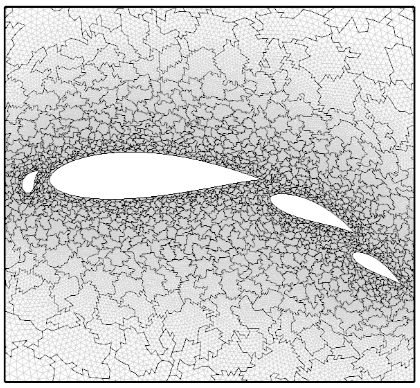
(b) Level 1



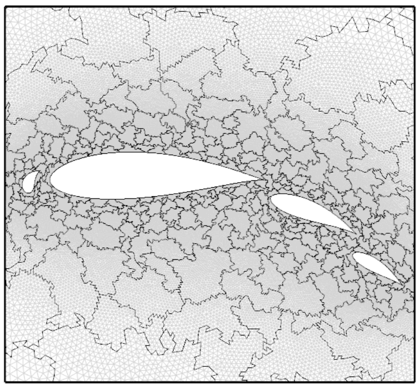
(c) Level 2



(d) Level 3



(e) Level 4



(f) Level 5

Figure B.1: Agglomerated multigrid levels for an unstructured mesh around a 4-element airfoil.

References

[1] Mavriplis, D., "Multigrid Techniques for Unstructured Meshes," *Notes prepared for 26th Computational Fluid Dynamics Lecture Series Program of the von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium*, 1995.

[2] Melson, N. D., Sanetrik, M. D., and Atkins, H. L., "Time-accurate Navier-Stokes calculations with multigrid acceleration," *6th Copper Mountain Conf. on Multigrid Methods*, 1993, pp. 423–439, NASA Conference Publication 3224.

[3] Bijl, H., Carpenter, M. H., and Vatsa, V. N., "Time integration schemes for the unsteady Navier-Stokes equations," *15th AIAA Computational Fluid Dynamics Conference, Anaheim, California*, 2001, AIAA Paper 2001–2612.

[4] Jothiprasad, G., Mavriplis, D. J., and Caughey, D., "Higher Order Time Integration Schemes for the Unsteady Navier-Stokes Equations on Unstructured Meshes," *32nd AIAA Fluid Dynamics Conference and Exhibit, St. Louis, Missouri*, 2002, AIAA Paper 2002–2734.

[5] Nadarajah, S., McMullen, M., and Jameson, A., "Non-Linear Frequency Domain Based Optimum Shape Design for Unsteady Three-Dimensional Flow," *Proceedings of the 44th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2006, AIAA Paper 2006–387.

[6] Nadarajah, S., McMullen, M., and Jameson, A., "Optimal Control of Unsteady Flows Using Time Accuraate and Non-Linear Frequency Domain Methods," *33rd AIAA Fluid Dynamics Conference and Exhibit, Orlando, FL, June 23-26,*, 2003, AIAA Paper 2003–3875.

[7] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28-8, August 1990, pp. 1381–1388.

[8] Baker, T. J. and Cavallo, P. A., "Dynamic adaptation for deforming tetrahedral meshes," *14th AIAA Computational Fluid Dynamics Conference, Norfolk, VA*, 1999, AIAA Paper 1999–3253.

[9] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional springs for two-dimensional dynamic unstructured fluid meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 163, No. 1-4, 1998, pp. 231–245.

[10] Johnson, A. A. and Tezduyar, T. E., "Simulation of multiple spheres falling in a liquid-filled tube," *Computer Methods in Applied Mechanics and Engineering*, Vol. 134, No. 3-4, 1996, pp. 351–373.

[11] Nielsen, E. and Anderson, W., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40-6, June 2002, pp. 1155–1163.

[12] Geuzaine, P., Grandmont, C., and Farhat, C., "Design and Analysis of ALE Schemes with Provable Second-Order Time-Accuracy for Inviscid and Viscous Flow Simulations," *Journal of Computational Physics*, Vol. 191, 2003, pp. 206–227.

[13] Mavriplis, D. and Yang, Z., "Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes," *Journal of Computational Physics*, Vol. 213, 2006, pp. 557–573.

[14] Balsara, D. and Shu, C.-W., "Monotonicity Preserving Weighted Essentially Non-oscillatory Schemes with Increasingly High Order of Accuracy," *Journal of Computational Physics*, Vol. 160, No. 2, 2000, pp. 405–452.

[15] Kroll, N., Heinrich, R., Krueger, W., and Nagel, B., "Fluid-Structure Coupling for Aerodynamic Analysis and Design: A DLR Perspective (Invited)," *46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, 2008, AIAA Paper 2008–561.

[16] Vendetti, D. and Darmofal, D., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, 2002, pp. 40–69.

[17] Vendetti, D. and Darmofal, D., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, 2003, pp. 22–46.

[18] Mavriplis, D. J., "Adaptive Meshing Techniques for Viscous Flow Calculations on Mixed Element Unstructured Meshes," *International Journal for Numerical Methods in Fluids*, Vol. 34, No. 2, 2000, pp. 93–111.

[19] Fidkowski, K. J. and Darmofal, D. L., "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 225, No. 2, 2007, pp. 1653–1672.

[20] Wang, L. and Mavriplis, D. J., "Adjoint Based h-p Adaptive Discontinuous Galerkin Methods for Aerospace Applications," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando, Florida*, 2009, AIAA Paper 2009–952.

[21] Rosendale, J. V., "Floating Shock Fitting via Lagrangian Adaptive Meshes," *ICASE Report No. 94-89*, November 1994.

[22] "iSight," Process Integration and Design Optimization Software, Engineous Software Inc.

[23] "ModelCenter," Process Integration and Design Optimization Software, Phoenix Integration Inc.

[24] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1989.

[25] Jameson, A., "Aerodynamic Shape Optimization using the Adjoint Method," *VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium*, 2003.

[26] Jameson, A. and Vassberg, J., "Computational Fluid Dynamics for Aerodynamic Design: Its Current and Future Impact," *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2001, AIAA Paper 2001–0538.

[27] Nadarajah, S. and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," *Proceedings of the 38th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2000, AIAA Paper 2000–0667.

[28] Jameson, A., Alonso, J., Reuther, J., Martinelli, L., and Vassberg, J., "Aerodynamic shape optimization techniques based on control theory," 1998, AIAA Paper 98–2538.

[29] Giles, M., Duta, M., and Muller, J., "Adjoint Code Developments Using Exact Discrete Approach," 2001, AIAA Paper 2001–2596.

[30] Mavriplis, D. J., "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44-1, January 2006, pp. 42–50.

[31] Mavriplis, D. J., "Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes," *AIAA Journal*, Vol. 45-4, April 2007, pp. 741–750.

[32] Elliott, J. and Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes," *AIAA Journal*, Vol. 35-9, 1997, pp. 1479–1485.

[33] Soto, R. and Yang, C., "An Adjoint-Based Design Methodology for CFD Optimization Problems," 2003, AIAA Paper 2003–0299.

[34] Ghayour, K. and Baysal, O., "Limit-Cycle Shape Optimization using Time-Dependent Transonic Equation," *Proceedings of the 14th Computational Fluid Dynamics Conference, Norfolk VA*, 1999, AIAA Paper 99–3375.

[35] Nadarajah, S. and Jameson, A., "Optimal Control of Unsteady Flows using A Time Accurate Method," *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference, Atlanta GA*, 2002, AIAA Paper 2002–5436.

[36] Rumpfkeil, M. and Zingg, D., "A General Framework for the Optimal Control of Unsteady Flows with Applications," *45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January*, 2007, AIAA Paper 2007–1128.

[37] Mani, K. and Mavriplis, D. J., "Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," *AIAA Journal*, Vol. 46-6, June 2008, pp. 1351–1364.

[38] Mavriplis, D. J., "Solution of the Unsteady Discrete Adjoint for Three-Dimensional Problems on Dynamically Deforming Unstructured Meshes," *Proceedings of the 46th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2008, AIAA Paper 2008–0727.

[39] Kirsch, U., *Structural Optimization: Fundamentals and Applications*, Springer, Berlin, 1993.

[40] Maute, K., Nikbay, M., and Farhat, C., "Coupled Analytical Sensitivity Analysis and Optimization of Three-Dimensional Nonlinear Aeroelastic Systems," *AIAA Journal*, Vol. 39, 2001, pp. 2051–2061.

[41] Barcelos, M. and Maute, K., "Aeroelastic design optimization for laminar and turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, No. 19-20, 2008, pp. 1813–1832.

[42] Martins, J., Alonso, J., and Reuther, J., "Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis," *CEAS Conference on Multidisciplinary Analysis and Optimization, Cologne, Germany, June 2001*, 2001.

[43] Willcox, K., *Reduced-Order Aerodynamic Models for Aeroelastic Control of Turbomachines*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2000.

[44] Palaniappan, K., Sahu, P., Alonso, J., and Jameson, A., "Design of Adjoint Based Laws for Wing Flutter Control," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando, Florida*, 2009, AIAA Paper 2009–148.

[45] Palaniappan, K., Sahu, P., Alonso, J., and Jameson, A., "Active Flutter Control using an Adjoint Method," *44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, 2006, AIAA Paper 2006–844.

[46] Giles, M. B. and Suli, E., "Adjoint Methods for PDEs: a posteriori error analysis and postprocessing by duality," *Acta Numerica*, 2002, pp. 145–236.

[47] Houston, P., Rannacher, R., and Suli, E., "A posteriori error analysis for stabilized finite-element approximations of transport problems," *Comput. Methods Appl. Mech. Engr.*, Vol. 190, 2000, pp. 1483–1508.

[48] Pierce, N. and Giles, M., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, No. 2, 2004, pp. 769–794.

[49] Giles, M., Duta, M., Muller, J.-D., and Pierce, N., "Algorithm developments for discrete adjoint methods," *AIAA Journal*, Vol. 41-2, 2003, pp. 198–205.

[50] Balasubramanian, R. and Newman III, J. C., "Comparison of Adjoint-based and Feature-based Grid Adaptation for Functional Outputs," *24th AIAA Applied Aerodynamics Conference, San Francisco, California*, 2006, AIAA Paper 2006–3314.

[51] Balasubramanian, R. and Newman III, J. C., "Adjoint-based error estimation and grid adaptation for functional outputs: Application to two-dimensional, inviscid, incompressible flows," *Computers and Fluids*, Vol. 38, No. 2, 2009, pp. 320–332.

[52] Nemec, M., Aftosmis, M. J., and Wintzer, M., "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries," *46th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2008, AIAA Paper 2008–725.

[53] Wintzer, M., Nemec, M., and Aftosmis, M. J., "Adjoint-Based Adaptive Mesh Refinement for Sonic Boom Prediction," *26th AIAA Applied Aerodynamics Conference, Honolulu, Hawaii*, 2008, AIAA Paper 2008–6593.

[54] Butcher, J. C., *Numerical methods for ordinary differential equations*, Wiley, Chicester, UK, 2003.

[55] Lambert, J. D., *Numerical methods for ordinary differential systems: The initial value problem*, Wiley, Chicester, UK, 1991.

[56] Hindmarsh, A. C. and Taylor, A. G., "PVODE and KINSOL: Parallel software for differential and nonlinear systems," UCRL–ID–129739, Lawrence Livermore National Laboratory.

[57] Estep, D., "A Posteriori Error Bounds and Global Error Control for Approximation of Ordinary Differential Equations," *SIAM Journal of Numerical Analysis*, Vol. 32, 1995, pp. 1–48.

[58] Cao, Y. and Petzold, L., "A Posteriori Error Estimation and Global Error Control for Ordinary Differential Equations by the Adjoint Method," *SIAM J. Scientific Computing*, Vol. 26, No. 2, 2004, pp. 359–374.

[59] Li, S. and Petzold, L., "Adjoint Sensitivity Analysis for Time-Dependent Partial Differential Equations with Adaptive Mesh Refinement," *Journal of Computational Physics*, Vol. 198, No. 1, 2004, pp. 310–325.

[60] Mani, K. and Mavriplis, D. J., "Discrete Adjoint based Time-Step Adaptation and Error Reduction Unsteady Flow Problems," *18th AIAA Computational Fluid Dynamics Conference , Miami FL*, 2007, AIAA Paper 2007–3944.

[61] Mani, K. and Mavriplis, D. J., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando, Florida*, 2009, AIAA Paper 2009–1495.

[62] Carpenter, M. H., Viken, S., and Nielsen, E., "The Efficiency of High-Order Temporal Schemes," AIAA Paper 2003-0086.

[63] Johnson, C., "Error Estimates and Adaptive Time Step Control for a Class of One-Step Methods for Stiff Ordinary Differential Equations," *SIAM Journal of Numerical Analysis*, Vol. 25, 1988, pp. 908–926.

[64] Byrd, R. H., Lu, P., and Nocedal, J., "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific and Statistical Computing*, Vol. 16,5, 1995, pp. 1190–1208.

[65] Zhu, C., Byrd, R. H., and Nocedal, J., "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization," *ACM Transactions on Mathematical Software*, Vol. 23, No. 4, 1997, pp. 550–560.

[66] Yang, Z. and Mavriplis, D. J., "Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes," *AIAA Journal*, Vol. 45-1, January 2007, pp. 138–150.

[67] Mavriplis, D. J., "Unstructured-Mesh Discretizations and Solvers for Computational Aerodynamics," *AIAA Journal*, Vol. 46-6, June 2008, pp. 1281–1298.

[68] van Leer, B., "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101–136.

[69] Hicks, R. and Henne, P., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15-7, 1978, pp. 407–412.

[70] Willcox, K. and Peraire, J., "Aeroelastic computations in the time domain using unstructured meshes," *International Journal for Numerical Methods in Engineering*, Vol. 40-13, 1997.

[71] Dwight, R. and Brezillon, J., "Effect of Various Approximations of the Discrete Adjoint on Gradient-Based Optimization," *Proceedings of the 44th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2006, AIAA Paper 2006–0690.

[72] Nielsen, E., Lu, J., Park, M., and Darmofal, D., "An Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids," 2003, AIAA Paper 2003–0272.

[73] Valarezo, W. O. and Mavriplis, D. J., "Navier-Stokes applications to high-lift airfoil analysis," *AIAA J. of Aircraft*, Vol. 32, No. 3, 1995, pp. 618–624.

[74] AGARD, *Compendium of Unsteady Aerodynamic Measurements*, Report No.702, 1982.

[75] Alonso, J. J. and Jameson, A., "Fully-Implicit Time-Marching Aeroelastic Solutions," *32nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January*, 1994, AIAA Paper 94–0056.

[76] Isogai, K., "Transonic Dip Mechanism of Flutter of a Sweptback Wing: Part II," *AIAA Journal*, Vol. 17, 1981, pp. 1240–1242.

[77] Vendetti, D. A., *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[78] Fidkowski, K. J. and Darmofal, D. L., "Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics: Overview and Recent Results," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, Orlando, Florida*, 2009, AIAA Paper 2009–1303.

[79] Carpenter, M. H. and Vatsa, V., "Higher Order Temporal Schemes with Error Controllers for Unsteady Navier-Stokes Equations," *17th AIAA Computational Fluid Dynamics Conference, Toronta, Ontario, Canada, June 2005*, 2005, AIAA Paper 2005–5245.