An Adjoint Method Augmented with Grid Sensitivities for Aerodynamic Optimization

by

Chad Oldfield

A thesis submitted in conformity with the requirements
for the degree of M.A.Sc.
Graduate Department of Aerospace Engineering
University of Toronto

Abstract

An Adjoint Method
Augmented with Grid Sensitivities
for Aerodynamic Optimization

Chad Oldfield
M.A.Sc.
Graduate Department of Aerospace Engineering
University of Toronto
2006

The discrete adjoint equations for an aerodynamic optimizer are augmented to explicitly include the sensitivities of the grid perturbation. The Newton-Krylov optimizer is paired with grid perturbations via the elasticity method with incremental stiffening. The elasticity method is computationally expensive, but exceptionally robust—high quality grids are produced, even for large shape changes. For the gradient calculation, instead of encompassing grid sensitivities in finite differenced terms for the adjoint equations, they are treated explicitly. This results in additional adjoint equations that must be solved. This augmented adjoint method requires less computational time than a function evaluation, and retains its speed as dimensionality is increased. The accuracy of the augmented adjoint method is excellent, allowing the optimizer to converge more fully. A discussion of the trade-off between lengthy development time and increased performance indicates that the method would be particularly well-suited to complicated three-dimensional configurations.

Acknowledgements

I would like to thank Professor David Zingg for his excellent supervision. His patience to allow me to work through problems individually helped me to develop a more thorough understanding, while his grasp of both fundamentals and current research kept my efforts from becoming redundant.

Thanks goes also to my peers at UTIAS, whose extensive knowledge is so readily shared. In particular, I enjoyed many enlightening discussions about adjoint equations, derivatives, and more with Markus Rumpfkeil. Iordan Iordanov gave me a better understanding of the mind of the computer. This work would have been utterly impossible without Marian Nemec and Anh Truong, who created the foundation upon which it is based.

I am grateful to my family and friends for keeping me properly inflated. My parents gave unwavering support and made formidable efforts to understand my work, despite its abstract nature. Thank you, Allison, for keeping my roots in the ground.

Financial support has been generously provided by the National Sciences and Engineering Research Council of Canada, the Society of Naval Architects and Marine Engineers, and the University of Toronto.

Contents

List of Figures

List of Tables

List of Symbols

Alphanumeric Symbols

A vector of Cartesian coordinates of the airfoil surface

$B()$ B-spline basis functions

C_D drag coeffieient

C_L lift coefficient

e total energy

E, F inviscid flux functions in x- and y-directions

E_p strain energy

E_v, F_v viscous flux functions in x- and y-directions

$\mathcal{F}$ objective function

F vector of external forces acting on nodes

G vector of cartesian coordinates of node locations

$H()$ Heaviside step function

c_i measures of geometric constraint violation

I identity matrix

$\mathcal{J}$ penalized design objective

J Jacobian matrix of coordinate transformation

K stiffness matrix

$\mathcal{L}$ Lagrangian

L arc length along a grid line

$\mathcal{O}$ design objective

Q vector conserved flow variables

R flow solver residual vector

r grid perturbation residual vector

$\mathcal{R}e$ Reynolds number

S viscous flux vector in ξ-direction

t time

U vector of Cartesian coordinates of nodal displacements

u, v x- and y-components of velocity

w vector of parameters for B-spline curve

w_D weight for meeting drag target

w_i weights for geometric constraint penalization

w_L weight for meeting lift target

X vector of design variables

x, y Cartesian coordinates

$X^{\mathrm{c}}, Y^{\mathrm{c}}$ B-spline control point coordinates

Greek Symbols

α angle of attack

δ centred difference operator

λ vector of Lagrange multipliers (adjoint variables) for grid perturbation equations

∇ backward difference operator

Π potential energy

ψ vector of Lagrange multipliers (adjoint variables) for flow equations

ρ density

ξ, η curvilinear coordinates

Superscripts

$*$ target value

$\bar{}$ using algebraic grid perturbation

$\tilde{}$ using original adjoint method

$\hat{}$ transformation to curvilinear space

(i) index for grid pertrubation increments

(n) total number of grid perturbation increments

T matrix transpose

Subscripts

$\left.\frac{\partial a}{\partial b}\right|_c$ partial differentiation of a with respect to b carried out with c held constant

a airfoil boundary nodes

e generic element

i interior nodes

j, k ξ- and η-direction nodal indices

o outer boundary nodes

t entire mesh

Abbreviations

BFGS Broyden-Fletcher-Goldfarb-Shanno

GMRES Generalized Minimal Residual

ILU Incomplete Lower Upper

Chapter 1

Introduction

The use of computational fluid dynamics in a numerical optimization framework is a valuable aerodynamic design tool. The viability of such tools is dependent on their ability to solve optimization problems in a reasonable amount of time, and on providing the robustness required for a fully automated solution process.

This work addresses these requirements, focusing on efficient gradient computation for an aerodynamic optimizer using a robust, but computationally expensive, grid perturbation method. It is shown that augmenting the discrete adjoint equations to explicitly include the grid perturbation can result in significantly improved run time and gradient convergence.

In this chapter, gradient evaluation and grid perturbation are first presented separately, then the resulting issue of grid sensitivities is discussed. This gives rise to the motivation and objectives of the thesis.

1.1 Gradient Evaluation Methods

While many optimization techniques make use of only the objective function value, the methods that are most computationally efficient are generally those that also use the value of the objective function gradient. Efficient gradient evaluation is essential to fast gradient-based optimizers.

The most straightforward way of computing a gradient is by means of finite differences. While this has the advantage of simple implementation, it has limited accuracy due to truncation error for large step sizes, and due to subtractive cancellation error for small step sizes. Additionally, the time required to evaluate a finite difference for all partial

derivatives in the gradient is long when compared to a flow solve, and the total time scales linearly with the number of design variables. The subtractive cancellation errors can be eliminated by using the complex step method. The theory for this method is laid out by Squire and Trapp [39], and its implementation is discussed by Martins *et al.* [22]. It allows very small step sizes to be used, thereby all but eliminating truncation error. However, the gradient calculation time is similar to that of finite differences.

Alternatively, the partial derivatives can be computed analytically. Differentiated code can be produced quickly with algorithmic differentiation programs in either of the forward or reverse modes. Griewank [13] provides a thorough discussion of the methods. In the forward mode, the differentiated code must be run once for each design variable, but in the reverse mode, the code must only be run once for each dependent variable (of which there is one[1]), so has a run-time that is independent of the number of design variables. The speed advantages of the reverse mode are offset by a very large memory requirement: the intermediate values of each variable must be stored.

A semi-analytic approach to derivative calculation is the adjoint method. It was originally introduced to aerodynamic shape optimization by Pironneau [32] and Jameson [15]. As with reverse mode algorithmic differentiation, this provides a gradient computation time that is independent of the number of design variables; this time is typically on the order of the time taken by one function evaluation. This is achieved by manipulating the calculation such that the bulk of the computation involves solving a linear system whose size is independent of the number of design variables. The memory requirement is more reasonable than that of reverse mode algorithmic differentiation, but it takes considerable development time to hand-code the derivatives required to form the adjoint equations.

The accuracy of the derivatives produced from an adjoint method is dependent on which of the two varieties of the method are used: continuous or discrete. In the continuous approach, the adjoint equations are derived from the continuous flow equations, then the equations are discretized and solved. Any discretization of the continuous adjoint equations may be used, and a carefully chosen discretization can result in computational savings. However, the computed gradients are different for each possible discretization of the adjoint system. This appears as an inconsistency between the computed gradient and the gradient one would expect when examining the behaviour discretized objective func-

[1]Even in multiobjective optimization, the objectives are usually summed together to give a single dependent variable.

tion; the discrepancy is on the order of the truncation error of the method and disappears as the meshes for the flow and adjoint solvers are refined.

For the discrete adjoint approach, the process is reversed: the discretized adjoint equations are derived from the discrete flow equations. The discrete adjoint equations are one of the possible discretizations of the continuous ones and, by construction, this is the same discretization as was used for the objective function. The aforementioned error is therefore nil, and the gradient accuracy is regulated by the tolerance to which the equations are solved. As a result, the optimizer may be able to converge more tightly [1].

In accordance with a philosophy favouring tightly converged solutions while maintaining the speed advantages of the adjoint method, the discrete adjoint method has been selected by Nemec [25], whose work is the foundation for this thesis. For the optimization of the objective function $\tilde{\mathcal{F}}$ of the flow variables, $\tilde{Q}$, and the design variables, X, the objective function gradient can be evaluated as follows. First, the flow solver is converged, as represented by setting its residual to zero.

$$\tilde{R}(\tilde{Q}, X) = 0 \tag{1.1}$$

Next, the adjoint vector, ψ, is found by solving the linear system

$$\left[\frac{\partial \tilde{R}}{\partial \tilde{Q}}\right]^{\mathrm{T}} \psi = -\left[\frac{\partial \tilde{\mathcal{F}}}{\partial \tilde{Q}}\right]^{\mathrm{T}} \tag{1.2}$$

where the derivatives are evaluated at the converged values of $\bar{Q}$. Then, the gradient of the objective function is found by evaluating

$$\frac{\partial \tilde{\mathcal{F}}}{\partial X} = \left.\frac{\partial \tilde{\mathcal{F}}}{\partial X}\right|_{\tilde{Q}} + \psi^{\mathrm{T}} \left.\frac{\partial \tilde{R}}{\partial X}\right|_{\tilde{Q}} \tag{1.3}$$

where the $|_{\tilde{Q}}$ notation indicates that $\tilde{Q}$ is held constant in the differentiation. As the flow solution is retained from computing the objective function, the solution of (1.2) represents the bulk of the computational work required in the gradient evaluation; it is the size of this system that is independent of the length of X, resulting in the speed benefits of the method.

1.2 Grid Perturbation Methods

When the shape of the airfoil is altered in the optimization process, the boundary of the computational domain that touches the airfoil must be altered accordingly. To compute

the flow solution for the altered airfoil, a new computational mesh that matches this altered boundary position must therefore be created.

One way to create meshes for the altered airfoils is simply to generate a new grid for each airfoil. While some authors have used this technique in optimization [7, 19, 35], it is undesirable for two reasons. First, grids that are suitable for turbulent calculations must usually be smoothed by elliptic or hyperbolic smoothers, which can take about as much time to run as the flow solver. Secondly, grid generation is a process that is difficult to automate, especially for complex geometries. A better alternative is to produce one high-quality mesh to use as a parent for all other meshes, then to perturb the parent mesh to give one that matches the required boundary shape.[2]

A simple grid perturbation technique for structured grids is to perturb each grid line running from the airfoil surface to the outer boundary individually. Burgreen *et al.* [6] computed the perturbation of each node on the grid line by linearly interpolating the perturbation between the airfoil and the far field. The method was later improved for highly curved grid lines by basing the interpolation on the arc length along the line [5]. In multiblock meshes, grid lines may not touch the airfoil or the far field boundary, and the method needs to be modified. Nemec [25] uses trigonometric functions to preserve orthogonality for 2D multi-element airfoils. Jones and Samareh-Abolhassani [16] implemented an algebraic perturber that perturbs the block boundaries and interiors separately, using transfinite interpolation in an arc length parameter space. As they are strictly algebraic, these methods have a very short run time. However, for large perturbations, they can generate poor quality or tangled meshes. This can result in the need for the user to generate a new mesh during the optimization process. This is a notable deficiency, as an optimizer using such a method lacks complete automation.

Some grid perturbation schemes draw on the extensive research carried out by the unsteady flow computation community in the area of dynamic meshing. One of these schemes involves considering each grid line segment to be a linear spring attached to the two nodes. A linear system that represents this network of fictitious springs can be formulated and solved, yielding the nodal perturbations due to a given boundary movement. While it has been found to be fairly efficient and is applicable to unstructured or structured meshes, this approach can give tangled meshes for large shape changes.

[2] A tempting variation on this is to obtain a grid by perturbing the most recently used grid instead of perturbing the parent grid. This is not advisable, as it results in an objective function that is path-dependent, and thus non-unique and non-differentiable.

Farhat *et al.* [11] improved the robustness of the method by adding torsional springs. This was later extended to three dimensions [9]. Samareh [38] used quaternion algebra to improve the treatment of three-dimensional rotations and to help preserve the near-surface quality of viscous meshes.

A more physically meaningful alternative is to consider the computational domain as an elastic medium. Like a block of rubber, when the boundaries of the medium are deformed, the interior deforms according to the equations of linear elasticity. Tezduyar *et al.* [41] use a finite element method to compute the interior deformations. They relate the stiffness of each element to the element size, making the grid stiffer near the airfoil, and causing the deformations to propagate further into the mesh. This linear elastic theory inherently breaks down for large deformations, so can give rise to tangled meshes. To correct this, Bar-Yoseph *et al.* [2] proposed a quasi-linear approach, wherein the deformed shape is computed through a series of linear increments. At each increment, the stiffness is locally increased in highly strained areas. The result is a scheme that has a relatively high computational cost, yet is robust even to large shape changes. This has been adopted by Nielsen and Anderson [28], Stein *et al.* [40], and Truong [42].

1.3 Grid Sensitivities in the Discrete Adjoint Method

Considering that the evaluation of an aerodynamic objective function involves perturbing the grid, it is to be expected that the sensitivities of the objective function to the design variables will in some way involve sensitivities of the grid perturbation algorithm. When evaluating the gradient using the discrete adjoint method (1.2–1.3), these mesh sensitivities are implicitly included in the terms $\partial\tilde{\mathcal{F}}/\partial X|_{\tilde{Q}}$ and $\partial\tilde{R}/\partial X|_{\tilde{Q}}$. In contrast to this, Anderson and Venkatakrishnan [1] show that grid sensitivities do not explicitly appear in the continuous adjoint formulation.[3]

Within the discrete adjoint method, a number of different approaches to computing grid sensitivities have been used. Kim *et al.* [17, 18] found that it is possible to neglect the grid sensitivities for design variables that do not involve the translation of a body, and when the flow is considered to be inviscid.

Nemec [25] and Martins *et al.* [21] used finite differences for the grid sensitivities. Since their optimizers both use algebraic grid perturbation, it is not computationally

[3]However, they note that it is important to consider grid sensitivities, and suggest a hybrid continuous-discrete approach.

onerous to repeatedly perturb the grid when forming the derivatives in this way. Burgreen and Baysal [5] and Le Moigne and Qin [20] hand differentiated arc length-based algebraic grid perturbers analytically. Bischof *et al.* [4] used automatic differentiation to reduce the human effort required to differentiate the more complicated algebraic grid perturbation method of Jones and Samareh-Abolhassani [16].

For three-dimensional unstructured meshes, Samareh [37, 36] introduced a new surface parameterization. It is a novel approach based on soft object animation, where a volume of space is deformed according to the design variables, causing the boundaries and the grid to be perturbed together. It was analytically differentiated to provide grid sensitivities.

More computationally expensive grid perturbation algorithms, such as the spring analogy and elasticity methods, have been differentiated using an adjoint approach. Maute *et al.* [23] present an aeroelastic optimizer for three-dimensional Euler flows. They use the spring analogy mesh perturber of Farhat *et al.* [11]. The adjoint problem is posed as a large linear system that includes the coupled aerodynamic, structural, and mesh movement terms. It is solved using a staggered procedure, giving adjoint variables for each of the flow, grid perturbation, and structural solvers. The method is shown to be efficient, but it is noted that both the time and memory requirements for the derivative of the the flow residual with respect to the interior node locations are considerable.

Recently, Nielsen and Park [30] presented an adjoint method for aerodynamic optimization. It computes an adjoint vector for the flow solver and then for the grid perturber. This was performed for an unstructured grid that is perturbed using an algorithm based on linear elasticity. The accuracy of the sensitivities was comparable to that obtained using direct differentiation, and the computational time was reduced dramatically. Mavriplis [24] took a similar approach using the spring analogy for grid perturbation.

While grid perturbation is more popular than regeneration in optimization, analytic and semi-analytic methods have also been used to differentiate grid generation codes. Sadrehaghighi *et al.* [35] use an algebraic grid generator to regenerate a grid at each optimizer iteration. Their sensitivity analysis includes analytic differentiation of the grid generation equations. Korivi *et al.* [19] provide grid sensitivities through algorithmic differentiation of the algebraic grid generator of Barger *et al.* [3]. For aerodynamic sensitivity analysis, Pagaldipti and Chattopadhyay [31] differentiated the equations governing elliptic and hyperbolic grid generation, yielding a system of equations that can be solved for the sensitivities of the grid generator.

1.4 Objectives

The objective of the current work is to provide an efficient and accurate sensitivity analysis for a code combining the efficient aerodynamic shape optimizer of Nemec [25, 26, 27] with the robust elasticity grid perturber of Truong [42]. This grid perturbation method has been selected for its robustness, which eliminates the need for remeshing, even for large airfoil shape changes.

Simply inserting the elasticity method into the original optimizer of Nemec [25] results in considerable computational expense, because the finite element solver must be run several times to compute the derivatives in (1.3). Furthermore, as the finite differences must be evaluated for each design variable, one would expect the CPU time to increase linearly with the number of design variables. This undermines the usual claim that an adjoint method provides a gradient computation time that is nearly independent of the number of design variables—and that is the adjoint method's principal advantage.

Instead, an adjoint approach will be used to treat the grid sensitivities. The method can be expected to give a gradient calculation time that is virtually independent of the number of design variables. Additionally, the method's accuracy may be increased over that of finite differences, because of the inherent errors in finite differences. While this accuracy is not important in itself, it is anticipated that it may allow the optimizer to converge in fewer iterations.

It is also expected that creating the adjoint code will involve considerable effort. It is therefore important to not only evaluate the above expected advantages of the method, but to weigh them against the development time. This feasibility assessment will indicate whether the approach should be adopted by the three-dimensional optimizer currently under development.

Chapter 2

Design Evaluation

In order to attempt numerical aerodynamic shape optimization, one must first develop a design evaluation capability that is sufficiently automated for a numerical optimizer to interface with it. The present algorithm for the analysis of the performance of an airfoil can be broken into four components. They are

1. the use of design variables to represent a wide variety of designs;
2. the perturbation of a computational mesh so that it can conform to these designs;
3. the flow analysis on the perturbed mesh; and
4. the use of the flow solution to produce a meaningful design objective.

These components are described in the following sections.

2.1 Airfoil Parameterization

For the optimizer, the design is fully defined by a set of design variables. These variables, represented by the vector X, may determine the shape of the airfoil and the angle of attack. While a 1:1 correspondence between the angle of attack and one of the elements of X is reasonable, the representation of the airfoil shape by the remaining elements is more complicated.

It is desirable to allow the airfoil to take on a wide variety of shapes while retaining the same chord length[1]. However, it is also expected that any optimum airfoil will be

[1]The analysis uses only non-dimensional values, so this non-dimensional chord length is unity.

smooth (except at the trailing edge). In the interest of simplifying the optimization problem, it is desirable to use the smallest practical number of design variables and to ensure that a given design variable has only a localized impact on airfoil shape.

To achieve this, the airfoil surface is represented by a B-spline, whereby the vertical position of some of the control points are given by design variables. The j^{th} point on the B-spline is given by

$$A_j = \begin{bmatrix} \sum_i X_i^{\text{c}} B_i(w_j) \\ \sum_i Y_i^{\text{c}} B_i(w_j) \end{bmatrix} \tag{2.1}$$

where A is the vector of positions on the airfoil surface, w_j is the parameter value for the j^{th} point on the airfoil surface, and $B_i(w_j)$ are B-spline basis functions. The basis functions are defined with a recursive relation, but computed using de Boor's efficient sequential implementation [8]. X^{c} and Y^{c} are the locations of the B-spline control points, and as stated above, some of Y^{c} correspond to design variables, X. This is described further in [25].

2.2 Grid Perturbation Scheme

Using the process developed by Truong [42], the grid is perturbed by treating the computational domain as an elastically deformable material. The edges of the material are fixed to the outer boundary and to the airfoil boundary, so when the airfoil is perturbed, the material deforms.

Using the same computational mesh as is used for flow analysis, the equations of linear elasticity are discretized and solved using a finite element method. Since the flow solver uses a structured C-mesh, the elements are quadrilaterals.

Linear elasticity is a theory that is only valid for small deformations, while the deformations encountered in an aerodynamic shape optimization problem may be large. This gives the possibility of producing a perturbed material that is entangled. While solving non-linear elasticity equations could solve this, it was deemed to be too computationally expensive. Instead, a quasi-linear method is used.

The quasi-linear algorithm computes the deformation in a series of linear steps. So, the airfoil surface is moved from the parent shape to the perturbed shape in a series of equal increments,

$$A^{(i)} = A^{(0)} + \frac{i}{n}\left(A^{(n)} - A^{(0)}\right) \tag{2.2}$$

where $A^{(i)}$ is the vector of coordinates on the airfoil surface for the i^{th} increment in a perturbation having a total of n increments[2]. $A^{(0)}$ is the parent airfoil, and $A^{(n)} = A$ is the fully perturbed airfoil. At each increment, the stiffness of the material is adjusted, then the mesh deformation is computed using a linear finite element code.[3]

The elastic properties of the material are chosen such that the high quality of the initial mesh is retained as much as possible. Young's modulus, E, is set to be proportional to the inverse of the cell area, so that small cells near the airfoil deform less. In addition, the change in stiffness from one increment to the next is based on cell distortion such that E increases to infinity as mesh entanglement is approached.

The exact formulation of Young's modulus and the discretization of linear elastic equations to form the element stiffness matrix, K_e, are included in Appendix C. It is shown that the total strain energy stored in the elastic medium is given by

$$E_p = \frac{1}{2} \sum_e U_e^{\mathrm{T}} K_e U_e \tag{2.3}$$

where U_e is an 8-element vector containing the x- and y-direction displacements of each of the corner nodes of the quadrilateral element e. This can also be written as

$$E_p = \frac{1}{2} U_t^{\mathrm{T}} K_t U_t \tag{2.4}$$

Where U_t and K_t are the displacement vector and stiffness matrix for the entire system. The dimensionality of these quantities is the number of degrees of freedom in the system ($2 \times$ number of nodes in two dimensions). Each entry in K_t is formed by summing the corresponding entries in each overlapping K_e.

The total strain energy is the sum of the potential energy, Π, and the external work, $U_t^{\mathrm{T}} F_t$, where F_t is the vector of external forces acting on each of the nodes.

$$\frac{1}{2} U_t^{\mathrm{T}} K_t U_t = \Pi + U_t^{\mathrm{T}} F_t \tag{2.5}$$

The steady solution of this system is such that the potential energy is minimized. This is obtained by setting the derivative of Π with respect to U_t to 0, which yields the following linear system:

$$K_t U_t = F_t \tag{2.6}$$

[2]This is defined for $i \in \{1, 2, \ldots, n\}$. For the C++ implementation, $i \in \{0, 1, \ldots, n-1\}$, so the equation must be altered accordingly.

[3]Though it is tempting to allow the number of increments to be chosen automatically based on how large the perturbation is, n is kept fixed for the entire optimization process. Failing to do so may introduce discontinuities into the design space.

At the interior nodes, F_t is set to zero, and the nodal displacements are unknown. However, at the boundary nodes, the nodal displacements are specified, and the reactions are unknown. Splitting the system into interior (i), outer boundary (o), and airfoil boundary (a) nodes,

$$\begin{bmatrix} K_i & K_{io} & K_{ia} \\ K_{oi} & K_o & K_{oa} \\ K_{ai} & K_{ao} & K_a \end{bmatrix} \begin{bmatrix} U_i \\ 0 \\ A^{(i)} - G_a^{(i-1)} \end{bmatrix} = \begin{bmatrix} 0 \\ F_o \\ F_a \end{bmatrix} \tag{2.7}$$

where $G^{(i-1)}$ is the solution for the grid at increment $i-1$. $G^{(i-1)}$ has two entries per node, which correspond to the x- and y-coordinates of the node. The reactions F_o and F_a are not required. Removing the respective equations gives

$$K_i U_i = -K_{ia} \left[A^{(i)} - G_a^{(i-1)} \right] \tag{2.8}$$

This allows the displacements of the interior nodes to be solved for. Writing this explicitly as the change in the grid from one increment to the next, and including the grid movement at the boundaries, results in

$$\begin{bmatrix} K_i \left[G_i^{(i)} - G_i^{(i-1)} \right] \\ G_o^{(i)} \\ G_a^{(i)} \end{bmatrix} = \begin{bmatrix} -K_{ia} \left[A^{(i)} - G_a^{(i-1)} \right] \\ G_o^{(0)} \\ A^{(i)} \end{bmatrix} \tag{2.9}$$

The boundary entries in $G^{(i)}$ can be found directly, but computing the interior entries in $G^{(i)}$ requires the inversion of a sparse, symmetric, positive definite linear system. The IML++ implementation [10] of the conjugate gradient method, with an ILU preconditioner, is used for this. Although the rest of the optimizer is written in FORTRAN, the grid perturber is a C++ code.

For the purposes of the sensitivity analysis, the grid perturbation equations are written as setting a residual, $r^{(i)}$, to zero.

$$0 = r^{(i)} = \begin{bmatrix} r_i^{(i)} \\ r_o^{(i)} \\ r_a^{(i)} \end{bmatrix} = \begin{bmatrix} K_i \left[G_i^{(i)} - G_i^{(i-1)} \right] + K_{ia} \left[A^{(i)} - G_a^{(i-1)} \right] \\ G_o^{(i)} - G_o^{(0)} \\ G_a^{(i)} - A^{(i)} \end{bmatrix} \tag{2.10}$$

Or, more simply,

$$0 = r^{(i)} = r^{(i)}(G^{(i)},\ G^{(i-1)},\ A^{(i)}(X)) \tag{2.11}$$

2.3 Flow Solver

The flow around the airfoil is computed by using an efficient Newton-Krylov method to solve the discretized Reynolds-averaged Navier-Stokes equations. Turbulent effects are predicted by the Spalart-Allmaras turbulence model. On a structured C-mesh, spatial derivatives are approximated using second-order centred finite differences, plus scalar artificial dissipation. The start-up algorithm uses implicit Euler time marching with the linear solve simplified using approximate factorization, and with explicit boundary conditions. Final convergence, however, is achieved using Newton's method, with implicit boundary treatment. The linear system at each Newton step is solved efficiently using the generalized minimum residual (GMRES) Krylov subspace method. The efficiency of the method is significantly improved through the use of an ILU preconditioner with limited fill, a Jacobian-free implementation of GMRES [34], and inexact convergence of the linear system. Nemec [25] provides a description of the present solver, which is based on the work of Pueyo [33]. The overall discretized equations are presented here.

Consider first the Reynolds-Averaged Navier-Stokes equations, written in differential form:

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = \frac{1}{\mathcal{R}e}\left(\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y}\right) \tag{2.12}$$

where x and y are the Cartesian coordinates in physical space, Q is the vector of conserved variables (mass, momentum, and energy), the spatial derivatives of E and F are inviscid fluxes, the spatial derivatives of E_v and F_v are viscous fluxes, and $\mathcal{R}e$ is the freestream Reynolds number. At the node (j,k), Q is given by

$$Q_{j,k} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}_{j,k} \tag{2.13}$$

The components of Q are all non-dimensional: ρ is density, u and v are the x- and y-components of velocity, and e is total energy.

The flow equations are transformed from the curved grid in rectangular physical coordinates, (x,y), to an equi-spaced rectangular grid in curvilinear computational coordinates, (ξ,η). This simplifies the finite differencing stencils to be used in the discrete equations. The computational coordinates, as shown in Figure 2.1, are

$$\xi = \xi(x,y) \qquad \eta = \eta(x,y) \tag{2.14}$$

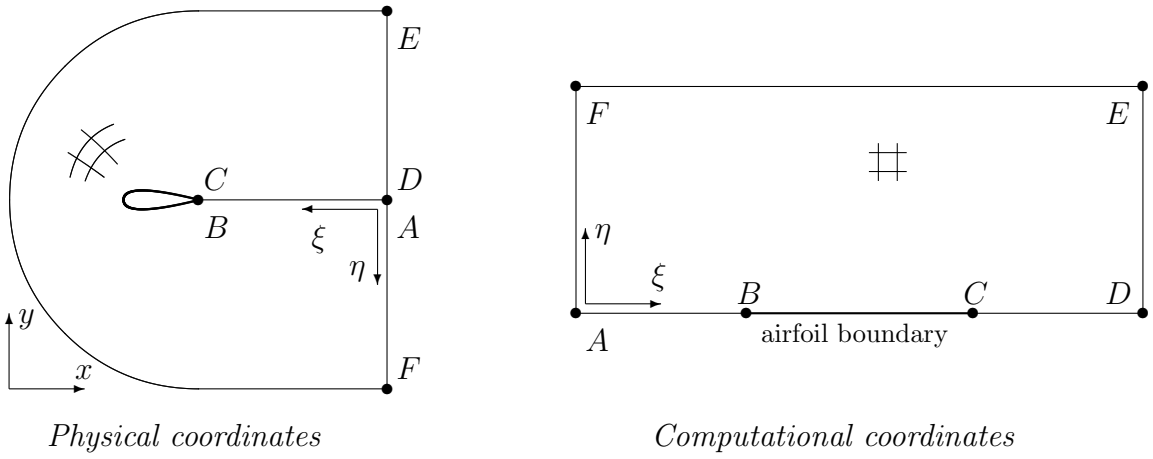

Figure 2.1: Coordinate transformation

In transforming to these coordinates, we use the following metrics

$$\begin{aligned} \xi_x &= Jy_\eta & \xi_y &= -Jx_\eta \\ \eta_x &= -Jy_\xi & \eta_y &= Jx_\xi \end{aligned} \tag{2.15}$$

$$J_{j,k} = \left(\frac{1}{x_\xi y_\eta - x_\eta y_\xi} \right)_{j,k} \tag{2.16}$$

where J is the Jacobian of the coordinate transformation—a diagonal matrix of size $j \times k$ with entries $J_{j,k}$. Notice that these metrics are all computed at each node using second-order centred differences:

$$\begin{aligned} x_\xi &= \delta_\xi x & x_\eta &= \delta_\eta x \\ y_\xi &= \delta_\xi y & y_\eta &= \delta_\eta y \end{aligned} \tag{2.17}$$

When the governing equations, (2.12), are transformed into computational coordinates, unsteadiness is neglected, and the thin-layer approximation is used, they become

$$0 = \frac{\partial \widehat{E}}{\partial \xi} + \frac{\partial \widehat{F}}{\partial \eta} - \frac{1}{\mathcal{R}e} \frac{\partial \widehat{S}}{\partial \eta} \tag{2.18}$$

where the fluxes are functions of the dependent conserved flow variables in computational space,

$$\widehat{Q} = J^{-1} Q \tag{2.19}$$

Discretizing the equations, one obtains

$$R = \delta_\xi \widehat{E} + \delta_\eta \widehat{F} - \frac{1}{\mathcal{R}e} \delta_\eta \widehat{S} - \nabla_\xi \widehat{E}_d - \nabla_\eta \widehat{F}_d = 0 \tag{2.20}$$

where R is the flow solver residual, to be set to zero. On the right-hand side, the first two terms are inviscid fluxes, the middle term is the viscous flux, and the rightmost two terms are artificial dissipation (to control high wavenumber fluctuations in the solution).

These terms and the turbulence model are described in Appendix B and in [25]. Here, we need only note that these terms are functions of the flow variables, the metrics of the transformation, and the angle of attack.[4,5] This allows the discretized flow equations to be written as

$$R(Q,\ G,\ X) = 0 \tag{2.21}$$

2.4 Design Objectives

Once the flow around an airfoil has been computed, the results can be used to evaluate the performance of that airfoil. Performance is represented by a single value, the design objective, $\mathcal{O}$.

Design objectives may be, for example, drag minimization at fixed lift, or drag to lift ratio:

$$\mathcal{O} = w_L \left(1 - \frac{C_L}{C_L^*}\right)^2 + w_D H(C_D - C_D^*) \left(1 - \frac{C_D}{C_D^*}\right)^2, \qquad \mathcal{O} = \frac{C_D}{C_L} \tag{2.22}$$

where C_D and C_L are the drag and lift coefficients, computed by integrating pressure and viscous stresses around the airfoil. C_L^* is the target lift, C_D^* is an unattainably low target drag, w_L and w_D are weights, and $H()$ is the Heaviside step function.

The design objective depends on of the flow solution, represented by the conserved quantities, Q, as they determine the pressures and viscous stresses acting on the airfoil. It is also a function of the coordinates of the grid nodes, G, as they affect the calculation of viscous stresses and determine the path of the line integral used to find the net force on the airfoil. Finally, it is a function of the design variables, X, as the angle of attack determines the direction in which the lift and drag components are resolved. The design objective can then be generally represented as

$$\mathcal{O} = \mathcal{O}(Q, G, X) \tag{2.23}$$

If $\mathcal{O}$ were used as the objective function in an optimization, it is possible that the optimum found would represent an infeasible design due to structural, operational, or

[4] In contrast to 2.13, the discretized Q (and $\widehat{Q}$) has a fifth element at each node; it is the dependent variable of the one-equation turbulence model

[5] Recall that the angle of attack is one of the design variables, X.

off-design considerations. These could be airfoils in which the upper and lower surfaces cross each other, the maximum thickness is too thin for a structural member to fit inside, the trailing edge bends abruptly up or down, the leading edge is a point, or the enclosed area is too small (i.e. for a fuel tank). An ideal way of avoiding these and other infeasible designs would be to use a multidisciplinary, multipoint approach, where other aspects of the aircraft, such as structural members in the airfoil, are also modelled, and where a variety of operating points are considered.

While multidisciplinary, multipoint optimization may offer a more complete design capability, it is somewhat more involved than strictly aerodynamic optimization. This added complexity makes multidisciplinary, multipoint optimization a poor choice for evaluating new concepts such as the present one. Therefore, in this work a simplified approach is taken: infeasible airfoils are avoided by constraining the airfoil geometry. Any combination of the following constraints may be used:

- the airfoil thickness must exceed specified minimum thicknesses at specified chordwise stations,
- the airfoil thickness must exceed a specified minimum thickness at any one point within specified range of chordwise stations,
- the area enclosed by the airfoil must exceed a specified minimum area, and
- the radius of curvature of the airfoil surface must exceed a minimum value, specified as a piecewise linear function of the chordwise station.

These inequality constraints depend solely on the design variables that determine the shape of the airfoil. They are implemented using a quadratic external penalty method. For the constraints $c_i(X) \leq 0$ having weights w_i, the constrained design objective is

$$\mathcal{J}(Q,\ G,\ X) = \mathcal{O}(Q,\ G,\ X) + \sum_i w_i c_i^2(X) H\left(c_i(X)\right) \tag{2.24}$$

Chapter 3

Adjoint Formulation

A rigorous method of deriving the discrete adjoint equations is through the use of Lagrange multipliers, as presented by Gunzburger [14]. Using this method, the variables X, $G^{(i)}$, and Q are considered to be independent. The optimization problem can then be posed as minimizing the design objective with respect to all these variables, subject to the constraint that the flow solver must have converged. The is augmented to also consider the convergence of the grid perturbation as a constraint. This is expressed as

$$\begin{aligned} \min \quad & \mathcal{J}(Q, G^{(n)}, X) \\ \text{w.r.t.} \quad & Q, G^{(n)}, X \\ \text{s.t.} \quad & 0 = r^{(i)}(G^{(i)}, G^{(i-1)}, A^{(i)}(X)), \quad i \in \{1, 2, \ldots, n\} \\ & 0 = R(Q, G^{(n)}, X) \end{aligned}$$

To enforce the constraints, let the Lagrangian, $\mathcal{L}$, be defined as follows:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}(\lambda^{(i)},\ \psi,\ Q,\ G^{(i)},\ X), \qquad i \in \{1, 2, \ldots, n\} \\ &= \mathcal{J} + \sum_{i=1}^{n} \lambda^{(i)\mathrm{T}} r^{(i)} + \psi^{\mathrm{T}} R \end{aligned} \tag{3.1}$$

where ψ and $\lambda^{(i)}$ are the Lagrange multipliers.

Setting each of the partial derivatives of the Lagrangian to zero then provides optimality conditions for a physically meaningful objective function:

$$\frac{\partial \mathcal{L}}{\partial \lambda^{(i)}} = 0 = r^{(i)}, \qquad i \in \{1, 2, \ldots, n\} \tag{3.2}$$

$$\frac{\partial \mathcal{L}}{\partial \psi} = 0 = R \tag{3.3}$$

$$\frac{\partial \mathcal{L}}{\partial Q} = 0 = \frac{\partial \mathcal{J}}{\partial Q} + \psi^{\mathrm{T}} \frac{\partial R}{\partial Q} \tag{3.4}$$

$$\frac{\partial \mathcal{L}}{\partial G^{(n)}} = 0 = \frac{\partial \mathcal{J}}{\partial G^{(n)}} + \lambda^{(n)\mathrm{T}} \frac{\partial r^{(n)}}{\partial G^{(n)}} + \psi^{\mathrm{T}} \frac{\partial R}{\partial G^{(n)}} \tag{3.5}$$

$$\frac{\partial \mathcal{L}}{\partial G^{(i)}} = 0 = \lambda^{(i)\mathrm{T}} \frac{\partial r^{(i)}}{\partial G^{(i)}} + \lambda^{(i+1)\mathrm{T}} \frac{\partial r^{(i+1)}}{\partial G^{(i)}}, \qquad i \in \{n-1, n-2, \ldots, 1\} \tag{3.6}$$

$$\frac{\partial \mathcal{L}}{\partial X} = 0 = \frac{\partial \mathcal{J}}{\partial X} + \sum_{i=1}^{n} \left(\lambda^{(i)\mathrm{T}} \frac{\partial r^{(i)}}{\partial A^{(i)}} \frac{\partial A^{(i)}}{\partial X} \right) + \psi^{\mathrm{T}} \frac{\partial R}{\partial X} \tag{3.7}$$

A number of approaches could be taken to find a solution to these optimality conditions to minimize the Lagrangian, including possibilities such as solving for all of the variables simultaneously—the approach that defines simultaneous analysis and design (SAND). Using this one-shot approach, a large-scale solver sets the entire gradient of the Lagrangian (the KKT system) to zero.

However, instead of attempting to minimize a function of so many variables, the approach taken here is one that delegates a great deal of the computation to the existing specialized solvers. For a given X, (3.2) can be solved using the grid perturbation code to yield each $G^{(i)}$. Using that solution, the flow solver can be used to solve (3.3) and yield Q. The linear systems in (3.4–3.6) can then be solved in the order they appear to give ψ and each $\lambda^{(i)}$. Finally, the right-hand side of (3.7) can be evaluated to yield a value for $\partial \mathcal{L}/\partial X$.

The process above is one that, for a given X, sets most of the entries in the gradient of the Lagrangian to zero, leaving only $\partial \mathcal{L}/\partial X$ nonzero:

$$\frac{\partial \mathcal{L}}{\partial \left\{\lambda^{(i)}, \psi, Q, G^{(i)}, X\right\}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dfrac{\partial \mathcal{L}}{\partial X} \end{bmatrix} \tag{3.8}$$

The optimizer therefore only needs to consider X, $\mathcal{L}$, and $\partial \mathcal{L}/\partial X$, provided that (3.2–3.6) are solved at X. This can be denoted as the following objective function:

$$\mathcal{F}(X) = \mathcal{L}, \quad \left\{ \lambda^{(i)}, \psi, Q, G^{(i)} \;\middle|\; \frac{\partial \mathcal{L}}{\partial \left\{\lambda^{(i)}, \psi, Q, G^{(i)}\right\}} = 0 \right\}, \quad i \in \{1, 2, \ldots, n\} \tag{3.9}$$

having a gradient given by

$$\frac{\partial \mathcal{F}}{\partial X} = \frac{\partial \mathcal{L}}{\partial X}, \quad \left\{ \lambda^{(i)}, \psi, Q, G^{(i)} \;\middle|\; \frac{\partial \mathcal{L}}{\partial \left\{\lambda^{(i)}, \psi, Q, G^{(i)}\right\}} = 0 \right\}, \quad i \in \{1, 2, \ldots, n\} \tag{3.10}$$

The function $\mathcal{F}$ and the gradient $\partial \mathcal{F}/\partial X$ are sent to the optimizer.

Notice that in evaluating the gradient, the adjoint equations, (3.4–3.6), are linear systems whose sizes are independent of the number of design variables. In evaluating the

gradient, the only step whose computational time depends on the number of design variables is the evaluation of $\partial\mathcal{L}/\partial X$ in (3.7). This step requires only matrix multiplication and addition, so is not computationally demanding. The time required to evaluate the gradient should therefore be nearly independent of the number of design variables.

Another common way of deriving adjoint equations is through the use of the chain rule, with careful attention paid to which functions are "implicit", and which are "explicit". It allows one to visualize ψ as a total derivative of $\mathcal{F}$ with respect to R, and $\lambda^{(i)}$ as a total derivative of $\mathcal{F}$ with respect to $r^{(i)}$. In this case, however, the chain rule approach to the derivation is considerably longer than the above, and is difficult to extend to different values of n. This alternate derivation, for $n = 2$, can be found in Appendix A.

Chapter 4

Implementation

Most of the computational effort required to solve these adjoint equations is in the linear and non-linear solvers, but the bulk of the human effort is in forming the terms for the equations. A summary of the required terms is provided in Table 4.1. An overview of the computation of each term follows a brief discussion of the treatment of the wake cut.

4.1 Wake Cut

Along the wake cut, there is a minor difference between the grids used by the flow solver and by the grid perturber. In the flow solver, G has two sets of overlapping nodes along the wake cut, while in the grid perturbation, $G^{(n)}$ has only one set of wake cut nodes. These quantities are related according to

$$\begin{bmatrix} G_{\text{nwc}} \\ G_{\text{wc1}} \\ G_{\text{wc2}} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & I \end{bmatrix} \begin{bmatrix} G^{(n)}_{\text{nwc}} \\ G^{(n)}_{\text{wc}} \end{bmatrix} \tag{4.1}$$

	Q	$G^{(i)}$	$A^{(i)}$	X
$\mathcal{J}$	$\partial\mathcal{J}/\partial Q$	$\partial\mathcal{J}/\partial G^{(n)}$		$\partial\mathcal{J}/\partial X$
R	$\partial R/\partial Q$	$\partial R/\partial G^{(n)}$		$\partial R/\partial X$
$r^{(i)}$		$\partial r^{(i)}/\partial G^{(i)}$	$\partial r^{(i)}/\partial A^{(i)}$	
$r^{(i+1)}$		$\partial r^{(i+1)}/\partial G^{(i)}$		
$A^{(i)}$				$\partial A^{(i)}/\partial X$

Table 4.1: Summary of required Jacobian matrices

where the subscripts 'nwc' indicates nodes not on the wake cut, 'wc1' and 'wc2' indicate the two sets of overlapping wake cut nodes in G, and 'wc' indicate the single set of wake cut nodes in $G^{(n)}$. The derivatives of $\mathcal{J}$ and R with respect to $G^{(n)}$ are therefore expanded as

$$\frac{\partial \mathcal{J}}{\partial G^{(n)}} = \frac{\partial \mathcal{J}}{\partial G}\frac{\partial G}{\partial G^{(n)}} , \qquad \frac{\partial R}{\partial G^{(n)}} = \frac{\partial R}{\partial G}\frac{\partial G}{\partial G^{(n)}} , \qquad \frac{\partial G}{\partial G^{(n)}} = \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & I \end{bmatrix} \tag{4.2}$$

This results in the overlapping wake cut terms being summed together in (3.5).

4.2 Differentiating the Design Objectives

The design objectives are expressed in terms of lift, drag and moment coefficients. These quantities are calculated by numerically integrating the pressure and viscous stresses acting on the airfoil.

First, the moment coefficient and net force coefficients in the chordwise and normal (x and y) directions are found; the trapezoidal rule is used for each of these integrals of the local pressure forces and the viscous stresses. The force coefficients are then resolved into the lift and drag directions (perpendicular and parallel to the freestream flow), which are rotated from the coordinate directions by the angle of attack, α. The lift, drag, and moment coefficients are then used in the calculation of the design objective, $\mathcal{J}$.

The design objective thus has a dependency on the flow solution, the grid node locations, and the angle of attack (a design variable). The terms $\partial\mathcal{J}/\partial Q$, $\partial\mathcal{J}/\partial G$, $\partial\mathcal{J}/\partial X$ are computed analytically. The derivations of these terms are omitted, as they are all relatively straightforward.

4.3 Differentiating the Flow Solver Residual

4.3.1 Derivative with Respect to the Conserved Flow Variables

In the flow solver, the fact that Newton's method is used to set $R = 0$ makes the calculation of $\partial R/\partial Q$ for the adjoint system far simpler. In iteration i of Newton's method, the following linear system is solved

$$\left.\frac{\partial R}{\partial \widehat{Q}}\right|_{G,X}^{(i)} \Delta\widehat{Q}^{(i)} = R^{(i)} \tag{4.3}$$

The notation $|_{G,X}$ on the flow Jacobian indicates that G and X are held constant while the partial differentiation is performed. This notation is discussed further in Appendix A. When the flow solver has converged, the flow Jacobian in (4.3) is very similar to $\partial R/\partial Q$, required for the adjoint system. This relation is found using (2.19):

$$\frac{\partial R}{\partial Q} = \left.\frac{\partial R}{\partial \widehat{Q}}\right|_{G,X} \frac{\partial \widehat{Q}}{\partial Q} = \left.\frac{\partial R}{\partial \widehat{Q}}\right|_{G,X} J^{-1} \tag{4.4}$$

Using this and noting that J is a diagonal matrix, (3.4) can be multiplied by J as follows:

$$\left(\left.\frac{\partial R}{\partial \widehat{Q}}\right|_{G,X}\right)^{\mathrm{T}} \psi = -J \left(\frac{\partial \mathcal{J}}{\partial Q}\right)^{\mathrm{T}} \tag{4.5}$$

With the adjoint equation written in this from, the subroutines that compute $\partial R/\partial \widehat{Q}|_{G,X}$ for the flow solver can be re-used to form the left-hand side[1]. Additionally, because the left-hand side of this equation is nominally the same as that of the flow solver, the same ILU-preconditioned GMRES algorithm is used to solve this adjoint system.

4.3.2 Derivative with Respect to the Grid

While the finite differencing stencils are much simpler as a result of the transformation from physical space to computational space with unit grid spacing, the transformation does not simplify $\partial R/\partial G$. The flow residual has a heavy dependency on the metrics of the coordinate transformation, and as seen in (2.15–2.17), these are all formed through finite differences of the grid node locations. $\partial R/\partial G$ must therefore be formed by differentiating first R with respect to the metrics, then the metrics with respect to G.

The storage requirements for $\partial R/\partial G$ are immense. However, when the Jacobian matrix is used in (3.4), it is pre-multiplied by ψ^{T}. So, instead of storing the entire Jacobian in memory, only the product $\psi^{\mathrm{T}}(\partial R/\partial G)$ is stored. It is formed by multiplying by ψ as each term is computed.

The full details of the differentiation of each term of R with respect to G can be found in Appendix B.

[1]The flow solver uses Jacobian-free GMRES, so the left-hand side in the flow solver is only an approximation of $\partial R/\partial \widehat{Q}|_{G,X}$. The preconditioner used in the flow solver, however, uses an analytically-formed flow Jacobian, and this is what is re-used in forming the left-hand side of the flow adjoint equation.

4.3.3 Derivative with Respect to the Angle of Attack

In the flow solver, the angle of attack is used to determine the flow direction of the freestream. For the flow residual, this results in a dependence on the angle of attack at the inflow/outflow boundary conditions. This is easily differentiated to give $\partial R/\partial \alpha$—the only nonzero term in $\partial R/\partial X$.

4.4 Differentiating the Grid Perturbation Residual

The derivatives of the grid perturbation residual, $r^{(i)}$, can be found from (2.10) as follows:

$$\frac{\partial r^{(i)}}{\partial G^{(i)}} = \begin{bmatrix} K_i^{(i)} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \tag{4.6}$$

$$\frac{\partial r^{(i+1)}}{\partial G^{(i)}} = \begin{bmatrix} -K_i^{(i+1)} & 0 & -K_{ia}^{(i+1)} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} \frac{\partial K_i^{(i+1)}}{\partial G^{(i)}} \left[G_i^{(i+1)} - G_i^{(i)}\right] + \frac{\partial K_{ia}^{(i+1)}}{\partial G^{(i)}} \left[A^{(i+1)} - G_a^{(i)}\right] \\ 0 \\ 0 \end{bmatrix} \tag{4.7}$$

$$\frac{\partial r^{(i)}}{\partial A^{(i)}} = \begin{bmatrix} K_{ia}^{(i)} \\ 0 \\ -I \end{bmatrix} \tag{4.8}$$

All of these terms are straightforward to form, with the exception of the derivatives of $K_i^{(i+1)}$ and $K_{ia}^{(i+1)}$ with respect to $G^{(i)}$. Considering that each entry in the stiffness matrix, K_t, is the sum of the appropriate entries in the element stiffness matrices, K_e, the derivatives of the stiffness matrix can be formed by similarly adding together the derivatives of the element stiffness matrices. Therefore, the intensive part of deriving $\partial K_i^{(i+1)}/\partial G^{(i)}$ and $\partial K_{ia}^{(i+1)}/\partial G^{(i)}$ is the derivation of $\partial K_e^{(i+1)}/\partial G_e^{(i)}$. This derivation is included in Appendix C.

The memory requirement of these Jacobian matrices is massive. However, they only appear in the adjoint equations when pre-multiplied by the adjoint vector. Specifically, the following two terms appear in (3.6, 3.7):

$$\lambda^{(i+1)\mathrm{T}} \frac{\partial r^{(i+1)}}{\partial G^{(i)}} \qquad \lambda^{(i)\mathrm{T}} \frac{\partial r^{(i)}}{\partial A^{(i)}}$$

The memory requirement is reduced by storing only the vector-matrix products: each entry in the matrix is immediately pre-multiplied by the adjoint vector when it is computed.

A similar approach is used to avoid storing the third-order tensor, $\partial K_e^{(i+1)}/\partial G_e^{(i)}$, which only appears when post-multiplied by the vectors given in (4.7), and when pre-multiplied by the adjoint vector.

The boundary terms in (4.6–4.8) are rather simple, so some simplifications can be made when solving the adjoint equations. In particular, notice that when (3.5, 3.6) are rearranged into the form in which they are solved for $\lambda^{(n)}$ and $\lambda^{(i)}$, the boundary terms of the left-hand side matrices are identity entries:

$$\left[\frac{\partial r^{(n)}}{\partial G^{(n)}}\right]^{\mathrm{T}} \lambda^{(n)} = \left[-\frac{\partial \mathcal{J}}{\partial G^{(n)}} - \psi^{\mathrm{T}} \frac{\partial R}{\partial G^{(n)}}\right]^{\mathrm{T}} \tag{4.9}$$

$$\left[\frac{\partial r^{(i)}}{\partial G^{(i)}}\right]^{\mathrm{T}} \lambda^{(i)} = \left[-\lambda^{(i+1)\mathrm{T}} \frac{\partial r^{(i+1)}}{\partial G^{(i)}}\right]^{\mathrm{T}}, \qquad i \in \{n-1, n-2, \ldots, 1\} \tag{4.10}$$

Hence, the outer boundary and airfoil boundary entries in the adjoint vectors can be computed directly, and the linear solution need only consider the interior scheme. This is expected, as the same simplification can be made when originally perturbing the grid. The left-hand side for the interior scheme is simply the transpose of the stiffness matrix; as the stiffness matrix is symmetric, this is identical to the left-hand side of the interior scheme of the grid perturbation. It is therefore reasonable to reuse the same ILU-preconditioned conjugate gradient method for solving these adjoint equations.

Simplifying the boundaries further, $\lambda_o^{(i)}$ and $\lambda_a^{(i)}$ are zero for $i \neq n$, and λ_o need not be computed at all, as it is not used in (3.6) or in (3.7).

While the rest of the adjoint code is written in FORTRAN, these portions, like the grid perturbation code, are written in C++.

4.5 Differentiating the Airfoil Boundary Location

B-splines are used for the airfoil surface parameterization, where the design variables are some of the control point locations. As a result, the sensitivity of the airfoil surface node locations to the design variables is also simple to compute. The B-spline shape is evaluated using (2.1).

The derivative of the airfoil node locations with respect to the design variables is thus

$$\frac{\partial A_j}{\partial X_i} = \begin{bmatrix} 0 \\ B_{i'}(w_j) \end{bmatrix} \tag{4.11}$$

where i' is the index of the control point corresponding to the i^{th} design variable.

4.6 Application to Algebraic Grid Perturbation

An alternative to using the elasticity method to perturb the grid is to use an algebraic approach. One such method is that used by Nemec [25]. Each grid line of constant ξ, extending from the airfoil to the outer boundary, is perturbed individually. The grid is moved according to

$$\bar{G}_{j,k}(A_j) = G^{(0)}_{j,k} + \left(A_j - A^{(0)}_j\right)\left(1 - \frac{L_{j,k}}{L_{j,k_{\max}}}\right) \tag{4.12}$$

where $L_{j,k}$ is the arc length along the grid line.

$$L_{j,k} = \sum_{k'=2}^{k} \sqrt{\left(x^{(0)}_{j,k'} - x^{(0)}_{j,k'-1}\right)^2 + \left(y^{(0)}_{j,k'} - y^{(0)}_{j,k'-1}\right)^2} \tag{4.13}$$

Just as only the y-coordinates of the airfoil surface are perturbed, the grid is also only perturbed vertically. The mesh remains unperturbed downstream of the airfoil, where the grid lines to not touch the airfoil.

When using this perturbation method, the augmented adjoint system is derived using the following definitions for the design objective and flow residual

$$\bar{\mathcal{J}} = \bar{\mathcal{J}}\left(Q, \bar{G}(X), X\right) \tag{4.14}$$

$$\bar{R} = \bar{R}\left(Q, \bar{G}(X), X\right) \tag{4.15}$$

An appropriate Lagrangian is

$$\bar{\mathcal{L}}(\psi, Q, X) = \bar{\mathcal{J}} + \psi^{\mathrm{T}}\bar{R} \tag{4.16}$$

The adjoint equation and gradient evaluation are then

$$\frac{\partial \bar{\mathcal{L}}}{\partial Q} = 0 = \frac{\partial \bar{\mathcal{J}}}{\partial Q} + \psi^{\mathrm{T}}\frac{\partial \bar{R}}{\partial Q} \tag{4.17}$$

$$\frac{\partial \bar{\mathcal{L}}}{\partial X} = 0 = \left.\frac{\partial \bar{\mathcal{J}}}{\partial X}\right|_{\bar{G}} + \frac{\partial \bar{\mathcal{J}}}{\partial \bar{G}}\frac{\partial \bar{G}}{\partial A}\frac{\partial A}{\partial X} + \psi^{\mathrm{T}}\left(\left.\frac{\partial \bar{R}}{\partial \bar{X}}\right|_{\bar{G}} + \frac{\partial \bar{R}}{\partial \bar{G}}\frac{\partial \bar{G}}{\partial A}\frac{\partial A}{\partial X}\right) \tag{4.18}$$

Most of the terms in this equation have the same form as those used in the augmented adjoint method, for elasticity grid perturbation. The one new term here relates to the algebraic grid perturbation. It can be differentiated easily:

$$\frac{\partial \bar{G}_{j,k}}{\partial A_j} = \left(1 - \frac{L_{j,k}}{L_{j,k_{\max}}}\right) \tag{4.19}$$

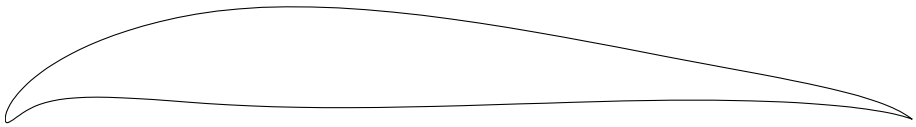

Figure 4.1: An undesirable optimum

4.7 Radius of Curvature Constraint

Four different geometric constraints are introduced in Section 2.4. Three of these constraints were implemented in the legacy code, but the radius of curvature constraint is a new addition.

This constraint is motivated by the avoidance of designs with highly curved portions that the optimizer may find to be beneficial for the selected operating condition, but that do not exhibit good design practice. For example, Figure 4.1 shows an optimized airfoil having a pointed nose that would result in considerable off-design separation. The airfoil also has an abruptly down-turned trailing edge that may not be found to be beneficial at all if the grid were refined.

The ideal solution to these problems is to solve a multipoint optimization problem on a fine mesh. However, for the purpose of verifying the present adjoint method, a less computationally demanding method is sufficient. One alternative is to rely on maintaining the quality of the initial airfoil, and to freeze a number of the control points near the nose and the tail. This approach is relatively effective at the nose, and also reduces the number of design variables. However, when starting with an un-cambered airfoil, it can result in an optimized tail that is abruptly upturned (or rather, the camber line abruptly changes from a downward to a zero slope). The exact effect of freezing a given number of control points is also difficult for the designer to visualize, and results in different shape limitations for different B-spline parameterizations.

Setting a minimum radius of curvature as a constraint allows the designer to set more intuitive targets that apply directly to the geometry, and do not depend on the parameterization. This comes at the expense of increased design variables and a more highly curved design space.

Constraining the radius just at the nose and tail would allow the optimizer to simply move the highly curved areas to just a small distance from the nose and tail. Therefore, the constraint is specified over the entire airfoil, using a piecewise linear target minimum radius of curvature.

The radius of curvature itself is computed at a given node as the radius of the circle passing through it and its two neighbouring airfoil surface nodes. For node j having neighbours a and b, this is given by

$$r_j = \sqrt{\frac{\left((x_b - x_j)^2 + (y_b - y_j)^2\right)\left((x_b - x_a)^2 + (y_b - y_a)^2\right)\left((x_a - x_j)^2 + (y_a - y_j)^2\right)}{x_a y_j - x_a y_b - x_j y_a + x_j y_b + x_b y_a - x_b y_j^2}} \tag{4.20}$$

The penalty for radius of curvature constraint violation is then given by:

$$w_{\mathrm{rc}} \sum_j \left(\frac{r_j}{r_j^*} - 1\right)^2 H\left(r_j^* - r_j\right) \tag{4.21}$$

where w_{rc} is a user-specified weight, r_j^* is the target minimum radius of curvature at the x-coordinate of node j, and $H()$ is the Heaviside step function.

In the formulation of $\partial\mathcal{J}/\partial G$, the derivative of the radius of curvature penalty is formed analytically.

Chapter 5

Validation

5.1 Test Case Details

Three test cases are used. In case A, the design objective is to minimize the drag to lift ratio for a subsonic airfoil. The Mach number is 0.25, and the Reynolds number is 2.88×10^6. The turbulent flow is computed on a 225×49 mesh (11025 nodes). The initial airfoil is a NACA0012, which is parametrized by a B-spline having 13 control points, as shown in Figure 5.1. The three control points at the leading edge and the two at the trailing edge are held fixed, while the y-coordinates of the remaining 8 control points are used as design variables. The angle of attack, initially 4°, is also a design variable. Six minimum thickness targets, given in Table 5.1, are used (enforced with a weight of 0.05). A piecewise linear distribution for the radius of curvature target is given in Table 5.2, and is shown graphically in Figure 5.2 (enforced with a weight of 0.001). The respective properties of the NACA0012 airfoil are included for reference. The radius of curvature target was chosen to be less than the radius of curvature of the NACA0012 and of the RAE2822 airfoil used in case C. In the critical areas of the nose and tail, the specified minimum radius of curvature is relatively close to the radius of curvature of the RAE2822.

The second case is a transonic airfoil, where the design objective is to minimize drag at a fixed lift coefficient. The Mach and Reynolds numbers are 0.78 and 2.0×10^7, respectively. The target lift coefficient is 0.75, and the target drag coefficient is 0.008. An enclosed area target of 0.8 times the area of the initial airfoil, a NACA0012, is used. Thickness constraints are 0.012 at a chordwise coordinate of 0.95, and a 0.11 anywhere between chordwise coordinates 0.15 and 0.4. Radius of curvature targets are the same

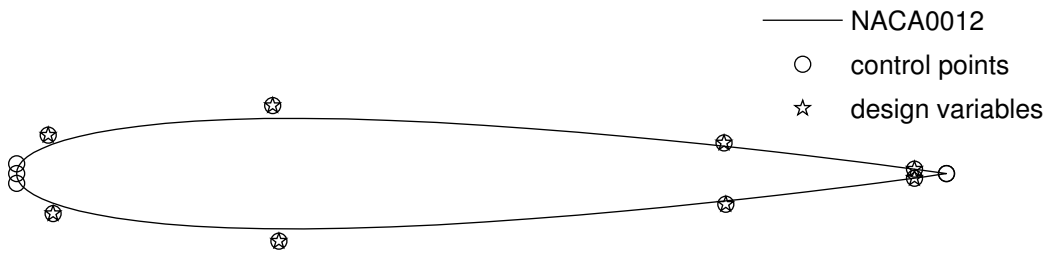

Figure 5.1: Airfoil parameterization and design variables, cases A and B

Table 5.1: Thickness targets (case A)

Chordwise coordinate	Target thickness	NACA0012 thickness
0.05	0.04	0.0706
0.35	0.11	0.1177
0.65	0.04	0.0809
0.75	0.03	0.0612
0.85	0.026	0.0389
0.95	0.012	0.0137
0.99	0.002	0.0028

Table 5.2: Radius of curvature target

Chordwise coordinate	Target radius of curvature	NACA0012 radius of curvature
0	0.01	0.016
0.005	0.01	0.030
0.2	0.3	1.357
0.8	0.3	7.528
1	0.2	5.551

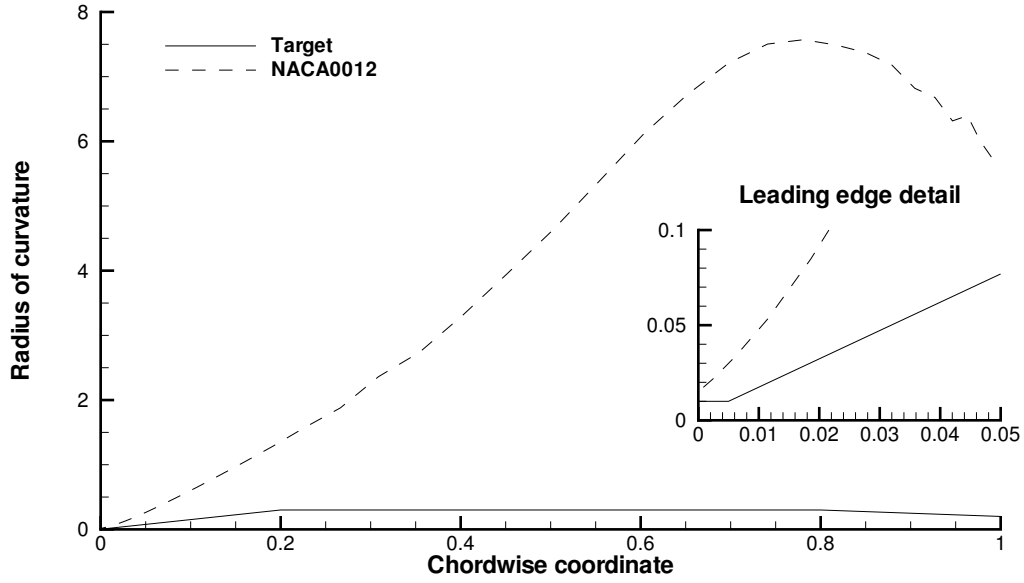

Figure 5.2: Radius of curvature target

as in case A. The weights for lift, drag, area, thickness, and radius of curvature are 10, 0.1, 1, 1, and 0.005, respectively.

In the third test case, the initial airfoil is the RAE2822, and a 198×60 mesh (11880 nodes) is used. As shown in Figure 5.3, the airfoil is parameterized with a B-spline having 15 control points, 10 of which are used as design variables. The objective, flow conditions, and constraints match those used in case B (note that 0.8 times the area of the NACA0012 is equivalent to 0.82958 times the area of the RAE2822). A comparison between the results of cases B and C can help in assessing the globality of the optimum, as well as the influence of the computational mesh and the airfoil parameterization.

The test cases were run using an Athlon64 3500+ processor with a clock speed of 2.2 GHz.

5.2 Gradient Accuracy

The accuracy of the gradient computed by the augmented adjoint method can be determined by comparing the gradient to a benchmark. The simplest benchmark is the finite difference method. Second-order centred finite differences are selected as a compromise between accuracy and computational expense.

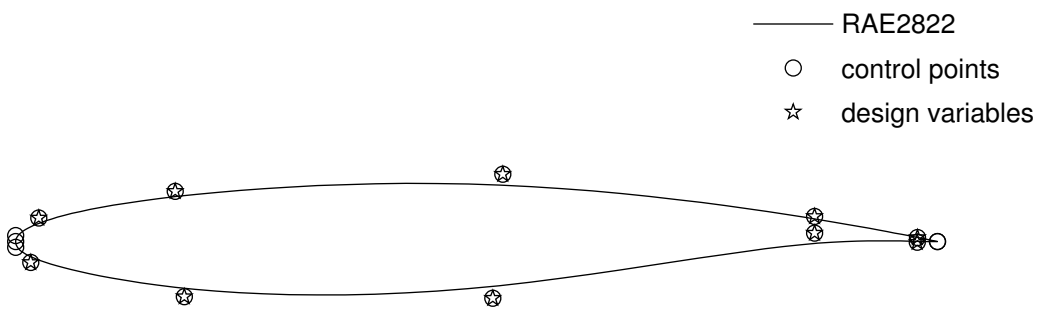

Figure 5.3: Airfoil parameterization and design variables, case C

Figure 5.4: Cancellation error in second-order centred differencing

$$\frac{\partial f}{\partial x} = \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} + O(\epsilon^2) \tag{5.1}$$

Finite differences are susceptible to two types of errors. First is the truncation of the Taylor series, as shown by the error term in (5.1). The second error is the cancellation error that occurs when subtracting the floating point numbers $f(x+\epsilon)$ and $f(x-\epsilon)$, as shown in Figure 5.4. The first error usually decreases as $\epsilon \to 0$, while the second error increases as $\epsilon \to 0$, so there is generally an optimal step size ϵ' that gives the least error. In order to improve the ability to validate the augmented adjoint method, some effort was made to estimate the finite differencing error, and to find the optimal step size.

The truncation error was estimated by computing a fourth-order finite difference using the same step size. The difference between the second- and fourth-order estimates is taken as the truncation error.

The cancellation error is more difficult to predict. The number of bits of agreement between the mantissas of $f(x+\epsilon)$ and $f(x-\epsilon)$ gives an estimate of how many *additional* bits are erroneous in their difference. However, the total error in the difference depends ultimately on the amount of error in $f(x+\epsilon)$ and $f(x-\epsilon)$. For the function being differentiated, this error is impractical to compute. Therefore, the cancellation error is avoided, and error estimation is done by selecting a step size where the truncation error dominates. This can be identified by plotting the truncation errors for several step sizes

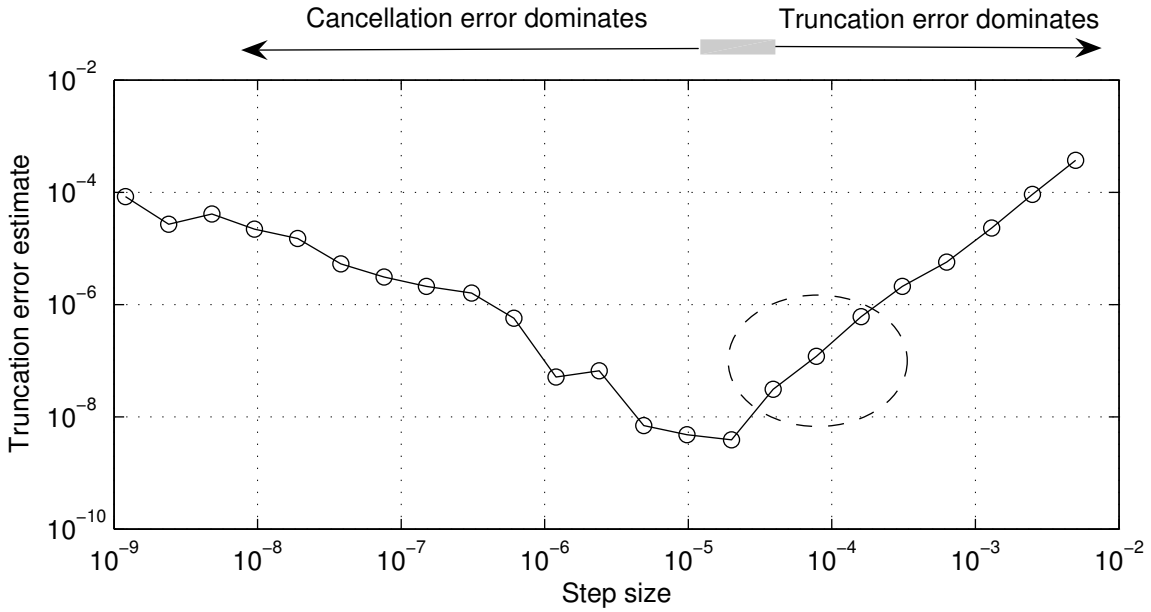

Figure 5.5: Truncation error estimate in finite differences

on logarithmic axes, and selecting the linear region having a slope of about 2 (the order of the method). An example of this is shown in Figure 5.5. The jagged shape to the left indicates that cancellation errors are corrupting the truncation error estimate, and the circled region shows step sizes where the truncation error estimate dominates and where the error is reasonably low.

Another gradient calculation method that is a useful benchmark is the original adjoint formulation, as described in Section 1.1. In (1.3), the terms $\partial\tilde{\mathcal{J}}/\partial X$ and $\psi^{\mathrm{T}}\partial\tilde{R}/\partial X$ are formed using finite differences. As in the basic finite difference approach, some effort was made here to estimate the error at a near-optimal step size.[1] Once the same technique is used to find the error in each of $\partial\tilde{\mathcal{J}}/\partial X$ and $\psi^{\mathrm{T}}\partial\tilde{R}/\partial X$, they are summed to estimate the error in the gradient. This does not include error incurred in the finite difference approximation used to compute $\partial\tilde{\mathcal{J}}/\partial\tilde{Q}$, or in the solution of the flow adjoint system.

Using this technique, Figure 5.6 was constructed for case A, using the elasticity grid perturbation method with $n = 2$. It shows the relative difference between the initial gradient computed using the augmented adjoint method and that computed using each

[1]This error estimation process is much more computationally intensive than a simple finite difference, and the step size selection is not automated, so it is used only for validating the derivatives, and not in the optimization runs in later sections.

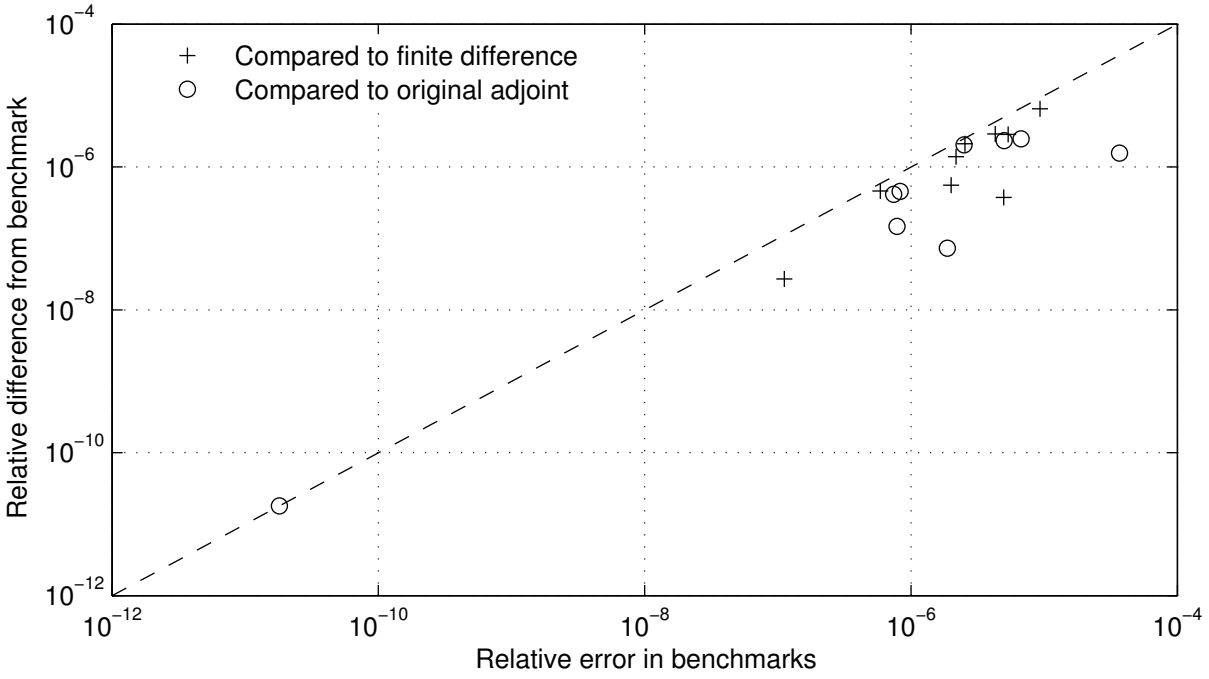

Figure 5.6: Validation of augmented adjoint gradient, case A

of the two benchmarks: finite differencing and the original adjoint method. The multiple points on the plot refer to different entries in the gradient vector. As both benchmarks have error in them, the estimate of this error is included on the horizontal axis. The fact that all points lie beneath the dashed line indicates that the augmented method is at least as accurate as both of the benchmarks. Referring to the relative difference, at least the first five nonzero digits are accurate.

For case B, the comparison of the augmented adjoint gradient to the benchmarks is shown in Figure 5.7. It can be seen that the method agrees very well with the original adjoint method. When comparing against finite differences, at least four digits agree, but one would expect better agreement, given the error estimate for the finite differences.

The principal cause of this lack of agreement is that the flow residual is a non-smooth function. In Appendix B, examples of absolute values and switches in the calculation of R can be found in the artificial dissipation (B.25, B.40) and turbulence modelling (B.70, B.74) terms, as well as in the choice between inflow (B.120) and outflow (B.122) at the far-field boundary.[2] A discussion of the treatment of a specific portion of the artificial dissipation terms is provided in Section B.3. Nemec [25] also found these effects to limit

[2]The pressure switch in the artificial dissipation and the circulation correction could also cause similar problems. Both of these features have been deactivated in the cases presented here.

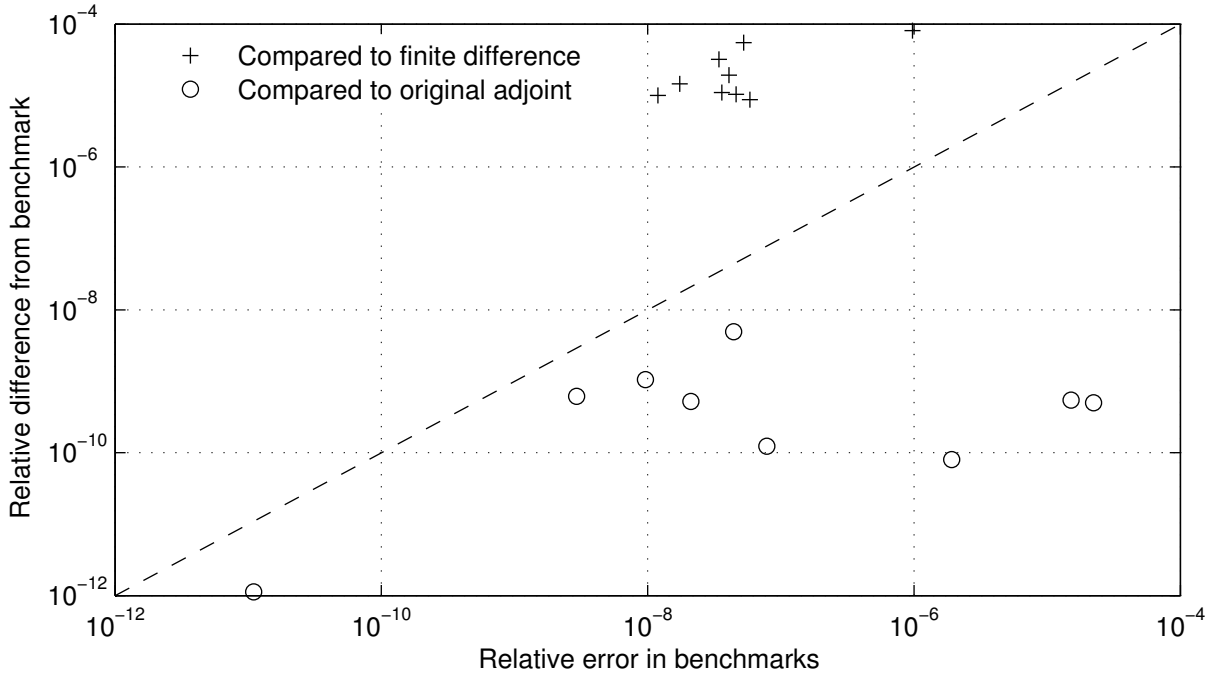

Figure 5.7: Validation of augmented adjoint gradient, case B

the gradient accuracy, and noted a similar magnitude of discrepancy between the finite differenced and adjoint gradients. As this error is in the flow Jacobian, a term common to both the original and augmented adjoint methods, it is reasonable that both adjoint methods differ from finite difference results by about the same amount.

Other factors that may contribute to this disagreement between the augmented adjoint gradient and the finite difference gradient are less important, and include

- Inaccurate estimation of finite differencing errors. The errors in the finite differences may not be adequately estimated by only the next two terms in the Taylor series (i.e. comparing second- and fourth-order finite differences);

- Errors in the solution of the augmented adjoint equations. For these equations, residual convergence was attained to a tolerance of 10^{-15}.

- Numerical errors that occur in the double precision operations involved in forming the terms for the augmented adjoint method.

A similar plot of the gradient accuracy for case C is shown in Figure 5.8. It exhibits similar characteristics to those found in case B, and has the same explanation for the apparent disagreement with finite differences.

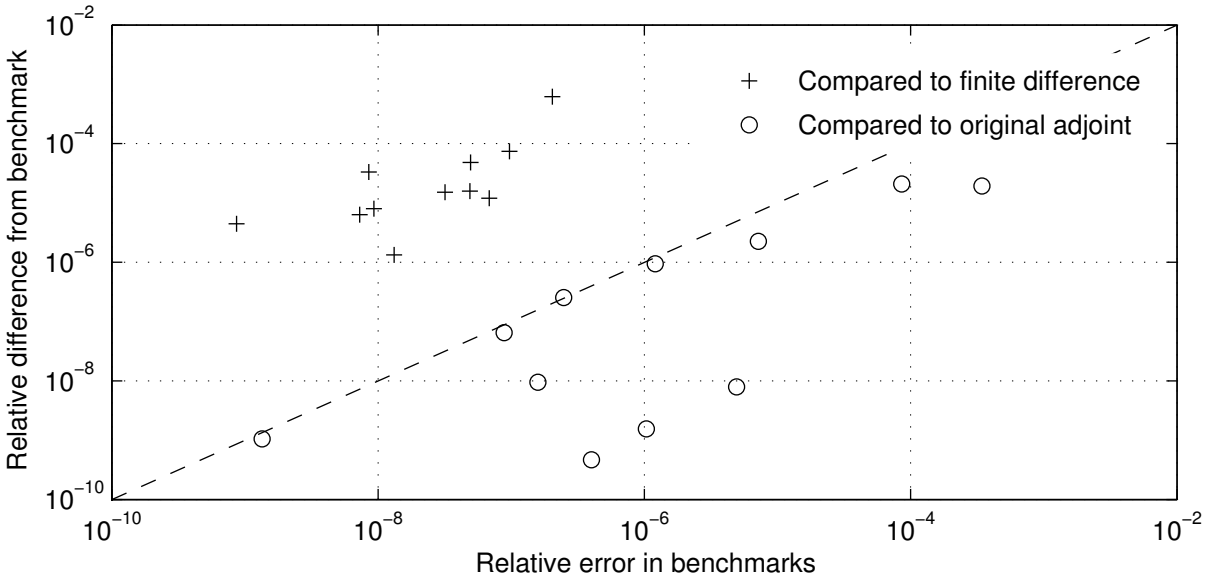

Figure 5.8: Validation of augmented adjoint gradient, case C

5.3 Gradient Evaluation Time

The CPU time required to evaluate the gradient was investigated, considering different grid perturbation techniques, gradient evaluation methods, and numbers of design variables. The number of design variables is varied by using a B-spline having five more control points than the desired number of design variables, so that the two control points at the trailing edge and the three at the leading edge remain fixed.

In each of Figures 5.9–5.11, the gradient evaluation time for the second design step in case A is plotted against the number of B-spline design variables (i.e. all design variables except the angle of attack), and the augmented adjoint method is contrasted with the original adjoint method. In Figure 5.9, algebraic grid perturbation is used. This shows little difference between the two gradient evaluation techniques, although the original method is slightly slower, and takes longer as the number of design variables is increased. This difference is indicative of the speed of the analytic calculation of $\partial R/\partial G$. The time taken by the augmented method actually decreases some with an increasing number of design variables; this is likely noise due to slightly different convergence of the solvers, as the airfoils for different surface parameterizations are not exactly the same.

Figures 5.10 and 5.11 use the elasticity grid perturbation method, with the number of increments, n, being 1 and 2, respectively. The time taken by the augmented method varies by less than 5% with differing numbers of design variables, and does not show an

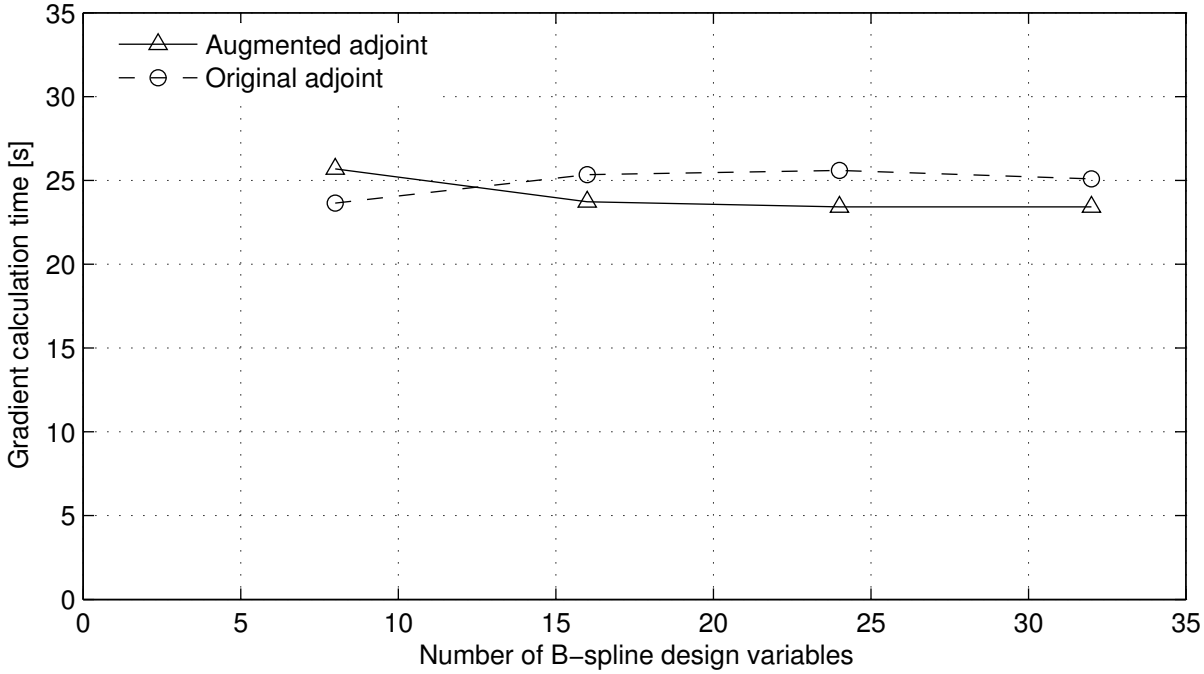

Figure 5.9: Gradient calculation time: algebraic method, case A

increasing trend. Additionally, the augmented adjoint method is several times faster than the original adjoint method, whose time requirements increase linearly with the number of design variables. The evaluation time for the original adjoint method is several times longer than that of a flow solve (near 50 seconds), so is the dominant computational cost in an optimization.

In an entire optimization, the gradient evaluation time is essentially constant, despite warm-starting each $\lambda^{(i)}$. This is shown in Figure 5.12, for case A with 8 design variables, using the elasticity grid perturbation with $n = 2$. The figure also shows that the function evaluation (grid perturbation plus flow solution) grows increasingly faster as the optimization progresses. This is due to warm-starting; the flow solve gradually becomes faster, while the grid perturbation accelerates abruptly near the end of the optimization. The gradient calculation initially requires about 80% as much time as the function evaluation.

The timing results for case B, shown in Figures 5.13–5.15, are very similar to those found in case A. Near the beginning of the optimization, the gradient calculation using the augmented adjoint method was found to require about 70% of the time required for a function evaluation.

The timing was not studied in great detail for case C, but near the beginning of the

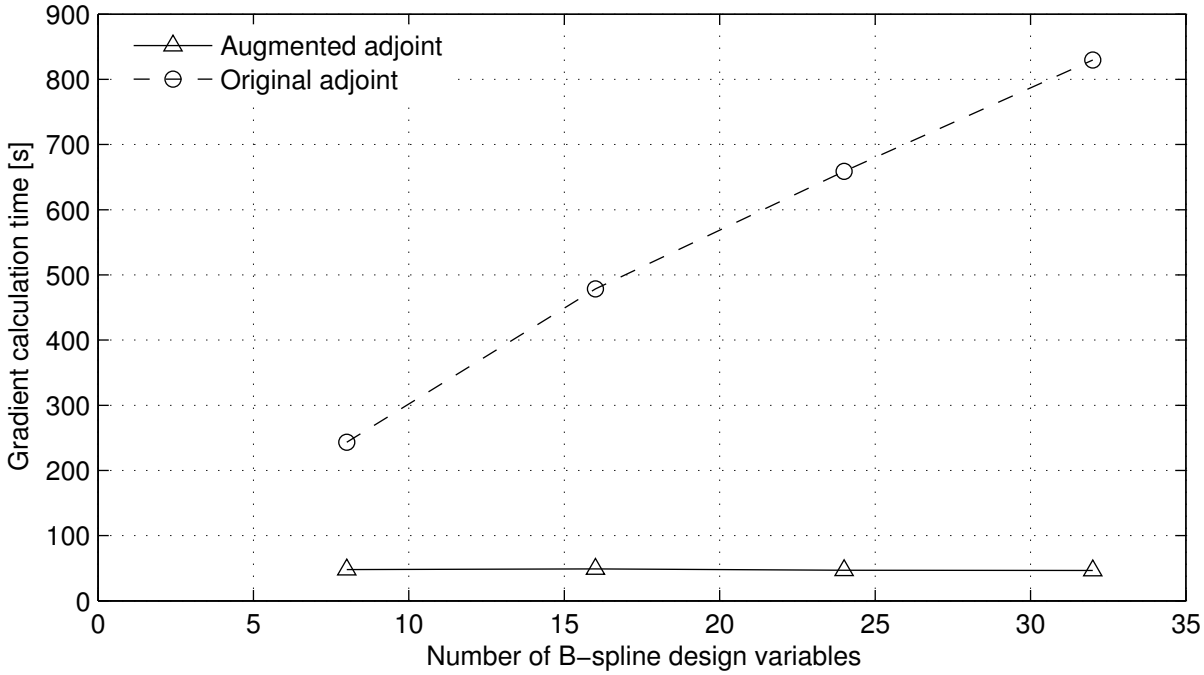

Figure 5.10: Gradient calculation time: elasticity method with $n = 1$, case A

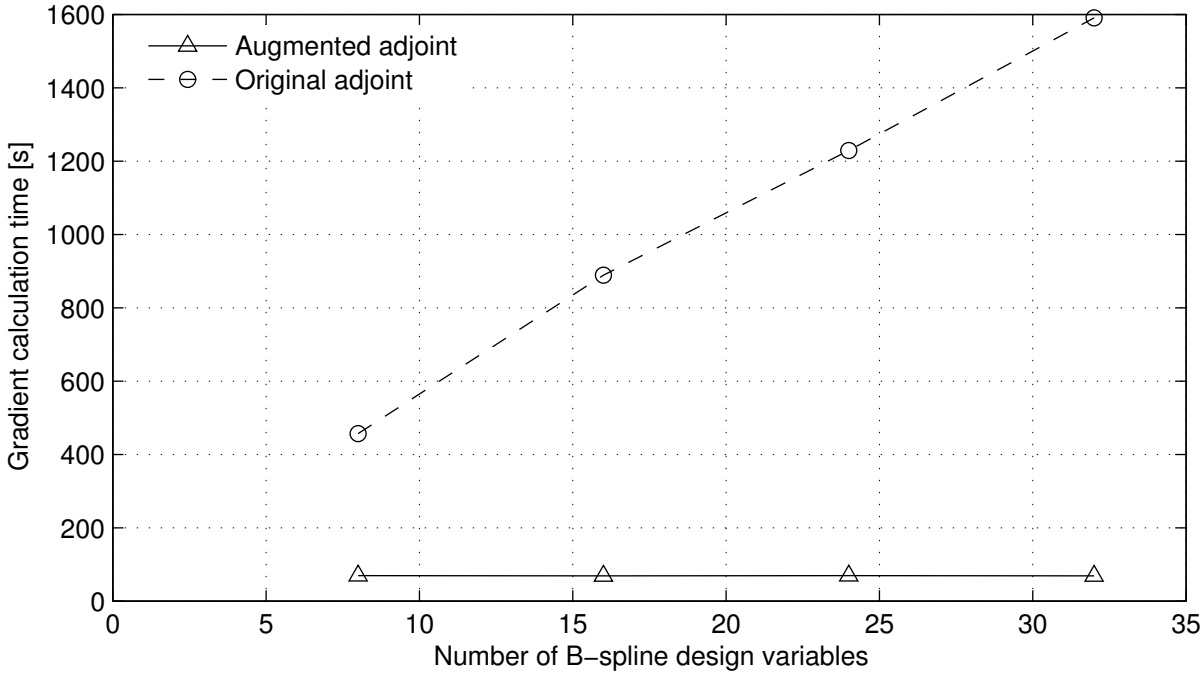

Figure 5.11: Gradient calculation time: elasticity method with $n = 2$, case A

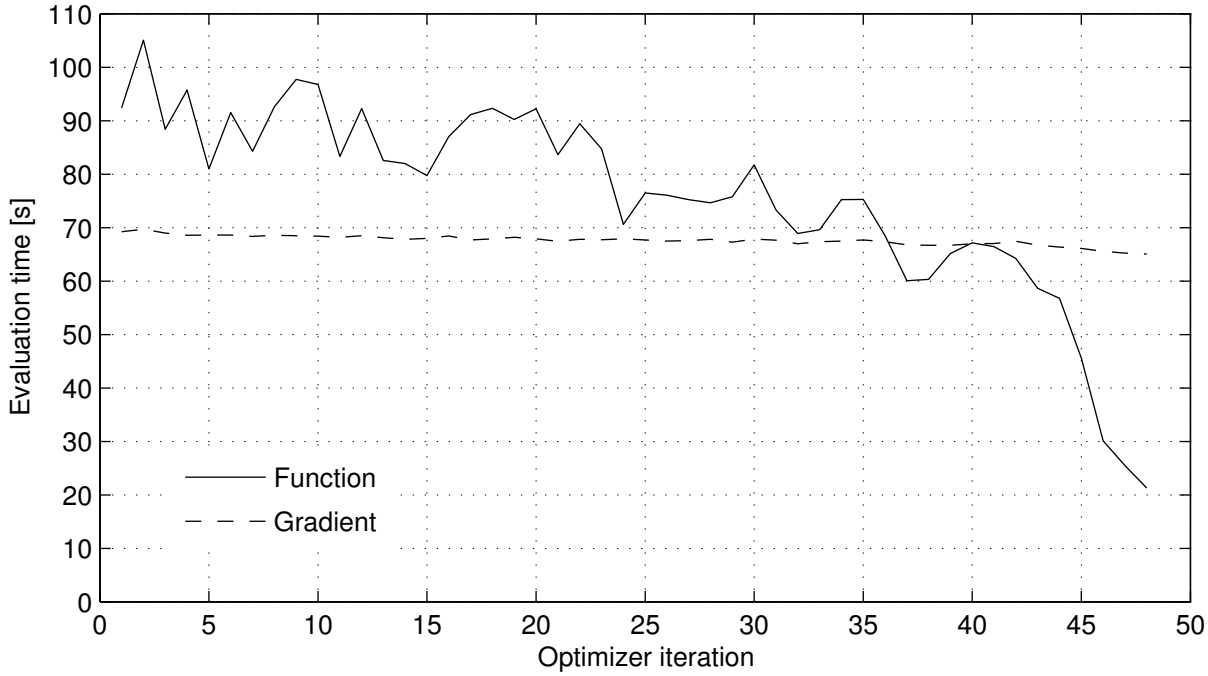

Figure 5.12: Change in evaluation times during optimization

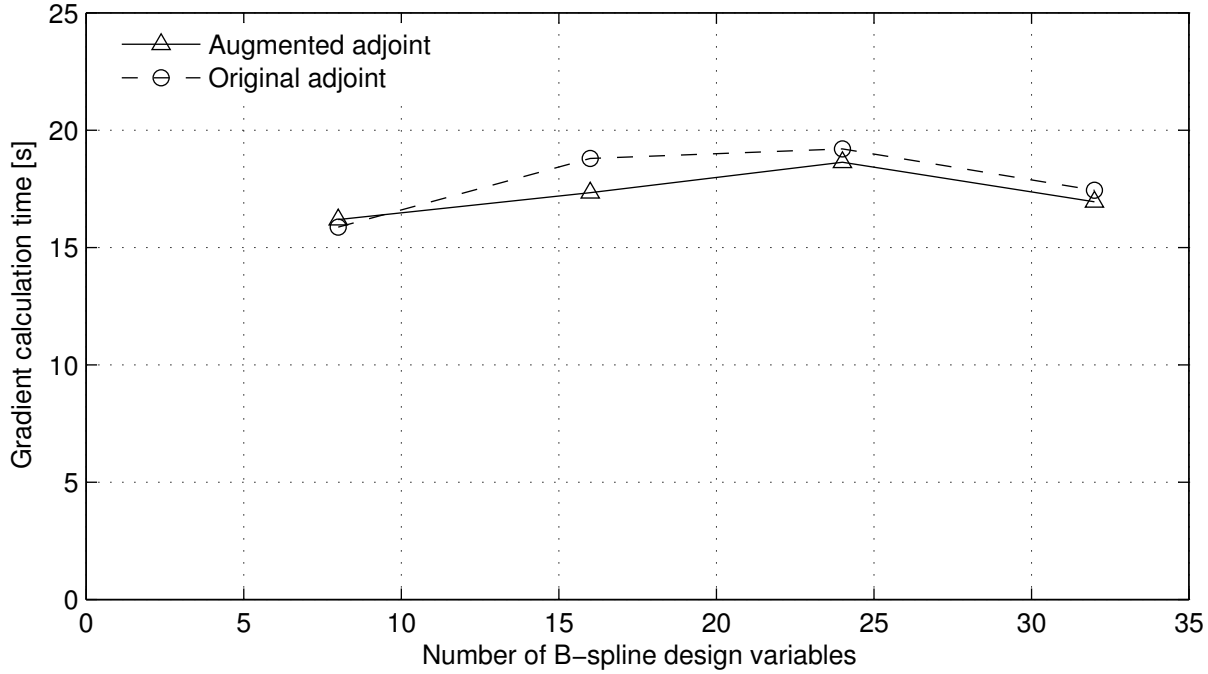

Figure 5.13: Gradient calculation time: algebraic method, case B

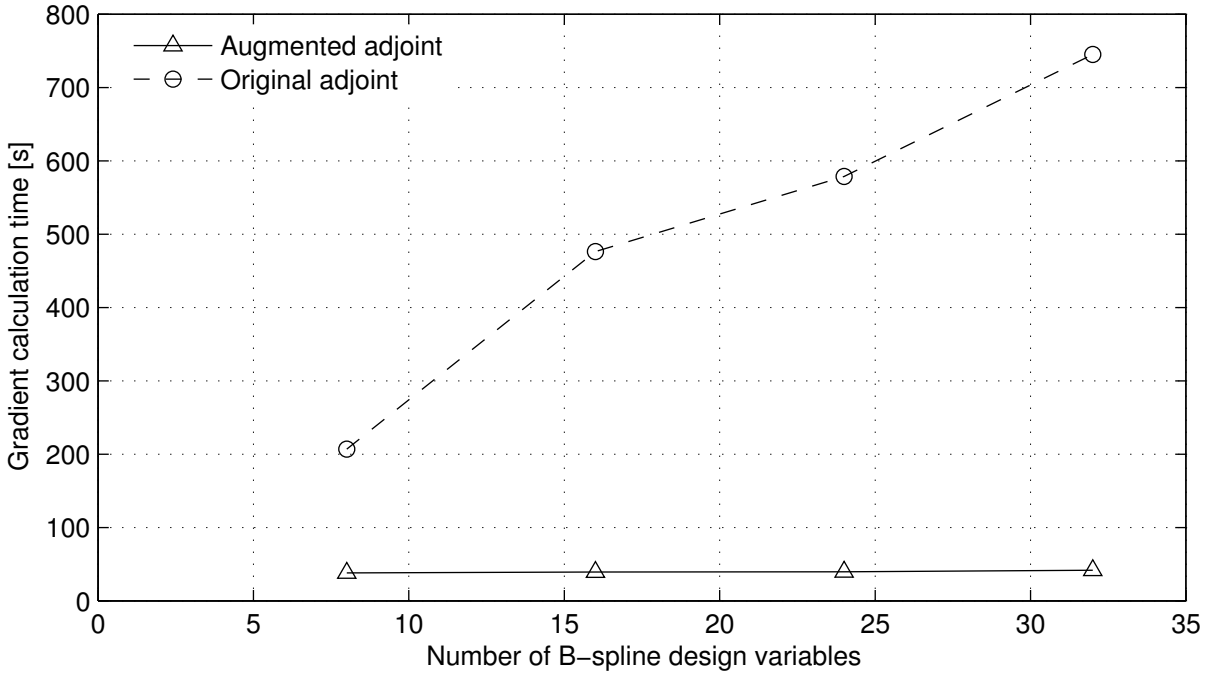

Figure 5.14: Gradient calculation time: elasticity method with $n = 1$, case B

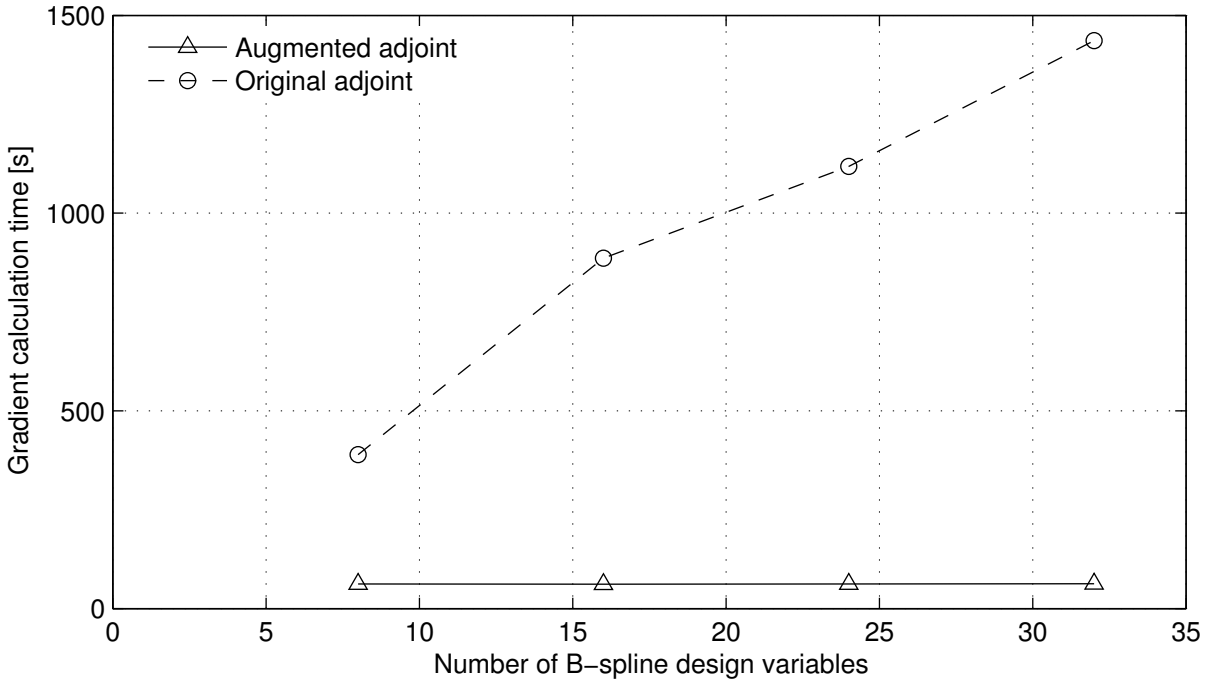

Figure 5.15: Gradient calculation time: elasticity method with $n = 2$, case B

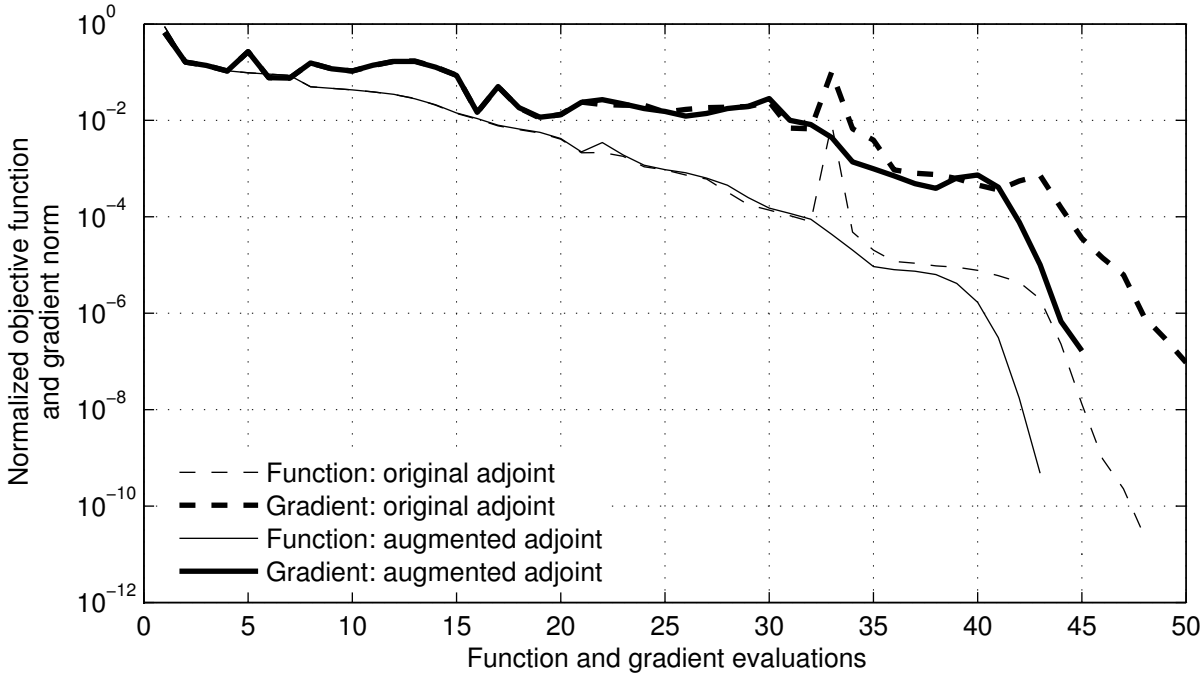

Figure 5.16: Optimizer convergence for case A: algebraic method

optimization, the augmented adjoint gradient evaluation required about three quarters of the time of a flow solve, and about one fifth of the time of a gradient evaluation by the original adjoint method. This is comparable to the results found in the other cases.

5.4 Optimizer Convergence

The convergence history for case A is shown in Figure 5.16 for the algebraic grid perturbation, and in Figures 5.17 and 5.18 for the elasticity grid perturbation with $n = 1$ and 2, respectively. The figures show the norm of the objective function gradient, and a normalization of objective function, given by

$$\mathcal{F}_{\text{normalized}} = \frac{\mathcal{F}}{\mathcal{F}_{\min}} - 1 \tag{5.2}$$

where $\mathcal{F}_{\min}$ is the minimum objective function value found. This accentuates the difference between the objective function at a given increment and the optimal value. The tolerance for gradient convergence was set lower than could be attained, so that the optimizer eventually stalls in each case, and each method's convergence ability is exposed.

The curves for the normalized objective function show that the first two nonzero digits of the objective function value are determined after the first 15–20 iterations. At the end

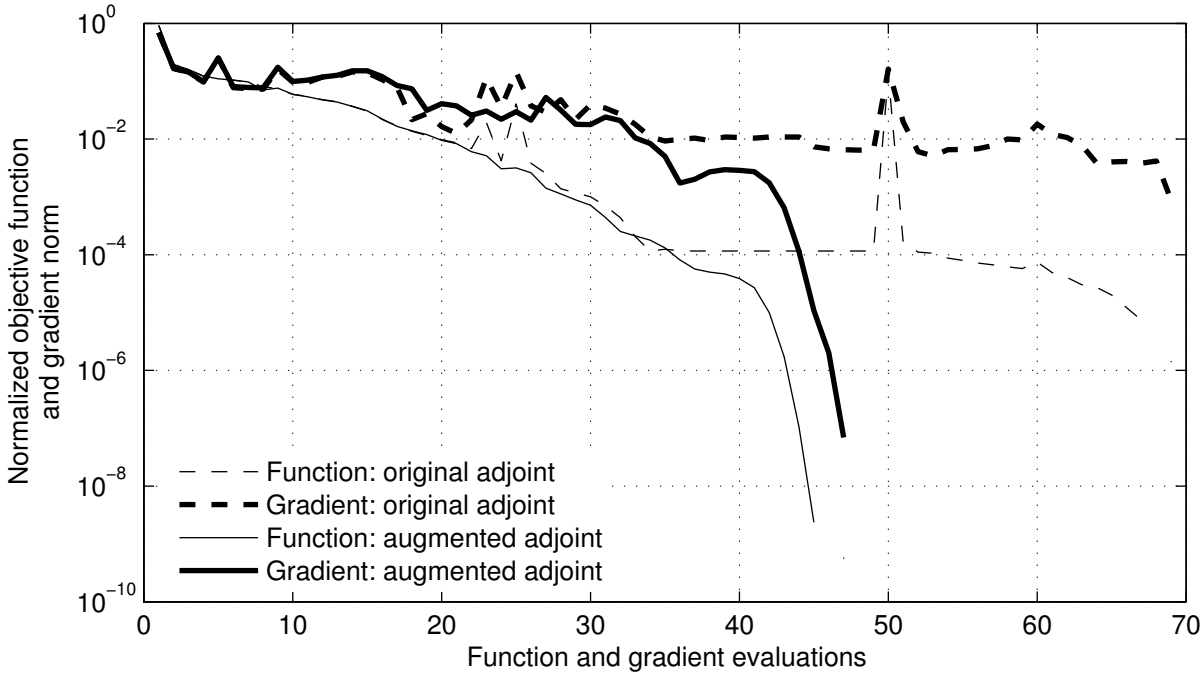

Figure 5.17: Optimizer convergence for case A: elasticity method with $n = 1$

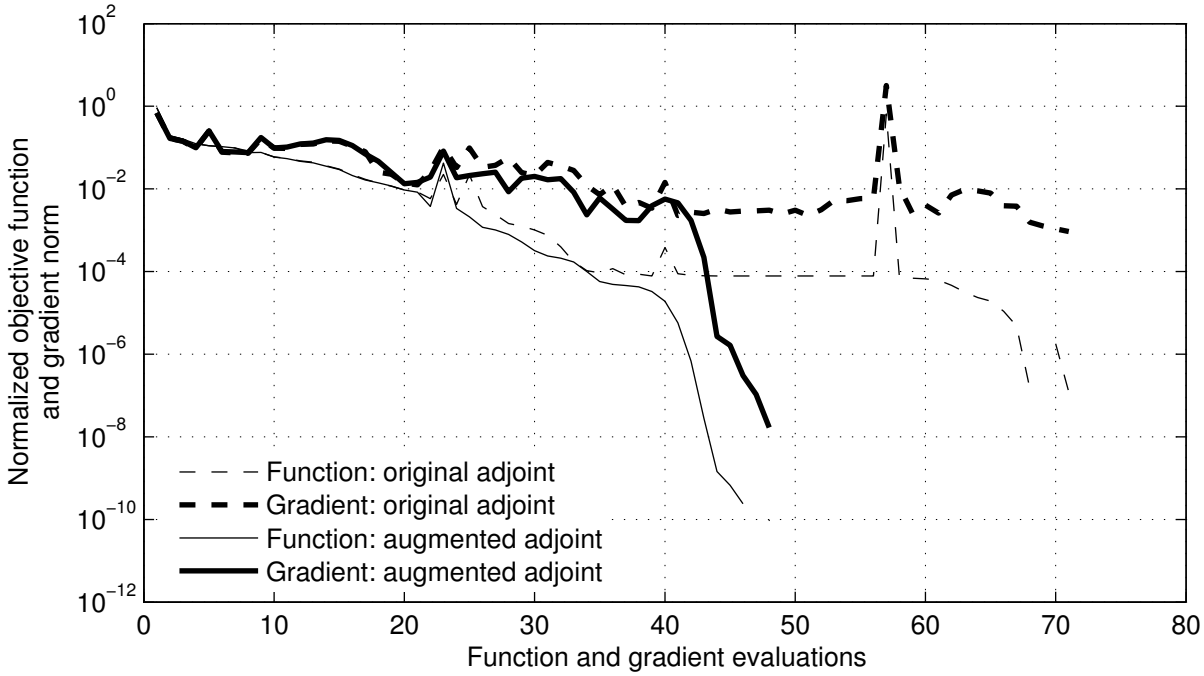

Figure 5.18: Optimizer convergence for case A: elasticity method with $n = 2$

of the optimization, between 5 and 10 digits of the objective function are unchanging.

In all cases, the original and augmented adjoint methods are very similar for the first several iterations. As the optimization progresses, the differences between the methods accumulate and their convergence histories diverge. In the end, it can be seen that the gradient converges to about the same level for both gradient calculation methods, when using the algebraic grid perturbation scheme.

When the elasticity method is used to perturb the grid, however, the augmented adjoint method allows much tighter convergence of the gradient than does the original adjoint method (about 4 orders of magnitude difference). This is due to convergence difficulties experienced when using the original adjoint method, and can be explained by gradient inaccuracy. Considering the gradient evaluation (1.3), some accuracy is lost in the finite differencing used to find $\partial\tilde{\mathcal{F}}/\partial X|_{\tilde{Q}}$ and $\partial\tilde{R}/\partial X|_{\tilde{Q}}$. The two quantities are then added together. Near convergence of the optimizer, their sum is a near-zero quantity, and so several of the leading digits cancel, and more cancellation error is incurred. When using the elasticity method, this effect is aggravated by increased error in the flow solver residual calculation, because the grid perturbation equations are solved imperfectly (though tolerance of 10^{-15} is used).

To verify that this is the cause, an error estimation like that in Section 5.2 was performed using the optimized airfoil produced using the augmented adjoint method ($n = 2$). At this optimal point, the error (estimated as in Section 5.2) in the gradient computed using the original adjoint method is so large that the sign of some entries in the gradient vector is uncertain.[3] This would clearly prevent the optimizer from converging as tightly as is permitted by the augmented adjoint method. Furthermore, these errors can be expected to be even larger in the actual optimization run, where the finite differencing step sizes are not as meticulously chosen.

Despite the fact that the convergence histories are different when using the original and augmented adjoint methods, they result in very similar optimized airfoils. The largest vertical displacement between the optimized airfoils is less than 10^{-6}, when the algebraic grid perturbation is used. For the elasticity grid perturbation, the difference is less than 10^{-3}; this is expected, given the loose convergence obtained with the original adjoint method.

The optimized airfoils found using the augmented adjoint method, for case A, are

[3]The same is true of the accuracy of the gradient computed with finite differences.

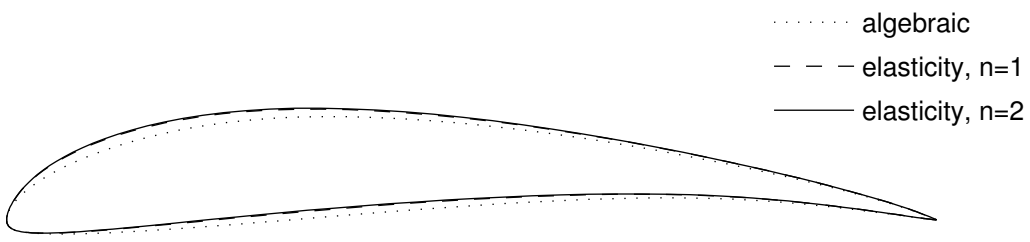

Figure 5.19: Optimized airfoils for case A

shown in Figure 5.19. While the optimal airfoil shape does not depend much on the gradient calculation method, the figure shows that the shape is noticeably dependent on the grid perturbation technique used. The two variants of the elasticity give very similar results, but the algebraic method gives a noticeably different shape. This may be a sign that the optimizer is manipulating the grid perturbation algorithm to alter the flow solution. These changes in the flow solution are likely due to errors resulting from the thin layer approximation: the algebraic grid perturbation does not maintain grid orthogonality near the airfoil's surface. It is important to use a grid perturbation algorithm that resists this sort of exploitation in order to ensure a physically realistic optimum.

To explore this, a new mesh was regenerated for each of the optimized airfoils, and the objective functions were recomputed on the new meshes. These objective functions include the constraint penalties; to maintain consistency, the penalties are not recomputed on the regenerated mesh, but are retained from the optimized result. These values, given in Table 5.3, show that the elasticity method with $n = 2$ provides the shape having the best performance, as computed on the regenerated mesh. Also, the comparison between the result computed on the perturbed and regenerated meshes is best for the elasticity method with $n = 2$, and worst for the algebraic method. This favours the elasticity method with $n = 2$.

Using the regenerated mesh for the shape optimized with algebraic grid perturbation, the optimizer was restarted. At the first iteration of this restart, the objective function value was 0.017005—noticeably larger than the result found with the regenerated mesh matching the optimized shape from the first optimization. The reason for this is differing

Table 5.3: Optimized airfoil performance using perturbed and regenerated meshes

Perturbation method	Objective function		
	Perturbed mesh	Regenerated mesh	Difference
Algebraic	0.016993	0.016685	0.000307
Algebraic, restarted	0.016791	0.016807	0.000016
Elasticity ($n = 1$)	0.016694	0.016744	0.000050
Elasticity ($n = 2$)	0.016681	0.016668	0.000013

treatment of constraints: when computing the objective function for the regenerated mesh matching the optimized shape from the first optimization, the numeric values for the constraint penalties were taken from those computed at the end of the optimization. However, for the first iteration of the restart, the constraints are computed using the new B-spline parameterization, distribution of airfoil surface nodes, and grid. There are therefore numerical differences between the two constraint penalties. Table 5.3 shows that the restart, which involves smaller shape changes, achieves better agreement between the optimized objective function found on the perturbed and the regenerated meshes. Due to the new B-spline, the restart is not the same optimization problem as before, so the actual objective function should not be compared to the other methods. However, the optimized airfoil is quite similar to that obtained with the elasticity method.

The effect of increased dimensionality on the airfoil shape is also interesting. If the number of design variables is increased to 24, then the optimizer has more control over the shape of the airfoil, and tries to circumvent the active constraints. This is clearly visible in Figure 5.20, where the optimized airfoils are thinner between thickness constraints. As mentioned in Section 2.4, multidisciplinary, multipoint optimization could be used to yield a more realistic design.

The optimizer convergence for case B shows trends similar to those found in case A. The normalized function and gradient convergence for the case where the elasticity method is used with $n = 2$ is shown in Figure 5.21. The figure shows that the augmented adjoint method actually takes one iteration longer to converge, but that it converges about three orders of magnitude more tightly.

Using the same grid perturbation, the difference between the y-coordinates of the optimized airfoils obtained using the original and augmented adjoint methods is less than 10^{-4}. The shape of the optimized airfoil is shown in Figure 5.22. The difference

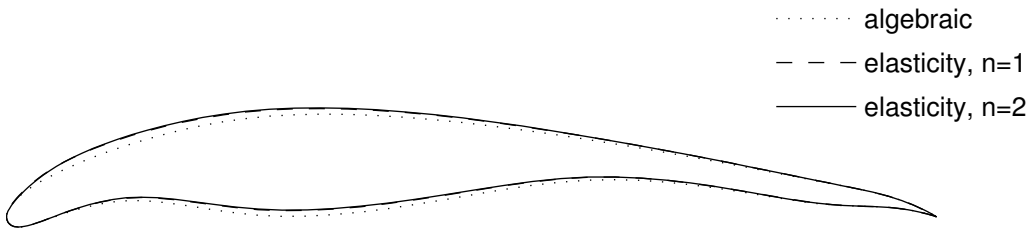

Figure 5.20: Optimal airfoils for case A, using 24 B-spline design variables

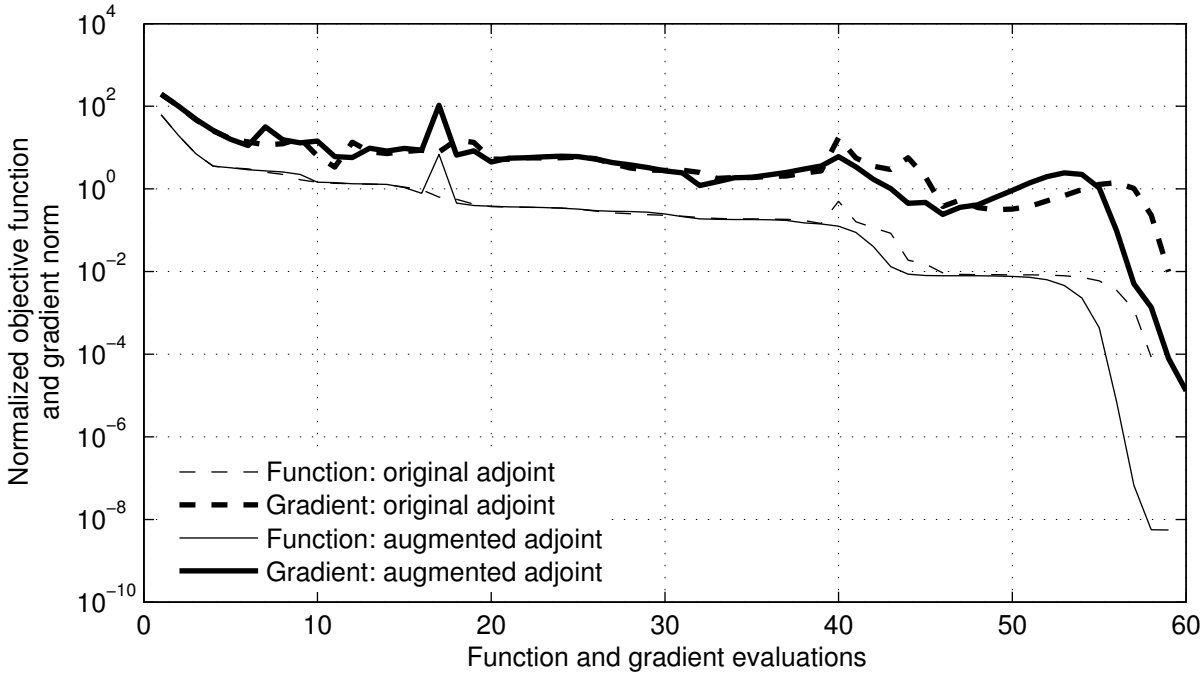

Figure 5.21: Optimizer convergence for case B: elasticity method with $n = 2$

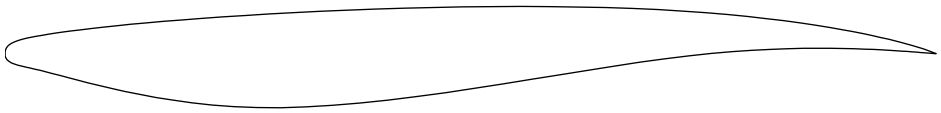

Figure 5.22: Optimized airfoil for case B

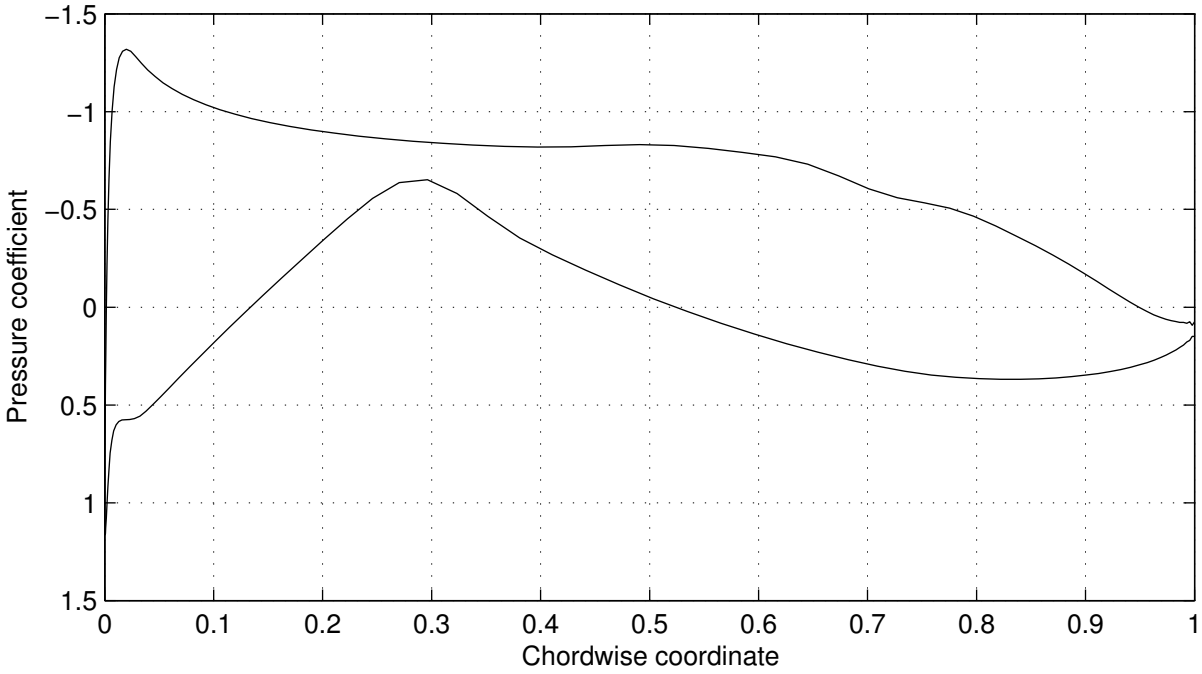

Figure 5.23: Optimized pressure distribution for case B

between the airfoils obtained with different grid perturbation methods is not significant in this case—less than 10^{-3}. This differs from case A, where using the algebraic grid perturbation gave a noticeably different airfoil. The reason for the difference is likely that shock elimination dominates the transonic problem, whereas viscous drag is more important for the subsonic airfoil—and thus so is mesh orthogonality. The optimized airfoil for case B is shock-free, as can be seen in the pressure distribution, Figure 5.23.

One of the fundamental limitations the BFGS optimizer is that it drives toward local, rather than global, optima. However, starting the BFGS optimizer from different starting points can increase confidence in the globality of the optimum—just as confidence is gained by multiple runs of a stochastic optimizer. In case C, the starting point is different

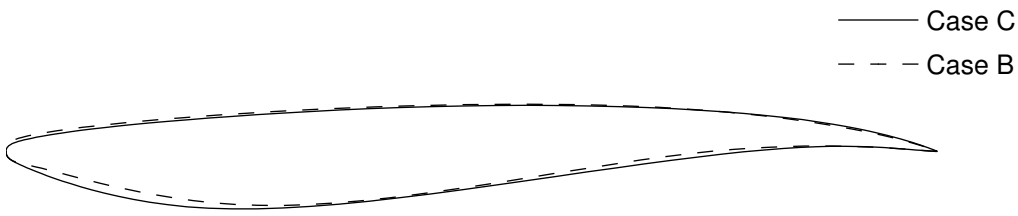

Figure 5.24: Optimized airfoils for cases B and C

from case B, but the optimization problem is essentially the same. The optimized shapes obtained in both cases are shown in Figure 5.24.

It can be seen that the two cases result in fairly similar airfoils; the upper surfaces are nearly identical. There is a small but noticeable difference between the lower surfaces near the nose and the tail. This agreement in shape is quite good, considering that the two cases have different numbers of design variables, and that increasing the number of design variables resulted in quite different optimal shapes in case A (see Figures 5.19 and 5.20). The agreement between the two optimized airfoils can also be viewed as good when considering that grids of only moderate density are used, so that grid convergence of the drag coefficient is not expected. This suggests that the globally optimal shape, free of parameterization and discretization limitations, is relatively well-described by the solutions to cases B and C.

Finally, the optimizer convergence plot for case C is shown in Figure 5.25. As in the previous cases, it shows the augmented adjoint method to allow considerably tighter gradient convergence than does the original adjoint method. Considerable improvement is made in the first dozen iterations, then the bulk of the optimization time is spent finding the active constraints, and the quadratic convergence of the optimizer is clearly visible at the end of the plot.

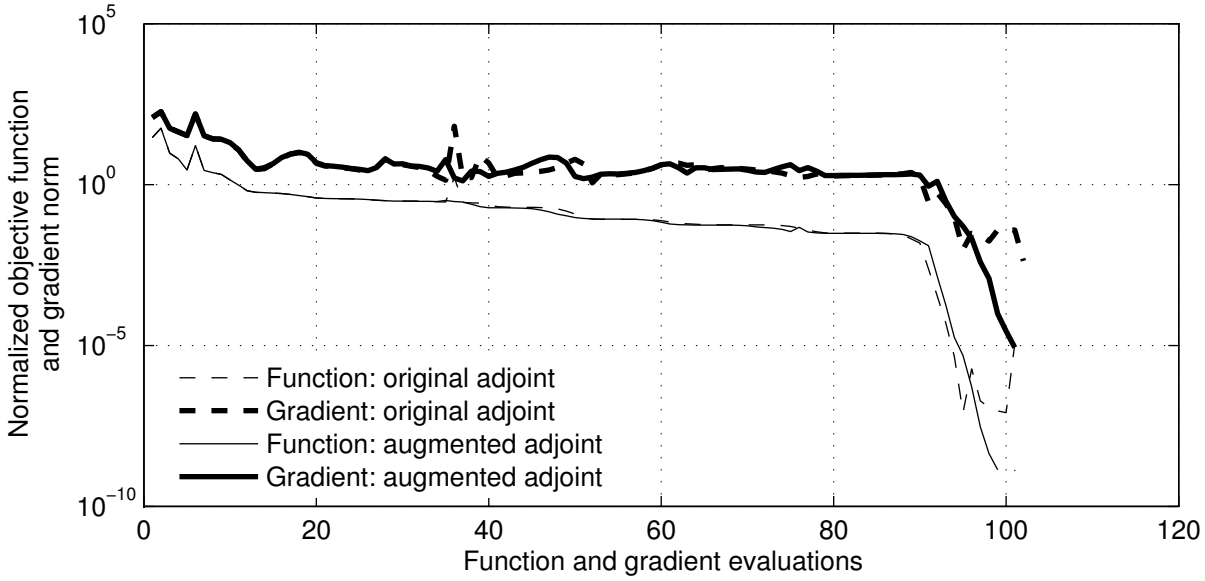

Figure 5.25: Optimizer convergence for case C

Chapter 6

Conclusion

6.1 Contributions

For an aerodynamic shape optimizer using either an algebraic method or the elasticity method for grid perturbation, the discrete adjoint equations have been augmented to explicitly include the grid perturbation. The accuracy and speed of the method have been studied, as has its effect on the optimizer performance.

Unlike previous adjoint methods that explicitly include grid sensitivities [23, 30, 24], this method treats a grid perturbation scheme that perturbs the grid in several increments.

A direct comparison of derivative accuracy, using finite differences and the original adjoint method as benchmarks, shows the augmented adjoint method is accurate. Additionally, the use of the augmented adjoint method allowed the optimizer to converge much more tightly than was possible using the original adjoint method. This is indicative of increased gradient accuracy, and provides the designer with increased confidence in the validity of the solution to the optimization problem.

The augmented adjoint method met the expectation of having a run time that is independent of the number of design variables. This run time is found to be about 70–80% of that of a function evaluation (grid perturbation plus flow solve), and represents a drastic improvement over the original adjoint method. When using the augmented adjoint method, the optimizer was sometimes able to converge in up to 30% fewer iterations, further decreasing the run time.

For subsonic airfoils, the optimized airfoil shape was found to be different, depending on whether the elasticity grid perturbation method or an algebraic grid perturbation

method was used. The optimized airfoil computed with the elasticity method with multiple increments was the best solution to the optimization problem.

The price paid for these performance improvements is the human time required to derive and program the augmented adjoint equations. The months taken to complete the implementation were mainly spent hand-coding functions to compute each of the terms required by the adjoint equations. Of the several terms to be handled, two were particularly time consuming: $\partial R/\partial G$ and $\partial r^{(i+1)}/\partial G^{(i)}$. The application of this method to different flow solvers and grid perturbers will therefore be toilsome.

6.2 Cost-Benefit Assessment

The underlying question of this thesis is whether the savings in CPU time gained from using an augmented adjoint method is worth the development cost. The factors to consider in answering this question are the number of design variables, the grid perturbation requirements, and the frequency of use of the code.

The results show that the benefits of the augmented adjoint method become more pronounced as the number of design variables increases. Therefore, problems with a large number of design variables, such as three-dimensional problems or two-dimensional problems with complex geometries, would receive the most benefit from the use of this method. It is noteworthy, however, that even for a modest eight design variables, the augmented adjoint method computed the gradient at least five times faster than the original method.

The augmented adjoint method is of little benefit if algebraic grid perturbation is used, and considerable hand-coding is required to augment the adjoint equations for any perturbation method ($\partial R/\partial G$ must be computed). The augmented method is therefore a poor investment of time if algebraic perturbation is adequate. This was found to be the case for transonic airfoils, but not for subsonic airfoils. It can be expected that the elasticity perturbation be required for problems involving large shape changes, important viscous effects, complicated topologies, or unstructured meshes.

Looking further into this issue, it may be worth considering the implementation of a *linear* elastic grid perturber (i.e. $n = 1$). This is because it eliminates the need to code the term $\partial r^{(i+1)}/\partial G^{(i)}$—one of the more time-consuming parts of the implementation. This may be reasonable for simple two-dimensional geometries with modest shape changes, because the optimized airfoils computed using $n = 1$ and $n = 2$ were found to be very

similar here.

The final consideration noted above is the frequency of use of the code. Obviously, if a code is only to be used once, then very little development time can be justified, and the method presented here is not worthwhile. Even the adjoint to the flow equations may not be worth the development time in a little-used code. For a code used on a daily basis, the scale tips the other way.

A factor that overwhelms this discussion, however, is that when using the elasticity grid perturbation with the original adjoint method, the optimizer is not able to converge reliably due to gradient inaccuracy. The elasticity grid perturbation method and its differentiation with the augmented adjoint equations should therefore be considered an inseparable unit. In a discrete adjoint optimization code used frequently enough to merit implementing this robust grid perturbation scheme, the augmented adjoint approach is essential.

6.3 Implementation Recommendations

In the implementation of the augmented adjoint method, the bulk of the work done is in hand-coding the required Jacobian matrix calculations, and in verifying the accuracy of each Jacobian matrix against finite difference approximations. Hand-coding derivatives is error-prone and fairly mechanical; there is potential instead to use some automated process, such as the complex step method or algorithmic differentiation.

Application of these automated tools is not a trivial task. Nielsen and Kleb [29] required automatic scripting routines to generate complex-variable code while allowing their real-variable Fortran95 code to coexist. Another option is to use templated C++ code, which would allow the same code to be reused for both purposes. Nielsen and Kleb [29] also noted that future work is required to handle the memory requirements of storing large Jacobian matrices. Maute *et al.* [23] experienced similar memory troubles as well as lengthy run-time for forming $\partial R/\partial G$. While the method presented here was hand-coded at length, both of these problems were avoided.

If an efficient, automatic means for computing the Jacobian matrices is not used in the optimizer, even an inefficient implementation would be very useful for verifying hand-coded derivatives. This would expediate the code development by removing uncertainty about the accuracy of finite differences used as benchmarks.

6.4 Applicability

The augmented adjoint method presented here has been demonstrated only for a specific class of high-fidelity optimization problems. However, one would expect the overall concepts to be applicable to a wide variety of other gradient-based optimization methods, grid topologies, grid perturbation schemes, and high-fidelity analyses.

For example, the optimality conditions in (3.2–3.7) are directly applicable to a fully coupled optimization approach, just as the original adjoint method was solved in coupled form by Gatsis and Zingg [12]. The method can also be extended to any grid topology for which the elasticity grid perturbation is valid, such as unstructured finite volume meshes. The grid sensitivities occurring in a spring analogy perturbation can be treated with an augmented adjoint method, as shown by Mavriplis [24]. Finally, similar concepts could also be used for treating grid sensitivities when using other flow solvers, finite-element models for structures or heat transfer, and high-fidelity multi-disciplinary analyses, such as aero-structural analysis.

References

[1] W. K. Anderson and V. Venkatakrishnan, *Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation*, Computers & Fluids, 28 (1999), pp. 443–480.

[2] P. Z. Bar-Yoseph, S. Mereu, S. Chippada, and V. J. Kalro, *Automatic monitoring of element shape quality in 2-D and 3-D computational mesh dynamics*, Computational Mechanics, 27 (2001), pp. 378–395.

[3] R. L. Barger, M. S. Adams, and R. R. Krishnan, *Automatic computation of Euler marching grids and subsonic grids for wing-fuselage configurations*. NASA TM 4573, Jul 1994.

[4] C. H. Bischof, A. Mauer, W. T. Jones, and J. Samareh, *Experiences with automatic differentiation applied to a volume grid generation code*, Journal of Aircraft, 35 (1998), pp. 569–573.

[5] G. W. Burgreen and O. Baysal, *Three-dimensional aerodynamic shape optimization using discrete sensitivity analysis*, AIAA Journal, 34 (1996), pp. 1761–1770.

[6] G. W. Burgreen, O. Baysal, and M. E. Eleshaky, *Improving the efficiency of aerodynamic shape optimization*, AIAA Journal, 32 (1994), pp. 69–76.

[7] A. Chattopadhyay and N. Pagaldipti, *Multidisciplinary optimization using semi-analytical sensitivity analysis procedure and multilevel decomposition*, Computers & Mathematics with Applications, 29 (1995).

[8] C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[9] C. Degand and C. Farhat, *A three-dimensional torsional spring analogy method for unstructured dynamic meshes*, Computers and Structures, 80 (2002), pp. 305–316.

[10] J. Dongarra, A. Lumsdaine, R. Pozo, and K. Remington, *A sparse matrix library in C++ for high performance architectures*, in Proceedings of the Second Object Oriented Numerics Conference, 1992, pp. 214–218.

[11] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne, *Torsional springs for two-dimensional dynamic unstructured fluid meshes*, Computer Methods in Applied Mechanics and Engineering, 163 (1998), pp. 231–245.

[12] J. Gatsis and D. W. Zingg, *A fully-coupled newton-krylov algorithm for aerodynamic optimization*, in 16th AIAA Computational Fluid Dynamics Conference, Orlando, FA, June 2003, AIAA Paper 2003–3956.

[13] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[14] M. Gunzburger, *Introduction to the mathematical aspects of flow control and optimisation*, in Inverse Design and Optimisation Methods, Brussels, Belgium, Apr. 1997, von Karman Institute of Fluid Dynamics.

[15] A. Jameson, *Aerodynamic design via control theory*, Journal of Scientific Computing, 3 (1988), pp. 233–260.

[16] W. T. Jones and J. Samareh-Abolhassani, *A grid generation system for miltidisciplinary design optimization*, in Proceedings of the AIAA 12th Computational Fluid Dynamics Conference, Washington, DC, 1995, AIAA Paper 1995–1689, pp. 474–482.

[17] H. Kim and K. Nakahashi, *Flap-deflection optimization for transonic cruise performance improvement of supersonic transport wing*, Journal of Aircraft, 38 (2001), pp. 709–717.

[18] H. Kim, S. Obayashi, and K. Nakahashi, *Aerodynamic optimization of supersonic transport wing using unstructured adjoint method*, AIAA Journal, 39 (2001), pp. 1011–1020.

[19] V. M. Korivi, P. A. Newman, and A. C. Taylor, III, *Aerodynamic optimization using sensitivity derivatives from a three-dimensional supersonic Euler code*, Journal of Aircraft, 35 (1998), pp. 405–411.

[20] A. Le Moigne and N. Qin, *Variable-fidelity aerodynamic optimization for turbulent flows using a discrete adjoint formulation*, AIAA Journal, 42 (2004), pp. 1281–1292.

[21] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther, *A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design*, Optimization and Engineering, 6 (2005), pp. 33–62.

[22] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso, *The complex-step derivative approximation*, ACM Transactions on Mathematical Software, 29 (2003), pp. 245–262.

[23] K. Maute, M. Nikbay, and C. Farhat, *Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method*, International Journal for Numerical Methods in Engineering, 56 (2003), pp. 911–933.

[24] D. J. Mavriplis, *Multigrid solution of the discrete adjoint for optimization problems on unstructured meshes*, AIAA Journal, 44 (2006).

[25] M. Nemec, *Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach*, PhD thesis, University of Toronto, 2003.

[26] M. Nemec and D. Zingg, *Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations*, AIAA Journal, 40 (2002), pp. 856–859.

[27] M. Nemec, D. Zingg, and T. Pulliam, *Multipoint and multi-objective aerodynamic shape optimization*, AIAA Journal, 42 (2004), pp. 1057–1065.

[28] E. J. Nielsen and W. K. Anderson, *Recent improvements in areodynamic design optimization on unstructured meshes*, AIAA Journal, 40 (2002), pp. 1155–1163.

[29] E. J. Nielsen and W. L. Kleb, *Efficient construction of discrete adjoint operators on unstructured grids using complex variables*, AIAA Journal, 44 (2006), pp. 827–836.

[30] E. J. Nielsen and M. A. Park, *Using an adjoint approach to eliminate mesh sensitivities in computational design*, AIAA Journal, 44 (2006), pp. 948–953.

[31] N. Pagaldipti and A. Chattopadhyay, *A discrete semianalytical procedure for aerodynamic sensitivity analysis including grid sensitivity*, Computers & Mathematics with Applications, 32 (1996), pp. 61–71.

[32] O. Pironneau, *On optimum design in fluid mechanics*, Journal of Fluid Mechanics, 64 (1974), pp. 97–110.

[33] A. Pueyo, *An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations*, PhD thesis, University of Toronto, 1998.

[34] Y. Saad, *SPARSKIT: a basic kit for sparse matrix computations*. http://www-users.cs.umn.edu/~saad/software/SPARSKIT/paper.ps, June 1994.

[35] I. Sadrehaghighi, R. E. Smith, and S. N. Twari, *Grid sensitivity and aerodynamic optimization of generic airfoils*, Journal of Aircraft, 32 (1995), pp. 1234–1239.

[36] J. A. Samareh, *Novel multidisciplinary shape parameterization approach*, Journal of Aircraft, 38 (2001), pp. 1015–1024.

[37] ——, *Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization*, AIAA Journal, 39 (2001), pp. 877–884.

[38] ——, *Application of quaternions for mesh deformation*. NASA TM–2002–211646, Apr. 2002.

[39] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM Review, 40 (1998), pp. 110–112.

[40] K. Stein, T. Tezduyar, and R. Benney, *Mesh moving techniques for fluid-structure interactions with large displacements*, Journal of Applied Mechanics, 70 (2003), pp. 58–63.

[41] T. E. Tezduyar, M. Behr, S. Mittal, and A. A. Johnson, *Computation of unsteady incompressible flows with the stabilized finite element methods: Space-time formulations, iterative strategies and massively parallel implementations*, in New methods in transient analysis: presented at the Winter Annual Meeting of the American Society of Mechanical Engineers, Anaheim, CA, Nov. 1992, ASME.

[42] A. Truong, *Development of grid movement algorithms suitable for aerodynamic optimization*, Master's thesis, University of Toronto, 2005.

Appendix A

Chain Rule Adjoint Derivation

In evaluating the objective function, $\mathcal{F}$, at a given set of values for the design variables, X, the following steps are required.

First, each of the increments of the grid perturbation are converged, as represented by setting the residuals to zero:

$$0 = r^{(i)}(G^{(i)}, G^{(i-1)}, A^{(i)}(X)), \quad i \in \{1, 2, \ldots, n\} \tag{A.1}$$

This implicitly defines each incremental grid as a function of the previous one. For $n = 2$, this is

$$G^{(1)} = G^{(1)}(A^{(1)}(X)) \tag{A.2}$$

$$G^{(2)} = G^{(2)}(G^{(1)}, A^{(2)}(X)) \tag{A.3}$$

Next, the flow solver is converged, as represented by setting the residual to zero:

$$0 = R(Q, G, X) \tag{A.4}$$

This implicitly defines the flow solution as a function of the grid and the design variables:

$$Q = Q(G, X) \tag{A.5}$$

Finally, the objective function is evaluated.

$$\mathcal{J} = \mathcal{J}(Q, G, X) \tag{A.6}$$

Or, when including all of the functional dependencies, this is

$$\mathcal{F} = \mathcal{F}\Big(Q\Big(G^{(2)}\Big(G^{(1)}(A^{(1)}(X)), A^{(2)}(X)\Big), X\Big),\ G^{(2)}\Big(G^{(1)}(A^{(1)}(X)), A^{(2)}(X)\Big),\ X\Big) \tag{A.7}$$

This could simply be written as $\mathcal{F}(X)$, but it is useful to retain the intermediate dependencies when forming the gradient of $\mathcal{F}$ with respect to X. Using the chain rule and considering $n = 2$, this gradient is

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial X} = &\frac{\partial \mathcal{F}}{\partial Q}\left(\frac{\partial Q}{\partial G^{(2)}}\left(\frac{\partial G^{(2)}}{\partial G^{(1)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X} + \frac{\partial G^{(2)}}{\partial A^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right) + \left.\frac{\partial Q}{\partial X}\right|_{G^{(2)}}\right) \\ &+ \left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\left(\frac{\partial G^{(2)}}{\partial G^{(1)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X} + \frac{\partial G^{(2)}}{\partial A^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right) + \left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}}\end{aligned} \tag{A.8}$$

This makes use of the following notation for partial derivatives. Given the function $f(x)$, where both f and x are vectors, the Jacobian matrix representing the derivatives of f with respect to x is $\frac{\partial f}{\partial x}$. Now suppose that f could also be written as $f(g(x),\ x)$, where g is also a vector. The derivatives of f with respect to x are then

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \left.\frac{\partial f}{\partial x}\right|_{g} \tag{A.9}$$

Here, the notation $\left.\frac{\partial f}{\partial x}\right|_{g}$ is used to represent the partial derivatives of f with respect to x, where variations in g are excluded. Specifically,

$$\left.\frac{\partial f}{\partial x}\right|_{g} = \left[\left.\frac{\partial f}{\partial x_1}\right|_{g} \quad \left.\frac{\partial f}{\partial x_2}\right|_{g} \quad \cdots \quad \left.\frac{\partial f}{\partial x_i}\right|_{g} \quad \cdots \quad \left.\frac{\partial f}{\partial x_{n-1}}\right|_{g} \quad \left.\frac{\partial f}{\partial x_n}\right|_{g}\right] \tag{A.10}$$

$$\left.\frac{\partial f}{\partial x_i}\right|_{g} = \lim_{\epsilon \to 0} \frac{f(g(x), x + \hat{u}_i\epsilon) - f(g(x), x)}{\epsilon} \tag{A.11}$$

where $\hat{u}_i$ is a unit vector in the i^{th} coordinate direction.

In (A.8), Q, $G^{(2)}$, and $G^{(1)}$ are implicitly defined functions, so their derivatives are difficult to compute. Consider first the derivatives of Q. Differentiating (A.4) with respect to G gives

$$0 = \frac{\partial R}{\partial Q}\frac{\partial Q}{\partial G} + \left.\frac{\partial R}{\partial G}\right|_{Q} \quad \Rightarrow \quad \frac{\partial Q}{\partial G} = -\left[\frac{\partial R}{\partial Q}\right]^{-1}\left.\frac{\partial R}{\partial G}\right|_{Q} \tag{A.12}$$

Differentiating (A.4) with respect to X, while holding G constant, gives

$$0 = \frac{\partial R}{\partial Q}\left.\frac{\partial Q}{\partial X}\right|_{G} + \left.\frac{\partial R}{\partial X}\right|_{Q,G} \quad \Rightarrow \quad \left.\frac{\partial Q}{\partial X}\right|_{G} = -\left[\frac{\partial R}{\partial Q}\right]^{-1}\left.\frac{\partial R}{\partial X}\right|_{Q,G} \tag{A.13}$$

These expressions for the derivatives of Q can now be substituted into (A.8) as follows:

$$\frac{\partial \mathcal{F}}{\partial X} = -\frac{\partial \mathcal{F}}{\partial Q}\left[\frac{\partial R}{\partial Q}\right]^{-1}\left(\left.\frac{\partial R}{\partial G^{(2)}}\right|_{Q}\left(\frac{\partial G^{(2)}}{\partial G^{(1)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X}+\frac{\partial G^{(2)}}{\partial A^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right)+\left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}}\right)$$
$$+\left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\left(\frac{\partial G^{(2)}}{\partial G^{(1)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X}+\frac{\partial G^{(2)}}{\partial A^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right)+\left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}} \tag{A.14}$$

Now let the adjoint vector ψ be defined by the linear system,

$$\psi^{\mathrm{T}} = -\frac{\partial \mathcal{F}}{\partial Q}\left[\frac{\partial R}{\partial Q}\right]^{-1} \quad \Rightarrow \quad \left[\frac{\partial R}{\partial Q}\right]^{\mathrm{T}}\Psi = -\left[\frac{\partial \mathcal{F}}{\partial Q}\right]^{\mathrm{T}} \tag{A.15}$$

Incorporating this and rearranging gives

$$\frac{\partial \mathcal{F}}{\partial X} = \left(\psi^{\mathrm{T}}\left.\frac{\partial R}{\partial G^{(2)}}\right|_{Q}+\left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\right)\left(\frac{\partial G^{(2)}}{\partial G^{(1)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X}+\frac{\partial G^{(2)}}{\partial A^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right)$$
$$+\psi^{\mathrm{T}}\left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}}+\left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}} \tag{A.16}$$

Differentiating (A.1) with respect to $A^{(2)}$, with $i = 2$, gives

$$0 = \frac{\partial r^{(2)}}{\partial G^{(2)}}\frac{\partial G^{(2)}}{\partial A^{(2)}}+\left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}} \quad \Rightarrow \quad \frac{\partial G^{(2)}}{\partial A^{(2)}} = -\left[\frac{\partial r^{(2)}}{\partial G^{(2)}}\right]^{-1}\left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}} \tag{A.17}$$

Differentiating (A.1) with respect to $G^{(1)}$, with $i = 2$, gives

$$0 = \frac{\partial r^{(2)}}{\partial G^{(2)}}\frac{\partial G^{(2)}}{\partial G^{(1)}}+\left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}} \quad \Rightarrow \quad \frac{\partial G^{(2)}}{\partial G^{(1)}} = -\left[\frac{\partial r^{(2)}}{\partial G^{(2)}}\right]^{-1}\left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}} \tag{A.18}$$

These expressions for the derivatives of $r^{(2)}$ can now be substituted into (A.16) as follows:

$$\frac{\partial \mathcal{F}}{\partial X} = -\left(\psi^{\mathrm{T}}\left.\frac{\partial R}{\partial G^{(2)}}\right|_{Q}+\left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\right)\left[\frac{\partial r^{(2)}}{\partial G^{(2)}}\right]^{-1}\left(\left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}}\frac{\partial G^{(1)}}{\partial A^{(1)}}\frac{\partial A^{(1)}}{\partial X}+\left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}}\frac{\partial A^{(2)}}{\partial X}\right)$$
$$+\psi^{\mathrm{T}}\left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}}+\left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}} \tag{A.19}$$

Now let the adjoint vector $\lambda^{(2)}$ be defined by the linear system,

$$\lambda^{(2)\mathrm{T}} = -\left(\psi^{\mathrm{T}}\left.\frac{\partial R}{\partial G^{(2)}}\right|_{Q}+\left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\right)\left[\frac{\partial r^{(2)}}{\partial G^{(2)}}\right]^{-1}$$

$$\left[\frac{\partial r^{(2)}}{\partial G^{(2)}}\right]^{\mathrm{T}} \lambda^{(2)} = -\left[\psi^{\mathrm{T}} \left.\frac{\partial R}{\partial G^{(2)}}\right|_{Q} + \left.\frac{\partial \mathcal{F}}{\partial G^{(2)}}\right|_{Q}\right]^{\mathrm{T}} \tag{A.20}$$

Substituting this into (A.19) gives

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial X} = \lambda^{(2)\mathrm{T}} \left(\left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}} \frac{\partial G^{(1)}}{\partial A^{(1)}} \frac{\partial A^{(1)}}{\partial X} + \left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}} \frac{\partial A^{(2)}}{\partial X} \right) \\ + \psi^{\mathrm{T}} \left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}} + \left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}}\end{aligned} \tag{A.21}$$

Differentiating (A.1) with respect to $A^{(1)}$, with $i = 1$, gives

$$0 = \frac{\partial r^{(1)}}{\partial G^{(1)}} \frac{\partial G^{(1)}}{\partial A^{(1)}} + \left.\frac{\partial r^{(1)}}{\partial A^{(1)}}\right|_{G^{(1)}} \quad \Rightarrow \quad \frac{\partial G^{(1)}}{\partial A^{(1)}} = -\left[\frac{\partial r^{(1)}}{\partial G^{(1)}}\right]^{-1} \left.\frac{\partial r^{(1)}}{\partial A^{(1)}}\right|_{G^{(1)}} \tag{A.22}$$

Substituting this into (A.21) gives

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial X} = \lambda^{(2)\mathrm{T}} \left(- \left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}} \left[\frac{\partial r^{(1)}}{\partial G^{(1)}}\right]^{-1} \left.\frac{\partial r^{(1)}}{\partial A^{(1)}}\right|_{G^{(1)}} \frac{\partial A^{(1)}}{\partial X} + \left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}} \frac{\partial A^{(2)}}{\partial X} \right) \\ + \psi^{\mathrm{T}} \left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}} + \left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}}\end{aligned} \tag{A.23}$$

Now let the adjoint vector $\lambda^{(1)}$ be defined by the linear system,

$$\lambda^{(1)\mathrm{T}} = -\lambda^{(2)\mathrm{T}} \left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}} \left[\frac{\partial r^{(1)}}{\partial G^{(1)}}\right]^{-1} \quad \Rightarrow \quad \left[\frac{\partial r^{(1)}}{\partial G^{(1)}}\right]^{\mathrm{T}} \lambda^{(1)} = -\left[\lambda^{(2)\mathrm{T}} \left.\frac{\partial r^{(2)}}{\partial G^{(1)}}\right|_{G^{(2)}}\right]^{\mathrm{T}} \tag{A.24}$$

Substituting this into (A.23) gives

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial X} = \lambda^{(1)\mathrm{T}} \left.\frac{\partial r^{(1)}}{\partial A^{(1)}}\right|_{G^{(1)}} \frac{\partial A^{(1)}}{\partial X} + \lambda^{(2)\mathrm{T}} \left.\frac{\partial r^{(2)}}{\partial A^{(2)}}\right|_{G^{(2)}} \frac{\partial A^{(2)}}{\partial X} \\ + \psi^{\mathrm{T}} \left.\frac{\partial R}{\partial X}\right|_{Q,G^{(2)}} + \left.\frac{\partial \mathcal{F}}{\partial X}\right|_{Q,G^{(2)}}\end{aligned} \tag{A.25}$$

This is the same system of equations as was derived in Chapter 3. The evaluation of $\mathcal{F}$ in (A.7) requires that a grid perturbation and flow solve be completed, and so is equivalent to (3.2, 3.3). The flow adjoint system, (A.15), is identical to (3.3). For the grid adjoint systems, (A.20, A.24) match up with (3.5, 3.6), respectively, where $n = 2$. Then the objective function gradient is given equivalently by (A.25) or by(3.7).

Appendix B

Forming $\partial R/\partial G$

The sensitivity of the flow residual, R, with respect to the grid node locations, G, involves several terms which can be considered independently. The Jacobian matrix is given by

$$\frac{\partial R}{\partial G} = \frac{\partial\left(\delta_\xi \widehat{E}\right)}{\partial G} + \frac{\partial\left(\delta_\eta \widehat{F}\right)}{\partial G} - \frac{1}{\mathcal{R}e}\frac{\partial\left(\delta_\eta \widehat{S}\right)}{\partial G} - \frac{\partial\left(\nabla_\xi \widehat{E}_d\right)}{\partial G} - \frac{\partial\left(\nabla_\eta \widehat{F}_d\right)}{\partial G} \tag{B.1}$$

Rather than storing the entire Jacobian matrix $\partial R/\partial G$, only its product with the row vector ψ^{T} is stored (see Section 4.3.2). This product is formed by computing a single row of $\partial R/\partial G$ at a time,[1] and immediately multiplying it by the appropriate element of ψ^T, and adding the result to the product.

The dependency of the flow residual on the grid node locations is almost exclusively via the metrics of the coordinate transformation between physical and computational space. The metrics are therefore outlined next, followed by a discussion of each term in R and $\partial R/\partial G$.

B.1 Metrics

The metrics of the coordinate transformation can be summarized by

$$\xi_x = Jy_\eta \qquad \xi_y = -Jx_\eta \tag{B.2}$$

$$\eta_x = -Jy_\xi \qquad \eta_y = Jx_\xi \tag{B.3}$$

[1] A row of $\partial R/\partial G$ represents the derivative of one element of the residual with respect to the entire grid. Considering that when R is calculated, one element is computed—as a function of the entire grid—at a time, this order of calculation of the derivative follows naturally.

$$J = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \tag{B.4}$$

$$U = \xi_x u + \xi_y v = J\left(y_\eta u - x_\eta v\right) \tag{B.5}$$

$$V = \eta_x u + \eta_y v = J\left(-y_\xi u + x_\xi v\right) \tag{B.6}$$

These quantities are computed by first forming x_ξ, x_η, y_ξ, and y_η using centred second-order finite differences, then using them to compute the other quantities. For example,

$$\left(x_\xi\right)_{j,k} \approx \delta_\xi x_{j,k} = \frac{x_{j+1,k} - x_{j-1,k}}{2} \tag{B.7}$$

The stencil is modified to a one-sided second-order finite difference at the boundaries. For example,

$$\left(x_\xi\right)_{j_{\max},k} \approx \delta_\xi x_{j_{\max},k} = \frac{3x_{j_{\max},k} - 4x_{j_{\max}-1,k} + x_{j_{\max}-2,k}}{2} \tag{B.8}$$

B.2 Inviscid Fluxes

The inviscid flux in ξ is given by

$$\delta_\xi \widehat{E} = \delta_\xi \left(J^{-1} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) \end{bmatrix} \right) \tag{B.9}$$

This can be rewritten as

$$\delta_\xi \widehat{E} = \delta_\xi \left(\begin{bmatrix} \rho(y_\eta u - x_\eta v) \\ \rho u(y_\eta u - x_\eta v) + y_\eta p \\ \rho v(y_\eta u - x_\eta v) - x_\eta p \\ (y_\eta u - x_\eta v)(e+p) \end{bmatrix} \right) \tag{B.10}$$

The term in [ ] represents a block column vector of which it is a typical block. It can be separated as follows

$$\delta_\xi \widehat{E} = \delta_\xi \left(\begin{bmatrix} -\rho v & \rho u \\ -\rho u v & \rho u^2 + p \\ -\rho v^2 - p & \rho u v \\ -v(e+p) & u(e+p) \end{bmatrix}_D \begin{bmatrix} x_\eta \\ y_\eta \end{bmatrix} \right) \tag{B.11}$$

The subscript D on the 4×2 block indicates that it represents a block diagonal matrix (overall dimensions: $4j_{\max}k_{\max} \times 2j_{\max}k_{\max}$) of which it is a typical block on the diagonal. For example,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}_D = \begin{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{1,1} & & & & & \\ & \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{2,1} & & & & \\ & & \ddots & & & \\ & & & \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{j_{\max}-1,k_{\max}} & & \\ & & & & \begin{bmatrix} a & b \\ c & d \end{bmatrix}_{j_{\max},k_{\max}} \end{bmatrix} \tag{B.12}$$

The 2×1 block represents a block column vector (overall dimensions: $2j_{\max}k_{\max} \times 1$), and can also be written as $\delta_\eta G$. The differentiation of this with respect to G (and pre-multiplication by ψ^{T}) then yields

$$\psi^{\mathrm{T}} \frac{\partial \left(\delta_\xi \widehat{E}\right)}{\partial G} = \psi^{\mathrm{T}} \delta_\xi \left(\begin{bmatrix} -\rho v & \rho u \\ -\rho uv & \rho u^2 + p \\ -\rho v^2 - p & \rho uv \\ -v(e+p) & u(e+p) \end{bmatrix}_D (\delta_\eta I) \right) \tag{B.13}$$

where I is the identity matrix (with $2j_{\max}k_{\max}$ rows). The derivative can be expressed in this form only because when the discrete operator $\delta_\eta()$ acts on some vector, the result is a linear combination of the elements of that vector. In a one-dimensional example, for the vector a, one could write $\delta a = Ma$, where M is a matrix that represents the finite difference stencil. For example, in the case of second-order finite differences in one dimension, M is given by

$$M = \begin{bmatrix} -\frac{3}{2} & 2 & -\frac{1}{2} & & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & & & \\ & -\frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & & \frac{1}{2} & -2 & \frac{3}{2} \end{bmatrix} \tag{B.14}$$

$$\begin{bmatrix}
\ddots & & \vdots & & & & \vdots & & & & \vdots & & & \\
 & 0 & \frac{1}{2} & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 & & \\
\cdots & -\frac{1}{2} & 0 & \frac{1}{2} & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & \\
 & 0 & -\frac{1}{2} & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 & & \\
 & & \vdots & & \ddots & & \vdots & & & & \vdots & & & \\
 & 0 & 0 & 0 & & 0 & \frac{1}{2} & 0 & & 0 & 0 & 0 & & \\
\cdots & 0 & 0 & 0 & \cdots & -\frac{1}{2} & 0 & \frac{1}{2} & \cdots & 0 & 0 & 0 & \cdots & \\
 & 0 & 0 & 0 & & 0 & -\frac{1}{2} & 0 & & 0 & 0 & 0 & & \\
 & & \vdots & & & & \vdots & & \ddots & & \vdots & & & \\
 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & \frac{1}{2} & 0 & & \\
\cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & -\frac{1}{2} & 0 & \frac{1}{2} & \cdots & \\
 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & -\frac{1}{2} & 0 & & \\
 & & \vdots & & & & \vdots & & & & \vdots & & \ddots &
\end{bmatrix}$$

Grid indices
$j-1,\ k-1$
$j\quad ,\ k-1$
$j+1,\ k-1$
$\vdots$
$j-1,\ k$
$j\quad ,\ k$
$j+1,\ k$
$\vdots$
$j-1,\ k+1$
$j\quad ,\ k+1$
$j+1,\ k+1$

Figure B.1: The interior scheme of $\delta_\xi I$

Using this, one finds that

$$\frac{\partial}{\partial a}(\delta a) = \frac{\partial}{\partial a}(Ma) = M\frac{\partial a}{\partial a} = MI = \delta I = M \tag{B.15}$$

So, the term δI is to be interpreted as a matrix representation of the finite difference stencil used to compute $\delta(\cdot)$.

For the two-dimensional case, the interior schemes for $\delta_\xi I$ (which appears later) and $\delta_\eta I$ are given in Figures B.1 and B.2, where each entry represents a 2×2 diagonal block (as each block of G has 2 elements). The boundary scheme is similar to that in (B.14).

In a general interior row of the Jacobian matrix, $\partial R_{j,k}/\partial G$, the only nonzero terms are the following four:

$$\frac{\partial\left(\delta_\xi \widehat{E}_{j,k}\right)}{\partial G_{j-1,k-1}} = \begin{pmatrix}\frac{-1}{2}\end{pmatrix}\begin{bmatrix} -\rho v & \rho u \\ -\rho uv & \rho u^2 + p \\ -\rho v^2 - p & \rho uv \\ -v(e+p) & u(e+p) \end{bmatrix}_{j-1,k} \begin{pmatrix}\frac{-1}{2}\end{pmatrix} \tag{B.16}$$

$$
\begin{bmatrix}
\ddots & & \vdots & & & & \vdots & & & & \vdots & & & \\
 & 0 & 0 & 0 & & \frac{1}{2} & 0 & 0 & & 0 & 0 & 0 & & \\
\cdots & 0 & 0 & 0 & \cdots & 0 & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 & \cdots & \\
 & 0 & 0 & 0 & & 0 & 0 & \frac{1}{2} & & 0 & 0 & 0 & & \\
 & & \vdots & & \ddots & & \vdots & & & & \vdots & & & \\
 & -\frac{1}{2} & 0 & 0 & & 0 & 0 & 0 & & \frac{1}{2} & 0 & 0 & & \\
\cdots & 0 & -\frac{1}{2} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \frac{1}{2} & 0 & \cdots & \\
 & 0 & 0 & -\frac{1}{2} & & 0 & 0 & 0 & & 0 & 0 & \frac{1}{2} & & \\
 & & \vdots & & & & \vdots & & \ddots & & \vdots & & & \\
 & 0 & 0 & 0 & & -\frac{1}{2} & 0 & 0 & & 0 & 0 & 0 & & \\
\cdots & 0 & 0 & 0 & \cdots & 0 & -\frac{1}{2} & 0 & \cdots & 0 & 0 & 0 & \cdots & \\
 & 0 & 0 & 0 & & 0 & 0 & -\frac{1}{2} & & 0 & 0 & 0 & & \\
 & & \vdots & & & & \vdots & & & & \vdots & & \ddots &
\end{bmatrix}
\qquad
\begin{array}{|c}
\text{Grid indices} \\ \hline
j-1,\ k-1 \\
j\quad ,\ k-1 \\
j+1,\ k-1 \\
\vdots \\
j-1,\ k \\
j\quad ,\ k \\
j+1,\ k \\
\vdots \\
j-1,\ k+1 \\
j\quad ,\ k+1 \\
j+1,\ k+1 \\
\end{array}
$$

Figure B.2: The interior scheme of $\delta_\eta I$

$$\frac{\partial\left(\delta_\xi \widehat{E}_{j,k}\right)}{\partial G_{j-1,k+1}} = \left(\frac{-1}{2}\right) \begin{bmatrix} -\rho v & \rho u \\ -\rho uv & \rho u^2 + p \\ -\rho v^2 - p & \rho uv \\ -v(e+p) & u(e+p) \end{bmatrix}_{j-1,k} \left(\frac{1}{2}\right) \tag{B.17}$$

$$\frac{\partial\left(\delta_\xi \widehat{E}_{j,k}\right)}{\partial G_{j+1,k-1}} = \left(\frac{1}{2}\right) \begin{bmatrix} -\rho v & \rho u \\ -\rho uv & \rho u^2 + p \\ -\rho v^2 - p & \rho uv \\ -v(e+p) & u(e+p) \end{bmatrix}_{j+1,k} \left(\frac{-1}{2}\right) \tag{B.18}$$

$$\frac{\partial\left(\delta_\xi \widehat{E}_{j,k}\right)}{\partial G_{j+1,k+1}} = \left(\frac{1}{2}\right) \begin{bmatrix} -\rho v & \rho u \\ -\rho uv & \rho u^2 + p \\ -\rho v^2 - p & \rho uv \\ -v(e+p) & u(e+p) \end{bmatrix}_{j+1,k} \left(\frac{1}{2}\right) \tag{B.19}$$

The code that evaluates (B.13) takes advantage of the high degree of similarity of the above four equations. For a given point in the $\delta\xi()$ stencil in (B.13), all but $(\delta_\eta I)$ is evaluated, and the result is sent to a separate function that handles the multiplication by the appropriate nonzero entries in $(\delta_\eta I)$. The benefits of this implementation are small

in this case, but the same method greatly simplifies the code for the more complicated terms in $\partial R/\partial G$. It also provides simpler treatment of boundaries in the stencils used for the metrics, and can be more readily expanded to higher order stencils for the metrics.

Similarly, the η-direction inviscid flux, when differentiated with respect to the grid node locations, is

$$\frac{\partial\left(\delta_\eta \widehat{F}\right)}{\partial G} = \delta_\eta \left(\begin{bmatrix} \rho v & -\rho u \\ \rho uv & -\rho u^2 - p \\ \rho v^2 + p & -\rho uv \\ v(e+p) & -u(e+p) \end{bmatrix}_D (\delta_\xi I) \right) \tag{B.20}$$

B.3 Artificial Dissipation

The ξ-direction artificial dissipation is expressed as

$$-\nabla_\xi(\widehat{E}_d)_{j,k} = -\nabla_\xi \left(d^{(2)}_{j+\frac{1}{2},k} \Delta_\xi Q_{j,k} - d^{(4)}_{j+\frac{1}{2},k} \Delta_\xi \nabla_\xi \Delta_\xi Q_{j,k} \right) \tag{B.21}$$

where ∇_ξ and Δ_ξ are backward and forward difference operators in the ξ-direction:

$$\nabla_\xi(\cdot)_{j,k} = (\cdot)_{j,k} - (\cdot)_{j-1,k} \qquad \Delta_\xi(\cdot)_{j,k} = (\cdot)_{j+1,k} - (\cdot)_{j,k} \tag{B.22}$$

The combinations of these operators acting on Q are adjusted near the boundaries.

The other coefficients are given by

$$d^{(2)}_{j+\frac{1}{2},k} = 2(\epsilon\sigma J^{-1})_{j+\frac{1}{2},k} \tag{B.23}$$

$$d^{(4)}_{j+\frac{1}{2},k} = \max\left[0,\ 2\kappa_4(\sigma J^{-1})_{j+\frac{1}{2},k} - d^{(2)}_{j+\frac{1}{2},k}\right] \tag{B.24}$$

$$\sigma_{j,k} = \left(|U| + a\sqrt{\xi_x^2 + \xi_y^2}\right)_{j,k} \tag{B.25}$$

$$\epsilon_{j,k} = \kappa_2 \left[\frac{1}{2}\Upsilon^*_{j,k} + \frac{1}{4}\left(\Upsilon^*_{j-1,k} + \Upsilon^*_{j+1,k}\right)\right] \tag{B.26}$$

$$\Upsilon^*_{j,k} = \max\left(\Upsilon_{j-1,k},\ \Upsilon_{j,k},\ \Upsilon_{j+1,k}\right) \tag{B.27}$$

$$\Upsilon_{j,k} = \frac{|p_{j-1,k} - 2p_{j,k} + p_{j+1,k}|}{|p_{j-1,k} + 2p_{j,k} + p_{j+1,k}|} \tag{B.28}$$

where κ_2 and κ_4 are user-defined constants. Mean properties are used at half-nodes, and can be expressed using an operator defined as:

$$(\cdot)_{j+\frac{1}{2},k} = \varpi_\xi(\cdot)_{j,k} = \frac{(\cdot)_{j,k} + (\cdot)_{j+1,k}}{2} \tag{B.29}$$

Notice that both ϵ and Q are independent of the grid node locations, G. As a result, the only dependency on G is through the spectral radius and metric Jacobian, σJ^{-1}. With this in mind, the derivative of the artificial dissipation with respect to G can be formed as follows. We begin by expressing the artificial dissipation for the interior nodes in matrix form:

$$-\nabla_\xi \widehat{E}_d = -\nabla_\xi \left[\left(\varpi_\xi d^{(2)} \right)_D \Delta_\xi Q - \left(\varpi_\xi d^{(4)} \right)_D \Delta_\xi \nabla_\xi \Delta_\xi Q \right] \tag{B.30}$$

This can be rearranged as

$$-\nabla_\xi \widehat{E}_d = -\nabla_\xi \left[\left(\Delta_\xi Q \right)_D \varpi_\xi d^{(2)} - \left(\Delta_\xi \nabla_\xi \Delta_\xi Q \right)_D \varpi_\xi d^{(4)} \right] \tag{B.31}$$

whose differentiation is more straightforward.

$$\frac{\partial(-\nabla_\xi \widehat{E}_d)}{\partial G} = -\nabla_\xi \left[\left(\Delta_\xi Q \right)_D \varpi_\xi \frac{\partial d^{(2)}}{\partial G} - \left(\Delta_\xi \nabla_\xi \Delta_\xi Q \right)_D \varpi_\xi \frac{\partial d^{(4)}}{\partial G} \right] \tag{B.32}$$

Here,

$$\frac{\partial d^{(2)}}{\partial G} = 2\epsilon_D \frac{\partial(\sigma J^{-1})}{\partial G} \tag{B.33}$$

and

$$\frac{\partial d^{(4)}}{\partial G} = \begin{cases} 0 & , \ \varpi_\xi \left(2\kappa_4 \sigma J^{-1} - d^{(2)} \right) \leq 0 \\ 2\left(\kappa_4 - \epsilon \right)_D \frac{\partial(\sigma J^{-1})}{\partial G} & , \ \varpi_\xi \left(2\kappa_4 \sigma J^{-1} - d^{(2)} \right) > 0 \end{cases} \tag{B.34}$$

Finally, consider σJ^{-1}.

$$\sigma J^{-1} = \left(|\xi_x u + \xi_y v| + a\sqrt{\xi_x^2 + \xi_y^2} \right) J^{-1} \tag{B.35}$$

$$= \left(|J y_\eta u - J x_\eta v| + aJ\sqrt{x_\eta^2 + y_\eta^2} \right) J^{-1} \tag{B.36}$$

Now J is always positive, so

$$\sigma J^{-1} = |y_\eta u - x_\eta v| + a\sqrt{x_\eta^2 + y_\eta^2} \tag{B.37}$$

Differentiation yields

$$\frac{\partial(\sigma J^{-1})}{\partial G_1} = \left[\text{sign}\left(y_\eta u - x_\eta v \right)(-v) + \frac{a x_\eta}{\sqrt{x_\eta^2 + y_\eta^2}} \right]_D \delta_\eta I \tag{B.38}$$

$$\frac{\partial(\sigma J^{-1})}{\partial G_2} = \left[\text{sign}\left(y_\eta u - x_\eta v \right) u + \frac{a y_\eta}{\sqrt{x_\eta^2 + y_\eta^2}} \right]_D \delta_\eta I \tag{B.39}$$

This applies to the interior scheme. The artificial dissipation term is not required at the boundary nodes themselves, but is required at the next-to-boundary nodes. The

leading backward difference operator in (B.32) is well defined there. However, near the boundaries, the stencils for $\Delta_\xi Q$ and $\Delta_\xi \nabla_\xi \Delta_\xi Q$ are altered just as in (B.21). For the calculation of $\partial \widehat{E}_d/\partial G$, these alterations change the constants by which $\partial d^{(2)}/\partial G$ and $\partial d^{(4)}/\partial G$ are multiplied by, but have no other effect.

The artificial dissipation in the η-direction is similar. However, (B.27) is omitted, and the spectral radius is defined as

$$\sigma_{j,k} = \left(|V| + a\sqrt{\eta_x^2 + \eta_y^2}\right)_{j,k} \tag{B.40}$$

As a result, in the η-direction, we have

$$\frac{\partial(\sigma J^{-1})}{\partial G_1} = \left[\text{sign}\left(x_\xi v - y_\xi u\right) v + \frac{a x_\xi}{\sqrt{x_\xi^2 + y_\xi^2}}\right]_D \delta_\xi I \tag{B.41}$$

$$\frac{\partial(\sigma J^{-1})}{\partial G_1} = \left[\text{sign}\left(x_\xi v - y_\xi u\right) (-u) + \frac{a y_\xi}{\sqrt{x_\xi^2 + y_\xi^2}}\right]_D \delta_\xi I \tag{B.42}$$

There is one wrinkle in this derivation. That is that the absolute value function used in the calculation of σ has an undefined derivative at $U = 0$ (or $V = 0$). For a viscous analysis, the no-slip boundary condition at the airfoil surface sets $U = V = 0$, and for an inviscid analysis, the condition $V_n = 0$ at the same boundary is equivalent to setting $V = 0$. The normal velocity, V_n is given in (B.106).

In a numerical calculation, one would expect that U and V not converge to exactly 0, but rather that they be very small positive or negative numbers. As a result, one can expect the derivative of σJ^{-1} to be well-defined. However, the exact values that U and V converge to are unpredictable, as are the signs of these values. This can be expected to hold even for flow solutions to very similar problems. That is, even towards the end of an optimization, when one flow solve very much resembles the next, the signs of U and V at the airfoil are random. This randomness would cascade into the derivative of the artificial dissipation with respect to the grid node locations, and eventually into the overall objective function gradient, producing noise in the design space. So, in the context of gradient-based optimization, calculating the derivative in the most strictly correct sense may cause convergence problems in the optimizer. Instead, the derivatives of $|U|$ and $|V|$ (only $|V|$ for an inviscid analysis) with respect to G are set to 0 at the boundary nodes where U and V are set to 0.[2] This approach is implemented in the

[2]When using centred differences to compute these terms, the step size is usually so large that one value (of U, V) is positive and the other is negative. Thus the finite difference gives a near-zero derivative, and agrees relatively well with the analytic approach.

interest of maintaining a smooth design space.

B.4 Viscous Fluxes

When using the thin layer approximation, the viscous flux in the streamwise direction is considered to be negligible, so is omitted. There is therefore only one viscous flux that appears in the residual; it is given by

$$\frac{-1}{Re}\frac{\partial \widehat{S}}{\partial \eta} \tag{B.43}$$

where

$$\widehat{S} = J^{-1}\begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x\left(u m_1 + v m_2 + m_4\right) + \eta_y\left(u m_2 + v m_3 + m_5\right) \end{bmatrix} \tag{B.44}$$

$$m_1 = (\mu + \mu_t)\left(\frac{4}{3}\eta_x u_\eta - \frac{2}{3}\eta_y v_\eta\right) \tag{B.45}$$

$$m_2 = (\mu + \mu_t)\left(\eta_y u_\eta + \eta_x v_\eta\right) \tag{B.46}$$

$$m_3 = (\mu + \mu_t)\left(-\frac{2}{3}\eta_x u_\eta + \frac{4}{3}\eta_y v_\eta\right) \tag{B.47}$$

$$m_4 = \left(\frac{\mu Pr^{-1} + \mu_t Pr_t^{-1}}{\gamma - 1}\right)\eta_x \partial_\eta(a^2) \tag{B.48}$$

$$m_5 = \left(\frac{\mu Pr^{-1} + \mu_t Pr_t^{-1}}{\gamma - 1}\right)\eta_y \partial_\eta(a^2) \tag{B.49}$$

$$Re = \frac{\rho_\infty c a_\infty}{\mu_\infty} \qquad Pr = \frac{\bar{c}_p \bar{\mu}}{\bar{\kappa}_t} \tag{B.50}$$

This includes derivatives of the form $\partial_\eta(\alpha_D \beta_D \partial_\eta \theta)$, where α is a term like $(\mu + \mu_t)$, β is a term like $\frac{4}{3}\eta_x^2/J$, and θ is one of u, v, or a^2. This combination of operators is discretized as

$$\partial_\eta(\alpha_D \beta_D \partial_\eta \theta) \approx \nabla_\eta\left[\left(\varpi_\eta \alpha\right)_D\left(\varpi_\eta \beta\right)_D\left(\Delta_\eta \theta\right)\right] \tag{B.51}$$

Consider the differentiation of the viscous flux with respect to the grid node locations, G, one element of $\widehat{S}$ at a time. The derivative of the first element is simply zero.

$$\frac{\partial}{\partial G}\left(-Re^{-1}\frac{\partial \widehat{S}_1}{\partial \eta}\right) = 0 \tag{B.52}$$

The derivatives of the second and third elements are

$$\frac{\partial}{\partial G}\left(\frac{-1}{Re}\frac{\partial \widehat{S}_2}{\partial \eta}\right) = \frac{-1}{Re}\nabla_\eta\Bigg\{ \left[\varpi_\eta(\mu+\mu_t)\right]_D\Bigg[(\Delta_\eta u)_D\frac{4}{3}\varpi_\eta\frac{\partial(\eta_x^2/J)}{\partial G} + (\Delta_\eta v)_D\frac{1}{3}\varpi_\eta\frac{\partial(\eta_x\eta_y/J)}{\partial G} + (\Delta_\eta u)_D\varpi_\eta\frac{\partial(\eta_y^2/J)}{\partial G}\Bigg]\Bigg\} \tag{B.53}$$

$$\frac{\partial}{\partial G}\left(\frac{-1}{Re}\frac{\partial \widehat{S}_3}{\partial \eta}\right) = \frac{-1}{Re}\nabla_\eta\Bigg\{ \left[\varpi_\eta(\mu+\mu_t)\right]_D\Bigg[(\Delta_\eta v)_D\varpi_\eta\frac{\partial(\eta_x^2/J)}{\partial G} +(\Delta_\eta u)_D\frac{1}{3}\varpi_\eta\frac{\partial(\eta_x\eta_y/J)}{\partial G} + (\Delta_\eta v)_D\frac{4}{3}\varpi_\eta\frac{\partial(\eta_y^2/J)}{\partial G}\Bigg]\Bigg\} \tag{B.54}$$

The derivative of the fourth element is:

$$\frac{\partial}{\partial G}\left(\frac{-1}{Re}\frac{\partial \widehat{S}_4}{\partial \eta}\right) = (\varpi_\eta u)_D\frac{\partial}{\partial G}\left(\frac{-1}{Re}\frac{\partial \widehat{S}_2}{\partial \eta}\right) + (\varpi_\eta v)_D\frac{\partial}{\partial G}\left(\frac{-1}{Re}\frac{\partial \widehat{S}_3}{\partial \eta}\right) + \frac{-1}{Re}\nabla_\eta\left\{\left[\frac{\varpi_\eta\left(\frac{\mu}{Pr}+\frac{\mu_t}{Pr_t}\right)}{\gamma-1}\right]_D(\Delta_\eta a^2)_D\varpi_\eta\left(\frac{\partial(\eta_x^2/J)}{\partial G}+\frac{\partial(\eta_y^2/J}{\partial G}\right)\right\} \tag{B.55}$$

These equations include the derivatives of various combinations of the metrics with respect to G. One of these is

$$\frac{\partial(\eta_x^2/J)}{\partial G} = \frac{\partial}{\partial G}\left(\frac{y_\xi^2}{x_\xi y_\eta - x_\eta y_\xi}\right) \tag{B.56}$$

Carrying out the differentiation and simplifying gives

$$\frac{\partial(\eta_x^2/J)}{\partial G} = \Big[-\eta_x^2 y_\eta \qquad \eta_x^2 x_\eta - 2\eta_x\Big]_D\delta_\xi I + \Big[\eta_x^2 y_\xi \qquad -\eta_x^2 x_\xi\Big]_D\delta_\eta I \tag{B.57}$$

Similarly, the derivatives of the other two combinations of metrics are

$$\frac{\partial(\eta_x\eta_y/J)}{\partial G} = \Big[\eta_x - \eta_x\eta_y y_\eta \qquad \eta_x\eta_y x_\eta - \eta_y\Big]_D\delta_\xi I + \Big[\eta_x\eta_x y_\xi \qquad \eta_x\eta_y x_\xi\Big]_D\delta_\eta I \tag{B.58}$$

$$\frac{\partial(\eta_y^2/J)}{\partial G} = \Big[2\eta_y - \eta_y^2 y_\eta \qquad \eta_y^2 x_\eta\Big]_D\delta_\xi I + \Big[\eta_y^2 y_\xi \qquad -\eta_y^2 x_\xi\Big]_D\delta_\eta I \tag{B.59}$$

B.5 Turbulence Model

Turbulent effects are modelled using the Spalart-Allmaras turbulence model. It is a one-equation model, so for turbulent cases, there is one more conserved variable for each node:

$$Q_5 = \tilde{\nu}, \qquad \widehat{Q}_5 = \frac{\tilde{\nu}}{J} \tag{B.60}$$

The turbulent viscosity can be computed from this working variable using

$$\nu_t = \tilde{\nu} f_{v1} \tag{B.61}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{B.62}$$

$$\chi = \frac{\tilde{\nu}}{\nu} \tag{B.63}$$

The steady-state model is rewritten in curvilinear coordinates, and expressed as setting a residual to zero. So, the overall flow residual has one extra element per grid node. These elements have convection (M), production (P), destruction (D), and diffusion (N) terms as follows.

$$R_5 = J^{-1}(M - P + D - N) \tag{B.64}$$

where

$$M = U\tilde{\nu}_\xi + V\tilde{\nu}_\eta \tag{B.65}$$

$$P = \frac{c_{b1}\tilde{S}\tilde{\nu}}{Re} \tag{B.66}$$

$$D = \frac{c_{w1} f_w}{Re}\left(\frac{\tilde{\nu}}{d_w}\right)^2 \tag{B.67}$$

$$N = \frac{1}{\sigma Re}\left[(1 + c_{b2})T_1 - c_{b2}T_2\right] \tag{B.68}$$

$\tilde{S}$ is given in terms of the magnitude of the vorticity, S, and f_{v2}:

$$\tilde{S} = SRe + \frac{\tilde{\nu} f_{v2}}{\kappa^2 d_w^2} \tag{B.69}$$

$$S = |S'| = |\xi_x v_\xi + \eta_x v_\eta - \xi_y u_\xi - \eta_y u_\eta| \tag{B.70}$$

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{B.71}$$

The wall function is

$$f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \tag{B.72}$$

where

$$g = r + c_{w2}\left(r^6 - r\right) \tag{B.73}$$

and

$$r = \min\left(10,\ \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d_w^2}\right) \tag{B.74}$$

d_w is the distance from the nearest wall (the airfoil).

For the diffusion term, T_1 and T_2 are given by

$$\begin{aligned} T_1 = {} & \xi_x \frac{\partial}{\partial \xi}\left((\nu+\tilde{\nu})\xi_x\tilde{\nu}_\xi\right) + \eta_x \frac{\partial}{\partial \eta}\left((\nu+\tilde{\nu})\eta_x\tilde{\nu}_\eta\right) \\ & + \xi_y \frac{\partial}{\partial \xi}\left((\nu+\tilde{\nu})\xi_y\tilde{\nu}_\xi\right) + \eta_y \frac{\partial}{\partial \eta}\left((\nu+\tilde{\nu})\eta_y\tilde{\nu}_\eta\right) \end{aligned} \tag{B.75}$$

$$T_2 = (\nu+\tilde{\nu})\left(\xi_x \frac{\partial}{\partial \xi}\left(\xi_x\tilde{\nu}_\xi\right) + \eta_x \frac{\partial}{\partial \eta}\left(\eta_x\tilde{\nu}_\eta\right) + \xi_y \frac{\partial}{\partial \xi}\left(\xi_y\tilde{\nu}_\xi\right) + \eta_y \frac{\partial}{\partial \eta}\left(\eta_y\tilde{\nu}_\eta\right)\right) \tag{B.76}$$

The parameters for the Spalart-Allmaras model are

$$c_{b1} = 0.1355, \qquad c_{b2} = 0.622, \qquad c_{v1} = 7.1 \tag{B.77}$$

$$c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}, \qquad c_{w2} = 0.3, \qquad c_{w3} = 2.0 \tag{B.78}$$

$$\kappa = 0.41, \qquad \sigma = 2/3 \tag{B.79}$$

When discretized, the convection term uses a first-order upwind stencil:

$$\tilde{\nu}_\xi = \begin{cases} \nabla_\xi\tilde{\nu} & ,\ U \geq 0 \\ \Delta_\xi\tilde{\nu} & ,\ U < 0 \end{cases} \qquad \tilde{\nu}_\eta = \begin{cases} \nabla_\eta\tilde{\nu} & ,\ V \geq 0 \\ \Delta_\eta\tilde{\nu} & ,\ V < 0 \end{cases} \tag{B.80}$$

The vorticity is computed with the usual second-order centred stencil, but with a limit on how small it can be.

$$S = \max\left(8.5\times10^{-10},\ |S'|\right), \qquad S' = \xi_x\delta_\xi v + \eta_x\delta_\eta v - \xi_y\delta_\xi u - \eta_y\delta_\eta u \tag{B.81}$$

When discretized, the distance from the wall is evaluated as

$$d_w = \sqrt{(x-\mathcal{W}x)^2 + (y-\mathcal{W}y)^2} \tag{B.82}$$

where the operator $\mathcal{W}(\cdot)$ returns the value at the nearest node on the airfoil surface. That is,

$$\mathcal{W}(\cdot)_{j,k} = (\cdot)_{j',1}, \qquad j' = \max\left(j_{\text{tail1}},\ \min(j_{\text{tail2}},\ j)\right) \tag{B.83}$$

Finally, T_1 and T_2 contain terms like those found in the viscous flux, and are discretized using the operator given in (B.51), in both the ξ- and η-directions. So, the discrete T_1 and T_2 are

$$\begin{aligned} T_1 &+ \eta_y \nabla_\eta \left[\left(\varpi_\eta(\nu+\tilde{\nu})\right)_D \left(\varpi_\eta \eta_y\right)_D \Delta_\eta \tilde{\nu} \right] \\ &+ \eta_x \nabla_\eta \left[\left(\varpi_\eta(\nu+\tilde{\nu})\right)_D \left(\varpi_\eta \eta_x\right)_D \Delta_\eta \tilde{\nu} \right] \\ &+ \xi_y \nabla_\xi \left[\left(\varpi_\xi(\nu+\tilde{\nu})\right)_D \left(\varpi_\xi \xi_y\right)_D \Delta_\xi \tilde{\nu} \right] \\ &= \xi_x \nabla_\xi \left[\left(\varpi_\xi(\nu+\tilde{\nu})\right)_D \left(\varpi_\xi \xi_x\right)_D \Delta_\xi \tilde{\nu} \right] \end{aligned} \tag{B.84}$$

$$\begin{aligned} T_2 = (\nu+\tilde{\nu})_D \Big\{ &\ \eta_y \nabla_\eta \left[\left(\varpi_\eta \eta_y\right)_D \Delta_\eta \tilde{\nu} \right] + \eta_x \nabla_\eta \left[\left(\varpi_\eta \eta_x\right)_D \Delta_\eta \tilde{\nu} \right] \\ &+ \xi_y \nabla_\xi \left[\left(\varpi_\xi \xi_y\right)_D \Delta_\xi \tilde{\nu} \right] + \xi_x \nabla_\xi \left[\left(\varpi_\xi \xi_x\right)_D \Delta_\xi \tilde{\nu} \right] \Big\} \end{aligned} \tag{B.85}$$

B.5.1 Convection

The derivative of the convection term, M/J, with respect to the grid can be written as

$$\begin{aligned} \frac{\partial}{\partial G}\left(\frac{M}{J}\right) = &\left\{ \begin{array}{l} \Delta_\eta \tilde{\nu} \ , \text{ if } V < 0 \\ \nabla_\eta \tilde{\nu} \ , \text{ if } V \geq 0 \end{array} \right\}_D \frac{\partial}{\partial G}\left(-y_\xi u + x_\xi v\right) \\ &+ \left\{ \begin{array}{l} \Delta_\xi \tilde{\nu} \ , \text{ if } U < 0 \\ \nabla_\xi \tilde{\nu} \ , \text{ if } U \geq 0 \end{array} \right\}_D \frac{\partial}{\partial G}\left(y_\eta u - x_\eta v\right) \end{aligned} \tag{B.86}$$

This becomes

$$\begin{aligned} \frac{\partial}{\partial G}\left(\frac{M}{J}\right) = &\left\{ \begin{array}{l} \Delta_\eta \tilde{\nu} \ , \text{ if } V < 0 \\ \nabla_\eta \tilde{\nu} \ , \text{ if } V \geq 0 \end{array} \right\}_D \left[v \qquad -u \right]_D \delta_\xi I \\ &+ \left\{ \begin{array}{l} \Delta_\xi \tilde{\nu} \ , \text{ if } U < 0 \\ \nabla_\xi \tilde{\nu} \ , \text{ if } U \geq 0 \end{array} \right\}_D \left[-v \qquad u \right]_D \delta_\eta I \end{aligned} \tag{B.87}$$

B.5.2 Production

In the production term of the Spalart-Allmaras turbulence model, $-P/J$, the dependency on the grid node locations is via J, S, and d_w. So, one can express the derivative as

$$\frac{\partial}{\partial G}\left(-\frac{P}{J}\right) = -c_{b1}\tilde{\nu}\frac{\partial}{\partial G}\left(\frac{S}{J}\right) - \frac{c_{b1}\tilde{\nu}^2 f_{v2}}{Re\kappa^2}\frac{\partial}{\partial G}\left(\frac{1}{J d_w^2}\right) \tag{B.88}$$

For the derivative of S/J, when $|S'| > 8.5 \times 10^{-10}$,

$$\frac{\partial}{\partial G}\left(\frac{S}{J}\right) = \text{sign}(S')\left\{\begin{bmatrix} -\delta_\eta u & -\delta_\eta v \end{bmatrix}_D \delta_\xi I + \begin{bmatrix} \delta_\xi u & \delta_\xi v \end{bmatrix}_D \delta_\eta I\right\} \tag{B.89}$$

However, when $|S'| < 8.5 \times 10^{-10}$, S is independent of G, and

$$\frac{\partial}{\partial G}\left(\frac{S}{J}\right) = S\left\{\begin{bmatrix} y_\eta & -x_\eta \end{bmatrix}_D \delta_\xi I + \begin{bmatrix} -y_\xi & x_\xi \end{bmatrix}_D \delta_\eta I\right\} \tag{B.90}$$

The derivative of $1/(Jd_w^2)$ is given by

$$\begin{aligned}\frac{\partial}{\partial G}\left(\frac{1}{Jd_w^2}\right) = \frac{1}{d_w^2}&\left\{\begin{bmatrix} y_\eta & -x_\eta \end{bmatrix}_D \delta_\xi I + \begin{bmatrix} -y_\xi & x_\xi \end{bmatrix}_D \delta_\eta I\right\} \\ &- \frac{2}{Jd_w^4}\begin{bmatrix} (x - \mathcal{W}x) & (y - \mathcal{W}y) \end{bmatrix}_D (I - \mathcal{W}I)\end{aligned} \tag{B.91}$$

Note that like the $\delta_\xi(\cdot)$ operator, $\mathcal{W}(\cdot)$ can be expressed as a premultiplication of its argument by a matrix. Thus, $\mathcal{W}I$ is interpreted in the same fashion as $\delta_\xi I$, that is,

$$\mathcal{W}I = \frac{\partial \mathcal{W}(\cdot)}{\partial(\cdot)} \tag{B.92}$$

B.5.3 Destruction

In the destruction term of the Spalart-Allmaras turbulence model, f_w, J, and d_w depend on G. So,

$$\frac{\partial(D/J)}{\partial G} = \frac{c_{w1}\tilde{\nu}^2}{Re}\left(\frac{\partial f_w}{\partial G}\frac{1}{Jd_w^2} + f_w \frac{\partial}{\partial G}\left(\frac{1}{Jd_w^2}\right)\right) \tag{B.93}$$

The derivative of $1/(Jd_w^2)$ is computed as in the production term. The derivative of f_w is lengthy, but straightforward:

$$\frac{\partial f_w}{\partial G} = \frac{c_{w3}^6\left(1 + c_{w3}^6\right)^{1/6}}{\left(g^6 + c_{w3}^6\right)^{7/6}}\left(1 - c_{w2} + 6c_{w2}r^5\right)\left(-\frac{r^2 Re\kappa^2}{\tilde{\nu}}\right)\left(d_w^2\frac{\partial S}{\partial G} + S\frac{\partial(d_w^2)}{\partial G}\right) \tag{B.94}$$

The derivative of S with respect to G, for nodes where $|S'| > 8.5 \times 10^{-10}$, is

$$\begin{aligned}\frac{\partial S}{\partial G} = J\text{sign}(S')&\left\{\begin{bmatrix} -\delta_\eta u & -\delta_\eta v \end{bmatrix}_D \delta_\xi I + \begin{bmatrix} \delta_\xi u & \delta_\xi v \end{bmatrix}_D \delta_\eta I\right\} \\ JS&\left\{\begin{bmatrix} -y_\eta & x_\eta \end{bmatrix}_D \delta_\xi I + \begin{bmatrix} y_\xi & -x_\xi \end{bmatrix}_D \delta_\eta I\right\}\end{aligned} \tag{B.95}$$

However, when $|S'| < 8.5 \times 10^{-10}$, S becomes independent of G, and

$$\frac{\partial S}{\partial G} = 0 \tag{B.96}$$

The square of the distance from the wall is differentiated as

$$\frac{\partial(d_w^2)}{\partial G} = 2\begin{bmatrix} (x - \mathcal{W}x) & (y - \mathcal{W}y) \end{bmatrix}_D (I - \mathcal{W}I) \tag{B.97}$$

B.5.4 Diffusion

In the diffusion term of the Spalart-Allmaras model, both T_1 and T_2 depend on G, so the grid sensitivity is

$$\frac{\partial}{\partial G}\left(-\frac{N}{J}\right) = \frac{-1}{\sigma Re}\left((1+c_{b2})\frac{\partial(T_1/J)}{\partial G} - c_{b2}\frac{\partial(T_2/J)}{\partial G}\right) \tag{B.98}$$

with

$$\begin{aligned}\frac{\partial(T_1/J)}{\partial G} = &\Big[\nabla_\eta\left(A_D\varpi_\eta\eta_y\right) \quad -\nabla_\eta\left(A_D\varpi_\eta\eta_x\right)\Big]_D\delta_\xi I \\ &+\Big[-\nabla_\xi\left(B_D\varpi_\xi\xi_y\right) \quad \nabla_\xi\left(B_D\varpi_\xi\xi_x\right)\Big]_D\delta_\eta I \\ &+x_\xi\nabla_\eta\left(A_D\varpi_\eta\frac{\partial\eta_y}{\partial G}\right) - y_\xi\nabla_\eta\left(A_D\varpi_\eta\frac{\partial\eta_x}{\partial G}\right) \\ &-x_\eta\nabla_\xi\left(B_D\varpi_\xi\frac{\partial\xi_y}{\partial G}\right) + y_\eta\nabla_\xi\left(B_D\varpi_\xi\frac{\partial\xi_x}{\partial G}\right)\end{aligned} \tag{B.99}$$

where

$$A_D = \left(\varpi_\eta(\nu+\tilde{\nu})\right)_D\left(\Delta_\eta\tilde{\nu}\right)_D \qquad B_D = \left(\varpi_\xi(\nu+\tilde{\nu})\right)_D\left(\Delta_\xi\tilde{\nu}\right)_D \tag{B.100}$$

and

$$\begin{aligned}\frac{\partial(T_2/J)}{\partial G} = (\nu+\tilde{\nu})_D\Bigg\{ &\Big[\nabla_\eta\left((\Delta_\eta\tilde{\nu})_D\varpi_\eta\eta_y\right) \quad -\nabla_\eta\left((\Delta_\eta\tilde{\nu})_D\varpi_\eta\eta_x\right)\Big]_D\delta_\xi I \\ &+\Big[-\nabla_\xi\left((\Delta_\xi\tilde{\nu})_D\varpi_\xi\xi_y\right) \quad \nabla_\xi\left((\Delta_\xi\tilde{\nu})_D\varpi_\xi\xi_x\right)\Big]_D\delta_\eta I \\ &+x_\xi\nabla_\eta\left((\Delta_\eta\tilde{\nu})_D\varpi_\eta\frac{\partial\eta_y}{\partial G}\right) - y_\xi\nabla_\eta\left((\Delta_\eta\tilde{\nu})_D\varpi_\eta\frac{\partial\eta_x}{\partial G}\right) \\ &-x_\eta\nabla_\xi\left((\Delta_\xi\tilde{\nu})_D\varpi_\xi\frac{\partial\xi_y}{\partial G}\right) + y_\eta\nabla_\xi\left((\Delta_\xi\tilde{\nu})_D\varpi_\xi\frac{\partial\xi_x}{\partial G}\right)\Bigg\}\end{aligned} \tag{B.101}$$

Here, the derivative of ξ_x is derived as

$$\begin{aligned}\frac{\partial\xi_x}{\partial G} &= \frac{\partial}{\partial G}\left(\frac{y_\eta}{x_\xi y_\eta - x_\eta y_\xi}\right) \\ &= \Big[-y_\eta J^2 y_\eta \quad y_\eta J^2 x_\eta\Big]_D\delta_\xi I + \Big[y_\eta J^2 y_\xi \quad -y_\eta J^2 x_\xi + J\Big]_D\delta_\eta I \\ &= -\Big[\xi_x^2 \quad \xi_x\xi_y\Big]_D\delta_\xi I - \Big[\xi_x\eta_x \quad \xi_x\eta_y - J\Big]_D\delta_\eta I\end{aligned} \tag{B.102}$$

Similarly,

$$\frac{\partial\xi_y}{\partial G} = -\Big[\xi_y\xi_x \quad \xi_y^2\Big]_D\delta_\xi I - \Big[\xi_y\eta_x + J \quad \xi_y\eta_y\Big]_D\delta_\eta I \tag{B.103}$$

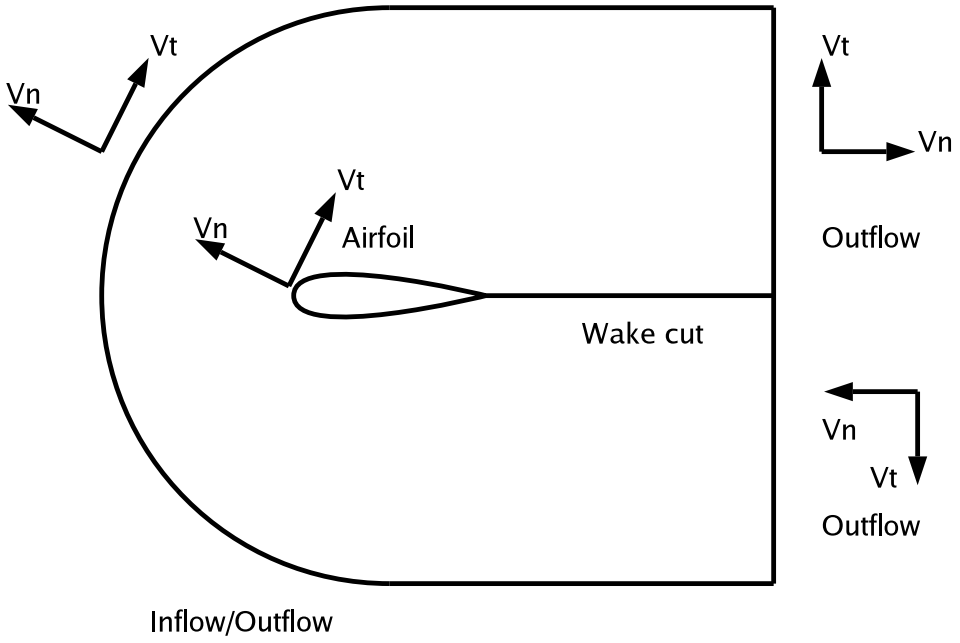

Figure B.3: Domain boundaries

$$\frac{\partial \eta_x}{\partial G} = -\Big[\eta_x\xi_x \qquad \eta_x\xi_y + J\Big]_D \delta_\xi I - \Big[\eta_x^2 \qquad \eta_x\eta_y\Big]_D \delta_\eta I \tag{B.104}$$

$$\frac{\partial \eta_y}{\partial G} = -\Big[\eta_y\xi_x - J \qquad \eta_y\xi_y\Big]_D \delta_\xi I - \Big[\eta_y\eta_x \qquad \eta_y^2\Big]_D \delta_\eta I \tag{B.105}$$

B.6 Boundary Conditions

The implicit boundary conditions used by the flow solver appear as extra blocks in the residual vector. These must be differentiated to compute $\partial R_{i,j}/\partial G$, where (i, j) lies on a boundary. The boundaries of the domain are shown in Figure B.3. The following subsections discuss each boundary in detail. The boundary conditions are computed using the components of the flow velocity that are normal and tangential to the domain boundary. These components, V_n and V_t, are shown for each boundary in Figure B.3, and will be considered first.

B.6.1 Normal and Tangential Velocity Components

The normal and tangential velocity components are computed differently, depending on which set of grid lines are aligned with the boundary. For the boundaries that lie on

ξ grid lines (i.e. the inflow/outflow boundary and the airfoil boundary), the velocity components are calculated using

$$V_n = \frac{\eta_x u + \eta_y v}{\sqrt{\eta_x^2 + \eta_y^2}} = \frac{-y_\xi u + x_\xi v}{\sqrt{y_\xi^2 + x_\xi^2}} \tag{B.106}$$

$$V_t = \frac{\eta_y u - \eta_x v}{\sqrt{\eta_x^2 + \eta_y^2}} = \frac{x_\xi u + y_\xi v}{\sqrt{y_\xi^2 + x_\xi^2}} \tag{B.107}$$

Differentiating with respect to the grid node locations yields

$$\frac{\partial V_n}{\partial G} = \left[\begin{array}{cc} vA - x_\xi B & -uA - y_\xi B \end{array} \right]_D (\delta_\xi I) \tag{B.108}$$

$$\frac{\partial V_t}{\partial G} = \left[\begin{array}{cc} uA - x_\xi C & vA - y_\xi C \end{array} \right]_D (\delta_\xi I) \tag{B.109}$$

where

$$A = (y_\xi^2 + x_\xi^2)^{-1/2} \tag{B.110}$$

$$B = (-y_\xi u + x_\xi v)(y_\xi^2 + x_\xi^2)^{-3/2} \tag{B.111}$$

$$C = (x_\xi u + y_\xi v)(y_\xi^2 + x_\xi^2)^{-3/2} \tag{B.112}$$

For the boundaries that lie on η grid lines (i.e. the two outflow boundaries), the velocity components are calculated using

$$V_n = \frac{\xi_x u + \xi_y v}{\sqrt{\xi_x^2 + \xi_y^2}} = \frac{y_\eta u - x_\eta v}{\sqrt{y_\eta^2 + x_\eta^2}} \tag{B.113}$$

$$V_t = \frac{-\xi_y u + \xi_x v}{\sqrt{\xi_x^2 + \xi_y^2}} = \frac{x_\eta u + y_\eta v}{\sqrt{y_\eta^2 + x_\eta^2}} \tag{B.114}$$

Differentiating with respect to the grid node locations yields

$$\frac{\partial V_n}{\partial G} = \left[\begin{array}{cc} -vA - x_\eta B & uA - y_\eta B \end{array} \right]_D (\delta_\eta I) \tag{B.115}$$

$$\frac{\partial V_t}{\partial G} = \left[\begin{array}{cc} uA - x_\eta C & vA - y_\eta C \end{array} \right]_D (\delta_\eta I) \tag{B.116}$$

where

$$A = (y_\eta^2 + x_\eta^2)^{-1/2} \tag{B.117}$$

$$B = (y_\eta u - x_\eta v)(y_\eta^2 + x_\eta^2)^{-3/2} \tag{B.118}$$

$$C = (x_\eta u + y_\eta v)(y_\eta^2 + x_\eta^2)^{-3/2} \tag{B.119}$$

B.6.2 Inflow/Outflow Boundary

The inflow/outflow boundary conditions are derived using Riemann invariants. When the flow direction is into the domain (inflow, $V_n \leq 0$), the residual terms are

$$R_{j,k_{\max}} = \begin{bmatrix} \left(V_n - \frac{2a}{\gamma-1}\right)_{j,k_{\max}} - \left(\left(V_{n,\infty}\right)_{j,k_{\max}} - \frac{2a_\infty}{\gamma-1}\right) \\ \left(V_n + \frac{2a}{\gamma-1}\right)_{j,k_{\max}} - \left(V_n + \frac{2a}{\gamma-1}\right)_{j,k_{\max}-1} \\ \left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max}} - S_\infty \\ \left(V_t\right)_{j,k_{\max}} - \left(V_{t,\infty}\right)_{j,k_{\max}} \\ \tilde{\nu}_{j,k_{\max}} - \tilde{\nu}_\infty \end{bmatrix} \tag{B.120}$$

where $\left(V_{n,\infty}\right)_{j,k_{\max}}$ is the freestream velocity component that is normal to the grid at node $j, k_{\max}$, and $\left(V_{t,\infty}\right)_{j,k_{\max}}$ is similar. They can be differentiated as laid out in Section B.6.1.

When differentiated with respect to the grid node locations, this becomes

$$\frac{R_{j,k_{\max}}}{\partial G} = \begin{bmatrix} \frac{\partial (V_n)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_{n,\infty})_{j,k_{\max}}}{\partial G} \\ \frac{\partial (V_n)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_n)_{j,k_{\max}-1}}{\partial G} \\ 0 \\ \frac{\partial (V_t)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_{t,\infty})_{j,k_{\max}}}{\partial G} \\ 0 \end{bmatrix} \tag{B.121}$$

The freestream quantities $(V_n)_\infty$ and $(V_t)_\infty$ differ from $(V_n)_{j,k}$ and $(V_t)_{j,k}$

The equations for outflow have more terms that are extrapolated from the solution:

$$R_{j,k_{\max}} = \begin{bmatrix} \left(V_n - \frac{2a}{\gamma-1}\right)_{j,k_{\max}} - \left(\left(V_{n,\infty}\right)_{j,k_{\max}} - \frac{2a_\infty}{\gamma-1}\right) \\ \left(V_n + \frac{2a}{\gamma-1}\right)_{j,k_{\max}} - \left(V_n + \frac{2a}{\gamma-1}\right)_{j,k_{\max}-1} \\ \left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max}} - \left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max}-1} \\ \left(V_t\right)_{j,k_{\max}} - \left(V_t\right)_{j,k_{\max}-1} \\ \tilde{\nu}_{j,k_{\max}} - \tilde{\nu}_{j,k_{\max}-1} \end{bmatrix} \tag{B.122}$$

When differentiated with respect to the grid node locations, this becomes

$$\frac{R_{j,k_{\max}}}{\partial G} = \begin{bmatrix} \frac{\partial (V_n)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_{n,\infty})_{j,k_{\max}}}{\partial G} \\ \frac{\partial (V_n)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_n)_{j,k_{\max}-1}}{\partial G} \\ 0 \\ \frac{\partial (V_t)_{j,k_{\max}}}{\partial G} - \frac{\partial (V_t)_{j,k_{\max}-1}}{\partial G} \\ 0 \end{bmatrix} \tag{B.123}$$

B.6.3 Outflow Boundary

For an inviscid analysis, Riemann invariants are also used at the two outflow boundaries downstream of the airfoil. The only differences between the treatment of these boundaries and the inflow/outflow boundary discussed above are the nodal indices and the calculation of V_n and V_t (see Section B.6.1).

When the flow is viscous, however, the Riemann invariants are replaced with first-order extrapolation at the outflow boundaries. At $j = 1$,

$$R_{1,k} = \begin{bmatrix} \rho_{1,k} - \rho_{2,k} \\ (\rho u)_{1,k} - (\rho u)_{2,k} \\ (\rho v)_{1,k} - (\rho v)_{2,k} \\ p_{1,k} - p_{2,k} \\ \tilde{\nu}_{1,k} - \tilde{\nu}_{2,k} \end{bmatrix} \tag{B.124}$$

This is independent of the grid node locations, so these elements of $\partial R/\partial G$ are zero. The same result is found for the outflow boundary at $j = j_{\max}$.

B.6.4 Airfoil Boundary

The airfoil boundary is treated differently, depending on whether the flow is viscous or inviscid. For viscous flow,

$$R_{j,1} = \begin{bmatrix} \rho_{j,1} - \rho_{j,2} \\ (\rho u)_{j,1} \\ (\rho v)_{j,1} \\ p_{j,1} - p_{j,2} \\ \tilde{\nu}_{j,1} \end{bmatrix} \tag{B.125}$$

Thus for viscous flow, the residual along the airfoil boundary is independent of the grid node locations. However, for inviscid flow,

$$R_{j,1} = \begin{bmatrix} (V_n)_{j,1} \\ (V_t)_{j,1} - 2(V_t)_{j,2} + (V_t)_{j,3} \\ p_{j,1} - 2p_{j,2} + p_{j,3} \\ \frac{\gamma p_{j,1}}{\gamma - 1} + \frac{\rho_{j,1}}{2}(u_{j,1}^2 + v_{j,1}^2) - \rho_{j,1} h_\infty \end{bmatrix} \tag{B.126}$$

So

$$\frac{\partial R_{j,1}}{\partial G} = \begin{bmatrix} \frac{\partial (V_n)_{j,1}}{\partial G} \\ \frac{\partial (V_t)_{j,1}}{\partial G} - 2\frac{\partial (V_t)_{j,2}}{\partial G} + \frac{\partial (V_t)_{j,3}}{\partial G} \\ 0 \\ 0 \end{bmatrix} \tag{B.127}$$

B.6.5 Wake Cut

The residual at the wake cut is computed as

$$R_{j,1} = \begin{bmatrix} (Q_1)_{j,1} - \frac{1}{2}\left((Q_1)_{j,2} + (Q_1)_{j_{\max}-j+1,2}\right) \\ (Q_2)_{j,1} - \frac{1}{2}\left((Q_2)_{j,2} + (Q_2)_{j_{\max}-j+1,2}\right) \\ (Q_3)_{j,1} - \frac{1}{2}\left((Q_3)_{j,2} + (Q_3)_{j_{\max}-j+1,2}\right) \\ p_{j,1} - \frac{1}{2}\left(p_{j,2} + p_{j_{\max}-j+1,2}\right) \\ (Q_5)_{j,1} - \frac{1}{2}\left((Q_5)_{j,2} + (Q_5)_{j_{\max}-j+1,2}\right) \end{bmatrix} \tag{B.128}$$

This is independent of the grid node locations.

Appendix C

Forming $\partial K_e/\partial G_e$

C.1 Element Stiffness Evaluation

C.1.1 Continuous Equations

Consider the strain energy stored in a 2D body:

$$E_p = \frac{1}{2}\int_V \epsilon^{\mathrm{T}}\sigma dV \tag{C.1}$$

where the strain, ϵ, and stress, σ, are

$$\epsilon = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = D \begin{bmatrix} u \\ v \end{bmatrix} \tag{C.2}$$

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \tag{C.3}$$

and u and v are the displacement of the material in the x- and y-directions. The stress and strain are related by the elasticity matrix, C.

$$\sigma = C\epsilon \tag{C.4}$$

$$C = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1}{2}-\nu \end{bmatrix} \tag{C.5}$$

where E is Young's modulus, and ν is Poisson's ratio.

This allows the strain energy to be written as a function of u.

$$E_p = \frac{1}{2}\int_V (Du)^{\mathrm{T}} C Du dV \tag{C.6}$$

C.1.2 Spatial Discretization

The strain energy is discretized in space by considering the body to be composed of several finite quadrilateral elements, so the integral over the body becomes the sum of the integrals over each of the elements. For each element, discrete approximations for strain and for the element volume integral are introduced.

The analytic expression for strain (C.2) is discretized using the displacements of the corner nodes of the element as follows:

$$\epsilon = B_e U_e \tag{C.7}$$

or,

$$\epsilon = \begin{bmatrix} J^*_{11} & J^*_{12} & 0 & 0 \\ 0 & 0 & J^*_{21} & J^*_{22} \\ J^*_{21} & J^*_{22} & J^*_{11} & J^*_{12} \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \begin{bmatrix} U_1 \\ v_1 \\ U_2 \\ v_2 \\ U_3 \\ v_3 \\ U_4 \\ v_4 \end{bmatrix} \tag{C.8}$$

This uses an interpolation function, $N_i(\xi, \eta)$, given by

$$N_i(\xi, \eta) = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i) \tag{C.9}$$

where i denotes the corner number (they numbered in a clockwise manner). The interpolation function is defined in a local coordinate system, (ξ, η), in which the corners of the element are $(\pm 1, \pm 1)$.

The Jacobian of the transformation, J, is given by

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{C.10}$$

and J^* is the inverse of J.

Using (C.7), the strain energy (C.6) is partly discretized and written as

$$E_p = \frac{1}{2}\sum_e \int_{V_e} U_e^{\mathrm{T}} B_e^{\mathrm{T}} C_e B_e U_e dV_e \tag{C.11}$$

Because U_e is not a function of the position within the element, it can be brought outside of the integral, giving

$$E_p = \frac{1}{2}\sum_e U_e^{\mathrm{T}} K_e U_e \tag{C.12}$$

where the element stiffness matrix is

$$K_e = \int_{V_e} B_e^{\mathrm{T}} C_e B_e dV_e \tag{C.13}$$

The numerical integration over the element volume is performed using Gaussian quadrature. In the local coordinates, we have

$$K_e = \int_{-1}^{1}\int_{-1}^{1} B_e^{\mathrm{T}} C_e B_e |J_e| d\eta d\xi = \sum_{i=1}^{p_\xi}\sum_{j=1}^{p_\eta} w_i w_j B_{ij}^{\mathrm{T}} C_e B_{ij} |J_{ij}| d\eta d\xi \tag{C.14}$$

where $p_\xi = p_\eta = 2$ are the number of Gauss points, and $w_i = w_j = 1$ are the weights at each point. The Gauss points are situated at $(\pm\frac{1}{\sqrt{3}}, \pm\frac{1}{\sqrt{3}})$. The subscripts on B and J indicate the Gauss point at which they are evaluated. $|J|$ denotes the determinant of J.

C.1.3 Young's Modulus

The spatial variation of Young's modulus is based on element size and quality. It is given by

$$E_e^{(i)} = \frac{1}{a_e}\left(\frac{\hat{\Phi}_e^{(i)}}{\hat{\Phi}_e^{(0)}}\right)^{\gamma} \tag{C.15}$$

where a_e is the area of the element, $\hat{\Phi}$ is the distortion measure, the superscript (i) indicates the increment of a multi-increment perturbation ($i = 0$ for the parent grid), and γ is a user-selected stiffening exponent.

The element area is given by half the magnitude of the cross product of the diagonal vectors of the element:

$$a_e = \frac{1}{2}\left[(G_{e,5} - G_{e,1})(G_{e,8} - G_{e,4}) - (G_{e,7} - G_{e,3})(G_{e,6} - G_{e,2})\right] \tag{C.16}$$

where the grid nodes for the element, G_e, are interpreted as

$$G_e = \begin{bmatrix} G_{e,1} \\ G_{e,2} \\ G_{e,3} \\ G_{e,4} \\ G_{e,5} \\ G_{e,6} \\ G_{e,7} \\ G_{e,8} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{C.17}$$

and (x_i, y_i) are the coordinates of the i^{th} corner of the element.

The ratio of the element distortion measure, $\hat{\Phi}_e$, to its original value acts to prevent the element quality from deteriorating. This reflects the philosophy of not improving the element quality, but rather retaining the high quality of the parent grid.[1] The distortion measure is computed as

$$\hat{\Phi}_e = \left[\sum_{j=1}^{4} \left(\hat{\Phi}_{\Delta_j} \right)^P \right]^{\frac{1}{P}}, \qquad P = 2 \tag{C.18}$$

Here, the subscript Δ_j refers to the j^{th} triangular sub-element within the quadrilateral, generally having vertices at the element corner nodes $j-1$, j, and $j+1$. Then, $\hat{\Phi}_{\Delta_j}$ is the ratio of the radii of the smallest circumscribed circle and the largest inscribed circle of that triangular sub-element:

$$\hat{\Phi}_{\Delta_j} = \left(\frac{R_\Delta}{r_\Delta} \right)_j = \left(\frac{s_\Delta l_1 l_2 l_3}{4 A_\Delta^2} \right)_j \tag{C.19}$$

where l are the lengths of the sides of the triangle, A_Δ is the area of the triangle, and s_Δ is the semi-perimeter.

$$(A_\Delta)_j = \left(\sqrt{s_\Delta (s_\Delta - l_1)(s_\Delta - l_2)(s_\Delta - l_3)} \right)_j, \qquad (s_\Delta)_j = \left(\frac{l_1 + l_2 + l_3}{2} \right)_j \tag{C.20}$$

With some simplification, one obtains

$$\hat{\Phi}_{\Delta_j} = \left(\frac{2 l_1 l_2 l_3}{(-l_1 + l_2 + l_3)(l_1 - l_2 + l_3)(l_1 + l_2 - l_3)} \right)_j \tag{C.21}$$

[1] A high quality parent grid is therefore imperative.

C.2 Differentiation

$$\frac{\partial K_e}{\partial G_e} = \sum_{i=1}^{p_\xi} \sum_{j=1}^{p_\eta} \left(w_i w_j \frac{\partial \left(B_{ij}^{\mathrm{T}} C_{ij} B_{ij} |J_{ij}| \right)}{\partial G_e} \right) \tag{C.22}$$

The main portion of this third-order tensor is $\partial(B_{ij}^{\mathrm{T}} C_{ij} B_{ij} |J_{ij}|$; consider its derivatives, neglecting the subscripts ij for simplicity. Before differentiating, however, notice that in (C.8), B is written in terms of J^*,

$$J^* = \frac{1}{|J|} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \tag{C.23}$$

It would therefore be simpler to differentiate the product $B|J|$ than B itself. One can therefore re-arrange, differentiate, and simplify as follows:

$$\begin{aligned} \frac{\partial}{\partial G_e} \left[\frac{(B^{\mathrm{T}}|J|) C (B|J|)}{|J|} \right] = & \frac{\partial (B^{\mathrm{T}}|J|)}{\partial G_e} CB + B^{\mathrm{T}} \frac{\partial C}{\partial G_e} B|J| \\ & + B^{\mathrm{T}} C \frac{\partial (B|J|)}{\partial G_e} - B^{\mathrm{T}} CB \frac{\partial |J|}{\partial G_e} \end{aligned} \tag{C.24}$$

Using tensor notation, this is of the form

$$\frac{\partial K_{ab}}{\partial G_c} \tag{C.25}$$

and it can be shown that the first and third terms in (C.24) are the transpose of each other (C is symmetric). That is,

$$\frac{\partial (B_{da}|J|)}{\partial G_c} C_{de} B_{eb} = \left(B_{da} C_{de} \frac{\partial (B_{eb}|J|)}{\partial G_c} \right) \delta_{ab} \tag{C.26}$$

where δ_{ab} is the Kronecker delta,

$$\delta_{ab} = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases} \tag{C.27}$$

The following sections describe the calculation of each of the Jacobian matrices in (C.24).

C.2.1 $\partial(B|J|)/\partial G_e$

Referring to (C.8) and (C.23), the product of B with the determinant of the metric Jacobian can be written as

$$B|J| = \begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \tag{C.28}$$

The product of these two matrices is of the form

$$B|J| = \begin{bmatrix} b_{\mathrm{B}} & 0 & b_{\mathrm{D}} & 0 & b_{\mathrm{F}} & 0 & b_{\mathrm{H}} & 0 \\ 0 & b_{\mathrm{A}} & 0 & b_{\mathrm{C}} & 0 & b_{\mathrm{E}} & 0 & b_{\mathrm{G}} \\ b_{\mathrm{A}} & b_{\mathrm{B}} & b_{\mathrm{C}} & b_{\mathrm{D}} & b_{\mathrm{E}} & b_{\mathrm{F}} & b_{\mathrm{G}} & b_{\mathrm{H}} \end{bmatrix} \tag{C.29}$$

where, for example

$$b_{\mathrm{A}} = -J_{21}\frac{\partial N_1}{\partial \xi} + J_{11}\frac{\partial N_1}{\partial \eta}, \qquad b_{\mathrm{B}} = J_{22}\frac{\partial N_1}{\partial \xi} - J_{12}\frac{\partial N_1}{\partial \eta} \tag{C.30}$$

The other terms are similar. This form for $B|J|$ allows for some minor reductions in memory usage and CPU time (both when forming it and multiplying it with other matrices); these benefits are also encountered for the derivative of $B|J|$.

The derivatives of these terms of the matrix are.

$$\frac{\partial b_A}{\partial G_e} = \begin{bmatrix} -\frac{\partial N_1}{\partial \eta}\frac{\partial N_1}{\partial \xi} + \frac{\partial N_1}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ -\frac{\partial N_2}{\partial \eta}\frac{\partial N_1}{\partial \xi} + \frac{\partial N_2}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ -\frac{\partial N_3}{\partial \eta}\frac{\partial N_1}{\partial \xi} + \frac{\partial N_3}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ -\frac{\partial N_4}{\partial \eta}\frac{\partial N_1}{\partial \xi} + \frac{\partial N_4}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \end{bmatrix}^{\mathrm{T}}, \qquad \frac{\partial b_B}{\partial G_e} = \begin{bmatrix} 0 \\ \frac{\partial N_1}{\partial \eta}\frac{\partial N_1}{\partial \xi} - \frac{\partial N_1}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ \frac{\partial N_2}{\partial \eta}\frac{\partial N_1}{\partial \xi} - \frac{\partial N_2}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ \frac{\partial N_3}{\partial \eta}\frac{\partial N_1}{\partial \xi} - \frac{\partial N_3}{\partial \xi}\frac{\partial N_1}{\partial \eta} \\ 0 \\ \frac{\partial N_4}{\partial \eta}\frac{\partial N_1}{\partial \xi} - \frac{\partial N_4}{\partial \xi}\frac{\partial N_1}{\partial \eta} \end{bmatrix}^{\mathrm{T}} \tag{C.31}$$

Again, the other terms are similar. Notice that the first element in $\partial b_A/\partial G_e$ and the second element in $\partial b_B/\partial G_e$ evaluate to zero.

These derivatives can be compiled to form the third-order tensor $\partial(B|J|)/\partial G_e$.

C.2.2 $\partial C/\partial G_e$

The only portion of the elasticity matrix, C, that depends on the element corner node locations, G_e, is Young's modulus, E. The derivative of C with respect to G_e is therefore

$$\frac{\partial C}{\partial G_e} = \left(\frac{\partial E}{\partial G_e}\right) \frac{1}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1}{2}-\nu \end{bmatrix} \tag{C.32}$$

and only the derivative of the scalar E with respect to the element corner node locations need be computed. Referring to (C.15), this derivative is

$$\frac{\partial E_e}{\partial G} = -\frac{1}{a_e^2}\left[\frac{\partial a_e}{\partial G_e}\right]\left(\frac{\hat{\Phi}_e}{\hat{\Phi}_e^{(0)}}\right)^{\gamma} + \frac{\gamma}{a_e \hat{\Phi}_e^{(0)}}\left(\frac{\hat{\Phi}_e}{\hat{\Phi}_e^{(0)}}\right)^{\gamma-1}\left[\frac{\partial \hat{\Phi}_e}{\partial G_e}\right] \tag{C.33}$$

The derivative of the element area with respect to the corner node locations is

$$\frac{\partial a_e}{\partial G_e} = \frac{1}{2}\begin{bmatrix} -(G_8-G_4) \\ (G_7-G_3) \\ (G_6-G_2) \\ -(G_5-G_1) \\ (G_8-G_4) \\ -(G_7-G_3) \\ -(G_6-G_2) \\ (G_5-G_1) \end{bmatrix}^{\mathrm{T}} \tag{C.34}$$

The element distortion derivative is

$$\frac{\partial \hat{\Phi}_e}{\partial G_e} = \frac{1}{\hat{\Phi}_e}\left(\sum_{j=1}^{4} \hat{\Phi}_{\Delta_j} \frac{\partial \hat{\Phi}_{\Delta_j}}{\partial G_e}\right) \tag{C.35}$$

where

$$\begin{aligned} \frac{\partial \hat{\Phi}_{\Delta_j}}{\partial G_e} = \quad & \frac{2l_2l_3}{t}\left(1 + \frac{l_1}{-l_1+l_2+l_3} + \frac{l_1}{l_1-l_2+l_3} + \frac{l_1}{l_1+l_2-l_3}\right)\frac{\partial l_1}{\partial G_e} \\ & + \frac{2l_1l_3}{t}\left(1 + \frac{l_2}{-l_1+l_2+l_3} + \frac{l_2}{l_1-l_2+l_3} + \frac{l_2}{l_1+l_2-l_3}\right)\frac{\partial l_2}{\partial G_e} \\ & + \frac{2l_1l_3}{t}\left(1 + \frac{l_3}{-l_1+l_2+l_3} + \frac{l_3}{l_1-l_2+l_3} + \frac{l_3}{l_1+l_2-l_3}\right)\frac{\partial l_3}{\partial G_e} \end{aligned} \tag{C.36}$$

$$t = (-l_1+l_2+l_3)(l_1-l_2+l_3)(l_1+l_2-l_3) \tag{C.37}$$

The differentiations of the lengths of the sides of the triangular sub-elements is straightforward. For example, for a side whose length is given by

$$l = \sqrt{(G_b - G_a)^2 + (G_{b+1} - G_{a+1})^2} \tag{C.38}$$

The nonzero components of $\partial l/\partial G_e$ are

$$\frac{\partial l}{\partial G_a} = \frac{G_a - G_b}{l} \tag{C.39}$$

$$\frac{\partial l}{\partial G_b} = \frac{G_b - G_a}{l} \tag{C.40}$$

$$\frac{\partial l}{\partial G_{a+1}} = \frac{G_{a+1} - G_{b+1}}{l} \tag{C.41}$$

$$\frac{\partial l}{\partial G_{b+1}} = \frac{G_{b+1} - G_{a+1}}{l} \tag{C.42}$$

C.2.3 $\partial|J|/\partial G_e$

The Jacobian of the coordinate transformation is differentiated with respect to the element corner nodes as follows:

$$\frac{\partial |J|}{\partial G_e} = \frac{\partial J_{11}}{\partial G_e} J_{22} + J_{11} \frac{\partial J_{22}}{\partial G_e} - \frac{\partial J_{12}}{\partial G_e} J_{21} - J_{12} \frac{\partial J_{21}}{\partial G_e} \tag{C.43}$$

where

$$\frac{\partial J_{11}}{\partial (G_e)_i} = \begin{bmatrix} \dfrac{\partial N_i}{\partial \xi} & 0 \end{bmatrix} \tag{C.44}$$

$$\frac{\partial J_{12}}{\partial (G_e)_i} = \begin{bmatrix} 0 & \dfrac{\partial N_i}{\partial \xi} \end{bmatrix} \tag{C.45}$$

$$\frac{\partial J_{21}}{\partial (G_e)_i} = \begin{bmatrix} \dfrac{\partial N_i}{\partial \eta} & 0 \end{bmatrix} \tag{C.46}$$

$$\frac{\partial J_{22}}{\partial (G_e)_i} = \begin{bmatrix} 0 & \dfrac{\partial N_i}{\partial \eta} \end{bmatrix} \tag{C.47}$$

The subscript i denotes the corner node index.