High-Fidelity Aerodynamic Shape Optimization with High-Order Spatial Discretization

by

Nicholas Albert Lee Holt

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

High-Fidelity Aerodynamic Shape Optimization with High-Order Spatial Discretization

Nicholas Albert Lee Holt
Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto
2014

A gradient-based optimization procedure for use with aerodynamic optimization has been developed to make use of a more accurate high-order analysis. Second-, third-, and fourth-order summation-by-parts operators are used with boundary conditions enforced by simultaneous approximation terms to improve stability. The optimizer has been extensively verified using the complex-step method and directional derivative test, and validated using an inverse design and inviscid twist optimization. A matrix-vector product technique is introduced which circumvents the large memory requirements of storing the high-order flux Jacobian matrix when solving the flow adjoint equations. Viscous optimization in two and three dimensions has shown direct benefits to high-order analysis within optimization; accurate analysis can lead to an improved design and is more likely to meet nonlinear constraints when re-evaluated on a very fine mesh. The number of design iterations appears to be unaffected by the order of accuracy, with reduced robustness causing occasional analysis failures. In several cases it has been shown that the third-order method costs nearly the same as the second-order method on a given grid, giving improved accuracy for almost no additional CPU cost.

Acknowledgements

Firstly, I'd like to thank Professor Zingg for providing me the opportunity to undertake this exciting and challenging project. I have learned more than I could have expected and improved my confidence and ability as an engineer throughout this project. You have taught me more than just technical understanding through your professional demeanor, motivation for perfection, and attention to the finest details.

I am grateful for the insight and comments provided by Prof. Ekmekci, Prof. Lavoie, Prof. Nair, and Prof. Steeves during my Research Assessment Committee meetings throughout my research.

I would like to thank all of my colleagues in the CFD lab for many insightful discussions; I have learned much more than is contained within this document as a result. I extend a special thanks to David for your patience in teaching me the fundamentals of summation-by-parts operators and other advanced topics. Another special thanks to Shahriar for your help with many of the details surrounding the span optimization test case studied in this thesis. Thank you to all of my fellow executives in the Aerospace Students' Association for your friendship and dedication to improving the student experience at UTIAS.

I am appreciative of the support provided by my family through my education. Thank you to Stacey for your help with early revisions of this document; your love and support has created many great memories during my time at UTIAS.

Financial support from the Kenneth M. Molson Foundation and the University of Toronto is gratefully acknowledged. Computations were performed on the GPC supercomputer at the SciNet HPC Consortium. SciNet is funded by: the Canada Foundation for Innovation under the auspices of Compute Canada; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

Nicholas Albert Lee Holt

University of Toronto Institute for Aerospace Studies
September 29, 2014

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Alphanumeric

$\mathcal{L}$	Lagrangian function
$\mathcal{J}$	objective function
ξ, η, ζ	curvilinear coordinates
ξ_x,ξ_y,ξ_z,η_x,η_y,η_z,ζ_x,ζ_y,ζ_z	grid metrics of the curvilinear coordinate transformation
a	sound speed
c	mean aerodynamic chord
e	total energy
I_x,I_y,I_z	metric invariants of the curvilinear coordinate transformation
J	Jacobian of the curvilinear coordinate transformation
p	pressure
Pr	Prandtl number, $Pr = 0.72$ for air
Re	Reynolds number
U,V,W	contravariant velocity components in the x-,y-, and z-direction respectively
u,v,w	Cartesian velocity components in the x-,y-, and z-direction respectively
x,y,z	physical space coordinates
$\boldsymbol{\mathcal{M}}^{(i)}$	vector of mesh movement residual at increment i
$\hat{\mathbf{E}}$,$\hat{\mathbf{F}}$,$\hat{\mathbf{G}}$	vector of transformed inviscid fluxes, in the ξ-,η-, and ζ-direction respectively
$\hat{\mathbf{E}}_v$,$\hat{\mathbf{F}}_v$,$\hat{\mathbf{G}}_v$	vector of transformed viscous fluxes, in the ξ-,η-, and ζ-direction respectively
$\hat{\mathbf{Q}}$	vector of transformed solution variables
$\boldsymbol{\lambda}$	vector of mesh adjoint variables
$\boldsymbol{\psi}$	vector of flow adjoint variables
$\mathbf{b}$	vector of B-spline control points

$\mathbf{E_v}$,$\mathbf{F_v}$,$\mathbf{G_v}$	vector of viscous fluxes, in the x-,y-, and z-direction respectively
$\mathbf{E}$,$\mathbf{F}$,$\mathbf{G}$	vector of inviscid fluxes, in the x-,y-, and z-direction respectively
$\mathbf{Q_v}$	vector of viscous variables
$\mathbf{Q}$	vector of solution variables
$\mathbf{v}$	vector of design variables
$\boldsymbol{\mathcal{R}}$	vector of the discrete flow residual

Greek

γ	specific heat ratio; $\gamma = 1.4$ for air
μ	molecular viscosity
ρ	density
τ	viscous stress components

Abbreviations

ABTE	Average Boundary Truncation Error
ASO	Aerodynamic Shape Optimization
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy number
FD	Finite-Difference
FGMRES	Flexible Generalized Minimal RESidual
GCROT	Generalized Conjugate Residual with inner-Orthogonalization, Truncated
GMRES	Generalized Minimal RESidual
IATA	International Air Transport Association
KKT	Karush-Kuhn-Tucker conditions
PDE	Partial Differential Equation
RANS	Reynolds-Averaged Navier-Stokes
SAT	Simultaneous Approximation Term
SBP	Summation By Parts
SNOPT	Sparse Nonlinear OPTimizer

Chapter 1

Introduction

1.1 Motivation

The aerospace industry is highly competitive, resulting in tight profit margins. Rising fuel costs have serious implications on the economic health of the aviation industry. Carbon dioxide (CO_2) emissions and their contribution to climate change provide further incentive to reduce fuel consumption in next generation aircraft. With aviation traffic expected to increase exponentially in the years to come, with just over a 5% growth in passenger traffic and a 1% growth in cargo traffic confirmed for 2013 relative to 2012 [44, 45], simply reducing air traffic is not an option.

To reinforce the environmental concerns of emissions within the aviation industry, the International Air Transport Association (IATA) has introduced the following aggressive emissions targets [42]:

- a cap on net aviation CO_2 emissions from 2020 (carbon-neutral growth);
- an average improvement in fuel efficiency of 1.5% per year from 2009 to 2020; and
- a reduction in net aviation CO_2 emissions of 50% by 2050, relative to 2005 levels.

The motivation of these targets is to reduce the effect that aviation has on climate change. In 2012, air transportation produced 689 million tonnes of CO_2, accounting for 2% of global anthropogenic CO_2 emissions [43]. IATA outlines the four major changes required to meet these aggressive targets:

- improved technology, including the deployment of sustainable low-carbon fuels;
- more efficient aircraft operations;
- infrastructure improvements, including modernized air traffic management systems;
- market-based measures to fill the remaining emissions gap.

Figure 1.1 shows the projected CO_2 emissions under several technology scenarios, highlighting that contributions from many fields of research will be required to achieve environmental sustainability in aviation.

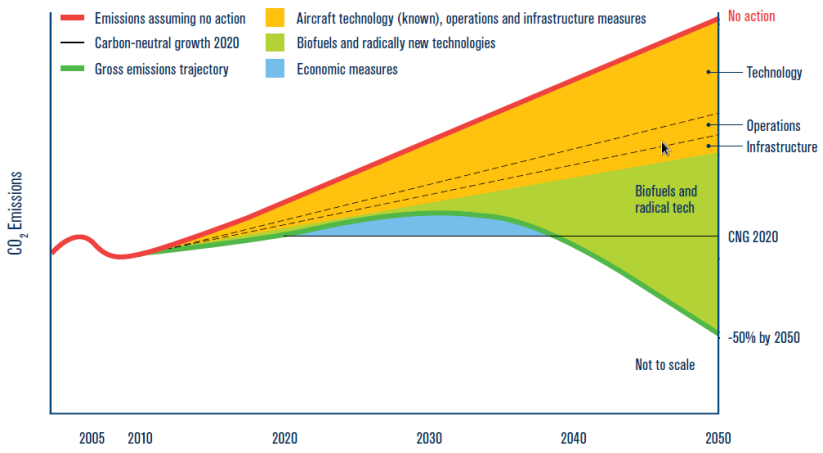


Figure 1.1: CO_2 emissions forecast under several technology scenarios. [43]

Observing from a more fundamental standpoint, the Breguet range equation suggests several ways that an aircraft can be made more efficient:

$$R = V \times \underbrace{\left(\frac{L}{D}\right)}_{\text{aerodynamic}} \times \underbrace{I_{\text{sp}}}_{\text{propulsion}} \times \underbrace{\ln\left(\frac{W_i}{W_f}\right)}_{\text{structural}}. \tag{1.1}$$

where R is the range of the aircraft, V is the aircraft's cruise speed, $\frac{L}{D}$ represents the lift-to-drag ratio, I_{sp} is the propulsive efficiency, and the initial and final weights of the aircraft are W_i and W_f respectively. Three engineering challenges are contained within Equation 1.1: engine efficiency, structural weight, and aerodynamics. While all aspects are important, this work considers the aerodynamic effects contained within the lift-to-drag ratio, and how an airframe can be modified to improve aerodynamics.

1.2 Aerodynamic Shape Optimization

Aerodynamic Shape Optimization (ASO) is an iterative procedure used to discover, with some degree of autonomy, the ideal aerodynamic configurations of aircraft. A gradient-based ASO algorithm is shown schematically in Figure 1.2. ASO can be used to optimize an existing aerodynamic shape to reduce drag through incremental shape changes, although the true power of ASO will be seen through exploratory shape optimization. Starting with unconventional geometries and defining mission and objective criteria appropriately, while enabling the geometry to change freely, can lead to large improvements in fuel efficiency. Numerical optimization will be important for these design problems as it may be misleading for aircraft designers to draw from conventional experience when designing unconventional aircraft, and ASO can provide the designers with an automated and comprehensive intuition for the design space. Developing a reliable optimizer for aerodynamic applications is a first step for unconventional aircraft design. The next step is making the optimizer as efficient as possible through algorithm improvements.

ASO was first introduced by Hicks *et al.* [40] in the optimization of two-dimensional airfoils. Hicks *et al.* performed the gradient computation using the finite-difference method. The use of the finite-difference method requires the careful selection of the perturbation parameter to balance truncation and

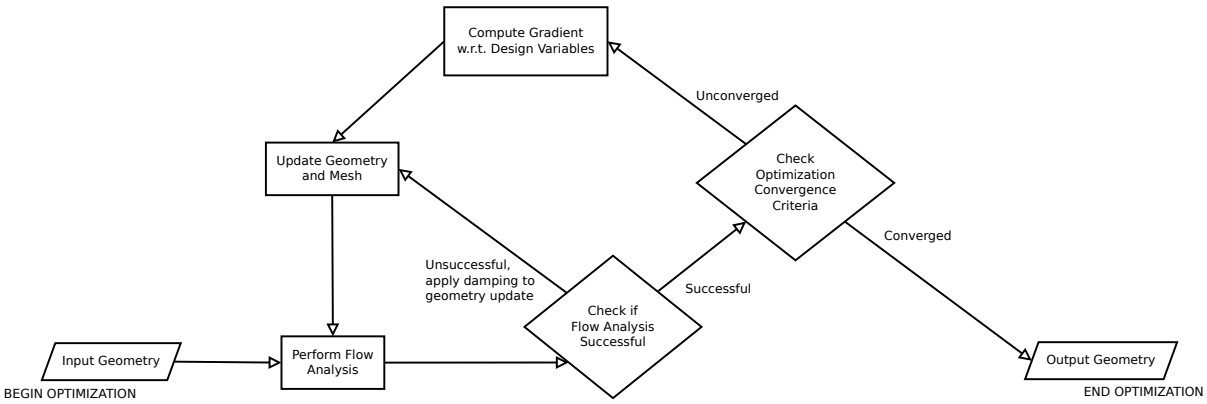


Figure 1.2: Schematic diagram outlining the steps involved in a gradient-based aerodynamic shape optimization algorithm.

subtractive cancellation error and its cost is dependent on the number of design variables. Jameson introduced the adjoint approach to computing the gradient [46], which avoids the step size dilemma while requiring the solution to a system of linear equations. The main advantage to this approach is that its computational cost is nearly independent of the number of design variables. The adjoint method applied to gradient-based ASO leads to an efficient algorithm. Further improvements can be made to the algorithm by improving the efficiency of the flow analysis. The use of high-order methods is one strategy to achieve this goal.

1.3 High-Order Methods

There are three prevalent techniques to reduce numerical error in a simulation. Increasing the number of nodes in the computational mesh is one strategy; this improves the spatial resolution to resolve finer features in the flow solution. Careful arrangement of the mesh nodes may also result in a reduction in error, i.e., accurately resolving boundary layers; this technique requires more information about the converged flow analysis in order to be effective and is not directly studied in this work. Adjoint-based adaptive mesh refinement is one strategy for intelligent node placement [65], and could be considered in future projects. An efficient strategy to reduce error, as shown by several authors [4, 17, 18, 25, 37, 39, 63], is to reduce the Taylor series truncation error with the use of high-order methods. This is done by increasing the order of accuracy of the derivative approximations in a finite-difference method, and could also be achieved in polynomial reconstruction based methods by increasing the degree of the polynomial used in the reconstruction of the solution data. Results from the 1st International Workshop on High-Order Computational Fluid Dynamics (CFD) Methods demonstrated definitive efficiency improvements for two-dimensional flows with the use of high-order methods, with such trends expected to transfer to three-dimensional flows with the introduction of the appropriate test cases in the future [87].

The efficiency improvements seen with high-order methods might seem counter-intuitive at first glance as it is almost certain that using high-order methods on a given grid will increase computational cost. However, the reduced error when using high-order methods indicates a potential for efficiency improvements. High-order methods can achieve a specified accuracy (such as accurately predicting drag within 0.1%) on a coarser mesh than the second-order method, and have demonstrated increased efficiency when measuring error per computational cost [18, 25, 63, 87]. While high-order has been shown

to be effective for numerical analysis alone, the trade-off relationship between computational resources and accuracy are presently unclear in the context of optimization since the gradient computation required by the optimization algorithm must now be considered. Literature review reveals only one example of optimization used in conjunction with high-order flow analysis [7, 8, 9, 10]. This specific example of two-dimensional ASO demonstrated a reduction in the optimization iterations required when high-order methods are used, with similar CPU costs for the second- and fourth-order finite volume methods and resulted in similar optimized shapes. Since optimization with high-order analysis is a relatively unexplored topic, it is important to continue to test the benefits of high-order spatial discretization as applied to ASO, as well as extend the efficiency analysis to the optimization of three-dimensional shapes.

Finite-difference operators which satisfy the summation-by-parts (SBP) property are used within the analysis algorithm to discretize the derivative operators. First introduced by Kreiss and Scherer [50], the SBP property ensures that stability is guaranteed for linear partial differential equations (PDEs) when used in conjunction with an appropriate boundary condition, which is a necessary but not sufficient condition for stability in nonlinear PDEs [21, 26, 51]. Since a curvilinear coordinate transformation is used in this work, the diagonal-norm SBP operators are chosen, as full and restricted-norm SBP operators cannot guarantee stability for the linear convection diffusion equation with a general curvilinear coordinate transformation [26, 67, 68, 81]. An unfortunate consequence to using diagonal-norm SBP operators is a reduced order of accuracy compared to the a full-norm operator of the same size. Mattson and Almquist have introduced techniques to stabilize the solution algorithm when using alternative norm matrices such as the full norm [56]. Boundary conditions are weakly enforced using a penalty method to preserve the SBP property and maintain stability.

1.4 Benchmark Test Cases

Several standard tests exist for numerical aerodynamic analysis codes to be compared, as well as experimental data for specific geometries. Standardized tests are less mature in the numerical optimization community and experimental data does not exist for the highly specific geometries obtained by the optimization algorithm. This deficiency is currently being addressed through the introduction of four standard tests which the ASO community is committed to studying regularly [30, 61, 62, 71]; some of these have been studied recently [12, 83]. The efficiency of the high-order algorithm will be evaluated by examining optimization cases which have similarities to some of these standardized shape optimization test cases.

1.5 Thesis Objectives

The objective of this thesis project is to incorporate an existing high-order finite-difference flow analysis algorithm with an existing gradient-based optimization algorithm. Most of the development work will involve adding high-order terms into the gradient computation; specifically the adjoint equations will be directly modified. The large memory requirements of storing the high-order flux Jacobian matrix will be addressed by developing an analytical transposed Jacobian-vector product strategy; the advantages and disadvantages of this method will be examined. It has been shown by several researchers that high-order methods have the potential to improve the efficiency of analysis algorithms. The flow analysis and the solution to the adjoint equations represent the most expensive components of the optimization

algorithm. It is an objective of this thesis to demonstrate that the efficiency gains observed in the analysis carry through to the optimization algorithm as a whole since it is presently unclear how the solution to the adjoint equations will be affected by the increased bandwidth caused by the inclusion of high-order terms.

Chapter 2

Numerical Solution to the Navier-Stokes Equations

ULTIMATELY, this work is concerned with the optimization of aerodynamic shapes, which will require repeated aerodynamic analysis using CFD. CFD can be applied with various levels of fidelity, driven by the underlying assumptions of the problem. This work considers both inviscid and viscous laminar aerodynamic analysis and optimization. Inviscid flow is a valid assumption for very high speed external flows where viscous effects are often negligible. The viscous laminar assumption is an important distinction, as the presented governing equations will be valid only for low Reynolds number flows to ensure that a steady flow is observed. The Reynolds-Averaged Navier-Stokes (RANS) equations are not modelled in this work, but allow for analysis of time-averaged turbulent flows at higher Reynolds numbers.

The finite-difference steady flow analysis tool used in this work was first introduced as a second-order inviscid flow solver by Hicken [29] and then extended to high-order by Dias [24]. Further work was done by Osusky [72] to include viscous and turbulent effects using second-order spatial discretization. The viscous laminar flow solver was extended to high-order by Del Rey Fernández and Zingg [22]. Improvements to the high-order viscous laminar flow solver have been included in the scope of this thesis. Analysis is possible using second-,third-, and fourth-order spatial discretization with this framework.

This chapter will first present the governing equations in a Cartesian coordinate system. A curvilinear coordinate transformation will subsequently be introduced, and the transformed governing equations will be presented, with limitations discussed. The flow solution algorithm will be presented, including the spatial discretization using SBP operators and SATs, artificial dissipation operators, and globalization strategies used for robust analysis.

2.1 Navier-Stokes Equations

2.1.1 Cartesian Coordinates

The three-dimensional Euler equations describe continuum, inviscid, compressible flow. The validity of the inviscid assumption relies on the Reynolds number, Re, being large. Euler analysis allows for accurate evaluation of induced drag and wave drag under conditions where the flow remains fully attached. To

account for frictional forces on the aerodynamic surface, viscous effects must be modelled with the Navier-Stokes equations. The compressible Navier-Stokes equations are given by

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \frac{1}{Re}\left(\frac{\partial \mathbf{E_v}}{\partial x} + \frac{\partial \mathbf{F_v}}{\partial y} + \frac{\partial \mathbf{G_v}}{\partial z}\right), \tag{2.1}$$

where x,y,z are the Cartesian coordinates and Re is the Reynolds number. The Navier-Stokes equations operate on the non-dimensional conservative variables,

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \tag{2.2}$$

which represent mass, momentum in the x-,y-, and z-direction, and total energy per unit volume. The inviscid flux vectors on the left-hand side of Equation 2.1 are defined as

$$\mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{bmatrix}, \tag{2.3}$$

and the viscous flux vectors on the right-hand side of Equation 2.1 are defined as

$$\mathbf{E_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \quad \mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix}. \tag{2.4}$$

The viscous stress and heat conduction terms are given by

$$\begin{aligned}
\tau_{xx} &= \frac{4}{3}\mu u_x - \frac{2}{3}\mu(v_y + w_z), \\
\tau_{yy} &= \frac{4}{3}\mu v_y - \frac{2}{3}\mu(u_x + w_z), \\
\tau_{zz} &= \frac{4}{3}\mu w_z - \frac{2}{3}\mu(u_x + v_y), \\
\tau_{xy} &= \mu(u_y + v_x), \\
\tau_{xz} &= \mu(u_z + w_x), \\
\tau_{yz} &= \mu(v_z + w_y), \\
\tau_{yx} &= \tau_{xy},\ \tau_{zx} = \tau_{xz},\ \tau_{zy} = \tau_{yz},
\end{aligned}$$

$$\begin{aligned}
E_{v,5} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \mu Pr^{-1}(\gamma-1)^{-1}\partial_x(a^2), \\
F_{v,5} &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + \mu Pr^{-1}(\gamma-1)^{-1}\partial_y(a^2), \\
G_{v,5} &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + \mu Pr^{-1}(\gamma-1)^{-1}\partial_z(a^2).
\end{aligned}$$

The ∂_x term is used as a shorthand for $\frac{\partial}{\partial x}$, u_x is used as shorthand for $\frac{\partial u}{\partial x}$, and so on. In the above, ρ is the density, a is the sound speed ($a = \sqrt{\gamma p/\rho}$), u,v,w are the Cartesian velocity components, e is

the total energy per unit volume, μ is the molecular viscosity, and the specific heat ratio, γ, for air is 1.4. The laminar Prandtl number is set as a constant of $Pr = 0.72$. Molecular viscosity is calculated using the non-dimensional form of Sutherland's law [88],

$$\mu = \frac{a^3(1 + \frac{S^*}{T_\infty})}{a^2 + \frac{S^*}{T_\infty}}, \tag{2.5}$$

where S^* is the Sutherland constant ($198.6^\circ R$), and T_∞ is the free-stream temperature, typically set to $460^\circ R$ for air.

One additional equation is required to form a closed set; the equation of state for an ideal gas is used to define the pressure, p, as

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right). \tag{2.6}$$

The Euler equations can be obtained in the limit as $Re \to \infty$. The Reynolds number, Re, is a ratio of the inertial forces to viscous forces, and is given by

$$Re = \frac{\rho_\infty a_\infty c}{\mu_\infty}. \tag{2.7}$$

Non-dimensional quantities are used in Equation 2.1 and are defined as [75]:

$$t = \frac{\tilde{t}\tilde{a}_\infty}{c}, \quad x = \frac{\tilde{x}}{c}, \quad y = \frac{\tilde{y}}{c}, \quad z = \frac{\tilde{z}}{c}, \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty}, \quad u_i = \frac{\tilde{u}_i}{\tilde{a}_\infty}, \quad e = \frac{\tilde{e}}{\tilde{\rho}_\infty \tilde{a}_\infty^2}, \quad \mu = \frac{\tilde{\mu}}{\tilde{\mu}_\infty},$$

where c is the reference length (typically the mean aerodynamic chord). The '∞' subscript denotes a free-stream value for the given quantity, while the '$\tilde{}$' denotes a dimensional quantity.

2.1.2 Curvilinear Coordinate Transformation

This finite-difference solution algorithm makes use of a general curvilinear coordinate transformation to transform from physical space to computational space. An important property of computational space is that $\Delta\xi = \Delta\eta = \Delta\zeta = 1$. This intuitively reads that all of the coordinates in computational space are indexed by integer values with unity increments. Explicitly, the curvilinear coordinates ξ, η, ζ are functions of the Cartesian coordinates x, y, z:

$$\begin{aligned} \xi &= \xi(x, y, z), \\ \eta &= \eta(x, y, z), \\ \zeta &= \zeta(x, y, z). \end{aligned}$$

In this work it is assumed that the grid is stationary and the coordinate transformation has no dependence on time. The grid metrics of the transformation are as follows:

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta), \quad & \eta_x &= J(z_\xi y_\zeta - y_\xi z_\zeta), \quad & \zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta), \\ \xi_y &= J(z_\eta x_\zeta - z_\zeta x_\eta), \quad & \eta_y &= J(x_\xi z_\zeta - z_\xi x_\zeta), \quad & \zeta_y &= J(z_\xi x_\eta - x_\xi z_\eta), \\ \xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta), \quad & \eta_z &= J(y_\xi x_\zeta - x_\xi y_\zeta), \quad & \zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta). \end{aligned} \tag{2.8}$$

The metric Jacobian is defined to be the determinant of the metric Jacobian matrix of the transformation:

$$\begin{aligned} J^{-1} &= \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \\ &= x_\xi y_\eta z_\zeta + y_\xi z_\eta x_\zeta + z_\xi x_\eta y_\zeta - x_\xi z_\eta y_\zeta - y_\xi x_\eta z_\zeta - z_\xi y_\eta x_\zeta \end{aligned} \tag{2.9}$$

The curvilinear coordinate transformation is implicitly described by the grid metrics and the metric Jacobian.

Applying the coordinate transformation to Equation 2.1, multiplying both sides by J^{-1}, and rearranging into strong conservative form, the Navier-Stokes equations can be rewritten as

$$\partial_t \hat{\mathbf{Q}} + \partial_\xi \hat{\mathbf{E}} + \partial_\eta \hat{\mathbf{F}} + \partial_\zeta \hat{\mathbf{G}} = \frac{1}{Re}\left(\partial_\xi \hat{\mathbf{E}}_\mathbf{v} + \partial_\eta \hat{\mathbf{F}}_\mathbf{v} + \partial_\zeta \hat{\mathbf{G}}_\mathbf{v}\right), \tag{2.10}$$

where the transformed variables and fluxes are related to their Cartesian counterparts by

$$\begin{aligned} \hat{\mathbf{Q}} &= J^{-1}\mathbf{Q}, \\ \hat{\mathbf{E}} &= J^{-1}\left(\xi_x \mathbf{E} + \xi_y \mathbf{F} + \xi_z \mathbf{G}\right), \quad & \hat{\mathbf{E}}_\mathbf{v} &= J^{-1}\left(\xi_x \mathbf{E}_\mathbf{v} + \xi_y \mathbf{F}_\mathbf{v} + \xi_z \mathbf{G}_\mathbf{v}\right), \\ \hat{\mathbf{F}} &= J^{-1}\left(\eta_x \mathbf{E} + \eta_y \mathbf{F} + \eta_z \mathbf{G}\right), \quad & \hat{\mathbf{F}}_\mathbf{v} &= J^{-1}\left(\eta_x \mathbf{E}_\mathbf{v} + \eta_y \mathbf{F}_\mathbf{v} + \eta_z \mathbf{G}_\mathbf{v}\right), \\ \hat{\mathbf{G}} &= J^{-1}\left(\zeta_x \mathbf{E} + \zeta_y \mathbf{F} + \zeta_z \mathbf{G}\right), \quad & \hat{\mathbf{G}}_\mathbf{v} &= J^{-1}\left(\zeta_x \mathbf{E}_\mathbf{v} + \zeta_y \mathbf{F}_\mathbf{v} + \zeta_z \mathbf{G}_\mathbf{v}\right). \end{aligned}$$

The transformed inviscid flux vectors can be simplified to

$$\hat{\mathbf{E}} = \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U(e+p) \end{bmatrix}, \quad \hat{\mathbf{F}} = \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V(e+p) \end{bmatrix}, \quad \hat{\mathbf{G}} = \begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W(e+p) \end{bmatrix} \tag{2.11}$$

where

$$U = \xi_x u + \xi_y v + \xi_z w, \tag{2.12}$$

$$V = \eta_x u + \eta_y v + \eta_z w, \tag{2.13}$$

$$W = \zeta_x u + \zeta_y v + \zeta_z w, \tag{2.14}$$

are the contravariant velocity components. The transformed viscous stress and heat conduction terms are given by

$$\begin{aligned} \tau_{xx} &= \frac{4}{3}\mu(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta) - \frac{2}{3}\mu(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta), \\ \tau_{yy} &= \frac{4}{3}\mu(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta) - \frac{2}{3}\mu(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta), \\ \tau_{zz} &= \mu(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta) - \frac{2}{3}\mu(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta), \end{aligned}$$

$$
\begin{aligned}
\tau_{xy} &= \mu(\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta + \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta),\\
\tau_{xz} &= \mu(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta + \xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta),\\
\tau_{yz} &= \mu(\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta + \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta),\\
\tau_{yx} &= \tau_{xy},\ \tau_{zx} = \tau_{xz},\ \tau_{zy} = \tau_{yz},\\
E_{v,5} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \mu Pr^{-1}(\gamma-1)^{-1}[\xi_x\partial_\xi(a^2) + \eta_x\partial_\eta(a^2) + \zeta_x\partial_\zeta(a^2)],\\
F_{v,5} &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + \mu Pr^{-1}(\gamma-1)^{-1}[\xi_y\partial_\xi(a^2) + \eta_y\partial_\eta(a^2) + \zeta_y\partial_\zeta(a^2)],\\
G_{v,5} &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + \mu Pr^{-1}(\gamma-1)^{-1}[\xi_z\partial_\xi(a^2) + \eta_z\partial_\eta(a^2) + \zeta_z\partial_\zeta(a^2)].
\end{aligned}
$$

The format of these equations can be more easily understood by rewriting the viscous fluxes with a more concise notation,

$$
\begin{aligned}
\hat{\mathbf{E}}_\mathbf{v} &= \begin{bmatrix} 0 & 0 & 0 & 0\\ (\alpha_{\xi,\xi})_{2,1} & (\alpha_{\xi,\xi})_{2,2} & (\alpha_{\xi,\xi})_{2,3} & 0\\ (\alpha_{\xi,\xi})_{3,1} & (\alpha_{\xi,\xi})_{3,2} & (\alpha_{\xi,\xi})_{3,3} & 0\\ (\alpha_{\xi,\xi})_{4,1} & (\alpha_{\xi,\xi})_{4,2} & (\alpha_{\xi,\xi})_{4,3} & (\alpha_{\xi,\xi})_{4,4}\end{bmatrix}\partial_\xi\mathbf{Q}_\mathbf{v} + \begin{bmatrix} 0 & 0 & 0 & 0\\ (\alpha_{\xi,\eta})_{2,1} & (\alpha_{\xi,\eta})_{2,2} & (\alpha_{\xi,\eta})_{2,3} & 0\\ (\alpha_{\xi,\eta})_{3,1} & (\alpha_{\xi,\eta})_{3,2} & (\alpha_{\xi,\eta})_{3,3} & 0\\ (\alpha_{\xi,\eta})_{4,1} & (\alpha_{\xi,\eta})_{4,2} & (\alpha_{\xi,\eta})_{4,3} & (\alpha_{\xi,\eta})_{4,4}\end{bmatrix}\partial_\eta\mathbf{Q}_\mathbf{v}\\
&\quad + \begin{bmatrix} 0 & 0 & 0 & 0\\ (\alpha_{\xi,\zeta})_{2,1} & (\alpha_{\xi,\zeta})_{2,2} & (\alpha_{\xi,\zeta})_{2,3} & 0\\ (\alpha_{\xi,\zeta})_{3,1} & (\alpha_{\xi,\zeta})_{3,2} & (\alpha_{\xi,\zeta})_{3,3} & 0\\ (\alpha_{\xi,\zeta})_{4,1} & (\alpha_{\xi,\zeta})_{4,2} & (\alpha_{\xi,\zeta})_{4,3} & (\alpha_{\xi,\zeta})_{4,4}\end{bmatrix}\partial_\zeta\mathbf{Q}_\mathbf{v}\\
&= A_{\xi,\xi}\partial_\xi\mathbf{Q}_\mathbf{v} + A_{\xi,\eta}\partial_\eta\mathbf{Q}_\mathbf{v} + A_{\xi,\zeta}\partial_\zeta\mathbf{Q}_\mathbf{v}. \qquad (2.15)
\end{aligned}
$$

where $\mathbf{Q}_\mathbf{v} = [u, v, w, a^2]^T$. Similar expressions can be derived for the other flux components:

$$
\begin{aligned}
\hat{\mathbf{F}}_\mathbf{v} &= A_{\eta,\xi}\partial_\xi\mathbf{Q}_\mathbf{v} + A_{\eta,\eta}\partial_\eta\mathbf{Q}_\mathbf{v} + A_{\eta,\zeta}\partial_\zeta\mathbf{Q}_\mathbf{v}, \qquad (2.16)\\
\hat{\mathbf{G}}_\mathbf{v} &= A_{\zeta,\xi}\partial_\xi\mathbf{Q}_\mathbf{v} + A_{\zeta,\eta}\partial_\eta\mathbf{Q}_\mathbf{v} + A_{\zeta,\zeta}\partial_\zeta\mathbf{Q}_\mathbf{v}. \qquad (2.17)
\end{aligned}
$$

Determining the α terms is a straightforward algebraic exercise. Each α term groups together quantities which depend on the flow variables and grid metrics. Putting the viscous flux vectors into the form of Equation 2.15 is beneficial when undertaking the exercise of analytically differentiating the governing equations, necessary to compute the gradient as part of the optimization algorithm. It is also important to remember when differentiating the viscous fluxes that the A matrices have an explicit dependence on $\mathbf{Q}$ through viscosity, μ, and the Cartesian velocity components: u, v, w; as well as an explicit dependence on the grid metrics.

As a result of applying the coordinate transformation and reforming the governing equations into strong conservative form, metric invariants must be satisfied:

$$
\begin{aligned}
I_x = (J^{-1}\xi_x)_\xi + (J^{-1}\eta_x)_\eta + (J^{-1}\zeta_x)_\zeta &= 0, \qquad (2.18)\\
I_y = (J^{-1}\xi_y)_\xi + (J^{-1}\eta_y)_\eta + (J^{-1}\zeta_y)_\zeta &= 0, \qquad (2.19)\\
I_z = (J^{-1}\xi_z)_\xi + (J^{-1}\eta_z)_\eta + (J^{-1}\zeta_z)_\zeta &= 0. \qquad (2.20)
\end{aligned}
$$

If the grid metrics are known analytically and derivatives are performed analytically in Equations 2.18–

2.20 then the metric invariants are automatically satisfied. With the introduction of the discrete derivative operators, it will be shown that the metric invariants are not guaranteed to be satisfied unless a particular metric computation is employed.

2.2 Flow Solution Algorithm

The continuous governing equations presented in Equation 2.10 are discretized and applied on a discrete set of grid coordinates using high-order finite-difference SBP derivative operators with boundary conditions enforced by SATs. The FD-SBP-SAT solution algorithm involves solving a nonlinear system of equations with several million degrees of freedom in an efficient manner. Each component of the analysis algorithm is discussed below in detail.

2.2.1 Spatial Discretization

The three-dimensional domain is discretized using structured multi-block grids. Multi-block grids enable the flow solver to operate on more complex geometries and allows for parallelization by assigning each sub-domain, consisting of one or many blocks, to a separate process. Each block is load balanced by the grid designer to evenly distribute the nodes across all blocks to improve the parallel scaling of the algorithm.

Summation-by-Parts Finite-Difference Operators for the First Derivative

Finite-difference operators which satisfy the SBP property are used within this solution algorithm to discretize the derivative operators. The SBP property ensures that stability is guaranteed for linear problems, which is a necessary but not sufficient condition for stability in nonlinear problems [51, 26, 21]. For a finite-difference operator, $\delta = H^{-1}\Theta$, with an inner product $(u, v)_H = u^T H v$, the SBP property is defined as

$$(u, \delta v)_H = -(\delta u, v)_H - u_0 v_0 + u_n v_n, \tag{2.21}$$

which is analogous to the integration by parts property. The subscripts 0 and n indicate the boundary values of u and v. The relation $\Theta + \Theta^T = E$ must be satisfied in order for the SBP property in Equation 2.21 to hold, where $E = \text{diag}(-1, 0, \ldots, 0, 1)$. H is called the norm matrix and is symmetric positive definite. In general, H is a block diagonal matrix,

$$H = \Delta\xi \begin{bmatrix} H_L & & \\ & I & \\ & & H_R \end{bmatrix}, \tag{2.22}$$

where the H_L and H_R are sub-matrices of H and I is the identity matrix. There are three popular structures for H_L and H_R. The full norm chooses H_L and H_R to be dense matrices. The restricted full norm takes the form $H_L = \begin{bmatrix} h_0 & \\ & \tilde{H}_L \end{bmatrix}$ and $H_R = \begin{bmatrix} \tilde{H}_R & \\ & h_0 \end{bmatrix}$, with $\tilde{H}_L$ and $\tilde{H}_R$ as dense matrices and h_0 a scalar. The diagonal-norm assumes $H_L = \text{diag}(h_1, h_2, \ldots, h_{2s-1}, h_{2s})$ and $H_R = \text{diag}(h_{2s}, h_{2s-1}, \ldots, h_2, h_1)$, where s is determined by the order of accuracy of the operator. To assist in stability, the diagonal-norm matrix is chosen for this work, as full and restricted norm ma-

trices cannot guarantee stability for the linear convection diffusion equation with a general curvilinear coordinate transformation [67, 68, 81, 26].

The SBP operators used in this work are shown in Appendix A, accompanied by a brief derivation. The interior scheme of the SBP operator is a $2s$-order accurate centred difference scheme. The boundary nodes of the SBP operator are of reduced order; the first and final $2s$ nodes use an s-order accurate one-sided or biased finite-difference scheme. The global order of accuracy of the SBP operator is $s+1$. The anti-symmetric nature of the interior scheme of the SBP operators introduces the need for added numerical dissipation. This can be accomplished by adding an artificial dissipation operator with a symmetric scheme on the interior, which will be discussed in more detail later in this section.

Introducing the discretized derivative operator, the governing equations can be represented in semi-discrete form,

$$\partial_t \hat{\mathbf{Q}} + \delta_\xi \hat{\mathbf{E}} + \delta_\eta \hat{\mathbf{F}} + \delta_\zeta \hat{\mathbf{G}} = \frac{1}{Re}\left(\delta_\xi \hat{\mathbf{E}}_\mathbf{v} + \delta_\eta \hat{\mathbf{F}}_\mathbf{v} + \delta_\zeta \hat{\mathbf{G}}_\mathbf{v}\right), \tag{2.23}$$

and the viscous fluxes must be modified to use the finite-difference derivative operator:

$$\hat{\mathbf{E}}_\mathbf{v} = A_{\xi,\xi}\delta_\xi \mathbf{Q}_\mathbf{v} + A_{\xi,\eta}\delta_\eta \mathbf{Q}_\mathbf{v} + A_{\xi,\zeta}\delta_\zeta \mathbf{Q}_\mathbf{v}, \tag{2.24}$$

$$\hat{\mathbf{F}}_\mathbf{v} = A_{\eta,\xi}\delta_\xi \mathbf{Q}_\mathbf{v} + A_{\eta,\eta}\delta_\eta \mathbf{Q}_\mathbf{v} + A_{\eta,\zeta}\delta_\zeta \mathbf{Q}_\mathbf{v}, \tag{2.25}$$

$$\hat{\mathbf{G}}_\mathbf{v} = A_{\zeta,\xi}\delta_\xi \mathbf{Q}_\mathbf{v} + A_{\zeta,\eta}\delta_\eta \mathbf{Q}_\mathbf{v} + A_{\zeta,\zeta}\delta_\zeta \mathbf{Q}_\mathbf{v}. \tag{2.26}$$

Summation-by-Parts Finite-Difference Operators for the Second Derivative

The second derivative can be approximated through application of the first derivative twice, known as the non-compact second-derivative operator. In the constant-coefficient case, the expansion of the matrix operator $\delta^{(2)} = \delta\delta$ will yield a second derivative operator with a larger bandwidth than is required. In general, variable coefficients will be required and the non-compact second derivative operator will take the form $\delta^{(2)}(B) = \delta B(\mathbf{Q})\delta$, where $B(\mathbf{Q})$ demonstrates an explicit dependence of the variable coefficient matrix $B = \mathrm{diag}(b_1, \ldots, b_N)$ on the solution variables $\mathbf{Q}$. A compact operator can be derived which reduces the bandwidth of $\delta^{(2)}$ and is related to the first-derivative operator by [21, 22, 57, 58]:

$$\delta^{(2)}(B) = H^{-1}\left(-\delta H B \delta - R + E B \delta_b\right), \tag{2.27}$$

where H corresponds to the diagonal norm in the first-derivative definition of δ, E is as defined previously, and R is a positive-semi-definite symmetric matrix which serves to reduce the bandwidth of the second-derivative operator. The boundary first-derivative operator, δ_b, is at least of order $s+1$ for a compact second-derivative operator. The choices $R = 0$ and $\delta_b = \delta$ correspond to the non-compact operator, which is the application of the first derivative twice. Del Rey Fernández *et al.* provide a more detailed description of how to construct R [21, 22].

The complicated structure of the compact second-derivative operator can be cumbersome to implement. A simplified structure of $\delta^{(2)}$ allows for a more intuitive construction of the variable coefficient matrix operator:

$$\delta^{(2)}(B) = \sum_i b_i M_i, \tag{2.28}$$

where b_i represents the variable coefficient at node i, and M_i is a sparse coefficient matrix unique to each node, with a repeating structure on the interior nodes of the operator; each of the M_i matrices are presented in Appendix A. The viscous flux residual contribution with the compact second-derivative

Table 2.1: L_2-norm of metric invariants comparing numerical and analytical metrics.

Invariant L_2-norm	**Second-order**	**Third-order**	**Fourth-order**
$\|\|(I_x)_{SBP}\|\|_2$	1.059E-13	4.575E-13	6.472E-13
$\|\|(I_y)_{SBP}\|\|_2$	1.146E-13	4.967E-13	6.536E-13
$\|\|(I_z)_{SBP}\|\|_2$	9.440E-14	3.871E-13	4.923E-13
$\|\|(I_x)_{map}\|\|_2$	4.221E-2	4.221E-2	4.223E-2
$\|\|(I_y)_{map}\|\|_2$	9.824E-2	1.169E-1	1.235E-1
$\|\|(I_z)_{map}\|\|_2$	1.005E-1	1.031E-1	1.031E-1

operator is used by replacing the double-derivative terms, in the ξ-direction for example, with

$$\begin{aligned}\delta_\xi A_{\xi,\xi}\delta_\xi \mathbf{Q_v} &\rightarrow \delta^{(2)}(A_{\xi,\xi})\mathbf{Q_v} \\ &= \sum_{i=1}^{N_\xi}(A_{\xi,\xi})_i M_i \mathbf{Q_v}.\end{aligned}$$

Grid Metrics

The grid metrics, such as ξ_x, are calculated using the same finite-difference SBP operator used to discretize fluxes. If grid metrics are not calculated using a conservative form, then the discrete form of the metric invariants shown in Equations 2.18–2.20 may not be satisfied. Such an oversight could lead to simple tests such as a uniform free stream flow failing to satisfy the Navier-Stokes equations.

As will be described in the next chapter, an analytical mapping is used in this work and the transformation from physical to computational space is known explicitly. It may be tempting to calculate the grid metrics and metric Jacobian analytically; however this is not recommended. It has been shown that computing metrics using finite differences leads to lower truncation error when compared to computing the metrics with a known analytical transformation [84, 86]. Use of the analytical grid metrics would also cause the discrete metric invariants to remain unsatisfied. Operator consistency is required between the grid metric computation and the flux derivatives.

The conservative form of the metrics used in this work is as follows:

$$\xi_x = \frac{J}{2}\left[(y_\eta z - y z_\eta)_\zeta + (y z_\zeta - y_\zeta z)_\eta\right]. \tag{2.29}$$

The averaging operation is present within Equation 2.29 because of the use of the two possible conservative forms of the grid metrics. Averaging conveys no preference over the two possible formulations. Similar expressions exist for the remaining grid metrics.

A simple numerical experiment is conducted to demonstrate that analytical metrics do not satisfy the metric invariants presented in Equations 2.18–2.20. The L_2-norm of the metric invariant equations is computed numerically for a specific mesh used in an optimization test within this thesis ($\boldsymbol{G}_1$ in the inviscid twist optimization study in Section 5.1). The grid metrics are computed using the SBP operators as shown in Equation 2.29 and compared to grid metrics which are computed analytically with the known B-spline mapping. Table 2.1 shows that the numerical grid metrics satisfy the discrete metric invariants to machine precision while the analytic grid metrics do not. An alternative to satisfying the metric

invariant equations would be to add the associated non-zero contribution to the residual vector. Adding additional terms through not satisfying the metric invariants will increase the time to compute the residual vector and will also introduce possible sources of programming error. This approach is not explored in this work, but may be an avenue to explore to increase robustness with the grid metric computation. It will be seen in the next chapter that the high-order algorithm presents grid generation challenges when numerical grid metrics are used.

Artificial Dissipation

Artificial dissipation serves several different purposes in the numerical flow analysis algorithm. Artificial dissipation used in this work is largely based off of the work of Mattsson *et al.* [59] and is similar to the operator used by Dias and Zingg [25] and Diener *et al.* [26]. A dissipation operator must have the following properties [59, 75]:

- couple nodes that have been decoupled by the centred-difference first-derivative operators;
- dissipate modes that are under-resolved due to mesh resolution;
- provide an energy estimate to prove stability for linear problems;
- adds a term of order greater than or equal to the order of accuracy of the discretization of the governing equations.

The interior scheme of the artificial dissipation operator is symmetric, and serves to stabilize the nonlinear solution algorithm. The dissipation operator is applied to the untransformed conservative variables. in the ξ-direction is as follows:

$$\delta_{\mathrm{AD},\xi}\mathbf{Q} = H_\xi^{-1} D_{d,\xi}^T J^{-1} B(\hat{\mathbf{Q}}) D_{d,\xi}\mathbf{Q} \tag{2.30}$$

where H_ξ is the diagonal-norm used to construct the first-derivative operator, $B(\hat{\mathbf{Q}})$ is a variable coefficient that dictates how much dissipation is to be added and is dependant on the solution, $\hat{\mathbf{Q}}$, and $D_{d,\xi}$ is a first-order undivided finite-difference approximation to the d-th derivative. Similar expressions exist for the dissipation contributions in the η- and ζ-directions. The matrix form of the undivided difference operators used are:

$$D_3 = \begin{bmatrix} -1 & 3 & -3 & 1 & & & \\ -1 & 3 & -3 & 1 & & & \\ & -1 & 3 & -3 & 1 & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 3 & -3 & 1 & \\ & & & -1 & 3 & -3 & 1 \\ & & & -1 & 3 & -3 & 1 \end{bmatrix}, \tag{2.31}$$

and

$$D_4 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 & & & & \\ 1 & -4 & 6 & -4 & 1 & & & & \\ 1 & -4 & 6 & -4 & 1 & & & & \\ & 1 & -4 & 6 & -4 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & -4 & 6 & -4 & 1 & \\ & & & & 1 & -4 & 6 & -4 & 1 \\ & & & & 1 & -4 & 6 & -4 & 1 \\ & & & & 1 & -4 & 6 & -4 & 1 \end{bmatrix}. \tag{2.32}$$

The D_3 operator is used for the second- and third-order algorithm and is applied to the half nodes, while D_4 is used for the fourth-order algorithm and is applied to the nodes directly. The variable coefficients are computed using matrix dissipation [82] for its proven error reduction capabilities [2] as follows:

$$B(\hat{\mathbf{Q}}) = \epsilon |A(\hat{\mathbf{Q}})|, \tag{2.33}$$

where $|A| = X\Lambda X^{-1}$ is a block diagonal matrix, ϵ dictates the amount of dissipation to be added, X is the eigenvector matrix of the flux Jacobian, and $|\Lambda| = \text{diag}(|\lambda_1|, |\lambda_2|, |\lambda_3|, |\lambda_3|, |\lambda_3|)$ where

$$\begin{aligned} \lambda_1 &= U + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}, & |\lambda_1| &= \max(\lambda_1, V_n\rho), \\ \lambda_2 &= U - a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}, & |\lambda_2| &= \max(\lambda_2, V_n\rho), \\ \lambda_3 &= U, & |\lambda_3| &= \max(\lambda_3, V_l\rho), \\ & & \rho &= |U| + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}. \end{aligned}$$

The quantities V_n and V_l serve as a lower limit to the numerical viscosity, and typically range from 0 to 0.35. The one-dimensional operator presented in Equation 2.30 can be extended to three dimensions through the use of Kronecker products [24, 29]. A scalar dissipation model, which applies the same amount of dissipation to each governing equation, provides a more stable analysis algorithm, however it is overly dissipative and is not used in this research.

This work only considers subsonic flows but if transonic flows were to be considered, a shock capturing scheme such as that of Jameson *et al.* [47] could be used to resolve shock waves in a stable way. This has been implemented in several instances [34, 74, 75], with Dias and Zingg applying it to a high-order inviscid transonic flow analysis algorithm [25].

Boundary and Interface Conditions

Boundary and interface conditions are enforced weakly using a penalty method to preserve the SBP property [21]. A penalty of the form $\alpha_{SAT}(X - X_{targ})$ is added to the residual vector on the boundary or interface nodes, where X has some dependence on the solution, X_{targ} is the target value, and α_{SAT} is chosen for stability.

Table 2.2 summarizes the various SATs used within this work. The viscous SATs are used in addition to the inviscid SATs when the Navier-Stokes equations are solved. The presented boundary conditions have been implemented by Hicken [29], Dias [24], and Osusky [72]. Interface conditions for inviscid

Table 2.2: SAT penalty terms for various boundary and interface types.

Inviscid SATs	α_{SAT}	**X**	$\mathbf{X}_{targ}$
wall/symmetry	$-H_b^{-1}J^{-1}A_\xi^+$	**Q**	$\mathbf{Q}_{V_t}$
far-field			$\mathbf{Q}_{\text{ff}}$
interface			$\mathbf{Q}_2$

Viscous SATs	α_{SAT}	**X**	$\mathbf{X}_{targ}$
wall	$\frac{H_b^{-1}\sigma^W}{Re}$	**Q**	$\mathbf{Q}_w$
	$\frac{H_b^{-1}\sigma^V}{Re}$	$\mathbf{E}_v$	$(\mathbf{E}_v)_w$
symmetry	$\frac{H_b^{-1}\sigma^W}{Re}$	**Q**	$\mathbf{Q}_{j+1,k,m}$
far-field	$\frac{H_b^{-1}\sigma^V}{Re}$	$\mathbf{E}_v$	$\mathbb{0}$
interface	$-\frac{H_b^{-1}\sigma^{V_2}}{JRe}B_{int}$	**Q**	$\mathbf{Q}_2$
	$\frac{H_b^{-1}\sigma^V}{Re}$	$\mathbf{E}_v$	$(\mathbf{E}_v)_2$

analysis were investigated in detail by Huan *et al.* [41]. The associated references should be consulted for a more in-depth review of the derivation and implementation of the SATs.

The solution at the boundary node with the velocity component normal to the boundary surface removed is Q_{V_t}; this is used for the inviscid wall and symmetry SAT. The far-field SAT target is the solution in the far-field: $Q_{\text{ff}} = [1, M_\infty \cos(\alpha), 0, M_\infty \sin(\alpha), \frac{1}{\gamma(\gamma-1)} + \frac{1}{2}M_\infty^2]^T$. Neighbouring blocks have coincident nodes at interfaces; Q_2 is the solution corresponding to the coincident interface node from the block neighbour. The term H_b is the first entry in the diagonal-norm matrix associated with the SBP operator, and $A_\xi^+ = \frac{A_\xi + |A_\xi|}{2}$ where $A_\xi = \frac{\partial \hat{\mathbf{E}}}{\partial \mathbf{Q}}$.

The viscous wall boundary conditions enforce the no-slip condition, $Q_w = [\rho, 0, 0, 0, e]^T$, which acts to zero the momentum equations at the solid surface. An additional wall boundary condition is added which enforces an adiabatic wall boundary condition, enforced by choosing a target viscous flux, $(E_v)_w$, with the energy component zeroed, i.e., $\partial_\zeta(a^2) = 0$ if the wall normal is in the ζ-direction. The symmetry boundary condition forces the boundary face node to take the value of the next most interior node in the direction of the boundary normal, which is $Q_{j+1,k,m}$ if the normal is in the ξ-direction. Both the wall and symmetry SATs contain a common factor of $\sigma^W \leq -\frac{\xi_x^2+\xi_y^2+\xi_z^2}{J}\frac{\mu}{2\rho}\max\left(\frac{\gamma}{\text{Pr}}, \frac{5}{3}\right)$. The far-field boundary is placed far away from the aerodynamic surface; the viscous SAT enforces zero viscous flux at the boundary. The factor of σ^V takes the value of 1 at the low-side boundary and -1 at the high-side boundary. The viscous interface boundary condition acts to make the solution and viscous flux nearly continuous across the boundary by choosing the target values to be the coincident block neighbouring node's solution vector and viscous flux vector respectively. The B_{int} matrix is related to the viscous Jacobian matrix and is derived based on Nordström *et al.* [66]. The factor is chosen as $\sigma^{V_2} \leq \frac{1}{2}$ for

stability. The B_{int} matrix for the ξ-direction is as follows:

$$B_{int,\xi} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_1u - a_2v - a_3w & a_1 & a_2 & a_3 & 0 \\ -a_2u - a_4v - a_5w & a_2 & a_4 & a_5 & 0 \\ -a_3u - a_5v - a_6w & a_3 & a_5 & a_6 & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & a_7 \end{pmatrix},$$

with the following substitutions:

$$a_1 = t_1\left(\frac{4}{3}\xi_x^2 + \xi_y^2 + \xi_z^2\right), \qquad a_2 = t_1\left(\frac{1}{3}\xi_x\xi_y\right),$$
$$a_3 = t_1\left(\frac{1}{3}\xi_x\xi_z\right), \qquad a_4 = t_1\left(\xi_x^2 + \frac{4}{3}\xi_y^2 + \xi_z^2\right),$$
$$a_5 = t_1\left(\frac{1}{3}\xi_y\xi_z\right), \qquad a_6 = t_1\left(\xi_x^2 + \xi_y^2 + \frac{4}{3}\xi_z^2\right),$$
$$a_7 = t_2\gamma\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right),$$
$$b_{51} = a_7\left(-\frac{e}{\rho} + (u^2 + v^2 + w^2)\right) - a_1u^2 - a_4v^2 - a_6w^2 - (a_2uv + a_3uw + aa_5vw),$$
$$b_{52} = -a_7u + a_1u + a_2v + a_3w, \qquad t_1 = \frac{\mu}{\rho},$$
$$b_{53} = -a_7u + a_2u + a_4v + a_5w, \qquad t_2 = \frac{\mu}{\rho Pr}.$$
$$b_{54} = -a_7u + a_3u + a_5v + a_6w,$$

2.2.2 Iteration to Steady State

A time-dependent problem is considered,

$$\partial_t\hat{\mathbf{Q}} = -\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}}), \tag{2.34}$$

where $\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}})$ is known as the discrete residual vector, and is given in its discrete form by

$$\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}}) = \boldsymbol{\mathcal{R}}_{\text{inviscid}} + \boldsymbol{\mathcal{R}}_{\text{viscous}} + \boldsymbol{\mathcal{R}}_{\text{AD}} + \boldsymbol{\mathcal{R}}_{\text{SAT}}, \tag{2.35}$$

where each residual contribution is constructed with the discrete finite-difference operators that have been presented, and are summarized in Appendix B. The goal is to solve the steady-state problem:

$$\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}}) = \mathbb{0}. \tag{2.36}$$

A globalized form of Newton's method is used to solve Equation 2.36, which uses pseudo-transient continuation to advance the solution to steady state. Since time accuracy is not important in a steady problem, the first-order implicit Euler time marching is used with the local linearization of $\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}})$ [52]:

$$\left[\frac{I}{\Delta t} + \frac{\partial\boldsymbol{\mathcal{R}}}{\partial\hat{\mathbf{Q}}}^{(n)}\right]\Delta\hat{\mathbf{Q}}^{(n)} = -\boldsymbol{\mathcal{R}}^{(n)}, \tag{2.37}$$

where the superscript n is the non-linear iteration index. As $\Delta t \to \infty$, Newton's method is obtained. Newton's method relies on the initial iterate to be sufficiently close to solution to converge quadratically. A poorly chosen initial iterate may lead to divergence of the solution algorithm. Care is taken to form a suitable initial guess with a stable start-up phase. The inexact-Newton phase proceeds when the start-up phase is completed until a solution sufficiently close to $||\boldsymbol{\mathcal{R}}||_2 = 0$ is obtained.

Local Time Stepping

Local time stepping introduces a time step specific to each cell [33, 75],

$$\Delta t_{j,k,m} = \frac{\Delta t^n_{\text{ref}}}{J_{j,k,m}(1 + \sqrt[3]{J_{j,k,m}})}. \tag{2.38}$$

This particular choice represents an approximately constant CFL number [75]. The reference time step Δt_{ref} is chosen differently for the start-up and inexact-Newton phases, as described below.

Jacobian-Free GMRES

Since the implicit Euler time marching method is employed in this algorithm, a linear system must be solved at each nonlinear iteration. The Generalized Minimal RESidual (GMRES) method developed by Saad and Shultz is used to solve the linear sub-problem at each nonlinear iteration [76, 77]. A flexible variant of GMRES, FGMRES, is used which allows the preconditioner to be updated at each iteration. GMRES uses a Krylov subspace, $\mathcal{K}_i = \text{span}\{v, Av, A^2v, \ \dots \ , A^{i-1}v\}$, to search for the minimum residual, and requires matrix-vector products Av to form the Krylov subspace. The linear system involves the flux Jacobian matrix; since high-order spatial discretization is used in this work, computing and storing the flux Jacobian matrix would be infeasible with the available hardware due to excessive memory requirements of the increased bandwidth finite-difference operators. Storage requirements of the high-order Jacobian are discussed in Chapter 4. The Jacobian-free GMRES algorithm uses an approximate Jacobian-vector product, facilitated by a Fréchet derivative [49],

$$\frac{\partial \boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}})}{\partial \hat{\mathbf{Q}}}\mathbf{v} \approx \frac{\boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}} + \epsilon_{\text{FD}}\mathbf{v}) - \boldsymbol{\mathcal{R}}(\hat{\mathbf{Q}})}{\epsilon_{\text{FD}}} \tag{2.39}$$

where the step size,

$$\epsilon_{\text{FD}} = \sqrt{\frac{N\delta_{\text{FD}}}{\mathbf{v}^T\mathbf{v}}}, \tag{2.40}$$

is carefully chosen to balance truncation and subtractive cancellation errors, N is the number of unknowns, and $\delta_{\text{FD}} = 10^{-13}$. Chisholm provides a detailed debugging guide for Jacobian-free GMRES as an appendix in his PhD thesis [15]. Michalek and Ollivier-Gooch have explored some two-dimensional inviscid flow analysis problems where computing and storing the high-order flux Jacobian matrix and applying it explicitly to generate matrix-vector products can be more efficient than using a Fréchet derivative approximation [60]. It is expected that, for three-dimensional analysis, the storage of the flow Jacobian will become excessively large, particularly when solving problems governed by the Navier-Stokes equations.

Preconditioning

Due to the stiffness of the linear system, it is often effective to use a preconditioner when solving the linear sub-problems introduced by the implicit Euler time marching method. A first-order flux Jacobian matrix is formed and an incomplete lower-upper (ILU) factorization is performed to build the preconditioner. To form the first-order flux Jacobian matrix, the second-order Euler and viscous fluxes are used with cross-derivative terms neglected in the viscous flux portion. The dissipation operator is condensed to a first-order dissipation operator and a dissipation lumping factor, σ, is introduced into ϵ in Equation 2.33 to increase the diagonal dominance of the preconditioner:

$$\epsilon_{\text{prec}} = \sigma\epsilon. \tag{2.41}$$

The modified dissipation operator uses the first-order dissipation exclusively, with a modified coefficient as shown above. The first-order dissipation scheme is identical to that presented in Equation 2.30, with D_d instead chosen to represent an undivided first-order approximation to the first-derivative,

$$D_1 = \begin{bmatrix} -1 & 1 & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}, \tag{2.42}$$

which is applied to the half nodes. Such an operator would be used in a first-order shock capturing scheme in a transonic analysis algorithm, but since only subsonic flows are analysed in this work it will only be used to form the preconditioner. Approximate-Schur parallel preconditioning is used [6, 32, 78]. A relatively high fill-level of 3 is used in the ILU factorization of the preconditioner in the flow analysis to improve preconditioning in the high-order methods.

Preliminary experimentation with a high-order preconditioner matrix for the inviscid Euler equations has shown no efficiency benefit over the first-order preconditioner, as the time to factorize the preconditioner matrix becomes excessively large. Michalak and Ollivier-Gooch have demonstrated efficiency gains in two-dimensional inviscid analysis problems when using a high-order flux Jacobian as the preconditioner matrix with a fill-level of 0 and 1 [60].

Start-up Phase

The purpose of a start-up phase is to provide the inexact-Newton phase with an initial iterate that is sufficiently close to the final solution that convergence is guaranteed. It is also important that the start-up phase be as quick as possible. A common approach to improving speed is to substitute the first-order Jacobian matrix for the flux Jacobian matrix in Equation 2.37; this is the approach taken in several other high-order algorithms [63, 25]. This approach is suitable for second-order since the first-order preconditioner matrix closely represents the second-order flux Jacobian. However, for high-order analysis with an increased bandwidth flux Jacobian matrix the first-order preconditioner no longer closely represents the flux Jacobian matrix. Encountering a negative pressure or density can destabilize the analysis algorithm. It is to prevent the solution from wandering into regions of non-physical behaviour that a more accurate matrix-vector product is desired. In this work the Fréchet derivative is used to

approximate Jacobian-vector products during the start-up phase.

Equation 2.37 is solved to a specified tolerance, ω, such that:

$$\left|\left|\boldsymbol{\mathcal{R}}^{(n)} + \left[\frac{I}{\Delta t} + \frac{\partial \boldsymbol{\mathcal{R}}^{(n)}}{\partial \hat{\mathbf{Q}}}\right]\Delta\hat{\mathbf{Q}}^{(n)}\right|\right|_2 \leq \omega||\boldsymbol{\mathcal{R}}^{(n)}||_2. \tag{2.43}$$

The parameter ω is typically chosen to be 0.01; however, the number of Krylov iterations is limited to reduce memory requirements of the linear solver. In the start-up phase when time steps are smaller, the linear system is often solved to the specified tolerance before the Krylov iteration limit is reached. It is possible that as the time step size increases the specified tolerance ω is not reached in the allowed number of Krylov iterations. The reference time step is chosen such that:

$$\Delta t^n_{\text{ref}} = ab^n. \tag{2.44}$$

Typical ranges for these time-stepping parameters are $a \in [10^{-4}, 10^{-2}]$ and $b \in [1.1, 1.3]$.

Inexact-Newton phase

The transition to the inexact-Newton phase is made when the relative residual norm has fallen below a specified cut-off, τ:

$$||\boldsymbol{\mathcal{R}}^{(n)}||_2 \leq \tau||\boldsymbol{\mathcal{R}}^{(0)}||_2 \tag{2.45}$$

and τ is typically chosen to be in the range $[10^{-4}, 10^{-2}]$. Equation 2.43 is solved to a similar tolerance as the start-up phase with ω typically chosen to be 0.01.

The reference time step is computed differently in the inexact-Newton phase, with more aggressive growth:

$$\Delta t^{(n)}_{\text{ref}} = \max\left(\alpha\left(\frac{||\boldsymbol{\mathcal{R}}^{(n)}||_2}{||\boldsymbol{\mathcal{R}}^{(0)}||_2}\right)^{-\beta}, \Delta t^{(n-1)}_{ref}\right) \tag{2.46}$$

where

$$\alpha = ab^n\left(\frac{||\boldsymbol{\mathcal{R}}^{(k)}||_2}{||\boldsymbol{\mathcal{R}}^{(0)}||_2}\right)^{\beta} \tag{2.47}$$

where k is the nonlinear iteration index of the last start-up phase iteration, and β is in the range $[1.5, 2.0]$. The analysis is considered complete when the relative residual norm falls below 10^{-10} or the absolute residual norm falls below 10^{-12}.

2.2.3 Force Integration

Lift and drag coefficients are evaluated throughout the flow analysis procedure and require a surface integration of the pressure and viscous stress forces. Hicken and Zingg have demonstrated super-convergence when using the diagonal-norm operator as an integration rule in conjunction with a dual-consistent formulation when using the SBP operator which corresponds to the diagonal-norm to discretize the derivative [37, 38, 39]. While this work does not consider a dual-consistent formulation, the diagonal-norm used to form the first-derivative is used as the integration rule. The lift and drag coefficients, C_L and

C_D respectively, can be computed with the following continuous integration:

$$C_L = \frac{1}{S} \oint C_p \overrightarrow{s} \cdot d\overrightarrow{S} \tag{2.48}$$

$$C_D = \frac{1}{S} \oint C_p \overrightarrow{r} \cdot d\overrightarrow{S} \tag{2.49}$$

where

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.50}$$

$$\overrightarrow{r} = \sin\ \alpha\hat{x} + \cos\ \alpha\hat{z} \tag{2.51}$$

$$\overrightarrow{s} = \cos\ \alpha\hat{x} - \sin\ \alpha\hat{z} \tag{2.52}$$

$$d\overrightarrow{S} = J^{-1}(\zeta_x\hat{x} + \zeta_y\hat{y} + \zeta_z\hat{z})d\xi d\eta \tag{2.53}$$

where S is the surface area, and α is the angle of attack. The directions ξ and η are assumed to be parallel to the surface.

The integrals are calculated using the norm matrix, H, used to construct the first-derivative, as it corresponds to an integration rule of order $2s$. Consider the surface integral of a generic function $f(x, y)$; applying the coordinate transformation and diagonal-norm integration rules H_ξ and H_η yields

$$\int_{y_1}^{y_2}\int_{x_1}^{x_2} f(x,y)dxdy = \int_{\eta_1}^{\eta_2}\int_{\xi_1}^{\xi_2} f(\xi,\eta)\left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right| d\xi d\eta \tag{2.54}$$

$$= \int_{\eta_1}^{\eta_2}\int_{\xi_1}^{\xi_2} f(\xi,\eta)|J^{-1}\zeta_x|d\xi d\eta \tag{2.55}$$

$$\approx \sum_{j=1}^{N_\eta}\sum_{i=1}^{N_\xi}(H_\eta)_{j,j}(H_\xi)_{i,i}f(i,j)dS, \tag{2.56}$$

where N_ξ and N_η are the number of nodes on the surface in the ξ- and η-directions respectively. The term J^{-1} accounts for the coordinate transformation and is computed with Equation 2.9.

Chapter 3

Geometry Parametrization, Mesh Movement, and Meshing Considerations

When considering a geometry parametrization and mesh movement strategy, design goals must be taken into consideration. An exploratory optimization algorithm must not restrict the design space with its surface definition and must allow for large shape changes with each surface and mesh update. It is also important that the mesh update not degrade the quality of the mesh significantly. These design goals have been taken into consideration with the selection of a B-spline tensor volume parametrization of the volume mesh [34] and the use of a multi-increment linear elastic mesh movement scheme [34, 85]

3.1 B-Spline Parametrization

In order to accommodate large shape changes while preserving the design space of the intended optimization problem, the multi-block computational mesh is approximated with a B-spline tensor volume in an integrated parametrization and mesh movement scheme developed by Hicken and Zingg [34]. The B-spline mappings take the form

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{k=1}^{N_k} \mathbf{B}_{ijk} \mathcal{N}_i^{(p)}(\xi) \mathcal{N}_j^{(p)}(\eta) \mathcal{N}_k^{(p)}(\zeta), \tag{3.1}$$

and are defined by a set of control points, $\mathbf{B}_{ijk}$, and a set of B-spline basis functions of order p, $\mathcal{N}^{(p)}$. Equation 3.1 defines the analytical mapping of the curvilinear coordinates $\boldsymbol{\xi} = (\xi, \eta, \zeta) \in \mathbb{R}^3 \mid \xi, \eta, \zeta \in [0, 1]$ to physical space $\mathbf{x} = (x, y, z)$. The B-spline basis functions in the ξ-direction are expressed as

$$\mathcal{N}_i^{(1)}(\xi) = \begin{cases} 1; & \text{if } T_i(\eta, \zeta) < \xi < T_{i+1}(\eta, \zeta) \\ 0; & \text{otherwise} \end{cases} \tag{3.2}$$

$$\mathcal{N}_i^{(p)}(\xi) = \left(\frac{\xi - T_i(\eta,\zeta)}{T_{i+p-1}(\eta,\zeta) - T_i(\eta,\zeta)}\right)\mathcal{N}_i^{(p-1)}(\xi) + \left(\frac{T_{i+p}(\eta,\zeta) - \xi}{T_{i+p}(\eta,\zeta) - T_{i+1}(\eta,\zeta)}\right)\mathcal{N}_{i+1}^{(p-1)}(\xi). \tag{3.3}$$

Similar expressions exist for the basis functions in the η- and ζ- directions for $\mathcal{N}_j^{(p)}(\eta)$ and $\mathcal{N}_k^{(p)}(\zeta)$ respectively. The knot values, $T_i(\eta,\zeta)$, in the interior of the B-spline volume are given by

$$T_i(\eta,\zeta) = [(1-\eta)(1-\zeta)]T_{i,(0,0)} + [\eta(1-\zeta)]T_{i,(1,0)} + [(1-\eta)\zeta]T_{i,(0,1)} + [\eta\zeta]T_{i,(1,1)}. \tag{3.4}$$

Similar expressions exist for the knot values in the η- and ζ-directions, $T_i(\zeta,\xi)$ and $T_i(\xi,\eta)$. The first p and last p knot values for a given B-spline curve are set to 0 and 1, respectively; the edge knot values $T_{i,(0,0)}$, $T_{i,(1,0)}$, $T_{i,(0,1)}$, and $T_{i,(1,1)}$ are constants. Given the edge knot values, the interior knots can be determined through bilinear interpolation from Equation 3.4. Subsequently, the control points are determined through a least squares fitting. This is done in a sequential manner, first fitting the block edges, then the sides, and finally the interior volumes. Since the parameter values ξ, η, and ζ are based on a chord-length parametrization, the knot values are inherently chord-length based as well, leading to a coarse B-spline volume mesh that mimics the bunching of the finer CFD mesh.

3.2 Mesh Movement

A robust linear elastic mesh movement strategy is used within this work. This mesh movement scheme treats each cell within the control point mesh as a linear elastic solid, with the cell stiffness dependant on the volume and distortion of the cell. Cells with a smaller volume or high distortion relative to their initial shape are treated as stiffer and more resistant to shape changes. This approach leads to infinite stiffness as mesh entanglement is approached.

Typically, coordinates of the B-spline control points on the surface of the geometry are taken as design variables within the optimization problem, with linear constraints enforcing some degree of coupling between the design variables. At each design iteration, the optimizer updates the design variables to change the aerodynamic shape. The remaining internal control points in the B-spline volume are updated incrementally treating each cell as a linear elastic solid,

$$\boldsymbol{\mathcal{M}}^{(i)}(\mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}) = K^{(i)}[\mathbf{b}^{(i)} - \mathbf{b}^{(i-1)}] - \mathbf{f}^{(i)} = 0; \quad i \in [1, m], \tag{3.5}$$

where $\boldsymbol{M}^{(i)}$ is the mesh movement residual, $\mathbf{b}^{(i)}$ the set of B-spline control point coordinates for the given volume, $K^{(i)}$ is the global stiffness matrix at increment i, and m the number of mesh movement increments. The stiffness matrix, $K^{(i)}$, is constructed such that cells with a small volume or high distortion resist deformation. The force vector $f^{(i)}$ is defined implicitly by the movement of the surface control point coordinates. Equation 3.5 is solved using the conjugate gradient method preconditioned with a Jacobi-Schwartz preconditioner [53], which exploits the symmetric positive definite structure of $K^{(i)}$. At increment i, the surface control points, $b_s^{(i)}$ are updated using a linear interpolation of the initial and final values, $b_s^{(0)}$ and $b_s^{(m)}$ respectively:

$$\mathbf{b}_s^{(i)} = \frac{i}{m}(\mathbf{b}_s^{(m)} - \mathbf{b}_s^{(0)}) + \mathbf{b}_s^{(0)}; \quad i \in [1, m]. \tag{3.6}$$

The linear-elasticity mesh movement approach is robust, but comes with an increased cost relative to algebraic mesh movement schemes. Applying the linear-elastic mesh movement scheme directly to the

Table 3.1: Minimum number of nodes on a single edge when high-order SBP operators are used.

Global Order	s	Min. First-Deriv. Size	Min. Second-Deriv. Size
2nd-order	1	3	5
3rd-order	2	9	13
4th-order	3	13	19
5th-order	4	17	25
6th-order	5	23	33
7th-order	6	29	41
8th-order	7	39	53
9th-order	8	47	63
10th-order	9	57	75

computational mesh would be expensive. The method is instead applied to the coarser control mesh defined by the B-spline control points. When applied to the control mesh, this mesh movement strategy contributes very little to the CPU time required for optimization when compared to some of the more CPU intensive tasks such as the flow analysis and flow adjoint solution. There are many references which demonstrate the robustness of the linear elastic mesh movement scheme used within this work [85, 34, 70].

Using multiple increments in the mesh movement scheme improves the robustness while maintaining linearity. To balance speed and robustness, five mesh movement increments are used unless stated otherwise.

3.3 Minimum Mesh Size with High-Order SBP Operators

When constructing a mesh for use with high-order, one must consider the minimum size of the SBP operator to be used. For the first-derivative operators shown in Appendix A, the operator must include all boundary nodes and at least one interior node. Since the boundary includes $2s$ nodes on each side, that leads to a minimum edge size of $4s + 1$. The compact second-derivative operator requires $3s$ boundary nodes at each boundary, for a minimum of $6s + 1$. Second-order is an exception to this rule as the outermost boundary stencil has the same form as the interior stencil. For diagonal-norm SBP operators of $s \geq 5$, a positive-definite diagonal norm is not guaranteed for the boundary structure that has been presented [79]. Instead, additional boundary nodes are included to increase the number of free parameters in the SBP operator while also introducing free parameters in H, allowing for the diagonal-norm to become positive definite under a specific range of free parameter choices. This increase in boundary nodes will cause an increase in the minimum mesh size for SBP operators of $s \geq 5$. Minimum mesh sizes for SBP operators of higher-order schemes are shown in Table 3.1. Schemes of higher-order than is considered in this work are shown to demonstrate the constraints of extending to even higher-order schemes. Minimum boundary extensions to maintain a positive definite norm have been demonstrated by Albin and Klarmann [1].

3.4 One-Dimensional Analysis of Stretching Functions

A common problem with high-order codes is their sensitivity to grid smoothness, and the presented algorithm is no exception. Specifically, the problem occurs during the numerical evaluation of the grid metrics. Numerical evaluation of the grid metrics is preferred because it produces lower truncation error than analytical metrics [84, 86] and the metric invariants in Equations 2.18–2.20 are formulated using numerical derivatives so cannot be satisfied using analytical grid metrics. The nature of the second-order SBP operator guarantees that the numerical grid metrics are evaluated with the same sign as the analytical metrics. High-order SBP operators make use of larger one-sided or biased differencing near the boundaries, which introduces the possibility of numerical grid metrics with sign opposite to the analytical metrics. This can lead to a negative metric Jacobian for at least one node within the domain and, since the curvilinear coordinate transformation is no longer invertible, causes the algorithm to terminate. It is important, therefore, to approach grid generation with some intuition as to what properties of the grid can cause a negative Jacobian when applied to a high-order algorithm.

Two popular stretching functions are analysed in detail: the geometric and hyperbolic tangent stretching functions. These stretching functions are one-dimensional and represent a subset of stretching functions used by popular meshing software such as ICEMCFD [5]. The stretching functions will be differentiated analytically and numerically. The numerical metric will be compared to the analytical metric to determine if the sign is correct. Limitations of each stretching function will be discussed in a way that recommendations can be made for users that need to generate a suitable grid for use with a high-order analysis algorithm.

3.4.1 Geometric Stretching

Stretching functions map the coordinates in computational space, $\xi \in [1, N]$, to a scaled physical space, $x \in [0, 1]$. The geometric mesh law is given by

$$x(\xi) = \frac{p^{\xi-1} - 1}{p^{N-1} - 1}. \tag{3.7}$$

where the stretching ratio, p, relates the spacing ratio of two adjacent cells. The analytical grid metric for the geometric stretching function is given by

$$\frac{dx}{d\xi} = \ln(p) \frac{p^{\xi-1}}{p^{N-1} - 1} \tag{3.8}$$

The numerical grid metrics of the geometric stretching function can be analysed exactly, and the conditions at which the numerical metric, $\frac{\partial x}{\partial \xi}$, becomes zero can be precisely determined. As an example, for any SBP operator δ_ξ, a linear system can be realized by applying δ_ξ to a point distribution generated by Equation 3.7 and solving polynomial equations for the spacing ratio which causes the metric to become zero,

$$\delta_\xi \mathbf{x} = \mathbb{0}. \tag{3.9}$$

All SBP operators used in this work are presented in Appendix A. The nodal distribution vector, $\mathbf{x}$, in Equation 3.9 is directly from Equation 3.7. A polynomial is formed for each row-vector product and the

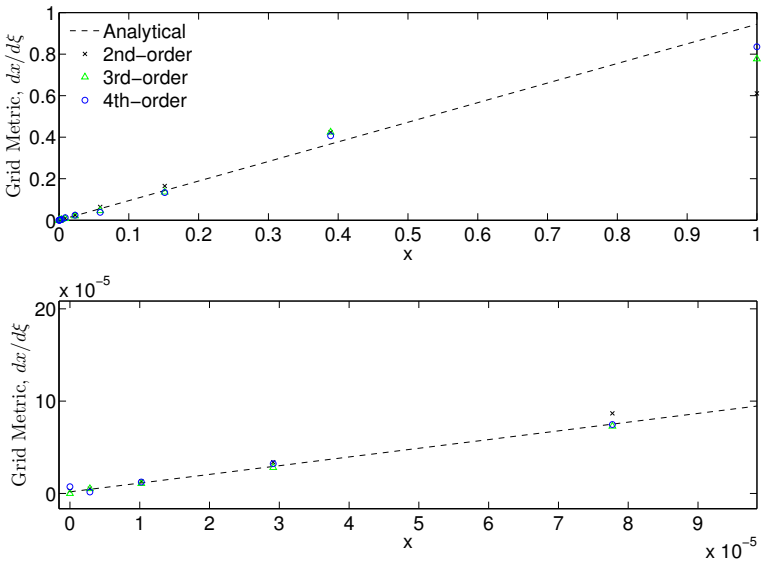


Figure 3.1: Grid metric of the one-dimensional geometric stretching function computed analytically and numerically on a grid of 15 nodes with a stretching ratio of 2.57. The bottom graph shows a close-up near the left boundary where a numerical grid metric becomes less than zero for the third-order method.

real-valued solutions to those polynomials indicate the ratios which cause the algorithm to produce grid metrics which are opposite in sign to their analytical value determined from Equation 3.8. The minimum real-valued root when the third-order SBP operator is used is $p \approx 2.5668$ for $p \in (1, \infty)$. Generating a grid with a stretching ratio larger than this will produce grid metrics of the wrong sign at a minimum of one grid node and will lead to termination of the algorithm due to an invalid curvilinear coordinate transformation. The cut-off ratio for the fourth-order operator is $p \approx 2.8310$; no cut-off ratio exists for the second-order operator; the only solutions to the polynomials generated by Equation (3.9) for the second-order case are $p = \{0, 1\}$, which are not valid stretching functions. The analytical and numerical grid metrics for $p = 2.57$ can be seen in Figure 3.1.

3.4.2 Hyperbolic Tangent Stretching

The hyperbolic tangent stretching function is commonly used because of the increased clustering of nodes near the boundaries which enable it to capture the solution within the boundary layer accurately [84]. Like the geometric stretching function, the hyperbolic tangent stretching function maps the coordinates in computational space, $\xi \in [1, N]$, to a scaled physical space, $x \in [0, 1]$; it is given by the following series of equations:

$$f = \frac{\xi - 1}{N - 1} - \frac{1}{2}, \tag{3.10}$$

$$U = 1 + \frac{\tanh(bf)}{\tanh(b/2)}, \tag{3.11}$$

$$x = \frac{U}{2A + (1 - A)U}. \tag{3.12}$$

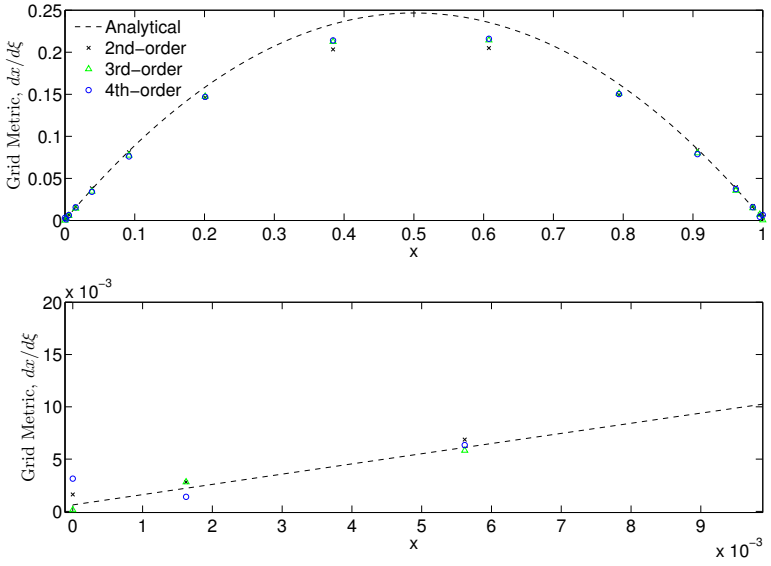


Figure 3.2: Grid metric of the one-dimensional hyperbolic tangent stretching function computed analytically and numerically on a grid of 15 nodes and both edge spacing values set to 0.001. The bottom graph shows a close-up near the left boundary where a numerical grid metric becomes less than zero for the third-order method.

In Equation 3.12, r relates the edge spacing parameters Sp_1 and Sp_2 by $r = Sp_2/Sp_1$, b is determined through the solution to a nonlinear equation, $\sinh(b) = \frac{b}{(N-1)\sqrt{Sp_1 Sp_2}}$, and $A = \sqrt{r}$. The analytical metrics can be determined with the use of chain rule:

$$\frac{dx}{d\xi} = \frac{dx}{dU}\frac{dU}{df}\frac{df}{d\xi}, \tag{3.13}$$

$$\frac{dx}{dU} = \frac{2A}{(2A + (1-A)U)^2}, \tag{3.14}$$

$$\frac{dU}{df} = b\frac{1 - \tanh(bf)^2}{\tanh(b/2)}, \tag{3.15}$$

$$\frac{df}{d\xi} = \frac{1}{N-1}. \tag{3.16}$$

The hyperbolic tangent stretching function is analysed numerically. The minimum spacing parameter is determined for a specified order of accuracy, number of grid nodes, and relative spacing parameter. This is done by using the bisection method to test the grid metrics for improper signs and successively narrow the search region. One hundred iterations are used in order to obtain spacing parameters to machine precision. Once again, no limitations are presented for second-order as the numerical grid metrics are guaranteed to have the correct sign. Figure 3.2 shows the grid metric across the domain under conditions which cause a negative Jacobian for the third-order method, with a close up near the left boundary where one such negative Jacobian is present. In this example Sp_1 and Sp_2 are both chosen as 0.001, with 15 grid nodes. Figure 3.3 shows the minimum allowable edge spacing parameter, Sp_1, possible to avoid negative metric errors under a wide range of conditions.

A one-dimensional analysis is important here as it allows the grid metric, ξ_x, and metric Jacobian

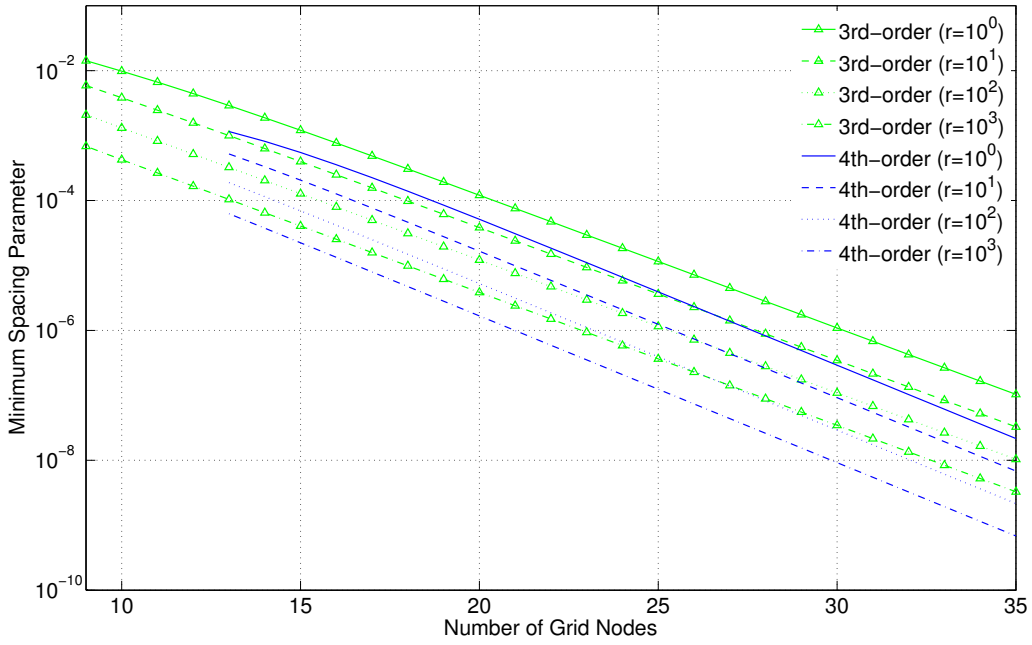


Figure 3.3: Minimum allowable mesh spacing parameter, Sp_1, for use with the hyperbolic tangent stretching function; various values of the edge spacing ratio, r, are shown

to be interchangeable. Based on the above analysis of the hyperbolic tangent stretching function, the following recommendations are made to users which are experiencing difficulty generating grids for use with a high-order algorithm:

- increase the number of nodes in the mesh;
- increase the endpoint spacing ratio, $r = \frac{Sp_2}{Sp_1}$;
- decrease the length of the physical domain while keeping the number of nodes constant;
- lastly, preference should be given to the fourth-order SBP operator when a troublesome mesh is encountered and use of a high-order algorithm is a paramount objective.

To conclude this analysis, it is important to remember that testing the Jacobian for negative values is often a great test to check that crossover of the mesh nodes has not occurred, for instance after a mesh movement algorithm has been applied. A negative Jacobian in the second-order algorithm indicates that cross-over of grid nodes has occurred; a negative Jacobian in the high-order algorithm can occur from cross-over and from nodes that have to high of a stretching ratio. This analysis is a starting point to guide the mesh generator in creating a structured mesh which has an invertible curvilinear coordinate transformation when used with a high-order algorithm. The suggested endpoint spacing values for the hyperbolic stretching function should be taken as optimistic values when cell skewness is introduced, but should be realistic when mesh lines are reasonably orthogonal. The presented one-dimensional analysis is precise in three-dimensional space when no cell skewness is introduced, since the metric Jacobian matrix is purely diagonal in this case.

Chapter 4

Gradient-Based Optimization

Optimization involves determining the maximum or minimum value of a desired quantity through manipulation of design variables and repeated analysis. Depending on the selected optimization algorithm, an optimal design may represent either a globally or locally optimal design. A design space is specified through a choice of design variables and accompanying constraints, as well as any nonlinear constraints related to the analysis such as volume or lift. Efficient optimization algorithms rely on efficient analysis algorithms; the chosen optimization algorithm will include the use of high-order analysis as it has been shown to be an efficient strategy to reduce error.

Zingg *et al.* compared a global optimization technique, a genetic algorithm, and a local optimization technique, a gradient-based algorithm. It was found that up to 200 times the number of function evaluations can be expected for a genetic algorithm compared to a gradient-based algorithm [89]. In the context of CFD a functional evaluation is rather expensive and, for this reason, global optimization strategies are not employed in this framework. Gradient-based optimization algorithms are desirable due to their faster convergence rates, requiring fewer function evaluations. Although a global optimum is not guaranteed, and the obtained design may be a local optimum in a multi-modal optimization problem, hybrid optimization strategies can be employed to obtain a globally optimal design. Hybrid multi-start algorithms have been examined by Chernukhin and Zingg [13, 14] and have demonstrated improved performance to a genetic algorithm while still obtaining the global optimum in a multi-modal design problem. Hybrid algorithms are not employed in this work, but present an efficient strategy to extend the presented optimization algorithm to obtain globally optimal aircraft designs.

This chapter formally introduces the design problem and presents the gradient-based algorithm that will be applied to obtain a local optimum. Linear sub-problems are solved in order to compute the gradient; the relevant linear equations will be presented and their solution strategies discussed. Several components of the optimizer will be subsequently verified and validated to provide a level of confidence in the chosen algorithm.

4.1 Optimization Algorithm

Aerodynamic shape optimization can be cast as a PDE-constrained minimization problem,

$$\begin{aligned} \min \quad & \mathcal{J}\left(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{Q}\right) \\ \text{w.r.t.} \quad & \mathbf{v} \\ \text{s.t.} \quad & \mathbf{c_i}\left(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{Q}\right) \leq \mathbb{0}, \mathbf{c_e}\left(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{Q}\right) = \mathbb{0}. \end{aligned} \tag{4.1}$$

The objective function to be minimized, $\mathcal{J}$, is a function of the design variables, $\mathbf{v}$, B-spline control points at the final mesh movement increment m, $\mathbf{b}^{(m)}$, and the flow variables, $\mathbf{q}$. The minimization problem is constrained by the residual vectors defining the physical flow and the mesh movement algorithm as well as any additional non-linear constraints such as a lift constraint, all of which are contained in the equality constraint, $\mathbf{c_e}$. Additional nonlinear inequality constraints may be introduced within $\mathbf{c_i}$.

With the selection of a gradient-based optimization algorithm, an efficient method to compute the gradient of the objective function with respect to the design variables, $\frac{\partial \mathcal{J}}{\partial \mathbf{v}}$, is required. The most straightforward approach to gradient computation is the use of finite-differences. The finite-difference method requires the perturbation of the design variables, with additional functional evaluations required for each perturbation. The cost of an aerodynamic analysis is expensive, so reducing the total number of functional evaluations is desirable. If very few design variables are used then this can be a viable approach; however, when practical problems are described, often several hundred design variables are required. Finite-differencing also requires careful selection of the perturbation parameter to carefully balance the effects of truncation error and subtractive cancellation error.

One possible alternative to the finite-difference method is the complex-step method, which perturbs the design variables by a complex valued perturbation. The complex-step method does not suffer from subtractive cancellation error, so the step size dilemma can be avoided by choosing a very small step size, such as 10^{-20} or smaller. The largest challenge with this method is designing analysis software that is accepting of complex valued design variables. Martins has developed an automated tool that will transform computer programs designed to operate with real valued variables to be compatible with complex valued variables [54, 55]. This tool could realistically be applied to the analysis software used; however Martins has shown that the complex valued analysis software performs slower than the real valued analysis. The cost of the gradient computation using either the finite-difference or complex-step method is clearly dependent on the number of design variables and the need for a more efficient strategy is apparent.

The adjoint method allows for the gradient to be computed through the intermediate solution of a linear system. The cost of solving the linear system for the adjoint variables is of the same order of magnitude as the cost of a flow analysis, and is nearly independent of the number of design variables [46]. The adjoint method was first used in the aerodynamic shape optimization context by Jameson [46]. The gradient computation is coupled with SNOPT [27]. The SNOPT user manual [28] can be consulted for specific details of the optimization algorithm, and Hicken's PhD thesis provides an overview of the pertinent algorithm features [29].

4.1.1 Adjoint Method for Gradient Computation

When the adjoint method is used, the design problem can be approached by developing a Lagrangian function,

$$\mathcal{L}(\mathbf{v}, \mathbf{Q}, \boldsymbol{\lambda}^{(i)}|_{i=1}^{m}, \boldsymbol{\psi}) = \mathcal{J}(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{Q}) + \sum_{i=1}^{m} \boldsymbol{\lambda}^{(i)T} \boldsymbol{\mathcal{M}}^{(i)}(\mathbf{v}, \mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}) + \boldsymbol{\psi}^T \boldsymbol{\mathcal{R}}(\mathbf{v}, \mathbf{b}^{(m)}, \mathbf{Q}), \tag{4.2}$$

which incorporates the flow and mesh residual constraints into the objective function through the use of Lagrange multipliers. In Equation 4.2, $\boldsymbol{\lambda}^{(i)}|_{i=1}^{m}$ and $\boldsymbol{\psi}$ are Lagrange multipliers that represent the mesh and flow adjoint variables respectively. $\boldsymbol{\mathcal{R}}$ and $\boldsymbol{\mathcal{M}}$ represent the flow and mesh movement residual equations and $b^{(i)}$ the control point coordinates of the B-spline volume at mesh movement increment i.

To obtain an optimal design, the first-order optimality conditions, also known as the Karush-Kuhn-Tucker (KKT) conditions, must be satisfied. The first KKT condition requires that the flow and mesh residual equations are satisfied. If a successful analysis and mesh-movement are performed, then this condition is automatically satisfied. The second KKT condition requires that the flow and mesh adjoint equations be solved for the adjoint variables, $\boldsymbol{\psi}$ and $\boldsymbol{\lambda}^{(i)}$. The adjoint variables can then be used to compute the gradient of the Lagrangian with respect to the design variables.

Flow Adjoint Equations

The flow adjoint equations are formed by requiring that the second KKT condition is satisfied. This requires that $\frac{\partial \mathcal{L}}{\partial \mathbf{Q}} = \mathbb{0}$. Expanding gives,

$$\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}}\right)^T \boldsymbol{\psi} = -\left(\frac{\partial \mathcal{J}}{\partial \mathbf{Q}}\right)^T, \tag{4.3}$$

where all variables are as previously defined. Equation 4.3 represents a system of linear equations and is solved using GCROT [19, 20, 36]. GCROT is a Krylov subspace method which uses GMRES as its inner method and, rather than restarting GMRES when reaching a Krylov subspace size limit, recycles a portion of the Krylov subspace in the previous GMRES iteration. By preserving some of the previous Krylov subspace, GCROT is less susceptible to stalling than restarted GMRES [29, 36]. A flexible variant of GCROT is used, which allows the preconditoner to be updated at every Krylov iteration. This is necessary due to the use of approximate-Schur preconditioning.

As can be seen in Equation 4.3, the transposed Jacobian is required to determine the flow adjoint variables. Since the flow adjoint equations are solved iteratively using a Krylov subspace method, only the transposed Jacobian-matrix-vector products with an arbitrary vector, $\mathbf{v}$, are required: $\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}}\right)^T \mathbf{v}$. Some researchers have elected to perform this matrix-vector product by first storing the Jacobian matrix explicitly and performing the transposed Jacobian-vector products using the stored matrix when required [34, 69]. In this work, an implicit transposed Jacobian-vector product approach is taken: $A^T v$ is formed analytically at each matrix-vector product request [3, 39, 11, 64]. This strategy circumvents the large memory requirements of storing the high-order Jacobian matrix, with a computational cost penalty at each iteration associated with recomputing Jacobian contributions. The trade-off between CPU time and memory will be made for second-order, since both methods are available only for second-order. Memory requirements for the first- and second-order flux Jacobian matrix are shown in Table 4.1 and

Table 4.1: Flux Jacobian memory measurements and estimates in gigabytes. Bolded entries indicate a memory estimate which is greater than the system can provide.

		Memory Per Block on Grid (GB)				
Order	**Stencil Size**	G_1	G_2	G_3	G_4	G_5
First-order (preconditioner)	16	0.03	0.04	0.06	0.10	0.14
Second-order	52	0.09	0.13	0.20	0.31	0.46
Third-order estimate	125	0.23	0.32	0.48	0.74	1.11
Fourth-order estimate	343	0.62	0.87	1.31	**2.04**	**3.05**
Nodes per block		6 859	10 051	15 979	26 011	40 204

were determined using the massif tool within valgrind [1]. Details of the grids used can be found in Table 5.4. Memory estimates are made for higher-order schemes with the assumption that linear growth in the memory is seen with respect to the stencil size. Cross-derivative terms due to the viscous terms cause the stencil size to be at minimum $(2s+1)^3$ in the interior, with boundary treatments increasing the memory requirements. When making memory estimates, the stencil size for third- and fourth- order are assumed to be dictated by the interior stencil size, which will be an underestimate of actual memory requirements. Considering that the available hardware has about 1.8GB of memory available per processor, it is clear from Table 4.1 that storing a high-order Jacobian has excessive memory requirements for a mesh of reasonable size. The relative cost of the implicit matrix-vector product technique is examined in Table 4.2. The time to solve the flow adjoint equations to a relative tolerance of 10^{-8} is recorded for the implicit and explicit matrix-vector product techniques. The implicit matrix-vector products are exactly as is described in this thesis, which do not require the storage of the Jacobian matrix, while the explicit matrix-vector products are as described by Osusky [70] and require the storage of the Jacobian matrix. A different dissipation model is used for the explicit strategy, which results in fewer iterations to solve the adjoint problem by no more than 15%; this does not obscure the results much since the difference in GCROT iterations is small. The data show that the implicit matrix-vector product strategy can take up to 3.89 times as long to converge as the explicit technique for a similar adjoint problem with the second-order method. This factor will likely increase with high-order methods as there are more terms to recompute at each iteration when the implicit method is used. Although this increase in cost is undesirable, it allows for larger block sizes to be used when performing a mesh refinement study in the presence of a system memory constraint. Subdividing the domain further into smaller blocks and distributing amongst more processors (while retaining the same amount of memory per processor) is an alternative approach to addressing memory issues when storing the high-order flux Jacobian matrix. This approach was not taken in this work as a consistent surface definition and consistent mesh refinement technique was desired, which requires the multi-block structure to remain constant across all mesh levels.

To further demonstrate the memory limitations of storing the full Jacobian matrix, estimates are made of the maximum allowable block sizes. This is done by linearly interpolating or extrapolating the data in Table 4.1 to determine the block size which would cause the memory of the Jacobian and preconditioner matrix to exceed the available memory (1.8GB per block). In this estimate, it is assumed that 25% of the memory is reserved for overhead (ILU factorization, Krylov vectors, etc.). The minimum block and edge sizes are determined using the values from Table 3.1 and the maximum edge length is

[1] Documentation for the valgrind massif tool can be found at http://valgrind.org/docs/manual/ms-manual.html

Table 4.2: Time comparison of matrix-vector product techniques for the second-order method.

	Time on Grid(s) (20 proc.)				
Matrix-Vector Product Type	G_1	G_2	G_3	G_4	G_5
Explicit	39.3	56.1	88.9	152.8	249.7
Implicit	133.1	205.5	346.1	592.8	966.7
Ratio	3.39	3.66	3.89	3.88	3.87

Table 4.3: Minimum and maximum block and edge size estimates for each order of accuracy when explicit matrix-vector products are used.

Order	**Block size lower limit**	**Block size upper limit**	**Isotropic edge size lower limit**	**Isotropic edge size upper limit**
Second-order	125	94 390	5	45
Third-order	2 197	44 545	13	35
Fourth-order	6 859	16 869	19	25

determined by taking the cube root of the maximum block size (assumes an equal edge length in all three directions). The minimum and maximum block and edge sizes are shown in Table 4.3. Considering the memory estimates in Table 4.1 are underestimates, the range of allowable edge lengths of the third- and fourth-order methods with explicit storage of the Jacobian matrix is too small to facilitate a proper grid refinement study. The reduced memory requirement of the implicit matrix-vector products is desirable as it increases the upper limit of the block size limitations.

Equation 4.3 represents a large portion of the development work required to complete this thesis project. This involved the introduction of a Jacobian-vector product strategy to avoid the memory limitations associated with storing the high-order flux Jacobian matrix, and extending the flow Jacobian computation to include high-order terms. The dissipation model was also differentiated as part of the development work since, with this framework, this particular dissipation model has only been used in analysis by Dias [24]. The objective function derivatives were also extended to include high-order integration using the diagonal-norm operator associated with the SBP operator.

Mesh Adjoint Equations

Mesh adjoint equations are also derived from the second of the KKT conditions, requiring that $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(i)}} = \mathbb{0}$, at each mesh movement increment i;

$$\left(\frac{\partial \boldsymbol{\mathcal{M}}^{(m)}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\lambda}^{(m)} = -\left(\frac{\partial \mathcal{J}}{\partial \mathbf{b}^{(m)}}\right)^T - \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\psi}, \tag{4.4}$$

$$\left(\frac{\partial \boldsymbol{\mathcal{M}}^{(i)}}{\partial \mathbf{b}^{(i)}}\right)^T \boldsymbol{\lambda}^{(i)} = -\left(\frac{\partial \boldsymbol{\mathcal{M}}^{(i+1)}}{\partial \mathbf{b}^{(i)}}\right)^T \boldsymbol{\lambda}^{(i+1)}, \quad i \in \{m-1, m-2, \ldots, 1\}. \tag{4.5}$$

The mesh adjoint equations are solved after the flow adjoint equations so that the flow adjoint variables are updated. The mesh adjoint equations are solved in reverse order with the final increment, Equation 4.4, solved first. The right-hand side of Equation 4.4 is manipulated through the use of chain rule to

simplify the implementation [29]:

$$-\left(\frac{\partial \mathcal{J}}{\partial \mathbf{b}^{(m)}}\right)^T - \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{b}^{(m)}}\right)^T \boldsymbol{\psi} = -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{b}^{(m)}}\right)^T \left[\left.\frac{\partial \mathcal{J}}{\partial \mathbf{g}}\right|_{\mathbf{m}} + \left(\left.\frac{\partial \mathcal{J}}{\partial \mathbf{m}}\right|_{\mathbf{g}} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{m}}\right) \frac{\partial \mathbf{m}}{\partial \mathbf{g}}\right]^T, \tag{4.6}$$

where $\mathbf{g} = \{x, y, z\}$ and $\mathbf{m} = \{\xi_x, \xi_y, \xi_z, \eta_x, \eta_y, \eta_z, \zeta_x, \zeta_y, \zeta_z\}$. Solution of the mesh adjoint equations is similar to that of the flow adjoint equations. The preconditioned conjugate gradient method is used. Development was required to extend the $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{m}}$ term in Equation 4.4 to include high-order terms, as well as including contributions due to the dissipation model. Matrix-vector products are performed without the explicit storage of the metric Jacobian matrix, $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{m}}$. The right-hand side of Equation 4.5 is constructed using the complex-step method; this is rather expensive in general but because the control point mesh size is orders of magnitude smaller than the CFD mesh, the cost relative to a flow analysis is small.

4.1.2 Gradient Computation

The gradient of the Lagrangian function, $\frac{\partial \mathcal{L}}{\partial \mathbf{v}}$, is provided to SNOPT. The up-to-date flow and mesh adjoint variables are used to form the gradient vector:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}} + \sum_{i=1}^{m} \left(\boldsymbol{\lambda}^{(i)T} \frac{\partial \boldsymbol{\mathcal{M}}^{(i)}}{\partial \mathbf{v}}\right) + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{v}}. \tag{4.7}$$

4.2 Verification and Validation

Before a practical problem can be approached, the optimization algorithm must be extensively verified and validated. Verification tests the implementation of the algorithm at various checkpoints to ensure self-consistency, checking that the flux Jacobian matrix is consistent with the flow residual equations for instance. Validation requires testing the algorithm on a set of fundamentally well understood problems.

In this section, verification of the algorithm will be achieved by testing that the Jacobian matrices, $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}}$ and $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{m}}$, are computed in a way that is consistent with the flow residual, $\boldsymbol{\mathcal{R}}$. Since this does not test the portions of the gradient computation associated with the geometry parametrization and mesh movement, a directional derivative test will also be performed. A successful directional derivative test will indicate an analytically precise derivative computation.

Validation will be achieved by performing two well known tests: an inverse design problem in viscous laminar flow and a twist optimization in inviscid flow. Successful completion of these tests indicates that the gradient-based optimization algorithm is capable of minimizing a well-defined objective function, and application to practical problems is warranted. All of the verification and validation tests are performed using the Navier-Stokes equations on grid $\boldsymbol{G}_1$ from the three-dimensional viscous optimization test case in the subsequent chapter, except for the inviscid twist optimization case which is governed by the Euler equations and is performed on multiple grid levels. The twist validation will not be presented in this chapter as it is extensively studied to compare the efficiency of the methods in the next chapter.

4.2.1 Flow Jacobian Verification

The complex-step method allows for derivatives to be approximated to machine accuracy [55]. The complex-step approximation cannot be used with GCROT to solve the adjoint problem due to the

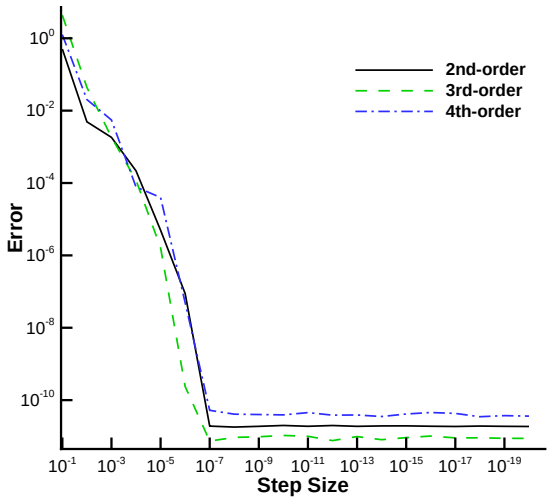


Figure 4.1: Transposed Jacobian-vector product test for each order of accuracy.

presence of the transpose operator on the left-hand side of Equation 4.3. It can, however, be used to verify the analytical transposed Jacobian-vector products by examining the following scalar quantity:

$$\begin{aligned}\mathbf{u}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \mathbf{v} &= (\mathbf{u}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \mathbf{v})^T \\ &= \mathbf{v}^T \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \right)^T \mathbf{u} \\ \text{error} &= \mathbf{u}^T \underbrace{\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \mathbf{v}}_{\text{approx.}} - \mathbf{v}^T \underbrace{\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \right)^T \mathbf{u}}_{\text{exact}},\end{aligned}$$

where $\mathbf{u}$ and $\mathbf{v}$ are random vectors, and the approximation is made through the complex-step method:

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \mathbf{v} \approx \text{Im} \left(\frac{\boldsymbol{\mathcal{R}}(\mathbf{Q} + i\epsilon \mathbf{v})}{\epsilon} \right), \tag{4.8}$$

where $i^2 = -1$. Assuming the analytical transposed Jacobian-vector products are correct, the only error is introduced by the complex-step approximation. The complex-step approximation is second-order with respect to its perturbation parameter, ϵ, so an $O(\epsilon^2)$ error should be measured which approaches machine precision when the step size becomes sufficiently small. Figure 4.1 shows the expected behaviour for the flux Jacobian test, verifying that the matrix-vector products used to solve the flow adjoint equations are indeed correct.

4.2.2 Mesh Adjoint Verification

The complex-step method is applied to verify that the implementation of the right-hand side of Equation 4.4 is correct. This simultaneously tests the objective function derivative and residual vector derivative with respect to the grid metrics. A separate test should be performed for each objective function. Figure 4.2 demonstrates that the analytical differentiation of the drag objective and residual vector with respect to the grid metrics has been performed correctly with an error less than machine precision. Similar tests were performed, with success, on the lift and lift-over-drag objectives.

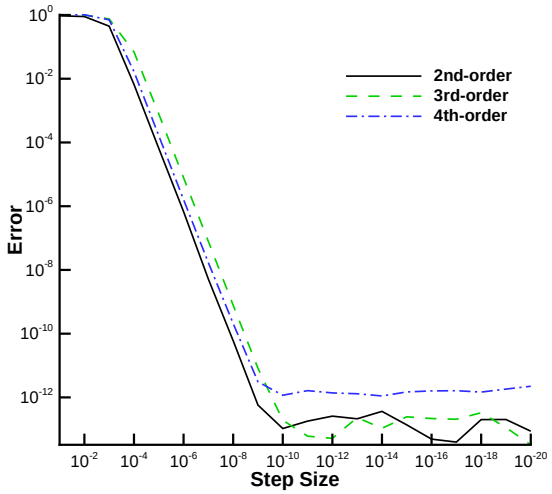


Figure 4.2: Complex-step test verifying the final increment in the mesh adjoint equations for a drag objective.

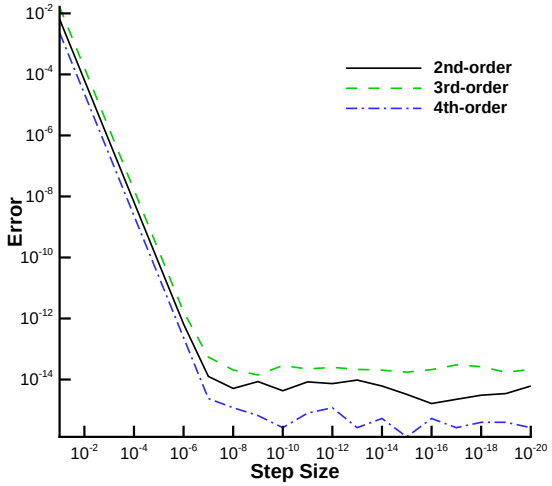


Figure 4.3: Complex-step verification of the drag derivative with respect to the flow variables

4.2.3 Objective Function Derivative, $\frac{\partial \mathcal{J}}{\partial \mathbf{Q}}$, Verification

The objective function derivative $\frac{\partial \mathcal{J}}{\partial \mathbf{Q}}$ seen on the right-hand side of Equation 4.3 is independently verified with the complex-step test. Once again, this needs to be done for all objective functions used within this work: drag, lift, and lift over drag.

Figure 4.3 demonstrates that the drag objective has been successfully differentiated to machine precision. Similar tests were performed with the other objectives used; all were successful.

4.2.4 Directional Derivative Verification

To verify the gradient, a directional derivative test is performed. The previous tests confirmed that the adjoint equations were set up properly, while this test confirms that the resultant adjoint variables lead to a correct derivative computation. This also tests the differentiation of the grid coordinates with respect to the B-spline control points, $\frac{\partial \mathbf{g}}{\partial \mathbf{b}^{(m)}}$, and the differentiation of the grid metrics with respect

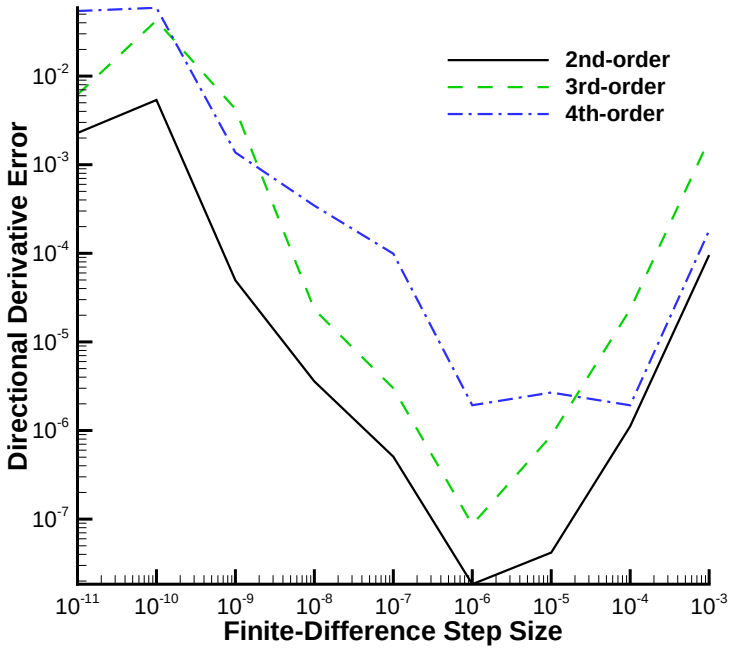


Figure 4.4: Directional derivative test for all orders of accuracy.

to the grid coordinates, $\frac{\partial \mathbf{m}}{\partial \mathbf{g}}$. The analytical gradient is computed using the adjoint method, with the directional derivative computed as

$$D_z \mathcal{J} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}} \mathbf{z}, \tag{4.9}$$

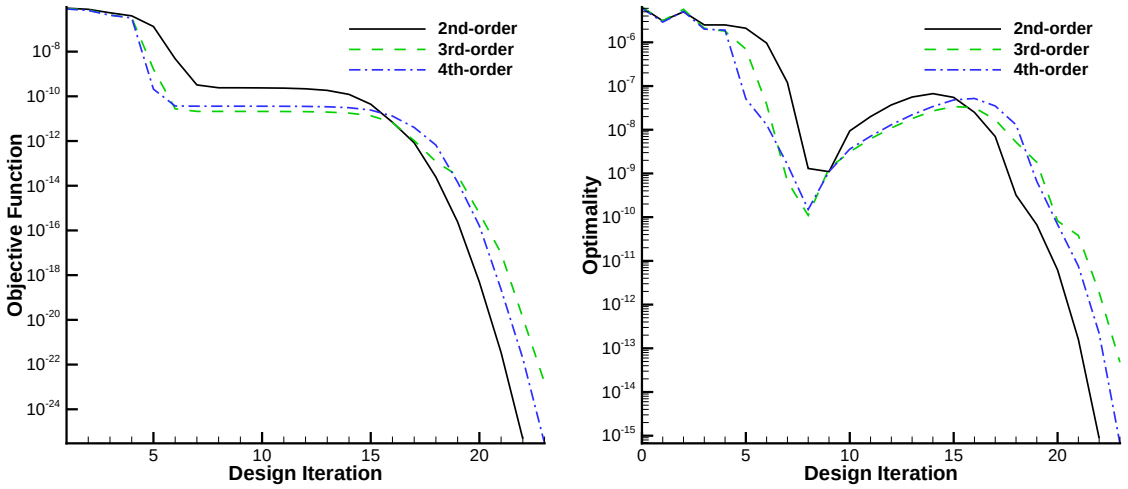
and direction z chosen to be

$$\mathbf{z}_i = \text{sign}\left[\left(\frac{\partial \mathcal{J}}{\partial \mathbf{v}} \right)_i \right]. \tag{4.10}$$

This particular choice of z gives a directional derivative equal to the L^1 norm of the gradient [29]. The analytical directional derivative is then compared to the second-order finite-difference approximation,

$$D_z \mathcal{J} = \frac{\mathcal{J}(\mathbf{v} + \epsilon \mathbf{z}) - \mathcal{J}(\mathbf{v} - \epsilon \mathbf{z})}{2\epsilon}. \tag{4.11}$$

The complex-step method used to verify the Jacobian cannot be used in this instance, as it would require complete complexification of the flow analysis algorithm. The results of the directional derivative test on the drag derivative can be seen in Figure 4.4. Similar results were obtained when applied to the other objectives used in this work. The minimum error for each case is below 10^{-5} in each case, and the error curve has a "V" shape; this indicates that the test was successful. For large step sizes truncation error in the finite-difference scheme dominates the error, while small step sizes leads to subtractive cancellation errors related to the finite precision of the stored variables. Since a second-order centred-difference is used in this case the truncation error should be quadratic with respect to the perturbation parameter, which can be roughly confirmed through inspection of Figure 4.4. If a distinct peak were not seen in Figure 4.4, it would indicate that the error in the analytical derivative is dominant, as demonstrated by Osusky [70].



(a) Merit function history for inverse design test.

(b) Optimiality history for inverse design test.

Figure 4.5: Convergence history of the inverse design test case.

4.2.5 Inverse Design

The inverse design case serves as further verification of the algorithm and is a starting point for validating the optimizer. It begins by performing a flow analysis on a baseline geometry and saving the surface pressure distribution. The optimization procedure is then started with one B-spline control point on the surface and the angle of attack perturbed, with the objective function defined as

$$\mathcal{J} = \frac{1}{2}\sum_{i=1}^{N_\eta}\sum_{j=1}^{N_\zeta}(H_\eta)_{i,i}\,(H_\zeta)_{j,j}\,(p_{i,j}-p_{i,j;\mathrm{targ}})^2\,\Delta A_{i,j}, \tag{4.12}$$

$$\Delta A_{i,j} = J^{-1}_{i,j}\sqrt{(\xi_x)^2_{i,j}+(\xi_y)^2_{i,j}+(\xi_z)^2_{i,j}}, \tag{4.13}$$

and is a numerical approximation to the surface integral of the squared difference between the current and target pressure distributions. In Equation 4.13 N_ξ and N_η are the number of surface nodes in the directions tangential to the surface, $\Delta A_{i,j}$ is the surface area element calculated at node (i,j), $p_{i,j}$ is the pressure at node (i,j), and $p_{i,j;\mathrm{targ}}$ is the target pressure at node (i,j). H accounts for the diagonal-norm integration rule, and the Jacobian factor, J^{-1}, accounts for the coordinate transformation when computing the surface integral. The goal of the inverse design case is to recover the unperturbed geometry by the action of forcing it to recover the pressure distribution of the unperturbed geometry.

The geometry considered is that used in Section 5.3, the three-dimensional viscous optimization case on grid $\boldsymbol{G}_1$. Each surface patch of the geometry is parametrized with 9 control points in the streamwise direction and 9 control points in the spanwise directions. The perturbed control point coordinates and angle of attack combine for a total of four design variables. Each flow analysis is solved to a relative residual drop of 10^{-10}, while the flow adjoint equations are solved to a relative tolerance of 10^{-8} and the mesh adjoint equations to a relative tolerance of 10^{-12}. The optimization procedure is carried out until the optimality condition is satisfied to a very tight absolute tolerance of 10^{-15}. The ILU fill level is chosen to be 3 for the flow analysis and flow adjoint solution to improve the preconditioning for higher-order problems.

The results of this validation study can be seen in Figure 4.5. The pressure distribution of the

Table 4.4: Inverse design wall time on 120 processors and design iterations required to meet optimality condition.

Order	**Wall Time (120 proc.) (hh:mm:ss)**	**Major Iterations**	**Function Evaluations**
Second-order	10:35:02	22	26
Third-order	11:42:43	23	27
Fourth-order	17:18:51	23	27

unperturbed geometry has been successfully recovered for each order of accuracy, and the inverse design test is successful. The perturbation parameter was chosen to be consistent across each inverse design problem so that the performance of the optimizer can be gauged. Table 4.4 shows the number of design iterations to reach convergence is nearly identical across all orders of accuracy, with each design cycle costing more for the higher-order methods.

4.3 Computational Cost of the Gradient Computation

The three-dimensional viscous laminar lift-constrained drag minization problem presented in Section 5.3 is examined to determine the relative cost of the gradient computation for the second-, third-, and fourth-order methods when using implicit matrix-vector products; the coarse mesh corresponds to $\boldsymbol{G}_1$ and the fine mesh corresponds to $\boldsymbol{G}_3$. The cost of the objective function evaluation is the sum of the cost to perform a mesh movement and the cost to perform a flow analysis. The cost of the gradient is divided into the cost to compute the drag gradient and the cost to compute the lift gradient. Within each gradient computation, the flow adjoint equations and mesh adjoint equations must be solved. The second-order explicit matrix-vector product gradient cost breakdown is shown in Table 4.5, and is an important comparison as the explicit matrix-vector products are expected to perform faster than implicit matrix-vector products. Tables 4.6–4.8 show the costs of each component of the objective function and gradient computation. It can be seen that the cost of the second- and third-order methods do not differ by much, but the fourth-order method is drastically increased in cost. It can also be seen that the majority of the cost increase is due to the increased stiffness of the fourth-order flow adjoint problem. The mesh movement time and mesh adjoint time are relatively constant since the control point mesh is identical for each case. Comparing the cost of the explicit and implicit matrix-vector product techniques to solve the flow adjoint problem reveals that the implicit technique requires 1.99 and 2.39 times the amount of CPU time compared to the explicit technique for $\boldsymbol{G}_1$ and $\boldsymbol{G}_3$ respectively. This highlights that the relative cost of the two matrix-vector product techniques is problem dependent, as a larger ratio was observed for the two-dimensional extruded problem. The relative cost is likely dependent on properties of the multi-block grid. Memory measurements were not made for this particular problem; however it is important to note that the $\boldsymbol{G}_4$ flow adjoint solution with explicit matrix-vector products failed due to excessive memory requirements. Clearly the implicit approach alleviates memory limitations of the current system for all orders of accuracy, including second-order problems.

Table 4.5: Breakdown of objective function and gradient CPU times for two different grid sizes with the second-order method using explicit matrix-vector products. Times are normalized by the total objective function evaluation time.

		grid size (nodes)			
		$\mathbf{8.230 \times 10^5}$		$\mathbf{2.927 \times 10^6}$	
		CPU time (120 proc.)		**CPU time** (120 proc.)	
component	**solve tol.**	**relative**	**absolute**	**relative**	**absolute**
mesh movement	(10^{-12})	0.0976		0.0271	
flow analysis	(10^{-10})	0.9024		0.9729	
objective function		**1.000**	**(393.4 s)**	**1.000**	**(1327.1 s)**
flow adjoint solution (drag)	(10^{-8})	0.1685		0.1816	
mesh adjoint solution (drag)	(10^{-12})	0.0678		0.0200	
drag gradient		**0.2363**		**0.2016**	
flow adjoint solution (lift)	(10^{-8})	0.2245		0.2504	
mesh adjoint solution (lift)	(10^{-12})	0.0611		0.0176	
lift gradient		**0.2856**		**0.2680**	
drag and lift gradient		**0.5219**	**(205.3 s)**	**0.4696**	**(623.2 s)**

Table 4.6: Breakdown of objective function and gradient CPU times for two different grid sizes with the second-order method using implicit matrix-vector products. Times are normalized by the total objective function evaluation time.

		grid size (nodes)			
		$\mathbf{8.230 \times 10^5}$		$\mathbf{2.927 \times 10^6}$	
		CPU time (120 proc.)		**CPU time** (120 proc.)	
component	**solve tol.**	**relative**	**absolute**	**relative**	**absolute**
mesh movement	(10^{-12})	0.0995		0.0265	
flow analysis	(10^{-10})	0.9005		0.9735	
objective function		**1.000**	**(382.8 s)**	**1.000**	**(1432.3 s)**
flow adjoint solution (drag)	(10^{-8})	0.3451		0.4017	
mesh adjoint solution (drag)	(10^{-12})	0.0637		0.0167	
drag gradient		**0.4099**		**0.4194**	
flow adjoint solution (lift)	(10^{-8})	0.4159		0.5426	
mesh adjoint solution (lift)	(10^{-12})	0.0637		0.0167	
lift gradient		**0.5157**		**0.5593**	
drag and lift gradient		**0.9255**	**(354.3 s)**	**0.9787**	**(1401.8 s)**

Table 4.7: Breakdown of objective function and gradient CPU times for two different grid sizes with the third-order method using implicit matrix-vector products. Times are normalized by the total objective function evaluation time.

		grid size (nodes)			
		$\mathbf{8.230 \times 10^5}$		$\mathbf{2.927 \times 10^6}$	
		CPU time (120 proc.)		**CPU time** (120 proc.)	
component	**solve tol.**	**relative**	**absolute**	**relative**	**absolute**
mesh movement	(10^{-12})	0.0938		0.0268	
flow analysis	(10^{-10})	0.9062		0.9732	
objective function		**1.000**	**(391.4 s)**	**1.000**	**(1431.7 s)**
flow adjoint solution (drag)	(10^{-8})	0.3998		0.4804	
mesh adjoint solution (drag)	(10^{-12})	0.0690		0.0184	
drag gradient		**0.4688**		**0.4988**	
flow adjoint solution (lift)	(10^{-8})	0.5508		0.6675	
mesh adjoint solution (lift)	(10^{-12})	0.0641		0.0168	
lift gradient		**0.6150**		**0.6843**	
drag and lift gradient		**1.0838**	**(424.2 s)**	**1.1831**	**(1693.9 s)**

Table 4.8: Breakdown of objective function and gradient CPU times for two different grid sizes with the fourth-order method using implicit matrix-vector products. Times are normalized by the total objective function evaluation time.

		grid size (nodes)			
		$\mathbf{8.230 \times 10^5}$		$\mathbf{2.927 \times 10^6}$	
		CPU time (120 proc.)		**CPU time** (120 proc.)	
component	**solve tol.**	**relative**	**absolute**	**relative**	**absolute**
mesh movement	(10^{-12})	0.0590		0.0135	
flow analysis	(10^{-10})	0.9410		0.9865	
objective function		**1.000**	**(617.1 s)**	**1.000**	**(2674.2 s)**
flow adjoint solution (drag)	(10^{-8})	1.2627		0.9417	
mesh adjoint solution (drag)	(10^{-12})	0.0433		0.0143	
drag gradient		**1.3059**		**0.9560**	
flow adjoint solution (lift)	(10^{-8})	1.3427		1.1503	
mesh adjoint solution (lift)	(10^{-12})	0.0413		0.0136	
lift gradient		**1.3841**		**1.1639**	
drag and lift gradient		**2.6900**	**(1660.0 s)**	**2.1199**	**(5669.1 s)**

Chapter 5

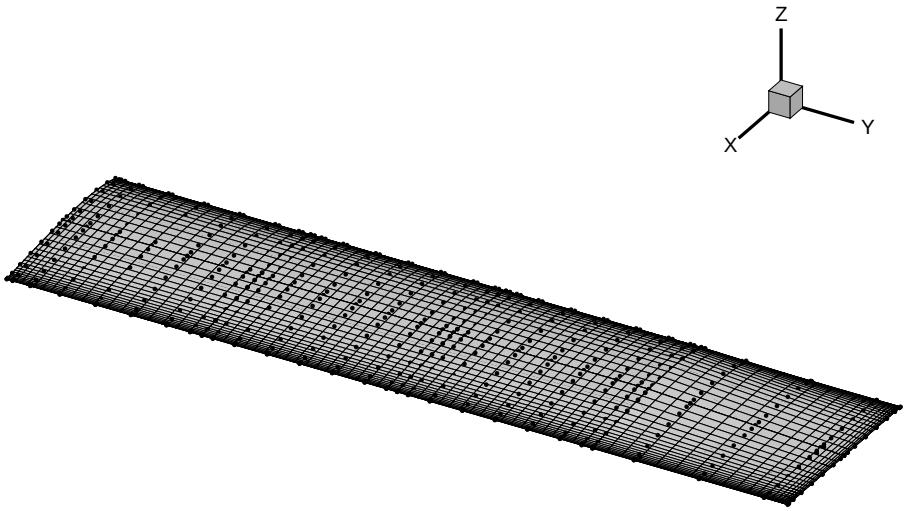
Aerodynamic Optimization with High-Order Spatial Discretization

This chapter demonstrates the efficiency of high-order spatial discretization within the context of numerical optimization. Efficiency gains for inviscid analysis have been demonstrated with the presented framework by Dias [24]; therefore one focus will be to examine the mesh refinement behaviour within the analysis algorithm for viscous laminar flows. The full optimization algorithm is applied to two and three-dimensional optimization problems, governed by the inviscid Euler equations and the laminar Navier-Stokes equations. An inviscid twist optimization is considered first, to provide validation for the algorithm and to examine the efficiency of high-order within inviscid optimization. A two-dimensional viscous laminar section optimization with several design variables and a three-dimensional viscous laminar span optimization with two effective design variables are studied to examine the efficiency of the optimization algorithm and study the role that accuracy plays on the final optimized shape.

5.1 Minimization of Induced Drag with Twist Design Variables

A rectangular wing with NACA 0012 sections and a semi-span of 4 chord units is optimized for minimum drag in inviscid subsonic flow conditions ($M = 0.3$); similar design cases have been studied by Hicken and Zingg [35], Telidetzki *et al.* [83], and Bisson *et al.* [12]. The surface of the wing is parametrized with 16 surface patches, each consisting of 9 control points in the streamwise direction and 9 control points in the spanwise direction except for the 4 wing tip patches which only have 5 control points in the spanwise direction to prevent the formation of a nonplanar wake. The initial control point distribution on the surface of the wing and the surface mesh can be seen in Figure 5.1a along with the mesh on the symmetry plane in Figure 5.2. Along the spanwise direction of the wing, the wing-tip patch varies from the root thickness to a zero-thickness line at the tip, as shown in Figure 5.1b; this was done to improve the grid quality to avoid metric Jacobian issues when using the high-order method. Design variables are chosen to be the z-coordinates of the B-spline control points and are coupled at each station so that sectional twist is an effective design variable. The first three sections, including the root section, are fixed to prevent nonplanar features. The sections in the wing-tip patch are constrained to have twist which varies linearly to prevent the formation of a winglet feature.

Five different meshes are used in this test case, with efforts made to keep all grids within the same



(a) Surface mesh and control points of untwisted geometry.

,

(b) Front view of geometry and control points.

Figure 5.1: Inviscid twist optimization: surface mesh and B-spline control points shown for the initial untwisted surface on mesh $\boldsymbol{G}_3$.

grid family by using parameter refinement. The hyperbolic tangent mesh parameters, A and b shown in Eq. 3.12, are computed on each block edge of the initial mesh. Mesh nodes are inserted and redistributed according to the hyperbolic tangent stretching function and using the stretching parameters computed initially; the volume mesh is subsequently generated using the B-spline mapping. Details of the five optimization grids along with the fine analysis mesh are shown in Table 5.1; each mesh is constructed with the far-field boundary 25 chord lengths from the surface in all directions. Linear theory predicts that an elliptical spanwise loading of a planar wing produces minimum drag [73]. This inviscid subsonic optimization test case serves to validate the algorithm through comparison to linear theory, as well as to quantify the efficiency of each method through comparison of optimal geometries on various levels of mesh refinement with various orders of accuracy. The span efficiency factor is given as

$$e = \frac{C_L^2}{\pi \Lambda C_{D_{\text{induced}}}}, \tag{5.1}$$

where Λ is the aspect ratio of the wing; in this case $\Lambda = 8$. Since a subsonic inviscid analysis is performed, C_D is assumed to be entirely lift-induced drag and can be used directly in Equation 5.1. Elliptical loading of the wing will lead to a span efficiency factor of 1.0, which is the best case scenario for a planar, inviscid, subsonic wing.

Algorithm inputs are selected to reflect both robustness and efficiency. Other than changing the SBP and dissipation operators to reflect the appropriate order of accuracy, the inputs are the same

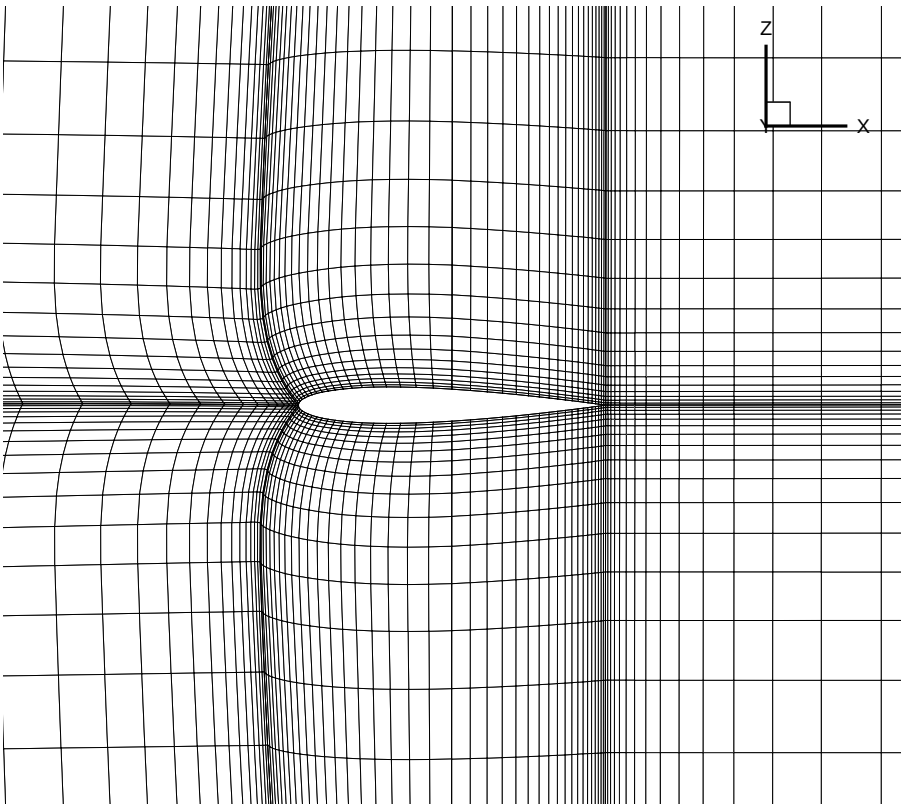


Figure 5.2: Inviscid twist optimization: details of mesh $\boldsymbol{G}_3$ in the plane of the wing root.

across all cases to facilitate a fair comparison. Flow analysis is performed with the pseudo-transient continuation parameters set to $a = 0.001$, $b = 1.2$, and $\beta = 1.5$. Artificial dissipation parameters are set to $\epsilon = 0.04$ for all directions with $V_n = 0.25$ and $V_l = 0.25$. A lift constraint of $C_L = 0.375$ is imposed, and the angle of attack is included as a design variable since the root section is fixed. The optimization algorithm solves the flow adjoint equations using the Krylov subspace method GCROT, and the linear sub-problems within flow analysis with FGMRES, both with ILU preconditioning, a fill-level of 3, and a dissipation lumping factor of 7.0 while limiting the size of the Krylov subspace to 70 vectors. The flow analysis switches to the inexact-Newton phase when the relative residual norm drops below 10^{-2}, and the flow analysis is solved to a relative tolerance of 10^{-10}. The flow adjoint equations are solved to a tolerance of 10^{-8} and the mesh adjoint equations to a tolerance of 10^{-12}. Convergence is reached when the optimality condition drops below 10^{-7} and a feasibility tolerance of 10^{-6} is satisfied. The angle of attack is adjusted on the initial geometry so that the lift constraint is satisfied with the initial design in each case.

First, to present evidence of validation for the optimization algorithm, the spanwise lift distribution of the optimized geometry on mesh $\boldsymbol{G}_5$ with the fourth-order method is examined and compared to an elliptical loading. Figure 5.3 shows a near elliptical loading of the wing, with an obvious deviation near the wing-tip. This deviation has been attributed to side-edge separation [35]. A span efficiency factor of 0.99743 has been observed for this case when evaluated with the fourth-order method on the fine mesh, $\boldsymbol{G}_F$, conforming with conclusions from linear theory and concluding a positive validation exercise. Similar spanwise lift distributions are observed for all cases; details can be seen in Appendix C.

Table 5.1: Inviscid twist optimization: details of grids used in refinement study.

Grid	Mesh Dimensions	Nodes	Off-wall Spacing	Leading/Trailing Edge Spacing
$\boldsymbol{G}_1$	120 blocks $\times$ 13^3	236 640	1.18E-2	1.24E-2
$\boldsymbol{G}_2$	120 blocks $\times$ 16^3	491 520	8.99E-3	9.49E-3
$\boldsymbol{G}_3$	120 blocks $\times$ 20^3	960 000	6.79E-3	7.22E-3
$\boldsymbol{G}_4$	120 blocks $\times$ 25^3	1 875 000	5.18E-3	5.55E-3
$\boldsymbol{G}_5$	120 blocks $\times$ 31^3	3 574 920	4.05E-3	4.34E-3
$\boldsymbol{G}_F$	960 blocks $\times$ 41^3	66 164 160	1.46E-3	1.54E-3

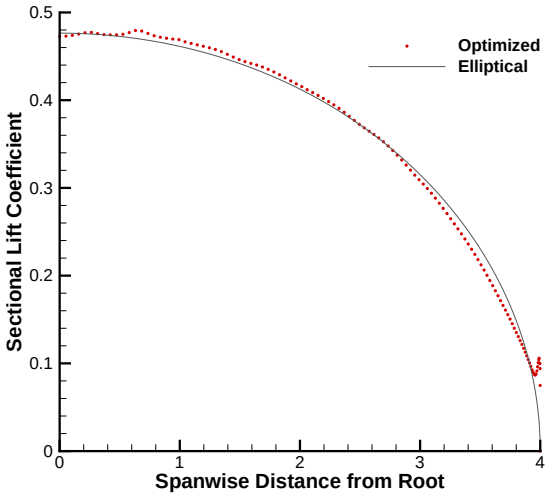


Figure 5.3: Inviscid twist optimization: spanwise lift distribution of optimized geometry on $\boldsymbol{G}_5$ with the fourth-order method is shown in comparison to an elliptical spanwise lift distribution

Cost of the high-order algorithm must now be evaluated. Table 5.2 shows the run time for each optimization. It can be seen from this information that the second-order and third-order methods have very similar costs, while the fourth-order method has a drastically increased cost. The number of optimization iterations on each grid remains fairly similar across all orders of accuracy, with an increase in optimization iterations with mesh refinement. Table 5.2 also shows that each case converged to below the requested optimality and feasibility tolerances.

One major goal of this thesis is to examine the effect that high-order spatial discretization has on the solution of the flow adjoint problem. Figure 5.4 shows the convergence histories of the linear flow adjoint equations. The second- and third-order methods converge in about the same time, independent of the grid, while the fourth-order method takes much longer. One possible explanation for this is that the fourth-order method uses an undivided approximation to the fourth-derivative to construct D_d in the dissipation model, while the second- and third-order methods use an undivided approximation to the third-derivative. It is unclear if the first-derivative or dissipation model contribute more toward the stiffness in the solution of the adjoint equations.

The aerodynamic performance of each optimized geometry is summarized in Table 5.3. With respect to analysis on their native meshes, each method is able to show a sizeable reduction in the drag coefficient.

Table 5.2: Inviscid twist optimization: summary of optimization data

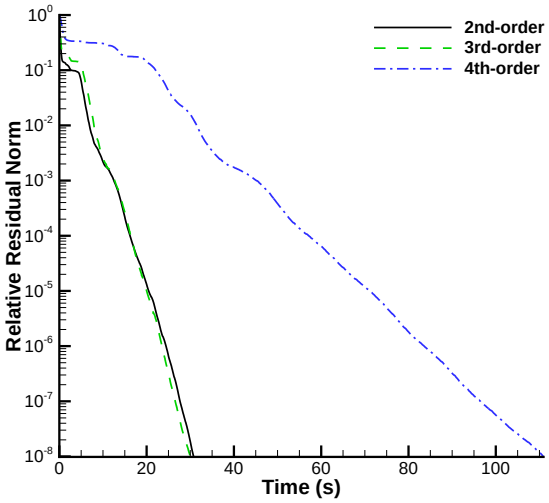
Grid	Wall Time (120 proc.) (hh:mm:ss)	Major Iterations	Function Evaluations	Final Feasibility	Final Optimality
Second-order					
$\boldsymbol{G}_1$	02:27:47	35	37	1.1E-12	7.4E-8
$\boldsymbol{G}_2$	05:10:55	45	47	7.8E-13	5.2E-8
$\boldsymbol{G}_3$	09:27:16	45	47	2.2E-13	8.6E-8
$\boldsymbol{G}_4$	26:18:42	59	61	2.7E-13	8.2E-8
$\boldsymbol{G}_5$	75:25:30	72	74	1.6E-12	9.6E-8
Third-order					
$\boldsymbol{G}_1$	03:12:15	47	49	1.7E-13	8.7E-8
$\boldsymbol{G}_2$	04:48:45	43	45	5.9E-13	6.0E-8
$\boldsymbol{G}_3$	10:24:32	52	54	1.2E-13	6.6E-8
$\boldsymbol{G}_4$	28:11:54	69	71	1.7E-13	5.4E-8
$\boldsymbol{G}_5$	74:05:52	79	81	5.2E-14	9.3E-8
Fourth-order					
$\boldsymbol{G}_1$	10:29:23	53	55	5.2E-13	7.3E-8
$\boldsymbol{G}_2$	08:05:28	52	54	7.9E-13	2.1E-8
$\boldsymbol{G}_3$	37:38:03	62	64	5.5E-13	5.3E-8
$\boldsymbol{G}_4$	120:08:06	82	84	7.5E-13	3.1E-8
$\boldsymbol{G}_5$	308:09:09	89	91	3.4E-12	3.6E-8

To make a clear comparison, the optimized surfaces are re-analysed on a fine mesh, $\boldsymbol{G}_F$, with the fourth-order method to improve accuracy. Since the lift constraint is not met exactly when evaluated on the fine mesh, and to properly compare to linear theory, the span efficiency factor, e, is used to compare the aerodynamic performance of the optimized geometries. When evaluated with the fourth-order method on the fine analysis mesh, all configurations achieve a span efficiency factor very close to 1.0.

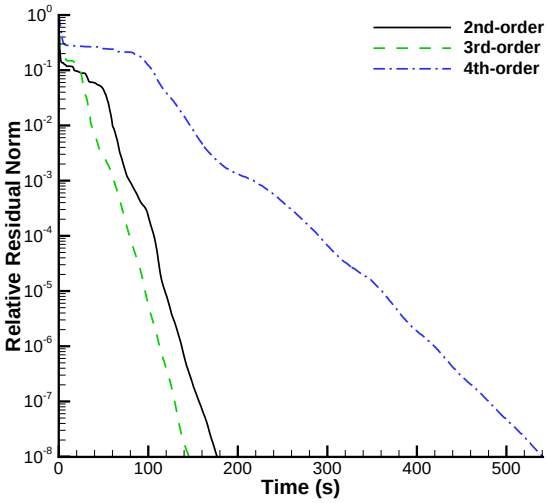
It is possible that this test case is not sufficiently complex to demonstrate the need for accurate analysis. In the context of inviscid optimization, Chernukhin and Zingg have also demonstrated an example of mesh refinement leading the similar optimized shapes [14], with such conclusions supported by Telidetzki *et al.* for inviscid optimization of a transonic airfoil in two-dimensions [83]. Only pressure drag is computed in the force integration under inviscid conditions, so it is possible that the inaccuracies are consistent under- or over-estimates, causing the optimizer to produce similar geometries. When considering viscous flow, mesh refinement may be more critical to obtaining the correct optimized shape as improving accuracy may have a large impact on the trade-off relationship between induced and viscous drag.

Table 5.3: Inviscid twist optimization: aerodynamic performance of optimized geometries

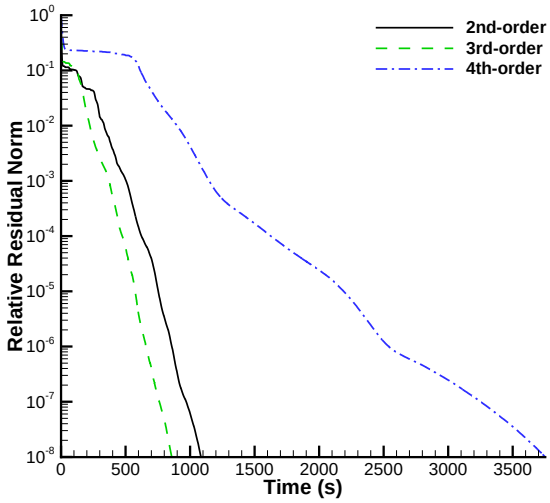
Grid	**Initial C_D** ($\times 10^{-4}$)	**Optimized C_D**	**Optimized $(C_L)_{G_F}$**	**Optimized $(C_D)_{G_F}$** ($\times 10^{-4}$)	**e**
Second-order					
G_1	225.21	204.43	1.4978	223.55	0.99972
G_2	215.68	192.35	1.4968	223.55	0.99907
G_3	217.29	189.92	1.4972	223.92	0.99764
G_4	224.52	194.83	1.4978	224.24	0.99663
G_5	226.86	201.88	1.4983	224.42	0.99619
Third-order					
G_1	290.33	265.69	1.4957	224.03	0.99620
G_2	262.84	237.61	1.4994	224.87	0.99491
G_3	252.70	228.16	1.5013	225.24	0.99457
G_4	251.59	226.75	1.5018	225.21	0.99499
G_5	248.54	226.98	1.5017	225.06	0.99562
Fourth-order					
G_1	167.26	140.88	1.5018	227.16	0.98647
G_2	212.78	189.53	1.5007	225.58	0.99261
G_3	231.69	211.42	1.5003	225.08	0.99458
G_4	239.74	222.38	1.4992	224.55	0.99618
G_5	240.91	226.31	1.4984	224.15	0.99743



(a) Flow adjoint convergence history on $\boldsymbol{G}_1$ at first design iteration.



(b) Flow adjoint convergence history on $\boldsymbol{G}_3$ at first design iteration.



(c) Flow adjoint convergence history on $\boldsymbol{G}_5$ at first design iteration.

Figure 5.4: Inviscid twist optimization: convergence histories on various grid levels for the flow adjoint equations with a drag objective at the first design iteration.

5.2 Aerodynamic Optimization in Two Dimensions with Viscous Effects

This test case examines the efficiency of the high-order optimization algorithm when applied in two-dimensional viscous laminar flow conditions. The flow is subsonic with a Mach number of 0.3 and a Reynolds number of 500 based on the chord length, and the initial shape is a NACA0012 airfoil. Due to limitations within the optimization algorithm, two-dimensional geometries cannot be directly considered for optimization. To circumvent this issue, the two-dimensional (x-z plane) grid is extruded in the y-direction 1 chord length. The number of nodes in the y-direction is chosen to be 19 to accommodate the minimum size of the fourth-order second-derivative operator. The geometry is parametrized with 9 control points in the streamwise direction and 5 in the extruded direction. Linear constraints are applied so that design variables along extruded airfoil mimic the design variables at the root directly. The z-coordinates of the B-spline control points are chosen as the design variables in this case, with angle of attack constant at 4 degrees to avoid unsteady flows. The objective of the optimization is to maximize $\frac{C_L}{C_D}$. There is a thickness constraint on the geometry such that the vertical separation of the control points cannot become smaller than 50% or larger than 200% of the initial separation of the NACA0012 control points. The upper control points are also restricted to not cross the chord line of $z = 0$.

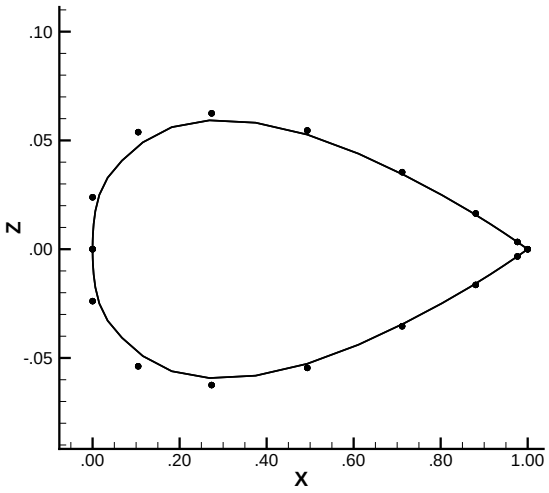
Srinath and Mittal have shown that the optimization of a two-dimensional NACA0012 airfoil at Reynolds numbers ranging from 10 to 500 is very sensitive to the Reynolds number [80]. This sensitivity to the flow conditions could provide incentive for accurate analysis and could make a case for high-order spatial discretization to improve the efficiency of the optimization. Derksen *et al.* have shown with particle image velocimetry experiments that flow over a NACA0012 section at a Reynolds number of 5000 is steady for an angle of attack less than 12 degrees [23]. Therefore it is reasonable to assume that flow will be steady under the provided operating conditions throughout the optimization. The details of the five meshes used in the optimization mesh refinement study are shown in Table 5.4, along with the two-dimensional fine analysis mesh; the same mesh refinement strategy used in the inviscid twist optimization study is used here. The surface control point distribution is shown in Figure 5.5a and the mesh near the aerodynamic surface is shown in Figure 5.5b.

Algorithm inputs are similar to those of the inviscid twist optimization case; the pseudo-transient time parameters are $a = 0.001$, $b = 1.2$, and $\beta = 1.6$. The analysis algorithm switches to the inexact-Newton phase when the relative residual has been reduced by a factor of 10^{-4}. The dissipation parameters are increased to improve stability: $\epsilon = 0.06$ in all directions while $V_l = 0.35$ and $V_n = 0.35$. The dissipation lumping factor is increased to 20.0 in this case to improve the preconditioning; attempts with lower dissipation lumping factors caused the solution algorithm to stall. There are no nonlinear constraints in this optimization. The flow analysis and mesh movement are solved to a relative tolerance of 10^{-10} and 10^{-12} respectively; the flow and mesh adjoint equations are solved to a relative tolerance of 10^{-8} and 10^{-12} respectively. Convergence is reached when the optimality falls below 10^{-7}. The convergence history of each optimization problem is summarized in Table 5.5; it can be seen that some of the designs have not converged; however a reduction in the objective function is observed. This is mostly due to the decreased robustness of the high-order algorithm, as the flow analysis does not converge with some of the more challenging geometries.

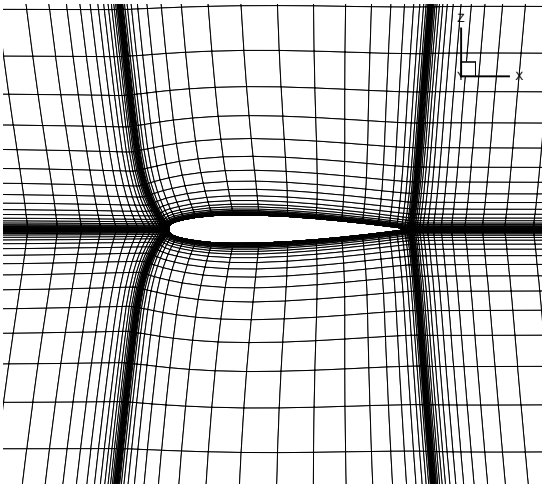
Table 5.6 shows the aerodynamic performance of each geometry evaluated with the fourth-order method on the fine mesh, $\boldsymbol{G}_F$. The final optimized geometries are shown in Appendix C. When analysed

Table 5.4: Viscous laminar section optimization: details of two-dimensional grids used in refinement study.

Grid	Mesh Dimensions	Nodes	2D Equivalent Nodes	Off-wall Spacing	Leading/Trailing Edge Spacing
$\boldsymbol{G}_1$	20 blocks $\times$ $19^2 \times 19$	137 180	7 220	5.65E-4	2.57E-4
$\boldsymbol{G}_2$	20 blocks $\times$ $23^2 \times 19$	201 020	10 580	4.35E-4	1.09E-4
$\boldsymbol{G}_3$	20 blocks $\times$ $29^2 \times 19$	319 580	16 820	3.26E-4	4.57E-5
$\boldsymbol{G}_4$	20 blocks $\times$ $37^2 \times 19$	520 220	27 380	2.44E-4	2.55E-5
$\boldsymbol{G}_5$	20 blocks $\times$ $46^2 \times 19$	804 080	42 320	1.90E-4	1.66E-5
$\boldsymbol{G}_F$	20 blocks $\times$ $61^2 \times 1$	74 420	74 420	1.36E-4	5.72E-5



(a) Surface control point distribution on grid $\boldsymbol{G}_3$.



(b) Mesh details near aerodynamic surface on grid $\boldsymbol{G}_3$.

Figure 5.5: Viscous laminar section optimization: grid and surface control point distributions on $\boldsymbol{G}_3$

on grid $\boldsymbol{G}_F$ with the fourth-order method, the initial NACA 0012 airfoil has $C_L = 0.218921$, $C_D = 0.179524$, leading to an initial lift to drag ratio of 1.219. Aside from a few anomalies, the general trend seen is that a more accurate analysis during the optimization leads to a better objective function when re-evaluated on the fine mesh. It appears that the accuracy of the functionals can shift the design space enough that the multi-modality of the design space begins to influence the optimized shapes. The result from the second-order method on grid $\boldsymbol{G}_1$ leads to a fundamentally different converged shape and in fact performs significantly better in the fine mesh analysis. The shape produced by the fourth-order method on grid $\boldsymbol{G}_5$ also appears to be converging to a different final shape, although the result is not converged. The fact that the geometry from $\boldsymbol{G}_1$ with the second-order method performs better than the others is likely an anomaly, as the optimized geometry in a multi-modal design problem is heavily influenced by the initial geometry. Srinath and Mittal studied a similar design problem and also observed multi-modality in the design space [80].

The relative cost of each method is difficult to interpret in this design example as there are several failed function evaluations which add to the total wall time. For the cases which converge well, third- and fourth-order appear to be competitive in efficiency with second-order. There are several scenarios

Table 5.5: Viscous laminar section optimization: summary of optimization data

Grid	Walltime (20 proc.) (hh:mm:ss)	Major Iterations	Function Evaluations	Initial Optimality	Final Optimality
Second-order					
$\boldsymbol{G}_1$	03:41:02	18	23	8.0E-2	2.2E-8
$\boldsymbol{G}_2$	07:03:56	16	29	8.8E-2	9.8E-7
$\boldsymbol{G}_3$	09:31:19	17	25	1.0E-1	4.7E-10
$\boldsymbol{G}_4$	12:20:24	15	20	1.1E-1	5.1E-8
$\boldsymbol{G}_5$	21:18:55	16	21	1.2E-1	2.5E-8
Third-order					
$\boldsymbol{G}_1$	07:06:25	16	32	1.1E-1	3.6E-6
$\boldsymbol{G}_2$	11:16:22	20	30	1.0E-1	3.8E-7
$\boldsymbol{G}_3$	30:16:05	28	56	1.2E-1	1.4E-6
$\boldsymbol{G}_4$	70:08:04	29	75	1.2E-1	1.5E-1
$\boldsymbol{G}_5$	96:01:10	25	62	1.2E-1	1.3E-1
Fourth-order					
$\boldsymbol{G}_1$	09:31:05	17	26	9.9E-2	5.0E-7
$\boldsymbol{G}_2$	08:04:50	13	16	1.1E-1	5.1E-8
$\boldsymbol{G}_3$	72:01:09	25	67	1.2E-1	2.5E-1
$\boldsymbol{G}_4$	24:32:00	14	17	1.2E-1	3.8E-8
$\boldsymbol{G}_5$	96:00:07	21	42	1.2E-1	3.7E-1

where the high-order method on a coarse mesh achieves a design which has a superior objective function for a reduced wall time compared to the second-order method; the objective function is evaluated on the fine mesh with the fourth-order method. The converged optimization problems are sorted by wall time in Table 5.7. Comparison of the results from the fourth-order method on $\boldsymbol{G}_4$ with the result from second-order on $\boldsymbol{G}_5$ shows that the fourth-order method can achieve a design with improved performance for roughly the same computational cost. The fourth-order method on $\boldsymbol{G}_2$ offers a better design than the second-order method on $\boldsymbol{G}_3$ for a significantly reduced computational cost.

To further study the influence of multi-modality, the optimized shape from the second-order optimization on $\boldsymbol{G}_1$ was used as the initial design for each optimization. Similar shapes to that of the second-order $\boldsymbol{G}_1$ result were obtained for each optimization which converged, confirming that the design space is multi-modal and that the accuracy of the analysis can influence the final design. Table 5.8 shows the aerodynamic performance of designs which successfully converged. It appears that a coarse mesh optimization using fourth-order analysis lead to improved designs over a finer mesh optimization with second-order analysis. High-order methods were unable to obtain a converged flow solution on the initial geometry for some of the grid levels; these cases are marked with DNC (Did Not Converge) in Table 5.8.

In conclusion, this design example demonstrates several scenarios where high-order methods can improve the final design for similar CPU costs. It also highlights robustness issues with the high-order flow-analysis. It was found that the accuracy of the discretization can change the final design of the airfoil, and a mutli-modality study confirms this hypothesis.

Table 5.6: Viscous laminar section optimization: aerodynamic performance of optimized sections. Cases which are not fully converged are marked with an asterisk.

Grid	$\left(\frac{C_L}{C_D}\right)_{init}$	$\left(\frac{C_L}{C_D}\right)_{final}$	$\left(\frac{C_L}{C_D}\right)_{final}$ **on Fine Mesh**
Second-order			
$\boldsymbol{G}_1$	1.471	3.212	2.810
$\boldsymbol{G}_2$	1.371	2.904	2.598
$\boldsymbol{G}_3$	1.298	2.764	2.618
$\boldsymbol{G}_4$	1.249	2.696	2.654
$\boldsymbol{G}_5$	1.236	2.684	2.675
Third-order			
$\boldsymbol{G}_1$	1.404	2.805	2.501
$\boldsymbol{G}_2$	1.335	2.779	2.565
$\boldsymbol{G}_3$	1.274	2.717	2.625
$\boldsymbol{G}_4$ *	1.245	2.460	2.434
$\boldsymbol{G}_5$ *	1.233	2.544	2.599
Fourth-order			
$\boldsymbol{G}_1$	1.309	2.937	2.624
$\boldsymbol{G}_2$	1.271	2.781	2.643
$\boldsymbol{G}_3$ *	1.237	2.666	2.659
$\boldsymbol{G}_4$	1.224	2.676	2.684
$\boldsymbol{G}_5$ *	1.222	2.787	2.960

Table 5.7: Viscous laminar section optimization: aerodynamic performance ordered by total computational cost. Improvement of the design computed on the fine mesh relative to the initial NACA0012 computed on the fine mesh is included.

Design Case	**Walltime (20 proc.) (hh:mm:ss)**	$\left(\frac{C_L}{C_D}\right)_{final}$ **on Fine Mesh**	**% Improvement**
$(\boldsymbol{G}_1)_{2nd-order}$	03:41:02	2.810	130.5
$(\boldsymbol{G}_2)_{2nd-order}$	07:03:56	2.598	113.1
$(\boldsymbol{G}_1)_{3rd-order}$	07:06:25	2.501	105.2
$(\boldsymbol{G}_2)_{4th-order}$	08:04:50	2.643	116.8
$(\boldsymbol{G}_1)_{4th-order}$	09:31:05	2.624	115.2
$(\boldsymbol{G}_3)_{2nd-order}$	09:31:19	2.618	114.8
$(\boldsymbol{G}_2)_{3rd-order}$	11:16:22	2.565	110.4
$(\boldsymbol{G}_4)_{2nd-order}$	12:20:24	2.654	117.7
$(\boldsymbol{G}_5)_{2nd-order}$	21:18:55	2.675	119.4
$(\boldsymbol{G}_4)_{4th-order}$	24:32:00	2.684	120.2
$(\boldsymbol{G}_3)_{3rd-order}$	30:16:05	2.625	115.3

Table 5.8: Viscous laminar section optimization: aerodynamic performance of optimized sections in multi-modality study. The initial shape is that of the second-order optimization in grid $\boldsymbol{G}_1$, as shown in Figure C.4a. Cases which are not fully converged are marked with an asterisk. Cases which could not obtain a converged flow analysis on the initial geometry are marked with DNC.

Grid	$\left(\frac{C_L}{C_D}\right)_{init}$	$\left(\frac{C_L}{C_D}\right)_{final}$	$\left(\frac{C_L}{C_D}\right)_{final}$ **on Fine Mesh**
Second-order			
$\boldsymbol{G}_1$	3.212	3.212	2.810
$\boldsymbol{G}_2$	2.999	3.023	2.876
$\boldsymbol{G}_3$	2.852	2.924	2.928
$\boldsymbol{G}_4$	2.810	2.919	2.954
$\boldsymbol{G}_5$	2.811	2.939	2.966
Third-order			
$\boldsymbol{G}_1$	DNC	—	—
$\boldsymbol{G}_2$	DNC	—	—
$\boldsymbol{G}_3$	DNC	—	—
$\boldsymbol{G}_4$	DNC	—	—
$\boldsymbol{G}_5$	DNC	—	—
Fourth-order			
$\boldsymbol{G}_1$	2.966	3.037	2.901
$\boldsymbol{G}_2$	2.875	2.976	2.954
$\boldsymbol{G}_3$ *	2.784	2.954	2.968
$\boldsymbol{G}_4$	2.787	2.938	2.972
$\boldsymbol{G}_5$	DNC	—	—

5.3 Trading Viscous and Induced Drag in Three Dimensions

This test case focuses on the trade-off between viscous and induced drag for a three-dimensional wing. The same NACA0012 geometry as the inviscid twist optimization is used, with a refinement in off-wall spacing. Increased mesh densities were chosen in this case as the minimum number of nodes in each direction is limited by the fourth-order second-derivative operator, as shown in Table 3.1; details of each mesh are shown in Table 5.9. The surface is parametrized with 16 surface patches and 9 control points in the streamwise direction and 9 control points in the spanwise direction on each surface patch; the control point distribution is shown in Figure 5.6 and the mesh at the symmetry plane of $\boldsymbol{G}_2$ is shown in Figure 5.7. Along with angle of attack, the y-coordinates of the control points serve as design variables but are coupled through the use of linear constraints so that the semi-span of the wing is the only effective geometric design variable. This is a lift-constrained drag minimization, with a drag objective function, $\mathcal{J} = C_D * S$, and a lift constraint of $C_L * S = 0.5$. The chord length is fixed in this case, so the projected area, S, is directly proportional to the semi-span design variable. The flow conditions were carefully chosen to avoid unsteady flow features, and so that the initial design has similar contributions of friction drag and pressure drag; for this design case $Ma = 0.3$ and $Re = 1500$, and initially $\alpha = 4°$; the initial design does not satisfy the equality lift constraint, and is significantly higher than the lift-target on each mesh as shown in Table 5.10. Once again, based on the experimental results from Derksen [23], the flow is expected to be steady in these flow conditions as long as the angle of attack does not become much higher than 10 degrees.

First a mesh refinement study is performed on the initial NACA 0012 wing, to appreciate the accuracy of each grid and order of accuracy combination. Table 5.10 shows the lift and drag on the NACA 0012 wing computed on each grid and with each order of accuracy and compares to the lift and drag computed on the fine mesh, $\boldsymbol{G}_F$, evaluated with the fourth-order method. Figure 5.8 shows the asymptotic behaviour of mesh refinement on the accuracy of lift and drag. Figure 5.9 compares the efficiency of each method at accurately predicting lift and drag. It can be seen from the figures that high-order methods produce lower error when compared to the second-order method when the same mesh is used, and as a result can produce lift and drag to a specified accuracy with a lower computational cost. The order of accuracy of each method is computed with a least-squares fitting of a power function on the absolute value of the data, as shown by dotted lines in Figures 5.8 and 5.9. Table 5.11 shows the order of accuracy for each method and the error is taken as the least squares fitting error. Close to the expected order of accuracy is seen for the C_D data, while C_L demonstrates expected trends.

The flow analysis algorithm input parameters are chosen again for a good mix of stability and speed. The pseudo-transient continuation parameters are set to $a = 0.001$, $b = 1.2$, and $\beta = 1.6$. The fourth-order method requires slightly more conservative time-stepping parameters for stability, so $b = 1.15$ for the fourth-order optimization cases. The analysis algorithm switches to the inexact-Newton phase when the relative residual has been reduced by a factor of 10^{-4}. The dissipation parameters are the same as the previous viscous case: $\epsilon = 0.06$ in all directions and $V_l = V_n = 0.35$. A dissipation lumping factor of 10.0 is chosen, and a fill-level of 3 is used for the preconditioning in the flow analysis and the flow adjoint equations, and the Krylov subspace is limited to 70 vectors. The relative tolerances for the flow analysis, mesh movement, flow adjoint, and mesh adjoint equations are 10^{-10},10^{-12},10^{-8}, and 10^{-12} respectively. The optimization algorithm continues until the feasibility falls below 10^{-6} and the optimality is below 10^{-7}. Table 5.12 shows the wall time required to converge each optimization. The final feasibility and optimality are shown to demonstrate that each design represents a fully converged

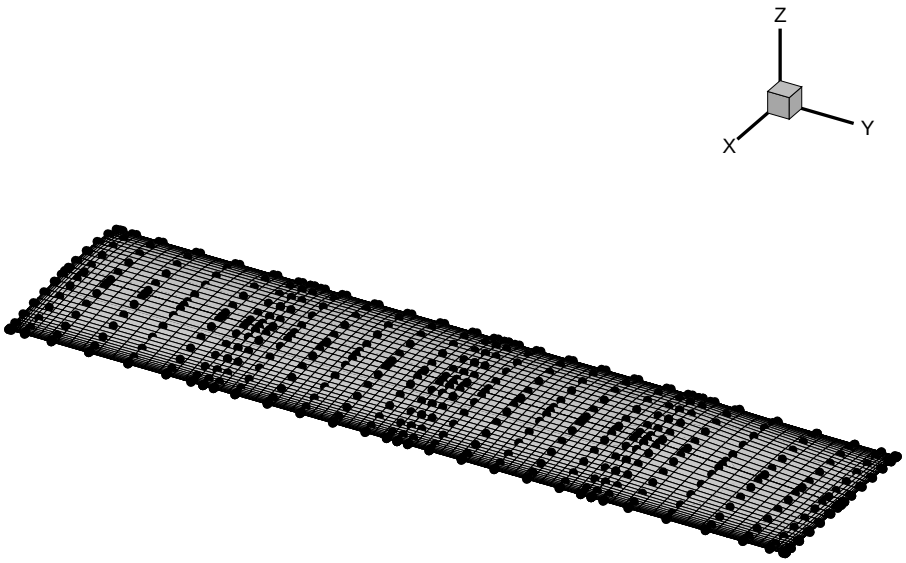


Figure 5.6: Viscous laminar span optimization: surface control point distribution and surface mesh on grid $\boldsymbol{G}_2$.

Table 5.9: Viscous laminar span optimization: details of grids used in refinement study.

Grid	**Mesh Dimensions**	**Nodes**	**Off-wall Spacing**
$\boldsymbol{G}_1$	120 blocks $\times$ 19^3	823 080	4.44E-4
$\boldsymbol{G}_2$	120 blocks $\times$ 23^3	1 460 040	3.66E-4
$\boldsymbol{G}_3$	120 blocks $\times$ 29^3	2 926 680	2.92E-4
$\boldsymbol{G}_4$	120 blocks $\times$ 37^3	6 078 360	2.22E-4
$\boldsymbol{G}_F$	960 blocks $\times$ 42^3	71 124 480	8.15E-5

solution to the PDE-constrained optimization problem.

The performance of each optimized geometry is evaluated on the fine mesh with the fourth-order method, and results are summarized in Table 5.14. The results show that the high-order methods are more consistent at meeting the lift constraint with the fine mesh analysis. This has implications on the objective value, as drag has a direct dependence on lift through lift-induced drag. In order to compare the results objectively, the fine mesh is corrected to meet the intended lift constraint by adjusting the angle of attack. This was not possible for some of the geometries as a very high angle of attack would be required to meet the constraint, likely leading to unsteady flow. Table 5.15 shows the adjusted objective, and demonstrates the effectiveness of high-order when used in optimization. While the second-order method performs well on a coarse mesh, the objective increased with mesh refinement. This result is undesirable as mesh refinement should improve the accuracy of the optimization. The high-order methods demonstrate the correct behaviour with mesh refinement, and lead to improved designs when the lift constraint is corrected on the fine mesh.

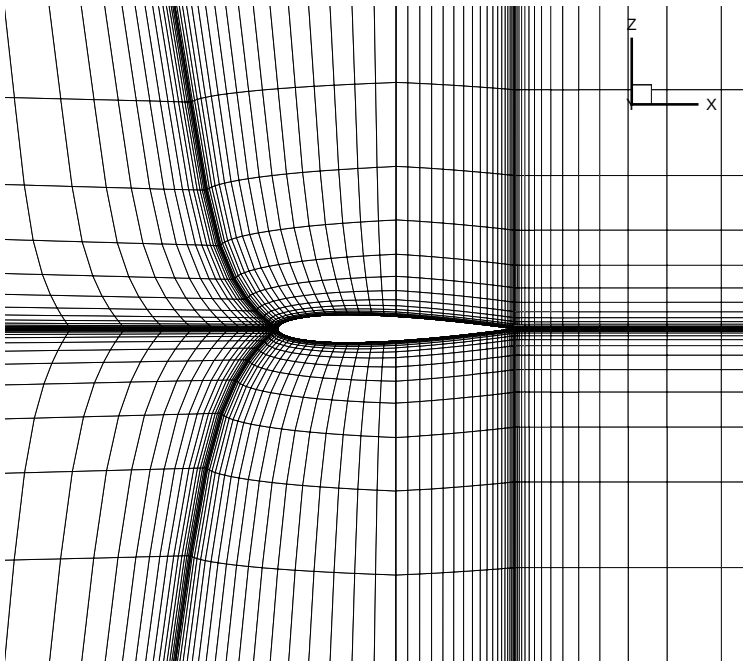
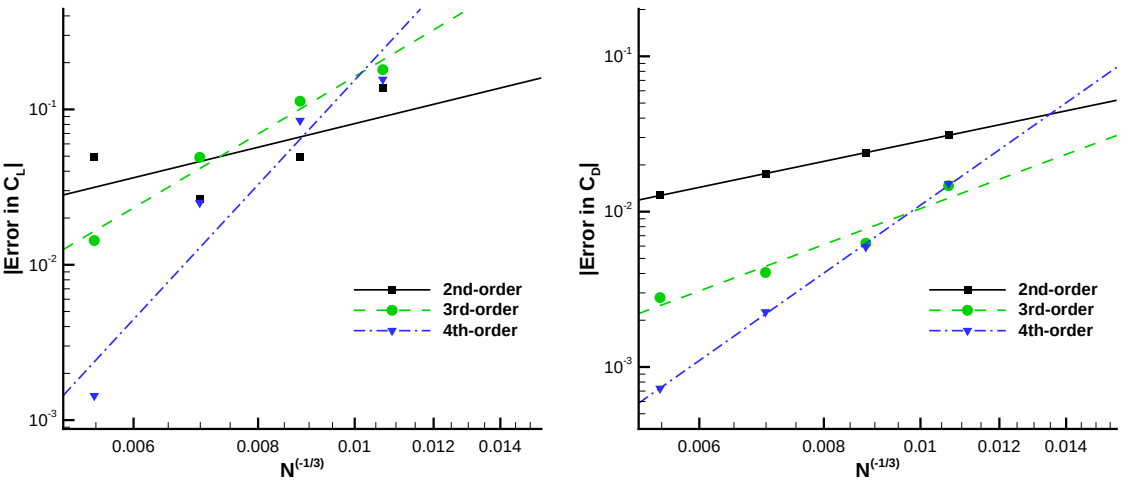


Figure 5.7: Viscous laminar span optimization: details of mesh $\boldsymbol{G}_2$ in the plane of the wing root.

Table 5.10: Viscous laminar flow analysis: flow analysis mesh refinement study of initial geometry.

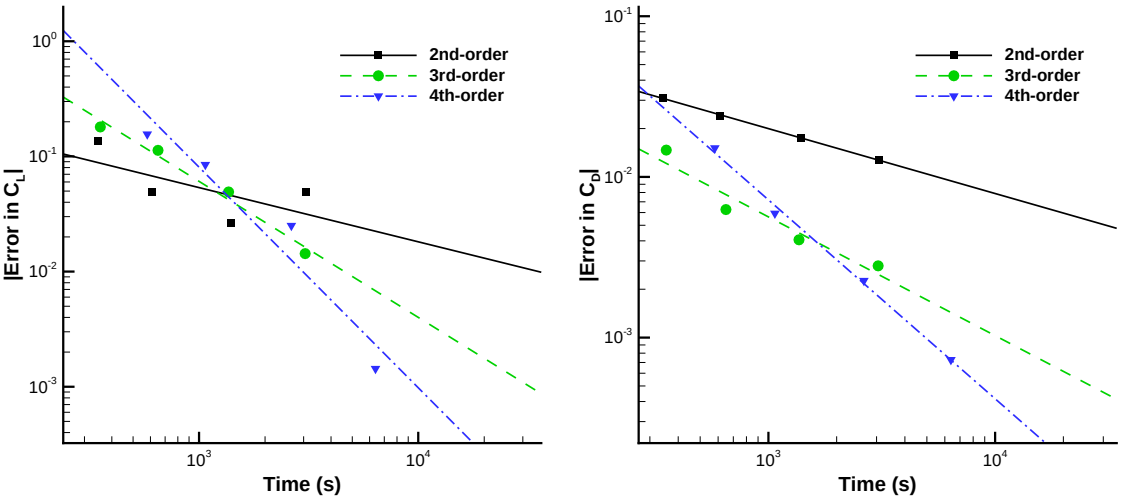
Grid	$C_L * S$	$C_D * S$	$C_L * S$ **Error (%)**	$C_D * S$ **Error (%)**
Second-order				
$\boldsymbol{G}_1$	0.81437	0.36514	20.21	-7.86
$\boldsymbol{G}_2$	0.72650	0.37243	7.24	-6.02
$\boldsymbol{G}_3$	0.65116	0.37875	-3.88	-4.43
$\boldsymbol{G}_4$	0.62836	0.38355	-7.25	-3.22
Third-order				
$\boldsymbol{G}_1$	0.85774	0.38161	26.61	-3.70
$\boldsymbol{G}_2$	0.79038	0.39003	16.67	-1.58
$\boldsymbol{G}_3$	0.72662	0.39223	7.25	-1.02
$\boldsymbol{G}_4$	0.69179	0.39349	2.11	-0.71
Fourth-order				
$\boldsymbol{G}_1$	0.83331	0.41142	23.00	3.82
$\boldsymbol{G}_2$	0.76220	0.40221	12.51	1.50
$\boldsymbol{G}_3$	0.70251	0.39855	3.70	0.57
$\boldsymbol{G}_4$	0.67891	0.39702	0.21	0.18
Exact (fourth-order on $\boldsymbol{G}_F$)				
$(\boldsymbol{G}_F)_{4th-order}$	0.67747	0.39629	–	–



(a) Mesh refinement behaviour of lift.

(b) Mesh refinement behaviour of drag.

Figure 5.8: Viscous laminar flow analysis: flow analysis mesh refinement study demonstrating the asymptotic error behaviour of each method. Dotted line shows the power function of best fit on the data; slopes of these lines are shown in Table 5.11.



(a) Efficiency of high-order methods at predicting lift.

(b) Efficiency of high-order methods at predicting drag.

Figure 5.9: Viscous laminar flow analysis: efficiency of high-order methods at accurately predicting lift and drag.

Table 5.11: Viscous laminar flow analysis: observed order of accuracy of lift and drag for each method.

	Observed Order	
Method	C_L	C_D
Second-order	1.57 ± 0.12	1.33 ± 0.08
Third-order	3.81 ± 0.68	2.39 ± 0.27
Fourth-order	6.94 ± 2.27	4.50 ± 0.95

Table 5.12: Viscous laminar span optimization: summary of optimization data.

Grid	Wall Time (120 proc.) (hh:mm:ss)	Major Iterations	Function Evaluations	Final Feasibility	Final Optimality
Second-order					
$\boldsymbol{G}_1$	02:14:39	8	9	1.2E-11	8.9E-9
$\boldsymbol{G}_2$	04:16:05	8	9	1.2E-13	3.0E-10
$\boldsymbol{G}_3$	08:48:30	7	8	8.0E-13	4.0E-10
$\boldsymbol{G}_4$	24:22:28	8	9	1.2E-13	2.0E-10
Third-order					
$\boldsymbol{G}_1$	02:18:19	8	9	6.2E-10	3.2E-8
$\boldsymbol{G}_2$	04:18:05	8	9	7.9E-11	1.8E-8
$\boldsymbol{G}_3$	10:59:47	9	10	2.8E-14	1.9E-10
$\boldsymbol{G}_4$	66:09:05	9	21	1.1E-8	4.8E-8
Fourth-order					
$\boldsymbol{G}_1$	05:51:55	8	9	8.2E-13	8.8E-9
$\boldsymbol{G}_2$	11:37:09	8	9	1.4E-11	1.4E-10
$\boldsymbol{G}_3$	60:47:14	10	13	3.6E-10	9.6E-7
$\boldsymbol{G}_4$	91:00:40	9	10	4.6E-12	5.4E-9

Table 5.13: Viscous laminar span optimization: design variables of various optimal shapes.

Grid	Optimal AoA (°)	Optimal Semi-span
Second-order		
$\boldsymbol{G}_1$	7.84	1.72
$\boldsymbol{G}_2$	6.53	2.21
$\boldsymbol{G}_3$	6.00	2.62
$\boldsymbol{G}_4$	6.23	2.66
Third-order		
$\boldsymbol{G}_1$	9.56	1.41
$\boldsymbol{G}_2$	8.59	1.67
$\boldsymbol{G}_3$	8.13	1.91
$\boldsymbol{G}_4$	8.12	2.03
Fourth-order		
$\boldsymbol{G}_1$	9.50	1.45
$\boldsymbol{G}_2$	8.36	1.77
$\boldsymbol{G}_3$	8.05	2.01
$\boldsymbol{G}_4$	8.07	2.01

Table 5.14: Viscous laminar span optimization: summary of aerodynamic performance of optimized geometries.

Grid	$(C_L * S)$	$(C_D * S)$	$(C_L * S)_{\boldsymbol{G}_F}$	$(C_D * S)_{\boldsymbol{G}_F}$
Second-order				
$\boldsymbol{G}_1$	0.500000	0.183726	0.389606	0.191412
$\boldsymbol{G}_2$	0.500000	0.225008	0.481497	0.235860
$\boldsymbol{G}_3$	0.500000	0.264373	0.559856	0.275104
$\boldsymbol{G}_4$	0.500000	0.273621	0.585124	0.282268
Third-order				
$\boldsymbol{G}_1$	0.500000	0.172220	0.316762	0.166869
$\boldsymbol{G}_2$	0.500000	0.195735	0.382758	0.191152
$\boldsymbol{G}_3$	0.500000	0.219777	0.452551	0.216799
$\boldsymbol{G}_4$	0.500000	0.232818	0.487392	0.230897
Fourth-order				
$\boldsymbol{G}_1$	0.500000	0.187240	0.330068	0.172434
$\boldsymbol{G}_2$	0.500000	0.211091	0.415019	0.201990
$\boldsymbol{G}_3$	0.500000	0.233060	0.478700	0.227419
$\boldsymbol{G}_4$	0.500000	0.240497	0.503569	0.237465

Table 5.15: Viscous laminar span optimization: angle of attack adjusted to meet lift constraint with the fourth-order fine mesh analysis.

Grid	**Adjusted AoA (°)**	$(C_L * S)_{\boldsymbol{G}_F}$	$(C_D * S)_{\boldsymbol{G}_F}$
Second-order			
$\boldsymbol{G}_2$	7.00	0.501525	0.240246
$\boldsymbol{G}_3$	5.14	0.500887	0.266970
$\boldsymbol{G}_4$	4.99	0.499936	0.270329
Third-order			
$\boldsymbol{G}_4$	9.29	0.500986	0.242773
Fourth-order			
$\boldsymbol{G}_3$	9.96	0.504592	0.247326
$\boldsymbol{G}_4$	7.95	0.500529	0.236341

Chapter 6

Conclusions, Contributions and Recommendations

An adjoint-based optimization algorithm was extended to include high-order analysis to allow for optimization to be performed on coarser meshes while maintaining a high level of accuracy.

The optimization algorithm has been verified with the use of the complex-step approximation to check the consistency of analytical differentiation in several portions of the gradient computation. The directional derivative test was performed on several objective functions to ensure that the gradient is analytically precise. Validation was successfully performed with an inverse design problem, and an inviscid twist optimization. A mesh refinement study was performed to demonstrate the accuracy and efficiency of the high-order viscous laminar flow analysis. It was found that high-order methods are more efficient than second-order methods at determining lift and drag accurately; high-order methods can achieve the same accuracy as second-order methods for a reduced computational cost. High-order analysis used in conjunction with an adjoint-based optimization algorithm was shown to produce aerodynamic shapes with improved performance with similar CPU time as a second-order method when applied to viscous flows. Experimentation with an inviscid subsonic twist optimization showed no particular benefit to improving accuracy of the analysis through either grid refinement or increase of order of accuracy, potentially due to the simplicity of the problem definition. High-order methods were also shown to meet the nonlinear constraints more closely when the optimal shape was evaluated with a more accurate analysis.

Several novel contributions were made in this thesis project:

- improvements to an existing high-order flow solver including general speed improvements, implementation of compact second-derivative operators, and incorporation of the diagonal norm as an integration rule for viscous flows;
- a first attempt at high-order grid generation criteria by examining geometric and hyperbolic stretching functions in one dimension;
- extension of adjoint equations to high-order with the addition of larger bandwidth SBP operators;
- exploration of a technique of analytically forming the transposed Jacobian-vector products in a way which does not require the Jacobian matrix to be explicitly stored. This technique has an increased computational cost (around 2 to 4 times as long to solve the adjoint problem, depending

on the mesh) but is a necessary expense since it is expected that the current system would not be able to store the high-order flux Jacobian matrix with the available memory with a mesh of a reasonable size.

The introduction of high-order spatial discretization applied to ASO leads to many new and interesting extensions. The following is a list of recommendations to improve the usefulness and extend the capabilities of the current framework:

- robustness of the high-order analysis should be improved, and could be done by using various safeguarding techniques [16] or alternative globalization strategies [31];
- improving preconditioning is extremely important for high-order, as larger bandwidth operators tend to increase the stiffness of the linear system of equations. The current preconditioner is modelled after a first-order approximation to the flux Jacobian and may not be adequate for discretizations higher than fourth-order, although convergence was achieved with this preconditioner in the solution of the flow residual equations and the flow adjoint equations for discretizations of up to fourth-order;
- a hybrid technique of combining explicit and implicit matrix-vector products could be explored which stores the inviscid and viscous double-derivative contributions and recomputes the cross-derivative terms at each matrix-vector product request, similar to the hybrid technique employed by Barth [11]. Since the cross-derivative terms contribute the most to memory requirements, this could lead to an efficient high-order adjoint solution methodology with modest memory requirements;
- extending the high-order analysis and gradient computation to include the effects of turbulence by use of a RANS turbulence model such as the Spalart-Allmaras turbulence model;
- diagonal-norm SBP operators have been explored in this work. The current framework would allow for extension to restricted full-norm SBP operators without much modification and could be stabilized using ideas from Mattsson and Almquist [56];
- SBP operator optimization is an avenue that can be explored. Free parameters can be introduced into an SBP operator by extending the boundary stencil [57]. These additional free parameters can be used to tailor the SBP operator to specific needs such as reducing principal truncation error, improve spectral radius, or reducing floating point operations by reducing bandwidth;
- order-sequencing could be employed to improve the efficiency of optimization. This would consist of optimizing first with a second-order algorithm, and switching to a high-order algorithm after a specified tolerance;
- finally, issues associated with the coordinate transformation with high-order spatial discretization should be investigated further – one possible approach is to use the free parameters available in the SBP operator to mitigate the negative Jacobian problem.

Appendices

Appendix A

Summation-by-Parts Operators

A.1 First-Derivative SBP Operators

Listed here are the first-derivative operators, which are applied directly to form the inviscid and viscous contributions to the discrete residual vector. In addition, these operators can be used to form a second-derivative approximation by applying the first-derivative operator twice. There are several advantages to using more advanced second-derivative operators when computing the derivatives of the viscous flux vectors [22], such as the reduced storage requirements for storage of the Jacobian matrix, reduced time to form an ILU factorization, and reduced truncation error. Such second-derivative operators are presented later in this appendix.

The operators presented here have a well defined structure. An SBP operator which is an exact derivative when applied to a discretized monomial of degree s has an interior order of accuracy of $2s$, a boundary order of accuracy of s, and a global order of accuracy of $s+1$. For a standard diagonal-norm SBP operator, the boundary scheme is formed by a $2s \times 2s$ block, while the remainder is defined by the symmetry of the operator. SBP operators with preferential properties can be constructed by expanding the boundary stencil to include more nodes. This can be used to the advantage of the designer by using the additional free parameters to optimize the operator, as is done by Mattsson *et al.* [57].

The SBP operator for the first-derivative has a general form of $\delta = H^{-1}\Theta$ where

$$H = \mathrm{diag}(h_1, \ldots, h_{2s}, 1, \ldots)$$

and

$$\Theta + \Theta^T = \mathrm{diag}(-1, 0, 0, \ldots, 0, 0, 1).$$

The interior scheme is predetermined and is a $2s$-order central-difference approximation to the first derivative. The interior stencil coefficients can be determined using the following:

$$(\delta \mathbf{u})_i = \sum_{v=-2s}^{2s} \alpha_v \mathbf{u}_{i+v} \tag{A.1}$$

$$\alpha_v = \frac{(-1)^{v+1}(s!)^2}{v(s+v)!(s-v)!} \tag{A.2}$$

where α_v is the coefficient of the vth right-hand term. The corresponding left-hand term is

$\alpha_{-v} = -\alpha_v$ and $\alpha_0 = 0$ [48]. In total there are $2s$ unknowns in H and $s(2s-1)$ unknowns in Θ, all from boundary contributions. These unknowns are used to satisfy the accuracy requirements of the SBP operator at the boundary. A system of equations can be formed by row-vector products with $\mathbf{x}^m$, which represents a restriction of a monomial of degree m on the grid, i.e., the ith component is x_i^m. The accuracy conditions require that

$$\Theta \mathbf{x}^m = mH\mathbf{x}^{m-1}; \quad m \in [0, s]. \tag{A.3}$$

This forms a set of $s+1$ equations for each boundary node, for a total of $2s(s+1)$ equations and $s(2s+1)$ unknowns. This leads to a balanced set of equations and unknowns for the second- and third-order operator, with one undetermined parameter in the fourth-order operator. One additional equation is required to form a complete set for the fourth-order operator; details of this additional equation are below.

A.1.1 Second-Order SBP Operator for the First Derivative

$$H_{\xi_i} = \Delta\xi_i \ \text{diag}\left(\frac{1}{2}, 1, \cdots\right)$$

$$\delta_{\xi_i} = \frac{1}{\Delta\xi_i}\begin{pmatrix} -1 & 1 & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{pmatrix}$$

A.1.2 Third-Order SBP Operator for the First Derivative

$$H_{\xi_i} = \Delta\xi_i \ \text{diag}\left(\frac{17}{48}, \frac{59}{48}, \frac{43}{48}, \frac{49}{48}, 1, \cdots\right)$$

$$\delta_{\xi_i} = \frac{1}{\Delta\xi_i}\begin{pmatrix} -\frac{24}{17} & \frac{59}{34} & -\frac{4}{17} & -\frac{3}{34} & & & & & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 & & & & & & & \\ \frac{4}{43} & -\frac{59}{86} & 0 & \frac{59}{86} & -\frac{4}{43} & & & & & & \\ \frac{3}{98} & 0 & -\frac{59}{98} & 0 & \frac{32}{49} & -\frac{4}{49} & & & & & \\ & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & & \\ & & & & & \frac{4}{49} & -\frac{32}{49} & 0 & \frac{59}{98} & 0 & -\frac{3}{98} \\ & & & & & & \frac{4}{43} & -\frac{59}{86} & 0 & \frac{59}{86} & -\frac{4}{43} \\ & & & & & & & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & & & & & \frac{3}{34} & \frac{4}{17} & -\frac{59}{34} & \frac{24}{17} \end{pmatrix}$$

A.1.3 Fourth-Order SBP Operator for the First Derivative

$$H_{\xi_i} = \Delta\xi_i \ \text{diag}\left(\frac{13649}{43200}, \frac{12013}{8640}, \frac{2711}{4320}, \frac{5359}{4320}, \frac{7877}{8640}, \frac{43801}{43200}, 1, \cdots\right)$$

$$\delta_{\xi_i} = \frac{1}{\Delta\xi_i}\begin{pmatrix}
-\frac{21600}{13649} & \delta_{1,2} & \delta_{1,3} & \delta_{1,4} & \delta_{1,5} & \delta_{1,6} & & & & & & & & & & \\
\delta_{2,1} & 0 & \delta_{2,3} & \delta_{2,4} & \delta_{2,5} & \delta_{2,6} & & & & & & & & & & \\
\delta_{3,1} & \delta_{3,2} & 0 & \delta_{3,4} & \delta_{3,5} & \delta_{3,6} & & & & & & & & & & \\
\delta_{4,1} & \delta_{4,2} & \delta_{4,3} & 0 & \delta_{4,5} & \delta_{4,6} & \frac{72}{5359} & & & & & & & & & \\
\delta_{5,1} & \delta_{5,2} & \delta_{5,3} & \delta_{5,4} & 0 & \frac{8640}{7877}\Theta_{5,6} & -\frac{1296}{7877} & \frac{144}{7877} & & & & & & & & \\
\delta_{6,1} & \delta_{6,2} & \delta_{6,3} & \delta_{6,4} & -\frac{43200}{43801}\Theta_{5,6} & 0 & \frac{32400}{43801} & -\frac{6480}{43801} & \frac{720}{43801} & & & & & & & \\
 & & & -\frac{1}{60} & \frac{3}{20} & -\frac{3}{4} & 0 & \frac{3}{4} & -\frac{3}{20} & \frac{1}{60} & & & & & & \\
 & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \\
 & & & & & -\frac{1}{60} & \frac{3}{20} & -\frac{3}{4} & 0 & \frac{3}{4} & -\frac{3}{20} & \frac{1}{60} & & & & \\
 & & & & & & -\frac{720}{43801} & \frac{6480}{43801} & -\frac{32400}{43801} & 0 & \frac{43200}{43801}\Theta_{5,6} & -\delta_{6,4} & -\delta_{6,3} & -\delta_{6,2} & -\delta_{6,1} \\
 & & & & & & & -\frac{144}{7877} & \frac{1296}{7877} & -\frac{8640}{7877}\Theta_{5,6} & 0 & -\delta_{5,4} & -\delta_{5.3} & -\delta_{5,2} & -\delta_{5,1} \\
 & & & & & & & & -\frac{72}{5359} & -\delta_{4,6} & -\delta_{4,5} & 0 & -\delta_{4,3} & -\delta_{4,2} & -\delta_{4,1} \\
 & & & & & & & & & -\delta_{3,6} & -\delta_{3,5} & -\delta_{3,4} & 0 & -\delta_{3,2} & -\delta_{3,1} \\
 & & & & & & & & & -\delta_{2,6} & -\delta_{2,5} & -\delta_{2,4} & -\delta_{2,3} & 0 & -\delta_{2,1} \\
 & & & & & & & & & -\delta_{1,6} & -\delta_{1,5} & -\delta_{1,4} & -\delta_{1,3} & -\delta_{1,2} & \frac{21600}{13649}
\end{pmatrix}$$

$$\delta_{1,2} = -\frac{7624}{40947} + \frac{43200}{13649}\Theta_{5,6} \qquad \delta_{2,1} = \frac{7624}{180195} - \frac{8640}{12013}\Theta_{5,6} \qquad \delta_{3,1} = -\frac{715489}{162660} + \frac{17280}{2711}\Theta_{5,6}$$
$$\delta_{1,3} = \frac{715489}{81894} - \frac{172800}{13649}\Theta_{5,6} \qquad \delta_{2,3} = -\frac{57139}{12013} + \frac{86400}{12013}\Theta_{5,6} \qquad \delta_{3,2} = \frac{57139}{5422} - \frac{43200}{2711}\Theta_{5,6}$$
$$\delta_{1,4} = -\frac{187917}{13649} + \frac{259200}{13649}\Theta_{5,6} \qquad \delta_{2,4} = \frac{745733}{72078} - \frac{172800}{12013}\Theta_{5,6} \qquad \delta_{3,4} = -\frac{176839}{8133} + \frac{86400}{2711}\Theta_{5,6}$$
$$\delta_{1,5} = \frac{735635}{81894} - \frac{172800}{13649}\Theta_{5,6} \qquad \delta_{2,5} = -\frac{91715}{12013} - \frac{129600}{12013}\Theta_{5,6} \qquad \delta_{3,5} = \frac{242111}{10844} - \frac{86400}{2711}\Theta_{5,6}$$
$$\delta_{1,6} = -\frac{89387}{40947} + \frac{43200}{13649}\Theta_{5,6} \qquad \delta_{2,6} = \frac{240569}{120130} - \frac{34560}{12013}\Theta_{5,6} \qquad \delta_{3,6} = -\frac{182261}{27110} + \frac{25920}{2711}\Theta_{5,6}$$

$$\delta_{4,1} = \frac{187917}{53590} - \frac{25920}{5359}\Theta_{5,6} \qquad \delta_{5,1} = -\frac{147127}{47262} + \frac{34560}{7877}\Theta_{5,6} \qquad \delta_{6,1} = \frac{89387}{131403} - \frac{43200}{43801}\Theta_{5,6}$$
$$\delta_{4,2} = -\frac{745733}{64308} + \frac{86400}{5359}\Theta_{5,6} \qquad \delta_{5,2} = \frac{91715}{7877} - \frac{129600}{7877}\Theta_{5,6} \qquad \delta_{6,2} = -\frac{240569}{87602} + \frac{172800}{43801}\Theta_{5,6}$$
$$\delta_{4,3} = \frac{176839}{16077} - \frac{86400}{5359}\Theta_{5,6} \qquad \delta_{5,3} = -\frac{242111}{15754} + \frac{172800}{7877}\Theta_{5,6} \qquad \delta_{6,3} = \frac{182261}{43801} - \frac{259200}{43801}\Theta_{5,6}$$
$$\delta_{4,5} = -\frac{165041}{32154} + \frac{43200}{5359}\Theta_{5,6} \qquad \delta_{5,4} = \frac{165041}{23631} - \frac{86400}{7877}\Theta_{5,6} \qquad \delta_{6,4} = -\frac{710473}{262806} + \frac{172800}{43801}\Theta_{5,6}$$
$$\delta_{4,6} = \frac{710473}{321540} - \frac{17280}{5359}\Theta_{5,6}$$

The fourth-order operator introduces one free parameter, $\Theta_{5,6}$. There are several options when choosing how to optimize an SBP operator. Minimizing the spectral radius of the operator may be ideal for an explicit scheme as it would allow for larger CFL number to be used, improving the overall speed of the algorithm [26]. Minimizing the Average Boundary Truncation Error (ABTE) presents an attempt at minimizing the truncation error in the local solution associated with the scheme [26, 22]. Minimizing the bandwidth of the boundary scheme is another approach; this reduces the number of matrix entries in the SBP operator. The criterion used for operator optimization in this work is to minimize the integrated error on the solution [21]. In this approach the principal error, $E_H = \delta x^{s+1} - (s+1)x^s$, is computed and its norm, $E_H^T H E_H$, is minimized. It is expected that this will reduce the truncation error associated with functionals, such as the lift and drag coefficients, since it accounts for the diagonal-norm integration scheme.

Table A.1: Various choices of fourth-order first-derivative free parameter

Criterion	$\Theta_{5,6}$
minimum error norm	$\frac{540222713}{764186400}$
minimum ABTE	$\frac{17171}{24300}$
minimum bandwidth	$\frac{89387}{129600}$

A.2 Second-Derivative Operators with Variable Coefficients

The second derivative operators used in this work have been derived by Del Rey Fernández. Details of how to derive second derivative operators can be found in the SBP operator review paper by Del Rey Fernández *et al.* [21]. Recall that $\delta^{(2)}(B) = \sum_{j=1}^{N} b_j M_j$ where b_j is the jth component of the diagonal variable-coefficient matrix. The M_i matrices are very sparse, so only the non-zero blocks are shown below, with the subscripts indicating the location of the columns and rows within the full structure of each M_i.

A.2.1 Second-Order SBP Operator for the Second Derivative with Variable Coefficients

$$(M_1)_{1:2,1:3} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} 2 & -3 & 1 \\ \frac{1}{2} & -\frac{1}{2} & 0 \end{pmatrix} \quad (M_2)_{1:3,1:3} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -1 & 1 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

$$(M_{\text{interior}})_{j-1:j+1,j-1:j+1} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

$$ED_b = \frac{1}{\Delta\xi_i}\begin{pmatrix} \frac{3}{2} & -2 & \frac{1}{2} & & \\ 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 \\ & & \frac{1}{2} & -2 & \frac{3}{2} \end{pmatrix}$$

A.2.2 Third-Order SBP Operator for the Second Derivative with Variable Coefficients

$$(M_1)_{1:4,1:5} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} \frac{2879991513763}{708927703404} & -\frac{1690268551303}{177231925851} & \frac{1084368403411}{118154617234} & -\frac{812081292415}{177231925851} & \frac{36658672979}{41701629612} \\ \frac{12}{17} & -\frac{59}{68} & \frac{2}{17} & \frac{3}{68} & 0 \\ -\frac{96}{731} & \frac{118}{731} & -\frac{16}{731} & -\frac{6}{731} & 0 \\ -\frac{36}{833} & \frac{177}{3332} & -\frac{6}{833} & -\frac{9}{3332} & 0 \end{pmatrix}$$

$$(M_2)_{1:3,1:3} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{59}{68} & 0 & \frac{59}{68} \\ 0 & 0 & 0 \\ \frac{59}{172} & 0 & -\frac{59}{172} \end{pmatrix}$$

$$(M_3)_{1:5,1:5} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{16}{731} & \frac{118}{731} & 0 & -\frac{118}{731} & \frac{16}{731} \\ \frac{2}{43} & -\frac{59}{172} & 0 & \frac{59}{172} & -\frac{2}{43} \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{118}{2107} & \frac{3481}{8428} & 0 & -\frac{3481}{8428} & \frac{118}{2107} \\ \frac{1}{129} & -\frac{59}{1032} & 0 & \frac{59}{1032} & -\frac{1}{129} \end{pmatrix}$$

$$(M_4)_{1:6,1:3} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} \frac{22234270742044366600996285}{923846935118389768309296 38} & \frac{11659070464799122754735 4443}{184769387023677953661859276} & -\frac{16727632297415354559950 05}{429696248892274310841533 2} \\ \frac{19761136381015462296161 77}{108687874719810560977564 28} & -\frac{53803981875030302899525 17}{108687874719810560977564 28} & \frac{45884983363308829505569}{126381249674198326718098} \\ -\frac{16727632297415354559950 05}{108687874719810560977564 28} & \frac{27072140184352209408285 71}{543439373599052804887821 4} & -\frac{87475370093429330119575 37}{108687874719810560977564 28} \\ -\frac{73087922992222291610446 675}{266285293063535874395032 486} & \frac{20146347569936722217937 635}{266285293063535874395032 486} & -\frac{17526637298626240216796 6}{309634061701785900459340 1} \\ \frac{10179232864704458960189 45}{306883410973582760407240 32} & -\frac{18083560397112721747745 11}{102294470324527586802413 44} & \frac{67591689042814771864657}{118947058516892542793504} \\ -\frac{39404316779336070437 83}{713682351101355256761024} & \frac{27313058433187407833395}{713682351101355256761024} & -\frac{11641319492235593482625}{118947058516892542793504} \end{pmatrix}$$

$$(M_4)_{1:6,4:6} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{73087922992222291610446675}{923846935118389768309296 38} & \frac{10179232864704458960189 45}{108687874719810560977564 28} & -\frac{39404316779336070437 83}{252762499348396653436196} \\ \frac{34146351813452071555826 5}{543439373599052804887821 4} & -\frac{91950307103962991937687}{639340439528297417515084} & \frac{46293319378283742090 5}{148683823146115678491 88} \\ -\frac{17526637298626240216796 6}{271719686799526402443910 7} & \frac{20277506712844431559397 1}{319670219764148708757542} & -\frac{34923958476706780447875}{319670219764148708757542} \\ -\frac{12857190984708587219119 16}{133142646531767937197516243} & \frac{37177481415904140161563 1}{156638407684432867291195 58} & -\frac{20702884236085504852 09}{364275366707983412305106} \\ \frac{37177481415904140161563 1}{153441705486791380203620 16} & -\frac{16733516466668624505202 4947}{306883410973582760407240 32} & \frac{68801299953147017116669}{713682351101355256761024} \\ -\frac{20702884236085504852 09}{356841175550677628380512} & \frac{68801299953147017116669}{713682351101355256761024} & -\frac{18185432907770156040113}{713682351101355256761024} \end{pmatrix}$$

$$(M_5)_{3:7,3:7} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{2}{43} & \frac{8}{43} & -\frac{6}{43} & 0 & 0 \\ \frac{8}{49} & -\frac{40}{49} & \frac{24}{49} & \frac{8}{49} & 0 \\ -\frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & \frac{1}{2} & -\frac{1}{8} \\ 0 & \frac{1}{6} & \frac{1}{2} & -\frac{5}{6} & \frac{1}{6} \\ 0 & 0 & -\frac{1}{8} & \frac{1}{6} & -\frac{1}{24} \end{pmatrix}$$

$$(M_6)_{4:8,4:8} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{2}{49} & \frac{8}{49} & -\frac{6}{49} & 0 & 0 \\ \frac{1}{6} & -\frac{5}{6} & \frac{1}{2} & \frac{1}{6} & 0 \\ -\frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & \frac{1}{2} & -\frac{1}{8} \\ 0 & \frac{1}{6} & \frac{1}{2} & -\frac{5}{6} & \frac{1}{6} \\ 0 & 0 & \frac{1}{8} & \frac{1}{6} & -\frac{1}{24} \end{pmatrix}$$

$$(M_{\text{interior}})_{j-2:j+2,j-2:j+2} = \frac{1}{(\Delta\xi_i)^2}\begin{pmatrix} -\frac{1}{24} & \frac{1}{6} & -\frac{1}{8} & 0 & 0 \\ \frac{1}{6} & -\frac{5}{6} & \frac{1}{2} & \frac{1}{6} & 0 \\ -\frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & \frac{1}{2} & -\frac{1}{8} \\ 0 & \frac{1}{6} & \frac{1}{2} & -\frac{5}{6} & \frac{1}{6} \\ 0 & 0 & -\frac{1}{8} & \frac{1}{6} & -\frac{1}{24} \end{pmatrix}$$

$$ED_b = \frac{1}{\Delta\xi_i}\begin{pmatrix} \frac{252525932147}{117745777728} & -\frac{124968006275}{29436444432} & \frac{66095117411}{19624296288} & -\frac{46470821123}{29436444432} & \frac{36658672979}{117745777728} & & \\ 0 & 0 & 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 & 0 & 0 \\ & & \frac{36658672979}{117745777728} & -\frac{46470821123}{29436444432} & \frac{66095117411}{19624296288} & -\frac{124968006275}{29436444432} & \frac{252525932147}{117745777728} \end{pmatrix}$$

A.2.3 Fourth-Order SBP Operator for the Second Derivative with Variable Coefficients

The fourth-order second-derivative operator is derived based on the work by Del Rey Fernández *et al.* [21], and is omitted here for brevity.

Appendix B

Analytical Differentiation of the Discrete Residual Vector

The adjoint method relies heavily on the accurate computation of the derivatives of all terms associated with the residual vector shown in Equation 2.36 and the objective function, $\mathcal{J}$. The flow adjoint equations requires the differentiation of the residual vector, $\boldsymbol{\mathcal{R}}$, with respect to the untransformed flow variables, $\mathbf{Q}$. The mesh adjoint equations require the differentiation of the residual vector with respect to the grid metrics $\mathbf{m} = [\xi_x, \xi_y, \xi_z, \eta_x, \eta_y, \eta_z, \zeta_x, \zeta_y, \zeta_z]^T$. This chapter provides basic instructions to differentiating residual contributions with respect to either the flow variables or the grid metrics.

B.1 Differentiation with Respect to Flow Variables

The algorithm used in this work requires a transposed Jacobian-vector product, $A^T\mathbf{v}$. It is not necessary to form the flux Jacobian matrix explicitly and store it; an alternative method is chosen to save on memory requirements [39, 3, 11, 64]. The products are analytically constructed, and the vector $\Phi = A^T\mathbf{v}$ is formed at every matrix-vector product request.

B.1.1 Inviscid Flux

The inviscid flux contribution to the residual vector is

$$\boldsymbol{\mathcal{R}}_{\text{inv}} = \delta_\xi \hat{\mathbf{E}} + \delta_\eta \hat{\mathbf{F}} + \delta_\zeta \hat{\mathbf{G}}, \tag{B.1}$$

where $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$, and δ take on their usual definitions as described in Chapter 2. Differentiating and simplifying leads to

$$\Phi_{\text{inv}} = \frac{\partial \boldsymbol{\mathcal{R}}_{\text{inv}}}{\partial \mathbf{Q}}^T \mathbf{v} \tag{B.2}$$

$$= \left(\frac{\partial \hat{\mathbf{E}}}{\partial \mathbf{Q}}\right)^T \delta_\xi^T \mathbf{v} + \left(\frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{Q}}\right)^T \delta_\eta^T \mathbf{v} + \left(\frac{\partial \hat{\mathbf{G}}}{\partial \mathbf{Q}}\right)^T \delta_\zeta^T \mathbf{v}, \tag{B.3}$$

where $\frac{\partial \hat{\mathbf{E}}}{\partial \mathbf{Q}}$ is a block diagonal matrix. Formatting in this way allows the program associated with computing Φ to take on a very similar structure to the program responsible for the computation of $\boldsymbol{\mathcal{R}}$.

B.1.2 Dissipation Operator

The artificial dissipation residual contribution is

$$\boldsymbol{\mathcal{R}}_{\text{AD}} = H_\xi^{-1} D_{d,\xi}^T B D_{d,\xi} J\hat{\mathbf{Q}} + H_\eta^{-1} D_{d,\eta}^T B D_{d,\eta} J\hat{\mathbf{Q}} + H_\zeta^{-1} D_{d,\zeta}^T B D_{d,\zeta} J\hat{\mathbf{Q}} \tag{B.4}$$

where $B = \epsilon A$, which represents a 5×5 matrix that applies non-uniform dissipation across the governing equations if matrix dissipation is applied. Differentiating and simplifying leads to

$$\Phi_{\text{AD}} = \frac{\partial \boldsymbol{\mathcal{R}}_{\text{AD}}}{\partial \mathbf{Q}}^T \mathbf{v} \tag{B.5}$$

$$= \sum_i^{\xi,\eta,\zeta} \left[D_{d,i}^T B D_{d,i} H_i^{-1} \mathbf{v} + \left(\frac{\partial B}{\partial \mathbf{Q}} D_{d,i} J\hat{\mathbf{Q}} \right)^T D_{d,i} H_i^{-1} \mathbf{v} \right] \tag{B.6}$$

where the properties $H^T = H$, $B^T = B$ and $\frac{\partial J\hat{\mathbf{Q}}}{\partial Q} = I$ are used. B can have a complicated dependence on Q, especially if a nonlinear shock capturing scheme is used:

$$\frac{\partial B}{\partial \mathbf{Q}} = \epsilon \frac{\partial A}{\partial \mathbf{Q}}$$

In the differentiation of B it is assumed that $\frac{\partial \epsilon}{\partial \mathbf{Q}} = 0$. The term $\frac{\partial A}{\partial \mathbf{Q}}$ is computed with the complex-step method [70]. If the nonlinear shock capturing scheme of Jameson *et al.* is to be included for application to transonic aerodynamic optimization, then ϵ must be analytically differentiated, as there is a direct dependence on the solution data in this case. Osusky has presented the necessary considerations for analytic differentiation of ϵ and has applied the algorithm to several transonic aerodynamic optimization problems [70, 69].

B.1.3 Viscous Flux

The viscous flux contributions to the residual vector are represented in a concise notation,

$$\boldsymbol{\mathcal{R}}_{\text{visc}} = -\sum_i^{\xi,\eta,\zeta} \sum_j^{\xi,\eta,\zeta} \delta_i A_{i,j} \delta_j \mathbf{Q}_{\mathbf{v}},$$

where $\mathbf{Q}_{\mathbf{v}} = [u, v, w, a^2]^T$. The inclusion of the double-derivative terms and the cross-derivative terms yield a total of 9 derivative combinations. In the above, the i index represents the outer derivative, while the j index represents the inner derivative. The double derivatives (where indices i and j take on the same direction) can be computed using the application of the first derivative twice, otherwise known as the non-compact second-derivative operator, or a specialized compact operator can be applied. The compact operator is preferred for its improved ILU factorization time and truncation error properties.

The viscous contribution to the adjoint update is

$$\Phi_{\text{visc}} = \frac{\partial \boldsymbol{\mathcal{R}}_{\text{visc}}}{\partial \mathbf{Q}}^T \mathbf{v} \tag{B.7}$$

$$= -\sum_i^{\xi,\eta,\zeta} \sum_j^{\xi,\eta,\zeta} \left[\left(\frac{\partial \mathbf{Q}_\mathbf{v}}{\partial \mathbf{Q}} \right)^T \delta_j^T A_{i,j} \delta_i^T \mathbf{v} + \mathbf{Q}_\mathbf{v}{}^T \delta_j^T \left(\frac{\partial A_{i,j}}{\partial \mathbf{Q}} \right)^T \delta_i^T \mathbf{v} \right], \tag{B.8}$$

where $Q = [\rho, \rho u, \rho v, \rho w, e]^T = [q_1, q_2, q_3, q_4, q_5]^T$, and the differentiation of the viscous variables Q_v forms a 4×5 matrix:

$$\frac{\partial \mathbf{Q}_\mathbf{v}}{\partial \mathbf{Q}} = \begin{pmatrix} \frac{-q_2}{q_1^2} & \frac{1}{q_1} & 0 & 0 & 0 \\ \frac{-q_3}{q_1^2} & 0 & \frac{1}{q_1} & 0 & 0 \\ \frac{-q_4}{q_1^2} & 0 & 0 & \frac{1}{q_1} & 0 \\ \frac{\gamma(\gamma-1)}{q_1^2}\left(\frac{q_2^2+q_3^2+q_4^2}{q_1} - q_5\right) & -\frac{\gamma(\gamma-1)q_2}{q_1} & -\frac{\gamma(\gamma-1)q_3}{q_1} & -\frac{\gamma(\gamma-1)q_4}{q_1} & -\frac{\gamma(\gamma-1)}{q_1} \end{pmatrix}.$$

The variable coefficient matrices, $A_{i,j}$ have a dependence on Q through both viscosity and the velocity components, (u, v, w). The differentiation of $A_{i,j}$ is a straightforward algebraic exercise.

B.1.4 Simultaneous Approximation Terms

Differentiating the SATs is complicated by communication between neighbouring blocks. Residual contributions to neighbouring blocks are appended to form one vector:

$$\begin{aligned} \boldsymbol{\mathcal{R}} &= [\boldsymbol{\mathcal{R}}_1, \boldsymbol{\mathcal{R}}_2]^T, \\ \mathbf{Q} &= [\mathbf{Q}_1, \mathbf{Q}_2]^T, \\ \mathbf{v} &= [\mathbf{v}_1, \mathbf{v}_2]^T, \\ \boldsymbol{\Phi} &= [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2]^T, \end{aligned}$$

where the subscripts 1 and 2 indicate the nodal contributions from block 1 and its neighbour, block 2. The local contribution to the matrix-vector product from the SATs takes the form $\Phi = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{Q}} \mathbf{v}$, which can be expressed as:

$$\begin{aligned} \boldsymbol{\Phi}_1 &= \left(\frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \mathbf{Q}_1} \right)^T \mathbf{v}_1 + \left(\frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \mathbf{Q}_1} \right)^T \mathbf{v}_2, \\ \boldsymbol{\Phi}_2 &= \left(\frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \mathbf{Q}_2} \right)^T \mathbf{v}_1 + \left(\frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \mathbf{Q}_2} \right)^T \mathbf{v}_2. \end{aligned}$$

This result can be used to improve the understanding of the implementation. For boundary SATs, such as the symmetry, wall, and far-field SATs, there is no communication required between blocks because the residual contribution only relies on local information. In this case, only $\frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \mathbf{Q}_1}$ and $\frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \mathbf{Q}_2}$ are non-zero. Interface SATs take on the form $\boldsymbol{\mathcal{R}}_{\text{SAT},1} = \alpha_{\text{SAT}}(\mathbf{X}_1 - \mathbf{X}_2)$, as described in Table 2.2. Differentiating

and simplifying with $\frac{\partial \mathbf{X}_1}{\partial \mathbf{Q}_2} = \frac{\partial \mathbf{X}_2}{\partial \mathbf{Q}_1} = 0$ leads to

$$\mathbf{\Phi}_1 = \alpha_{\text{SAT},1} \left(\frac{\partial \mathbf{X}_1}{\partial \mathbf{Q}_1} \right)^T \mathbf{v}_1 - \alpha_{\text{SAT},2} \left(\frac{\partial \mathbf{X}_1}{\partial \mathbf{Q}_1} \right)^T \mathbf{v}_2, \tag{B.9}$$

$$\mathbf{\Phi}_2 = -\alpha_{\text{SAT},1} \left(\frac{\partial \mathbf{X}_2}{\partial \mathbf{Q}_2} \right)^T \mathbf{v}_1 + \alpha_{\text{SAT},2} \left(\frac{\partial \mathbf{X}_2}{\partial \mathbf{Q}_2} \right)^T \mathbf{v}_2. \tag{B.10}$$

This form requires minimal communication across block interfaces; on block 1, only the boundary node of the neighbouring vector $\mathbf{v}_2$ is required to be transmitted.

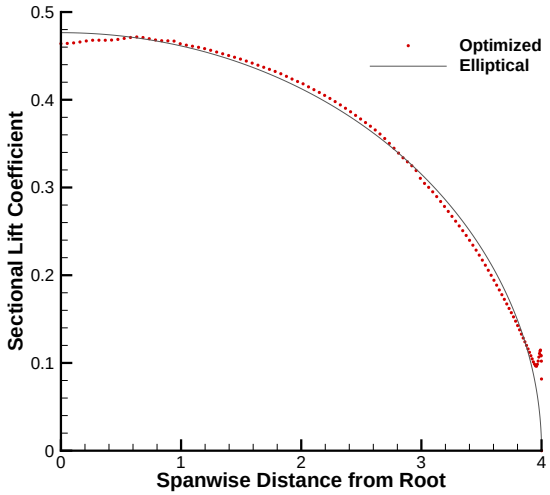
B.2 Differentiation with Respect to Grid Metrics

Differentiation with respect to the grid metrics is very similar to differentiation with respect to the flow variables, but some terms can be simplified. Limited information will be provided since the matrix-vector product approach taken in this work is identical for the flow adjoint equations and mesh adjoint equations. In the mesh adjoint equations, derivatives are taken with respect to the grid metrics, $\mathbf{m}$.

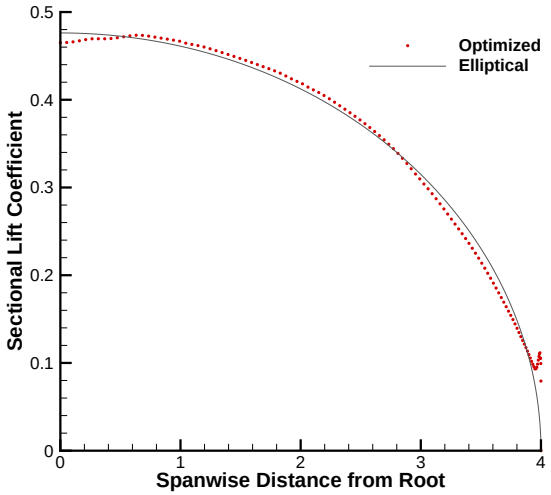
The inviscid contributions take on an identical form to Equation B.3. The contribution from the dissipation model takes a similar form to Equation B.6, with $\frac{\partial B}{\partial \mathbf{m}} = 0$ used to simplify the result. The viscous contributions are similar to Equation B.8, with $\frac{\partial \mathbf{Q}_v}{\partial \mathbf{m}} = 0$ used to simplify. SAT contributions are similar to Equations B.9 and B.10, but must account for an additional term which arises from the differentiation of the Jacobian of the transformation in Equation 2.9, $\frac{\partial J}{\partial \mathbf{m}}$.

Appendix C

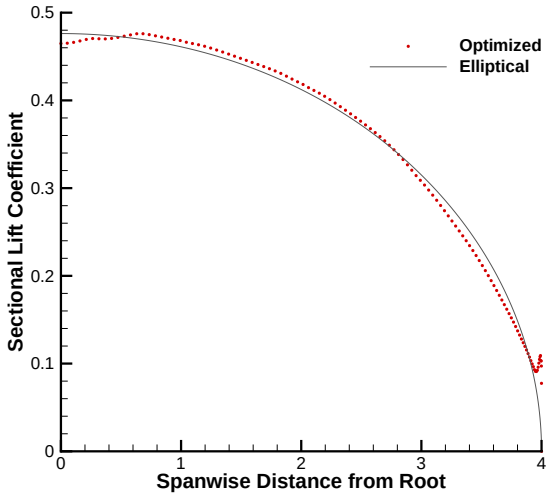
Ancillary Optimization Data



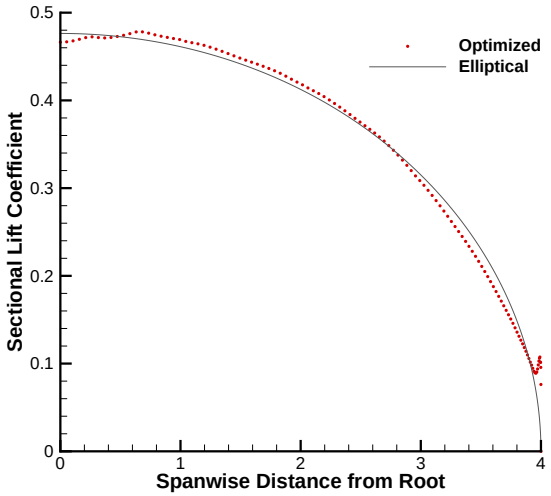
(a) Spanwise lift distribution of geometry optimized on grid G_1.



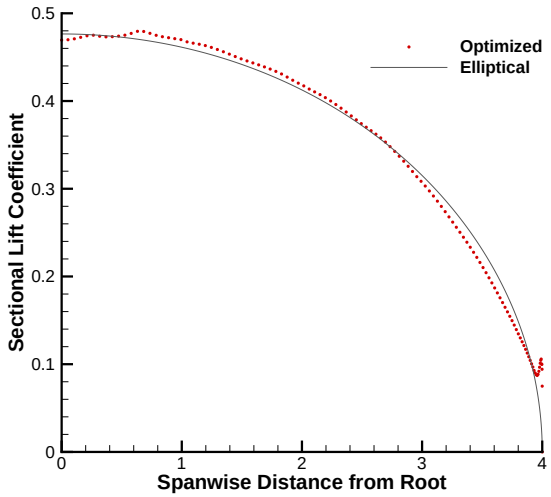
(b) Spanwise lift distribution of geometry optimized on grid G_2.



(c) Spanwise lift distribution of geometry optimized on grid G_3.

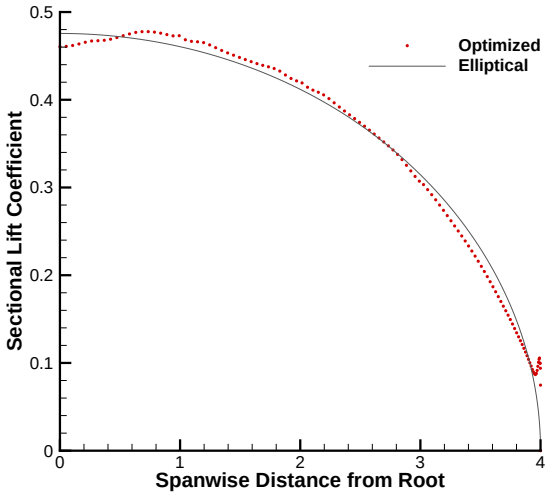


(d) Spanwise lift distribution of geometry optimized on grid G_4.

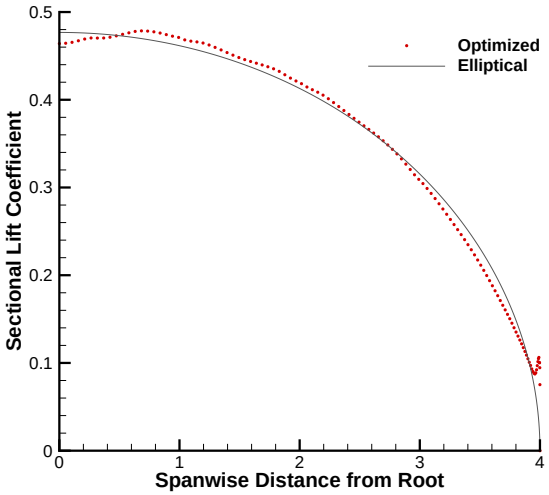


(e) Spanwise lift distribution of geometry optimized on grid G_5.

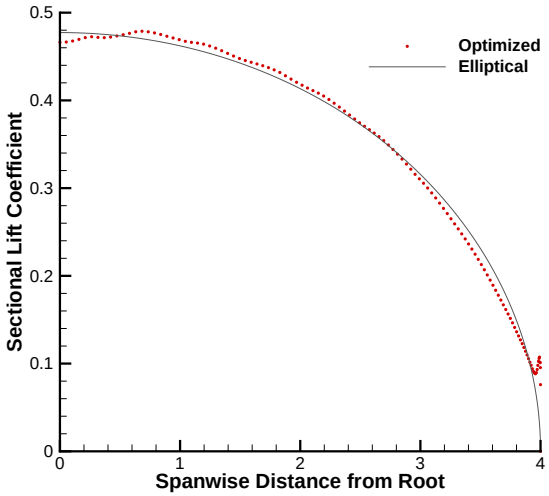
Figure C.1: Spanwise lift distribution of optimized geometries using the second-order method for the inviscid twist optimization case. Fine mesh analysis is performed on grid G_F with the fourth-order method on the optimal design to obtain the spanwise lift distribution.



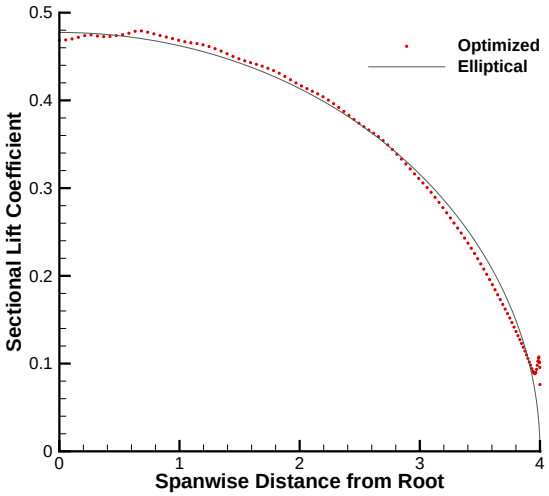
(a) Spanwise lift distribution of geometry optimized on grid G_1.



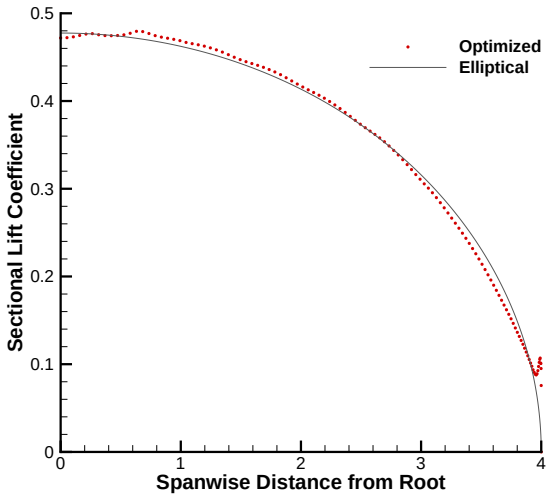
(b) Spanwise lift distribution of geometry optimized on grid G_2.



(c) Spanwise lift distribution of geometry optimized on grid G_3.

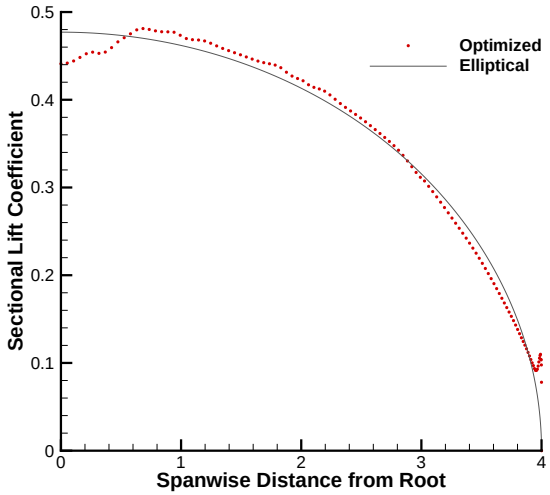


(d) Spanwise lift distribution of geometry optimized on grid G_4.

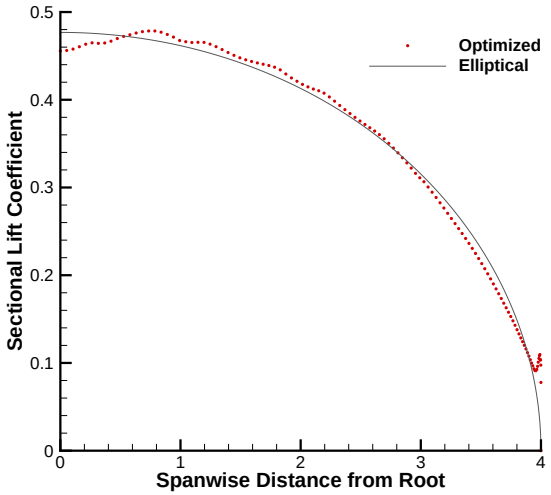


(e) Spanwise lift distribution of geometry optimized on grid G_5.

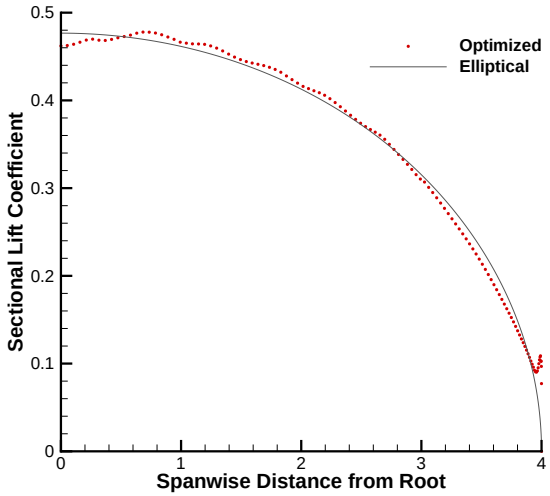
Figure C.2: Spanwise lift distribution of optimized geometries using the third-order method for the inviscid twist optimization case. Fine mesh analysis is performed on grid G_F with the fourth-order method on the optimal design to obtain the spanwise lift distribution.



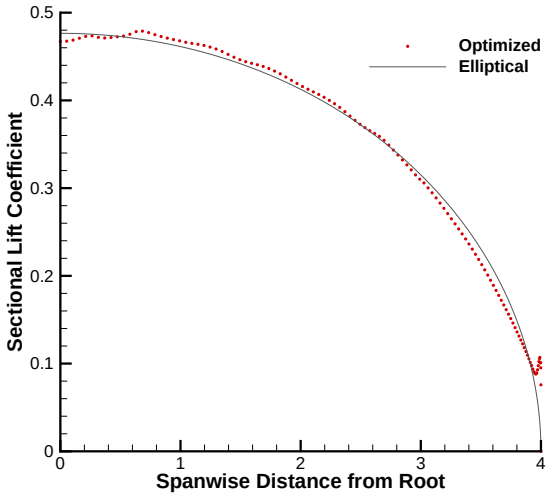
(a) Spanwise lift distribution of geometry optimized on grid G_1.



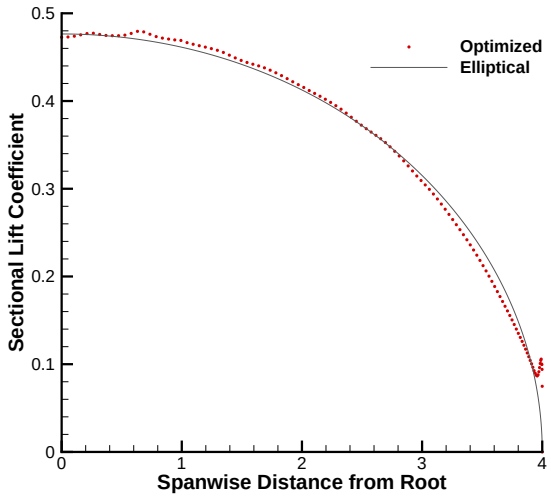
(b) Spanwise lift distribution of geometry optimized on grid G_2.



(c) Spanwise lift distribution of geometry optimized on grid G_3.



(d) Spanwise lift distribution of geometry optimized on grid G_4.



(e) Spanwise lift distribution of geometry optimized on grid G_5.

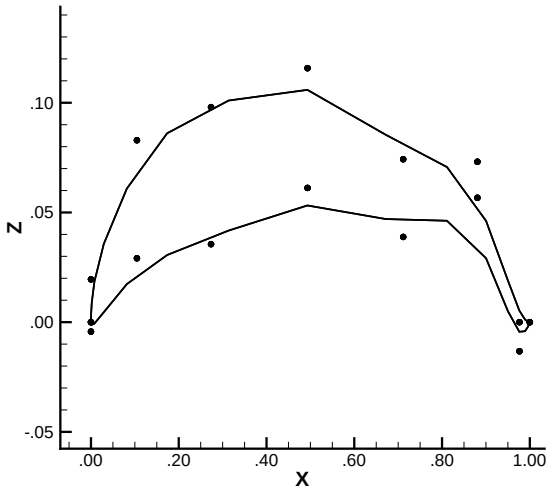
Figure C.3: Spanwise lift distribution of optimized geometries using the fourth-order method for the inviscid twist optimization case. Fine mesh analysis is performed on grid G_F with the fourth-order method on the optimal design to obtain the spanwise lift distribution.

Table C.1: Summary of adjoint solve with a drag objective at the first design iteration of the inviscid twist optimization study for various accuracy levels. Computations performed over 120 processors.

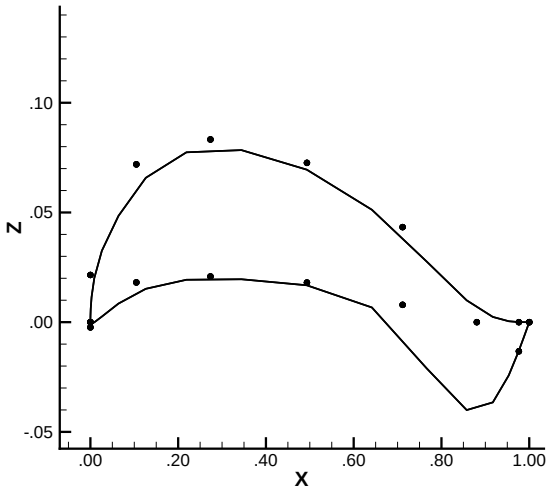
Grid	Residual evaluation time(s)	Krylov iterations	Adjoint time(s)	Time Fraction of Flow Analysis
Second-order				
G_1	0.00382	810	30.7	0.441
G_2	0.00816	1057	73.0	0.531
G_3	0.01470	1424	176.7	0.637
G_4	0.03233	1657	396.4	0.690
G_5	0.06850	2259	1084.6	0.875
Third-order				
G_1	0.00458	774	30.0	0.417
G_2	0.01042	1031	72.3	0.509
G_3	0.01827	1134	145.3	0.520
G_4	0.03759	1428	351.3	0.604
G_5	0.08006	1718	857.9	0.682
Fourth-order				
G_1	0.00589	2710	110.9	0.683
G_2	0.01291	3425	259.8	0.711
G_3	0.02172	3977	540.7	0.682
G_4	0.04582	5358	1374.5	0.764
G_5	0.09489	7243	3751.8	0.821

Table C.2: Summary of adjoint solve with a lift objective at the first design iteration of the inviscid twist optimization study for various accuracy levels. Computations performed over 120 processors.

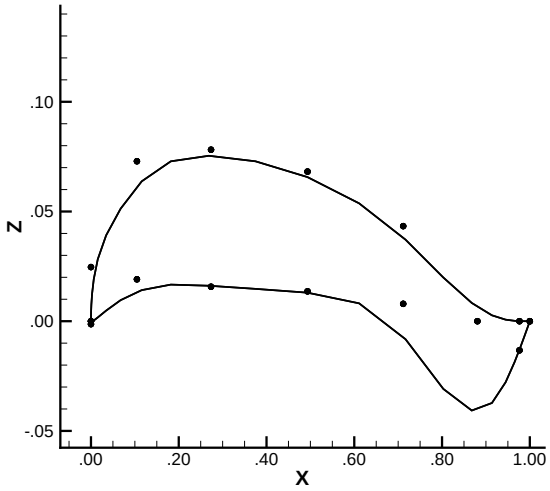
Grid	Residual evaluation time (s)	Krylov iterations	Adjoint time(s)	Time Fraction of Flow Analysis
Second-order				
G_1	0.00382	895	33.8	0.486
G_2	0.00816	1145	78.5	0.570
G_3	0.01470	1491	185.1	0.667
G_4	0.03233	1812	432.1	0.753
G_5	0.06850	2349	1128.6	0.911
Third-order				
G_1	0.00458	818	31.8	0.442
G_2	0.01042	1026	71.9	0.506
G_3	0.01827	1236	159.1	0.570
G_4	0.03759	1594	392.1	0.674
G_5	0.08006	1957	972.4	0.773
Fourth-order				
G_1	0.00589	3134	128.1	0.789
G_2	0.01291	3703	279.4	0.765
G_3	0.02172	4459	603.6	0.761
G_4	0.04582	5973	1527.9	0.849
G_5	0.09489	8355	4339.9	0.950



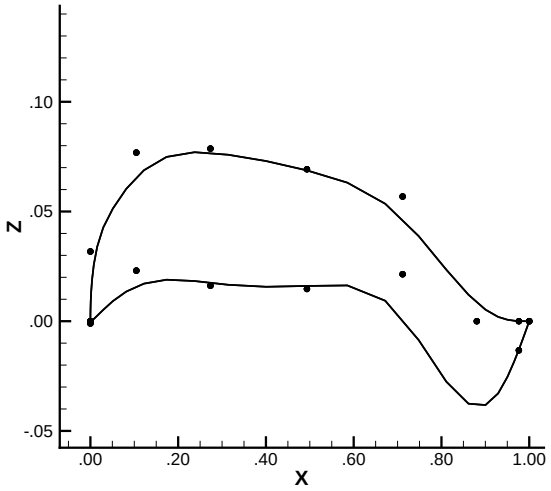
(a) Optimized sectional shape on grid G_1.



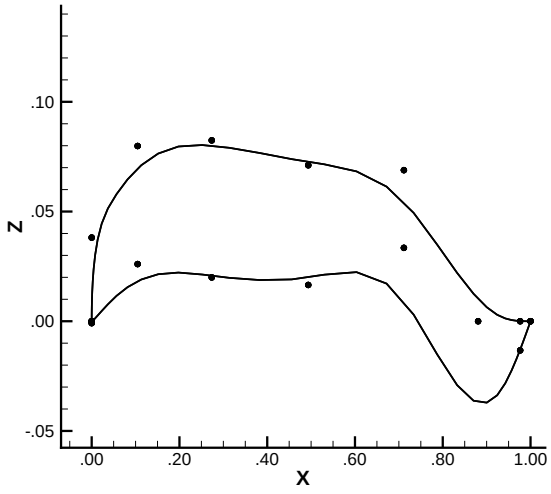
(b) Optimized sectional shape on grid G_2.



(c) Optimized sectional shape on grid G_3.

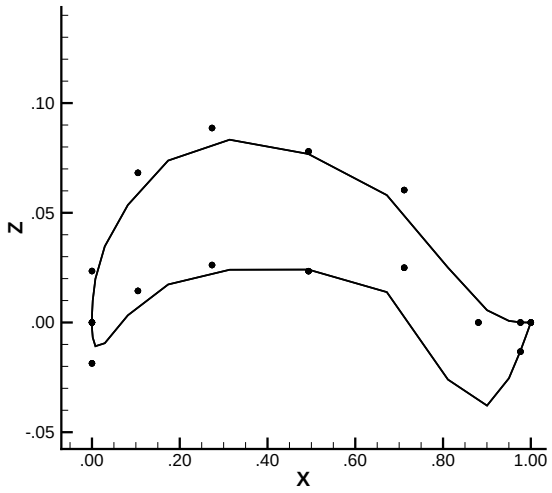


(d) Optimized sectional shape on grid G_4.

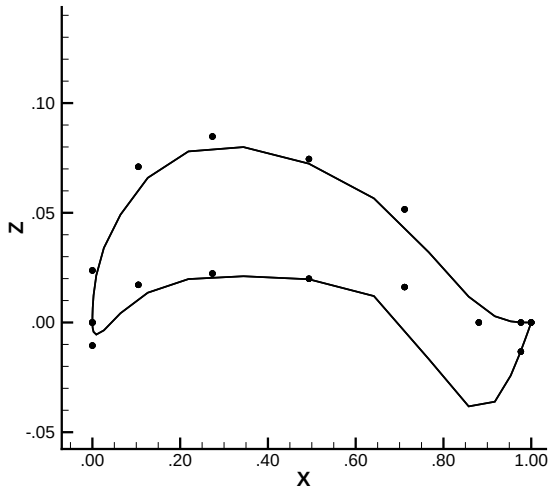


(e) Optimized sectional shape on grid G_5.

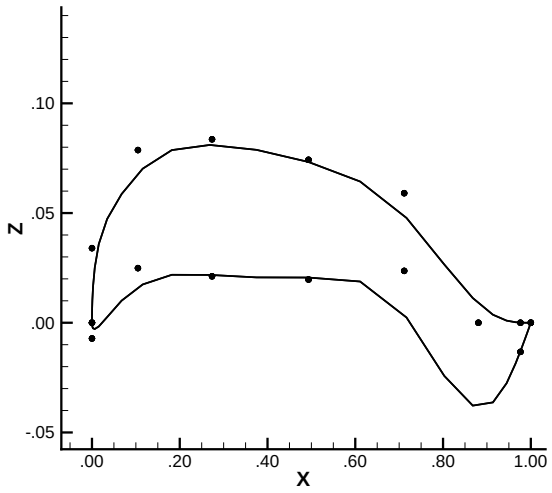
Figure C.4: Optimized sectional shapes obtained with the second-order method for the two-dimensional viscous laminar section optimization. Control points are shown as solid dots and the B-spline surface as a solid line.



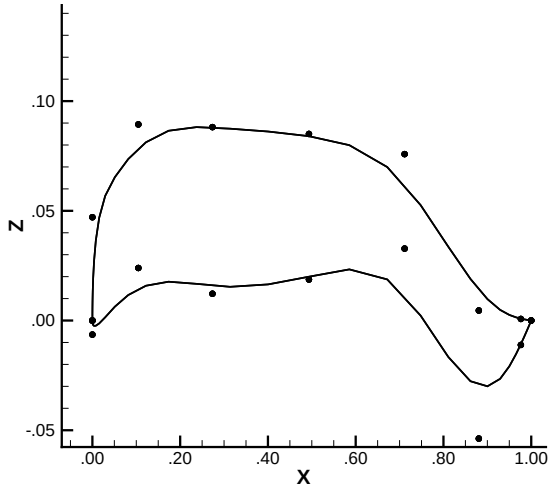
(a) Optimized sectional shape on grid G_1.



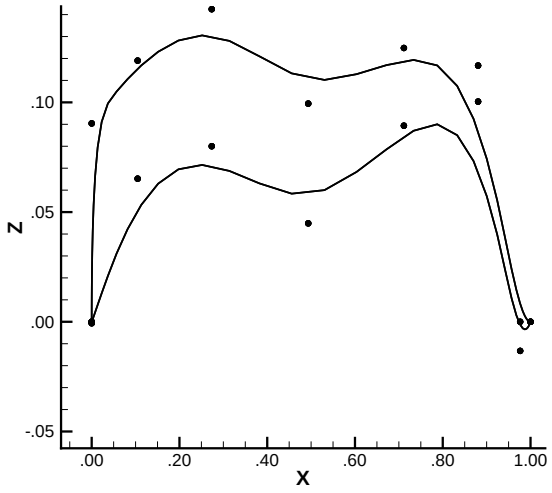
(b) Optimized sectional shape on grid G_2.



(c) Optimized sectional shape on grid G_3.

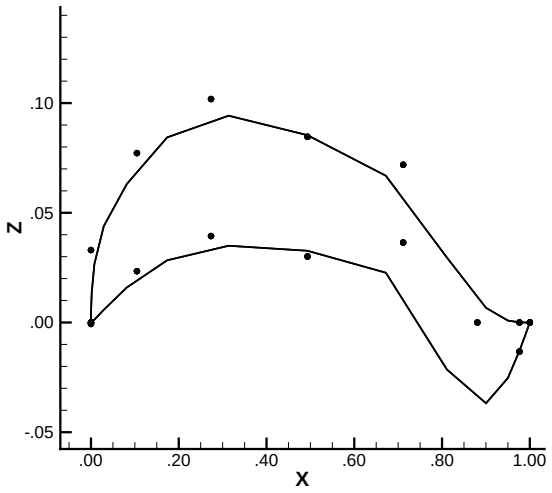


(d) Optimized sectional shape on grid G_4; optimization did not fully converged.

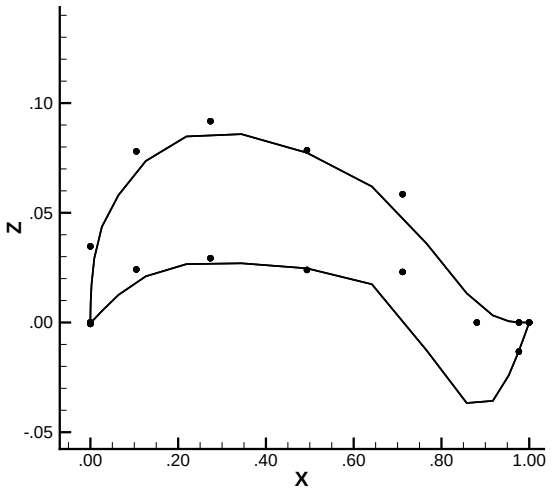


(e) Optimized sectional shape on grid G_5; optimization did not fully converged.

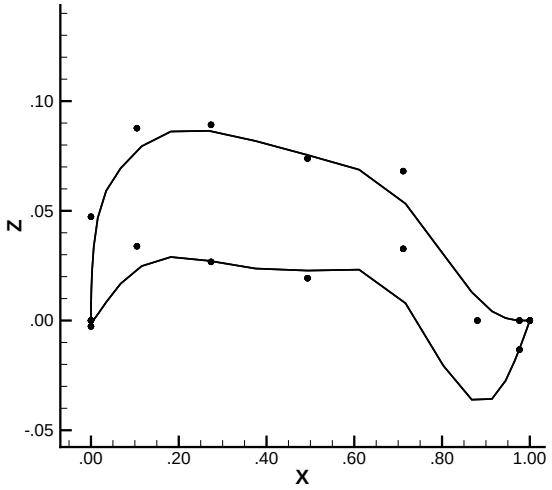
Figure C.5: Optimized sectional shapes obtained with the third-order method for the two-dimensional viscous laminar section optimization. Control points are shown as solid dots and the B-spline surface as a solid line.



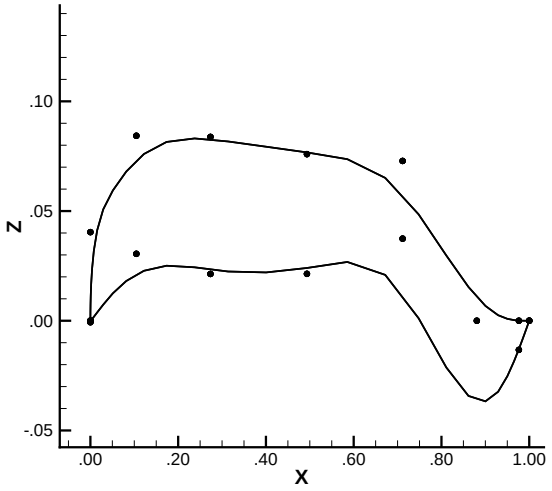
(a) Optimized sectional shape on grid G_1.



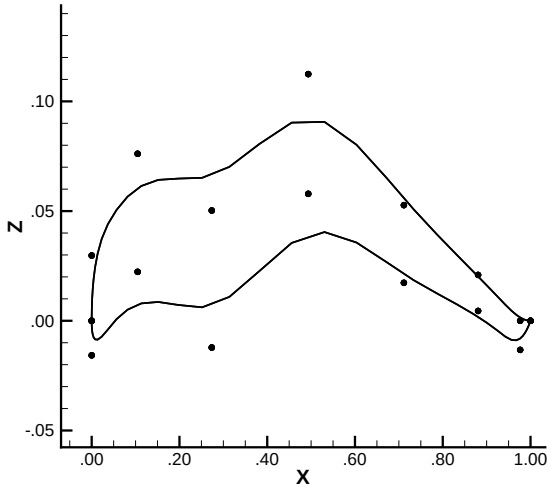
(b) Optimized sectional shape on grid G_2.



(c) Optimized sectional shape on grid G_3; optimization did not fully converge.



(d) Optimized sectional shape on grid G_4.



(e) Optimized sectional shape on grid G_5; optimization did not fully converge.

Figure C.6: Optimized sectional shapes obtained with the fourth-order method for the two-dimensional viscous laminar section optimization. Control points are shown as solid dots and the B-spline surface as a solid line.

Table C.3: Summary of adjoint solve with a lift-over-drag objective at the first design iteration of the two-dimensional viscous laminar section optimization study for various accuracy levels. Computations performed over 20 processors.

Grid	Residual evaluation time (s)	Krylov iterations	Adjoint time (s)	Time Fraction of Flow Analysis
Second-order				
G_1	0.04304	449	129.0	0.359
G_2	0.06514	475	194.9	0.389
G_3	0.11275	521	347.1	0.417
G_4	0.21004	551	585.6	0.433
G_5	0.30703	614	966.4	0.452
Third-order				
G_1	0.05579	492	181.7	0.436
G_2	0.08314	506	268.3	0.456
G_3	0.14158	527	441.5	0.476
G_4	0.24525	586	792.0	0.523
G_5	0.38519	654	1365.1	0.561
Fourth-order				
G_1	0.07748	807	403.9	0.603
G_2	0.11464	810	582.0	0.599
G_3	0.18933	969	1085.9	0.722
G_4	0.33281	943	1734.5	0.673
G_5	0.49748	1021	2826.0	0.713

Table C.4: Summary of adjoint solve with a drag objective at the first design iteration of the three-dimensional viscous laminar span optimization study for various accuracy levels. Computations performed over 120 processors.

Grid	Residual evaluation time (s)	Krylov iterations	Adjoint time (s)	Time Fraction of Flow Analysis
Second-order				
G_1	0.047200	385	130.6	0.379
G_2	0.081063	427	252.2	0.411
G_3	0.185486	480	570.0	0.409
G_4	0.387711	586	1431.8	0.467
Third-order				
G_1	0.052035	372	154.971	0.437
G_2	0.099623	425	311.003	0.478
G_3	0.214524	469	682.157	0.499
G_4	0.462070	574	1780.625	0.584
Fourth-order				
G_1	0.071036	1056	777.568	1.339
G_2	0.132531	1110	1066.840	0.998
G_3	0.276627	1331	2512.152	0.952
G_4	0.586296	1623	6357.037	0.996

Table C.5: Summary of adjoint solve with a lift objective at the first design iteration of the three-dimensional viscous laminar span optimization study for various accuracy levels. Computations performed over 120 processors.

Grid	Residual evaluation time (s)	Krylov iterations	Adjoint time (s)	Equivalent residual evaluations
Second-order				
G_1	0.04720	508	171.5	0.498
G_2	0.08106	626	370.2	0.604
G_3	0.18548	652	771.9	0.554
G_4	0.38771	816	1995.5	0.651
Third-order				
G_1	0.05203	513	214.0	0.603
G_2	0.09962	583	428.8	0.660
G_3	0.21452	653	950.0	0.696
G_4	0.46207	751	2347.1	0.769
Fourth-order				
G_1	0.07103	1502	826.8	1.424
G_2	0.13253	1556	1496.2	1.400
G_3	0.27662	1626	3070.1	1.164
G_4	0.58629	2041	7989.1	1.252

References

[1] N. Albin and J. Klarmann, *Existence of SBP operators with diagonal norm.* Department of Mathematics, Kansas State University, March 2014.

[2] S. R. Allmaras, *Contamination of laminar boundary layers by artificial dissipation in Navier-Stokes solutions*, in Conference on Numerical Methods in Fluid Dynamics, Reading UK, April 1992.

[3] W. K. Anderson and D. L. Bonhaus, *Airfoil design on unstructured grids for turbulent flows*, AIAA Journal, 37 (1999), pp. 185–191.

[4] J. Andren, H. Gao, M. Yano, D. L. Darmofal, C. Ollivier-Gooch, and Z. J. Wang, *A comparison of high-order methods on a set of canonical aerodynamic applications*, in 20th AIAA Computational Fluid Dynamics Conference, no. AIAA–2011–3230, Honolulu, Hawaii, June 2011.

[5] ANSYS, Inc., *ICEM CFD 12.1 User Manual*, November 2009.

[6] K. P. Apponsah, *A load-balancing tool for structured multi-block CFD applications applied to a parallel Newton-Krylov algorithm*, Master's thesis, University of Toronto, 2012.

[7] M. B. Azab, *Aerodynamic optimization using high-order finite-volume CFD simulations*, PhD thesis, The University of British Columbia, October 2011.

[8] M. B. Azab and C. Ollivier-Gooch, *High-order two dimensional aerodynamic optimization using unstructured grids and adjoint sensitivity computations*, in 48th AIAA Aerospace Sciences Meeting, no. AIAA–2010–1433, Orlando, Florida, January 2010.

[9] ——, *Constrained and unconstrained aerodynamic and quadratic programming optimization using high-order finite-volume method and adjoint sensitivity computations*, in 49th AIAA Aerospace Sciences Meeting, no. AIAA–2011–183, Orlando, Florida, January 2011.

[10] ——, *High-order aerodynamic optimization using new hybrid sequential quadratic programming-particle swarm intelligence technique*, in 50th AIAA Aerospace Sciences Meeting, no. AIAA–2012–0730, Nashville, Tennessee, January 2012.

[11] T. J. Barth, *Parallel CFD algorithms on unstructured meshes*, tech. rep., , 1995. AGARD–R–807.

[12] F. Bisson, S. Nadarajah, and D. Shi-Dong, *Adjoint-based aerodynamic optimization of benchmark problems*, in 52nd Aerospace Sciences Meeting, no. AIAA–2014–0412, National Harbor, Maryland, January 2014.

[13] O. Chernukhin, *Global optimization algorithms for aerodynamic design*, Master's thesis, University of Toronto, 2011.

[14] O. Chernukhin and D. W. Zingg, *Multimodality and global optimization in aerodynamic design*, AIAA Journal, 51 (2013), pp. 1342–1354.

[15] T. Chisholm, *A fully coupled Newton-Krylov solver with a one-equation turbulence model*, PhD thesis, University of Toronto, 2006.

[16] D. L. Darmofal, S. R. Allmaras, M. Yano, and J. Kudo, *An adaptive, high-order discontinuous galerkin finite element method for aerodynamics*, in 21st AIAA Computational Fluid Dynamics Conference, no. AIAA–2013–2871, San Diego, CA, June 2013.

[17] S. De Rango, *High-order spatial discretization for turbulent aerodynamic flows*, PhD thesis, University of Toronto, 2001.

[18] S. De Rango and D. W. Zingg, *Higher-order spatial discretization for turbulent aerodynamic computations*, AIAA Journal, 39 (2001), pp. 1296–1304.

[19] E. de Sturler, *Nested Krylov methods based on GCR*, Journal of Computational and Applied Mathematics, 67 (1996), pp. 15–41.

[20] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM Journal of Numerical Analysis, 36 (1999), pp. 864–889.

[21] D. C. Del Rey Fernández, J. E. Hicken, and D. W. Zingg, *Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations*, Computers and Fluids, 95 (2014), pp. 171–196.

[22] D. C. Del Rey Fernández and David W. Zingg, *High-order compact-stencil summation-by-parts operators for the second derivative with variable coefficients*, in ICCFD7, no. ICCFD7–2012–2803, Big Island, Hawaii, July 2012.

[23] R. W. Derksen, M. Agelinchaab, and M. Tachie, *Characteristics of the flow over a NACA 0012 airfoil at low Reynolds numbers*, WIT Transactions on Engineering Sciences, 59 (2008), pp. 143–152.

[24] S. C. Dias, *A high-order parallel Newton-Krylov flow solver for the Euler equations*, Master's thesis, University of Toronto, 2009.

[25] S. C. Dias and D. W. Zingg, *A high-order parallel Newton-Krylov flow solver for the Euler equations*, in 19th AIAA Computational Fluid Dynamics Conference, no. AIAA–2009–3657, San Antonio, Texas, June 2009.

[26] P. Diener, E. N. Dorband, E. Schnetter, and M. Tiglio, *Optimized high-order derivative and dissipation operators satisfying summation by parts, and applications in three-dimensional multi-block evolutions*, Journal of Scientific Computing, 32 (2007), pp. 109–145.

[27] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: an SQP algorithm for large-scale constrained optimization*, SIAM Journal on Optimization, 12 (2002), pp. 979–1006.

[28] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, June 2008.

[29] J. E. Hicken, *Efficient algorithms for future aircraft design: contributions to aerodynamic shape optimization*, PhD thesis, University of Toronto, 2009.

[30] J. E. Hicken, *Aerodynamic design optimization workshop: Twist optimization case*, tech. rep., Rensselaer Polytechnic Institute Dept. of Mechanical, Aerospace, and Nuclear Eng., Troy, NY, United States, 12180, January 2013.

[31] J. E. Hicken, H. Buckley, M. Osusky, and D. W. Zingg, *Dissipation-based continuation: a globalization for inexact-Newton solvers*, in 20th AIAA Computational Fluid Dynamics Conference, no. AIAA–2011–3237, Honolulu, Hawaii, June 2011.

[32] J. E. Hicken, M. Osusky, and D. W. Zingg, *Comparison of parallel preconditioners for a Newton-Krylov flow solver*, in ICCFD6, 2010.

[33] J. E. Hicken and D. W. Zingg, *Parallel Newton-Krylov solver for the Euler equations discretized using simultaneous-approximation terms*, AIAA Journal, 46 (2008), pp. 2773–2786.

[34] ——, *Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement*, AIAA Journal, 48 (2010), pp. 400–413.

[35] ——, *Induced drag minimization of nonplanar geometries based on the Euler equations*, AIAA Journal, 48 (2010), pp. 2564–2575.

[36] ——, *A simplified and flexible variant of GCROT for solving nonsymmetric linear systems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 1672–1694.

[37] ——, *Superconvergent functional estimates from summation-by-parts finite-difference discretizations*, SIAM Journal on Scientific Computing, 33 (2011), pp. 893–922.

[38] ——, *Summation-by-parts operators and high-order quadrature*, Journal of Computational and Applied Mathematics, 237 (2013), pp. 111–125.

[39] ——, *Dual consistency and functional accuracy: a finite-difference perspective*, Journal of Computational Physics, 256 (2014), pp. 161–182.

[40] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats, *An assesment of airfoil design by numerical optimization*, tech. rep., NASA/TMX-3902, National Aeronautics and Space Administration, Ames Research Center, Moffat Field, CA,94035-1000, July 1974.

[41] X. Huan, J. E. Hicken, and D. W. Zingg, *Interface and boundary schemes for high-order methods*, in 19th AIAA Computational Fluid Dynamics Conference, no. AIAA–2009–3658, San Antonio, Texas, June 2009.

[42] International Air Transport Association, *Aviation and climate change: Pathway to carbon-neutral growth in 2020*, tech. rep., IATA.

[43] ——, *Air passenger market analysis*, tech. rep., IATA, December 2013.

[44] ——, *Technology roadmap*, tech. rep., IATA, June 2013.

[45] INTERNATIONAL CIVIL AVIATION ORGANIZATION, *2013 ICAO air transport results confirm robust passenger demand, sluggish cargo market*, tech. rep., ICAO, December 2013.

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

[47] A. JAMESON, W. SCHMIDT, AND E. TURKEL, *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes*, in 14th Fluid and Plasma Dynamics Conference, no. AIAA Paper 91–1259, Palo Alto, CA, 1981.

[48] I. R. KHAN AND R. OHBA, *Closed-form expressions for the finite difference approximations of first and higher derivatives based on Taylor series*, Journal of Computational and Applied Mathematics, (1999), pp. 179–193.

[49] D. A. KNOLL AND D. E. KEYES, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.

[50] H. O. KREISS AND G. SCHERRER, *Finite element and finite difference methods for hyperbolic partial differential equations*, Mathematical Aspects of Finite Elements in Partial Differential Equations, (1974), pp. 195–212.

[51] L. LEHNER, O. REULA, AND M. TIGLIO, *Multi-block simulations in general relativity: high-order discretizations, numerical stability, and applications*, General Relativity and Quantum Cosmology, 22 (2005), pp. 5283–5321.

[52] H. LOMAX, T. H. PULLIAM, AND D. W. ZINGG, *Fundamentals of Computational Fluid Dynamics*, Springer-Verlag, 2nd ed., 2001.

[53] J. A. MAEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[54] J. R. R. A. MARTINS, *A coupled-adjoint method for high-fidelity aero-structural optimization*, PhD thesis, Stanford University, October 2002.

[55] 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.

[56] K. MATTSSON AND M. ALMQUIST, *A solution to the stability issues with block norm summation by parts operators*, Journal of Computational Physics, 253 (2013), pp. 418–442.

[57] K. MATTSSON, M. ALMQUIST, AND M. H. CARPENTER, *Optimal diagonal-norm SBP operators*, Journal of Computational Physics, 264 (2014), pp. 91–111.

[58] K. MATTSSON AND J. NORDSTRÖM, *Summation by parts operators for finite difference approximations of second derivatives*, Journal of Computational Physics, 199 (2004), pp. 503–540.

[59] K. MATTSSON, M. SVÄRD, AND J. NORDSTRÖM, *Stable and accurate artificial dissipation*, Journal of Scientific Computing, 21 (2004), pp. 57–79.

[60] C. Michalak and C. Ollivier-Gooch, *Globalized matrix-explicit Newton-GMRES for the high-order accurate solution of the Euler equations*, Computers and Fluids, 39 (2010), pp. 1156–1167.

[61] S. Nadarajah, *Aerodynamic design optimization: Drag minimization of the NACA 0012 in transonic inviscid flow*, tech. rep., McGill University, July 2013.

[62] ——, *Aerodynamic design optimization: Drag minimization of the RAE 2822 in transonic inviscid flow*, tech. rep., McGill University, July 2013.

[63] A. Nejat and C. Ollivier-Gooch, *A high-order accurate unstructured finite volume Newton-Krylov algorithm for inviscid compresible flows*, Journal of Computational Physics, 227 (2008), pp. 2582–2609.

[64] M. Nemec and M. J. Aftosmis, *Toward automatic verification of goal-oriented simulations.* preprint submitted to JANNAF, July 2014.

[65] M. Nemec, M. J. Aftosmis, and M. Wintzer, *Adjoint-based adaptive mesh refinement for complex geometries*, in 46th AIAA Aerospace Sciences Meeting, no. AIAA–2008–0725, Reno, NV, January 2008.

[66] J. Nordström, J. Gong, E. van der Weide, and M. Svärd, *A stable and conservative high-order multi-block method for the compressible navier-stokes equations*, Journal of Computational Physics, 228 (2009), pp. 9020–9035.

[67] P. Olsson, *Summation by parts, projections, and stability, I*, Mathematics of Computation, 64 (1995), pp. 1035–1065.

[68] ——, *Summation by parts, projections, and stability, II*, Mathematics of Computation, 64 (1995), pp. 1473–1493.

[69] L. Osusky and D. W. Zingg, *Application of an efficient Newton-Krylov algorithm for aerodynamic shape optimization based on the Reynolds-averaged Navier Stokes equations*, in 21st AIAA Computational Fluid Dynamics Conference, no. AIAA–2013–2584, San Diego, CA, June 2013.

[70] L. M. Osusky, *A novel numerical tool for aerodynamic shape optimization in turbulent flow*, PhD thesis, University of Toronto, 2014.

[71] L. M. Osusky and D. W. Zingg, *Lift-constrained drag minimization of a wing allowing section and twist variation with flow governed by the Reynolds-averaged Navier-Stokes equations*, tech. rep., University of Toronto Institute for Aerospace Studies, July 2013.

[72] M. Osusky, *A parallel Newton-Krylov-Schur algorithm for the Reynolds-averaged Navier-Stokes equations*, PhD thesis, University of Toronto, 2013.

[73] W. F. Phillips, S. R. Fugal, and R. E. Spall, *Minimizing induced drag with wing twist, computational-fluid-dynamics validation*, Journal of Aircraft, 43 (2006), pp. 437–444.

[74] T. H. Pulliam, *Efficient solution methods for the Navier-Stokes equations*, tech. rep., Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Rhode–Saint–Genèse, Belguim, January 1986.

[75] T. H. Pulliam and D. W. Zingg, *Fundamental Algorithms in Computational Fluid Dynamics*, Springer, 2014.

[76] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2nd ed., 2003.

[77] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[78] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM Journal on Scientific Computing, 21 (1999), pp. 1337–1356.

[79] B. Sjögreen and H. C. Yee, *On tenth order central spatial schemes*, in 5th International Symposium on Turbulence and Shear Flow Phenomena, no. UCRL–CONF–230974, Munich, Germany, August 2007.

[80] D. N. Srinath and S. Mittal, *Optimal airfoil shapes for low reynolds number flows*, International Journal for Numerical Methods in Fluids, 61 (2009), pp. 355–381.

[81] M. Svärd, *On coordinate transformation for summation-by-parts operators*, Journal of Scientific Computing, 20 (2004), pp. 29–42.

[82] R. C. Swanson and E. Turkel, *On central-difference and upwind schemes*, Journal of Computational Physics, 101 (1992), pp. 292–306.

[83] K. Telidetzki, L. Osusky, and D. W. Zingg, *Application of Jetstream to a suite of aerodynamic shape optimization problems*, in 52nd Aerospace Sciences Meeting, no. AIAA–2014–0908, National Harbor, Maryland, January 2014.

[84] J. F. Thompson, Z. Warsi, and C. W. Mastin, *Numerical Grid Generation*, Elsevier Science Publishing Co., Inc., 1985.

[85] A. H. Truong, C. A. Oldfield, and D. W. Zingg, *Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization*, AIAA Journal, 46 (2008), pp. 1695–1704.

[86] M. Vinokur, *On one-dimensional stretching functions for finite-difference calculations*, Journal of Computational Physics, 50 (1983), pp. 215–234.

[87] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal, *High-order CFD method: current status and perspective*, International Journal for Numerical Methods in Fluids, 72 (2013), pp. 811–845.

[88] F. M. White, *Viscous Fluid Flow*, McGraw-Hill Book Company, New York, 1974.

[89] D. W. Zingg, M. Nemec, and T. H. Pulliam, *A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization*, European Journal of Computational Mechanics, 17 (2008), pp. 103–126.