

A NEWTON-KRYLOV APPROACH TO AERODYNAMIC SHAPE OPTIMIZATION IN
THREE DIMENSIONS

by

Timothy Man-Ming Leung

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

Copyright © 2010 by Timothy Man-Ming Leung

Abstract

A Newton-Krylov Approach to Aerodynamic Shape Optimization in Three Dimensions

Timothy Man-Ming Leung

Doctor of Philosophy

Graduate Department of Aerospace Science and Engineering

University of Toronto

2010

A Newton-Krylov algorithm is presented for aerodynamic shape optimization in three dimensions using the Euler equations. An inexact-Newton method is used in the flow solver, a discrete-adjoint method to compute the gradient, and the quasi-Newton optimizer to find the optimum. A Krylov subspace method with approximate-Schur preconditioning is used to solve both the flow equation and the adjoint equation. Basis spline surfaces are used to parameterize the geometry, and a fast algebraic algorithm is used for grid movement. Accurate discrete-adjoint gradients can be obtained in approximately one-fourth the time required for a converged flow solution. Single- and multi-point lift-constrained drag minimization optimization cases are presented for wing design at transonic speeds. In all cases, the optimizer is able to efficiently decrease the objective function and gradient for problems with hundreds of design variables.

DEDICATED TO LILIAN,
MY WIFE, FRIEND AND SOUL MATE.

Acknowledgements

The past several years at UTIAS has been a wonderful intellectual experience for me.

I would like to thank my supervisor, Professor David W. Zingg, for giving me the opportunity to study under his guidance. His valuable insights and suggestions were often the starting points of many ideas. I would also like to extend my thanks to the members of my Doctoral Examination Committee, Professor Joaquim R. R. A. Martins, and Professor Jorn S. Hansen, for their experience and enthusiasm.

My most sincere thanks to past and present colleagues at the CFD Lab. I am most indebted to Jason Hicken, who developed much of the flow solver technology. Special thanks also go to CFD lab administrators Scott Northrup and Sean McDonald for their help with everything Linux, and to answer some of my most ridiculous questions. Finally, I would like to thank Michal Osusky for his help with grid generation, and to John Gatsis, James McDonald, Lana Olague, Jai Sachdev and Peterson Wong for their lively and engaging discussions over the course of my time here.

Financial support from the Government of Ontario through the Ontario Graduate Scholarship for Science and Technology, the Kenneth Molson Fellowship, Bombardier Aerospace, MITACS and the University of Toronto is gratefully appreciated.

Timothy Leung

University of Toronto Institute for Aerospace Studies

January 14, 2010

Contents

List of Figures	ix
List of Tables	xi
List of Symbols and Abbreviations	xii
1 Introduction	1
1.1 Motivation	1
1.1.1 Airline Profitability	1
1.1.2 Environmental Concerns	3
1.1.3 Costs in the Design Cycle	3
1.1.4 Trends in Aircraft Design	4
1.2 Aerodynamic Shape Optimization	5
1.2.1 Early Research	5
1.2.2 Newton-Krylov Flow Solver	5
1.2.3 Optimizers	6
1.2.4 Gradient Evaluation	9
1.2.5 Geometry Parameterization	11
1.2.6 Multi-Point Optimization	12
1.3 Thesis Objectives	12
1.4 Thesis Outline	13
2 Optimization Problem	15
2.1 Problem Formulation	15
2.2 Objective Functions	16
2.2.1 Lift-Constrained Drag Minimization	16
2.2.2 Inverse Design	18
2.3 The Newton-Krylov Approach	18

3	Geometry Parameterization and Design Variables	20
3.1	B-Spline Curve and Surface Formulation	21
3.2	Surface Grid Approximation and Generation	22
3.3	Design Variables	24
3.3.1	Planform Design Variables	25
3.4	Geometric Constraints	27
3.4.1	Volume Constraint	28
3.4.2	Thickness Constraints	28
4	Flow Analysis	29
4.1	Governing Equations	29
4.2	Coordinate Transformation	30
4.3	Spatial Discretization	32
4.3.1	Inviscid Fluxes	32
4.3.2	Artificial Dissipation	33
4.3.3	Block Interfaces and Boundary Conditions	35
4.4	Newton-Krylov Flow Solver	39
4.4.1	Preconditioned GMRES	41
4.4.2	Approximate Schur Preconditioning	42
5	Optimizer	46
5.1	Gradient Evaluation	46
5.1.1	Finite-Difference Gradient	46
5.1.2	Direct and Adjoint Gradients	47
5.1.3	Complex-Step Gradients	50
5.2	BFGS Optimizer	50
5.3	Design Variable Scaling	52
6	Grid Movement Algorithm	54
7	Validation	57
7.1	Overview	57
7.2	Flow Solver Accuracy	57
7.3	Comparison of Grid Movement Algorithms	63
7.4	Gradient Accuracy Evaluation	68
7.5	Inverse Design	71

8	Design Examples	76
8.1	Single-Point Lift-Constrained Drag Minimization	76
8.2	Effects of Scaling on Optimizer Convergence	83
8.3	Multi-Point Drag Minimization	84
8.4	Optimization Using Thickness Constraints	87
8.5	Effect of the Initial Geometry	94
9	Conclusions and Recommendations	100
9.1	Summary of the Newton-Krylov Algorithm	100
9.2	Conclusions and Original Contributions	101
9.2.1	Optimizer Effectiveness	101
9.2.2	Application of Geometric and Performance Constraints	102
9.2.3	Gradient Accuracy	103
9.2.4	Geometry Parameterization and Design Variables	103
9.2.5	Grid Movement Algorithm	104
9.3	Future Work	104
	References	106
	Appendices	
A	Additional Notes on Range Equation	121
B	Alternative Parameterization Strategies	123
B.1	Hicks-Henne Bump Functions	123
B.2	Discrete Approach	125
C	Derivation of the Adjoint Method Using Lagrange Multipliers	127
D	GMRES	129
D.1	GMRES	129
D.2	Right-Preconditioned GMRES	130
D.3	Flexible GMRES	130
E	Derivation of Partial Derivatives for Adjoint Calculations	132
E.1	Derivation of $\partial\mathcal{J}/\partial\mathbf{Q}$	132
E.2	Derivation of $\partial\mathcal{J}/\partial\alpha$	133
E.3	Derivation of $\partial\mathbf{R}/\partial\alpha$	134

List of Figures

1.1	Aircraft orders for Boeing and Airbus since 1989.	2
2.1	Solving for objective function	16
3.1	Example of third order B-spline basis functions	21
3.2	Example of a third-order B-spline curve.	22
3.3	B-spline surface parameterization of an ONERA M6 wing.	24
3.4	Examples of changing sweep angle	26
3.5	Positive dihedral angles at wing root and mid-span.	27
3.6	Wing twist example.	27
4.1	Mapping from physical domain to computational domain	31
4.2	An example H-H topology grid around a DPW-W1 wing	33
4.3	Stencil for evaluating flux and dissipation without SATs	35
4.4	One-dimensional domain with two sub-domains	36
4.5	Stencil for SAT terms at block interfaces	38
4.6	Computing \mathbf{Q}_{BODY} for SAT terms at body surface boundaries	39
6.1	Algebraic grid movement algorithm applied to a structured volume grid	55
6.2	Example of algebraic grid movement algorithm.	56
7.1	12-block H-H grid around an ONERA M6 wing	59
7.2	Convergence history for flow solver validation case at $M = 0.699$, $\alpha = 3.06^\circ$	60
7.3	Pressure coefficients on the ONERA M6 wing at $M = 0.669$, $\alpha = 3.06$	61
7.4	Mach number contours for the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08^\circ$	61
7.5	Pressure coefficients on the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08$	62
7.6	Convergence history for flow solver validation case at $M = 0.835$, $\alpha = 4.08^\circ$	63
7.7	Convergence history for flow solver validation case at $M = 0.835$, $\alpha = 4.08^\circ$	64
7.8	Mach number contours for the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07^\circ$	64
7.9	Pressure coefficients on the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07$	65
7.10	Coarse ONERA M6 grid used	66
7.11	B-spline parameterization of the coarse ONERA M6 grid	66
7.12	New grids generated with the two algorithms	67
7.13	Comparison of grid movement methods, with $\Delta\Lambda = +5^\circ$	68
7.14	48-block H-H grid around a ONERA M6 wing	72
7.15	B-spline parameterization of the ONERA M6 wing surface	73
7.16	Symmetry plane of initial grid for inverse design	73
7.17	Convergence histories for the inverse design validation case	74
7.18	Comparison of initial and final wing cross-sections for inverse design	75

8.1	B-spline surface parameterization of the ONERA M6 wing for single- and multi-point optimization	78
8.2	Convergence history for the single-point optimization of the ONERA M6 wing	79
8.3	Spanwise lift distribution for the single-point optimization	80
8.4	Comparison of Mach contours on the wing top surface for the single-point optimization case	80
8.5	Surface pressure coefficient plots and wing sections at six spanwise stations for the single-point optimization case	81
8.6	Mach contours after 20 and 50 optimization iterations	82
8.7	Convergence histories with different design variable scaling	83
8.8	Drag divergence plot for the optimized wing	85
8.9	Convergence history for the three-point optimization case	86
8.10	Comparison of wing section shapes	87
8.11	Drag divergence plot of the (three-point) optimized wing at fixed $C_L = 0.307$	88
8.12	48-block grid around a DPW-W1 used for the design examples	90
8.13	B-spline parameterization of the DPW-W1 wing	91
8.14	Convergence history for the optimization of the DPW-W1 wing	92
8.15	Surface pressure coefficient plots and wing sections at six spanwise stations for the DPW-W1 single-point optimization case	93
8.16	Drag coefficients at $C_L = 0.50$ for the DPW-W1 optimization case	94
8.17	12-block grid around tapered wings used for the design examples	96
8.18	Convergence histories with both initial geometries	97
8.19	Initial and final geometries for the design case with different initial geometries.	98
8.20	Final pressure coefficients and wing sections for both initial geometries	99
B.1	A set of 20 Hicks-Henne bump functions	124
B.2	Approximating an RAE 2822 airfoil using Hicks-Henne bump function	124
B.3	Example of a surface perturbation using a 3D Hicks-Henne formulation.	125
E.1	Triangular elements used to compute body forces.	133
F.1	Drag divergence of various optimized wings at $C_L = 0.307$	136
F.2	Comparison of wing section shapes of single-point optimized wings at various span-wise locations	137

List of Tables

1.1	Breakdown of operating costs for major airlines.	2
1.2	Comparison of genetic algorithms and gradient-based optimizers.	9
1.3	Other gradient-based approaches for 3D aerodynamic shape optimization.	14
5.1	Change in design variable during a typical optimization cycle.	53
7.1	Lift and drag coefficients for the ONERA M6 wing at $M = 0.699$, $\alpha = 3.06^\circ$. . .	59
7.2	Lift and drag coefficients for the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08^\circ$. . .	61
7.3	Lift and drag coefficients for the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07^\circ$. .	63
7.4	Gradients for the subsonic L/D maximization case with $\kappa_2 = 0.0$	69
7.5	Gradients for transonic lift-constrained drag minimization case with $\kappa_2 = 0.0$. .	69
7.6	Gradients for the subsonic L/D maximization case with $\kappa_2 = 1.0$	70
7.7	Gradients for transonic lift-constrained drag minimization case with $\kappa_2 = 1.0$. .	70
8.1	Planform parameters for the ONERA M6 wing	77
8.2	Drag reduction vs. iteration count for single-point optimization	82
8.3	Summary of different optimizers	88
8.4	Planform parameters for the DPW-W1 wing	89
8.5	Thickness targets for each thickness constraint	91
8.6	DPW-W1 optimization results compared to other optimizers	93
8.7	Planform parameters for the tapered wing	95
8.8	Leading-edge thickness constraints	95
8.9	Trailing-edge thickness constraints	96
8.10	Final C_L and C_D values for different initial geometries	97
F.1	Final L/D ratios and sweep angles of various optimized wings	136

List of Symbols and Abbreviations

Alphanumeric

- A**, **A**₁ Flow Jacobian and first-order flow Jacobian matrices¹
- $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$ Discrete inviscid convective fluxes in the computational domain
- G**, **G**_S Coordinates of the computational grid and volume grid
- M** Preconditioner matrix
- Q** Vector of discrete conservative flow variables
- Q**_{BODY}, **Q**_{FF} Body surface and far-field boundary boundary conditions
- R** Residual vector
- \mathcal{B} BFGS approximation of the inverse of the Hessian²
- \mathcal{G} Gradient vector
- \mathcal{G}_T Composite gradient for multi-point optimization
- \mathcal{H} Hessian matrix
- \mathcal{J} Objective function
- $\mathcal{J}_{p,V}$, $\mathcal{J}_{p,T}$ Penalty terms in the objective function for volume and thickness constraints
- \mathcal{J}_T Composite objective function for multi-point optimization
- \mathcal{L} , \mathcal{D} Lift and drag coefficients multiplied by the wing's reference (planform) area
- $\mathcal{N}_{i,k}$, $\mathcal{M}_{i,k}$ i -th basis functions of order k
- \mathcal{P}_n Search direction in the design space for the n -th iteration
- \mathcal{R} Residual normalized by the residual from first iteration
- \mathcal{U} , \mathcal{V} Matrices containing B-spline basis functions

¹Discrete quantities at grid nodes, such as grid coordinates **G**, flow variables **Q**, inviscid fluxes $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$, residual **R**, matrices **A**, **M**, and adjoint vector Ψ are all expressed in capitalized bold face while the continuous equivalent Q , \hat{E} , \hat{F} and \hat{G} are written in italics.

²Variables in the design space such as design variables, objective function \mathcal{J} , gradient \mathcal{G} , search direction \mathcal{P} , Hessian \mathcal{H} and its BFGS approximation \mathcal{B} are all expressed in capitalized script font.

\mathcal{X}_B	Vector or matrix containing the B-spline control points
$\mathcal{X}, \mathcal{X}_s$	Design variables, scaled design variables
\vec{a}	Position vector along a B-spline curve or surface
\vec{x}	Coordinates of a grid node
a	Speed of sound
a_j	Coefficients for the j -th Hicks-Henne bump function
AR	Wing aspect ratio
b	Wing span (from wing-tip to wing-tip)
c	Wing chord
c_1, c_2	Sufficient decrease and curvature coefficients for Wolfe conditions
C_j	Inequality constraints equations
C_L, C_D, C_p	Lift, drag and pressure coefficients
d_i	i -th element in the knot vector
e	Total energy
E, F, G	Inviscid convective fluxes in physical space
f_j	Hicks-Henne bump functions
h	Finite-difference step size
J	Metric Jacobian
j, k, m	Node indices
L, D	Lift and drag
M	Mach number
N_p	Number of surface nodes
N_X, N_{GX}	Total number of design variables, number of geometric design variables
p	Pressure
Q	Flow variables in physical space
R	Range of an aircraft
S	Wing reference (planform) area
S_k	Normalized arc-length along a grid line
T	Engine thrust

U, V, W Contravariant velocities

u, v, w Velocities in physical space

V Speed of the aircraft

W_F, W_S, W_P, W_R Weights of fuel, structure, payload and reserve fuel

W_i, W_f Initial and final weight of an aircraft

x, y, z Coordinates in physical space

Greek

α Angle of attack

β_n Step size for the optimization iteration n

Γ Dihedral angle

γ Ratio of specific heats

\hat{e}_k k -th unit vector

κ_2, κ_4 Second- and fourth-difference dissipation coefficients

Λ Sweep angle

λ Wing taper ratio

Ψ Adjoint vector

Ω Wing geometric twist angle

ω_L, ω_D Relative weights on lift and drag

ω_V, ω_T Penalty weights on volume and thickness constraints

$\phi(\beta)$ Objective function value along the search direction

ρ Density

ξ, η, ζ Coordinates in computational space

Abbreviations

AD Automatic/algorithmic Differentiation

BFGS Quasi-Newton optimizer based on Broyden, Fletcher, Goldfarb and Shanno

CAD Computer-Aided Design

CAPRI Computational Analysis and Programming Interface

CFD Computational fluid dynamics

CG Conjugate gradient Krylov method

FGMRES Flexible General Minimal RESidual Krylov method
GHG Greenhouse gases
GMRES General Minimal RESidual Krylov method
HPACF High Performance Advanced Computing Facility
IPCC Intergovernmental Panel on Climate Change
LAPACK Linear Algebra PACKage
LE Wing leading edge
MDO Multi-Disciplinary Optimization
MPI Message Passing Interface
NK Newton-Krylov method
NKS Newton-Krylov-Schwarz method
NURBS Non-Uniform Rational Basis Spline
RANS Reynolds-Averaged Navier-Stokes equations
RCEP Royal Commission on Environmental Pollution
SBP Summation by Parts
SNOPT Sparse Nonlinear OPTimizer
SQP Sequential Quadratic Programming
TE Wing trailing edge
TSFC Thrust Specific Fuel Consumption
UTIAS University of Toronto Institute for Aerospace Studies

Chapter 1

Introduction

1.1 Motivation

In the competitive business of commercial aviation, the type of aircraft that is designed, as well as what methodology is used to design those aircraft, depends heavily on what is the most cost effective. To the airlines (“operators”), the cost of ownership of an aircraft includes cost of purchase or lease, operating costs, and maintenance costs. In particular, fuel cost is the most significant expense for an operator. From the point of view of the aircraft manufacturers, there is the cost of development of an aircraft—both in time and in human resources—and manufacturing costs. For example, the development costs of the Airbus A380 and Boeing 787 programs are estimated at €11 billion and US\$10 billion respectively, although exact figures are difficult to obtain. Finally, there are also environmental concerns, most significantly through greenhouse gas (GHG) emissions at high altitudes, and the impact aviation has on global climate change, but this may include noise and pollution generated near airports as well.

1.1.1 Airline Profitability

The price of crude oil has risen dramatically in the past decade, from a low of \$11USD a barrel in early 1999 to more than \$140USD in the summer of 2008 [136, 194], before retreating to below \$50USD in early 2009 amidst a global economic recession. As jet fuel prices are closely linked to crude oil prices [78], this means significant uncertainties in fuel cost. Since 2006, fuel cost has replaced labour cost as the single most significant expense for an operator, accounting for 25.5% of the cost globally [2] (Table 1.1). In addition, the volatility of fuel prices is often cited as a main contributor to financial difficulties in the airline industry, which resulted in more than 25 airlines declaring bankruptcy worldwide in 2008.¹ In order to cope, airlines have

¹Through a process called hedging, operators sign long-term contracts with fuel suppliers to lock in future deliveries of fuel at a fixed cost to limit the uncertainty over future costs. But as American and European airlines

	North American		Europe		Asia Pacific		All Major Airlines	
	2001	2006	2001	2006	2001	2006	2001	2006
Labour	36.2%	25.2%	27.2%	25.8%	17.2%	17.2%	28.3%	23.3%
Fuel	13.4%	26.6%	12.2%	20.5%	15.7%	30.4%	13.6%	25.5%
Aircraft Rental/Lease	5.5%	3.7%	2.9%	3.1%	6.3%	2.4%	5.0%	3.5%

Table 1.1: Breakdown of operating costs for major airlines.

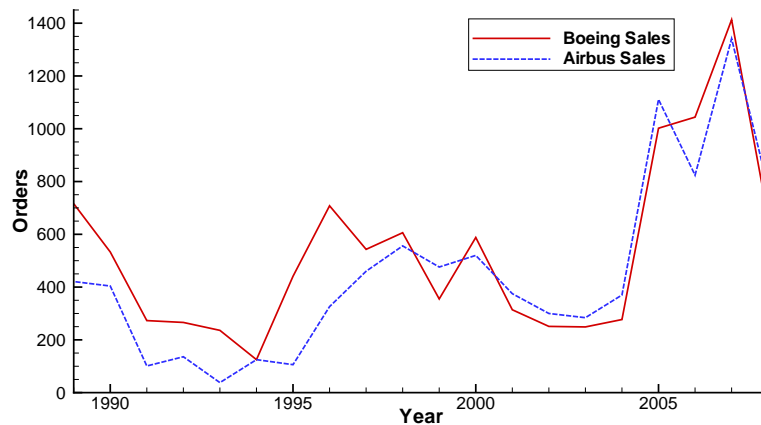


Figure 1.1: Aircraft orders for Boeing and Airbus since 1989.

downloaded some of the costs to consumers, usually in the form of fuel surcharges or reduction of in-flight services. In more drastic measures, many airlines grounded hundreds of their least fuel efficient aircraft during the summer of 2008 when fuel price was at its peak.

Despite the financial difficulties, the global airline industry is expected to grow. Both Airbus SAS and The Boeing Company, the world's two largest aircraft manufacturers, posted record sales for their commercial jets in recent years, with 1413 and 1341 sold respectively in 2007, and 668 and 777 respectively in 2008 [193, 195]. In contrast, Boeing and Airbus sold only 239 and 284 jets in 2003 (Figure 1.1). This growth in civil transport is aided in part by the tremendous expansion of the industry in emerging economic markets in Asia and the Middle East. But the sales figures also suggest that the rising fuel cost is forcing operators to accelerate the renewal of their fleets, and to retire their less efficient aircraft.

As an aircraft usually has a service life of more than 20 years, the impact of decisions on what types of aircraft the operators purchase and operate today will remain well into the 21st century.

had a relatively low proportion of their 2008 fuel supplies hedged, they were especially sensitive to fluctuations in fuel prices.

1.1.2 Environmental Concerns

Continuing expansion of the global aviation industry also means increasing environmental impact from air travel. Environmental effects from aviation are due primarily to greenhouse gas emissions, of which carbon dioxide emission is a main focus, but also includes the formation of condensation trails (“contrails”), other emissions such as ozone, methane, oxides of nitrogen and oxides of sulphur, as well as other forms of pollution such as noise and particulates from unburned fuel.

Carbon dioxide emission is of particular interest because its effects in the atmosphere are measured in decades while effects from other emissions decrease after hours or days. There is strong evidence to show that carbon dioxide in the atmosphere is a main contributor to climate change. In current estimates, GHG emissions from aviation sources account for 2-3% of all emissions from human sources globally, according to the Intergovernmental Panel on Climate Change (IPCC). While the percentage has been steady for the past decade, the actual amount of emission has been rising. In addition, the IPCC estimates that the effects of GHG emissions at high altitudes are three times more significant than at sea level [150].

However, unlike automobiles—a larger source of GHG emissions—where “green” vehicles such as hybrids are widely available, there are currently no immediate alternatives to the air-breathing turbine engine technology used in commercial aircraft. The Royal Commission on Environmental Pollution (RCEP) in the UK [1], as well as other advocacy groups, propose a drastic—although unlikely—modal change: abandoning mass air transportation for short-haul flights in favour of high speed railways that are more environmentally sustainable. However, more likely ways to address environmental concerns will be through the optimal design of the aircraft, i.e. improvements in aerodynamics, structures and engine, as well as optimizing the aircraft’s operation, which may also include streamlining take-off and landing procedures at airports.

1.1.3 Costs in the Design Cycle

Whether an aircraft design is financially successful depends on whether the manufacturer can sell enough aircraft to recover the development cost. In the design of civil transportation aircraft, projected sales of more than 500 are generally needed to justify the development cost. From the moment where a market for a new aircraft is identified to when that new aircraft fills that market gap, there is tremendous pressure to shorten the design cycle. Delays in the design cycle may cost the company sales and profit, and additional development costs may be downloaded to the operators. In terms of the aerodynamic design, traditionally, the design process has relied heavily on experimental techniques such as wind-tunnel or flight tests.

However, these techniques are slow and expensive.

With the improvement in computer technology, wind-tunnel tests during early stages of the design cycle have largely been replaced by computational fluid dynamics (CFD), which allows designers to quickly and accurately solve flows around complex shapes. It is now possible to accurately solve three-dimensional turbulent flows over an entire aircraft in a few hours [118, 94]. From this point of view, CFD is seen as a major revolution to replace expensive experimental techniques to obtain reliable data on aerodynamic performance. However, merely replacing wind-tunnel tests with CFD does not address the question of what the *optimal* design is for a given objective, and the design process still relies on a cut-and-try approach. As aircraft become more refined, it becomes increasingly difficult and time-consuming to make even the slightest improvements. Furthermore, in exploring unconventional aircraft configurations such as the blended-wing-body, designers do not necessarily have the aid of empirical data that can guarantee a competitive design, and the cut-and-try approach is expensive and risky at best. In that sense, aerodynamic shape optimization represents a second revolution in the design process, by automating and speeding up the process of obtaining the optimal shape.

1.1.4 Trends in Aircraft Design

There are three main trends in the shape design of the next generation of cost effective and environmentally friendly aircraft. Similar views on this subject have been expressed by Mc-Masters [119]. All three trends can benefit from an automated shape optimization tool with short turn-around time.

- *Continuing evolution of the current wing-fuselage configuration* is the likely approach in the short term because it does not involve drastic engineering advances. The design process can be guided by the wealth of empirical data gathered from previous designs.
- *Modification of the wing-fuselage configuration* may be explored using novel ideas in terms of wing shapes and placements. Many of the proposed novel approaches are based on empirical theories. This makes testing them on an optimizer an appealing course of action.
- *A shift to unconventional shapes* that represent a more radical departure from traditional configurations may be a long-term solution. For example, the blended-wing-body configuration [191, 103] has been identified as a particularly attractive alternative.

1.2 Aerodynamic Shape Optimization

1.2.1 Early Research

The idea of aerodynamic shape optimization is not a new concept. According to Gunzburger [59], as early as 1687, Newton proposed and claimed to have “solved” the problem of minimizing hydrodynamic resistance on a revolving surface. However, practical application of aerodynamic optimization has not been realized until recently with the development of high-fidelity CFD codes and improved computers. The first practical application of aerodynamic optimization using a gradient-based method is generally credited to Hicks *et al.* [75]. An optimizer using the method of feasible directions, based on the conjugate-gradient method, was used to design airfoil sections at transonic speeds, and the flow solution was obtained using a potential flow solver. This research was extended to 3D wing design by Hicks and Henne [74] using a steepest-descent optimizer. Another early application include that of Vanderplaats [184].

The early applications of optimization were the first to focus directly on improving the performance measures of a geometry. This is in contrast to another well-established design approach, the inverse design method. In the inverse design method, a desirable target pressure distribution is specified, presumably by an experienced aerodynamicist, and the shape of the wing or airfoil is adjusted to recover the target. The method was first introduced by Lighthill [104] in 1945, and this approach was used to design the Liebeck high-lift airfoils [102]. It should be noted that the Liebeck airfoils have been obtained using aerodynamic shape optimization by Driver and Zingg [32], without exploiting any knowledge of Stratford pressure recovery, used in [102]. Other applications of inverse design include Carlson [21], Tranen [181] and Henne [68]. More recently, Fejtek *et al.* [41] extended inverse design to 3D in the design of a complete aircraft. Giles and Drela [48] showed that the cost of an inverse design can be as low as one flow solve. Their method used the 2D Euler equations coupled with a boundary layer correction. Generally, the limitation of the inverse method is with difficulties in specifying a desirable target pressure, especially for problems involving multiple operating conditions and complex flow features such as flow separation and turbulence. In this thesis, an inverse design problem is presented to validate the optimizer.

1.2.2 Newton-Krylov Flow Solver

One major challenge in high-fidelity aerodynamic shape optimization is the repeated computation of the objective function, which requires the flow solution to be solved efficiently. For gradient-based optimizers, a typical optimization cycle requires several hundred flow solves. The number is significantly higher with genetic (evolutionary) algorithms, where hundreds or thousands of flow solutions may be necessary at each generation, depending on the number of

design variables. To this end, the Newton-Krylov (NK) flow solvers are among the most efficient approaches to solving the flow equations. The Krylov method GMRES [163] is found to be especially well-suited for CFD applications. Early work using the NK method can be found in Brown and Saad [16], Wigton *et al.* [192], Johann *et al.* [87], Shakib [172] and Venkatakrishnan and Mavriplis [189]. Other notable examples of NK solvers can be found in [152, 47, 91, 203, 197]. Summaries of different approaches for NK flow solvers can be found in [203, 90].

As the scale and complexity of the CFD and shape optimization problems increase, so too does the demand on computing power. The development of the Beowulf-class parallel computers [177] and standardization of message passing software libraries such as the Message Passing Interface (MPI) [55] promote parallel methods that are now common in CFD codes. For NK based flow solvers, two parallel preconditioners for GMRES based on domain-decomposition are most popular. The first is based on the alternating procedure by Schwarz [170]. Schwarz proved that a computational domain can be divided into overlapping sub-domains, the governing equations solved in each sub-domain sequentially (or in parallel), and a converged solution can be found iteratively. This procedure is used as an effective parallel preconditioner for Krylov methods. The earliest references to an overall “Newton-Krylov-Schwarz” (NKS) framework include those of Cai *et al.* [20], Gropp *et al.* [56] and Keyes [88], which demonstrated the feasibility of the method for parallel CFD applications. Venkatakrishnan [187, 188] also described additive and multiplicative Schwarz methods as parallel preconditioners for GMRES. Other examples of NKS solvers can be found in [6, 10, 174, 171, 200, 57].

Another domain-decomposition strategy that is gaining popularity is based on the Schur complement method. Saad and Sasonkina [164] formulated an approximate-Schur preconditioner for a flexible variant of GMRES (“FGMRES”) [161]. At the moment, there are few examples of using approximate-Schur preconditioners in flow solvers. Hicken and Zingg [73] is one notable example. Preconditioners based on the Schur complement are more common in constrained optimization using sequential quadratic programming (SQP) methods, see for example, Biros and Ghatti [11, 12].

The flow solver used in this thesis is based on work by Nichols and Zingg [139] and Hicken and Zingg [73] for the 3D Euler equations on structured multi-block grids.

1.2.3 Optimizers

The choice of an optimizer has a profound impact on how the solution to the optimization problem is obtained, in terms of the computational effort and speed. Optimizers used for aerodynamic shape optimization problems can generally be grouped into three classes: (a) gradient-free methods, (b) gradient-based optimizers, which include both constrained and unconstrained methods, and (c) one-shot methods.

The general advantage of a gradient-free method is that the flow solver can be treated as a black box that is interchangeable with other analysis codes with little modification, thus avoiding the expensive development cost for gradient calculations (particularly with adjoint methods). Therefore, these methods are more suitable for one-time use of an optimizer, while a gradient-based method is more effective when the optimizer and analysis code are used repeatedly. This class of methods includes classical direct search methods such as the Nelder-Mead simplex method [127]. The Nelder-Mead simplex method is rarely used in aerodynamic shape optimization, for recent examples, see Duvigneau and Visonneau [33] and Désidéri and Janka [31].

Genetic (evolutionary) algorithms [52], which are loosely based on the process of natural selection, are the most popular gradient-free methods for aerodynamic shape optimization. Recent examples of their use can be found in Holst and Pulliam [76], Epstein and Peigin [39], Obayashi [146], Oyama [148] and Sasaki *et al.* [168]. Genetic algorithms have several advantages over other gradient-free and gradient-based methods. The most important is their robustness against local optima. They can also handle discontinuities in the design space, which are problematic for gradient calculations. Also, the algorithm can handle discrete and categorical design variables by using a bitwise representation of the design variables. However, this does not appear to be a major issue with aerodynamic shape optimization problems.

Genetic algorithms also have several disadvantages compared to gradient-based methods. The first is the convergence criterion. With genetic algorithms, the fittest members of the population may not improve for many generations, and it is very difficult to determine whether the optimizer has converged. In addition, genetic algorithms require a population many times the number of design variables. For 3D optimization cases with hundreds of design variables (such as in the design examples in Chapter 8), this could mean thousands of flow solves at each generation. The need for more processors represents a substantial investment. However, genetic algorithms can be easily made parallel by assigning distinct processors to compute the flow solutions. For a comparison between genetic algorithms and gradient-based optimizers, see Zingg *et al.* [201].

One-shot methods are also known as all-at-once methods, fully-coupled methods, and the simultaneous analysis and design approach. The term *one-shot* is believed to have been coined by Ta'asan *et al.* [179]. It is based on the same Lagrangian formulation as the gradient-based methods (Appendix C), but the flow equations and the first-order optimality conditions are coupled, thus avoiding repeated flow and gradient evaluations. However, the linear system from the coupled system is twice the size of the flow equations plus the number of design variables, and scales linearly with the number of objective functions considered. Gunzburger [59] provides an excellent overview of the approach. Frank and Shubin [43] compared the one-shot method

with an SQP optimizer for a one-dimensional nozzle flow problem, and concluded that the one-shot method is considerably more efficient, but less robust than gradient-based methods. In the one-shot method of Gatsis and Zingg [46], the update of the state, design and costate (adjoint) variables are done using the Newton method, with a line-search algorithm. Similar work is also done by Hazra [66, 65] and Hazra and Jameson [67] for 2D and 3D optimization. This method uses a pseudo time-stepping with a preconditioner based on reduced SQP methods. The one-shot method of Iollo *et al.* [79] is based on loosely coupling the flow and adjoint system, each of the flow and adjoint systems are solved a few steps before updating. Other applications can be found in Gumbert *et al.* [58].

Gradient-based methods remain the most popular approach for aerodynamic shape optimization problems, since they can be the most cost effective approach. A brief comparison between genetic algorithms and gradient-based optimizers is shown in Table 1.2. There are two types of gradient-based methods, for *constrained* and *unconstrained* problems, depending on how flow and geometric constraints are handled.

In terms of unconstrained optimization problems, the steepest descent method is most popular, it was used by Hicks and Henne [74], and is used more recently by Nadarajah and Jameson [124], Jameson *et al.* [85], Mavriplis [117], and Jameson *et al.* [86] for the design of low-sweep transonic wings. A disadvantage of the steepest descent method is that convergence is very slow near the optimum. The alternative is to use quasi-Newton methods such as BFGS [17, 42, 53, 173]. Nemeč and Zingg [128, 132, 133, 134] applied it with a Newton-Krylov flow solver in the design of airfoil shapes. Elliott and Peraire [35, 36, 37] also used BFGS in the design of transonic wing-body configurations and business jets. Nielsen and Anderson [140] used another quasi-Newton optimizer DFP for unconstrained optimization. A limited-memory variant of BFGS (L-BFGS) [144] can be used as an alternative for large-scale problems where the number of design variables is high. Unconstrained optimization has also been applied to optimization of unsteady flows, see Nadarajah [125] who uses a steepest descent method, and Rumpfkeil and Zingg [160, 159] using the L-BFGS method. For this thesis, the BFGS optimizer is used.

For constrained optimization problems, optimizers based on sequential quadratic programming (SQP) are most popular. In particular, SNOPT [50] is used most widely in aerodynamic shape optimization problems. It uses a BFGS update of the Hessian of the Lagrangian. Recently the optimizer is used by Hicken and Zingg [72] in 3D optimization. SNOPT is used by Kim *et al.* [89] to optimize the Boeing 747 and business jet configurations, and Carpentieri *et al.* [24] in the design of airfoil shapes. Liu [105] compared the unconstrained BFGS results with SNOPT for 2D airfoil design problems, and found that in certain problems, SNOPT outperforms BFGS. Another optimizer KSOPT [198], which also uses a quasi-Newton update, is used by Anderson and Venkatakrisnan [9]. Cliff *et al.* [28] uses the NPSOL [51] optimizer

	Gradient-based	Genetic Algorithm
Variables	Continuous	Discrete, categorical, continuous
Search	Local	Global
Computational cost	Inexpensive (with adjoint) Expensive (with finite-difference)	Expensive
Development time	Long (with adjoint) Short (with finite-difference)	Short

Table 1.2: Comparison of genetic algorithms and gradient-based optimizers.

in the design of supersonic transport jets.

1.2.4 Gradient Evaluation

Finite differencing is the simplest approach for obtaining the objective function gradient and was popular in early research [75, 74, 184]. Its implementation is straightforward; however, its limitations presented major hurdles preventing the widespread use of gradient-based optimization applications. The first disadvantage is that the cost of a gradient evaluation scales with the number of design variables. The number of design variables in early research was therefore limited. The second disadvantage is that finite-difference schemes are prone to numerical errors. The shortcomings of the finite-difference method are further discussed in Section 5.1.1.

A major breakthrough to address the first limitation was the development of the adjoint method, which was pioneered by Pironneau [151] for optimization of fluid flows, and by Jameson [80] for aerodynamic shape optimization. The main advantage of the adjoint method is that the cost of a gradient calculation can be made nearly independent of the number of design variables. The method is further divided into the *continuous* and *discrete* approaches. In the continuous approach, the governing equations are first differentiated to formulate the adjoint equation, and then discretized. In contrast, in the discrete approach, the governing equation is first discretized, and then differentiated to formulate the adjoint equation. The advantage of the latter is that the adjoint solver can be easily added to an existing flow solver. Also, the gradient computed from the discrete approach is *exact* since the gradient computed is based on the exact linearization of the discrete equations. However, both approaches have been implemented successfully in aerodynamic design optimization. The first applications by Jameson and Pironneau use the continuous approach. Other notable examples of the use of this approach include those of Jameson *et al.* [83] and Reuther *et al.* [156, 157] using multi-block structured grids, and Anderson and Venkatakrisnan [9] using unstructured grids. Other applications using the continuous approach can be found in [85, 96]. Applications using the discrete adjoint

approach can be found in [140, 126, 125, 132, 135, 129, 116, 117, 24]. Extensive discussions on the two approaches can be found in [59, 124, 49]. The adjoint method is also used by Nemec and Aftosmis [130] and Lee-Rausch *et al.* [95] to estimate the error of the flow solution.

The direct method, or the *flow sensitivity method*, follows a similar derivation as the adjoint method. This method is used in airfoil shape optimization by Wong [196]. The derivation and discussion of the direct method and the adjoint method are presented in Section 5.1.2. The efficiency of the direct method compared to the adjoint method depends on the number of design variables and objective functions: the direct method is more efficient if the number of objective functions is higher than the number of design variables, while the adjoint method is more efficient when the number of design variables is higher.

Gradient computation using the complex-step method is another notable alternative to address the second limitation of the finite-difference method, because it effectively eliminates subtractive cancellation errors, and the complex step-size can be arbitrarily small [113]. This method is especially popular in multi-disciplinary optimization (MDO) research. The pros and cons of the complex-step gradient are discussed in Section 5.1.3. The method was successfully applied by Sturdza [178] in the design optimization of supersonic aircraft. Newman *et al.* [137, 138] also used the method for the aero-structural design of a wing. Anderson *et al.* [8] analyzed its use for unstructured flow solvers, and found that the complex-step gradient roughly doubles the flow solver time and memory requirements, but is able to produce much more accurate gradients at small step-sizes. The complex-step method is often used as a validation tool to test gradient accuracy; see for example, Martins [110, 111], who used a coupled adjoint method for aero-structural optimization. The complex-step method is not used in this thesis, since it requires a complex version of GMRES, which is not readily available. However, this thesis uses an approach similar to that of Nielsen and Kleb [142], who incorporated a complex-step method in the formulation of the flow Jacobian matrix for discrete adjoint gradient calculations. A more general study on the use of complex-step method can be found in the works by Martins *et al.* [113, 112].

Automatic differentiation (AD), also known as algorithmic differentiation or computational differentiation, is based on a systematic application of the chain rule of differentiation to each operation in a computer program. Because this method is analytical, the AD derivatives are exact. However, the resulting source code may be difficult to maintain. Popular AD implementations that have been used for aerodynamic optimization problems include ADIFOR [13], and more recently, TAPENADE [63]. Bischof *et al.* [14] used ADIFOR to generate differentiable volume grids for MDO applications. Anderson *et al.* [8] finds that the cost of using ADIFOR to evaluate derivatives does not significantly increase with increasing the number of design variables. TAPENADE is used by Hascoët *et al.* [64] in minimization of the sonic boom,

Vázquez *et al.* [186] in the design of flexible wings, and Martinelli and Beux *et al.* [109] in geometry parameterization. Mader *et al.* [107] use AD to form the Jacobian matrix for computing the discrete adjoint gradient as an alternative to the complex step. Additional information about AD methods can be found in Griewank [54].

1.2.5 Geometry Parameterization

Many different approaches to geometry parameterization have been implemented for aerodynamic shape optimization. Selecting a suitable parameterization strategy requires compromising between defining a geometry with as few parameters² as possible, which helps convergence of the optimizer, and maintaining the greatest amount of flexibility to capture the true optimal shape, which requires more parameters, but may slow down the optimizer. Surveys and comparisons of different geometry parametrization methods can be found in [122, 25, 199, 167, 166].

Maintaining the greatest flexibility requires defining the geometry with a very high number of design variables. An obvious choice is using the coordinates of the surface grid nodes as design variables. This approach is used successfully by Jameson [80] and Mohammadi [120] with a smoothing operator, and more recently in [85, 96]. However, when used in large-scale 3D problems with turbulent flows, the number of design variables may be prohibitively large, affecting the convergence of the optimizer. The pros and cons of this approach are further discussed in Appendix B.2.

Other parameterization strategies aim at defining the geometry with fewer design variables. One notable method is the Hicks-Henne bump function, which belongs to a class of parameterization strategies using *analytical shape functions*. It was introduced in [74] at a time when the number of design variables had a profound impact on the cost of gradient evaluation. In this method, smooth “bumps” (shape functions) are added to a baseline airfoil to change its geometry. The method is used successfully in a wide range of optimization applications. Examples of this approach can be found in Nadarajah and Jameson [124] and Reuther *et al.* [157]. Appendix B.1 presents further discussion on this method, as well as a simple way to generalize it for 3D optimization. The PARSEC method [175] uses engineering airfoil design parameters such as airfoil thickness, leading-edge curvature and trailing-edge angle, etc. to define the airfoil. Other geometry parameterization strategies include the class/shape function transformation technique (the CST method) by Kulfan [92], the control grid method by Anderson *et al.* [7], and the embedded Bézier shape parameterization of Désidéri and Janka [31].

Another popular approach is the *spline approach*, which is used in this thesis. The current work uses basis splines (B-splines). Examples of B-splines can be found in [133, 134, 45]. B-

²Since *design variables* are parameters that evolve during the optimization cycle, the two terms can be used interchangeably.

splines are found to be better suited than Bezier curves in allowing more local shape control and can be easily integrated into a computed-aided design (CAD) environment. Another similar approach uses non-uniform rational B-splines (NURBS) [165, 97], which further allow closed-loop curves to be defined. In the B-spline formulation of Fudge and Zingg [45], the B-spline controls can be grouped together to add another layer of planform variables.

A new trend for geometric parameterization is using a CAD-based approach, which allows the optimizer direct access to the CAD parameters that define a shape. One promising approach is the Computational Analysis and Programming Interface (CAPRI) by Haines [61, 60]. It is used in aerodynamic shape optimization studies such as Jameson *et al.* [85] and Nemec *et al.* [131], and in MDO studies such as Deremaux *et al.* [30], and Choi *et al.* [27].

1.2.6 Multi-Point Optimization

Many examples of wing design optimization consider only a single operating condition, such as a fixed Mach number or a fixed lift coefficient (“single-point optimization”). This serves to prove the effectiveness of an algorithm but is not a practical approach to design. A wing must operate in a range of conditions in the flight envelope, for example, a range of Mach number and lift coefficients. Trade-offs and compromises must be made between different operating conditions to achieve an overall optimum. Nemec and Zingg [135] studied multi-point optimization of airfoils using a discrete-adjoint method. Zingg and Elias [204] investigated automatic selection of sampling points and weights. The work is extended in Zingg and Billing [202] to include multiple cruise and dive conditions. Studies in multi-point wing design include those by Reuther *et al.* [156, 157], Cliff *et al.* [28], Leoviriyakit and Jameson [96]. Jameson *et al.* [82] studied multi-point aero-structural optimization of wings.

1.3 Thesis Objectives

The main objectives of this thesis are:

1. Extend the Newton-Krylov approach pioneered by Nemec and Zingg [128, 133, 132, 134] (for aerodynamic shape optimization of airfoils in two-dimensions) to wing optimization in three-dimensions. A comparison of the NK approach with other popular approaches is shown in Table 1.3. The Newton-Krylov approach is used in the wing design using structured multi-block grids. Note that structured grids are intended for relatively “clean” geometries. Structured grid flow solvers are faster than unstructured flow solvers, which are better suited for more complicated geometries.

In this approach, the inexact Newton method is used in the flow solver while the quasi-

Newton method is used as the optimizer. A preconditioned Krylov subspace method is used to solve the linear systems in both the flow solver and the adjoint equations. This approach had been found to be very robust in a wide range of 2D problems. The Newton-Krylov approach will form a framework for future optimization with viscous flows, and eventually as a component for large-scale MDO applications. To achieve this objective, original contributions are considered in the development and integration of key components of the optimizer, including:

- a 3D geometry parameterization technique which will provide the optimizer with geometric design variables
 - an efficient 3D grid perturbation algorithm
 - adjoint gradient computation
 - an appropriate implementation of the optimizer, and
 - implementation of the appropriate geometric constraints
2. Address the issues and challenges in the implementation of the Newton-Krylov approach, including:
- the accuracy of the discrete adjoint gradient
 - the efficiency of the optimizer, and
 - the effect the starting geometry may have on optimizer convergence and the final optimal shape
3. Characterize the performance, efficiency and effectiveness of the new 3D Newton-Krylov algorithm through application to the design of wings in transonic flows. Design examples will be shown for single- and multi-point optimization cases using lift-constrained drag minimization.

1.4 Thesis Outline

The thesis is divided into nine chapters. Following this introduction, Chapter 2 presents the framework of the optimization problem. The governing equations and flow analysis method are presented in Chapter 4, and the optimization algorithm is presented in Chapter 5. An algebraic grid movement algorithm is presented in Chapter 6. The numerical algorithm is validated through a series of representative cases in Chapter 7. Results from design examples are presented in Chapter 8. Based on these results, concluding remarks and recommendations

	Flow Solver	Gradient	Optimizer	Geom. Param
Jameson <i>et al.</i> [82]	RANS Structured	Continuous Adjoint	Steepest descent	surface nodes
Mavriplis [117]	RANS Unstructured Multigrid	Discrete Adjoint Multigrid	Steepest descent	surface nodes
Nielsen & Anderson [140]	RANS Unstructured Implicit Euler	Discrete Adjoint Preconditioned GMRES	DFP	free-form deformation
Newton-Krylov (Current)	Euler Structured MB Newton-Krylov	Discrete adjoint Preconditioned FGMRES	BFGS	B-spline control surfaces

Table 1.3: Other gradient-based approaches for 3D aerodynamic shape optimization.

for further research are presented in Chapter 9. Additional information on the development of this thesis as well as additional results can be found in [101, 100, 99, 98, 205].

Chapter 2

Optimization Problem

2.1 Problem Formulation

The goal of aerodynamic shape optimization is to find a shape parameterized by a set of design variables \mathcal{X} such that a scalar objective function \mathcal{J} (also known as the *cost function*) is minimized:

$$\min_{\mathcal{X}} \mathcal{J}(\mathcal{X}, \mathbf{Q}) \quad (2.1)$$

Here, \mathbf{Q} is the vector of state variables representing the flow solution at each node of the computational grid. To ensure the optimizer yields a physically realistic shape, a set of equality and/or inequality constraints may be imposed. For this research, only geometric constraints that are functions of design variables are considered:

$$C_j(\mathcal{X}) \leq 0 \quad (2.2)$$

In addition, the discrete steady-state flow equations must also be satisfied:

$$\mathbf{R}(\mathcal{X}, \mathbf{Q}) = 0 \quad (2.3)$$

which implicitly defines $\mathbf{Q} = \mathbf{Q}(\mathcal{X})$. The flow constraint in this case may be specified by the discrete Euler equations for inviscid flows, or by the Reynolds-averaged Navier-Stokes (RANS) equations for viscous and turbulent flows. The Euler equations, discussed in Chapter 4, are used in this thesis. In practice, the flow solution is first computed for a given \mathcal{X} , and the objective function is computed based on the state variables and design variables, as shown in Figure 2.1. When the geometric constraints are applied as a penalty term in the objective function, then the overall optimization problem can be regarded as an unconstrained problem; the optimizer treats the objective function as a function of design variables alone, i.e. $\mathcal{J} = \mathcal{J}(\mathcal{X})$.

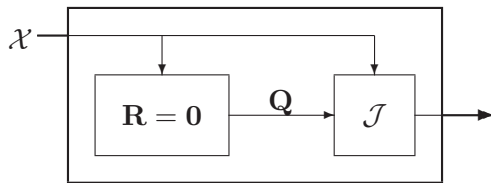


Figure 2.1: Solving for objective function

2.2 Objective Functions

Determining which shape is optimal depends on how optimality is defined. For example, a wing that is optimized for maximum range at transonic speeds is unlikely to be optimized for endurance flights at subsonic speeds. Therefore the proper selection of the objective function is crucial. For aerodynamic shape optimization problems, the objective functions are generally based on performance measures, such as lift and drag coefficients (C_L , C_D). In particular, lift-constrained drag minimization is an important problem to consider. For inverse design problems, the objective function is based on a prescribed surface pressure distribution. Other objective functions not considered for this research may include endurance factor [77] or sonic boom reduction [126].

2.2.1 Lift-Constrained Drag Minimization

In the design of long-range cruise configurations, one of the most important equations in aircraft design is the Breguet range equation [15, 5], which relates aerodynamic, structural and propulsion factors that influence the range R of an aircraft. This equation is obtained if the angle of attack and speed are assumed to be constant during flight.¹ For transonic aircraft with turbofan or turbojet engines, the equation can be expressed in this form:

$$R = \frac{V}{\text{TSFC}} \frac{L}{D} \ln \left[1 + \frac{W_F}{W_P + W_S + W_R} \right] \quad (2.4)$$

where V is the speed of the aircraft, TSFC is the engine's specific fuel consumption at the flight altitude, and W_F , W_S , W_P and W_R are weights of fuel, structure, payload and reserve fuel, respectively. When these factors are constant, the range of an aircraft can be optimized by maximizing aerodynamic efficiency (i.e. lift-to-drag ratio L/D). To highlight the importance of aerodynamic efficiency, consider an aircraft with a similar configuration as a Boeing 737-900ER.² According to (2.4), for a fixed range, a 1% drag penalty in drag will result in about

¹Such conditions will result in a *cruise-climb* flight path. Note that the range equation for a different type of flight path would be different, such as the case with altitude and angle of attack held constant. Derivations of those equations can be found in Appendix A.

²The empty weight of a 737-900ER is about 42500kg, the maximum take-of weight is 85000kg. The aircraft has a fuel capacity of 24300L.

4% reduction in payload weight.

The goal of the lift-constrained drag minimization problem is to improve the aerodynamic efficiency while maintaining a required lift. For level flight, it is required that lift equals to weight, i.e. $L = W$. For this type of problem, the objective function is of the form:

$$\mathcal{J} = \omega_L \left(1 - \frac{\mathcal{L}}{\mathcal{L}^*}\right)^2 + \omega_D \left(1 - \frac{\mathcal{D}}{\mathcal{D}^*}\right)^2 \quad (2.5)$$

where

$$\mathcal{L} = C_L \cdot S$$

$$\mathcal{D} = C_D \cdot S$$

are the lift and drag coefficients multiplied by the wing's reference area S , which is the planform (projected) area.³ The objective function is formulated this way such that if S changes during the optimization, the target lift can still be maintained by increasing C_L . Targets in lift and drag are specified by the user in \mathcal{L}^* and \mathcal{D}^* , and weights ω_L , ω_D specify the relative importance of maintaining lift vs. drag reduction. If the target lift is attainable and target drag is not, this objective function represents lift-constrained drag minimization with the lift constraint appearing as a penalty term. The advantage of an objective function in the form of (2.5) is that additional terms (which may be considered as constraints) can be easily added.

For inviscid flows, the minimum induced drag predicted by Prandtl's lifting-line theory [4] provides a guideline to select the target drag $\mathcal{D}^* = C_D^* \cdot S$:

$$C_{D,\text{induced}}^* = \frac{(C_L^*)^2}{\pi AR} \quad (2.6)$$

where AR is the wing's aspect ratio. Note that \mathcal{D}^* must be selected with care because it affects the design space. A target that is set too low will affect the convergence of the optimizer, but if the target is set too high, the optimizer will converge without finding the shape with the minimum drag. For the design examples in this thesis, the target is set to a physically unattainable value just below the minimum predicted induced drag.

In optimization problems where a constraint on lift is automatically satisfied, the objective function can simply be the drag coefficient:

$$\mathcal{J} = C_D \quad (2.7)$$

For example, in Zingg and Billings [202], the angle-of-attack of an airfoil is considered as a state variable that is adjusted during flow solve to satisfy an input C_L . This objective function is also valid if a constrained optimization method such as SNOPT [50] is used; the target lift will appear explicitly as an equality constraint.

³This is equivalent to the lift and drag force scaled by the dynamic pressure, $\frac{1}{2}\rho V^2$.

The objective function in (2.5) is contrasted with:

$$\mathcal{J} = \frac{C_D}{C_L} \quad (2.8)$$

Both (2.8) and (2.5) maximize the lift-to-drag ratio, but differ in that in (2.8), no constraint is placed on the maintaining lift. Note that for inviscid optimization problems of a wing alone, directly maximizing C_L/C_D is problematic. Without viscous effects, drag coefficient always scales with $(C_L)^2$, i.e. $C_D \rightarrow 0$ as $C_L \rightarrow 0$, and (2.8) can be minimized simply by changing the wing's angle of attack such that no lift is generated. However, (2.8) is useful in viscous flow problems. In this case, total drag does not vanish as $C_L \rightarrow 0$. This objective function is used with a viscous flow solver in the design of transonic airfoils [205], and also in the design of high-lift configurations [133].

2.2.2 Inverse Design

The inverse design problem is also considered. The objective function for this problem is:

$$\mathcal{J} = \frac{1}{2} \sum_1^{N_p} \left[(C_p)_i - (C_p^*)_i \right]^2 \quad (2.9)$$

In this case, the user specifies a known target pressure distribution $(C_p^*)_i$ on each body surface node i . By minimizing \mathcal{J} , the optimizer recovers the shape that minimizes the least-square error. Inverse design is strictly not a true optimization problem, since minimizing \mathcal{J} does not guarantee that the final geometry shape is in any way optimal. In addition, it requires the user to have previous knowledge of an ideal pressure distributions. However, inverse design is valuable as a validation tool for the optimizer.

2.3 The Newton-Krylov Approach

The optimization problem in Section 2.1 is solved using a Newton-Krylov approach applied to a gradient-based optimizer. This approach was pioneered by Nemeć and Zingg [132]. In general, gradient-based optimizers for unconstrained problems follow the procedures shown in Algorithm 2.1. In our current context, the Newton-Krylov approach refers to using the Newton method and/or a Krylov subspace method to obtain the flow solution, objective function gradient and the search direction. The steady-state flow solution is first computed using the Newton method with pseudo-transient continuation. An approximate-Newton start-up stage is used. A Krylov method is used to solve the linear system at every Newton iteration (Chapter 4). An objective function is then computed based on the flow solution, and penalty terms are added to the objective function if geometric constraints are violated. A gradient direction in

the design space is then computed (Section 5.1.2) using the adjoint method, with the adjoint system solved by the same Krylov method used in the flow solver. Based on the gradient, a quasi-Newton search direction is computed (Section 5.2). Finally, a line search algorithm is applied to compute a proper step length along the search direction. It should be noted that in the current implementation, once the proper step length is found, the objective function and gradient have already been computed for the updated design variables. A new grid is generated using a grid movement algorithm (Chapter 6) after each iteration. Grid movement is also required during gradient computation.

Algorithm 2.1: Gradient-based optimization

Data: \mathcal{X}_0
Result: \mathcal{X}, \mathcal{J}

- 1 Initial shape \mathcal{X}_0
- 2 **for** $n = 0, 1, \dots$ **do**
- 3 Generate grid \mathbf{G} based on \mathcal{X} using a grid movement algorithm
- 4 Solve for \mathbf{Q}_n where $\mathbf{R}(\mathcal{X}_n, \mathbf{Q}_n) = 0$ and compute \mathcal{J}
- 5 Compute objective function gradient \mathcal{G}
- 6 Compute search direction \mathcal{P}_n
- 7 Compute step-length β_n
- 8 Update design variables $\mathcal{X}_{n+1} = \mathcal{X}_n + \beta_n \mathcal{P}_n$
- 9 **end**
- 10 Continue to iterate until \mathcal{G} converged

Chapter 3

Geometry Parameterization and Design Variables

Design variables are design parameters that are allowed to evolve during the optimization cycle. These variables may be continuous (e.g. “change of sweep angle at the leading edge”), discrete (e.g. “the number of engines”) or categorical (e.g. “biplane vs. boxed wing”). For aerodynamic shape optimization, design variables are generally continuous, while discrete and categorical design variables are more common in multi-disciplinary optimization (MDO) applications. When using a gradient-based optimizer such as the current Newton-Krylov algorithm, it is further required that both the design variables and the design space be smooth and continuous. Design variables are usually part of a geometry parameterization strategy, but they may also include operating parameters such as angle of attack (α) or Mach number (M).

A good geometry parameterization strategy is crucial to the efficiency and effectiveness of the optimizer. A method that uses many design variables (e.g. locations of surface nodes) has more flexibility over allowable shapes, and is therefore more likely to capture the true aerodynamic optimum, but the high number of design variables may cause the optimizer to converge slowly. On the other hand, a method that uses only a few variables (e.g. planform variables) may converge in much fewer iterations, but not necessarily to the true optimum. In general, a good strategy is one that makes a good compromise between speed and flexibility.

A geometry parameterization strategy using basis spline (B-spline) curves and surfaces is considered based on Fudge *et al.* [45]. One advantage of using B-spline curves is their local nature. Each control point has a unique associated basis function that is non-zero in only part of the curve. The B-spline curve also has the added ability to change its order k while holding the number of control points N fixed (assuming that $N > k$). Similar B-spline formulations using NURBS (Non-uniform rational B-spline) have been used for aerodynamic shape optimization by Samareh [165], Nielsen *et al.* [140] and Lepine *et al.* [97]. Two popular alternatives to the

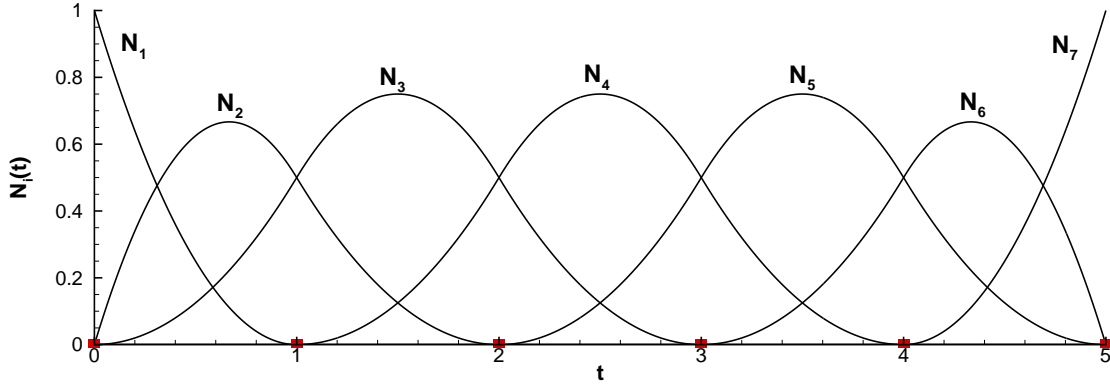


Figure 3.1: Example of third order B-spline basis functions

spline approach, the Hicks-Henne bump function, and the discrete approach, are discussed in Appendix B.

3.1 B-Spline Curve and Surface Formulation

The k -th order B-spline parametric representation of a curve using N control points and basis functions is given by:

$$\vec{a}(t) = \sum_{i=1}^N (\mathcal{X}_B)_i \mathcal{N}_{i,k}(t) \quad (3.1)$$

where \vec{a} is the position vector along the curve at a distance t from the origin, \mathcal{X}_B are the locations of the B-spline control points, and $\mathcal{N}_{i,k}(t)$ are the basis functions of order k , defined by the recursive Cox-deBoor relationships [158]:

$$\mathcal{N}_{i,1}(t) = \begin{cases} 1 & \text{if } d_i \leq t < d_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

and

$$\mathcal{N}_{i,k}(t) = \left[\frac{t - d_i}{d_{i+k-1} - d_i} \right] \mathcal{N}_{i,k-1}(t) + \left[\frac{d_{i+k} - t}{d_{i+k} - d_{i+1}} \right] \mathcal{N}_{i+1,k-1}(t) \quad (3.3)$$

where d_i represents the elements of a uniform open knot vector given by:

$$d_i = \begin{cases} 0 & 1 \leq i \leq k \\ i - k & k + 1 \leq i \leq N \\ N - k + 1 & N + 1 \leq i \leq N + k \end{cases} \quad (3.4)$$

Using the definitions in (3.2)–(3.4), Figure 3.1 shows the basis functions generated for a third-order ($k = 3$) B-spline curve with seven control points ($N = 7$). Locations of the knot elements are shown in red. A B-spline curve using those basis functions is shown in Figure 3.2.

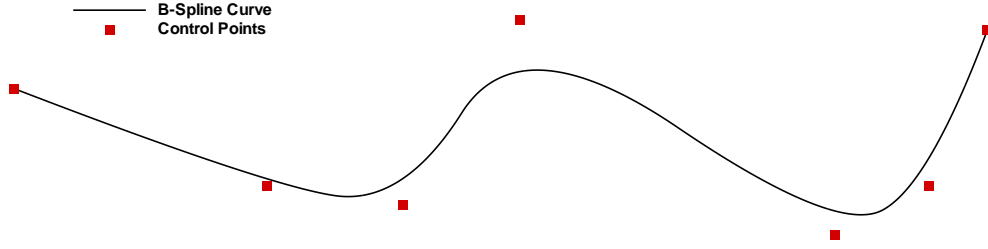


Figure 3.2: Example of a third-order B-spline curve.

The B-spline curve formulation (3.1) can be extended to describe a surface in 3D space:

$$\vec{a}(s, t) = \sum_{i=1}^M \sum_{j=1}^N (\mathcal{X}_B)_{i,j} \mathcal{M}_{i,l}(s) \mathcal{N}_{j,k}(t) \quad (3.5)$$

where $\vec{a}(s, t)$ is now the position vector of the surface at a parametric distance of (s, t) from the origin of the surface. The basis functions \mathcal{N} (k -th order) and \mathcal{M} (l -th order) as well as the knot vectors are calculated in the same way as specified in (3.2)–(3.4). In this thesis, the orders of the basis functions are the same (i.e. $k = l$), and the second subscript is dropped for simplicity. Furthermore, the design examples presented in Chapter 8 all use cubic ($k = l = 3$) basis functions. This B-spline surface formulation can be further extended to parameterize an entire computational grid, see for example Hicken and Zingg [71]. To better visualize how the surface is generated, (3.5) can be written as:

$$\vec{a}(s, t) = \sum_{i=1}^M \mathcal{D}_i(t) \mathcal{M}_i(s) \quad (3.6)$$

where

$$\mathcal{D}_i(t) = \sum_{j=1}^N (\mathcal{X}_B)_{i,j} \mathcal{N}_j(t) \quad (3.7)$$

3.2 Surface Grid Approximation and Generation

Before the start of the optimization cycle, the surface grid is extracted from an existing volume grid and then parameterized using B-spline control surfaces. Consider a structured surface grid with I and J nodes (indices $i = 1 \dots I$, $j = 1 \dots J$) in the parametric directions s and t . The

B-spline representation (3.6)–(3.7) can be described in discrete matrix form as:

$$\begin{aligned}\mathbf{G}_S &= \mathcal{U}\mathcal{D} \\ \mathcal{D} &= \mathcal{X}_B\mathcal{V}\end{aligned}\tag{3.8}$$

where

$$\mathbf{G}_S = \begin{bmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & & \vdots \\ x_{I1} & \cdots & x_{IJ} \end{bmatrix}\tag{3.9}$$

is the surface grid containing the x , y or z coordinates for each surface grid node j , k , while \mathcal{U} and \mathcal{V} store the basis function values at parametric distances s and t from the grid origin:

$$\mathcal{U} = \begin{bmatrix} \mathcal{N}_1(s_1) & \cdots & \mathcal{N}_M(s_1) \\ \vdots & & \vdots \\ \mathcal{N}_1(s_I) & \cdots & \mathcal{N}_M(s_I) \end{bmatrix}, \quad \mathcal{V} = \begin{bmatrix} \mathcal{M}_1(t_1) & \cdots & \mathcal{M}_1(t_J) \\ \vdots & & \vdots \\ \mathcal{M}_N(t_1) & \cdots & \mathcal{M}_N(t_J) \end{bmatrix}\tag{3.10}$$

The distances s and t are calculated based on the nodal indices:

$$s_i = \frac{i-1}{I-1}(m-k+2)\tag{3.11}$$

$$t_j = \frac{j-1}{J-1}(n-k+2)\tag{3.12}$$

The matrix \mathcal{D} is an intermediate matrix of size $M \times J$, and \mathcal{X}_B is a matrix containing the x , y or z coordinates of the B-spline control points:

$$\mathcal{X}_B = \begin{bmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & & \vdots \\ x_{M1} & \cdots & x_{MN} \end{bmatrix}\tag{3.13}$$

The control point locations are found by first solving for \mathcal{D} , and then \mathcal{X}_B in the least-squares problems in (3.8).¹ This process is repeated for each of the three coordinates. Figures 3.3a and 3.3b show the parameterization of an ONERA M6 wing.

To generate a new surface grid, the intermediate matrix \mathcal{D} in (3.8) is first generated based on the new control point locations (\mathcal{X}_B in (3.13)), and then the new surface grid \mathbf{G}_S is generated. Following the previous example, Figures 3.3c and 3.3d show the generation of a surface grid after one B-spline control point has been moved in the z -direction. In this example, the surface grid is perturbed only near the perturbed control point. Based on the new surface grid, a new volume grid can be generated using a grid movement algorithm (see Chapter 6).

¹The least-squares problem is solved using the QR factorization routine in the LAPACK (Linear Algebra PACKage) library, which is a set of FORTRAN 77 routines for solving systems of linear equations.

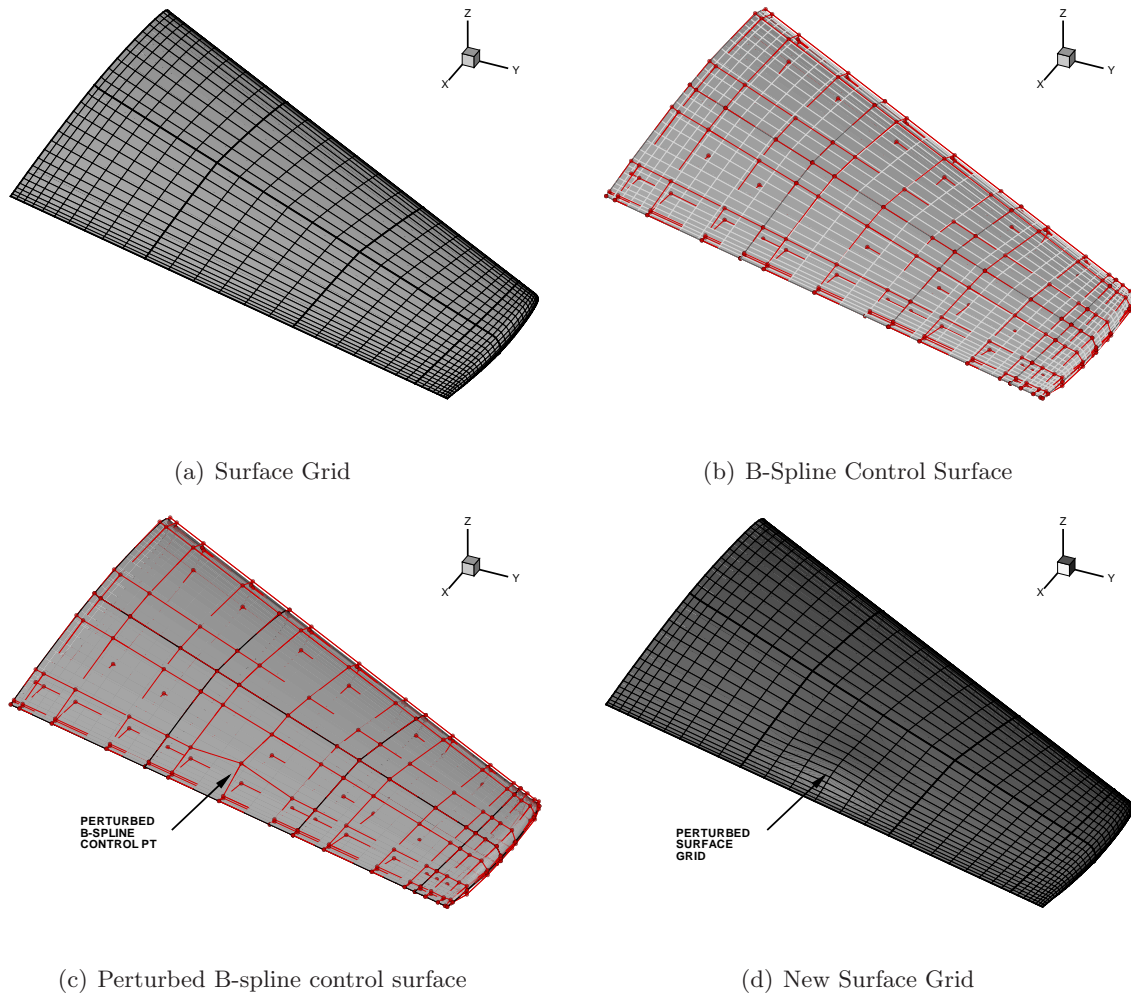


Figure 3.3: B-spline surface parameterization of an ONERA M6 wing.

3.3 Design Variables

All three degrees of freedom (translation in x -, y - and z -directions) of any B-spline control point may be considered valid design variables. While using all three degrees of freedom of every control point gives the optimizer the greatest flexibility to find the aerodynamic optimum, it may increase the number of design variables. Instead, a two-level approach is used. At the planform level, B-spline control points are grouped into a reduced set of planform variables based on general engineering practices. This provides the designer with a more intuitive set of parameters. At the wing section level, the z -coordinate of each individual control point may move independently to adjust the wing section shape.

3.3.1 Planform Design Variables

The B-spline control points can be grouped into a reduced set of planform variables. General planform variables used to describe wings are considered: sweep (Λ), dihedral (Γ), twist (Ω), semi-span ($b/2$) and chord (c). Other planform parameters such as taper ratio (λ) and aspect ratio (AR) are extracted from the above planform variables.

The leading-edge (LE) and trailing-edge (TE) sweep angles are defined for each B-spline section j :

$$\Lambda_j = \tan^{-1} \left[\frac{y_j - y_{j-1}}{x_j - x_{j-1}} \right] \quad (3.14)$$

This is shown in Figure 3.4a. The design variable in this case is the change in the sweep angle from the original geometry ($\Delta\Lambda_j$). This parameterization strategy allows both LE and TE sweep angles to change independently at each section. However, in practice, the LE sweep angle is generally constant from wing root to tip, while the TE sweep angle may change at one or two spanwise locations. Examples of modifying an ONERA M6 wing are shown in Figure 3.4. In Figure 3.4b, an additional 10 degrees is added to the LE sweep at the wing root, and the wing is sheared along the chord-wise direction, i.e. TE sweep also changes. The wing's surface area and enclosed volume are therefore preserved. In Figure 3.4c, LE and TE sweep angles are treated separately. In this case, the TE sweep is reduced at the wing root, and also after the 5th B-spline section. The wing's volume is not preserved in this example. This example also shows that the wing's taper ratio can be effectively controlled by independently changing the LE and TE sweep angles. The optimizer limits the range of allowable sweep angles to between $\pm 60^\circ$ to maintain grid integrity.² For larger changes in geometry, regriding may be required. However, in the design examples presented in this thesis, sweep angles at each optimization iteration are generally well within this range.

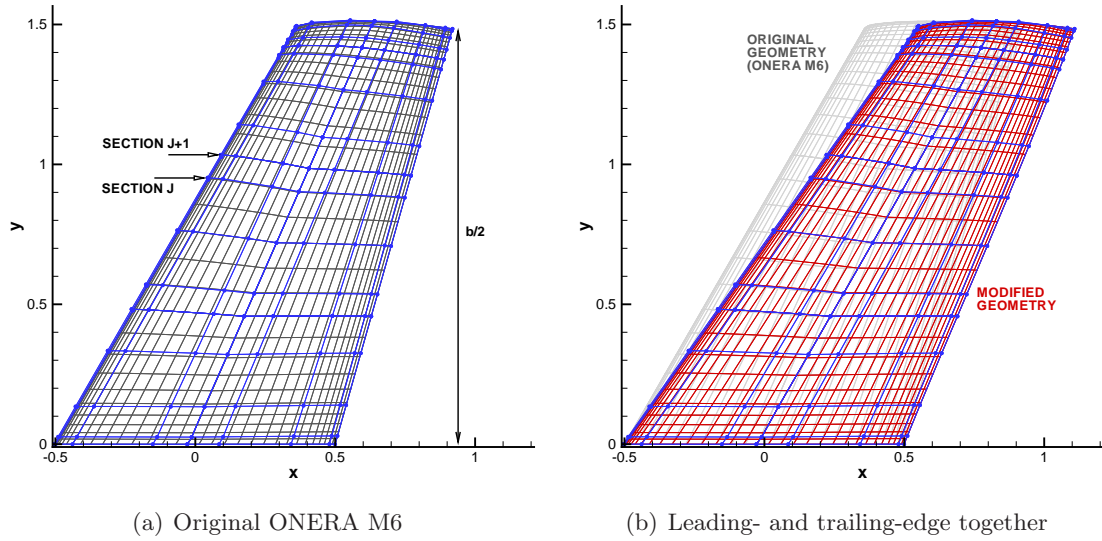
Similarly, the dihedral angle at section j is calculated relative to the $j - 1$ section:

$$\Gamma_j = -\tan^{-1} \left[\frac{z_j - z_{j-1}}{x_j - x_{j-1}} \right] \quad (3.15)$$

Like sweep angles, the change in dihedral angle at each section j can be considered as a design variable. An example of changing the dihedral angle is shown in Figure 3.5. The angle is modified at the wing root, and at the 11th and 16th sections with the values: $\Delta\Gamma_1 = 5.0^\circ$, $\Delta\Gamma_{11} = 20.0^\circ$, and $\Delta\Gamma_{16} = 20.0^\circ$. The changes at these sections are propagated towards the wing tip. As the dihedral angle changes, the B-spline control points are translated in the z -direction, and the wing's semi-span and planform area are maintained. In practice, this method

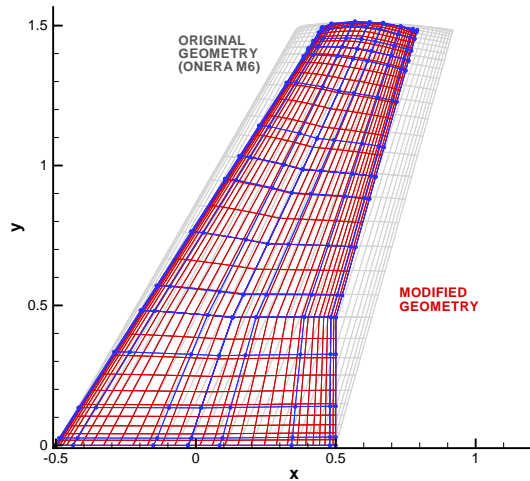
²This is done by applying a hyperbolic tangent function to the input $\Delta\Lambda$ supplied by the optimizer:

$$\Delta\Lambda_{\text{actual}} = \left(\frac{\pi}{3} - |\Lambda_{\text{init}}| \right) \times \tanh(\Delta\Lambda_{\text{input}})$$



(a) Original ONERA M6

(b) Leading- and trailing-edge together



(c) Leading- and trailing-edge treated independently

Figure 3.4: Examples of changing sweep angle

should not be used for large $\Delta\Gamma$, such as in the design of a winglet, as the integrity of the grid may be compromised.

Another design variable is the change in the wing's geometric twist (Ω). It is defined as change in the wing section's angle of attack from root to wing tip.³ Changing the twist is done by translating the z -coordinates of the B-spline control points, based on their distance from the leading edge. The twist angle varies linearly from wing root to wing tip. The modified wing surface has the same planform as the original. An example is shown in Figure 3.6. In the example, a 20° twist is added to an ONERA M6 wing.

³This is contrasted with the wing's *aerodynamic* twist, which is defined as the angle between the zero-lift angle of any spanwise section and the zero-lift angle of the root section. Aerodynamic twist is created through modifying the wing's cross-section.

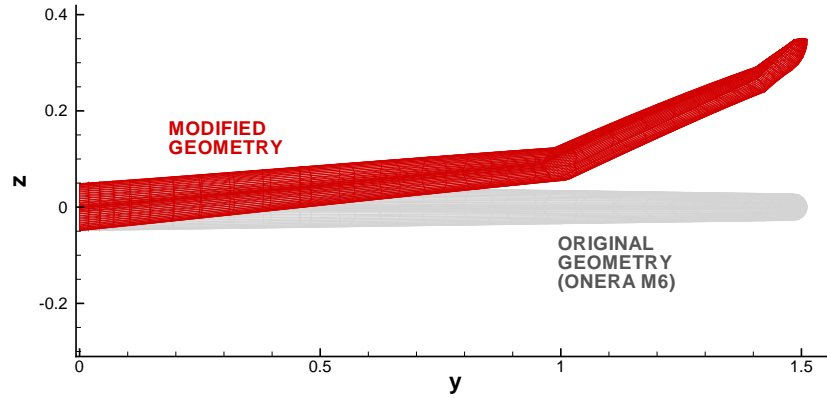


Figure 3.5: Positive dihedral angles at wing root and mid-span.

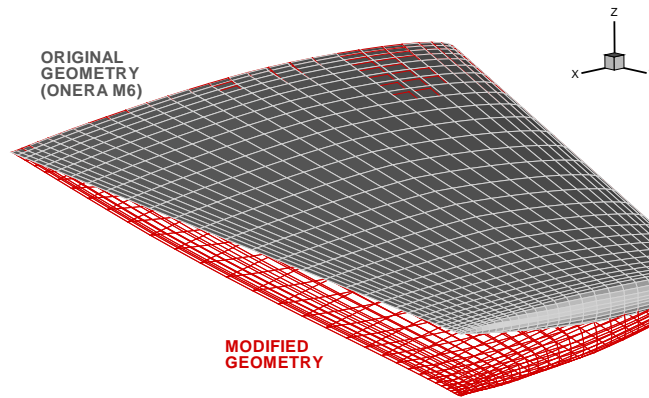


Figure 3.6: Wing twist example.

Finally, the change in the wing's semi-span ($\Delta b/2$) can also be used as a design variable. In this case, the wing's original semi-span is defined in Figure 3.4a.

3.4 Geometric Constraints

There are several main purposes to apply geometric constraints: (1) to ensure a physically feasible design, such as preventing wing surfaces from crossing over; (2) to exclude shapes that are difficult to manufacture; and (3) to enforce other engineering limitations that are related to structural and fuel storage requirements. Only constraints that are functions of the design variables are considered. For this research, volume and thickness constraints are implemented. When the constraints are violated, a quadratic penalty term is added to the objective function:

$$\mathcal{J} = \mathcal{J}_0 + \sum_j (\mathcal{J}_p)_j \quad (3.16)$$

where \mathcal{J}_0 is the design objective function described in Section 2.2, and $(\mathcal{J}_p)_j$ is the penalty term for each constraint violation.

3.4.1 Volume Constraint

For the volume constraint, the volume enclosed by the geometry V is allowed to deviate from the original volume V_0 by a user-specified factor v_f . Written in the form of (2.2), the volume constraint is expressed as:

$$|V - V_0| - v_f V_0 \leq 0 \quad (3.17)$$

By setting $v_f = 0$ (3.17) becomes an equality constraint. If the constraint is violated, the penalty term is added to the objective function:

$$\mathcal{J}_{p,V} = \omega_V \frac{(|V - V_0| - v_f V_0)^2}{V_0^2} \quad (3.18)$$

The penalty weight ω_V is user-supplied.

3.4.2 Thickness Constraints

For thickness constraints, a minimum thickness is specified at a fixed relative position (x/c , $2y/b$) on the wing. The constraint i is expressed as:

$$t_i^* - t_i \leq 0 \quad (3.19)$$

where t_i is the current thickness and t_i^* is the target minimum thickness. A penalty term is added only if t_i is lower than t_i^* :

$$\mathcal{J}_{p,i} = \left(1 - \frac{t_i}{t_i^*}\right)^2 \quad \text{if } t_i < t_i^* \quad (3.20)$$

The contributions from all thickness constraints are summed and multiplied by a user supplied weight ω_T :

$$\mathcal{J}_{p,T} = \omega_T \sum_i \mathcal{J}_{p,i} \quad (3.21)$$

Chapter 4

Flow Analysis

In order to obtain the lift, drag coefficients or pressure distribution needed to compute the objective functions, a Newton-Krylov flow solver is used to obtain the flow solution. This flow solver is largely based on Hicken and Zingg [70, 73], Nichols and Zingg [139], Nemec *et al.* [135, 128], Pueyo and Zingg [152] and Pulliam [153].

4.1 Governing Equations

The governing equations referred to in Section 2.1 are the Euler equations, which assume the fluid to be inviscid. In conservative form, they can be written as:

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0 \quad (4.1)$$

where Q is the vector of conservative variables, and E , F and G are the inviscid convective fluxes:

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e + p) \end{bmatrix}, \quad F = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ v(e + p) \end{bmatrix}, \quad \text{and} \quad G = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ w(e + p) \end{bmatrix} \quad (4.2)$$

Here, ρ is the density of the fluid, u , v and w are the velocity components of fluid, and e is fluid's total energy. The vectors Q , E , F and G are functions of x , y and z . From the equation of state for a perfect gas, pressure p can be related to the flow variables by:

$$p = (\gamma - 1) \left[e - \frac{1}{2}(u^2 + v^2 + w^2) \right] \quad (4.3)$$

where the ratio of specific heats γ is 1.4 for a diatomic gas such as air.

4.2 Coordinate Transformation

A generalized coordinate transformation is used to map the curvilinear structured grid (in the physical domain with coordinates x , y and z) into a square grid (in the computational domain with coordinates ξ , η , ζ) based on Pulliam [153] and Pulliam and Steger [154]:

$$\begin{aligned}\xi &= \xi(x, y, z) \\ \eta &= \eta(x, y, z) \\ \zeta &= \zeta(x, y, z)\end{aligned}\tag{4.4}$$

In the computational domain, the grid spacing between nodes is equal to one. This substantially simplifies the spatial discretization procedures. A 2D example is shown in Figure 4.1.

In the computational domain, the Euler equations can be re-written in the form:

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{G}}{\partial \zeta} = 0\tag{4.5}$$

where

$$\hat{Q} = J^{-1}Q, \quad \hat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho uU + \xi_x p \\ \rho vU + \xi_y p \\ \rho wU + \xi_z p \\ (e + p)U \end{bmatrix}, \quad \hat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ \rho wV + \eta_z p \\ (e + p)V \end{bmatrix}, \quad \text{and} \quad \hat{G} = J^{-1} \begin{bmatrix} \rho W \\ \rho uW + \zeta_x p \\ \rho vW + \zeta_y p \\ \rho wW + \zeta_z p \\ (e + p)W \end{bmatrix}\tag{4.6}$$

and U , V and W are contravariant velocities, given by:

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \xi_y & \eta_y & \zeta_y \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}\tag{4.7}$$

The partial derivative terms in the matrix are referred to as the grid metrics. Directly computing the metric terms is not straightforward. Instead, the metric terms are computed based on the reverse transformation:

$$\begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \xi_y & \eta_y & \zeta_y \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1}\tag{4.8}$$

By taking the inverse of the right-hand-side matrix in (4.8), the partial derivatives in (4.7) are expressed as:

$$\begin{aligned}\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) & \xi_y &= J(z_\eta x_\zeta - z_\zeta x_\eta) & \xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) \\ \eta_x &= J(z_\xi y_\zeta - z_\zeta y_\xi) & \eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) & \eta_z &= J(y_\xi x_\zeta - y_\zeta x_\xi) \\ \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) & \zeta_y &= J(z_\xi x_\eta - z_\eta x_\xi) & \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi)\end{aligned}\tag{4.9}$$

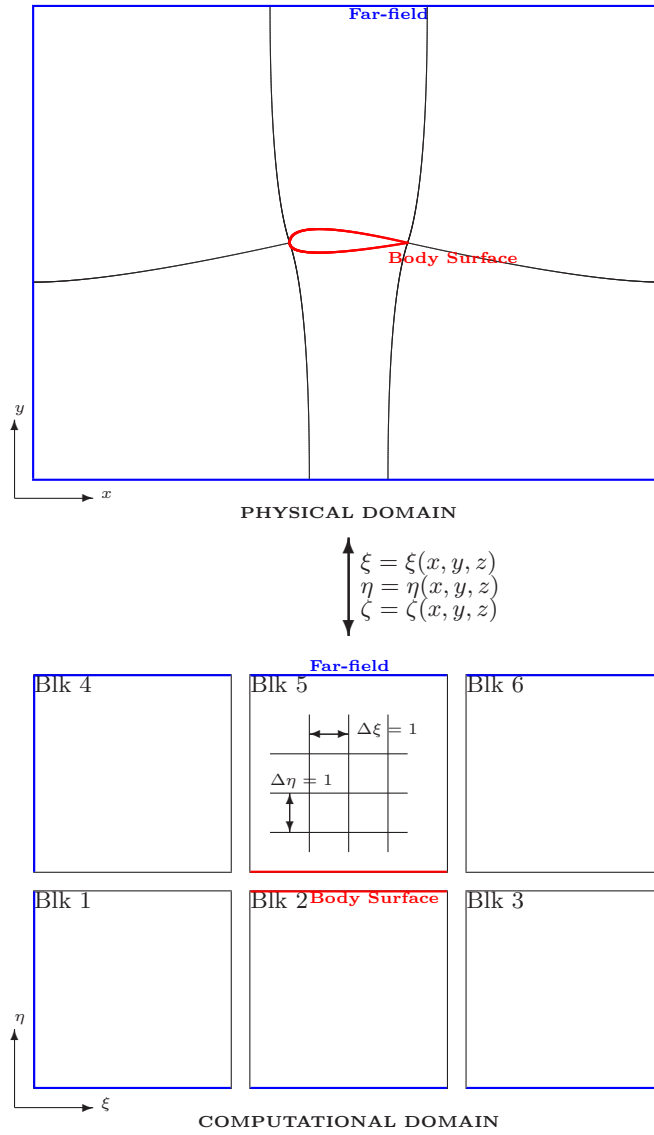


Figure 4.1: Mapping from physical domain to computational domain

The partial derivatives with respect to ξ , η and ζ can be obtained by applying finite differencing at each grid node. However, if centred differencing is used to compute the metric terms, the metric invariants are not satisfied [153, 154]. Instead, the centred difference is averaged. For example ξ_x is expressed as:

$$\xi_x = [(\mu_\zeta \delta_\eta y)(\mu_\eta \delta_\zeta z) - (\mu_\eta \delta_\zeta y)(\mu_\zeta \delta_\zeta z)] \quad (4.10)$$

where δ is the centred difference operator, and μ is an average operator. Finally, J^{-1} is the Jacobian (determinant) of the transformation, given by:

$$J^{-1} = (x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta + x_\zeta y_\eta z_\xi) \quad (4.11)$$

4.3 Spatial Discretization

The Euler equations in (4.5) can be re-written in the form:

$$\frac{1}{J} \frac{\partial Q}{\partial t} + R(Q) = 0 \quad (4.12)$$

where

$$R(Q) = \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{G}}{\partial \zeta} \quad (4.13)$$

is the flow residual. In spatial discretization, the continuous flow variables $Q(x, y, z)$ are stored as discrete quantities at the nodes of the computational grid, i.e. $\mathbf{Q}_{j,k,m}$ is stored at each nodal location with indices (j, k, m) in each block. Similarly, only nodal values for the inviscid convective fluxes ($\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$) and residual \mathbf{R} are computed. The vectors $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$ and \mathbf{R} are defined the same way analogous to $Q \rightarrow \mathbf{Q}$.¹ Figure 4.2 shows a 12-block H-H topology grid around a DPW-W1 wing, generated using the commercially available software program ICEMCFD.

A finite-difference discretization scheme is used to approximate the derivatives of the inviscid fluxes $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$ and $\hat{\mathbf{G}}$ at each node. The spatial discretization scheme turns the system of PDEs into a coupled system of ODEs. This scheme is based on [139, 153]. In a distributed-memory parallel environment, each block of the computational grid is stored in memory locations of different processors. Communications between the processors are only needed for interface nodes. After spatial discretization, the Euler equations in transformed coordinates can be expressed in semi-discrete form:

$$\frac{1}{J} \frac{d\mathbf{Q}}{dt} + \mathbf{R}(\mathbf{Q}) = \mathbf{0} \quad (4.14)$$

where $\mathbf{Q} = [\mathbf{Q}_{1,1,1}, \mathbf{Q}_{1,1,2}, \dots, \mathbf{Q}_{j,k,m}]^T$ is a vector of discrete flow variables, and \mathbf{R} is the discrete residual, which also contains the appropriate boundary conditions.

4.3.1 Inviscid Fluxes

At the interior nodes, spatial discretization of the inviscid fluxes is done using second-order centred-differencing, with an artificial dissipation term added. The discrete residual \mathbf{R} at each interior node can be expressed as:

$$\mathbf{R}(\mathbf{Q})_{j,k,m} = \frac{\hat{\mathbf{E}}_{j+1,k,m} - \hat{\mathbf{E}}_{j-1,k,m}}{2} - \mathbf{D}_{j,k,m}^{(\xi)} + \frac{\hat{\mathbf{F}}_{j,k+1,m} - \hat{\mathbf{F}}_{j,k-1,m}}{2} - \mathbf{D}_{j,k,m}^{(\eta)} + \frac{\hat{\mathbf{G}}_{j,k,m+1} - \hat{\mathbf{G}}_{j,k,m-1}}{2} - \mathbf{D}_{j,k,m}^{(\zeta)} \quad (4.15)$$

$$\frac{\hat{\mathbf{G}}_{j,k,m+1} - \hat{\mathbf{G}}_{j,k,m-1}}{2} - \mathbf{D}_{j,k,m}^{(\zeta)} \quad (4.16)$$

¹For clarity, \hat{Q} , \hat{E} , \hat{F} , \hat{G} and \hat{R} represent continuous functions of flow variables and fluxes in the ξ , η and ζ coordinates, while the discrete form is presented in bold face: \mathbf{Q} , $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$ and \mathbf{R} .

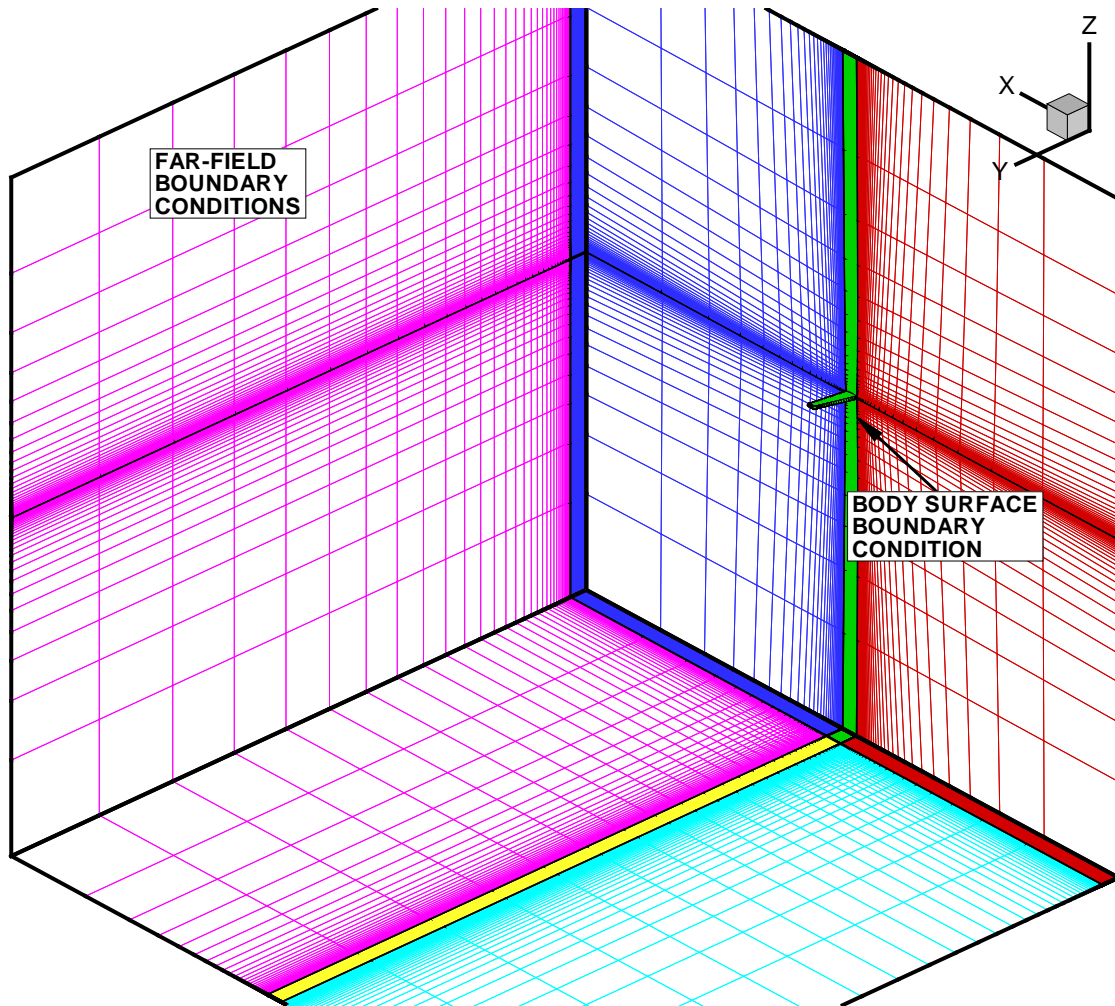


Figure 4.2: An example H-H topology grid around a DPW-W1 wing

where $\mathbf{D}_{j,k,m}^{(\xi)}$, $\mathbf{D}_{j,k,m}^{(\eta)}$ and $\mathbf{D}_{j,k,m}^{(\zeta)}$ are artificial dissipation terms in the ξ , η and ζ directions.

4.3.2 Artificial Dissipation

Artificial dissipation is used because nonlinear processes governed by the Euler equations introduce a continuous production of high-frequency components into the flow field. Their presence may cause physically unrealistic properties or oscillations, and create instabilities in the numerical solution. In real flows, these high-frequency components, such as shocks, are damped by the viscosity of the fluid. However, no such damping mechanism exists in the Euler equation, where the fluid is assumed to be inviscid. The dissipation is based on the work by Jameson *et al.* [84], which has both second-difference (first-order) and fourth-difference (third-order) terms. Dissipation for a one-dimensional domain is shown here, but the expression can easily be shown for

three-dimensional cases. The artificial dissipation terms in a one-dimensional case are:

$$\mathbf{D} = \underbrace{\Delta_\xi^T \mathbf{D}^{(2)} \Delta_\xi \mathbf{Q}}_{\text{2nd-Difference}} + \underbrace{\Delta_\xi^T \mathbf{D}^{(4)} \Delta_\xi \mathbf{B} \Delta_\xi^T \Delta_\xi \mathbf{Q}}_{\text{4th-Difference}} \quad (4.17)$$

where the matrix $\Delta_\xi \in \mathbb{R}^{(J-1) \times J}$ represents the first-order forward-difference operator, its transpose Δ_ξ^T represents the backward difference operator, and $\mathbf{B} \in \mathbb{R}^{J \times J}$ is a diagonal matrix:

$$\Delta_\xi = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}; \text{ and } \mathbf{B} = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix} \quad (4.18)$$

The diagonal matrices $\mathbf{D}^{(2)} \in \mathbb{R}^{(J-1) \times (J-1)}$ and $\mathbf{D}^{(4)} \in \mathbb{R}^{(J-1) \times (J-1)}$ contain the coefficients for second- and fourth-difference dissipation:

$$\mathbf{D}^{(2)} = \begin{bmatrix} d_{1\frac{1}{2}}^{(2)} & & & & \\ & d_{2\frac{1}{2}}^{(2)} & & & \\ & & \ddots & & \\ & & & d_{J-\frac{1}{2}}^{(2)} & \end{bmatrix} \quad \mathbf{D}^{(4)} = \begin{bmatrix} d_{1\frac{1}{2}}^{(4)} & & & & \\ & d_{2\frac{1}{2}}^{(4)} & & & \\ & & \ddots & & \\ & & & d_{J-\frac{1}{2}}^{(4)} & \end{bmatrix} \quad (4.19)$$

where the coefficients are given by:

$$d_{j+\frac{1}{2}}^{(2)} = 2 (\epsilon \sigma J^{-1})_{j+\frac{1}{2},k,m} \quad (4.20)$$

and

$$d_{j+\frac{1}{2}}^{(4)} = \max \left[0, 2\kappa_4 (\sigma J^{-1})_{j+\frac{1}{2}} - d_{j+\frac{1}{2}}^{(2)} \right] \quad (4.21)$$

The values at the half nodes $j + \frac{1}{2}$ are calculated by averaging between the two nodal points:

$$(\cdot)_{j+\frac{1}{2}} = \frac{(\cdot)_{j+1} + (\cdot)_j}{2} \quad (4.22)$$

and σ is the spectral radius of the flux Jacobian:

$$\sigma = |U| + a \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \quad (4.23)$$

$$\epsilon_j = \kappa_2 [0.5\Upsilon_j^* + 0.25(\Upsilon_{j-1}^* + \Upsilon_{j+1}^*)] \quad (4.24)$$

$$\Upsilon_j^* = \max[\Upsilon_{j-1}, \Upsilon_j, \Upsilon_{j+1}] \quad (4.25)$$

$$\Upsilon_j = \frac{|p_{j+1} - 2p_j + p_{j-1}|}{|p_{j+1} + 2p_j + p_{j-1}|} \quad (4.26)$$

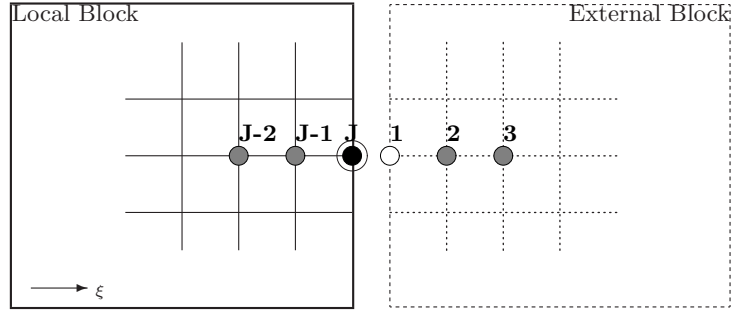


Figure 4.3: Stencil for evaluating flux and dissipation without SATs

The term Υ_j acts as a pressure switch that is activated near shocks, while Υ_j^* acts as a smoother for Υ_j . (4.21) is used to switch between using second-difference dissipation near shocks, and fourth-difference elsewhere. At block boundaries, values of ϵ , Υ^* and Υ at boundary nodes J are extrapolated from the adjacent interior nodes. The dissipation constants κ_2 and κ_4 are user supplied.

4.3.3 Block Interfaces and Boundary Conditions

Figure 4.3 shows the general stencils for residual evaluation at block interfaces. Here, node J on the local block is coincident with node 1 on the adjacent external block. In order to evaluate the fourth-difference dissipation term in the ξ direction at node J , information is required from nodes $J - 1$ and $J - 2$ in the local block, and from nodes 2 and 3 from the adjacent block. In a parallel implementation, this adjacent block is likely to be stored in a memory location that is only accessible by another processor. In this case, flow variables near block interfaces must be passed to the appropriate processors via message passing, affecting the scalability of the parallel method. Additionally, for accurate flux and dissipation evaluation at node J , the grid must be at least C^1 -continuous across the interface [73]. This can potentially limit the types of grids that can be used effectively.

This difficulty is addressed through the application of simultaneous approximation terms (SATs). The method has been successfully used to enforce boundary conditions [22] and block interfaces [23]. SATs are derived using summation-by-parts (SBP) operators. For further information about the inviscid SATs for the Euler equations, see Hicken and Zingg [73]. An example is shown of the application of SATs in a quasi-one-dimensional problem with a converging-diverging nozzle, using the Euler equations. The method is generalized for 3D problems encountered in this thesis. In the quasi-1D example, numerical dissipation terms are neglected.

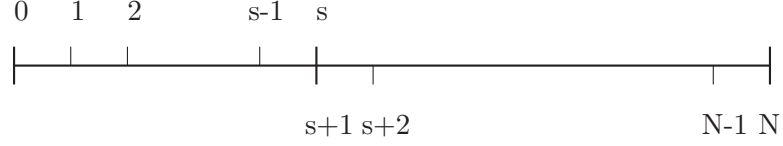


Figure 4.4: One-dimensional domain with two sub-domains

The flow equations can be written as:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} - \mathbf{G} = \mathbf{0} \quad (4.27)$$

where

$$\mathbf{Q} = \begin{bmatrix} \rho S \\ \rho u S \\ \rho E S \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} \rho u S \\ (\rho u^2 + p) S \\ \rho u H S \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0 \\ p \frac{\partial S}{\partial x} \\ 0 \end{bmatrix} \quad (4.28)$$

and S is the nozzle cross-section area. The computational domain is divided into two sub-domains, as shown in Figure 4.4. In this example, the nodes are equally spaced. The interface nodes of the two sub-domains are coincident, i.e. $x_s = x_{s+1}$. In each of the sub-domains, the semi-discrete form of the equation can now be written using SBP operators:

$$\begin{aligned} (\mathcal{P}_L \otimes \mathcal{I}_3) \frac{d\mathbf{Q}_L}{dt} + (\mathcal{Q}_L \otimes \mathcal{I}_3) \mathbf{E}_L - (\mathcal{P}_L \otimes \mathcal{I}_3) \mathbf{G}_L &= \boldsymbol{\Sigma}_L \\ (\mathcal{P}_R \otimes \mathcal{I}_3) \frac{d\mathbf{Q}_R}{dt} + (\mathcal{Q}_R \otimes \mathcal{I}_3) \mathbf{E}_R - (\mathcal{P}_R \otimes \mathcal{I}_3) \mathbf{G}_R &= \boldsymbol{\Sigma}_R \end{aligned} \quad (4.29)$$

where \mathcal{I}_3 is a 3×3 identity matrix. The SBP operators for the left sub-domain are given by:

$$\mathcal{P}_L = \begin{bmatrix} \Delta x_0 & & & & \\ & \Delta x_1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \Delta x_s \end{bmatrix}, \quad \mathcal{Q}_L = \frac{1}{2} \begin{bmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 0 & 1 \\ & & & & -1 & 1 \end{bmatrix} \quad (4.30)$$

where $\Delta x_0 = (x_1 - x_0)/2$, $\Delta x_s = (x_s - x_{s-1})/2$, and $\Delta x_i = (x_{i+1} - x_{i-1})/2$ for $i = 2 \dots s-1$. SBP operators on the right domain are similarly defined. Note that multiplying $\mathcal{P}_L^{-1} \mathcal{Q}_L$ recovers the second-order centred differencing operator for the interior nodes, and the first-order one-sided differencing operator at the boundary and interface nodes. The flow variable, inviscid flux and source terms for the left and right sub-domains are given by:

$$\begin{aligned} \mathbf{Q}_L &= [\mathbf{Q}_1, \dots, \mathbf{Q}_s]^T, & \mathbf{Q}_R &= [\mathbf{Q}_{s+1}, \dots, \mathbf{Q}_N]^T \\ \mathbf{E}_L &= [\mathbf{E}_1, \dots, \mathbf{E}_s]^T, & \mathbf{E}_R &= [\mathbf{E}_{s+1}, \dots, \mathbf{E}_N]^T \\ \mathbf{G}_L &= [\mathbf{G}_1, \dots, \mathbf{G}_s]^T, & \mathbf{G}_R &= [\mathbf{G}_{s+1}, \dots, \mathbf{G}_N]^T \end{aligned} \quad (4.31)$$

The operator \otimes is the Kronecker product for matrices.² Finally, the penalty terms on the right-hand-side of (4.30) are given by:

$$\boldsymbol{\Sigma}_L = \begin{bmatrix} -\frac{1}{2}(|A| + A)(\mathbf{Q}_1 - \mathbf{Q}_{BC,L}) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ -\frac{1}{2}(|A| - A)(\mathbf{Q}_s - \mathbf{Q}_{s+1}) \end{bmatrix}, \quad \boldsymbol{\Sigma}_R = \begin{bmatrix} -\frac{1}{2}(|A| + A)(\mathbf{Q}_{s+1} - \mathbf{Q}_s) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ -\frac{1}{2}(|A| - A)(\mathbf{Q}_N - \mathbf{Q}_{BC,R}) \end{bmatrix} \quad (4.32)$$

At the interface, a penalty term is applied when $\mathbf{Q}_s \neq \mathbf{Q}_{s+1}$. Similarly, for boundary conditions, penalty terms are added when $\mathbf{Q}_1 \neq \mathbf{Q}_{BC,L}$ and $\mathbf{Q}_N \neq \mathbf{Q}_{BC,R}$. Body surface and far-field boundary conditions will be described for 3D cases later in this section.

To compute the penalty term at the interface, the averaged state on both sides of the interface $\frac{1}{2}(\mathbf{Q}_s + \mathbf{Q}_{s+1})$ is used to compute the (3×3) Jacobian matrix (A). The matrix $|A|$ is determined from $|A| = X|\Lambda|X^{-1}$, where X are the right eigenvectors of A , and

$$|\Lambda| = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \quad \text{with} \quad \begin{aligned} \lambda_1 &= \max(|u + a|, V_n \sigma(A)) \\ \lambda_2 &= \max(|u - a|, V_n \sigma(A)) \\ \lambda_3 &= \max(|u|, V_l \sigma(A)) \end{aligned} \quad (4.33)$$

where $\sigma(A)$ is the spectral radius of A , and the constants V_n and V_l are used to avoid the effect of overly small eigenvalues in the Jacobian matrix at the interface. For transonic flows, $V_n = 0.25$ and $V_l = 0.025$ are used based on [26]. The eigenvectors can be found in Pulliam [153]. The $|A| - A$ term in (4.32) ensures that the appropriate boundary conditions are imposed to the incoming and outgoing waves at the boundary. Therefore, the Riemann invariants do not need to be calculated.

This formulation can be extended to general 3D problems. Figure 4.5 shows the new stencil with SATs. In general, \mathbf{Q}_j in the local block and \mathbf{Q}_1 in the external block do not have the same values. To evaluate the residual at interface node (J, k, m) , including dissipation terms, one-sided differencing is applied in the ξ direction, and centred differencing is applied in the η and ζ directions:

$$\begin{aligned} \mathbf{R}(\mathbf{Q})_{J,k,m} &= \hat{\mathbf{E}}_{J,k,m} - \hat{\mathbf{E}}_{J-1,k,m} - \mathbf{D}_{j,k,m}^{(\xi)} + \\ &\quad \frac{\hat{\mathbf{F}}_{J,k+1,m} - \hat{\mathbf{F}}_{J,k-1,m}}{2} - \mathbf{D}_{j,k,m}^{(\eta)} + \\ &\quad \frac{\hat{\mathbf{G}}_{J,k,m+1} - \hat{\mathbf{G}}_{j,k,m-1}}{2} - \mathbf{D}_{j,k,m}^{(\eta)} + (\boldsymbol{\Sigma}_L)_{J,k,m} \end{aligned} \quad (4.34)$$

²If \mathcal{A} is a $m \times n$ matrix and \mathcal{B} is a $p \times q$ matrix, then $\mathcal{C} = \mathcal{A} \otimes \mathcal{B}$ is a $mp \times nq$ matrix whose elements are given by $\mathcal{C}_{p(i-1)+k, q(j-1)+l} = \mathcal{A}_{ij} \mathcal{B}_{kl}$

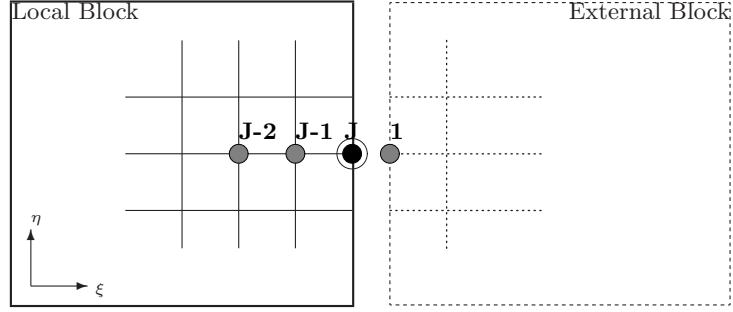


Figure 4.5: Stencil for SAT terms at block interfaces

The treatment of dissipation terms near block interfaces is the same as at domain boundaries described in Section 4.3.2. For second-difference dissipation, a $(-1, 1, 0)$ stencil replaces the interior stencil $(-1, 2, -1)$ at boundary node J . For fourth-difference dissipation, a first-order stencil $(-1, 2, -1, 0, 0)$ is used at the boundary node J , and a second-order stencil $(-1, 4, -5, 2, 0)$ at the $J - 1$ node [115].

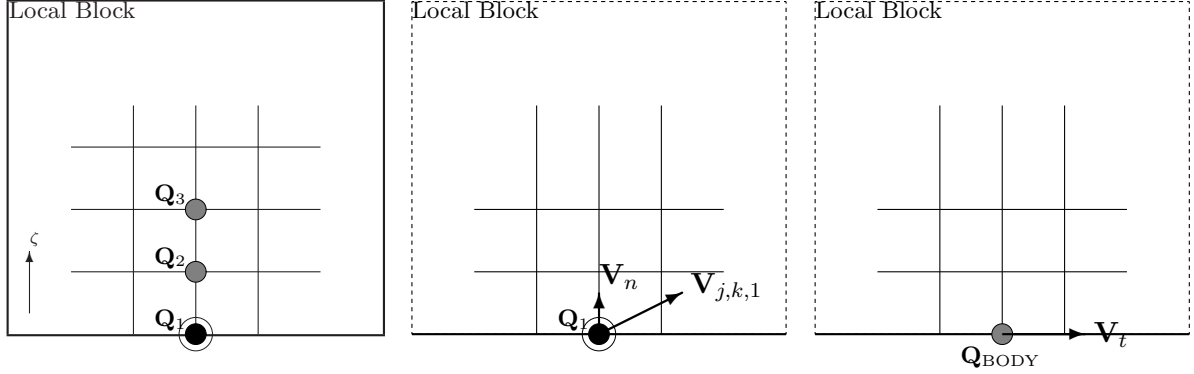
Flow variable $\mathbf{Q}_{j,k,m}$ from the external block is now coupled to the local block through the penalty term $(\boldsymbol{\Sigma}_{\mathbf{L}})_{j,k,m}$. The penalty term is derived in the same way as in (4.32), but with the matrices $|\Lambda|$, A and X now 5×5 matrices.

The application of the SATs to boundary conditions is similar to interfaces. Boundaries include the body surface and the far-field regions of the physical domain. Figure 4.2 shows the location of the boundaries. As shown in (4.32), applying the boundary conditions means choosing appropriate values for \mathbf{Q}_{BODY} (body surfaces) and \mathbf{Q}_{FF} (far-field).

In the Euler equations, the body surface is treated as a solid wall where no fluid is allowed to flow through. Therefore, the velocity vector normal to the surface must be zero, and the only velocity component of the flow is tangent to the body surface. The value for \mathbf{Q}_{BODY} is determined from the boundary node. An example is shown in Figure 4.6 to demonstrate the application of SATs at a body surface boundary at $m = 1$. As in the case of the block interface, flux and dissipation terms are calculated using one-sided differencing at boundary node $(j, k, 1)$ (Figure 4.6a). The normal component of the flow velocity \mathbf{V}_n at $(j, k, 1)$ is then computed (Figure 4.6b). It is given by the equation:

$$\mathbf{V}_n = \frac{\zeta_z}{\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}} u \hat{\mathbf{i}} + \frac{\zeta_x}{\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}} v \hat{\mathbf{j}} + \frac{\zeta_y}{\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}} w \hat{\mathbf{k}} \quad (4.35)$$

The normal component \mathbf{V}_n is subtracted from $\mathbf{V}_{j,k,1}$ to obtain the tangential velocity \mathbf{V}_t . The



(a) Construct residual terms using one-sided differencing (b) Find normal velocity component and subtract from $\mathbf{V}_{j,k,1}$ (c) Find \mathbf{Q}_{BODY} using tangential velocity

Figure 4.6: Computing \mathbf{Q}_{BODY} for SAT terms at body surface boundaries

body boundary is therefore:

$$\mathbf{Q}_{\text{BODY}} = \frac{1}{J} \begin{bmatrix} \rho_{j,k,1} \\ \rho_{j,k,1} \mathbf{V}_t \cdot \hat{\mathbf{i}} \\ \rho_{j,k,1} \mathbf{V}_t \cdot \hat{\mathbf{j}} \\ \rho_{j,k,1} \mathbf{V}_t \cdot \hat{\mathbf{k}} \\ \rho_{j,k,1} H_\infty - p_{j,k,1} \end{bmatrix} \quad (4.36)$$

The SATs are then added to the residual vector in the same way as the block interface, shown in (4.35). This same boundary condition is also applied to the symmetry plane.

The same strategy is used for the far-field boundaries. In this case, \mathbf{Q}_{FF} contains the far-field values:

$$\mathbf{Q}_{\text{FF}} = \begin{bmatrix} 1.0 \\ M_\infty \cos \alpha \\ 0 \\ M_\infty \sin \alpha \\ \frac{1}{\gamma(\gamma-1)} + \frac{1}{2} M_\infty^2 \end{bmatrix} \quad (4.37)$$

where α is the angle of attack of the wing.

4.4 Newton-Krylov Flow Solver

A steady-state solution is used to compute the objective function. At steady state, where $d\mathbf{Q}/dt = \mathbf{0}$, the residual must vanish:

$$\mathbf{R}(\mathbf{Q}) = \mathbf{0} \quad (4.38)$$

This is the flow constraint described in (2.3). In the Newton method, the iterative process begins with an initial guess $\mathbf{Q}^{(0)}$ based on free-stream properties, noting that the initial guess is unlikely to satisfy the steady-state requirement, i.e. $\mathbf{R}(\mathbf{Q}^{(0)}) \neq \mathbf{0}$. A correction term $\Delta\mathbf{Q}^{(n)}$ is sought such that $\mathbf{R}(\mathbf{Q}^{(n)} + \Delta\mathbf{Q}^{(n)}) = \mathbf{0}$. The first-order terms in the Taylor series expansion about $\mathbf{Q}^{(n)}$ gives:

$$\mathbf{R}(\mathbf{Q}^{(n)} + \Delta\mathbf{Q}^{(n)}) \approx \mathbf{R}(\mathbf{Q}^{(n)}) + \mathbf{A}^{(n)} \Delta\mathbf{Q}^{(n)} \quad (4.39)$$

where the matrix

$$\mathbf{A}^{(n)} = \left. \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right|^{(n)} \quad (4.40)$$

is the flow Jacobian. Setting the left-hand side of (4.39) to zero, the correction $\Delta\mathbf{Q}$ can be obtained by solving the linear system:

$$\mathbf{A}^{(n)} \Delta\mathbf{Q}^{(n)} = -\mathbf{R}^{(n)} \quad (4.41)$$

where $\mathbf{R}^{(n)} = \mathbf{R}(\mathbf{Q}^{(n)})$. Since only the first-order term in the Taylor series expansion in (4.39) is used, this process must be repeated until a suitable convergence is obtained. The update at each iteration is given by:

$$\mathbf{Q}^{(n+1)} = \mathbf{Q}^{(n)} + \Delta\mathbf{Q}^{(n)} \quad (4.42)$$

The linear system in (4.41) is solved at each iteration n (“outer iteration”) until a steady-state flow solution is obtained. For optimization problems, the adjoint gradient (Section 5.1.2) requires the flow Jacobian from a fully converged solution in order to obtain an accurate gradient, and therefore ten orders of magnitude reduction in the residual L_2 -norm is needed:

$$\frac{\|\mathbf{R}^{(n)}\|_2}{\|\mathbf{R}^{(0)}\|_2} \equiv \mathcal{R}^{(n)} < 10^{-10} \quad (4.43)$$

For general flow analysis, in order to compute lift and drag (L and D), the norm of the residual vector only has to decrease by a few orders of magnitude.

The Newton method converges quadratically when \mathbf{Q} is sufficiently close to the solution. With each iteration, the number of significant digits doubles. However, during start-up, when the iterate $\mathbf{Q}^{(n)}$ is far from the steady-state solution, convergence may not be possible. An approximate-Newton start-up phase is used to help find a suitable initial iterate for the full Newton method.

In the approximate-Newton phase, a first-order flow Jacobian (denoted by \mathbf{A}_1) replaces the full Jacobian matrix \mathbf{A} in (4.41). The first-order Jacobian is formed by combining the second- and fourth-difference dissipation terms to reduce the bandwidth of the matrix:

$$\hat{\kappa}_2 = \kappa_2 + \sigma\kappa_4; \quad \hat{\kappa}_4 = 0 \quad (4.44)$$

A value of $4 \leq \sigma \leq 6$ is recommended based on [139]. Replacing \mathbf{A} with \mathbf{A}_1 does not affect the steady-state solution, since it does not modify the residual calculation on the right hand side. The approximate-Newton method can be used to solve the flow solution to steady state [203]. However, the convergence rate is generally slower. A spatially varying time step is also added at each node i as a globalization strategy:

$$\Delta t_i^{(n)} = \frac{\Delta t_{\text{ref}}^{(n)}}{J_i(1 + \sqrt[3]{J_i})} \quad (4.45)$$

where the reference time step for iteration n is defined as

$$\Delta t_{\text{ref}}^{(n)} = A(B)^n \quad (4.46)$$

Values of $A = 0.1$ and $B = 1.5$ are used. The value for B can be lowered if convergence in early iterations is difficult. To summarize, during the approximate-Newton start-up phase, the linear system solved at each iteration n is given by:

$$\left[\mathbf{T}^{(n)} \otimes \mathcal{I}_5 + \mathbf{A}_1^{(n)} \right] \Delta \mathbf{Q}^{(n)} = \mathbf{R}^{(n)} \quad (4.47)$$

where $\mathbf{T}^{(n)}$ is a diagonal matrix containing the reciprocal of the local time step at each node, and \mathcal{I}_5 is a 5×5 identity matrix.

The flow solver switches to the Newton phase when the normalized residual has dropped below a threshold $\mathcal{R}^{(n)} < \tau$. A value of $\tau = 0.1$ is used based on [73]. During the Newton phase, the pseudo-transient time step continues to be added to the diagonal of the matrix, with the reference time step based on Mulder and van Leer [123]:

$$\Delta t_{\text{ref}}^{(n)} = \max \left[\alpha \left(\|\mathcal{R}^{(n)}\|_2 \right)^{-\beta}, \Delta t_{\text{ref}}^{(n-1)} \right] \quad (4.48)$$

with $\alpha = 1.0$ and $\beta = 1.75$. As the residual decreases, the time step approaches infinity, and the full Newton step is recovered.

4.4.1 Preconditioned GMRES

At each iteration during the approximate-Newton and Newton phases, a sparse linear system in the form

$$\mathbf{Ax} = \mathbf{b} \quad (4.49)$$

must be solved. For the size of the problems encountered in most CFD applications, directly solving the matrix (e.g. Gaussian elimination, lower-upper factorization) is too expensive to be practical, while stationary iteration methods (Jacobi, Gauss-Seidel) converge too slowly. Instead, a Krylov subspace method can be used to solve the system inexactly. In particular, the General Minimal Residual (GMRES) [163, 162] method is found to be well-suited for CFD

applications. The basic GMRES algorithm is presented in Appendix D. Krylov methods do not require the matrix \mathbf{A} to be inverted, instead only using a matrix-vector product with \mathbf{A} . This matrix-vector product in GMRES can be approximated using forward differencing:

$$\mathbf{A}\mathbf{v} \approx \frac{\mathbf{R}(\mathbf{Q} + \epsilon\mathbf{v}) - \mathbf{R}(\mathbf{Q})}{\epsilon} \quad (4.50)$$

The perturbation parameter ϵ is chosen based on [143] such that truncation error and round-off errors are minimized:

$$\epsilon = \sqrt{\frac{N\delta}{\mathbf{v}^T\mathbf{v}}} \quad (4.51)$$

where N is the number of unknowns and $\delta = 10^{-13}$. With this approximation, the flow Jacobian \mathbf{A} does not have to be formed, leading to a ‘‘Jacobian-free’’ approach.

To further improve the convergence of the GMRES algorithm, the linear system in (4.49) is right preconditioned with the matrix \mathbf{M} :

$$(\mathbf{A}\mathbf{M}^{-1})(\mathbf{M}\mathbf{x}) = \mathbf{b} \quad (4.52)$$

Note that right-preconditioning the system does not change the final solution \mathbf{x} . Ideally, the preconditioner matrix \mathbf{M} is a matrix that closely approximates \mathbf{A} , but significantly easier to invert. Therefore, by preconditioning, the eigenvalues of $\mathbf{A}\mathbf{M}^{-1}$ are much closer to unity than those of \mathbf{A} , thus improving the efficiency of GMRES. It should be noted that the preconditioner matrix \mathbf{M} still needs to be formed and inverted.

The linear system is then solved inexactly, such that

$$\left\| \mathbf{R}^{(n)} - \left(\mathbf{T}^{(n)} \otimes \mathcal{I}_5 + \mathbf{A}^{(n)} \right) \Delta \mathbf{Q}^{(n)} \right\|_2 \leq \eta_n \left\| \mathbf{R}^{(n)} \right\|_2 \quad (4.53)$$

The forcing parameter η_n gradually decreases from 0.5 to 0.01 using the equation proposed by Eisenstat and Walker [34]:

$$\eta_n = \max \left[0.01, \eta_{n-1}^{(1+\sqrt{5})/2} \right] \quad (4.54)$$

As demonstrated by Dembo *et al.* [29], by approximately solving (4.52), enormous computational effort can be saved while retaining the convergence properties of the Newton method.

For problems using single processors, incomplete lower-upper (ILU) factorization has been shown to be an excellent preconditioner. However, in a parallel environment, ILU preconditioners suffer from the same deficiencies as direct solvers, namely that the factorization is not easily made parallel. Instead, a preconditioner based on domain decomposition is implemented.

4.4.2 Approximate Schur Preconditioning

The approximate-Schur preconditioner presented here follows the development by Saad and Sosenkina [164, 162]. In formulating the approximate-Schur preconditioner for GMRES, the

matrix is divided among S sub-domains:

$$\begin{bmatrix} A_{11} & \cdots & A_{1s} \\ \vdots & & \vdots \\ A_{s1} & \cdots & A_{ss} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_s \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_s \end{bmatrix} \quad (4.55)$$

where $A_{i1} \dots A_{is}$ are stored on processor i . The sub-domains are generally non-overlapping. With careful ordering of nodes, the diagonal blocks (A_{ii}) and off-diagonal blocks (A_{ij}) in each sub-domain can be written as:

$$A_{ii} = \begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 \\ E_{ij} \end{bmatrix} \quad (4.56)$$

where B_i are the internal entries, C_i are the interface entries, and E_i , E_{ij} and F_i represent the coupling between internal and interface entries. The solution vector \mathbf{x} and the right-hand-side \mathbf{b} can be similarly partitioned for internal and interface entries:

$$x_i = \begin{bmatrix} u_i \\ y_i \end{bmatrix}, \quad b_i = \begin{bmatrix} f_i \\ g_i \end{bmatrix} \quad (4.57)$$

The part of the linear system that is local to the sub-domain i can now be written as:

$$\begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix} \begin{bmatrix} u_i \\ y_i \end{bmatrix} + \begin{bmatrix} 0 \\ \sum_{i \neq j} E_{ij} y_j \end{bmatrix} = \begin{bmatrix} f_i \\ g_i \end{bmatrix} \quad (4.58)$$

The local entries u_i can be solved once the interface entries y_i are known:

$$u_i = B_i^{-1}(f_i - F_i y_i) \quad (4.59)$$

Substituting u_i into the equation for y_i the following linear system for the interface can be obtained:

$$S_i y_i + \sum_{i \neq j} E_{ij} y_j = g_i - F_i B_i^{-1} f_i \quad (4.60)$$

where S_i is the ‘‘local Schur complement matrix’’, defined as:

$$S_i = C_i - F_i B_i^{-1} E_i \quad (4.61)$$

S_i is generally dense. When written out for all sub-domains, the overall linear system from (4.60) now becomes:

$$\begin{bmatrix} S_1 & E_{12} & \cdots & \cdots & E_{1s} \\ E_{21} & S_2 & E_{23} & \cdots & E_{2s} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ E_{s1} & E_{s2} & E_{s3} & \cdots & S_s \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ \vdots \\ y_s \end{bmatrix} = \begin{bmatrix} g'_1 \\ \vdots \\ \vdots \\ \vdots \\ g'_s \end{bmatrix} \quad (4.62)$$

The left-hand-side matrix is referred to as the “global Schur complement matrix”. In theory, this Schur complement system can be assembled and solved for each y_i , then for the internal unknowns u_i in (4.58). However, solving this linear system exactly is not competitive. Instead, a system that approximates (4.62) is used as a preconditioner for the global problem (4.49).

Suppose A_{ii} has been factored into $A_{ii} = L_i U_i$, where

$$L_i = \begin{bmatrix} L_{B_i} & 0 \\ E_i U_{B_i}^{-1} & L_{S_i} \end{bmatrix}; \quad U_i = \begin{bmatrix} U_{B_i} & L_{B_i}^{-1} F_i \\ 0 & U_{S_i} \end{bmatrix} \quad (4.63)$$

It can be shown that

$$S_i = L_{S_i} U_{S_i} \quad (4.64)$$

Therefore the LU decomposition of the local Schur complement matrix can be obtained through the LU decomposition of the local matrix A_{ii} . Similarly, an approximate factorization of S_i can be found by the ILU factorization of A_{ii} . The interface unknowns y_i can be solved approximately using this approximate local Schur complement system:

$$y_i = U_{S_i}^{-1} L_{S_i}^{-1} \left[g_i - F_i B_i^{-1} f_i - \sum_{i \neq j} E_{ij} y_j \right] \quad (4.65)$$

This equation is equivalent to a single block-Jacobi iteration, which can be solved using a Krylov-method such as GMRES. Once y_i is found and exchanged on all sub-domains, the local unknowns u_i can be solved approximately, again with ILU factorization.

A linear solver that uses the approximate Schur preconditioner must be “flexible”, in that the preconditioning may change from one inner iteration to the next. In light of this, a flexible variant of GMRES (“FGMRES”) [161] is used. The overall approximate-Schur preconditioning technique with modifications by Hicken and Zingg [70] is summarized in Algorithm 4.1.

Algorithm 4.1: Approximate Schur Preconditioner

Data: m, η, \mathbf{u} **Result:** \mathbf{v}

```

1  $\mathbf{v} = U_{S_i}^{-1} L_i^{-1} \mathbf{u}$  // get residual of reduced system for a guess of zero
2  $\mathbf{w}_{(i)}^{(1)} = P\mathbf{v}$  // P projects onto the space of internal-interface unknowns
3 set  $\beta = \|\mathbf{w}_{(i)}^{(1)}\|_2$ ,  $\mathbf{w}_{(i)}^{(1)} \leftarrow \mathbf{w}_{(i)}^{(1)}/\beta$ , and  $H = 0$ 
4 for  $j = 1, m$  do
5   obtain external interface values,  $\mathbf{w}_{(i,\text{ext})}^{(j)}$ 
6    $\mathbf{w}_{(i)}^{(j+1)} = E_i \mathbf{w}_{(i,\text{ext})}^{(j)}$  // perform external matrix-vector product
7    $\mathbf{w}_{(i)}^{(j+1)} \leftarrow U_{S_i}^{-1} L_{S_i}^{-1} \mathbf{w}_{(i)}^{(j+1)}$  // apply diagonal block of Schur complement
8    $\mathbf{w}_{(i)}^{(j+1)} \leftarrow \mathbf{w}_{(i)}^{(j)} + \mathbf{w}_{(i)}^{(j+1)}$  // finish the matrix-vector product
9   for  $k = 1, j$  do Gram-Schmidt orthogonalization
10     $H_{k,j} = (\mathbf{w}_{(i)}^{(j+1)})^T \mathbf{w}_{(i)}^{(k)}$ 
11     $\mathbf{w}_{(i)}^{(j+1)} \leftarrow \mathbf{w}_{(i)}^{(j+1)} - H_{k,j} \mathbf{w}_{(i)}^{(k)}$ 
12   end
13    $H_{j+1,j} = \|\mathbf{w}_{(i)}^{(j+1)}\|_2$ 
14    $\mathbf{w}_{(i)}^{(j+1)} \leftarrow \mathbf{w}_{(i)}^{(j+1)} / H_{j+1,j}$ 
15   if reduced system residual tolerance  $\leq \eta$  then
16     set  $m = j$  and exit
17   end
18 end
19 Define  $W_m = [\mathbf{w}_{(i)}^{(1)}, \dots, \mathbf{w}_{(i)}^{(m)}]$ 
20 Compute  $\mathbf{y}_{(i)} = W_m \mathbf{z}^{(m)}$  where  $\mathbf{z}^{(m)} = \min_{\mathbf{z}} \|\beta e_1 - H\mathbf{z}\|_2$  and  $e_1 = [1, 0, \dots, 0]^T$ 
21 obtain external interface preconditioned values,  $\mathbf{y}_{(i,\text{ext})}$ 
22  $\mathbf{v} \leftarrow \mathbf{v} + P^T P(\mathbf{u} - E_i \mathbf{y}_{(i,\text{ext})} - \mathbf{v})$  // updates internal-interface only
23  $\mathbf{v} \leftarrow U_i^{-1} [I + P^T P(L_i^{-1} - I)] \mathbf{v}$  // forward solve is applied to interface only

```

Chapter 5

Optimizer

5.1 Gradient Evaluation

5.1.1 Finite-Difference Gradient

In the finite-difference gradient calculation, a second-order gradient can be calculated using a centred-difference scheme for each design variable \mathcal{X}_k :

$$\mathcal{G}_k \approx \frac{\mathcal{J}[\mathcal{X} + h\hat{e}_k, \mathbf{Q}(\mathcal{X} + h\hat{e}_k)] - \mathcal{J}[\mathcal{X} - h\hat{e}_k, \mathbf{Q}(\mathcal{X} - h\hat{e}_k)]}{2h} + \mathcal{O}(h^2), \quad k = 1 \dots N_X \quad (5.1)$$

where N_X is the number of design variables, \hat{e}_k is the k -th unit vector, and the finite-difference step size is given by [128]:

$$h = \max(\epsilon \cdot |\mathcal{X}_k|, 1 \times 10^{-4}) \quad (5.2)$$

For the optimization cases examined in Chapter 8, $10^{-4} > \epsilon > 10^{-6}$ is found to be accurate. The advantage of finite-differencing is that it is easy to implement, in that the flow solver can be treated as a “black-box”. However, finite-difference gradients suffer from two important weaknesses. The first is related to the cost of a gradient evaluation. In addition to the flow solve required to compute the objective function, two additional flow solves per design variable are required to compute the gradient. The cost can be reduced to only one additional flow solve per design variable if first-order forward differencing is used. This makes optimization with a large number of design variables prohibitive.

The second issue with finite-differencing is that the accuracy of the gradient is sensitive to the step size h . Generally, finite differences are susceptible to two types of error. The first is a truncation error, which arises because higher order terms in the Taylor series expansion are dropped in formulating the finite-difference operator. In the case of centred differencing (5.1), the truncation error is on the order of $\mathcal{O}(h^2)$. The second error is a subtractive cancellation error. This is due to the bit-wise representation of floating point values. The truncation error

decreases as $h \rightarrow 0$, while the subtractive cancellation generally increases as $h \rightarrow 0$. The selection of the proper step size is often difficult and time consuming. The finite-difference gradient is useful for debugging purposes and to validate gradient accuracies of other methods.

5.1.2 Direct and Adjoint Gradients

In the adjoint and direct methods, the gradient is expressed by taking the partial derivatives of the flow vectors and the design variables:

$$\mathcal{G} = \frac{\partial}{\partial \mathcal{X}} [\mathcal{J}(\mathcal{X}, \mathbf{Q})] = \frac{\partial \mathcal{J}}{\partial \mathcal{X}} \Big|_{\mathbf{Q}} + \frac{\partial \mathcal{J}}{\partial \mathbf{Q}} \Big|_{\mathcal{X}} \frac{\partial \mathbf{Q}}{\partial \mathcal{X}} \quad (5.3)$$

This expression is valid as long as $\partial \mathbf{Q} / \partial \mathcal{X}$ is continuous. Also, from the flow constraint, the residual vector must be zero (steady-state solution) regardless of the design variables, thus:

$$\frac{\partial}{\partial \mathcal{X}} [\mathbf{R}(\mathcal{X}, \mathbf{Q})] = \frac{\partial \mathbf{R}}{\partial \mathcal{X}} \Big|_{\mathbf{Q}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \Big|_{\mathcal{X}} \frac{\partial \mathbf{Q}}{\partial \mathcal{X}} = \mathbf{0} \quad (5.4)$$

Solving for $\partial \mathbf{Q} / \partial \mathcal{X}$ in (5.4) and substituting it back into (5.3) results in the following expression:

$$\mathcal{G} = \frac{\partial \mathcal{J}}{\partial \mathcal{X}} - \underbrace{\frac{\partial \mathcal{J}}{\partial \mathbf{Q}} \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right)^{-1} \frac{\partial \mathbf{R}}{\partial \mathcal{X}}}_{\text{ADJOINT}}^{\text{DIRECT}} \quad (5.5)$$

The expression in (5.5) can be solved using either the adjoint method or the direct method to obtain the gradient. In the direct method, also known as the flow-sensitivity method, an intermediate problem is solved for each design variable \mathcal{X}_k :

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right) \Phi = \frac{\partial \mathbf{R}}{\partial \mathcal{X}_k} \quad k = 1 \dots N_X \quad (5.6)$$

where $\Phi = d\mathbf{Q}/d\mathcal{X}_k$. The solution can then be substituted back into (5.5) to obtain the gradient. Note that the left hand side of (5.6) is the same as (4.41) for the flow solution. This suggests that the Jacobian-free preconditioned FGMRES method used in the flow solver can be used again to solve (5.6). In the direct method, the objective function does not appear in (5.6), and the same solution Φ applies regardless of the objective function. However, a different linear system must be solved for each design variable. Therefore this method is most suitable when the number of objective functions and constraints is higher than the number of design variables. For most aerodynamic shape optimization problems, where there are hundreds of design variables but usually only one objective function and a few constraints, the direct method is not as efficient.

In the adjoint method, the intermediate problem is defined as:

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right)^T \Psi = \frac{\partial \mathcal{J}}{\partial \mathbf{Q}} \quad (5.7)$$

This is known as the *adjoint equation*, and the vector Ψ is known as the adjoint variable. The adjoint variable is a vector of the Lagrange multipliers for each flow variable. Like the direct method, the adjoint equation can also be solved with preconditioned FGMRES. Again, the left-hand-side of (5.7) is similar to (4.41) for the flow solution, with both matrices differing only by a transpose. In this case, the transpose operator prevents the use of the Jacobian-free approach in FGMRES, and the flow Jacobian $\partial\mathbf{R}/\partial\mathbf{Q}$ matrix must be formed. The matrix is obtained by hand linearization of the residual vector, except at the block interfaces and boundaries, where linearization of SATs is done using complex steps. It should also be noted that the dissipation terms contain non-differentiable functions, and the linearization process assumes that the dissipation coefficients are frozen. Similarly, the right-hand-side term $\partial\mathcal{J}/\partial\mathbf{Q}$ is also hand-derived for each objective function (Appendix E.1). Similar methods for solving the discrete adjoint equation include those of Nielsen and Anderson [140], who also solve the equation using GMRES, with a pseudo time step added, and the adjoint solution is marched in time. In contrast, in Mavriplis [116, 117], the adjoint system is solved using an explicit multigrid method adapted from the flow solver. Similarly, Elliott and Peraire [36, 35, 37] use an explicit relaxation algorithm used in the flow solver.

Substituting Ψ back into (5.5), the gradient expression becomes:

$$\mathcal{G} = \frac{\partial\mathcal{J}}{\partial\mathcal{X}} - \Psi^T \frac{\partial\mathbf{R}}{\partial\mathcal{X}} \quad (5.8)$$

An important advantage of the adjoint method is that size of the linear system in (5.7) depends on the number of state variables, and not the number of design variables. That is, if the partial derivatives $\partial\mathcal{J}/\partial\mathcal{X}$ and $\partial\mathbf{R}/\partial\mathcal{X}$ can be evaluated inexpensively, the cost of a gradient calculation is independent of the number of design variables. These partial derivatives can be evaluated either analytically or approximated using finite differencing.

The approach used in this thesis is to use second-order centred differencing to compute the partial derivatives:

$$\frac{\partial\mathbf{R}}{\partial\mathcal{X}_k} \approx \frac{\mathbf{R}[\mathcal{X} + h\hat{e}_k, \mathbf{Q}] - \mathbf{R}[\mathcal{X} - h\hat{e}_k, \mathbf{Q}]}{2h} \quad (5.9)$$

$$\frac{\partial\mathcal{J}}{\partial\mathcal{X}_k} \approx \frac{\mathcal{J}[\mathcal{X} + h\hat{e}_k, \mathbf{Q}] - \mathcal{J}[\mathcal{X} - h\hat{e}_k, \mathbf{Q}]}{2h} \quad (5.10)$$

The same finite-difference stepsize h is reused from (5.1). Note that evaluating these partial derivative terms does not require additional flow solves. Using this approach, for a problem with N_{GX} geometric design variables, each gradient computation will require $2 \times N_{GX}$ calls to the grid movement algorithm. Finite-differencing is easy to implement, and it can be adapted easily to different grid movement algorithms.

The choice to use finite-differencing to obtain the partial derivative terms is motivated by the fact that the algebraic grid movement algorithm (Chapter 6) is inexpensive. For example,

to generate the new grid in the example in the previous section required 0.2 seconds on 48 processors. For a design case with 170 design variables, the partial derivatives can be evaluated in just over one minute, which is much less than the time required for a flow solution. However, this approach is not suitable with more computationally intensive methods such as spring analogy or linear elasticity, where the time required to generate a new grid is of the order of a flow solution. It should be noted that when using finite-differencing to evaluate the partial derivatives, gradient accuracy may suffer from the same numerical errors as using a finite-difference gradient.

The alternative to finite-differencing is the exact evaluation of the partial derivatives. In general terms, they can be written as:

$$\frac{\partial \mathcal{J}}{\partial \mathcal{X}_k} = \frac{\partial \mathcal{J}}{\partial \mathbf{G}} \underbrace{\frac{\partial \mathbf{G}}{\partial \mathbf{G}_S}}_{\partial \mathcal{J} / \partial \mathbf{G}_S} \frac{\partial \mathbf{G}_S}{\partial \mathcal{X}_k} \quad (5.11)$$

$$\frac{\partial \mathbf{R}}{\partial \mathcal{X}_k} = \frac{\partial \mathbf{R}}{\partial \mathbf{G}} \frac{\partial \mathbf{G}}{\partial \mathbf{G}_S} \frac{\partial \mathbf{G}_S}{\partial \mathcal{X}_k} \quad (5.12)$$

where \mathbf{G} is the volume grid, and \mathbf{G}_S is the surface grid. Exact differentiation involves first differentiating the surface grid \mathbf{G}_S with respect to the design variable \mathcal{X}_i . For B-spline control point design variables, this is done by differentiating the least-squares problem in the geometry parameterization (3.8). However, differentiation with respect to planform variables is considerably more complicated.

For lift-constrained drag minimization problems using inviscid flows, the objective function only depends on flow variables on the body surface \mathbf{Q}_{BODY} . It is therefore possible to differentiate the objective function with respect to the surface grid, i.e. $\partial \mathcal{J} / \partial \mathbf{G}$ by differentiating the grid metrics (4.7) at the surface. However, for general problems that require flow variables away from the surface, such as viscous flow problems, the volume grid \mathbf{G} is differentiated with respect to the surface grid \mathbf{G}_S by differentiating the grid movement algorithm. Finally, the objective function or the residual vector is differentiated with respect to the volume grid, which involves differentiating the grid metrics of the entire computational grid.

As an additional note, exact differentiation with respect to the grid and design variables is necessary with computationally expensive grid movement algorithms, where finite differencing is prohibitively expensive. In the case of the linear elasticity method, the volume grid is expressed as the solution to the problem:

$$r(\mathbf{G}, \mathbf{G}_S, \mathcal{X}) = \mathbf{0} \quad (5.13)$$

where r is the grid residual. A linear system needs to be solved to obtain the grid whenever the design variables are perturbed. The grid equation (5.13) can be considered as an additional constraint for the optimization problem described in Chapter 2. In this case, an additional

adjoint system (“grid adjoint”) must be solved, but no additional calls to the grid movement algorithm are required during gradient evaluation. Examples of the dual adjoint approach can be found in [147, 183]. On 2D problems, it leads to a more complete convergence.

5.1.3 Complex-Step Gradients

The complex-step gradient, previously mentioned in Section 1.2.4, is another method to address the shortcomings of the finite-difference method. It is based on the Taylor series expansion of the objective function at $\mathcal{J}(\mathcal{X})$, with an imaginary perturbation of each design variable (ih). Therefore, for each design variable \mathcal{X}_k ¹:

$$\mathcal{J}(\mathcal{X} + i\hat{\epsilon}_k h) = \mathcal{J}(\mathcal{X}) + i\hat{\epsilon}_k h \frac{\partial \mathcal{J}}{\partial \mathcal{X}_k} - \frac{h^2}{2} \frac{\partial^2 \mathcal{J}}{\partial \mathcal{X}_k^2} - i\mathcal{O}(h^3) + \mathcal{O}(h^4) \quad k = 1 \dots N_X \quad (5.14)$$

Solving for the imaginary part of this expansion yields a second-order approximation to the gradient:

$$\mathcal{G}_k \approx \frac{\text{Im}[\mathcal{J}(\mathcal{X} + i\hat{\epsilon}_k h)]}{h} + \mathcal{O}(h^2) \quad (5.15)$$

Both (5.1) and (5.15) have a truncation error of $\mathcal{O}(h^2)$, but with a complex step, there is no subtraction between two approximations and therefore the gradient is not susceptible to subtractive cancellation errors. Thus a much smaller stepsize h can be used compared to finite differencing (e.g. $h = 10^{-20}$). Also, only one additional objective function evaluation is required for each design variable, compared to two in the second-order finite-difference method. However, in order to take advantage of this, the flow solver must be modified to accept complex numbers. This roughly doubles the memory usage and time required for a flow solve [8]. However, elements of the complex-step method can be incorporated into the adjoint method. For example, Nielsen and Kleb [141] used a complex step to linearize the residual vector to obtain the flow Jacobian. The current work uses the same method for the SATs at block boundaries.

5.2 BFGS Optimizer

When minimizing an unconstrained function using a gradient-based optimizer, the goal is to advance the design variables to a local optimum where the gradient L_2 -norm is zero. Here, the optimization problem is solved using the quasi-Newton optimizer BFGS (Broyden-Fletcher-Goldfarb-Shanno) coupled with a backtracking line search algorithm. A brief description is presented here. Further information can be found in [128, 145].

¹The implicit relationship $\mathbf{Q}(\mathcal{X})$ is dropped for simplicity.

The optimizer begins with an initial shape defined by \mathcal{X}_0 , noting that it is unlikely that the initial design variables represent a minimizer of $\mathcal{J}(\mathcal{X})$. A search direction is sought to quickly reach the minimum. In the Newton method, the search direction \mathcal{P} is based on a quadratic approximation of the objective function at \mathcal{X} , and, defining a new function $m(\mathcal{P})$:

$$\mathcal{J}(\mathcal{X} + \mathcal{P}_n) \approx \mathcal{J}(\mathcal{X}) + \mathcal{P}^T \mathcal{G} + \frac{1}{2} \mathcal{P}^T \mathcal{H} \mathcal{P} \doteq m(\mathcal{P}) \quad (5.16)$$

where $\mathcal{H} = \nabla \nabla^T \mathcal{J}(\mathcal{X})$ is the Hessian matrix. By setting $\nabla m = \mathbf{0}$, it is obvious that the search direction is given by:

$$\mathcal{P} = -\mathcal{H}^{-1} \mathcal{G} \quad (5.17)$$

Since $m(\mathcal{P})$ is only a quadratic approximation, the search direction \mathcal{P} does not necessarily lead to the true minimum in a single step. Therefore this method is then repeated until the optimizer has converged to a local minimum. The Newton direction is guaranteed to be a descent direction as long as the Hessian matrix \mathcal{H} is positive definite. In a quasi-Newton method, the Hessian is approximated. The BFGS approximation of the inverse of the Hessian ($\mathcal{B} \approx \mathcal{H}^{-1}$) is obtained using the following formula:

$$\mathcal{B}_{n+1} = \mathcal{B}_n - \frac{\mathcal{B}_n v_n (\mathcal{B}_n v_n)^T}{v_n^T \mathcal{B}_n v_n} + \frac{\delta_n \delta_n^T}{\delta_n^T v_n} + v_n^T \mathcal{B}_n v_n (r r^T) \quad (5.18)$$

where

$$\begin{aligned} r &= \frac{\delta_n}{\delta_n^T v_n} - \frac{\mathcal{B}_n v_n}{v_n^T \mathcal{B}_n v_n} \\ \delta_n &= \mathcal{X}_{n+1} - \mathcal{X}_n \\ v_n &= \mathcal{G}_{n+1} - \mathcal{G}_n \end{aligned} \quad (5.19)$$

This formula is based on noting that the change in gradient in successive iterations yields information about the Hessian. The approximation becomes increasingly accurate as the optimization progresses. Note that inaccuracies in gradient computations may lead to problems in approximating the inverse Hessian. In such cases, it may be necessary to reset the matrix. During the first iteration where previous gradient information is unavailable, \mathcal{B} is set to the identity matrix, i.e. the search direction is the steepest descent direction.

Once the search direction is found, the design variables are advanced in that direction by a step length β_n :

$$\mathcal{X}_{n+1} = \mathcal{X}_n + \beta_n \mathcal{P}_n \quad (5.20)$$

The Newton method has a natural step length of $\beta = 1$. However, since the objective function is generally not a convex quadratic, this step length may not actually lead to a satisfactory

decrease in the objective function [145], or it may step outside of the design space, where the flow solver does not converge. If this is the case, a line search algorithm is applied along the search direction to find a new step length.

A function is defined for the objective function along the search direction:

$$\phi(\beta) = \mathcal{J}(\mathcal{X}_n + \beta\mathcal{P}_n) \quad (5.21)$$

Its derivative is simply the inner product between the gradient and the search direction:

$$\phi'(\beta) = \mathcal{G}(\mathcal{X} + \beta\mathcal{P})^T \mathcal{P} \quad (5.22)$$

An acceptable step length is one that satisfies the strong Wolfe conditions:

$$\begin{aligned} \phi(\beta) &\leq \phi(0) + \beta c_1 \phi'(0) \\ |\phi'(\beta)| &\leq c_2 |\phi'(0)| \end{aligned} \quad (5.23)$$

The first condition is the *sufficient decrease condition*, which requires that the objective function be decreased by a value proportional to the step length β and a fraction of $\phi'(0)$. The second is the *curvature condition*, which requires that the slope at the new iterate be no more than c_2 times the previous iterate. Typically for quasi-Newton methods, $c_1 = 10^{-4}$ and $c_2 = 0.9$ are used [145]. When the step length does not satisfy the conditions, a line search is performed. A cubic interpolant is constructed for the function $\phi(\beta)$, based on $\phi(\beta_k)$, $\phi(\beta_{k-1})$, $\phi'(\beta_k)$ and $\phi'(\beta_{k-1})$. The minimum of the interpolant is either at its endpoint or in the interior, which can be found by the line-search iteration until (5.23) is satisfied:

$$\beta_{k+1} = \beta_k - (\beta_k - \beta_{k-1}) \left[\frac{\phi'(\beta_k) + r_2 - r_1}{\phi'(\beta_k) - \phi'(\beta_{k-1}) + 2r_2} \right] \quad (5.24)$$

where

$$r_1 = \phi'(\beta_{k-1}) + \phi'(\beta_k) - 3 \frac{\phi(\beta_{k-1}) - \phi(\beta_k)}{\beta_{k-1} - \beta_k} \quad (5.25)$$

$$r_2 = \sqrt{r_1^2 - \phi'(\beta_{k-1})\phi'(\beta_k)} \quad (5.26)$$

In some cases when the Wolfe conditions cannot be satisfied after $k \approx 15$ line search iterations, the line search algorithm is considered *stalled*. In this case, the optimizer automatically resets the search direction to the steepest descent direction, and the approximate Hessian inverse is reset to the identity matrix.

5.3 Design Variable Scaling

Proper scaling of the design variables is crucial to the performance of the optimizer. An optimization problem is considered poorly scaled if changes in one design variable produce

Variable	Measured in	Typical $\Delta\mathcal{X}$
α	degrees	10^{-1}
B-spline pts	z -coord z/c	10^{-5} to 10^{-3}
Λ	radians	10^{-1}
Ω	radians	10^{-2}

Table 5.1: Change in design variable during a typical optimization cycle.

much larger variations in the value of the objective function than changes in other variables. Here, the problem arises when there is a mixture of B-spline control point design variables and planform variables. Using the existing geometry parameterization strategy, B-spline control point design variables are measured by the change in z -coordinate, with the wing root chord scaled to 1.0. Typically, in an optimization run, changes in these design variables are between 10^{-5} and 10^{-3} . Sweep, dihedral, twist angles are measured in radians; in a typical optimization run, the change in design variable value is on the order of 10^{-2} to 10^{-1} . The angle-of-attack design variable is measured in degrees, and the typical change during an optimization cycle is on the order of 10^{-1} to 10^0 . The design variables are summarized in Table 5.1.

Two scaling approaches based on Zingg *et al.* [205] are considered. In the first method, B-spline design variables are scaled by their initial z -coordinates, and the angle-of-attack design variable is scaled by its initial value. Planform variables are not scaled. In the second method, B-spline design variables are scaled by the square root of their initial z -coordinates and the angle-of-attack design variables are scaled by the square root of their initial value. Again, planform variables are not scaled. The design variables are scaled by their initial scaling:

$$\mathcal{X}_s = L^{-1}\mathcal{X} \quad (5.27)$$

where the entries in \mathcal{X}_s are the scaled design variables, those in \mathcal{X} are the unscaled design variables, and the diagonal matrix L contains the initial unscaled design variables. Effects of design variable scaling on the convergence of the optimizer are shown for a single-point optimization case in Section 8.2.

Chapter 6

Grid Movement Algorithm

During the course of an optimization cycle, changes in the geometric design variables affect the surface geometry $\mathbf{G}_S = \mathbf{G}_S(\mathcal{X})$ through the geometry parameterization strategy, which in turn changes the volume grid \mathbf{G} through a grid generation or a grid movement algorithm. Regardless of whether a gradient-based optimizer or gradient-free optimizer such as the genetic algorithm is used, a unique volume grid is necessary for each function evaluation.

For Cartesian grids, which are not body-fitted, a new grid can be generated in a small fraction of the cost of a flow solve [3], and it is possible to create, from scratch, a new grid for both flow solution and evaluation of partial derivatives at each optimization iteration. However, this approach is not without problems. In the case of body-fitted grids (both structured and unstructured), grid generation is considerably more complicated, and is the topic of ongoing research. Most grid generation algorithms have not achieved the speed and level of automation in order to be practical for shape optimization applications. A review of grid generation methods can be found in [180]. However, expensive grid re-generation can be avoided if a suitable grid movement algorithm is employed. Each time the wing surface grid changes, the volume grid is adjusted accordingly. Popular grid movement algorithms include the spring analogy methods [40, 140, 108], linear elasticity method [182, 183, 108, 140], hybrid method [71] and algebraic methods [128, 19, 156, 157].

A fast and robust algebraic grid movement algorithm is implemented. This algebraic method is based on Nemeč [128]. The movement of nodes $k = 2$ to k_{\max} along a grid line are determined by the algebraic equation:

$$\vec{x}_k^{\text{new}} = \vec{x}_k^{\text{old}} + \frac{\Delta \vec{x}_1}{2} [1 + \cos(\pi S_k)] \quad (6.1)$$

where $\Delta \vec{x}_1$ is the displacement of the surface node, and

$$S_k = \frac{\sum_{i=2}^k |\vec{x}_i - \vec{x}_{i-1}|}{\sum_{i=2}^{k_{\max}} |\vec{x}_i - \vec{x}_{i-1}|} \quad (6.2)$$

is the normalized arc-length distance along the grid line.

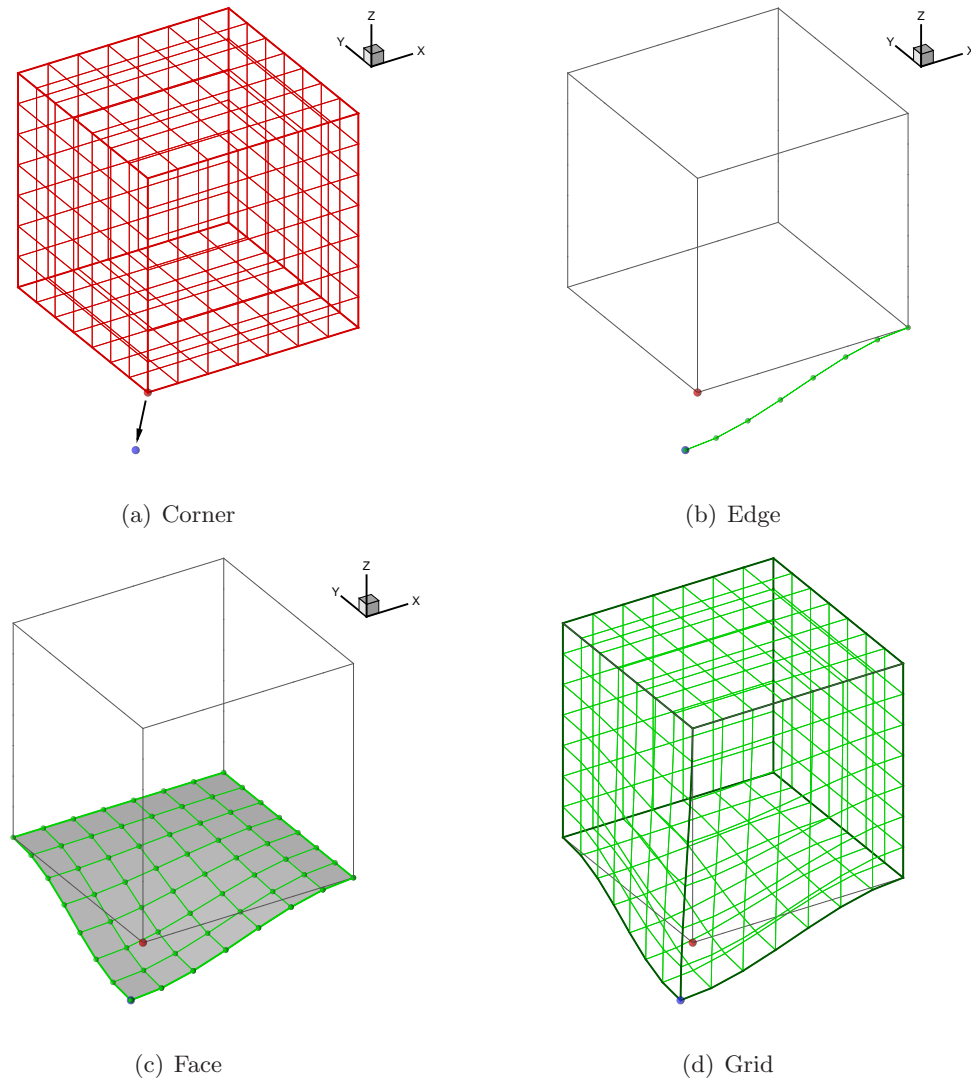
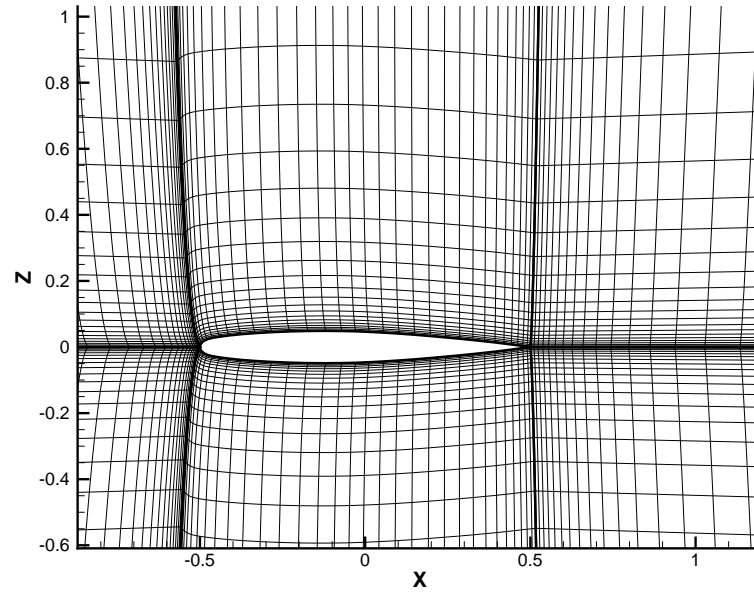


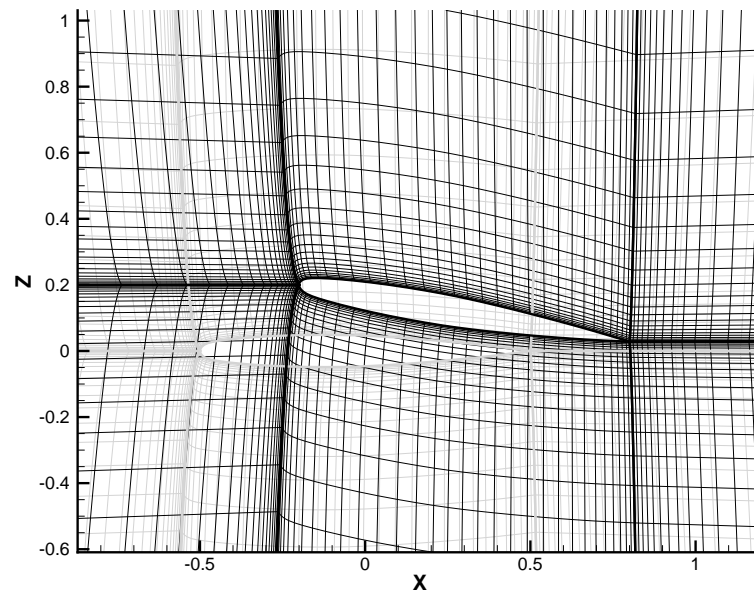
Figure 6.1: Algebraic grid movement algorithm applied to a structured volume grid

For illustration purpose, the algorithm is applied to a coarse $8 \times 8 \times 8$ grid shown in Figure 6.1. In this example, the location of a corner is moved (Figure 6.1a). This may correspond to cases where the corner of a block is on the surface of the geometry. In this example, (6.1) is first applied to the grid line in the ξ direction to generate a new edge (Figure 6.1b). The equation is then applied to all the grid lines in the η direction to generate a new face (Figure 6.1c). Finally, the equation is applied to all grid lines in the ζ direction to generate a new volume grid (Figure 6.1d). If an edge of the block lies on the surface of the wing, the algebraic equation will only be used to generate the face, and then the volume grid, i.e. steps (b)–(d) only. Similarly, if an entire face of the block lies on the surface, the algorithm is only used to generate the volume grid based on the surface grid, i.e. applying steps (c) and (d) only.

An example of the grid movement algorithm is shown in Figure 6.2. In this example, the



(a) ONERA M6 grid at symmetry plane



(b) Perturbed grid at symmetry plane

Figure 6.2: Example of algebraic grid movement algorithm.

root location of an ONERA M6 wing is displaced, and the wing is rotated to a positive angle of attack. The grid at the symmetry plane is shown. Although the grid movement algorithm does not explicitly guarantee the new grid to be of good quality, this method has been found to be effective for most aerodynamic shape optimization applications as long as the block boundary is sufficiently far from the body surface.

Chapter 7

Validation

7.1 Overview

Flow solver and optimization results are obtained on the High Performance Advanced Computing Facility (HPACF) at University of Toronto's Institute for Aerospace Studies (UTIAS). The HPACF is a distributed-memory Beowulf-class cluster. The cluster uses Intel Itanium 2 processors with a CPU speed of 1.5GHz. Each node consists of 4 processors, with 8GB of shared memory per node. The nodes are connected by a high-bandwidth low-latency Myrinet network. Communication between processors is done using message-passing interface (MPI) [55] standard. Each component of the Newton-Krylov algorithm are individually validated, and the results are presented in this chapter.

7.2 Flow Solver Accuracy

The accuracy of the flow solver is validated against experimental measurements on the ONERA M6 wing by Charpin and Schmitt [169]. It should be noted that validation against experimental results is difficult, since an Euler flow solver does not account for viscous and turbulent effects, such as flow separation, and interactions between shocks and the boundary layer. However, the flow solver should be able to accurately capture the locations of the shocks.

Validation cases are presented for three transonic operating conditions:

Case 1: $M = 0.6990$, $\alpha = 3.06^\circ$

Case 2: $M = 0.8350$, $\alpha = 4.08^\circ$

Case 3: $M = 0.8831$, $\alpha = 4.07^\circ$

In each test case, flow solutions are computed using three combinations of dissipation constants:

$$\begin{aligned} \text{A: } & \kappa_2 = 1.0, \quad \kappa_4 = 0.02 \\ \text{B: } & \kappa_2 = 0.0, \quad \kappa_4 = 0.10 \\ \text{C: } & \kappa_2 = 2.0, \quad \kappa_4 = 0.02 \end{aligned}$$

The first two sets of dissipation constants (A and B) are used in all flow validation cases. The third set of dissipation constants (C) reflects a more aggressive approach to removing the oscillations around the shock. This is only used in Cases 2 and 3.

The first set of dissipation terms (A) are values suggested by Nichols [139]. For the second set of terms (B), second-difference dissipation is turned off while a higher fourth-difference constant is used. This set of constants is used in the design examples presented in Chapter 8. In constructing the flow Jacobian matrix ($\partial\mathbf{R}/\partial\mathbf{Q}$) for adjoint gradient calculation, the dissipation terms are considered frozen, and the pressure switch (4.26 and 4.25) is not linearized. Therefore, second-difference dissipation must be turned off in order to obtain accurate objective function gradients. This is further discussed in Section 7.4. Without second-difference dissipation, the higher κ_4 value is found to be necessary for the flow solver to converge. Ideally this issue can be addressed by linearization of the pressure switch.

Flow solutions are obtained on a 12-block H-H topology grid with 1,008,000 nodes. The grid is shown in Figure 7.1. This grid is generated using an elliptical grid generator GEM3D [62]. In this grid, the wing's semi-span ($b/2$) is scaled to unity (wing root chord $c_{\text{root}} = 0.674$, wing planform/reference area $S = 0.563$). On both top and bottom surfaces, there are 60 nodes in the spanwise direction and 40 nodes in the chordwise directions. Off-wall spacing is 10^{-3} , and the far-field boundary is at a minimum of 22.0 semi-spans away from any point on the wing surface. The flow solution is considered converged when $\|\mathbf{R}(\mathbf{Q})\|_2 < 10^{-10}$. Using 12 processors, i.e. one block per processor, the flow solution takes an average 50 minutes to converge.

The first validation case ($M = 0.699$, $\alpha = 3.06^\circ$) is the lowest Mach number for which experimental results are available. For this case, the flow solver convergence histories are plotted against iterations and CPU time in Figure 7.2. On the top surface, there is a mild shock near the leading edge, while the lower surface is shock free. Pressure coefficients at six spanwise locations

$$\frac{2y}{b} = [0.20, 0.40, 0.65, 0.80, 0.90, 0.96]$$

are plotted against experimental results in Figure 7.3. In both results, the flow solver accurately predicts the location and strength of the shock. Without second difference dissipation, the flow solver slightly over-estimates the strength of the shock, and changing the dissipation constants did not significantly affect the final solution. However, in shock-free regions, such as on the lower surface, and downstream of the shock on the top surface, there is excellent agreement

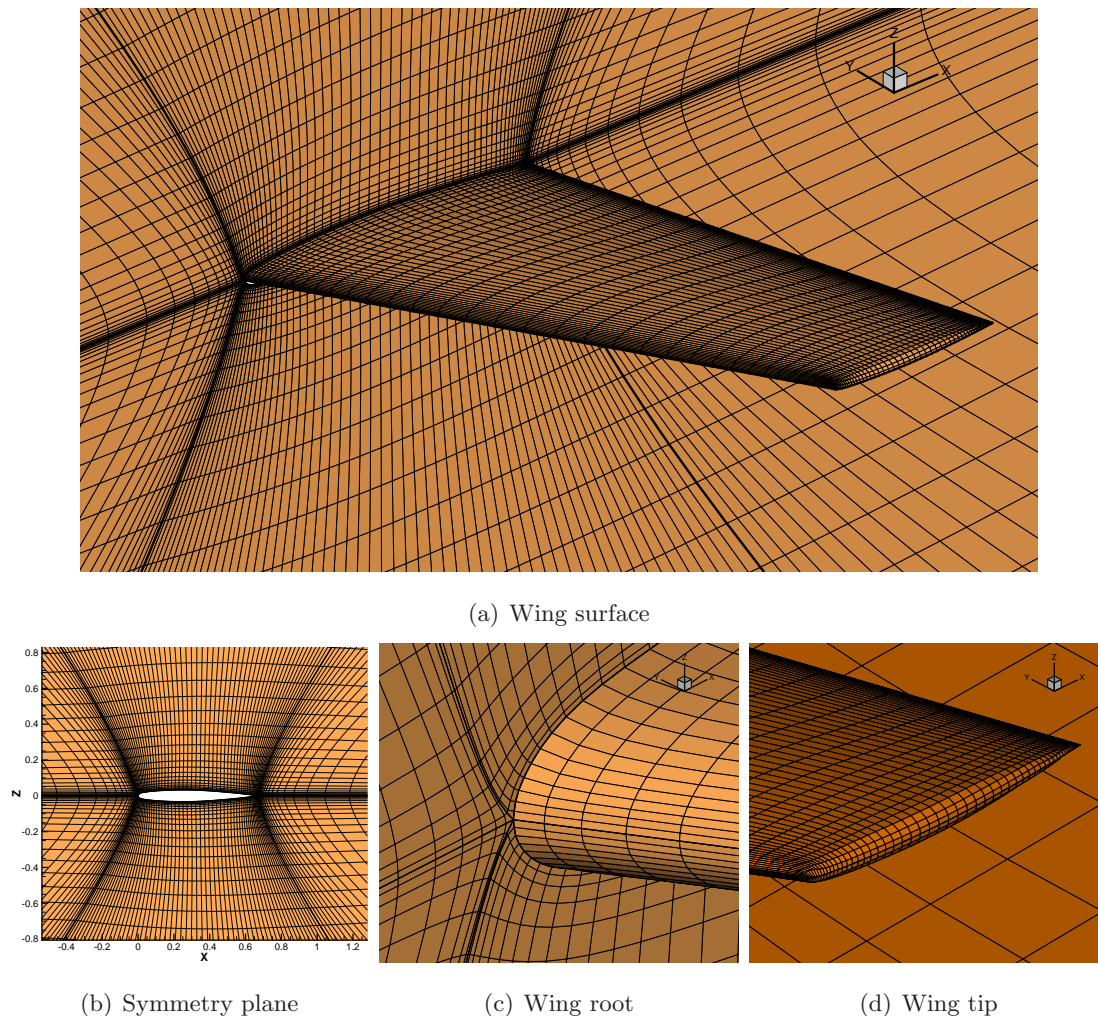


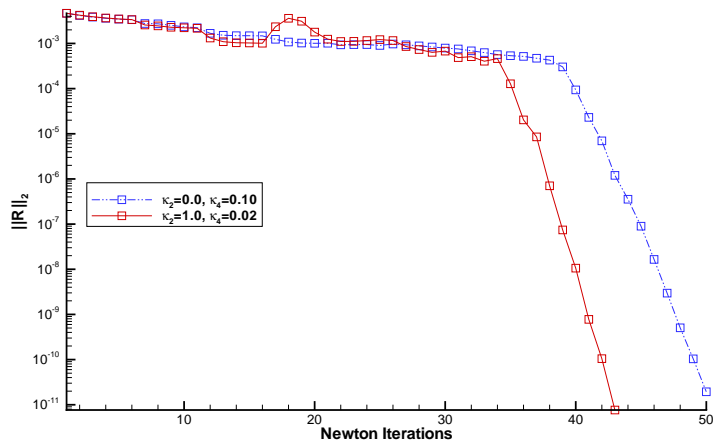
Figure 7.1: 12-block H-H grid around an ONERA M6 wing

Dissipation Terms	C_L	C_D
$\kappa_2 = 1.0, \kappa_4 = 0.02$	0.2352	0.006274
$\kappa_2 = 0.0, \kappa_4 = 0.10$	0.2392	0.006548

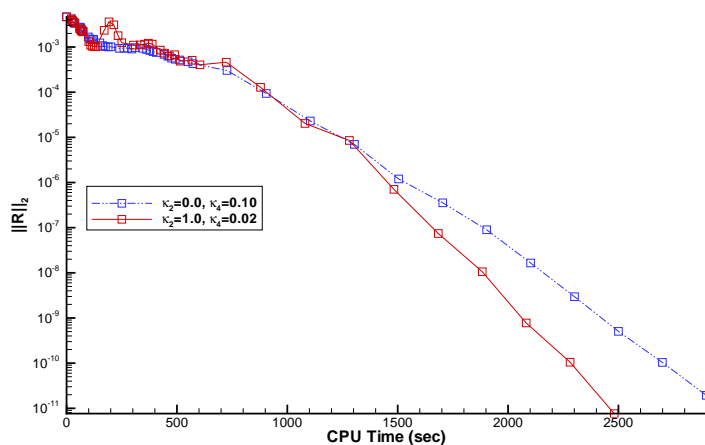
Table 7.1: Lift and drag coefficients for the ONERA M6 wing at $M = 0.699$, $\alpha = 3.06^\circ$

between numerical and experimental results. Lift and drag coefficients from the converged solutions are shown in Table 7.1. In this case, changing the dissipation terms changed the C_D value by 4%. This difference should be taken into account when doing an optimization cycle.

In the second validation case ($M = 0.835$, $\alpha = 4.08^\circ$), there are two shocks on the top surface of the wing. The bottom surface of the wing is shock free. A surface Mach number contour plot for this operating condition is shown in Figure 7.4. Pressure coefficients at six spanwise locations are plotted against experimental results in Figure 7.5. Similar to the previous case, in



(a) Newton Iterations



(b) CPU Time

Figure 7.2: Convergence history for flow solver validation case at $M = 0.699$, $\alpha = 3.06^\circ$

flow regions where second-difference dissipation is turned off—in this case the bottom surface of the wing—there is excellent agreement between experimental and numerical results. On the top surface, the general locations and strengths of the shocks are well-captured by the flow solver, although again, the strength of the shocks near the wing root is over-estimated in the numerical solutions. There is also an oscillation in the flow solution near the shocks when κ_2 values of 0.0 or 1.0 are used. To fully eliminate the oscillation requires $\kappa_2 = 2.0$. The convergence history is shown in Figure 7.6. Lift and drag coefficients from the converged solutions are shown in Table 7.2.

The operating conditions for the third case ($M = 0.8831$, $\alpha = 4.07^\circ$) are the closest to the operating conditions used in the optimization examples. In this case, the shape of the shocks on the top surface are similar to Case 2, but the shocks are stronger. The bottom surface again remains shock free. All three combinations of dissipation constants are used in this test case,

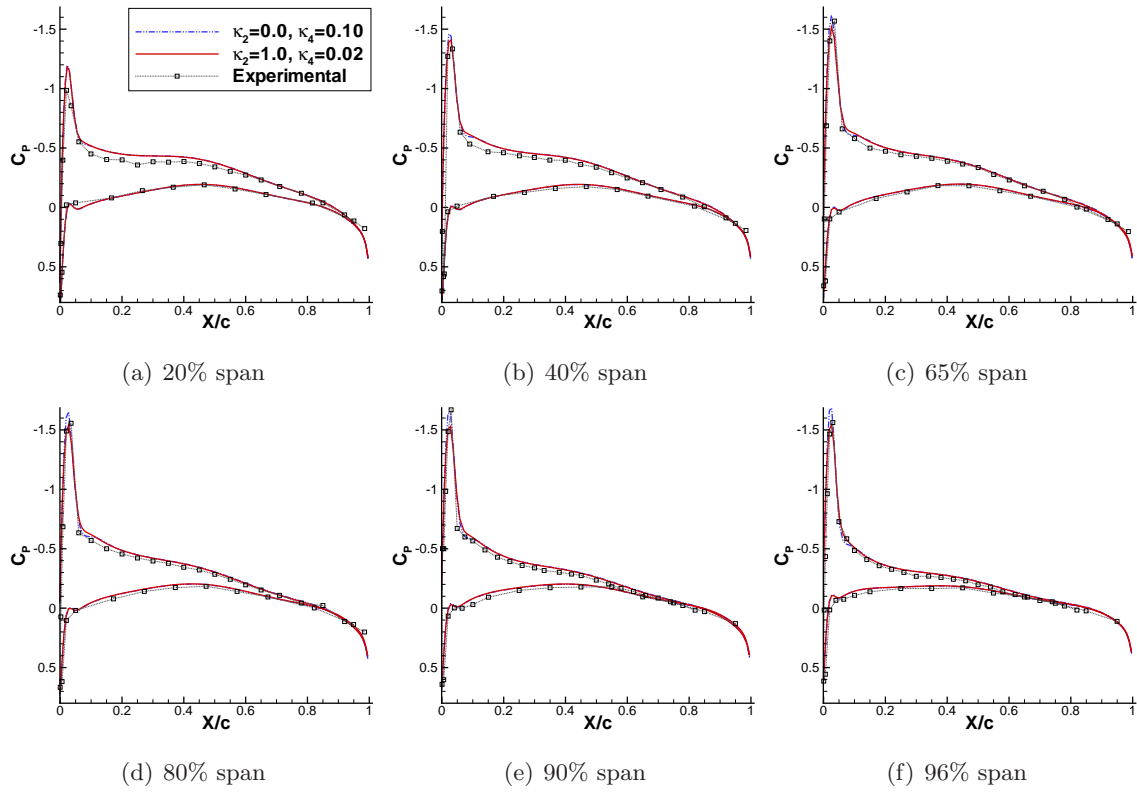


Figure 7.3: Pressure coefficients on the ONERA M6 wing at $M = 0.669$, $\alpha = 3.06$

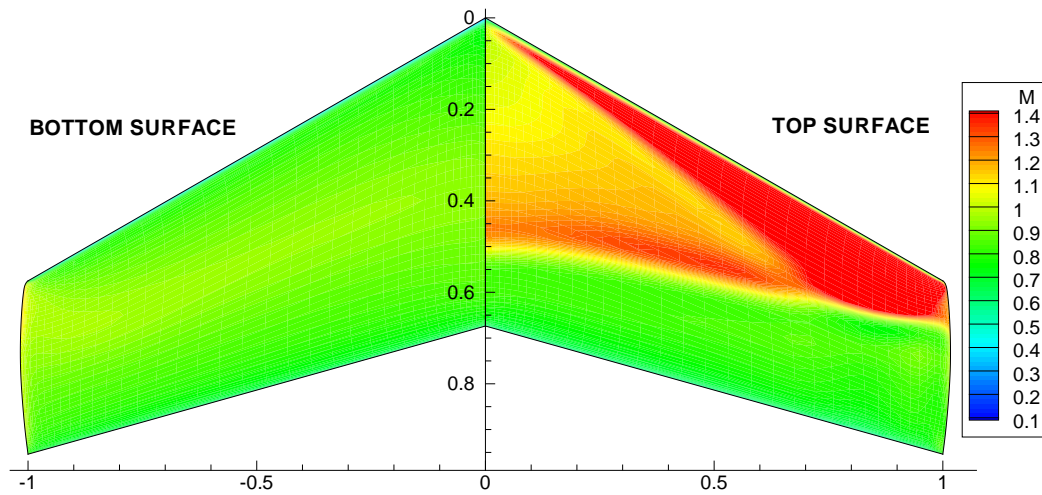


Figure 7.4: Mach number contours for the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08^\circ$

Dissipation Terms	C_L	C_D
$\kappa_2 = 1.0, \kappa_4 = 0.02$	0.3814	0.02108
$\kappa_2 = 0.0, \kappa_4 = 0.10$	0.3906	0.02163
$\kappa_2 = 2.0, \kappa_4 = 0.02$	0.3756	0.02086

Table 7.2: Lift and drag coefficients for the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08^\circ$

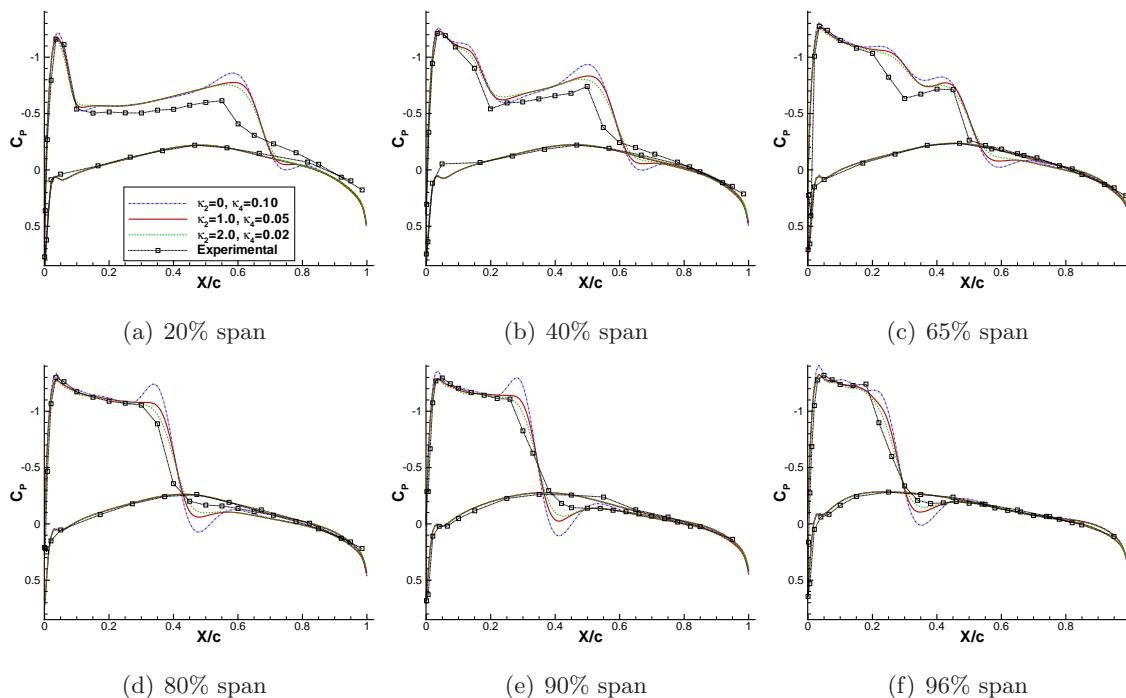
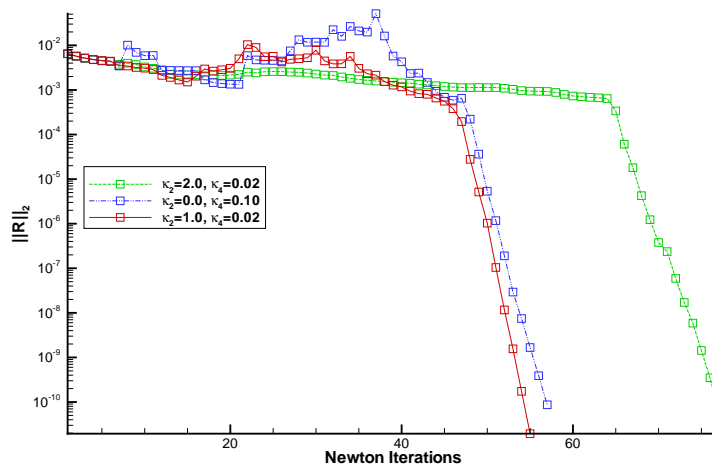


Figure 7.5: Pressure coefficients on the ONERA M6 wing at $M = 0.835$, $\alpha = 4.08$

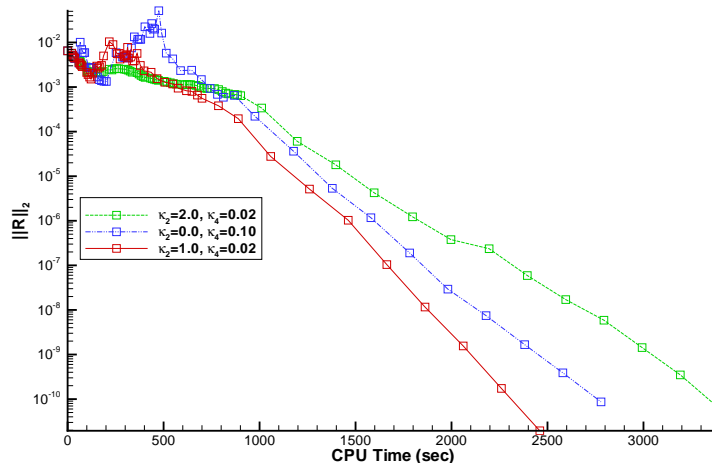
but with $\kappa_2 = 2.0$, the flow solution did not converge, as the flow solver encountered negative densities after a Newton update. The convergence histories are shown in Figure 7.7.

Surface Mach number contours are shown in Figure 7.8. Pressure coefficients at six span-wise locations are plotted against experimental results in Figure 7.9. In this case, it is noted that numerical results are very different from published experimental results, in terms of the placement and the strength of the shocks. The discrepancy may be attributed to the fact that viscous and turbulence effects are not modelled in the flow solver. The increased viscous effects at this high Mach number may lead to flow features that the flow solver is unable to model. These may include interactions between the shock and boundary layer, or shock-induced flow separation.

From the point of view of an optimization case, it is noted that the shock-free regions of the flow field—such as the bottom surface of the wing—are well modelled by the flow solver with $\kappa_2 = 0.0$. This means that although the computed flow field may have inaccuracies in the beginning of the optimization cycle, the final optimized geometry will remain optimal, as long as the final geometry is shock-free or has weak shocks. Lift and drag coefficients from the converged solutions are shown in Table 7.3.



(a) Iterations



(b) Time

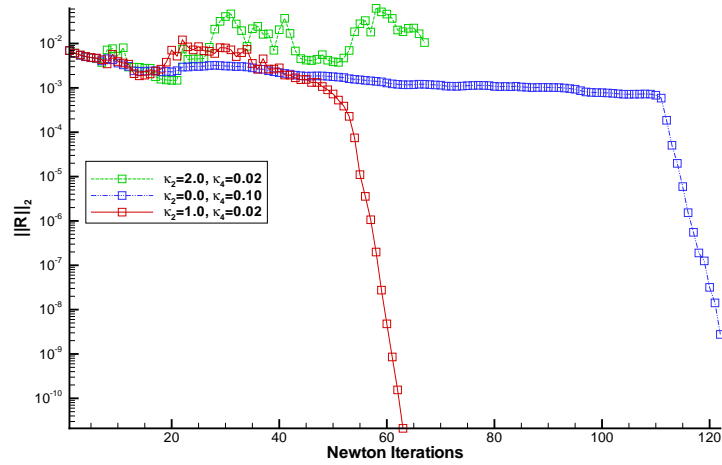
Figure 7.6: Convergence history for flow solver validation case at $M = 0.835$, $\alpha = 4.08^\circ$

Dissipation Terms	C_L	C_D
$\kappa_2 = 1.0$, $\kappa_4 = 0.02$	0.4470	0.03857
$\kappa_2 = 0.0$, $\kappa_4 = 0.10$	0.4587	0.03969

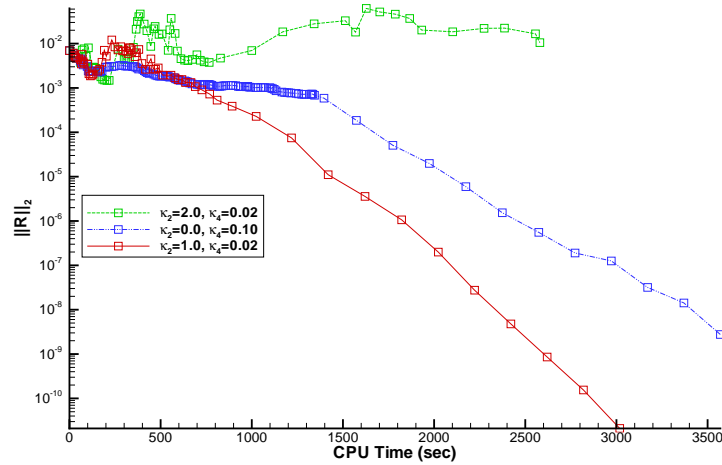
Table 7.3: Lift and drag coefficients for the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07^\circ$

7.3 Comparison of Grid Movement Algorithms

In this section, the algebraic grid movement algorithm described in Chapter 6 is validated against the linear elasticity method of Truong *et al.* [182, 183]. The test cases are done using a 146,004-node grid around an ONERA M6 wing, as shown in Figure 7.10. Each surface is parameterized using a B-spline surface with 10 control points in the chordwise direction, and 15 in the spanwise direction, as shown in Figure 7.11.



(a) Iterations



(b) CPU Time

Figure 7.7: Convergence history for flow solver validation case at $M = 0.835$, $\alpha = 4.08^\circ$

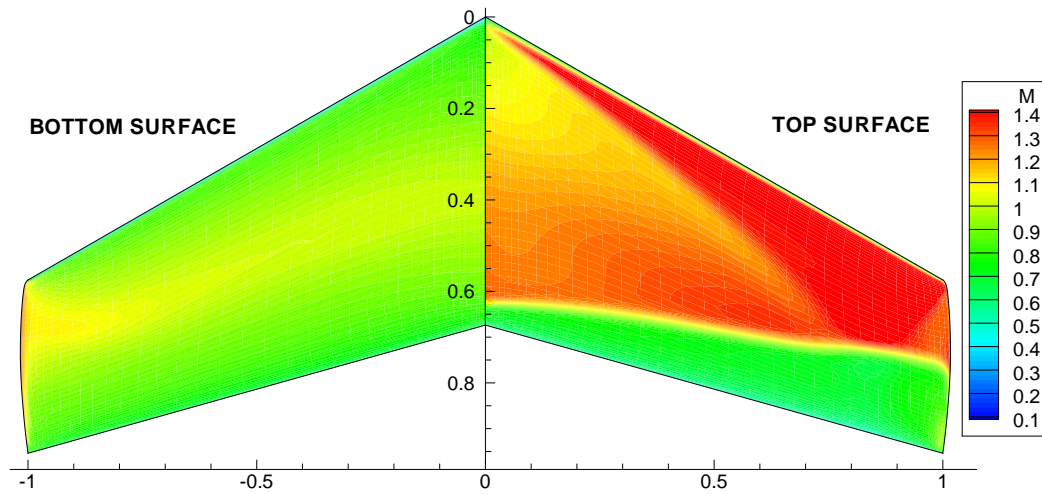


Figure 7.8: Mach number contours for the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07^\circ$

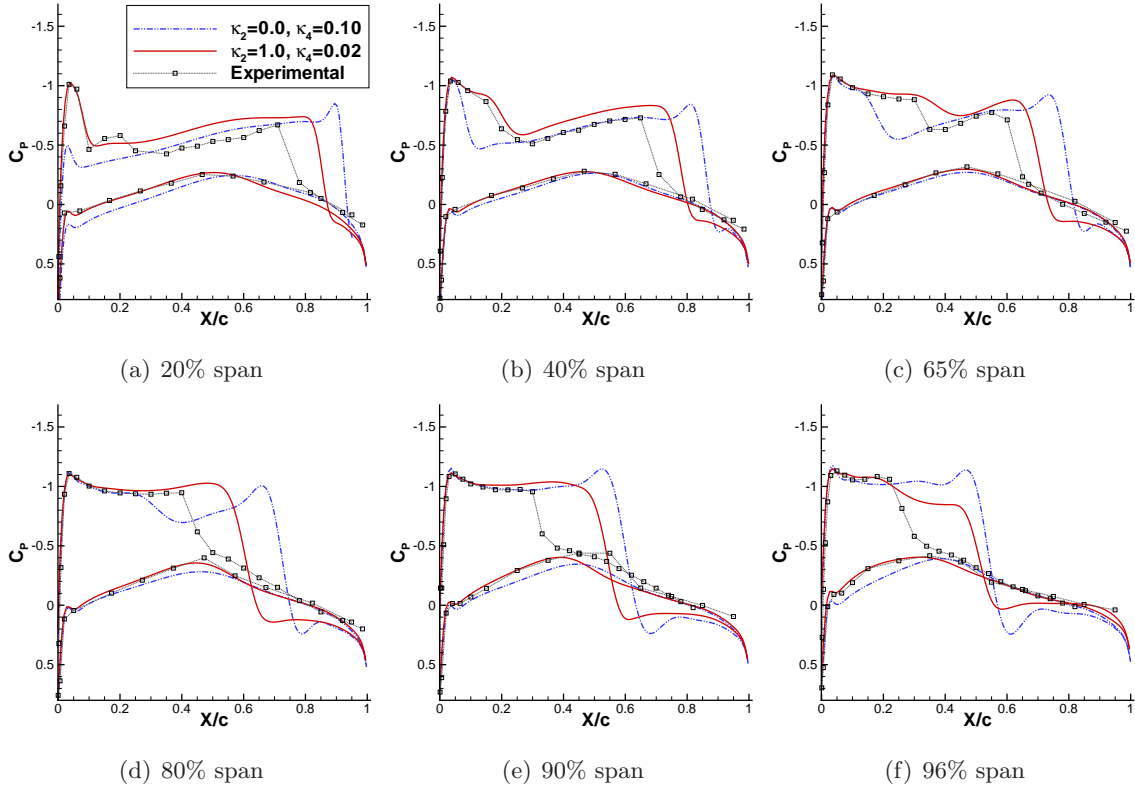
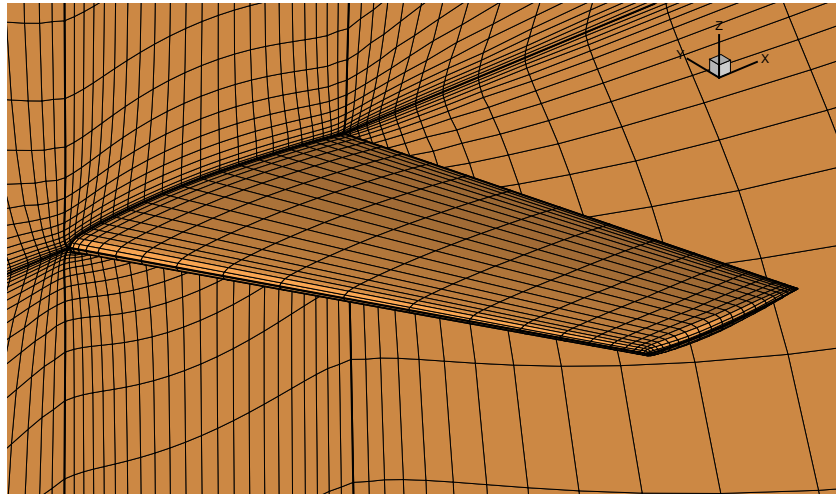


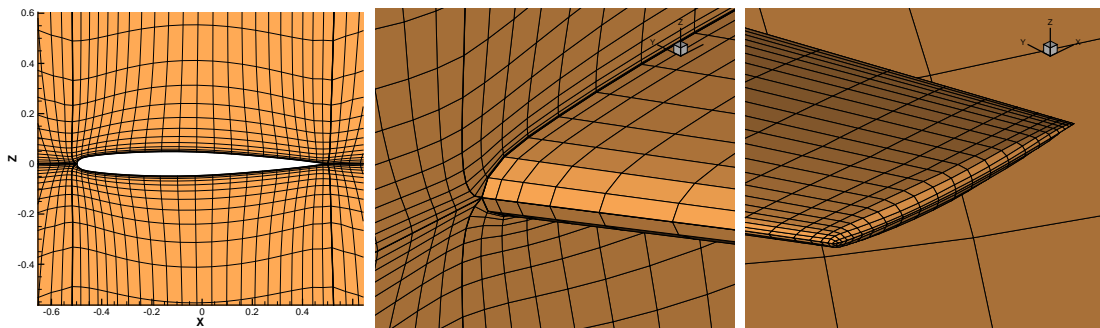
Figure 7.9: Pressure coefficients on the ONERA M6 wing at $M = 0.8831$, $\alpha = 4.07$

In the first validation case, a new surface grid is generated by moving the location of the wing root, the angle of attack of the wing with respect to the far-field boundary, as well as the sweep angle, using the geometry parameterization strategy in Section 3. A new volume grid is then generated using both methods. Figure 7.12 shows different views of the resulting grids. Both algorithms are able to maintain a good quality grid, but the linear elasticity method is better able to maintain grid line orthogonality near the surface, as seen by examining the grid lines at the block boundaries. In terms of the performance, using a single processor, the algebraic method required about 3.0 seconds to complete, while the linear elasticity method took about 919.0 seconds to solve the stiffness matrix inexactly using the conjugate gradient (CG) Krylov method.

For the second validation case, a new wing surface is generated with an additional leading-edge sweep angle of $\Delta\Lambda_{LE} = 5^\circ$ added at the wing root. This amount of change in the geometry is typical in the optimization cases presented in this thesis. New grids are then produced using two algorithms. Using the new grids, flow solutions are computed for $0.60 < M_\infty < 0.89$ at a fixed angle of attack $\alpha = 3.06^\circ$. The resulting lift and drag coefficients are compared in Figure 7.13. It shows good agreement in C_L and C_D values for both algorithms. The algebraic method required 3.0 seconds to complete, while the linear elasticity method took about 800



(a) Wing surface



(b) Symmetry plane

(c) Wing root

(d) Wing tip

Figure 7.10: Coarse ONERA M6 grid used

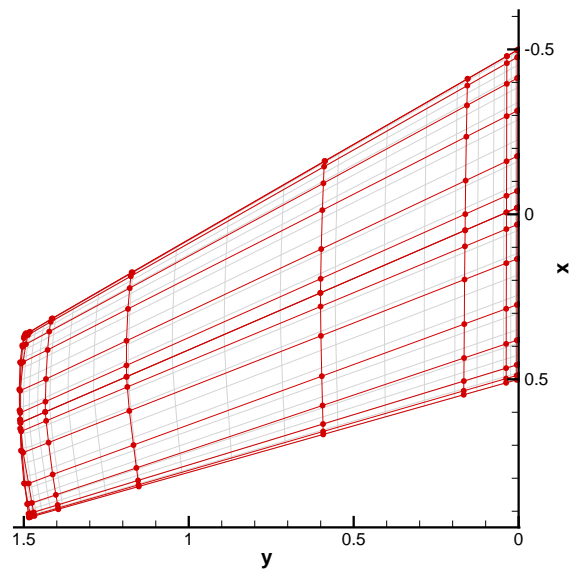
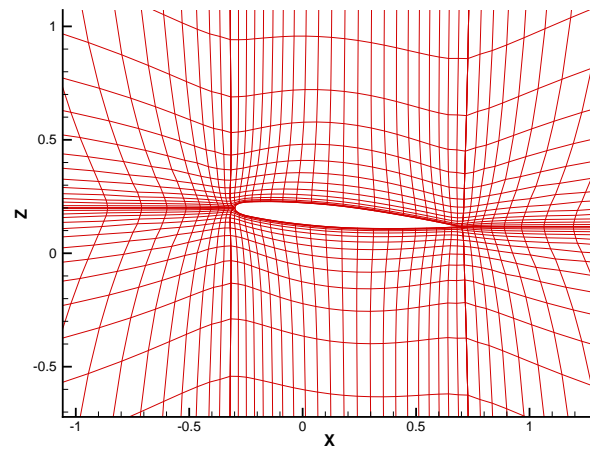
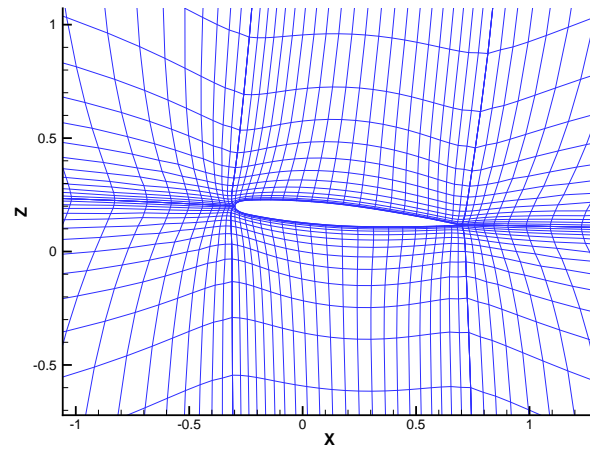


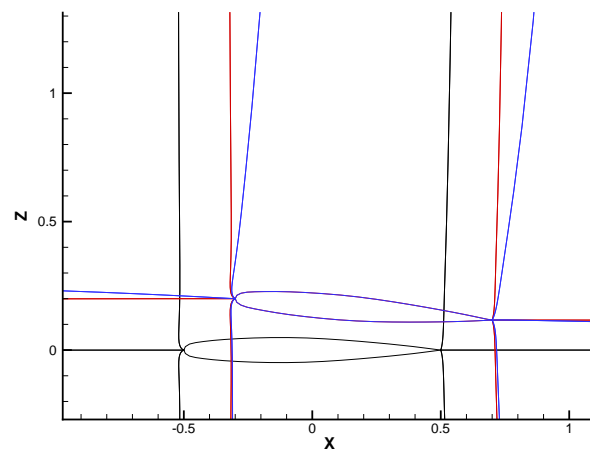
Figure 7.11: B-spline parameterization of the coarse ONERA M6 grid



(a) Algebraic



(b) Linear elasticity



(c) Block boundaries

— Algebraic — Linear elasticity

Figure 7.12: New grids generated with the two algorithms

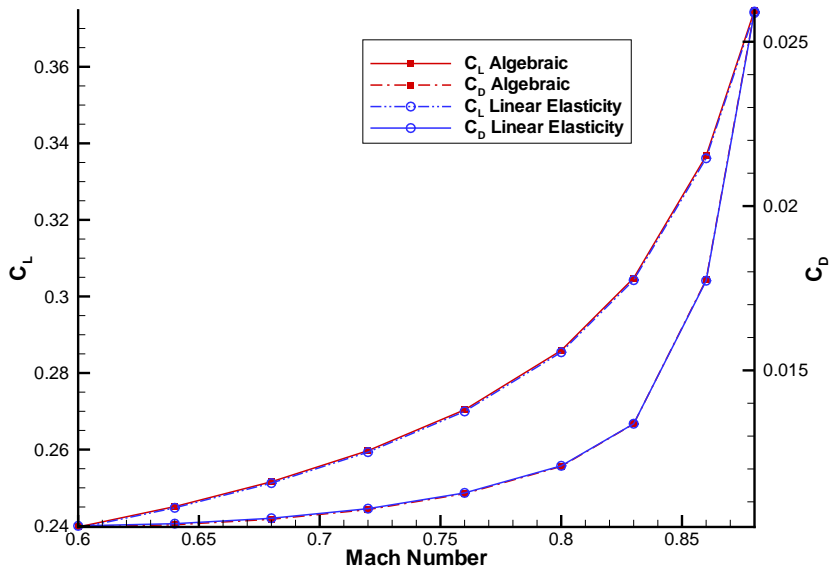


Figure 7.13: Comparison of grid movement methods, with $\Delta\Lambda = +5^\circ$

seconds on a single processor.

7.4 Gradient Accuracy Evaluation

In this validation test, discrete adjoint gradients are compared to finite-difference gradients to establish their accuracy. Two cases that are considered to be representative of the range of optimization problems the optimizer may encounter are tested:

1. A subsonic lift-to-drag maximization problem
2. A transonic lift-constrained drag minimization problem

In each of these test cases, the flow solution is converged to 10 orders of magnitude reduction in residual. For the adjoint gradient calculations, the linear residual of the adjoint equations is converged by eight orders of magnitude. Finite-difference gradients are obtained using second-order centred differencing, with a step size of $h = 10^{-5} \times \mathcal{X}_i$ for each design variable i . The same step size is also used when using second-order centred differencing to evaluate partial derivatives ($\partial\mathcal{J}/\partial\mathcal{X}$ and $\partial\mathbf{R}/\partial\mathcal{X}$) in the adjoint method for the B-spline, sweep and Mach number design variables (5.10). In general, a step size of $10^{-4} \times \mathcal{X}_i > h > 10^{-8} \times \mathcal{X}_i$ is accurate. Similar values are found by Nemeć [128] and Wong [196] for 2D problems. For angle-of-attack design variables, $\partial\mathcal{J}/\partial\alpha$ is hand derived (Appendix E.2), while $\partial\mathbf{R}/\partial\alpha$ is computed using the complex step method (Appendix E.3). Gradient accuracy tests are done on a 12-block H-H topology grid around an ONERA M6 grid with 430,000 nodes.

Design Var.	Adjoint	FD	Relative Diff. (%)
α	0.00658046	0.00658046	0.00001
B-Spline 1	0.01213377	0.01213290	0.0072
B-Spline 2	0.15142742	0.15142677	0.00043
B-Spline 3	-0.00217632	-0.00217520	0.0518
Λ_{LE}	0.00239485	0.00239575	0.0376

Table 7.4: Gradients for the subsonic L/D maximization case with $\kappa_2 = 0.0$

Design Var.	Adjoint	FD	Relative Diff. (%)
α	-234.120307	-234.120306	0.0000001
M	-1289.998655	-1289.998563	0.0000071
B-Spline 1	-79.9814306	-79.9814300	0.0000004
B-Spline 2	-53.248897	-53.249054	0.0002949
B-Spline 3	-256.041027	-256.041210	0.0000716
Λ_{LE}	-3.2832813	-3.2832809	0.0000128

Table 7.5: Gradients for transonic lift-constrained drag minimization case with $\kappa_2 = 0.0$

Two cases are used to validate the accuracy of the adjoint gradient. The first is a subsonic case at a free-stream Mach number of $M = 0.3$ and an angle of attack of 2.0° . The minimization of C_D/C_L (2.8) is used as the objective function. The second case is a transonic case with a free-stream Mach number of $M = 0.84$ at an angle of attack of 2.0° . The lift-constrained drag minimization objective function (2.5) is used, with the following targets and weights for lift and drag:

$$\begin{aligned} C_L^* &= 0.307 & \omega_L &= 100.0 \\ C_D^* &= 0.00877 & \omega_D &= 1.0 \end{aligned}$$

The design variables include change in sweep angle at the leading edge ($\Delta\Lambda_{LE}$) from the original configuration, three arbitrarily selected B-spline control points and the angle of attack. In addition, Mach number is also considered in the transonic case. In order to obtain five or more significant figures in gradient accuracy, it is necessary to reduce the linear residual in the adjoint system by at least eight orders of magnitude.

The finite-difference and adjoint gradients are first computed with no second-difference dissipation ($\kappa_2 = 0.0$). The results are shown in Table 7.4 for the subsonic case (with $\kappa_4 = 0.02$) and in Table 7.5 for the transonic case (with $\kappa_4 = 0.10$). It shows excellent agreement between the two methods. Relative differences between finite-difference and adjoint gradients are less than 0.0002% for the third B-spline control points, and several orders magnitude smaller for other design variables. Note that since the partial derivatives are evaluated with finite

Design Var.	Adjoint	FD	Relative Diff. (%)
α	0.0065 2033	0.0065 0845	0.182
B-Spline 1	0.01189 0762	0.01189 3416	0.022
B-Spline 2	0.149 028037	0.149 976489	0.636
B-Spline 3	-0.0020 90154	-0.0020 88376	0.085
Λ_{LE}	0.002 596489	0.002 653282	2.365

Table 7.6: Gradients for the subsonic L/D maximization case with $\kappa_2 = 1.0$

Design Var.	Adjoint	FD	Relative Diff. (%)
α	-238. 790372	-236. 633832	0.9031102
M	-1 190.044217	-1 287.863731	8.2198218
B-Spline 1	- 95.454369	- 93.312395	2.2439767
B-Spline 2	-49. 316700	-49. 800163	0.9803240
B-Spline 3	- 280.230102	- 277.522871	0.9660742
Λ_{LE}	-3. 509072	-3. 497541	0.3286186

Table 7.7: Gradients for transonic lift-constrained drag minimization case with $\kappa_2 = 1.0$

differencing, it may still suffer the same numerical issues as finite-difference gradients.

The two test cases are then repeated with second-difference dissipation turned on ($\kappa_2 = 1.0$, $\kappa_4 = 0.02$). The results are shown in Table 7.6 for the subsonic case, and in Table 7.7 for the transonic case. In the subsonic case, turning on second-difference dissipation did not significantly change the adjoint or the finite-difference gradient. Also, there is still good agreement between finite-difference and adjoint gradients with $\kappa_2 = 1.0$. The disagreements in the gradient in this case is higher than with $\kappa_2 = 0.0$. In the transonic case, however, the accuracy of the adjoint gradient suffers more with second-difference dissipation turned on.

This disagreement can be attributed to the way the Jacobian matrix is computed. The pressure switch that is used to turn on the second-difference dissipation near shocks ((4.25) and (4.26)) is a non-differentiable function. In the hand derivation of the Jacobian terms, the dissipation terms are assumed to be frozen. In a shock-free subsonic case where second-difference dissipation terms are usually turned off throughout the entire flow field, turning the κ_2 term on and off has little impact on the accuracy of the adjoint gradient. However, in the transonic case with shocks present, second-difference dissipation is used, and the impact of κ_2 on gradient accuracy is significant. In the optimization cases presented in Chapter 8, only fourth-difference dissipation is used.

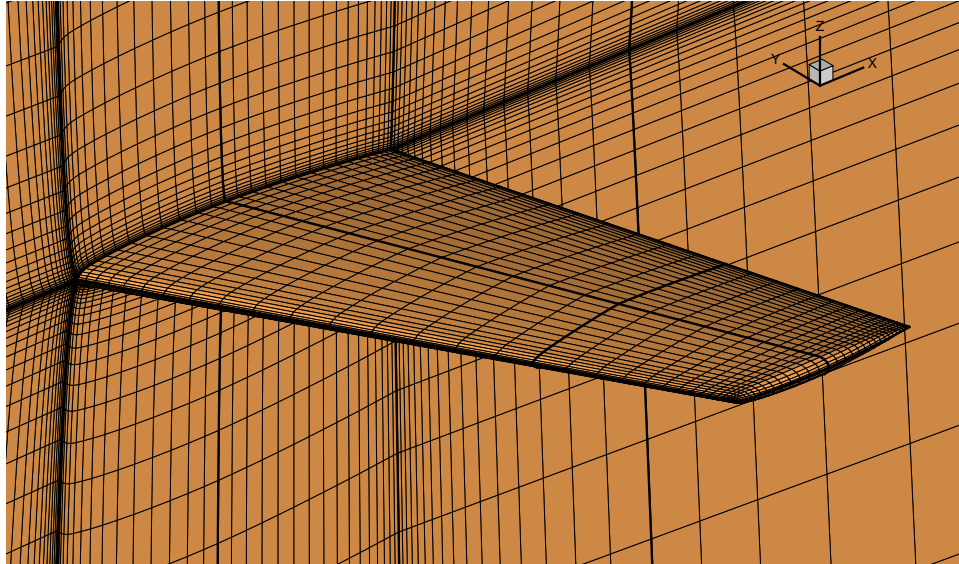
7.5 Inverse Design

The goal of inverse design is to find a wing geometry that best matches a target pressure distribution (2.9). In this validation case, the target pressure is based on the surface pressure of the ONERA M6 wing at $M_\infty = 0.80$ and $\alpha = 3.0^\circ$. The flow solution for the target pressure is computed using the dissipation constants $\kappa_2 = 0.0$ and $\kappa_4 = 0.10$. The grid used is a 48-block H-H topology grid with 457,776 nodes around the ONERA M6 wing. The grid is shown in Figure 7.14.

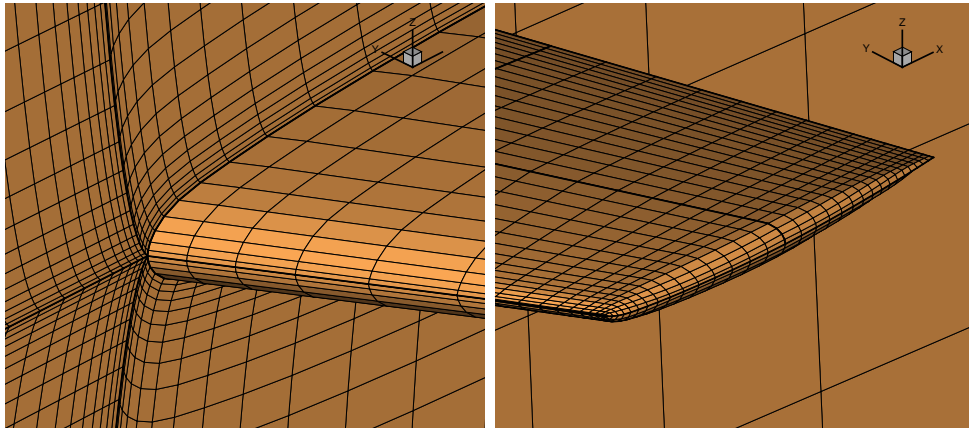
Both the top and bottom surfaces of the wing are parameterized with four cubic B-spline surfaces, as described in Section 3.1. Each B-spline surface has five control points in the chordwise direction and seven in the spanwise direction, shown in Figure 7.15. The design variables are the z -coordinates of 132 B-spline control points, highlighted in blue in Figure 7.15. They include every control point except near the leading edge and trailing edge. In addition, the angle of attack (α) is also a design variable. The B-spline control point design variables are perturbed from the ONERA M6 wing to generate the initial geometry. Thus the initial geometry has the same planform as the ONERA M6, but with a different cross-section. The initial wing is shown at the wing root (symmetry plane) in Figure 7.16. The initial angle of attack was also perturbed to $\alpha_0 = 3.5^\circ$.

The inverse design validation case is performed with the design variable scaling methods described in Section 5.3. The convergence histories for all three cases are shown in Figure 7.17. In the case where the design variable is scaled by the square root of the initial value (Scaling=2, red-line), convergence rate is the fastest. The objective function decreased by five orders of magnitude, and the gradient L_2 -norm has also decreased by three orders of magnitude after about 100 iterations. This is excellent convergence behaviour for such a large number of design variables. In the cases where design variables are scaled by their initial values (Scaling=1, green line) and with no scaling applied to the design variables (Scaling=0, blue line), the optimizer converged the slowest.

The wing section geometries are shown with Scaling=2 at six spanwise locations in Figure 7.18 after 5, 10, 20 and 30 iterations. After about 30 iterations, the geometry is indistinguishable from the ONERA M6 wing.

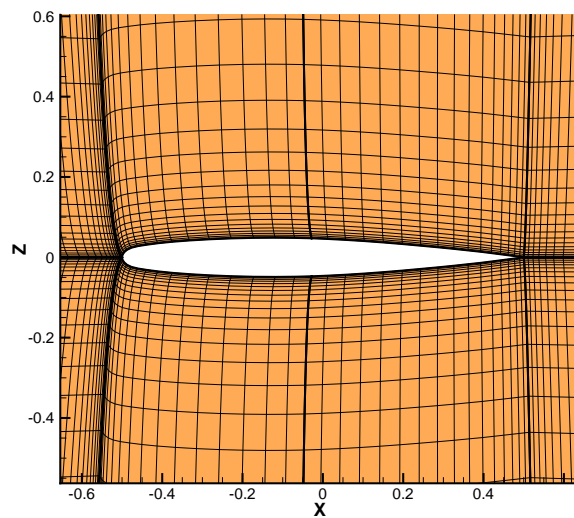


(a) Wing surface



(b) Wing Root

(c) Wing Tip



(d) Symmetry plane

Figure 7.14: 48-block H-H grid around a ONERA M6 wing

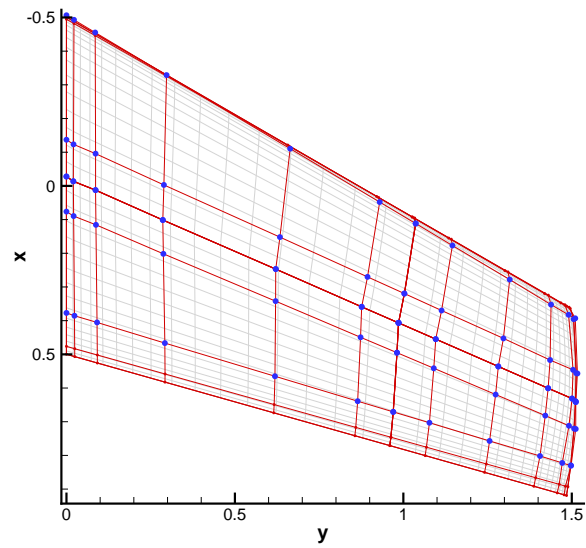


Figure 7.15: B-spline parameterization of the ONERA M6 wing surface

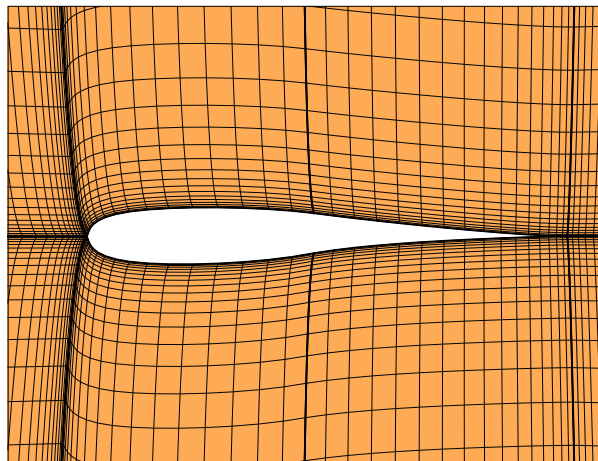


Figure 7.16: Symmetry plane of initial grid for inverse design

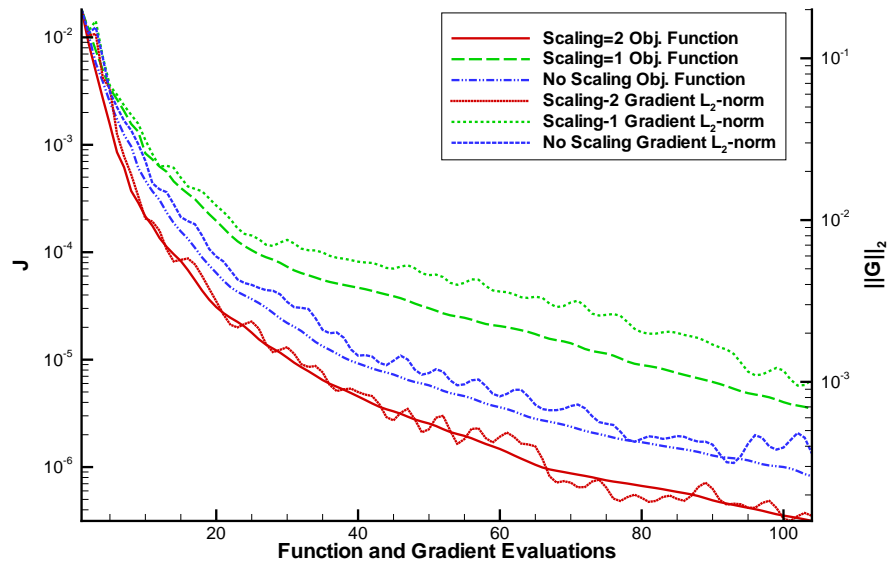


Figure 7.17: Convergence histories for the inverse design validation case

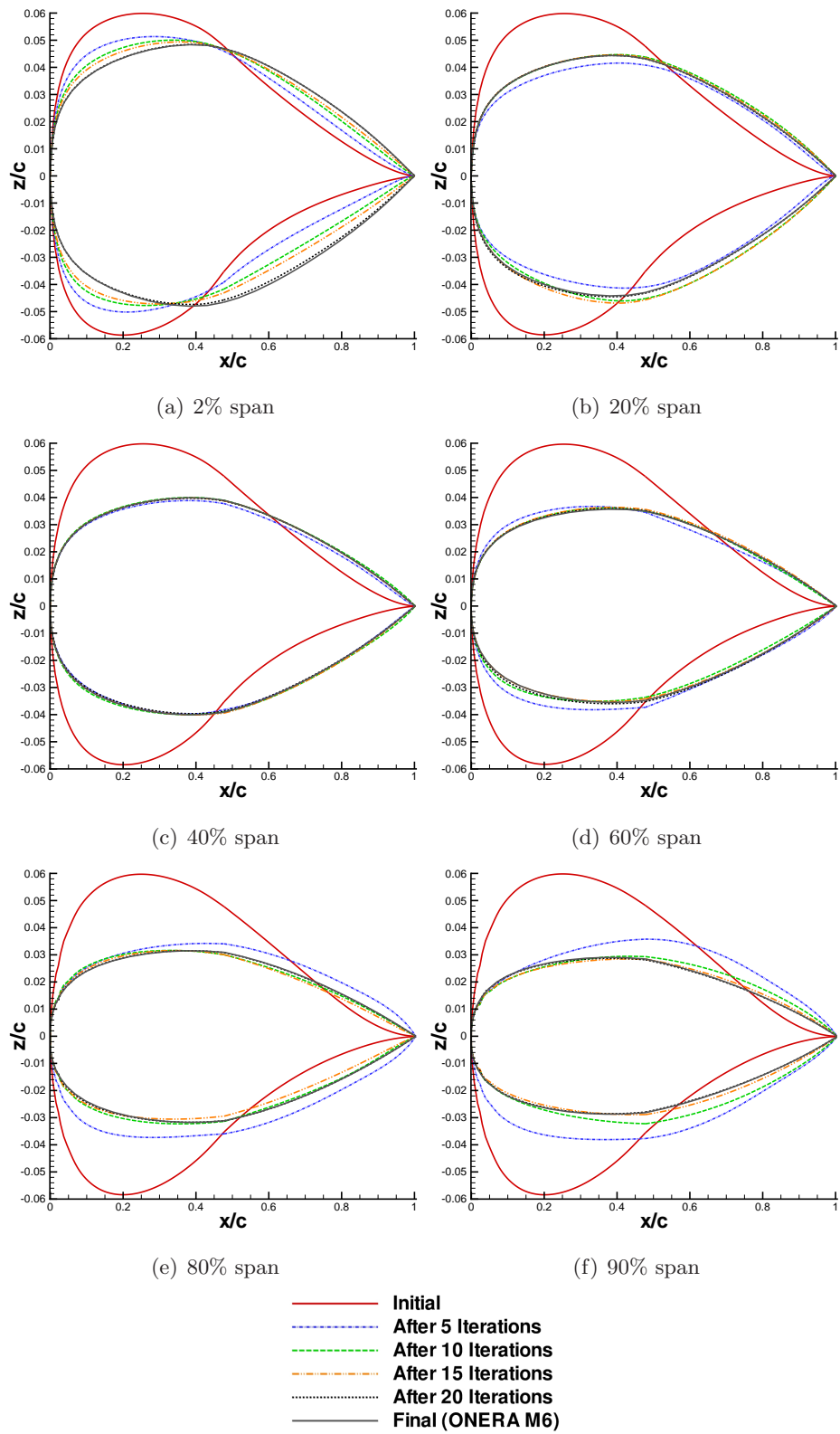


Figure 7.18: Comparison of initial and final wing cross-sections for inverse design

Chapter 8

Design Examples

The Newton-Krylov algorithm described in Chapters 2–6 is used in the design of wings in a series of representative cases using the lift-constrained drag minimization objective function (2.5). In Section 8.1, results are presented for a single-point optimization of an ONERA M6 wing. The single-point results are expanded to a multi-point case with three design operating conditions in Section 8.3. The effects of design variable scaling on convergence is shown in Section 8.2. In these three examples, a volume constraint is added to preserve the volume enclosed by the wing at each iteration. Single-point optimization results are also presented for a geometry based on the DPW-W1 wing [185] in Section 8.4. A thickness constraint is added to maintain a minimum wing thickness. Finally, results are presented for a study to determine the effect of starting geometry on the final optimized solution in Section 8.5, in particular, whether a wing with forward sweep represent a locally optimum design. In this final case, both volume and thickness constraints are used.

For all design examples, second-difference dissipation is turned off ($\kappa_2 = 0.0$) to ensure an accurate discrete gradient, while fourth-difference dissipation coefficient of $\kappa_4 = 0.10$ is used. Recall from Section 7.2, in the absence of second-difference dissipation, the higher κ_4 value is needed at higher transonic Mach numbers—such as those encountered in the design examples—in order for the flow solver to converge. Flow solutions are converged by 10 orders of magnitude. For the adjoint system, the linear residual is converged by eight orders of magnitude.

8.1 Single-Point Lift-Constrained Drag Minimization

A single-point optimization case is presented for drag minimization of an ONERA M6 wing at transonic speed. The goal of this optimization case is to minimize drag at $M = 0.90$ while maintaining the lift of the original ONERA M6 wing. The wing initially operates at $\alpha = 2.50^\circ$.

Parameter	Value
Root chord (c_{root})	1.0
Semi-span ($b/2$)	1.5
Taper ratio (γ)	0.56
Aspect ratio (AR)	3.8
Leading-edge sweep angle (Λ_{LE})	30.0°
Trailing-edge sweep angle (Λ_{TE})	15.8°

Table 8.1: Planform parameters for the ONERA M6 wing

At this operating condition, the lift and drag coefficients are:

$$C_L = 0.307$$

$$C_D = 0.02804$$

The lift-constrained drag minimization objective function (2.5) is used for this case, with the following targets and weights in the lift and drag terms¹:

$$C_L^* = 0.308 \quad \omega_L = 100.0$$

$$C_D^* = 0.00746 \quad \omega_D = 1.0$$

The high ω_L value is found to be necessary to maintain the lift coefficient. Based on linear lifting-line theory (2.6), the minimum induced drag is $C_D = 0.00795$. The target drag is therefore set to a value that is purposely unattainable.

The volume grid around the ONERA M6 wing is the same H-H topology grid used in the inverse design validation case (Section 7.5). The grid has 457,776 nodes. There are 33 nodes in each of the chordwise and spanwise directions on the top and bottom surfaces of the wing. The chord at the wing root is normalized to $c_{\text{root}} = 1.0$. Off-wall spacing is $0.003c_{\text{root}}$, and far-field boundaries are at least $22c_{\text{root}}$ away from any point on the wing surface. The ONERA M6 wing grid is shown in Figure 7.14. Planform parameters are shown in Table 8.1.

The wing surface geometry is parameterized using cubic B-spline surfaces. Each of the top and bottom surfaces of the wing is divided into eight surface patches, each fitted with a 5×5 cubic B-spline surface. Overall, there are nine control points in the chordwise direction, and 17 in the spanwise direction. The B-spline control surfaces are shown for the top surface in Figure 8.1. For this optimization case, z -coordinates of 165 B-spline control points are used as design variables. Control points that are used as design variables are highlighted in blue in Figure 8.1. This essentially includes every control point, except near the leading edge and

¹Using the formulation in (2.5), the targets are set to $\mathcal{L}^* = C_L^* \cdot S = 0.351$ and $\mathcal{D}^* = C_D^* \cdot S = 0.0085$ using a planform area of $S = 1.14$. For this design example, the planform area does not change from one iteration to another, this formulation is equivalent.

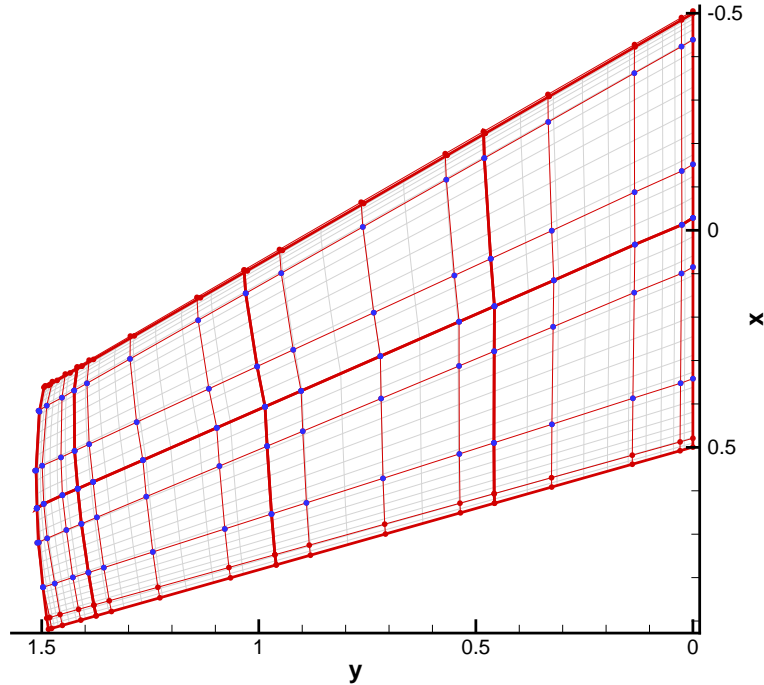


Figure 8.1: B-spline surface parameterization of the ONERA M6 wing for single- and multi-point optimization

trailing edge, where the control points are fixed to prevent wing surface cross-over. Additionally, the change in leading-edge sweep angle at the wing root ($\Delta\Lambda_{LE_1}$), changes in two sweep angles near the wing tip ($\Delta\Lambda_{LE_{14}}$, $\Delta\Lambda_{LE_{15}}$), geometric twist (Ω) and angle of attack (α) are also considered, for a total of 170 design variables. Note that as sweep angle changes, the wing is sheared along the chordwise direction, and the planform area S is maintained during the course of the optimization. B-spline control point design variables are scaled by the square root of their initial z -coordinates (second scaling method described in Section 5.3). Further discussions on the effect of design variable scaling are presented in the next section. Finally, a volume constraint (3.18) is added to maintain the wing's volume (with $v_f = 0.0$). A penalty weight of $\omega_V = 50.0$ is used.

During the optimization cycle, the flow solver requires an average of 8.2 minutes to reduce the residual by 10 orders of magnitude, using 48 processors. The adjoint solver takes an average of 2.2 minutes to reduce the residual by eight orders of magnitude. That the flow solution takes about four times longer to solve than the adjoint solution is consistent with previous experience with a 2D adjoint solver [132].

The optimization convergence history is shown in Figure 8.2. After 231 iterations, which took about 44 hours, the objective function has reduced by more than two orders of magnitude and the gradient L_2 -norm has reduced by three orders of magnitude. At this point, the lift and

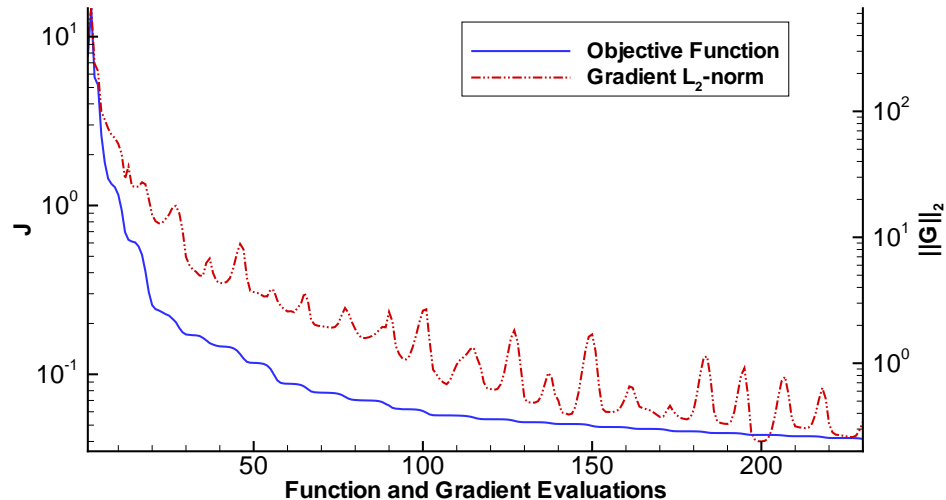


Figure 8.2: Convergence history for the single-point optimization of the ONERA M6 wing

drag terms are:

$$C_L = 0.307$$

$$C_D = 0.008936$$

This represents a 68% improvement in drag with no loss in lift. Note that the final drag is within 15% of the minimal induced drag predicted by the lifting-line theory. It is not known whether this is caused by numerical errors, or whether improved flexibility in the geometry parameterization is needed. The spanwise lift distribution for the ONERA M6 and the optimized geometries are compared to the elliptical lift distribution in Figure 8.3. It shows that the lift distribution of the optimized geometry more closely matches the elliptical lift distribution, indicating that the induced drag has been reduced.

In the final geometry, the sweep angle Λ_{LE} at the wing root is increased from 30.0° to 43.7° . The angle of attack increased slightly from 2.50° to 2.62° . A negative twist angle of $\Omega = -6.07^\circ$ is added, creating a washout at the wing tip. Surface Mach contours of the final geometry are compared to the ONERA M6 wing in Figure 8.4. It shows that the optimizer has eliminated the shocks on the upper surface,² thus minimizing the wave drag component. The shock reduction can be seen further in the surface pressure coefficient plots at six spanwise stations, as shown in Figure 8.5. The optimized wing's cross-sections are also compared to the ONERA M6 wing in Figure 8.5. Drag reductions after 20, 50 and 100 iterations for this design example are shown in Table 8.2. The Mach contours on the wing's top surface after 20 and 50 iterations are shown in Figure 8.6. In this design example, most of the drag reduction occurs in the first 20 iterations, which takes about 4 hours using 48 processors.

²Note that near the wing root, there are initially two shocks on the top surface.

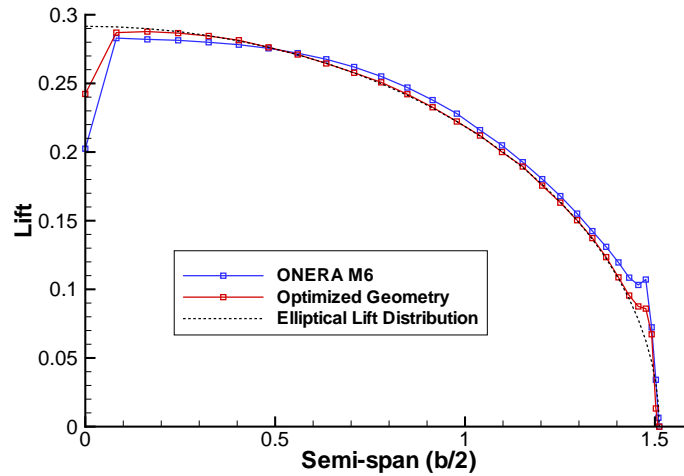


Figure 8.3: Spanwise lift distribution for the single-point optimization

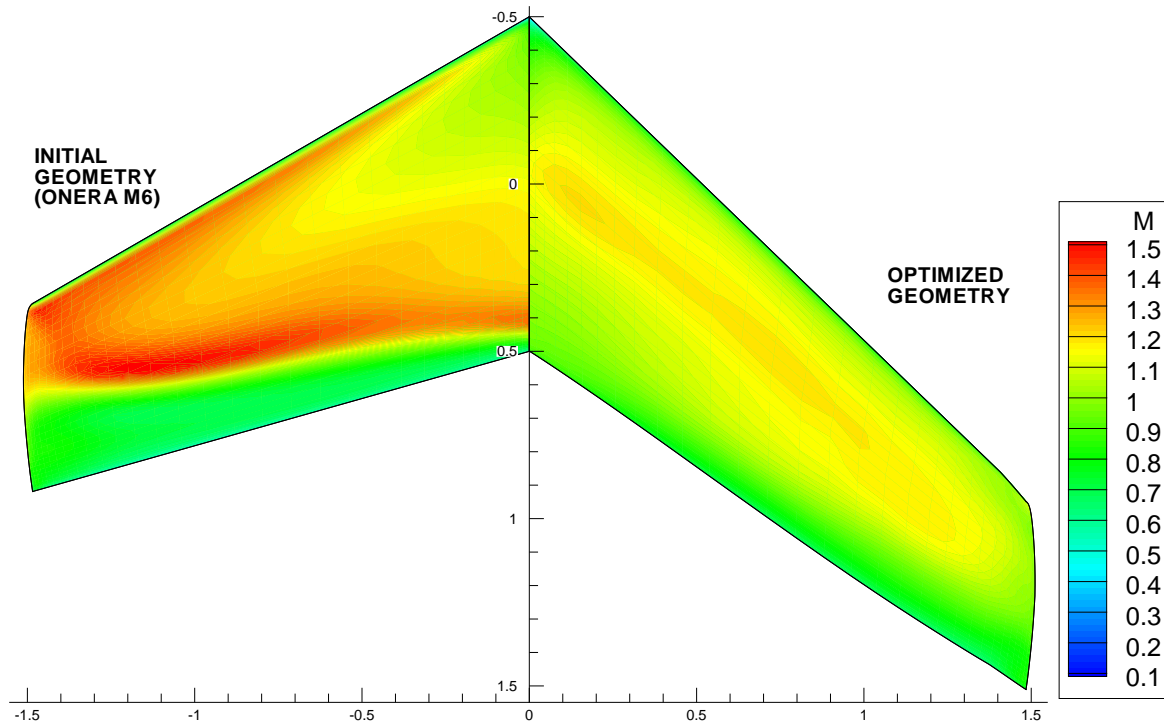


Figure 8.4: Comparison of Mach contours on the wing top surface for the single-point optimization case

During this optimization example, it is noted that the line search algorithm is rarely used, in that the full Newton step $\beta = 1.0$ is usually sufficient. In rare cases when the line search algorithm is called, the strong Wolfe conditions can generally be satisfied within two to three line-search steps.

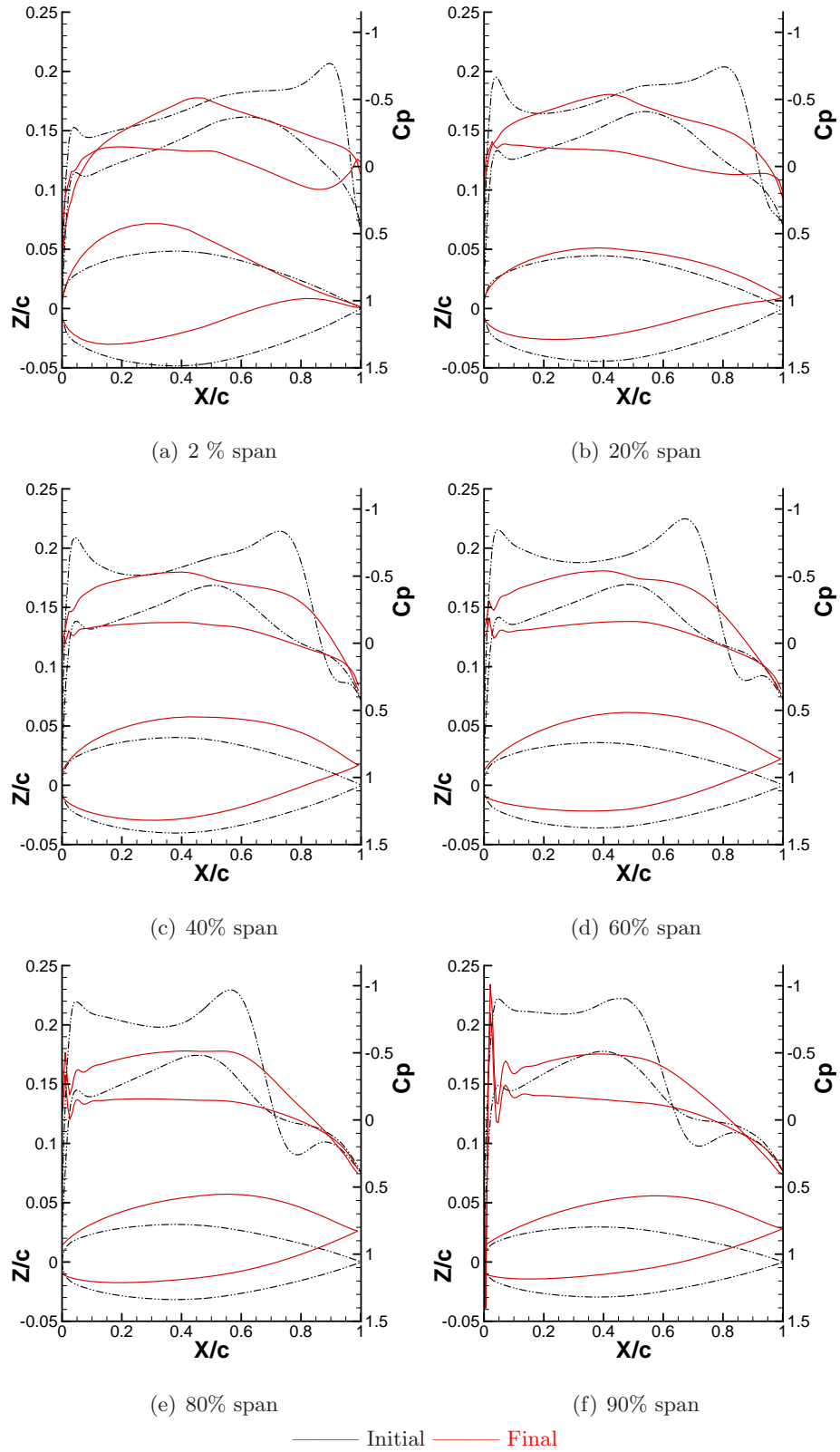


Figure 8.5: Surface pressure coefficient plots and wing sections at six spanwise stations for the single-point optimization case

Iteration	C_D	Drag Reduction
1	0.02804	—
20	0.01121	60%
50	0.01002	64%
100	0.00928	67%

Table 8.2: Drag reduction vs. iteration count for single-point optimization

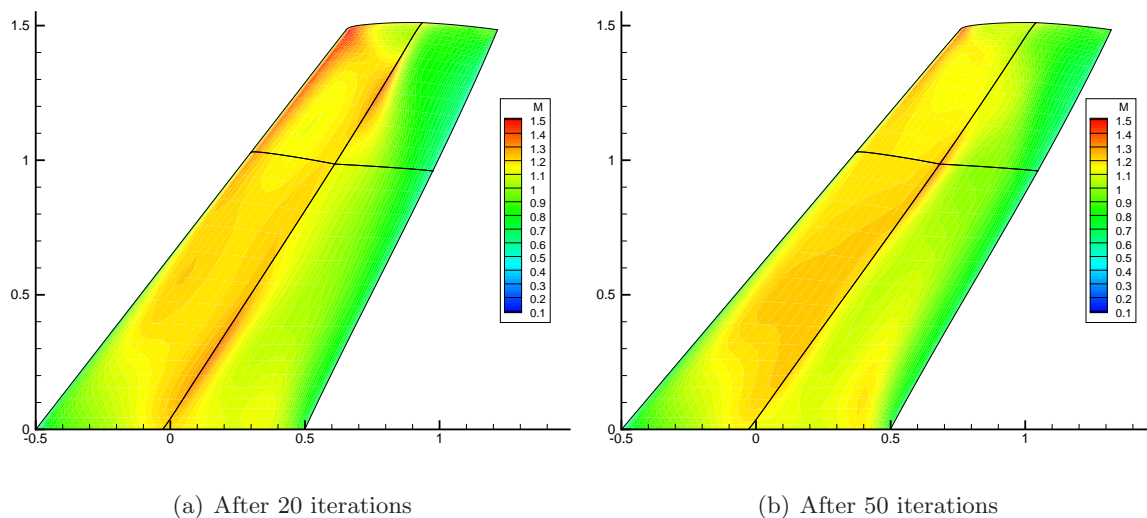


Figure 8.6: Mach contours after 20 and 50 optimization iterations

To demonstrate the robustness of the optimizer in design problems over a range of Mach numbers, additional single-point optimization results are presented in Appendix F. As a final note, in this design example, the increase in the sweep angle is particularly noteworthy. One obvious implication is its effect on structural weight. The aerodynamic shape optimization does not take into account whether an aerodynamically optimum design will have any additional weight, and also ignores the structural loading on the wing. Additional structural weight, which appears in the denominator in the logarithmic term in (2.4), will certainly affect the range of an aircraft. The aerodynamic optimization also does not take into account trim and longitudinal stability of the aircraft. So while the final geometry may be aerodynamically optimal based on this particular objective function, additional constraints and considerations must be given to ensure that the optimizer returns a practical design. Finally, the single-point optimization does not address the fact that a wing must be optimized for a range of flight conditions in the flight envelope. This last issue is addressed in Section 8.3.

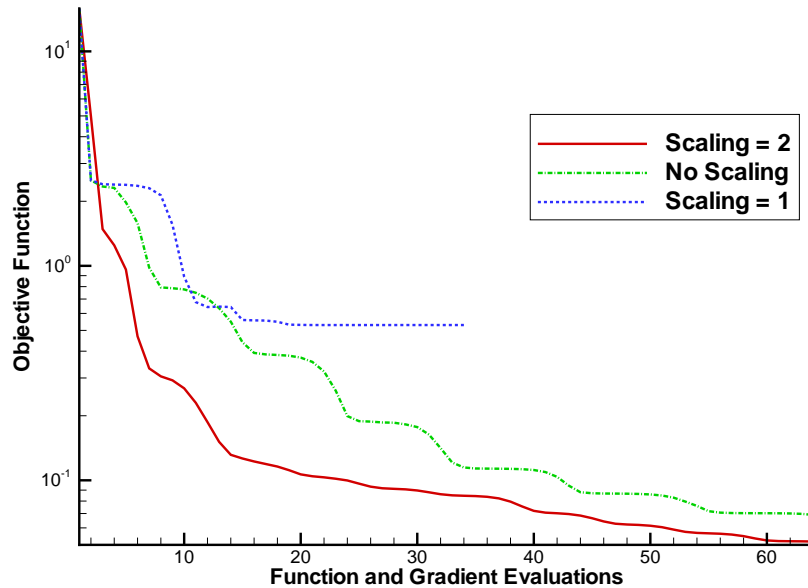


Figure 8.7: Convergence histories with different design variable scaling

8.2 Effects of Scaling on Optimizer Convergence

A single-point optimization case is presented to demonstrate the effects of design variable scaling on optimizer convergence. This example is a single-point optimization problem at an operating Mach number of $M = 0.88$. The following targets and weights on lift and drag are used with (2.5):

$$\begin{aligned} C_L^* &= 0.355 & \omega_L &= 50.0 \\ C_D^* &= 0.0102 & \omega_D &= 1.0 \end{aligned}$$

The optimization case is performed using the design variable scaling strategies discussed in Section 5.3, and the resulting objective function convergence histories are plotted in Figure 8.7. In this particular case, convergence is the slowest without any scaling (green line); the optimizer requires double the number of iterations to decrease the objective function by two orders of magnitude. In general, convergence is found to be the slowest when no design variable scaling is applied, particularly in cases where there is a mixture of planform, B-spline control point and angle-of-attack design variables. In this current example, optimizer convergence is fastest when the design variables are scaled by the square-root of the initial coordinates (Scaling=2; red line). In other transonic lift-constrained drag minimization cases shown in Appendix F, this scaling method also appears to be the most robust. This scaling method is used in all optimization cases presented in this chapter. Finally, when the B-spline control point design variables are scaled by their initial values (Scaling=1, blue line) the optimizer stalled during a line search. This line search stalling also occurred in some of the cases shown in Appendix F.

This study demonstrates that the convergence rate of the optimizer indeed depends on the scaling of the design variables. In particular, there is evidence to suggest that the optimizer convergence is fastest when the design variables are scaled by the square root of their initial values. An in-depth study is required to further investigate the effectiveness and efficiency of each scaling method.

8.3 Multi-Point Drag Minimization

A well-known issue with single-point optimization is that a wing optimized for one operating condition may perform poorly under different conditions. Generally, for single-point optimization, the optimizer is able to remove all shocks at the design operating condition. However, with even a small change in Mach number, shocks may develop rapidly. Using potential flow theory to analyze the flow around a wing, Morawetz [121] demonstrated that shock-free transonic flows are isolated solutions, and that any small perturbations will cause the formation of a shock wave.

In the single-point example presented in Section 8.1, the optimizer was able to significantly decrease the drag at the specified operating condition at $M = 0.90$ and $C_L = 0.307$. Compared to the original ONERA M6 wing, the performance improvement is significant over all transonic speeds, as shown in the drag divergence plots in Figure 8.8a. This is not surprising, since the ONERA M6 wing is not designed using supercritical airfoil sections. However, when the performance of the optimized wing is examined more closely, in Figure 8.8b, it is noted that drag increases at speeds both lower and higher than $M = 0.90$.

The goal of a multi-point optimization is to improve the performance of the wing over a wider range of conditions. In the current Newton-Krylov approach, a composite objective function is defined as a weighted sum of the objective functions from each operating point i :

$$\mathcal{J}_T = \sum_i \omega_i \mathcal{J}_i \quad (8.1)$$

Each operating point can have a different operating Mach number, and the objective function itself, as well as the targets in lift and drag, may also be different for each operating point. The composite objective function can be considered as an approximation to the weighted integral:

$$\mathcal{J}_T \approx \int_{M_1}^{M_2} \omega(M) \mathcal{J}(M) dM \quad (8.2)$$

to minimize the objective function over the range of Mach numbers of interest. Following (8.1), a composite gradient is summed in similar fashion:

$$\mathcal{G}_T = \sum_i \omega_i \mathcal{G}_i \quad (8.3)$$

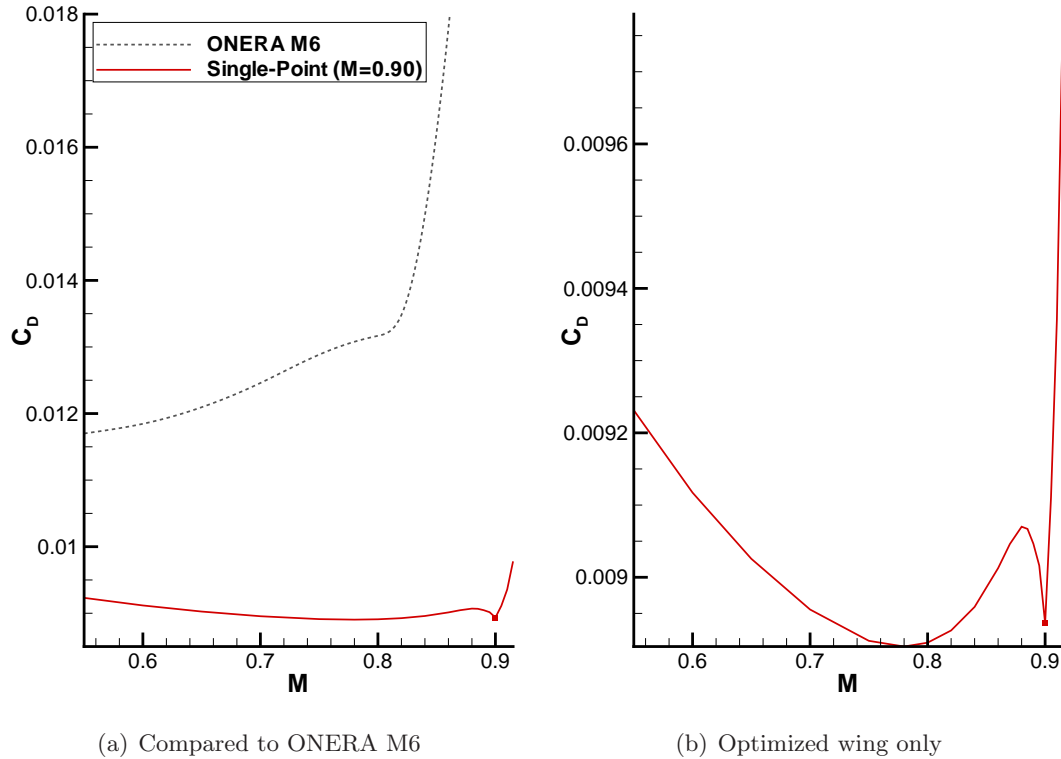


Figure 8.8: Drag divergence plot for the optimized wing

Note that the flow solution and adjoint system must be solved at each operating point in order to compute the composite gradient. The composite objective function and gradient (\mathcal{J}_T , \mathcal{G}_T) replace \mathcal{J} and \mathcal{G} in the optimizer.

Results are presented for a three-point optimization case. The goal of this case is to minimize the mean C_D of the wing in the range $0.70 < M < 0.90$, at a fixed lift coefficient $C_L = 0.307$. Note that in this case, each operating point has the same target lift and drag coefficients (C_L^* , C_D^*), rather than the same target lift and drag, which scales with the square of Mach number. The latter case would mean specifying the same $C_L^* M^2$ and $C_D^* M^2$ over all operating points. In the current case, it is assumed that as an aircraft increases its speed, it is able to climb to a higher altitude (lower ρ and a) such that the same lift is generated. Between $M = 0.7$ and $M = 0.9$, this means a change of altitude of more than 8000ft. In a more extensive multi-point optimization that involves dozens of operating conditions, such as the 2D cases studied in Billings and Zingg [202] and Buckley and Zingg [18], operating conditions will include a range of lift coefficients and Mach numbers, to account for take-off, cruise and dive conditions. In Figure 8.8b, within the range of interest, C_D is highest when $M = 0.88$, and it also increases below $M = 0.70$. The reason for the dip in C_D near $M = 0.76$ is not obvious, but similar drag divergence behaviour is also observed in [82]. Based on this figure, operating points for the multi-point optimization are chosen based on where C_D values need to be reduced. The

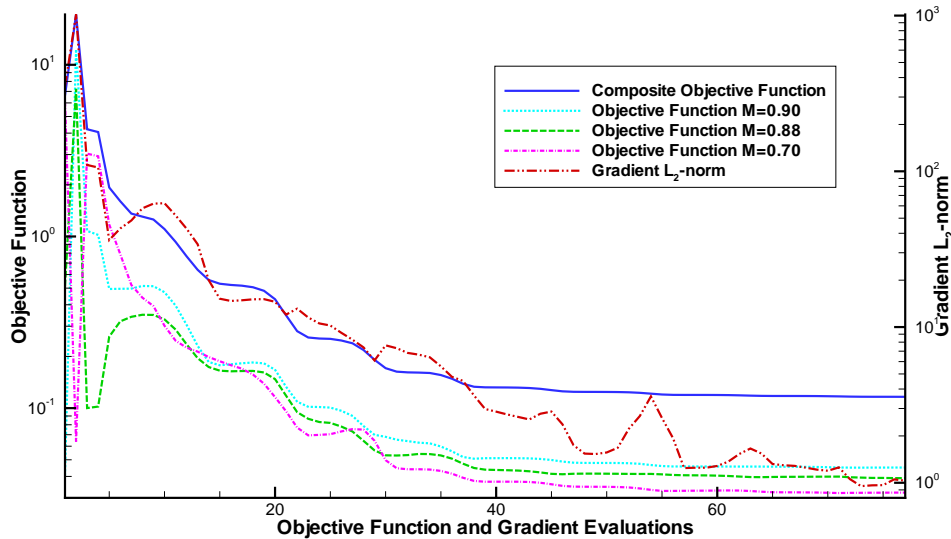


Figure 8.9: Convergence history for the three-point optimization case

selected operating points are: $M_1 = 0.70$, $M_2 = 0.88$ and $M_3 = 0.90$. In this case, each operating point is weighted equally ($\omega_i = 1.0$), but the weighting may change at the discretion of the designer.

The same design variables from the single-point optimization case (Section 8.1) are re-used, and the angles of attack at $M_1 = 0.70$ and $M_2 = 0.88$ are also considered as design variables (total 172 design variables). The optimization begins with the final geometry from the single-point case. The same targets and weights for lift and drag from the single-point optimization case are re-used:

$$\begin{aligned} C_L^* &= 0.308 & \omega_L &= 100.0 \\ C_D^* &= 0.00746 & \omega_D &= 1.0 \end{aligned}$$

As well, design variable scaling is also re-used from the single-point case.

The convergence history for the three-point optimization case is shown in Figure 8.9 after 78 iterations, which took 52 hours. The composite objective function and gradient L_2 -norm have both reduced by about three orders of magnitude. Note that each iteration involves solving one flow solution and one adjoint solution *for each operating point*. In the final geometry, the leading-edge sweep angle is slightly increased to $\Lambda_{LE} = 44.3^\circ$. The final washout angle is also slightly reduced to $\Omega = -5.32^\circ$. The final operating angles of attack are $\alpha = 1.37, 1.38^\circ$ and 2.01° for the operating Mach numbers of $M = 0.70, 0.88$ and 0.90 respectively. The wing cross-sections at six spanwise locations are compared to the single-point results in Figure 8.10. The final wing cross-section shape is very similar to the single-point result. Although the changes in the wing cross-section are small, there is a substantial difference in the performance of the two wings. This can be seen in the drag divergence plot of the optimized wing at fixed

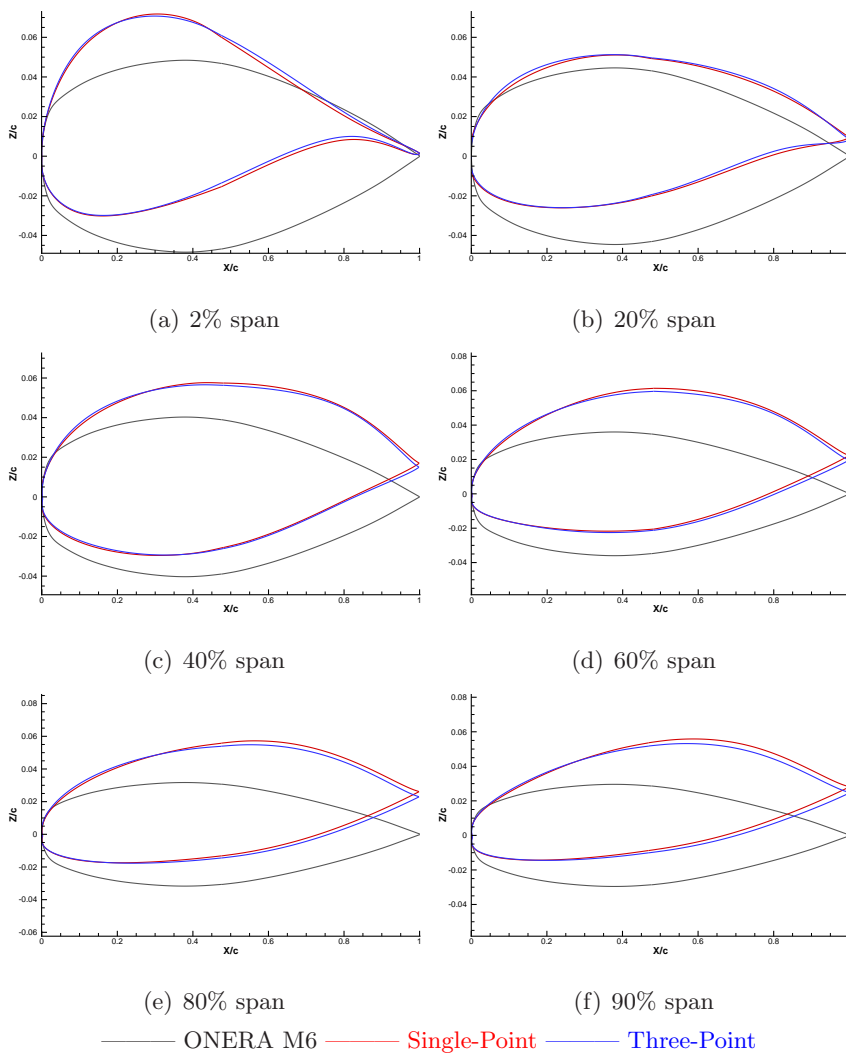
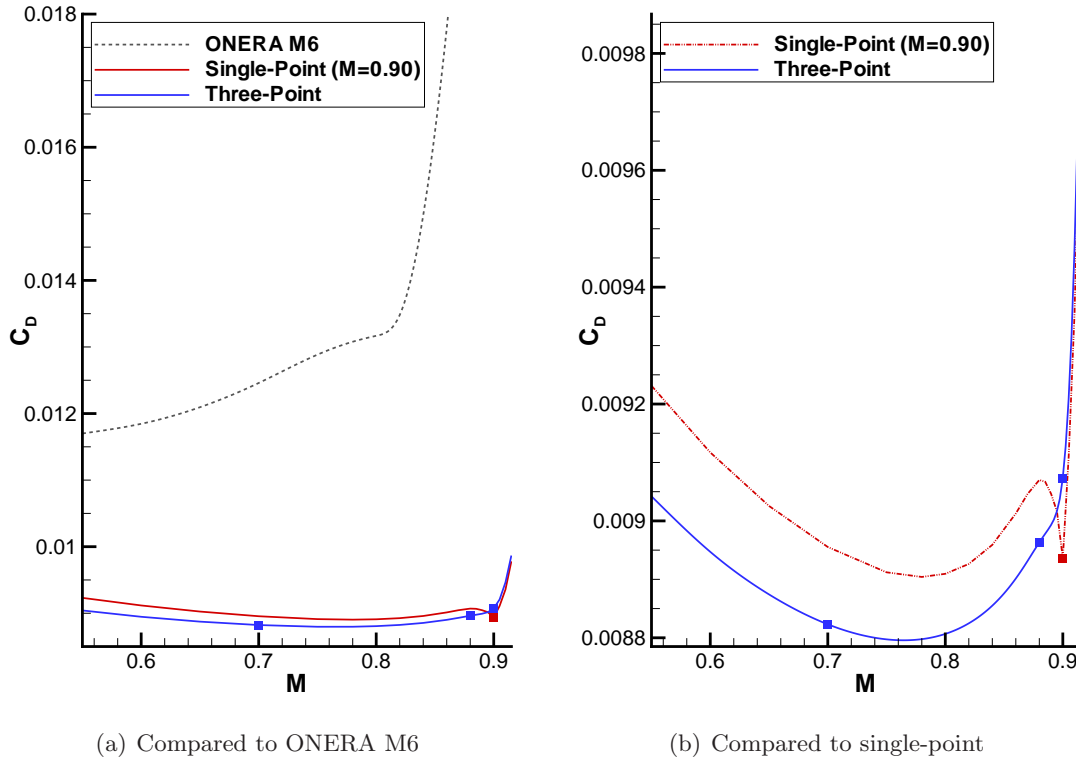


Figure 8.10: Comparison of wing section shapes

$C_L = 0.307$, shown in Figure 8.11. As expected, drag coefficients at $M = 0.70$ and $= 0.88$ have been decreased, while that at $M = 0.90$ has increased slightly. The figure clearly shows that the mean drag over the range of Mach numbers shown has substantially decreased.

8.4 Optimization Using Thickness Constraints

The goal of this single-point optimization case is to minimize the drag coefficient of the DPW-W1 wing at $M_\infty = 0.76$, while maintaining a lift coefficient of $C_L = 0.5$. The DPW-W1 wing originally created for the 3rd AIAA Drag prediction workshop (DPW-III) [185] has a blunt trailing edge which is not suitable for an inviscid flow solver. A modified wing with a sharp trailing edge is used instead. This case is taken from an optimization test suite described by Epstein *et al.* [38]. The suite includes both single-point as well as multi-point optimization

Figure 8.11: Drag divergence plot of the (three-point) optimized wing at fixed $C_L = 0.307$

Optimizer	Method	Design Variables	No. of Variables
SYN107	Continuous adjoint	Surface nodes	Thousands
MPOPT	Response surface	Spline at 4 spanwise stations	35
OPTIMAS	Genetic algorithm	Bézier curve at spanwise stations	55
Current: NK	Discrete adjoint	B-spline surface	176

Table 8.3: Summary of different optimizers

cases.³ Optimization results using the current Newton-Krylov approach are compared to results in [38] using three other optimizers: SYN107 from Intelligent Aerodynamics International [80, 83], MPOPT from the Boeing Aircraft Company [93] and OPTIMAS from Israel Aerospace Industries [149]. Each optimizer uses a different objective function to achieve the optimization goal. The different optimizers are summarized in Table 8.3. Note that all these optimizers use viscous flow solvers in the optimization, but the current algorithm uses an inviscid flow solver.

For this case, the lift-constrained drag minimization objective function (2.5) is used, with

³The current optimization case most closely corresponds to cases S4, M1 and O3 in [38].

Parameter	Value
Root chord (c_{root})	1.0
Semi-span ($b/2$)	3.0
Taper ratio (γ)	0.51
Aspect ratio (AR)	8.0
TE sweep (Λ_{LE})	17.2°
LE sweep (Λ_{TE})	7.9°

Table 8.4: Planform parameters for the DPW-W1 wing

the following targets and weights in lift and drag⁴:

$$\begin{aligned} C_L^* &= 0.502 & \omega_L &= 100.0 \\ C_D^* &= 0.00995 & \omega_D &= 1.0 \end{aligned}$$

The volume grid used in this example is an H-H topology structured grid with 48 blocks. In this grid, the chord at the wing root is normalized to $c_{\text{root}} = 1.0$. Off-wall spacing is set to $0.003c_{\text{root}}$, and far-field boundaries are at least $22c_{\text{root}}$ away from any point on the wing surface. The grid has a total of 733,788 nodes. On each of the top and bottom surfaces, there are 40 nodes in the chordwise direction and 66 nodes in the spanwise direction. The grid is shown in Figure 8.12, and planform parameters are shown in Table 8.4.

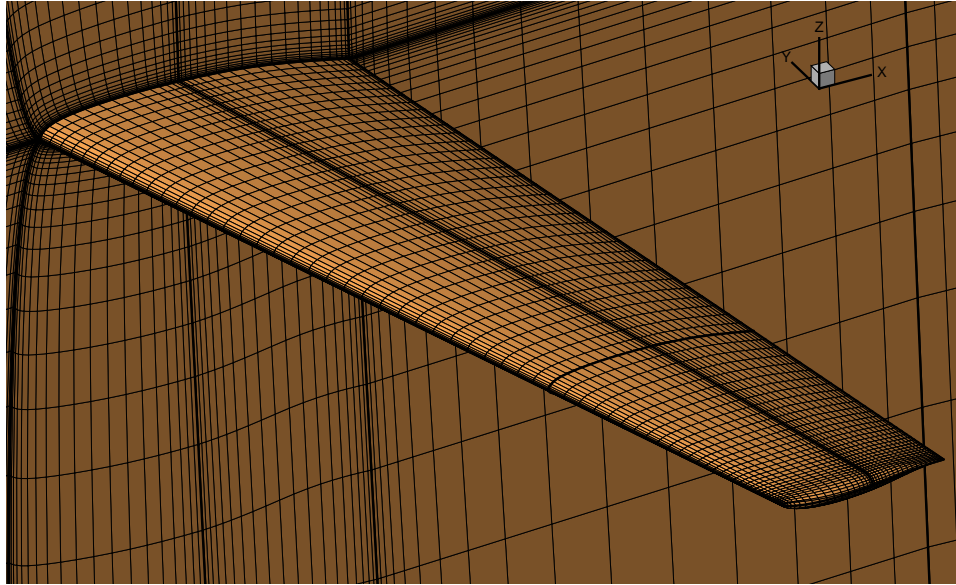
The wing surface is parameterized using eight cubic B-spline control surfaces to represent the wing (four on the wing’s top surface, four on the bottom). Each surface has 6 control points in the chordwise direction and 7 in the spanwise direction, i.e. on each of the top and bottom surfaces, there are 13 control points in the spanwise direction, and 11 points in the chordwise direction. The parameterization for the wing’s top surface is shown in Figure 8.13. Design variables include the z -coordinates of all B-spline control points except near the leading- and trailing-edges. There are a total of 175 B-spline control point design variables. The angle of attack is also a design variable. Following [38], no additional planform design variables are used.

The minimum thickness of the wing is constrained at two chordwise locations:

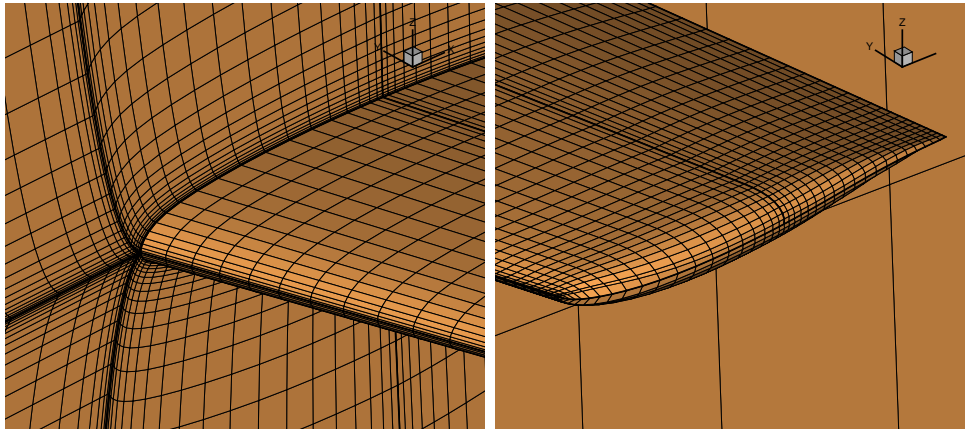
$$\begin{aligned} (t/c)^* &= 0.120 & \text{at } (x/c) &= 0.20 \\ (t/c)^* &= 0.059 & \text{at } (x/c) &= 0.75 \end{aligned}$$

Both thickness constraints are implemented at 17 discrete spanwise locations along the wing. The locations of the constraints and the thickness targets are summarized in Table 8.5. The

⁴The DPW-W1 wing has a planform area of $S = 2.26$. As the planform remains fixed for this optimization case, the equivalent targets are for $\mathcal{L}^* = 1.135$ and $\mathcal{D}^* = 0.022$.

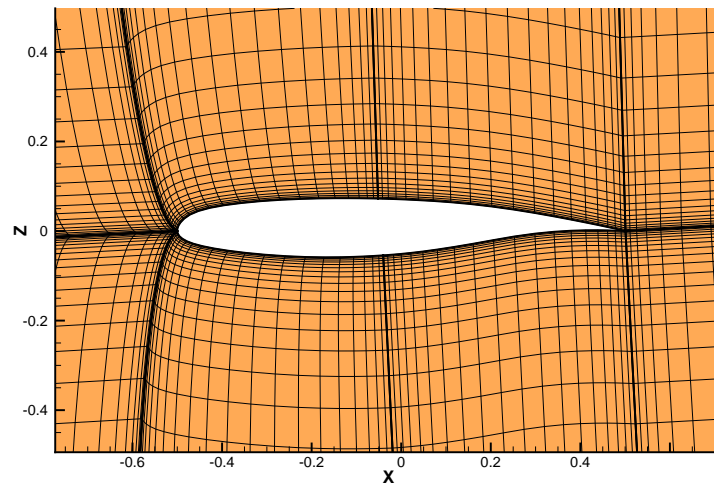


(a) Wing surface



(b) Wing root

(c) Wing tip



(d) Symmetry plane

Figure 8.12: 48-block grid around a DPW-W1 used for the design examples

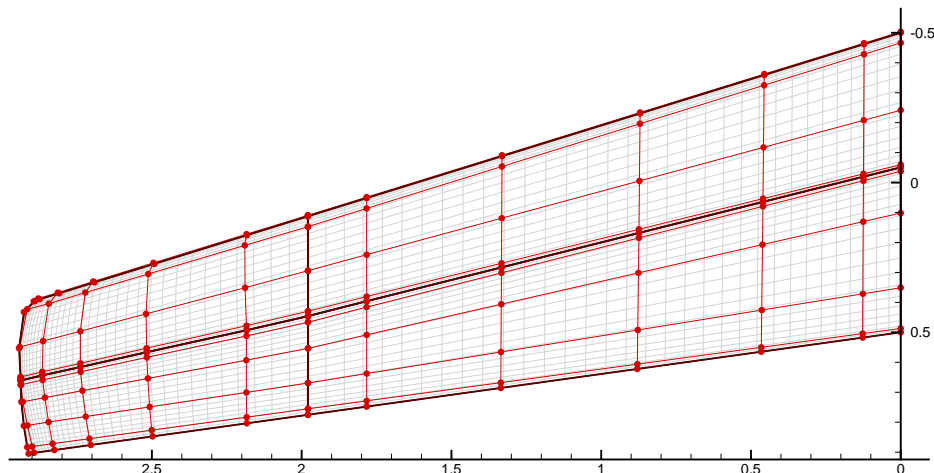


Figure 8.13: B-spline parameterization of the DPW-W1 wing

$2y/b$	t^* at $x/c = 0.20$	t^* at $x/c = 0.75$
0.03	0.118	0.058
0.10	0.114	0.056
0.15	0.111	0.054
0.20	0.108	0.053
0.27	0.104	0.051
0.35	0.099	0.048
0.40	0.096	0.047
0.46	0.093	0.045
0.53	0.089	0.043
0.60	0.085	0.041
0.66	0.081	0.039
0.73	0.077	0.037
0.80	0.073	0.035
0.85	0.070	0.034
0.90	0.067	0.032
0.96	0.063	0.031
0.98	0.062	0.030

Table 8.5: Thickness targets for each thickness constraint

targets are specified as the absolute thickness of the wing at the constraint location. The constraints are enforced using a penalty method (3.21). A penalty weight of $\omega_T = 50.0$ is used. A constraint for maximum thickness is also described in [38], but it is found to be unnecessary in this particular design case.

The convergence history is shown in Figure 8.14 after 96 iterations, which took 32 hours with 48 processors. At this point, both the objective function and the gradient L_2 -norm have reduced by about two orders of magnitude. Similar to the results in Section 8.1, the line-search algorithm is rarely used; a full Newton step generally satisfies the strong Wolfe conditions. All

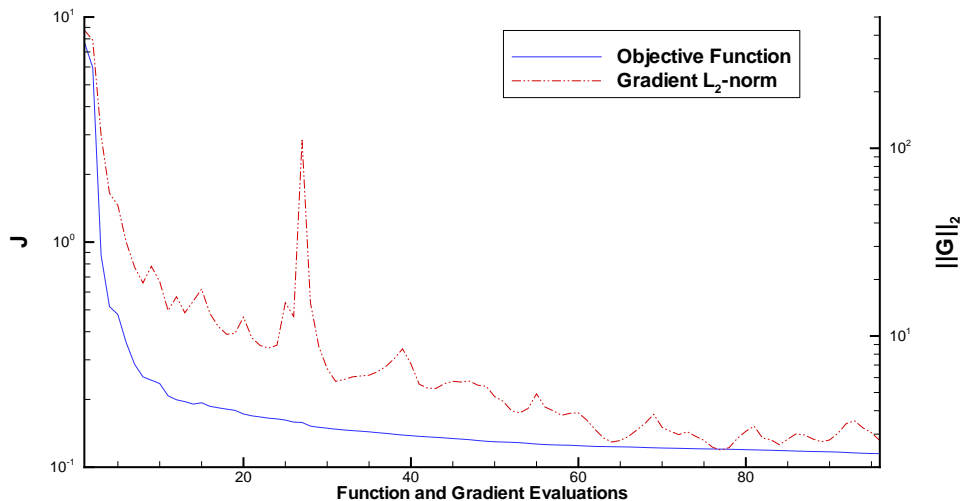


Figure 8.14: Convergence history for the optimization of the DPW-W1 wing

thickness constraints are active at the end of the optimization, indicating that the thickness at those locations is slightly below the specified minimum. The optimized wing's cross-sections and pressure distributions at six spanwise locations are plotted against the original DPW-W1 wing in Figure 8.15. It shows that the shock on the top surface of the wing has been removed. The oscillation on the top surfaces on the wings in the C_p plots is due to the application of the SATs at the block boundary on the top surface. In comparing the final geometries, the optimized wing using the current algorithm is similar to those presented in [38].

The drag divergence plot of the optimized wing is compared to the DPW-W1 wing in Figure 8.16. At the operating point of $M = 0.76$, the drag on the optimized wing is 18% lower than the original DPW-W1 wing, although there is improvement even for $M < 0.76$. However, at speeds above $M = 0.78$, the optimized wing does not perform as well as the original DPW-W1 wing.

The drag reduction using the Newton-Krylov approach is compared to the other optimizers Table 8.6. The drag reduction reported by the NK approach is comparable to the other optimizers. Note that the three other optimizers all use a RANS flow solver, while an Euler flow solver is used by the Newton-Krylov optimizer. In the drag terms reported in [38], SYN107 and MPOPT results do not distinguish between reduction in friction drag, wave drag and induced drag. However, the results reported by the OPTIMAS code, the optimized shape had no improvements in viscous drag: all improvement come from reducing wave drag and induced drag.

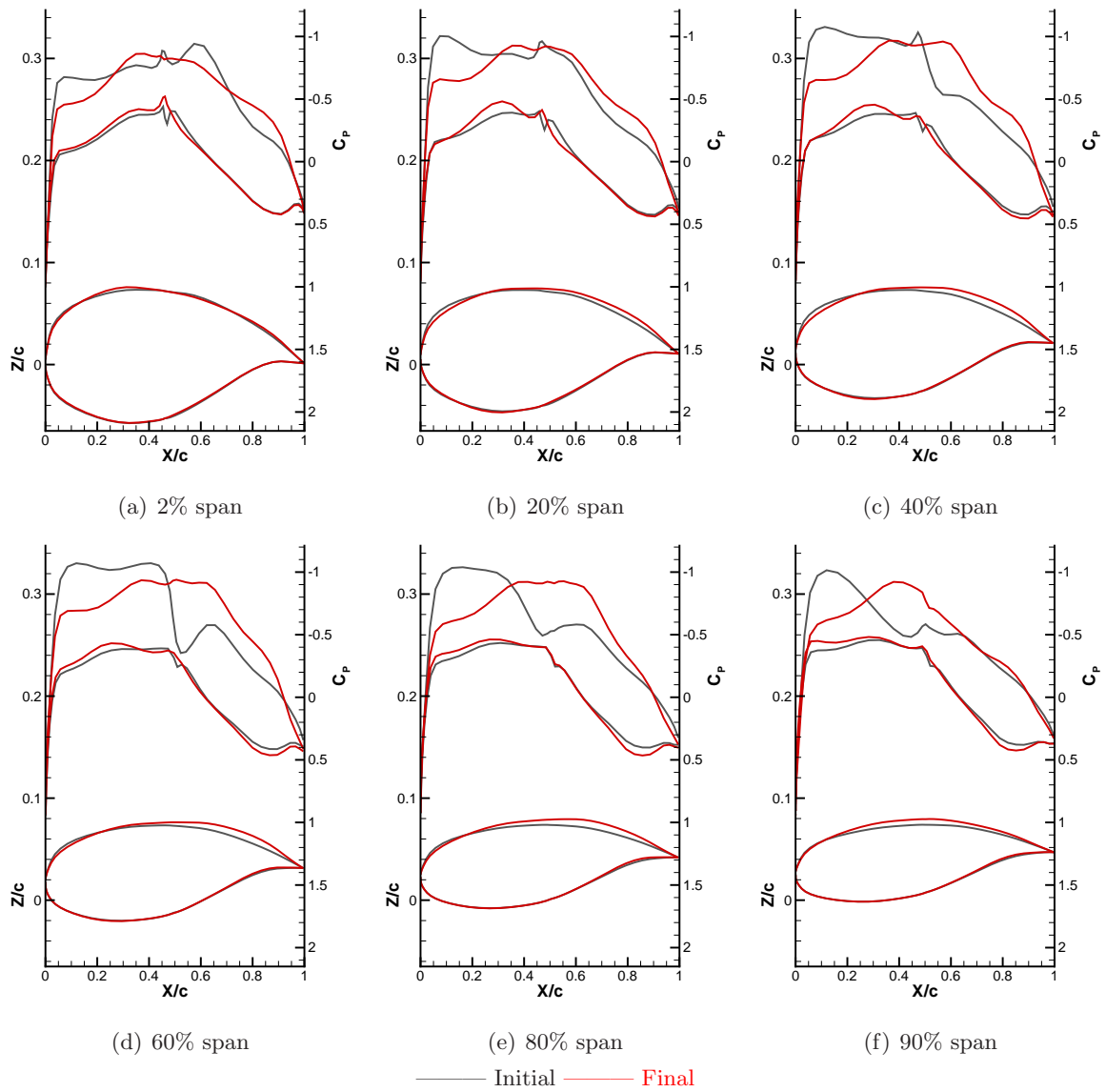


Figure 8.15: Surface pressure coefficient plots and wing sections at six spanwise stations for the DPW-W1 single-point optimization case

Optimizer	Drag Reduction
SYN107	-0.00129
MPOPT	-0.00118
OPTIMAS	-0.00171
Current: NK	-0.00140

Table 8.6: DPW-W1 optimization results compared to other optimizers

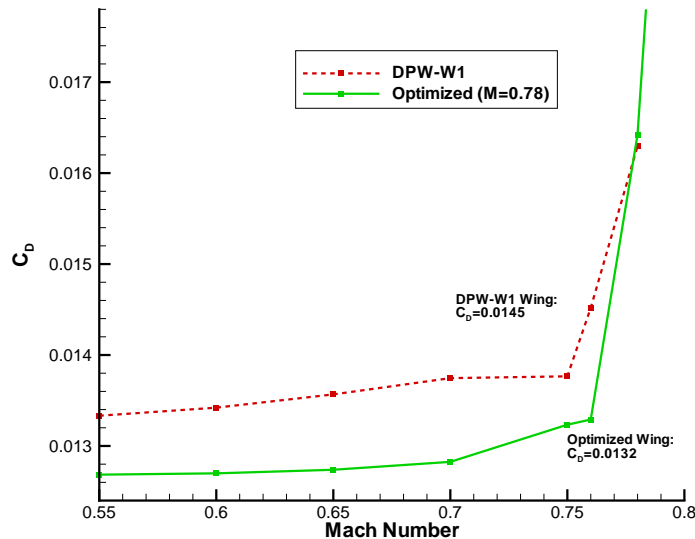


Figure 8.16: Drag coefficients at $C_L = 0.50$ for the DPW-W1 optimization case

8.5 Effect of the Initial Geometry

The final design example studies the impact of the initial geometry on the wing's final shape, and to determine the uniqueness of the optimum achieved by the optimizer: whether or not two starting geometries will converge to a unique solution. Hicken and Zingg have shown in the design of wingtip devices [72, 69] that multiple local optima can exist. The focus of this design example is on the sweep angle of the wing at transonic speeds.

The goal of this optimization is to find the shape, or shapes, which minimizes drag at an operating Mach number of $M = 0.76$ and a lift coefficient of $C_L = 0.40$. The following targets and weight on lift and drag are used for (2.5):

$$\begin{aligned} C_L^* &= 0.402 & \omega_L &= 100.0 \\ C_D^* &= 0.00610 & \omega_D &= 1.0 \end{aligned}$$

The optimization cases are performed using two initial geometries. The first geometry is a tapered wing with planform parameters shown in Table 8.7 (CASE I). The original wing has a cross-section shape that is constant from wing root to wing tip. The second geometry (CASE II) has the same wing cross-section as CASE I, but the wing has an initial forward sweep angle of $\Lambda_{LE} = -6.3^\circ$ at the leading edge.

The volume grids around the two wings are 12-block H-H topology grids with 320,292 nodes. On the top and bottom surfaces, there are 31 and 41 nodes on the chordwise and spanwise directions respectively. The wing semi-span is scaled to unity, and the root chord is scaled to $c_{\text{root}} = 0.667$. Offwall spacing is 10^{-3} , and the far-field boundaries are at least $37.5c_{\text{root}}$ away

Parameter	Value
Root chord (c_{root})	0.667
Semi-span ($b/2$)	2.00
LE sweep angle (Λ_{LE})	5.2°
TE sweep angle (Λ_{TE})	-5.2°
Twist angle (Ω)	0.00
Dihedral angle (Γ)	0.00
Taper ratio (γ)	0.55
Aspect ratio (AR)	8.27
Planform area (S)	0.967

Table 8.7: Planform parameters for the tapered wing

$2y/b$	t^*	$2y/b$	t^*
0.01	0.0398	0.68	0.0266
0.05	0.0390	0.77	0.0249
0.20	0.0361	0.89	0.0225
0.40	0.0321	0.96	0.0211
0.60	0.0282	0.99	0.0206

Table 8.8: Leading-edge thickness constraints

from any point on the wing's surface. The two initial geometries and their respective grids are shown in Figure 8.17. At the start of the optimization runs, the drag coefficients at $C_L = 0.40$ are $C_D = 0.0366$ and $C_D = 0.0355$ respectively for CASE I and CASE II.

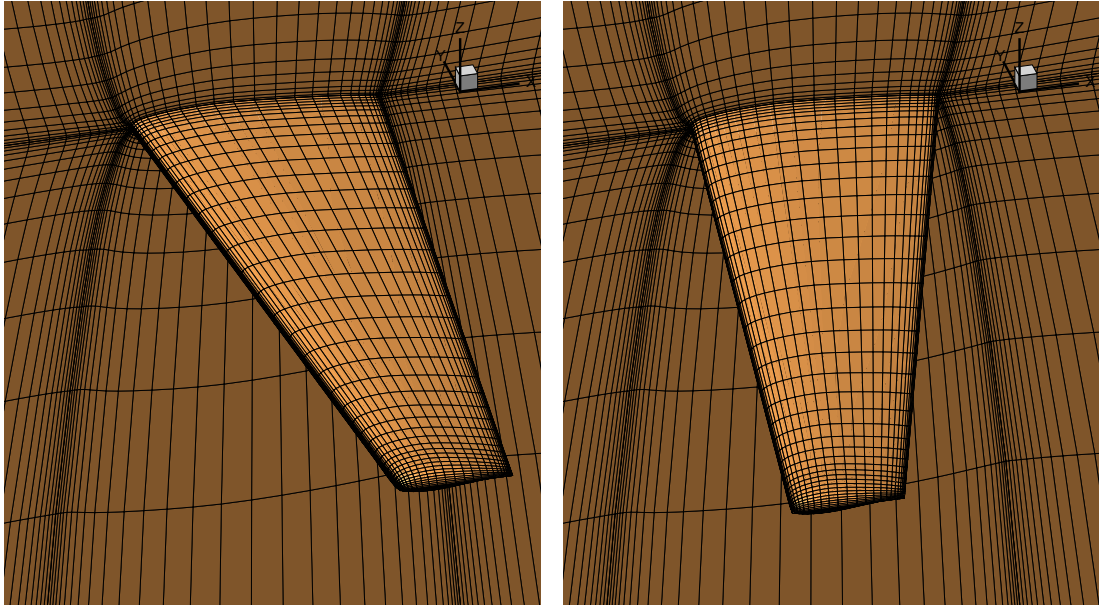
Both the top and bottom surfaces of each wing are parameterized with cubic B-spline surfaces. Each surface has 12 control points in the spanwise direction and 11 in the chordwise direction. The z -coordinate of every B-spline control point is a design variable, except at the leading and trailing edges, where the control points are fixed. There are 230 B-spline control point design variables in total. In addition, the change in the wing's sweep angle at wing root ($\Delta\Lambda_{LE}$), twist ($\Delta\Omega$) and the angle of attack (α) are also design variables.

A volume constraint (3.17) is used to maintain the volume enclosed by the wings, with the parameters $v_f = 0.0$ and $\omega_V = 50.0$ used, following Section 8.1. Thickness constraints are implemented near the leading and trailing edges to prevent grid crossover:

$$\begin{aligned} (t/c)^* &= 0.04 & \text{at} & \quad (x/c) = 0.02 \\ (t/c)^* &= 0.002 & \text{at} & \quad (x/c) = 0.97 \end{aligned}$$

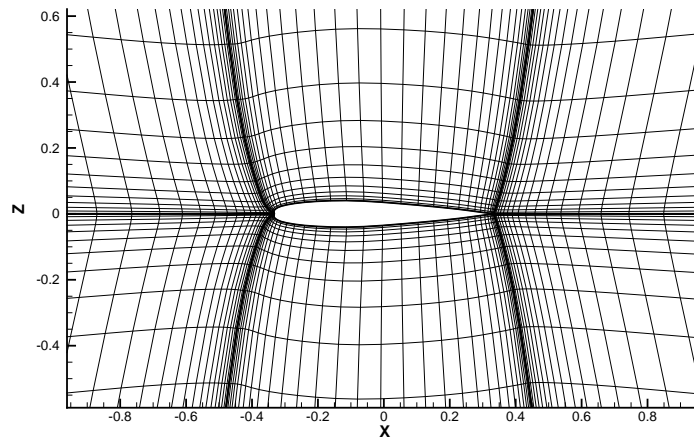
The actual thickness constraints are specified as an absolute thickness at a single location, as shown in Tables 8.8 and 8.9. A penalty weight of $\omega_T = 50.0$ is used.

The objective function and gradient convergence histories are shown for both initial geome-



(a) Tapered wing

(b) Tapered wing with forward sweep



(c) Symmetry plane

Figure 8.17: 12-block grid around tapered wings used for the design examples

$2y/b$	t^*	$2y/b$	t^*
0.01	0.0398	0.68	0.0266
0.05	0.0390	0.77	0.0249
0.20	0.0361	0.89	0.0225
0.40	0.0321	0.96	0.0211
0.60	0.0282	0.99	0.0206

Table 8.9: Trailing-edge thickness constraints

tries in Figure 8.18 after 250 optimization iterations for CASE I and 147 iterations for CASE II. In CASE I, the objective function and gradient L_2 -norm have decreased by almost four

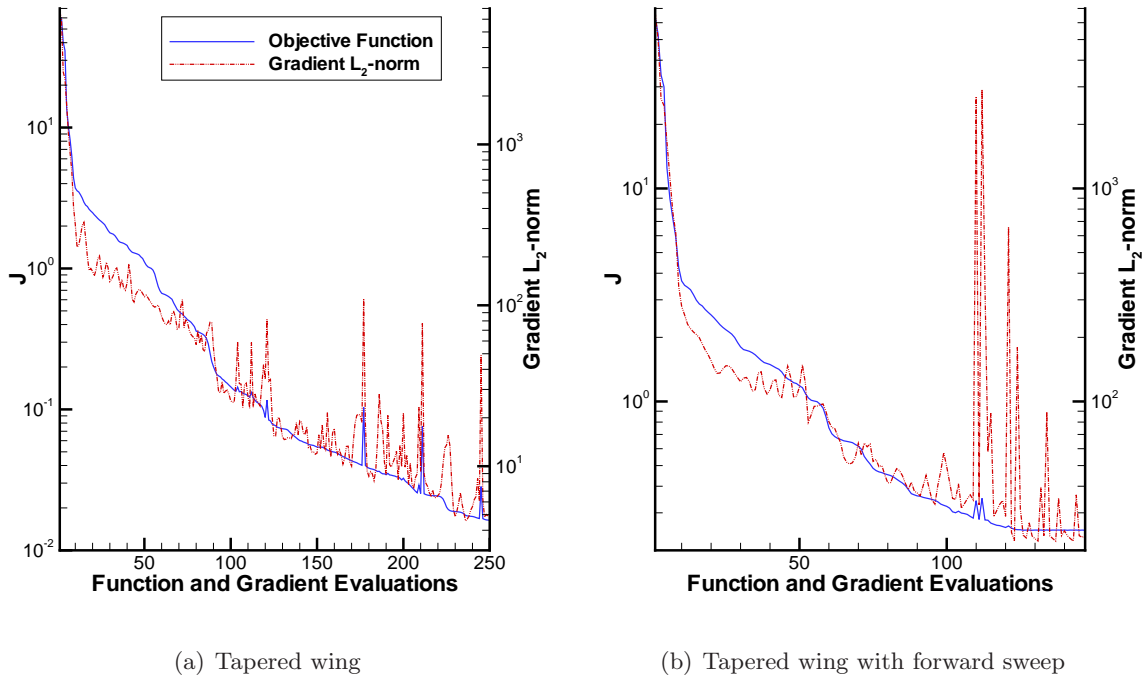


Figure 8.18: Convergence histories with both initial geometries

Initial Geometry	Initial C_D at $C_L = 0.40$	Final C_L	Final C_D
Tapered Wing	0.0366	0.400	0.00671
Tapered wing with forward sweep	0.0355	0.399	0.00902

Table 8.10: Final C_L and C_D values for different initial geometries

orders of magnitude, and for CASE II, there was close to three orders of reduction. Most of the reduction in the objective function is achieved during the first 20 iterations, consistent with previous results in Sections 8.1 and 8.4. In both cases, both the volume and thickness constraints are active in the final geometry, with the final volumes and thicknesses within 0.15% of the target. The final geometries from both cases have resulted in improvements in the aerodynamic efficiencies, as shown by the C_L and C_D values in Table 8.10. Based on the convergence histories, it appears that both cases have converged to a local optimum. However, despite both cases have converged to a local optimum, their final planform geometries are very different. In CASE I, the wing swept back to a LE sweep angle of $\Lambda_{LE} = 28.6^\circ$, and washout angle of $\Omega = -3.60^\circ$. This is in contrast with CASE II. In this case, the forward sweep angle continued to increase, to $\Lambda_{LE} = -29.4^\circ$ in the final geometry. A smaller washout angle of $\Omega = -0.60^\circ$ is also added. Also, the optimized geometry from both cases produced very different C_D values, with the forward sweep case resulting in a substantially higher final C_D value. This may be due to y component of the flow near the leading and trailing edges of the wings. In the forward

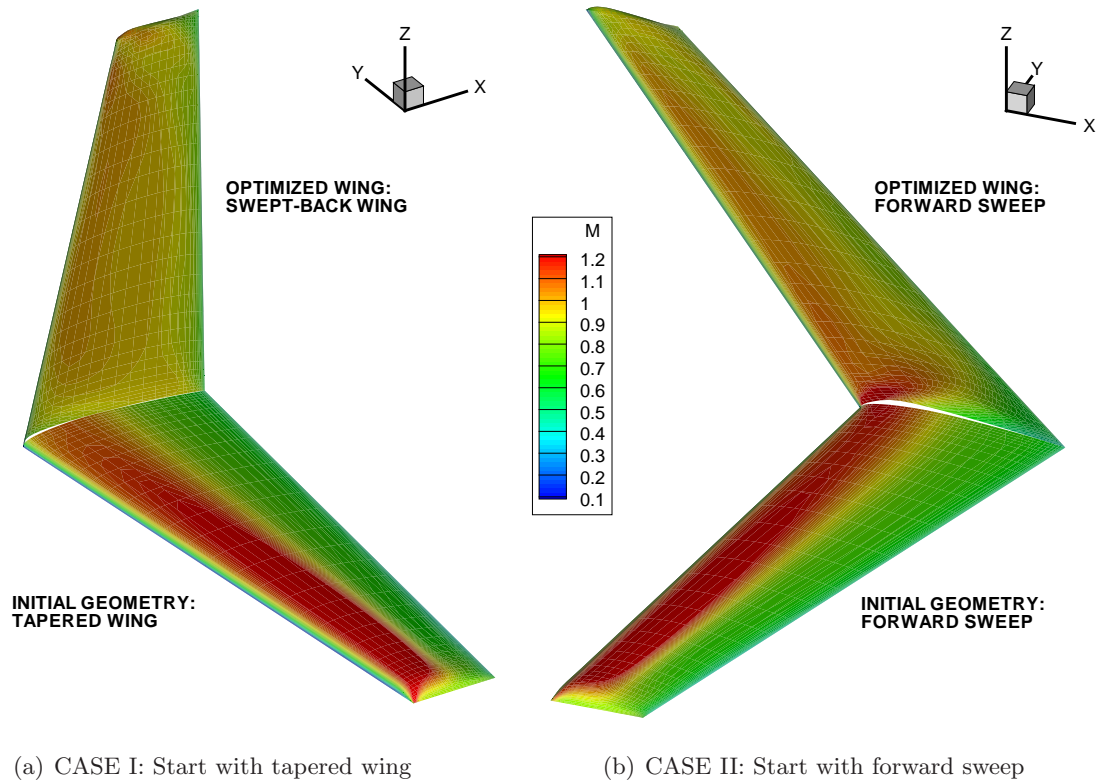


Figure 8.19: Initial and final geometries for the design case with different initial geometries.

sweep case, the y -component is towards the fuselage, while for the swept-back wing, the flow is towards the wing tip.

For both cases, the final wings are compared to their respective initial geometries in Figure 8.19. They are shown with Mach contours on the wing surfaces. The figures show that the shock (and therefore wave drag) have been eliminated in the optimized geometries. This can be seen clearly in the pressure coefficient plots in Figure 8.20.

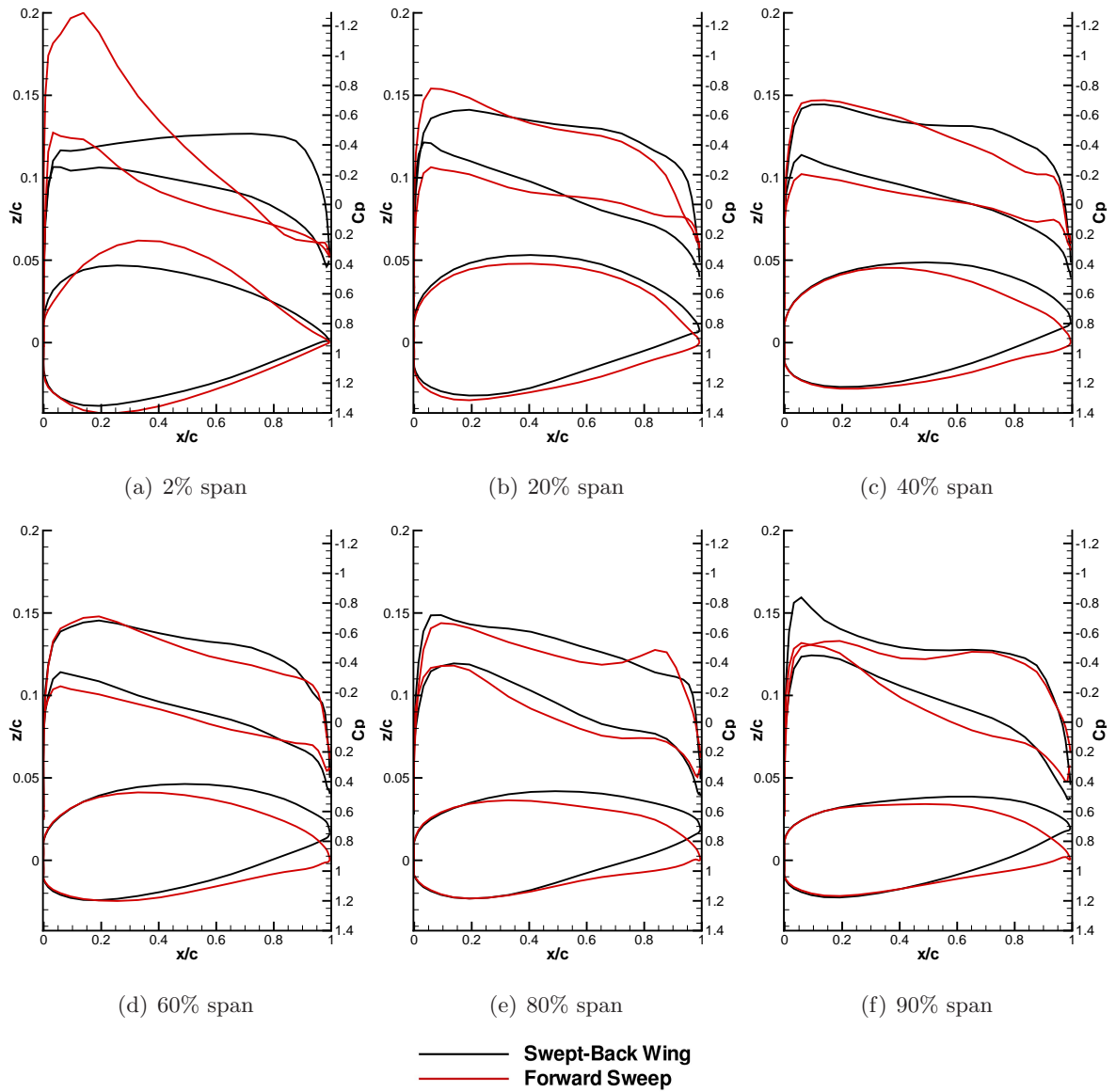


Figure 8.20: Final pressure coefficients and wing sections for both initial geometries

Chapter 9

Conclusions and Recommendations

The primary objectives of this thesis are to extend the Newton-Krylov approach for aerodynamic shape optimization pioneered by Nemec and Zingg [128, 133, 132, 134] to wing optimization in three dimensions, to address the issues arising from this extension, and to characterize the performance of the three-dimensional algorithm in a series of design examples. The goal is to develop a framework for an aerodynamic shape optimization tool that can be used in industrial applications. Ultimately, this optimizer should be incorporated as part of a multi-disciplinary optimization approach.

9.1 Summary of the Newton-Krylov Algorithm

A summary of the overall algorithm is as follows. The Newton-Krylov approach refers to using the Newton method and/or a Krylov subspace method to obtain the flow solution, objective function gradient and the search direction. In the current implementation, a Jacobian-free Newton method with pseudo-transient globalization is used to solve the steady Euler equations in order to compute the objective function. The quasi-Newton gradient-based optimizer BFGS is used to find the optimal design shape. The optimization problem is treated as an unconstrained problem, with geometric constraints cast as penalty terms in the objective function. The linear systems arising from each iteration of the flow solver as well as the adjoint system for the gradient calculation are solved using the Krylov subspace method FGMRES. In both the flow solution and the adjoint equations, the linear system is preconditioned with an approximate-Schur preconditioner to improve convergence of the FMGRES algorithm.

The current Newton-Krylov method is built upon a parallel 3D inviscid Newton-Krylov flow solver and discrete adjoint solver by Hicken and Zingg [73], which is based on flow solvers by Nichols and Zingg [139], Nemec *et al.* [135, 128] and Pueyo and Zingg [152]. The spatial discretization is based on the finite-difference scheme of Pulliam [153], with artificial dissipation

terms based on Jameson *et al.* [84].

9.2 Conclusions and Original Contributions

The focus of this thesis is on the speed and efficiency of the algorithm, as it is applied to the design of wings at transonic speeds. Since the Euler equations do not account for viscous and turbulent effects, validation cases and design examples focus on minimization of wave drag and induced drag at transonic speeds. Based on the validation cases discussed in Chapter 7 and the design examples shown in Chapter 8, the advantages of the Newton-Krylov approach to aerodynamic shape optimization can be summarized:

- The Newton-Krylov flow solver is fast, efficient and robust. The algorithm is able to obtain a fully converged flow solution on a grid with one-million nodes in 50 minutes using 12 processors, while in the design problems, a fully converged flow solution can be obtained in eight minutes on a half-million node grid using 48 processors. The flow solver can converge for a wide range of flow conditions from subsonic to transonic flows as well as for different geometries. This robustness is key in the optimizer's ability to do multi-point optimization that may involve both subsonic and transonic flows.
- The Krylov method FGMRES used in the flow solver can be used to efficiently solve the discrete adjoint equation to obtain the gradient. Solution to the adjoint system can be obtained in roughly one-fourth the time of a flow solution.
- The quasi-Newton optimizer BFGS is able to obtain good design improvements in very few design iterations, while offering a more complete convergence than the steepest descent method if the designer needs to find a genuine optimum.

In addition, original contributions and findings for each component of the Newton-Krylov optimizer are presented in following subsections.

9.2.1 Optimizer Effectiveness

A quasi-Newton gradient-based unconstrained optimizer based on BFGS has been implemented. This optimizer is capable of handling both single- and multi-point optimization problems, with seamless restart capability if the optimizer is stopped. A line-search algorithm with backtracking capabilities based on Nemeč [128] has also been implemented. An automatic restart feature using a steepest descent first step is used if the line search algorithm fails.

The optimization algorithm is found to be very efficient, in that it is able to converge to a local optimum in a relatively small number of iterations. For problems with about 170-200

design variables, the optimizer generally reduces the gradient L_2 -norm by more than three order of magnitude after about 200 design iterations, where each design iteration consists of one objective function evaluation and one gradient evaluation. The cost of an adjoint gradient evaluation is on the order of an objective function evaluation. Overall, the cost of 200 design iterations is comparable to the cost of 300 functional evaluations. The number of functional evaluations is much fewer than gradient-free optimizers such as genetic algorithms, but appears high compared to other published results for gradient-based optimization, where the cost of a design optimization that is one order of magnitude more expensive than a flow solution is often quoted, e.g. Jameson *et al.* [82]. However, in those cases, convergence histories are usually not shown, and it is likely that the optimizer is not converged. For comparison, the current Newton-Krylov optimizer is also able to achieve significant design improvements after only about 10-20 iterations, which is consistent with other published works. It is concluded that the overall Newton-Krylov approach is a fast and efficient alternative that is competitive against other gradient-based optimizers.

With regards to the optimized geometry from the design examples in Chapter 8, changing the sweep angle of the wing is found to be a direction of improvement that is favoured by the optimizer. In most cases, the sweep angle increases, and higher final sweep angles are obtained by the optimizer at higher Mach numbers. Note that increasing sweep may lead to heavier wings (increased structural weight W_S) which may decrease the aircraft's range, as noted in the Breguet range equation (2.4). If the geometry parameterization is given enough flexibility to control the cross-sectional shape of the wing, it is possible to eliminate shocks without changing the sweep angle.

9.2.2 Application of Geometric and Performance Constraints

To ensure a physically realistic design and to enforce design limitations, two geometric constraints have been implemented: a volume constraint limits the change in the volume enclosed by the wing, and thickness constraints specify minimum wing thicknesses at specific locations on the wing. They are applied as penalty terms in the objective function. In addition, a lift constraint is incorporated into the objective function itself. This allows the optimization problem to remain an unconstrained problem.

In the design cases presented, most the constraints are active, i.e. they have small non-zero penalty terms, at the end of the optimization cycle. This means that the constraints are violated slightly. Such violations can be avoided with careful selection of constraint targets and weights. However, it may also be advantageous to use a constrained optimizer such as SNOPT [50] which also uses a quasi-Newton optimizer.

9.2.3 Gradient Accuracy

Both discrete adjoint and finite-difference methods for gradient calculation have been implemented. A study to determine the accuracy of the adjoint gradient against finite differencing has been performed. The accuracy of the discrete-adjoint gradient is found to be excellent compared to second-order centred differencing. Gradients computed using both methods differ by less than 0.01% in both subsonic and transonic validation cases with two different objective functions. The discrete adjoint gradient is obtained at a substantially lower cost than finite differencing. For transonic flow problems, good agreement is only obtained when second-difference dissipation terms are turned off.

It should be noted that finite differencing is still used to evaluate partial derivatives $\partial\mathcal{J}/\partial\mathcal{X}$ and $\partial\mathbf{R}/\partial\mathcal{X}$ in the adjoint gradient calculations. With the inexpensive algebraic grid movement method currently used, finite differencing is an obvious option to implement. However, the adjoint gradient may therefore be affected by the same numerical errors as finite-differencing, namely truncation and subtractive cancellation errors. In this case, hand linearization of these partial derivatives, while labour intensive, is a good alternative.

In addition, the linearization of the residual vector (flow Jacobian matrix) assumes that the second-difference dissipation terms are frozen. In order to obtain an accurate gradient, second-difference dissipation must be turned off during the optimization cycle. In single-point optimization, the final geometry is usually shock-free, i.e. effects from turning off second-difference dissipation are small. However, for more complicated multi-point optimization cases where the optimized geometry may not be shock free, having second-difference dissipation will improve the accuracy of the flow solution.

9.2.4 Geometry Parameterization and Design Variables

An efficient geometry parameterization strategy using B-spline control surfaces is implemented. The algorithm extends the capabilities of the CAD-free parameterization of Fudge *et al.* [45, 44] in its improved flexibility for spanwise control of the geometry. The current method uses two-levels of design variables to allow two levels of control over the geometry. At the top level, a set of planform variables control the overall shape of the wing using a reduced set of design variables, and at the lower level, individual B-spline control points change the wing's spanwise cross section shapes. Moreover, the angle of attack and free-stream Mach number are implemented as design variables as well.

This parameterization method is an effective technique to manage design variables; an entire wing can be parameterized using 200 to 300 B-spline control points. The number of design variables is similar to using the Hicks-Henne bump functions (Appendix B.1), e.g. parameterization

of wing sections at four spanwise stations, using 15 basis functions at each section will require 120 design variables. The number of design variables is also far below the discrete method using surface nodes (Appendix B.2), which will require more than 2000 design variables for the grids used in the design examples, even if only z -coordinates are considered. This number will be closer to 7000 if all three degrees of freedom are used.

9.2.5 Grid Movement Algorithm

A fast and robust algebraic grid movement algorithm based on Nemec [128] has been implemented. This algorithm provides an inexpensive and robust alternative to generate a new computational grid at each iteration, as well as for evaluations of partial derivatives. The algorithm is found to work well even for large changes in geometry—especially for sweep angle changes—as long as the block boundaries are sufficiently far from the body surface. In the event that the new grid produces inverted or degenerate cells, the negative Jacobian is detected and the optimizer backtracks.

One weakness of the grid movement algorithm is the range of geometry changes that it can handle. Although the algorithm performs well for the optimization problems studied in Chapter 8, it may fail for larger changes in geometries than those encountered, such as “growing” a winglet from a previously flat wing.

9.3 Future Work

Based on the findings and conclusions, the following future work should be considered:

1. Extension of the flow solver to the Reynolds Averaged Navier-Stokes (RANS) equations with a suitable turbulence model, such as the Spalart-Allmaras model [176], will be necessary for more accurate drag prediction, and to better capture realistic flow phenomena such as flow separation, local unsteadiness, and interaction between shocks and a boundary layer. In particular, since there is no flow separation in the Euler equations, locally optimal shapes that are permitted using the Euler equation may not be realistically optimal. A RANS solution may give a more comprehensive understanding of the optimal shapes.
2. Perform a range optimization by directly maximizing the range using (2.4). It should be noted that without any suitable structural models, and without accounting for viscous drag effects, such range optimization always results in a higher operating Mach number, and a highly swept-back wing optimized for that operating condition. The idea for range

maximization is prompted by results by Jameson *et al.* [86] which suggested that optimizing the range leads to wings with lower sweep angles operating at lower Mach numbers. Such an optimization case will involve using the Mach number as a design variable. The Mach number at which an aircraft operates has profound impact on the sweep angle, which is related to the range through structural weight W_S . An aircraft's operating Mach number also affects the engine's specific fuel consumption [114]. The latter is addressed by casting the fuel consumption term as a function of Mach number in the objective function. Additionally, one way to address the effects of additional weight on the range of the aircraft is to cast any weight penalty as an appropriate penalty term in the objective function, similar to thickness and volume constraints. A better alternative, however, is to incorporate a suitable structural model into the optimization for a full aero-structural optimization.

3. The capabilities of the geometry parameterization must be further generalized in order to give it enough flexibility to define and modify wing-fuselage and unconventional configurations, and to ensure the integrity of the surface grid at the junction between the wing and the fuselage. Such generalization is needed for optimization of a wing in the presence of a fuselage, or more comprehensive optimization of the wing and the fuselage together. Further improvements in the geometry parameterization may include allowing flexible control over the local twist of the wing in the spanwise direction.
4. In order to optimize a wing-fuselage configuration and unconventional aircraft designs, other constraints based on performance measures should be investigated. For example, a pitching moment constraint ($C_M^* = 0$) may be required to ensure the stability of the aircraft.
5. By casting the geometric constraints as penalty terms in the objective function allowed the use of the unconstrained optimizer BFGS. However, the selection of targets and penalty weights for these geometric constraint's is not trivial. One alternative is to use a constrained optimizer such as SNOPT [50] which also uses a quasi-Newton optimizer. A study to compare the effectiveness and efficiency of SNOPT against the unconstrained optimizer BFGS may be useful to determine which optimizer is better suited for this type of optimization problems.
6. In order to perform multi-point optimization problems where the final geometry may not be shock-free, improvements in gradient accuracy is required when second-difference dissipation terms are turned on. This can be done in one of several ways. One way is to linearize the differentiable functions in the dissipation terms in (4.20), (4.23), (4.24)

and (4.26), while freezing the non-differentiable portions in (4.21) and (4.25). Another approach is to apply the complex-step method to the dissipation terms. This approach is similar to what is already done with the SATs at block boundaries, and also similar to the approach by Nielsen and Kleb [141] to form the flow Jacobian using the complex-step method entirely. An alternative may be to use automatic differentiation to form the Jacobian [107]. Another area where the gradient accuracy can be improved is in the partial derivative with respect to the design variables $\partial\mathcal{J}/\partial\mathcal{X}$ and $\partial\mathbf{R}/\partial\mathcal{X}$. The partial derivative of the objective function can be obtained by hand linearization, as described in Section 5.1.2. The partial derivative with respect to the residual vector may be obtained using complex step. A more accurate gradient may also improve the efficiency of the optimizer, leading to a more complete convergence of the optimizer.

7. Finally, a study of different strategies for geometry control may be needed to examine their fidelity (how many design variables are needed for comparable resolution), effectiveness (how well the different methods are able to capture an aerodynamic optimum) and efficiency (how the methods affect the optimizer convergence).

References

- [1] *The environmental effects of civil aircraft in flight*, special report, Royal Commission on Environmental Pollution, 2002.
- [2] *Airline fuel and labour cost share*, IATA economic briefing, International Air Transportation Association, 2007.
- [3] M. J. AFTOSMIS, M. J. BERGER, AND J. E. MELTON, *Robust and efficient Cartesian mesh generation for component-based geometry*, AIAA Paper 97-0196, American Institute of Aeronautics and Astronautics, 1997.
- [4] J. D. ANDERSON, *Fundamentals of Aerodynamics*, McGraw Hill, 1984.
- [5] ———, *A History of Aerodynamics and Its Impact on Flying Machines*, Cambridge University Press, 1997.
- [6] W. K. ANDERSON, W. D. GROPP, D. K. KAUSHIK, D. E. KEYES, AND B. F. SMITH, *Achieving high sustained performance in an unstructured mesh CFD application*, in Proceedings of Supercomputing '99, 1999.
- [7] W. K. ANDERSON, S. L. KARMAN, AND C. BURDYSHAW, *Geometry parameterization using control grid*, AIAA Paper 2008-6028, American Institute of Aeronautics and Astronautics, September 2008.
- [8] W. K. ANDERSON, J. C. NEWMAN, D. L. WHITFIELD, AND E. J. NIELSEN, *Sensitivity analysis for the Navier-Stokes equations on unstructured meshes using complex variables*, AIAA Paper 99-3294, American Institute of Aeronautics and Astronautics, 1999.
- [9] W. K. ANDERSON AND V. VENKATAKRISHNAN, *Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation*, AIAA Paper 97-0643, American Institute of Aeronautics and Astronautics, 1997.
- [10] T. J. BARTH AND S. W. LINTON, *An unstructured mesh Newton solver for compressible fluid flow and its implementation*, AIAA Report 95-0221, American Institute for Aeronautics and Astronautics, 1995.
- [11] G. BIROS AND O. GHATTAS, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver*, SIAM Journal on Scientific Computing, 27 (2005), pp. 687–713.
- [12] ———, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange-Newton solver and its application to optimal control of steady viscous flows*, SIAM Journal on Scientific Computing, 27 (2005), pp. 714–739.

- [13] C. BISCHOF, A. CARLE, G. CORLISS, A. GRIEWANK, AND P. HOVLAND, *ADIFOR—generating derivative codes from Fortran programs*, Scientific Programming, (1991), pp. 1–29.
- [14] C. H. BISCHOF, A. MAUER, W. T. JONES, AND J. SAMAREH, *Experiences with automatic differentiation applied to a volume grid generation code*, Journal of Aircraft, 35 (1998).
- [15] L. BREGUET, *Aerodynamical efficiency and the reduction of air transport costs*, Aeronautical Journal, 26 (1922), pp. 307–313.
- [16] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM Journal of Scientific and Statistical Computing, 11 (1984), pp. 450–481.
- [17] C. G. BROYDEN, *The convergence of a class of double-rank minimization algorithms*, Journal of the Institute of Mathematics and its Applications, 6 (1970), pp. 76–90.
- [18] H. BUCKLEY AND D. W. ZINGG, *Airfoil optimization using practical aerodynamic design requirements*, AIAA Paper 2009–3516, American Institute of Aeronautics and Astronautics, 2009.
- [19] G. W. BURGREN AND O. BAYSAL, *Three-dimensional aerodynamic shape optimization using discrete sensitivity analysis*, AIAA Journal, 32 (1996), pp. 1761–1770.
- [20] X.-C. CAI, W. D. GROPP, D. E. KEYES, AND M. D. TIDRIRI, *Parallel implicit methods for aerodynamics*, in Proceedings of the 7th International Conference on Domain Decomposition, D. E. Keyes and J. Xu, eds., 1995, pp. 465–470.
- [21] L. A. CARLSON, *Transonic airfoil design using Cartesian coordinates*, NASA CR 2578, National Aeronautics and Space Administration, 1976.
- [22] M. H. CARPENTER, D. GOTTLIEB, AND S. ABARBANEL, *Time-stable boundary conditions for finite difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes*, Journal of Computational Physics, 111 (1994), pp. 220–236.
- [23] M. H. CARPENTER, J. NORDSTRÖM, AND D. GOTTLIEB, *A stable and conservative interface treatment of arbitrary spatial accuracy*, Journal of Computational Physics, 148 (1999), pp. 341–356.
- [24] G. CARPENTIERI, M. J. L. VAN TOOREN, AND B. KOREN, *Improving the efficiency of aerodynamic shape optimization on unstructured meshes*, AIAA Paper 2006-298, American Institute of Aeronautics and Astronautics, Reno, NV, 2006.
- [25] P. CASTONGUAY AND S. K. NADARAJAH, *Effects of shape parameterization on aerodynamic shape optimization*, AIAA Paper 2007-59, American Institute of Aeronautics and Astronautics, January 2007.
- [26] T. T. CHISHOLM AND D. W. ZINGG, *A Newton-Krylov algorithm for turbulent aerodynamic flows*, AIAA Paper 2003-0071, American Institute of Aeronautics and Astronautics, 2003.

- [27] S. CHOI, J. J. ALONSO, I. M. KROO, AND M. WINTZER, *Multi-fidelity design optimization of low-boom supersonic business jets*, AIAA Paper 2004-4371, American Institute of Aeronautics and Astronautics, 2004.
- [28] S. E. CLIFF, J. J. REUTHER, D. A. SAUNDERS, AND R. M. HICKS, *Single-point and multipoint aerodynamic shape optimization of high-speed civil transport*, *Journal of Aircraft*, 38 (2001), pp. 997–1005.
- [29] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, *SIAM Journal of Numerical Analysis*, 19 (1982), pp. 400–408.
- [30] Y. DEREMAUX, K. WILLCOX, AND R. HAIMES, *Physically-based, real-time visualization and constraint analysis in multidisciplinary design optimization*, AIAA Paper 2003-3876, American Institute of Aeronautics and Astronautics, 2003.
- [31] J.-A. DÉSIDÉRI AND A. JANKA, *Multilevel shape parameterization for aerodynamic optimization - application to drag and noise reduction of transonic/supersonic business jet*, in *European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004*, Jyväskylä, Finland, July 2004.
- [32] J. DRIVER AND D. W. ZINGG, *Numerical aerodynamic optimization incorporating laminar-turbulent transition prediction*, *AIAA Journal*, 45 (2007), pp. 1810–1816.
- [33] R. DUVIGNEAU AND M. VISONNEAU, *Shape optimization of incompressible and turbulent flows using the simplex method*, AIAA Paper 2001-2533, American Institute of Aeronautics and Astronautics, June 2001.
- [34] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, *SIAM Journal of Scientific Computing*, 17 (1996), pp. 16–32.
- [35] J. ELLIOTT AND J. PERAIRE, *Aerodynamic optimization on unstructured meshes with viscous effects*, AIAA Report 97-1849, American Institute of Aeronautics and Astronautics, Snowmass Village, CO, USA, June 1997.
- [36] ———, *Practical three-dimensional aerodynamic design and optimization using unstructured meshes*, *AIAA Journal*, 35 (1997), pp. 1479–1485.
- [37] ———, *Constrained, multipoint shape optimization for complex 3D configurations*, *Aeronautical Journal*, 102 (1998).
- [38] B. EPSTEIN, A. JAMESON, S. PEIGIN, D. ROMAN, N. HARRISON, AND J. VASSBERG, *Comparative study of three-dimensional wing drag minimization by different optimization techniques*, *Journal of Aircraft*, 46 (2008), pp. 526–541.
- [39] B. EPSTEIN AND S. PEIGIN, *Optimization of 3D wings based on Navier-Stokes solutions and genetic algorithms*, *International Journal of Computational Fluid Dynamics*, 20 (2006), pp. 75–92.
- [40] C. FARHAT, C. DEGAND, B. KOOBUS, AND M. LESOINNE, *Torsional springs for two-dimensional dynamic unstructured fluid meshes*, *Computer Methods in Applied Mechanics and Engineering*, 163 (1998), pp. 231–245.

- [41] I. FEJTEK, D. JONES, G. WALLER, E. HANSEN, AND S. OBAYASHI, *A transonic wing inverse design capability for complete aircraft configurations*, AIAA Paper 2001-2443, American Institute of Aeronautics and Astronautics, Anaheim, CA, USA, 2001.
- [42] R. FLETCHER, *A new approach to variable metric algorithms*, Computer Journal, 13 (1970), pp. 317–322.
- [43] P. D. FRANK AND G. R. SHUBIN, *A comparison of optimization-based approaches for a model computational aerodynamic design problem*, Journal of Computational Physics, 98 (1992), pp. 74–89.
- [44] D. FUDGE, *A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization*, Master’s thesis, University of Toronto, 2004.
- [45] D. FUDGE, D. W. ZINGG, AND R. HAIMES, *A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization*, AIAA Paper 2005-0451, American Institute of Aeronautics and Astronautics, January 2005.
- [46] J. GATSIS AND D. W. ZINGG, *A fully-coupled Newton-Krylov algorithm for aerodynamic optimization*, AIAA Paper 2003-3956, American Institute of Aeronautics and Astronautics, 2003.
- [47] P. GEUZAIN, I. LEPOT, F. MEERS, AND J.-A. ESSERS, *Multilevel Newton-Krylov algorithms for computing compressible flows on unstructured meshes*, AIAA Report 99-3341, American Institute for Aeronautics and Astronautics, 1999.
- [48] M. B. GILES AND M. DRELA, *Two-dimensional transonic aerodynamic design method*, AIAA Journal, 25 (1987), pp. 1199–1206.
- [49] M. B. GILES AND N. A. PIERCE, *In introduction to the adjoint approach design*, Flow, Turbulence and Combustion, 65 (2000), pp. 393–415.
- [50] P. E. GILL, W. MURRAY, AND M. A. SAUNDERS, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Review, 47 (2005), pp. 99–131.
- [51] P. E. GILL, W. MURRAY, M. A. SAUNDERS, AND W. A. WRIGHT, *User’s guide for NPSOL: A FORTRAN package for nonlinear programming*, Dept. of Operational Research TR SOL86-2, Stanford University, Stanford, CA, USA, 1986.
- [52] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman, Boston, MA, USA, 1989.
- [53] D. GOLDFARB, *A family of variable metric updates derived by variational means*, Mathematics of Computation, 24 (1970), pp. 23–26.
- [54] A. GRIEWANK, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics, SIAM, 2000.
- [55] W. GROPP, E. LUSK, AND A. SKJELLUM, *Using MPI: portable parallel programming with the message-passing interface*, MIT Press, Cambridge, Mass, second ed., 1999.
- [56] W. D. GROPP, D. E. KEYES, AND M. D. TIDRIRI, *Parallel implicit solvers for steady, compressible aerodynamics*, in Parallel Computational Fluid Dynamics: New Trends and Advances, A. Ecer et al., eds., 1993, pp. 391–399.

- [57] C. P. T. GROTH AND S. A. NORTHRUP, *Parallel implicit adaptive mesh refinement scheme for body-fitted multi-block mesh*, AIAA Paper 2005-5333, June 2005.
- [58] C. R. GUMBERT, P. A. NEWMAN, AND G. J.-W. HOU, *Simultaneous aerodynamic analysis and design optimization (SAADO) for a 3-D flexible wing*, AIAA Paper 2001-1107, Reno, NV, USA, January 2001.
- [59] M. D. GUNZBURGER, *Inverse design and optimization methods*. The Von Karman Institute For Fluid Dynamics Lecture Series 1997-05, April 1997.
- [60] R. HAIMES, *CAPRI: Master-model Interface Specification*, MIT, Revision 2.10, December 2004.
- [61] ———, *CAPRI: Solid Modeling Based Infrastructure for Engineering Analysis and Design*, MIT, Revision 2.10, December 2004.
- [62] D. HALL, *Three-dimensional elliptic grid generator*, Master's thesis, University of Toronto, 1992.
- [63] L. HASCOËT AND V. PASCUAL, *TAPENADE 2.1 user's guide*, Technical Report 0300, INRIA, France, 2004.
- [64] L. HASCOËT, M. VÁZQUEZ, AND A. DERVIEUX, *Automatic differentiation for optimum design, applied to sonic boom reduction*, in Proceedings of the International Conference on Computational Science and its Applications, ICCSA'03, Montreal, Canada, LNCS 2668, Springer, 2003, pp. 85–94.
- [65] S. B. HAZRA, *Direct treatment of state constraints in aerodynamic shape optimization using simultaneous pseudo-time-stepping*, AIAA Journal, 45 (2007), pp. 1988–1996.
- [66] ———, *Multigrid one-shot method for aerodynamic shape optimization*, SIAM Journal of Scientific Computing, 30 (2008), pp. 1527–1547.
- [67] S. B. HAZRA AND A. JAMESON, *One-shot pseudo-time method for aerodynamic shape optimization using the Navier-Stokes equations*, AIAA Paper 2007-1470, American Institute of Aeronautics and Astronautics, Reno, Nevada, USA, 2007.
- [68] P. A. HENNE, *Inverse transonic wing design method*, Journal of Aircraft, 18 (1981), pp. 121–127.
- [69] J. E. HICKEN, *Efficient Algorithms for Future Aircraft Design: Contributions to Aerodynamic Shape Optimization*, PhD thesis, University of Toronto, 2009.
- [70] J. E. HICKEN AND D. W. ZINGG, *A parallel Newton-Krylov flow solver for the Euler equations on multi-block grids*, AIAA Paper 2007-4333, American Institute of Aeronautics and Astronautics, 2007.
- [71] ———, *Integrated parameterization and grid movement using B-spline meshes*, AIAA Paper 2008-6079, American Institute of Aeronautics and Astronautics, September 2008.
- [72] ———, *An investigation of induced drag minimization using a Newton-Krylov algorithm*, AIAA Report 2008-5807, American Institute for Aeronautics and Astronautics, Victoria, BC, 2008.

- [73] ———, *Parallel Newton-Krylov solver for the Euler equations discretized using simultaneous-approximation terms*, AIAA Journal, 46 (2008), pp. 2773–2786.
- [74] R. M. HICKS AND P. A. HENNE, *Wing design by numerical optimization*, Journal of Aircraft, 15 (1978), pp. 407–412.
- [75] R. M. HICKS, E. M. MURMAN, AND G. N. VANDERPLAATS, *An assessment of airfoil design by numerical optimization*, NASA TM X-3092, NASA, July 1974.
- [76] T. L. HOLST AND T. H. PULLIAM, *Transonic wing shape optimization using a genetic algorithm*, Fluid Mechanics and its Applications, 73 (2003), pp. 245–252.
- [77] J. HUA, F. M. FONG, P. YANG JAY LIU, AND D. W. ZINGG, *Optimization of long-endurance airfoils*, AIAA Paper 2003-3500, American Institute of Aeronautics and Astronautics, 2003.
- [78] INTERNATIONAL AIR TRANSPORT ASSOCIATION, *Jet fuel price development*. http://www.iata.org/whatwedo/economics/fuel_monitor/price_development.htm, 2008. [Online; accessed 30-April-2008].
- [79] A. IOLLO, G. KURUVILA, AND S. TA’ASAN, *Pseudo-time method for optimal shape design using the Euler equations*, NASA CR 198205, 1995.
- [80] A. JAMESON, *Aerodynamic design via control theory*, Journal of Scientific Computing, 3 (1988), pp. 233–260.
- [81] A. JAMESON AND S. KIM, *Reduction of the adjoint gradient formula in the continuous limit*, AIAA Paper 2003-0040, American Institute of Aeronautics and Astronautics, 2003.
- [82] A. JAMESON, K. LEOVIRYAKIT, AND S. SHANKARAN, *Multi-point aero-structural optimization of wings including planform variations*, AIAA Paper 2007-764, American Institute of Aeronautics and Astronautics, 2007.
- [83] A. JAMESON, L. MARTINELLI, AND N. PIERCE, *Optimum aerodynamic design using the Navier-Stokes equations*, Theoretical Fluid Dynamics, 10 (1998), pp. 213–237.
- [84] 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, Palo Alto, CA, 1981. AIAA Paper 81-1259.
- [85] A. JAMESON, SRIRIAM, L. MARTINELLI, AND B. HAIMES, *Aerodynamic shape optimization of complete aircraft configurations using a unstructured grids*, AIAA Paper 2004-0533, American Institute of Aeronautics and Astronautics, 2004.
- [86] A. JAMESON, J. C. VASSBERG, AND S. SHANKARAN, *Aerodynamic-structural design studies of low-sweep transonic wings*, AIAA Paper 2008-145, American Institute of Aeronautics and Astronautics, January 2008.
- [87] Z. JOHANN, T. J. R. HUGHES, AND F. SHAKIB, *A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids*, Computational Methods in Applied Mechanics and Engineering, 87 (1991), pp. 281–304.

- [88] D. E. KEYES, *Aerodynamic applications of Newton-Krylov-Schwarz solvers*, in Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics, M. Deshpande, S. Desai, and R. Narasimha, eds., New York, 1995, Springer, pp. 1–20.
- [89] S. KIM, K. HOSSEINI, K. LEOVIRIYAKIT, AND A. JAMESON, *Enhancement of adjoint design methods via optimization of adjoint parameters*, AIAA Paper 2005-448, American Institute of Aeronautics and Astronautics, Reno, NV, 2005.
- [90] 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.
- [91] D. A. KNOLL AND W. J. RIDER, *A multigrid preconditioned Newton-Krylov method*, SIAM Journal on Scientific Computing, 21 (1999), pp. 691–710.
- [92] B. M. KULFAN, *Universal parametric geometry representation method*, Journal of Aircraft, 45 (2008), pp. 142–158.
- [93] S. T. LEDOUX, W. W. HERLING, J. FATTA, AND R. R. RATCLIFF, *Multidisciplinary design optimization system using higher order analysis code*, AIAA Report 2004-4567, American Institute for Aeronautics and Astronautics, 2004.
- [94] E. M. LEE-RAUSCH, N. T. FRINK, D. J. MAVRIPLIS, R. D. RAUSCH, AND W. E. MILHOLEN, *Drag prediction on a DLR-F6 transport configuration using unstructured grid solvers*, AIAA Paper 2004-0000, American Institute of Aeronautics and Astronautics, January 2004.
- [95] E. M. LEE-RAUSCH, M. A. PARK, W. T. JONES, AND E. J. NIELSEN, *Application of parallel adjoint-based error estimation and anisotropic grid adaptation for three-dimensional aerospace configurations*, AIAA Paper 2005-4842, American Institute of Aeronautics and Astronautics, 2005.
- [96] K. LEOVIRIYAKIT AND A. JAMESON, *Multipoint wing planform optimization via control theory*, AIAA Paper 2005-0450, American Institute of Aeronautics and Astronautics, Reno, NV, 2005.
- [97] J. LEPINE, F. GUIBAULT, AND J.-Y. TREPANIER, *Optimized nonuniform rational B-spline geometrical representation for aerodynamic design of wings*, AIAA Journal, 39 (2001), pp. 2033–2041.
- [98] T. M. LEUNG AND D. W. ZINGG, *A Newton-Krylov approach for aerodynamic shape optimization of wings*, AIAA Paper 2008-5806, American Institute of Aeronautics and Astronautics, September 2008.
- [99] —, *Aerodynamic shape optimization of wings at transonic speeds*, in 56th CASI Aeronautics Conference and Annual General Meeting, Kanata, ON, May 2009.
- [100] —, *High-fidelity aerodynamic shape optimization using a parallel Newton-Krylov approach*, Paper CFDSC2009-4A1, CFD Society of Canada, Kanata, ON, May 2009.
- [101] —, *Single- and multi-point aerodynamic shape optimization using a parallel Newton-Krylov approach*, AIAA Paper 2009-3803, American Institute of Aeronautics and Astronautics, June 2009.

- [102] R. LIEBECK, *A class of airfoils designed for high lift in incompressible flow*, Journal of Aircraft, 10 (1973), pp. 610–617.
- [103] ———, *Design of the blended-wing-body subsonic transport*, Journal of Aircraft, 41 (2004).
- [104] M. J. LIGHTHILL, *A new method of two-dimensional aerodynamic design*, R&M 2112, Aeronautical Research Council, June 1945.
- [105] P.-Y. J. LIU, *Comparison of optimization algorithms applied to aerodynamic design*, Master’s thesis, University of Toronto, 2003.
- [106] H. LOMAX, T. H. PULLIAM, AND D. W. ZINGG, *Fundamentals of Computational Fluid Dynamics*, Springer-Verlag, 2001.
- [107] C. A. MADER, J. R. R. A. MARTINS, J. J. ALONSO, AND E. VAN DER WEIDE, *ADjoint: An approach for the rapid development of discrete adjoint solvers*, AIAA Journal, 46 (2008), pp. 863–873.
- [108] D. G. MARTINEAU AND J. M. GEORGALA, *A mesh movement algorithm for high quality generalized meshes*, AIAA Paper 2004-0614, American Institute of Aeronautics and Astronautics, 2004.
- [109] M. MARTINELLI AND F. BEUX, *Multi-level gradient-based methods and parametrization in aerodynamic shape design*, European Journal of Computational Mechanics, 17 (2008), pp. 169–197.
- [110] J. R. R. A. MARTINS, J. J. ALONSO, AND J. J. REUTHER, *High-fidelity aerostructural design optimization of a supersonic business jet*, Journal of Aircraft, 41 (2004), pp. 523–530.
- [111] ———, *A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design*, Optimization and Engineering, 6 (2005), pp. 33–62.
- [112] J. R. R. A. MARTINS, I. A. KROO, AND J. J. ALONSO, *An automated method for sensitivity analysis using complex-step variables*, AIAA Paper 2000-0689, American Institute of Aeronautics and Astronautics, January 2000.
- [113] J. R. R. A. MARTINS, P. STURDZA, AND J. J. ALONSO, *The complex-step derivative approximation*, ACM Trans. Math. Softw., 29 (2003).
- [114] J. D. MATTINGLY, *Elements of propulsion: gas turbines and rockets*, American Institute of Aeronautics and Astronautics, Reston, VA, 2006.
- [115] K. MATTSSON, M. SVÄRD, AND J. NORDSTRÖM, *Stable and accurate artificial dissipation*, Journal of Scientific Computing, 21 (2004), pp. 57–79.
- [116] D. MAVRIPLIS, *Formulation and multigrid solution of the discrete adjoint for optimization problems on unstructured meshes*, AIAA Paper 2005-0319, American Institute of Aeronautics and Astronautics, 2005.
- [117] ———, *A discrete adjoint for optimization problems on three-dimensional unstructured meshes*, AIAA Paper 2006-50, American Institute of Aeronautics and Astronautics, 2006.

- [118] G. MAY, E. VAN DER WEIDE, A. JAMESON, AND L. MARTINELLI, *Drag prediction of the DLR-F6 configuration*, AIAA Paper 2004-0396, American Institute of Aeronautics and Astronautics, January 2004.
- [119] J. H. MCMASTERS, *A U.S. perspective on future commercial airliner design*, tech. rep., von Karman Institute for Fluid Dynamics, June 2005.
- [120] B. MOHAMMADI, *A new optimal shape design procedure for inviscid and viscous turbulent flows*, International Journal for Numerical Methods in Fluids, 25 (1997), pp. 183–203.
- [121] C. MORAWETZ, *On the non-existence of continuous transonic flows past profiles*, Communications of Pure and Applied Mathematics, 9 (1956), pp. 45–48.
- [122] A. MOUSAVI, P. CASTONGUAY, AND S. K. NADARAJAH, *Survey of shape parameterization techniques and its effect on three-dimensional aerodynamic shape optimization*, AIAA Paper 2007-3837, American Institute of Aeronautics and Astronautics, June 2007.
- [123] W. A. MULDER AND B. VAN LEER, *Experiments with implicit upwind methods for the Euler equations*, Journal of Computational Physics, 59 (1985), pp. 232–246.
- [124] S. NADARAJAH AND A. JAMESON, *A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization*, AIAA Paper 2000-0667, American Institute of Aeronautics and Astronautics, 2000.
- [125] S. K. NADARAJAH, *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*, PhD thesis, Stanford University, 2003.
- [126] S. K. NADARAJAH, A. JAMESON, AND J. J. ALONSO, *Sonic boom reduction using an adjoint method for wing-body configurations in supersonic flow*, AIAA Paper 2002-5547, American Institute of Aeronautics and Astronautics, 2002.
- [127] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, The Computer Journal, 7 (1965), pp. 308–313.
- [128] M. NEMEC, *Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach*, PhD thesis, University of Toronto, 2003.
- [129] M. NEMEC AND M. J. AFTOSMIS, *Adjoint algorithm for CAD-based shape optimization using a Cartesian method*, AIAA Paper 2005-4987, American Institute of Aeronautics and Astronautics, June 2005.
- [130] —, *Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes*, AIAA Paper 2007-4187, American Institute of Aeronautics and Astronautics, 2007.
- [131] M. NEMEC, M. J. AFTOSMIS, AND T. H. PULLIAM, *CAD-based aerodynamic design of complex configurations using a Cartesian method*, AIAA Paper 2004-0113, American Institute of Aeronautics and Astronautics, 2004.
- [132] M. NEMEC AND D. W. ZINGG, *Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations*, AIAA Journal, 40 (2002), pp. 1146–1154.
- [133] —, *Optimization of high-lift configurations using a Newton-Krylov algorithm*, AIAA Paper 2003-3957, American Institute of Aeronautics and Astronautics, June 2003.

- [134] M. NEMEC, D. W. ZINGG, AND T. PULLIAM, *Multi-point and multi-objective aerodynamic shape optimization*, AIAA Paper 2003-5548, American Institute of Aeronautics and Astronautics, June 2003.
- [135] ———, *Multipoint and multi-objective aerodynamic shape optimization*, AIAA Journal, 42 (2004), pp. 1057–1065.
- [136] NEW MEXICO INSTITUTE OF MINING AND TECHNOLOGY, *NM pricesheet*. <http://octane.nmt.edu/gotech/Marketplace/Prices.aspx>, 2008. [Online; accessed 30-April-2008].
- [137] J. C. NEWMAN, W. K. ANDERSON, AND D. L. WHITFIELD, *Multidisciplinary sensitivity derivatives using complex variables*, Report MSSU-COE-ERC-98-08, July 1998.
- [138] J. C. NEWMAN, D. L. WHITFIELD, AND W. K. ANDERSON, *Step-size independent approach for multidisciplinary sensitivity analysis*, Journal of Aircraft, 40 (2003).
- [139] J. NICHOLS AND D. W. ZINGG, *A three-dimensional multi-block Newton-Krylov flow solver for the Euler equations*, AIAA Paper 2005-5230, American Institute of Aeronautics and Astronautics, 2005.
- [140] E. NIELSEN AND W. K. ANDERSON, *Recent improvements in aerodynamic design optimization on unstructured meshes*, AIAA Journal, 40 (2002), pp. 1155–1163.
- [141] E. J. NIELSEN AND B. KLEB, *Efficient construction of discrete adjoint operators on unstructured grids by using complex variables*, AIAA Paper 2005-0324, American Institute of Aeronautics and Astronautics, 2005.
- [142] E. J. NIELSEN AND W. L. KLEB, *Efficient construction of discrete adjoint operators on unstructured grids using complex variables*, AIAA Journal, 44 (2006), pp. 827–836.
- [143] E. J. NIELSEN, R. W. WALTERS, W. K. ANDERSON, AND D. E. KEYES, *Application of Newton-Krylov methodology to a three-dimensional Euler code*, AIAA Paper 99-1733, American Institute of Aeronautics and Astronautics, 1995.
- [144] J. NOCEDAL, *Updating quasi-Newton matrices with limited storage*, Mathematics of Computation, 35 (1980), pp. 773–782.
- [145] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer-Verlag, 1999.
- [146] S. ODAYASHI, *Aerodynamic optimization with evolutionary algorithms*, tech. rep., Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Inverse Design and Optimizations, Brussels, Belgium, 1997.
- [147] C. OLDFIELD, *An adjoint method augmented with grid sensitivities for aerodynamic optimization*, Master’s thesis, University of Toronto, 2006.
- [148] A. OYAMA, *Wing Design Using Evolutionary Algorithms*, PhD thesis, Tohoku University, 2000.
- [149] S. PEIGIN AND B. EPSTEIN, *Robust drag minimization of aerodynamic wings in engineering environment*, Journal of Aircraft, 43 (2006), pp. 582–594.

- [150] J. E. PENNER ET AL., *Aviation and the Global Atmosphere: A Special Report of IPCC Working Groups I and III in Collaboration with the Scientific Assessment Panel to the Montreal Protocol on Substances that Deplete the Ozone Layer*, Cambridge University Press, New York, 1999.
- [151] O. PIRONNEAU, *Optimal Shape Design for Elliptic Systems*, Springer-Verlag, 1983.
- [152] A. PUEYO AND D. W. ZINGG, *Efficient Newton-Krylov solver for aerodynamic computations*, AIAA Journal, 36 (1998), pp. 1991–1997.
- [153] 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, Brussels, Belgium, Jan. 1986.
- [154] T. H. PULLIAM AND J. STEGER, *Implicit finite-difference simulations of three-dimensional compressible flow*, AIAA Journal, 18 (1980), pp. 159–167.
- [155] D. P. RAYMER, *Aircraft Design: A Conceptual Approach*, American Institute of Aeronautics and Astronautics, Reston, VA, 4th ed., 2006.
- [156] J. J. REUTHER, A. JAMESON, J. J. ALONSO, M. J. RIMLINGER, AND D. SAUNDERS, *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1*, Journal of Aircraft, 26 (1999), pp. 51–60.
- [157] ———, *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2*, Journal of Aircraft, 26 (1999), pp. 61–74.
- [158] D. F. ROGERS AND J. A. ADAMS, *Mathematical Elements for Computer Graphics*, McGraw-Hill, 2 ed., 1990.
- [159] M. RUMPFKEIL AND D. ZINGG, *The optimal control of unsteady flows with a discrete adjoint method*, Optimization and Engineering, (2008).
- [160] ———, *Unsteady optimization using a discrete adjoint approach applied to aeroacoustic shape design*, AIAA Paper 2008-18, American Institute of Aeronautics and Astronautics, Reno, NV, USA, January 2008.
- [161] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific Computing, 14 (1993), pp. 461–469.
- [162] ———, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, second ed., 2003.
- [163] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear problems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.
- [164] Y. SAAD AND M. SOSONKINA, *Distributed Schur complement preconditioning*, SIAM Journal for Scientific Computing, 21 (1999), pp. 1337–1356.
- [165] J. A. SAMAREH, *A novel shape parameterization approach*, Report NASA/TM-1999-209116, NASA, May 1999.

- [166] ———, *Status and future of geometry modeling and grid generation for design and optimization*, *Journal of Aircraft*, 36 (1999), pp. 97–104.
- [167] ———, *Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization*, *AIAA Journal*, 36 (1999), pp. 97–104.
- [168] D. SASAKI, M. MORIKAWA, S. OBAYASHI, AND K. NAKAHASHI, *Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms*, in *Lecture Notes in Computer Science*, Springer, 2001, pp. 639–652.
- [169] V. SCHMITT AND F. CHARPIN, *Pressure distributions on the ONERA-M6-wing at transonic Mach numbers*, report of the fluid dynamics panel working group 04, AGARD AR 138, May 1979.
- [170] H. A. SCHWARZ, *Ueber einen grenzübergang durch alternirendes verfahren*, in *Gesammelte Mathematische Abhandlungen*, vol. 2, Springer-Verlag, Berlin, 1890, pp. 133–143.
- [171] V. SEIDL, M. PERIĆ, AND S. SCHMIDT, *Space- and time-parallel Navier-Stokes solver for 3d block-adaptive Cartesian grids*, in *Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers*, A. Ecer et al., eds., 1995, pp. 577–584.
- [172] F. SHAKIB, *Finite-Element Analysis of the Compressible Euler and Navier-Stokes Equations*, PhD thesis, Stanford University, 1988.
- [173] D. F. SHANNO, *Conditioning of quasi-newton methods for function minimization*, *Mathematics of Computation*, 24 (1970), pp. 647–656.
- [174] R. S. SILVA AND R. C. ALMEIDA, *Performance of an Euler solver using a distributed system*, in *Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers*, A. Ecer et al., eds., 1995, pp. 175–182.
- [175] H. SOBIECZKY, *Parametric airfoils and wings*, *Notes on Numerical Fluid Mechanics*, 68 (1998), pp. 71–88.
- [176] P. R. SPALART AND S. R. ALLMARAS, *A one-equation turbulence model for aerodynamic flows*, *AIAA Paper 92-0439*, American Institute of Aeronautics and Astronautics, January 1992.
- [177] T. L. STERLING, *Beowulf Cluster Computing with Linux*, MIT Press, Cambridge, Massachusetts, 2002.
- [178] P. STURDZA, *An Aerodynamic Design Method for Supersonic Natural Lamina Flow Aircraft*, PhD thesis, Stanford University, December 2003.
- [179] S. TA’ASAN AND M. D. S. G. KURUVILA, *Aerodynamic design and optimization in one shot*, *AIAA Paper 92-0025*, American Institute of Aeronautics and Astronautics, Reno, NV, USA, Jan. 1992.
- [180] J. F. THOMPSON, B. K. SONI, AND N. P. WEATHERILL, eds., *Handbook of Grid Generation*, CRC Press, 1999.
- [181] T. L. TRANEN, *A rapid computer-aided transonic airfoil design method*, *AIAA Paper 74-501*, American Institute of Aeronautics and Astronautics, Palo Alto, CA, June 1974.

- [182] A. TRUONG, *Development of grid movement algorithms suitable for aerodynamic optimization*, Master's thesis, University of Toronto, 2005.
- [183] A. H. TRUONG, C. OLDFIELD, AND D. W. ZINGG, *Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization*, AIAA Paper 2007-3952, American Institute of Aeronautics and Astronautics, 2007.
- [184] G. N. VANDERPLAATS, *Efficient algorithm for numerical airfoil optimization*, *Journal of Aircraft*, 16 (1979), pp. 842–847.
- [185] J. C. VASSBERG, E. N. TINOCO, M. MANI, O. P. BRODERSEN, B. EISFELD, R. A. WAHLS, J. H. MORRISON, T. ZICKUHR, K. R. LAFLIN, AND D. MAVRIPLIS, *Summary of the third AIAA CFD drag prediction workshop*, AIAA Paper 2007-260, American Institute of Aeronautics and Astronautics, Reno, Nevada, 2007.
- [186] M. VÀZQUEZ, A. DERVIEUX, AND B. KOOBUS, *A methodology for the shape optimization of flexible wings*, *Engineering Computations*, 23 (2006), pp. 344–367.
- [187] V. VENKATAKRISHNAN, *Parallel implicit unstructured grid Euler solvers*, Report 1994-4, Institute for Computer Applications in Science and Engineering, 1994.
- [188] ———, *Implicit schemes and parallel computing in unstructured grid CFD*, Report 1995-28, Institute for Computer Applications in Science and Engineering, 1995.
- [189] V. VENKATAKRISHNAN AND D. J. MAVRIPLIS, *Implicit solvers for unstructured meshes*, *Journal of Computational Physics*, 105 (1993), pp. 83–91.
- [190] N. X. VINH, *Flight Mechanics of High-Performance Aircraft*, Cambridge University Press, New York, 1993.
- [191] S. WAKAYAMA AND I. KROO, *The challenge and promise of blended-wing-body configuration*, AIAA Paper 98-4736, American Institute of Aeronautics and Astronautics, 1998.
- [192] L. B. WIGTON, N. J. YU, AND D. P. YOUNG, *GMRES acceleration of computational fluid dynamics codes*, AIAA Report 85-1494, American Institute for Aeronautics and Astronautics, 1985.
- [193] WIKIPEDIA, *The Boeing company*. <http://active.boeing.com/commercial/orders/index.cfm>, 2008. [Online; last accessed 11-April-2008].
- [194] ———, *Oil price increases since 2003*. http://en.wikipedia.org/wiki/Oil_price_increases_since_2003, 2008. [Online; accessed 30-April-2008].
- [195] ———, *Airbus*. <http://en.wikipedia.org/wiki/Airbus>, 2009. [Online; last accessed 14 April, 2008].
- [196] P. WONG, *Aerodynamic optimization using the flow sensitivity approach*, Master's thesis, University of Toronto, 2001.
- [197] P. WONG AND D. ZINGG, *Three-dimensional aerodynamic computations on unstructured grids using a newton-krylov approach*, *Computers & Fluids*, 37 (2007), pp. 107–120.
- [198] G. A. WRENN, *An indirect method for numerical optimization using the kreisselmeiersteinhauser function*, NASA CP-4220, March 1989.

- [199] H.-Y. WU, S. YANG, F. LIU, AND H.-M. TSAI, *Comparison of three geometric representations of airfoils for aerodynamic optimization*, AIAA Paper 2003-4095, American Institute of Aeronautics and Astronautics, January 2003.
- [200] Z. ZHAO AND R. B. PELZ, *A parallel, globally-implicit Navier-Stokes solver for design*, in *Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers*, A. Ecer et al., eds., 1995, pp. 593–600.
- [201] D. ZINGG, M. NEMEC, AND T. PULLIAM, *A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization*, *Revue Européenne de Mécanique Numérique*, 17 (2008), pp. 103–126.
- [202] D. W. ZINGG AND L. BILLING, *Toward practical aerodynamic design through numerical optimization*, AIAA Paper 2007–3950, American Institute of Aeronautics and Astronautics, 2007.
- [203] D. W. ZINGG, S. DE RANGO, AND A. PUEYO, *Advances in algorithms for computing aerodynamic flows*, in *Frontiers in Computational Fluid Dynamics - 2000*, D. A. Caughey and M. M. Hafez, eds., World Scientific Publishing Company, 2000.
- [204] D. W. ZINGG AND S. ELIAS, *Aerodynamic optimization under a range of operating conditions*, *AIAA Journal*, 44 (2006), pp. 2787–2792.
- [205] D. W. ZINGG, T. M. LEUNG, L. DIOSADY, A. TRUONG, S. ELIAS, AND M. NEMEC, *Improvements to a Newton-Krylov adjoint algorithm for aerodynamic shape optimization*, AIAA Paper 2005-4857, American Institute of Aeronautics and Astronautics, September 2005.

Appendix A

Additional Notes on Range Equation

For an aircraft powered by turbojet or turbofan engines, the thrust specific fuel consumption (TSFC) is defined as the weight of fuel burned per thrust per unit time:

$$\text{TSFC} = \frac{\text{Weight of fuel burned per unit time (N/s)}}{\text{Unit thrust (N)}} \quad (\text{A.1})$$

For a specific power plant, TSFC is a function of the operating Mach number [114]. Based on this definition, the distance flown by an aircraft per unit weight of fuel burned can be calculated:

$$\frac{\text{distance (m)}}{\text{weight of fuel (N)}} = \frac{dR}{dW} = \frac{V}{\text{TSFC} \cdot T} = \frac{V}{\text{TSFC}} \frac{L}{D} \frac{1}{W} \quad (\text{A.2})$$

where T is the engine's thrust. This expression is based on assuming that during level flight, lift equals the weight of the aircraft, $L = W$, and drag equals the thrust, $D = T = \frac{WD}{L}$. To obtain the range equation requires integrating the above equation with respect to weight:

$$R = - \int_{W_i}^{W_f} \frac{V}{\text{TSFC}} \frac{L}{D} \frac{dW}{W} \quad (\text{A.3})$$

where W_i and W_f are the initial and final weights respectively. Note that this integral ignores climb and descent during take-off and landing. This assumption is valid for most civil transport aircraft, where level flying accounts for most of the duration of a flight.

The common form of the range equation, known as the Breguet range equation, is obtained by directly integrating (A.3), and assuming that only W varies during flight:

$$R = \frac{V}{\text{TSFC}} \frac{L}{D} \ln \left[\frac{W_i}{W_f} \right] \quad (\text{A.4})$$

The initial and final weights can be defined as:

$$\begin{aligned} W_i &= W_S + W_P + W_R + W_F \\ W_f &= W_S + W_P + W_R \end{aligned} \quad (\text{A.5})$$

where W_F , W_S , W_P and W_R are weights of fuel, structure, payload and reserve fuel respectively. Substituting these terms into W_i and W_f , (A.4) can be written as:

$$R = \frac{V}{\text{TSFC}} \frac{L}{D} \ln \left[1 + \frac{W_F}{W_P + W_S + W_R} \right] \quad (\text{A.6})$$

which is equation in (2.4). Based on this equation, the range can be maximized by:

- decreasing structural weight W_S ,
- increasing engine's efficiency by decreasing TSFC,
- flying as fast as possible, i.e. increasing V , and
- optimizing the aerodynamic efficiency (lift-to-drag ratio L/D) at that operating speed.

Note that in obtaining (A.4), the aircraft's speed V and the L/D ratio are assumed to be constant during the entire flight. The lift generated must also always equal to the weight. Therefore, as W decreases during flight, the lift generated must be decreased by reducing ρ (by increasing altitude). This would result in a *cruise-climb* flight profile. Generally, this type of flight profile is not permitted for transport aircraft, because of the desire of air-traffic controllers to keep all aircraft at a constant altitude and airspeed. The Concorde was the only exception that uses the cruise-climb flight profile, as its operating altitudes were higher than any other commercial aircraft. On longer flights, however, air traffic control may allow an aircraft to climb to a more optimal altitude in a "stairstep" flight profile as fuel is expended [155].

In the alternative derivation of the range equation, as described by Vinh [190], V is expressed as a function of the weight and lift coefficient, again assuming that $L = W$ during level flight:

$$L = W = \frac{1}{2} \rho V^2 S C_L \quad \longrightarrow \quad V = \sqrt{\frac{2W}{\rho S C_L}} \quad (\text{A.7})$$

Substituting this expression into (A.3) results in the following integral:

$$R = - \int_{W_i}^{W_f} \sqrt{\frac{2}{\rho S}} \frac{1}{\text{TSFC}} \frac{\sqrt{C_L}}{C_D} \frac{dW}{\sqrt{W}} \quad (\text{A.8})$$

Assuming again that only weight varies, the integral yields:

$$R = \frac{2}{\text{TSFC}} \sqrt{\frac{2}{\rho S}} \frac{\sqrt{C_L}}{C_D} \left(\sqrt{W_i} - \sqrt{W_f} \right) \quad (\text{A.9})$$

In this case, in obtaining (A.9), the aircraft's speed varies with the weight, while density ρ is assumed to be constant, i.e. constant altitude. Therefore, as W decreases during flight, the aircraft must slow down to decrease lift.

It should be noted that neither range equations, (A.4) nor (A.9), accurately represents an actual flight path of an aircraft. A more comprehensive characterization of an aircraft's flight profile is needed to calculate its range.

Appendix B

Alternative Parameterization Strategies

A geometry parameterization strategy based on B-spline control surfaces is described in Chapter 3. This appendix describes two notable alternative approaches, the Hicks-Henne bump function, and the discrete approach. Both methods can be easily adapted into the current Newton-Krylov algorithm. The advantages and drawbacks of each method are discussed.

B.1 Hicks-Henne Bump Functions

The sinusoidal bump functions were first introduced by Hicks *et al.* [75] for 2D airfoil optimization, and then in Hicks and Henne [74] for 3D wing optimization. This approach remains one of the most popular geometry parameterization methods. In this strategy, the changes from the original geometry z are described by a weighted sum of N bump functions:

$$z = z_{\text{basis}} + \sum_{j=1}^N a_j f_j(x) \quad (\text{B.1})$$

where z_{basis} is the z -coordinate of the original airfoil shape, and a_j are the coefficients, which are used as design variables. The bump functions f_j are defined in Wu *et al.* [199] as:

$$f_j(x) = \sin^4(\pi x^{m_j}), \quad m_j = \frac{\log 0.5}{\log x_{M_j}} \quad (\text{B.2})$$

The functions are defined for chord-wise coordinates $0 \leq x \leq 1$, and x_{M_j} are pre-selected values corresponding to the location where f_j is maximum. This equation is used in [199]:

$$x_{M_j} = 0.5 \times [1 - \cos \theta_j], \quad \theta_j = \frac{\pi j}{N + 1} \quad (\text{B.3})$$

It should be noted that this formulation of the bump functions can vary. For example, Castonquay and Nadarajah [25] use only a sine function rather than its fourth power, while Hicks

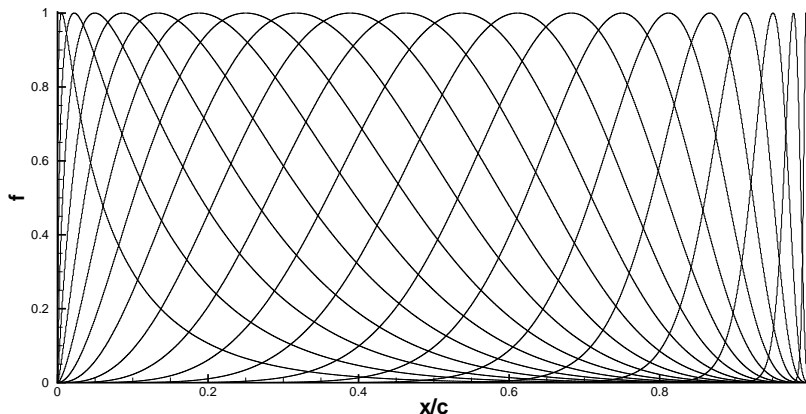


Figure B.1: A set of 20 Hicks-Henne bump functions

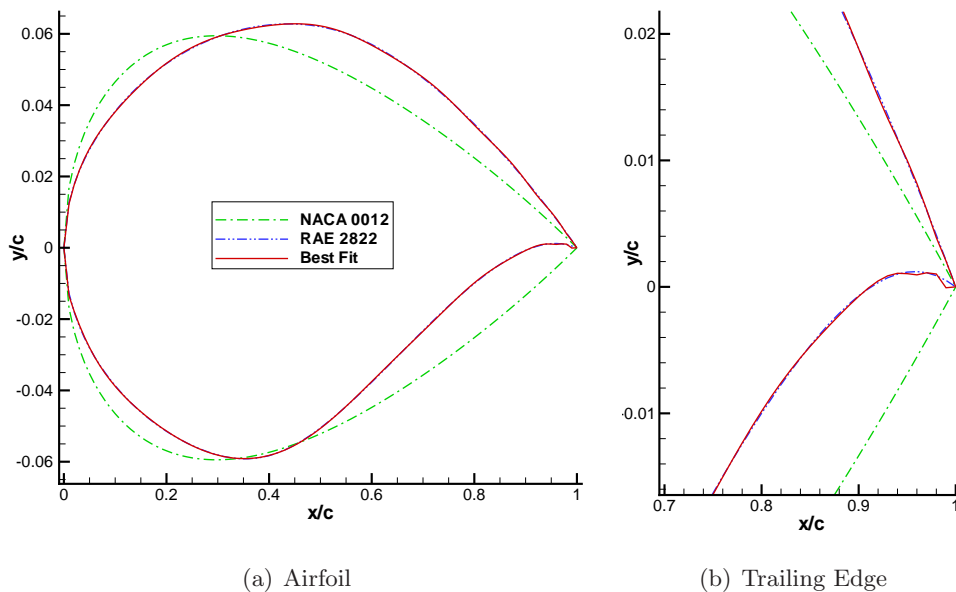


Figure B.2: Approximating an RAE 2822 airfoil using Hicks-Henne bump function

and Henne [74] only used five shape functions. Using the formulation in (B.2) and (B.3), the Hicks-Henne bump functions with $N = 20$ are shown in Figure B.1. In Figure B.2, an RAE 2822 airfoil is approximated as perturbations from a NACA 0012 airfoil using the bump functions described in Figure B.1. Twenty bump functions are used on each of the top and bottom surface of the airfoil.

The bump functions in Figure B.1 share similarities to the B-spline basis functions (Figure 3.1) in shapes. Also, both methods are able to control the airfoil shapes locally. In the case of the B-spline approach, because the control points have 3 degrees of freedom, it offers more flexibility than the Hicks-Henne functions. In the previous example, the Hick-Henne bump functions are not able to well represent the airfoil near the trailing edge.

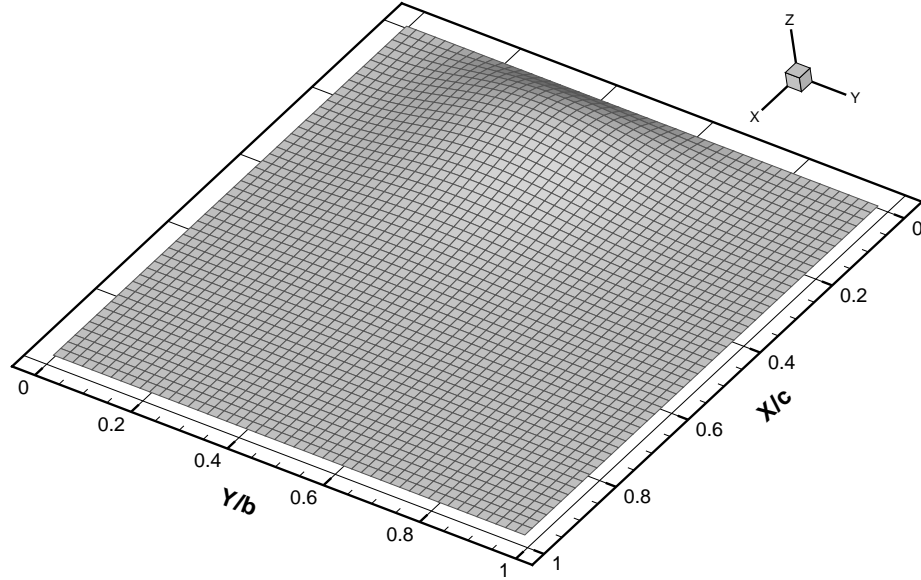


Figure B.3: Example of a surface perturbation using a 3D Hicks-Henne formulation.

It is possible to extend the Hicks-Henne functions to three-dimensions in a way that is analogous to extending B-spline curves to surfaces. In this formulation, the perturbations on the wing can be described as:

$$z = z_{\text{basis}} + \sum_{k=1}^N \sum_{j=1}^N a_{jk} f_j(x) g_k(y) \quad (\text{B.4})$$

To the author's knowledge, this formulation has not been used. An example is shown in Figure B.3. The surface is perturbed from a flat plate by setting the coefficient $a_{4,7} = 0.5$.

B.2 Discrete Approach

In the discrete approach—also known as the free surface approach—to geometry parameterization, all three coordinates (x, y, z) of each surface node can be treated as a design variable. This method is easy to implement, as it requires no explicit external parameterization scheme; thus it can be adapted to work with any existing body-fitted grids. The obvious advantage of this approach is that it gives the optimizer the highest amount of flexibility to describe a shape. Additionally, if the discrete approach is used with the algebraic grid movement method in Chapter 6, exact derivation of $\partial\mathcal{J}/\partial\mathcal{X}$ and $\partial\mathbf{R}/\partial\mathcal{X}$, required for the adjoint gradient calculations, can be simplified.

It is noted that the discrete approach, when used without modification, may result in non-smooth surface geometries during the optimization cycle. This may create problems for the flow

solver to converge to a steady-state solution, and a non-smooth geometry may be difficult to manufacture. In contrast, the B-spline curve is always smooth. If all three degrees of freedom of every surface node are used as design variables, the number of design variables may be very high. For example, consider the DPW-W1 grid used in the design examples. Using only the z -coordinates of the surface nodes as design variables, there will be more than 5600 design variables, compared to less than 200 with the B-spline control points. For large-scale viscous 3D problems, which may have over a million surface nodes, the approach may become prohibitively expensive. In this case, the cost is not associated with the cost of a gradient computation, since the cost of an adjoint gradient is nearly independent of the number of design variables. Rather, the cost is associated with the convergence of the optimizer. Some of these shortcomings are addressed by Jameson and Kim by projecting the gradient into a Sobolev space [81].

Appendix C

Derivation of the Adjoint Method Using Lagrange Multipliers

In the optimization problem in Section 2.1, a scalar objective function $\mathcal{J}(\mathcal{X}, \mathbf{Q})$ is minimized subject to the equality constraint $\mathbf{R}(\mathcal{X}, \mathbf{Q}) = \mathbf{0}$. Vector quantities \mathcal{X} (design variables), \mathbf{Q} (discretized flow variables), and the residual \mathbf{R} are highlighted in bold face following the notation used in Chapter 4. Note that since \mathbf{R} is a vector of size N (number of flow variables), the optimization problem is in fact constrained by N equations. The Lagrangian is defined as:

$$\mathcal{L} = \mathcal{J}(\mathcal{X}, \mathbf{Q}) + \boldsymbol{\Psi}^T \mathbf{R}(\mathcal{X}, \mathbf{Q}) \quad (\text{C.1})$$

where $\boldsymbol{\Psi}$ is the vector of Lagrange multipliers. For clarity, the Lagrangian is a scalar function of many variables:

$$\mathcal{L} = \mathcal{L}(\mathcal{X}_1, \dots, \mathcal{X}_{N_x}, \mathbf{Q}_1, \dots, \mathbf{Q}_N, \psi_1, \dots, \psi_N) \quad (\text{C.2})$$

where N_x is the number of design variables. At any stationary point we must have:

$$\nabla \mathcal{L} = 0 \quad (\text{C.3})$$

Taking the partial derivative of \mathcal{L} with respect to the Lagrange multipliers recovers the discrete flow constraint equation:

$$\nabla_{\boldsymbol{\Psi}} \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \psi_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \psi_N} \end{bmatrix} = \mathbf{R}(\mathcal{X}, \mathbf{Q}) = \mathbf{0} \quad (\text{C.4})$$

Similarly, taking the partial derivative of \mathcal{L} with respect to the state variables results in the adjoint equation:

$$\nabla_{\mathbf{Q}} \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{Q}_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{Q}_N} \end{bmatrix} = \frac{\partial \mathcal{J}}{\partial \mathbf{Q}} - \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} = \mathbf{0} \quad (\text{C.5})$$

Finally, taking the partial derivative of the Lagrangian with respect to the design variable recovers the objective function gradient equation (Equation 5.8):

$$\nabla_{\mathcal{X}} \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathcal{X}_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathcal{X}_{N_x}} \end{bmatrix} = \frac{\partial \mathcal{J}}{\partial \mathcal{X}} - \mathbf{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathcal{X}} = 0 \quad (\text{C.6})$$

The necessary conditions (Equations C.4, C.5 and C.6) are a system of coupled partial differential equations, the solution of which yields the optimal solution for \mathcal{X} , \mathbf{Q} and $\mathbf{\Psi}$. It is possible to solve this coupled system and obtain the optimal states and design variables. This is called a *one-shot* method or simultaneous analysis and design (SAND). This method has been applied to aerodynamic shape optimization. See Gatsis and Zingg [46] for example. Note that the size of this coupled system is much larger than the state system, therefore in many applications it may not be desirable to solve this coupled system directly.

Appendix D

GMRES

An iterative method may be used to solve the non-symmetric linear system from the Newton method formulation inexactly (Section 4.4). Classical methods such as Jacobi, Gauss-Seidel or successive overrelaxation (SOR), discussed in [106], may be used if the system is diagonally dominant, which is not the case with the present spatial discretization. Classical methods are easy to implement, but they generally converge slowly. Instead, a Krylov subspace iterative method is a more attractive alternative. The Generalized Minimal Residual (GMRES) algorithm by Saad and Shultz [163] and its variants have been found to be particularly efficient methods for CFD and aerodynamic shape optimization problems.

D.1 GMRES

The GMRES algorithm approximates the solution to the linear system

$$\mathbf{Ax} = \mathbf{b} \tag{D.1}$$

with the iterate

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{z} \tag{D.2}$$

where \mathbf{x}_0 is the initial guess. The correction \mathbf{z} is obtained from minimizing the linear residual $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$ over the Krylov subspace span by:

$$\text{span} \left[\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{k-1}\mathbf{r}_0 \right] \tag{D.3}$$

where k is the dimension of the Krylov subspace, and \mathbf{r}_0 is the linear residual based on the initial guess:

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0 \tag{D.4}$$

The basic GMRES algorithm is shown in Algorithm D.1. Note that when computing the flow solution, the matrix \mathbf{A} is the flow Jacobian matrix at each Newton iteration. In this case,

Algorithm D.1: GMRES

Data: \mathbf{A} , \mathbf{b}
Result: \mathbf{x}

- 1 Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}$, $\beta = \|\mathbf{r}_0\|_2$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
- 2 **for** $j = 1 \dots m$ **do**
- 3 Compute $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$
- 4 **for** $i = 1 \dots j$ **do**
- 5 $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$
- 6 $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
- 7 **end**
- 8 Compute $h_{j+1,j} = \|\mathbf{w}_j\|_2$
- 9 Compute $\mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$
- 10 Define $\mathbf{V}_m = [v_1 \dots v_m]$ and $\mathbf{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m}$
- 11 **end**
- 12 Compute \mathbf{y}_m the minimizer of $\|\beta\mathbf{e}_1 - \mathbf{H}_m\mathbf{y}\|_2$ and $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$

the matrix-vector product in Line 3 of Algorithm D.1 can be approximated using forward differencing (4.50).

D.2 Right-Preconditioned GMRES

When the system is right-preconditioned, a linear system in the form:

$$(\mathbf{A}\mathbf{M}^{-1})(\mathbf{M}\mathbf{x}) = \mathbf{b} \tag{D.5}$$

is solved. Right preconditioning the system does not change the solution \mathbf{x} . When right preconditioning is applied to GMRES, every reference to the matrix \mathbf{A} in Algorithm D.1 is replaced by $\mathbf{A}\mathbf{M}^{-1}$, and references to \mathbf{x} are replaced by $\mathbf{M}\mathbf{x}$. In this case, the work vector \mathbf{w}_j in Line 6 is now defined as:

$$\mathbf{w}_j = \mathbf{A}\mathbf{z} \quad \text{where} \quad \mathbf{M}\mathbf{z} = \mathbf{v}_j \tag{D.6}$$

It is therefore necessary to solve the preconditioning system

$$\mathbf{M}\mathbf{z} = \mathbf{v}_j \tag{D.7}$$

Similarly, \mathbf{V}_m in Line 12 is replaced by:

$$\mathbf{Z} = \mathbf{M}^{-1}\mathbf{V}_m \tag{D.8}$$

D.3 Flexible GMRES

In the right-preconditioned GMRES, the solution after m Krylov (inner) iterations can be expressed as a linear combination of the preconditioned vector $z_i = \mathbf{M}^{-1}\mathbf{v}_i$ for $i = 1 \dots m$. The

Algorithm D.2: FGMRES

Data: \mathbf{A} , \mathbf{M} , \mathbf{b} **Result:** \mathbf{x}

- 1 Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}$, $\beta = \|\mathbf{r}_0\|_2$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
 - 2 **for** $j = 1 \dots m$ **do**
 - 3 Compute $\mathbf{z}_j = \mathbf{M}_j^{-1}\mathbf{v}_j$
 - 4 Compute $\mathbf{w}_j = \mathbf{A}\mathbf{z}_j$
 - 5 **for** $i = 1 \dots j$ **do**
 - 6 $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$
 - 7 $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
 - 8 **end**
 - 9 Compute $h_{j+1,j} = \|\mathbf{w}_j\|_2$
 - 10 Compute $\mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$
 - 11 Define $\mathbf{Z}_m = [z_1 \dots z_m]$ and $\mathbf{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m}$
 - 12 **end**
 - 13 Compute \mathbf{y}_m the minimizer of $\|\beta e_1 - \mathbf{H}_m \mathbf{y}\|_2$ and $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{Z}_m \mathbf{y}_m$
-

vectors are all obtained using the same preconditioner \mathbf{M}^{-1} . In the *flexible* variant of GMRES (FGMRES), a different preconditioner can be used at each inner iteration:

$$\mathbf{M}_j \mathbf{z}_j = \mathbf{v}_j \tag{D.9}$$

The FGMRES algorithm is shown in Algorithm D.2. The algorithm is almost identical to the standard GMRES, and the differences are highlighted in red. The additional cost of FGMRES over GMRES is that the set of vectors $\{\mathbf{z}_j\}_{j=1 \dots m}$ from each inner iteration must be saved. However, the FGMRES algorithm has the flexibility that any iterative method can be used as a preconditioner, including classical methods mentioned above, multigrid, as well as Krylov methods such as GMRES. This property is needed when using the approximate Schur preconditioner (Chapter 4.4.2). Additional information about FGMRES can be found in [161].

Appendix E

Derivation of Partial Derivatives for Adjoint Calculations

E.1 Derivation of $\partial\mathcal{J}/\partial\mathbf{Q}$

The right-hand-side of the adjoint system, $\partial\mathcal{J}/\partial\mathbf{Q}$, is hand-derived for each objective function. Using the chain rule, for objective functions based on lift and drag (\mathcal{L} , \mathcal{D}), the partial derivative at each node i can be expressed as:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{Q}_i} = \frac{\partial\mathcal{J}}{\partial\mathcal{L}} \frac{\partial\mathcal{L}}{\partial p_i} \frac{\partial p_i}{\partial\mathbf{Q}_i} + \frac{\partial\mathcal{J}}{\partial\mathcal{D}} \frac{\partial\mathcal{D}}{\partial p_i} \frac{\partial p_i}{\partial\mathbf{Q}_i} \quad (\text{E.1})$$

The equation is applicable to inviscid flows with pressure forces only, and must be modified for viscous and turbulent flows. $\partial\mathcal{J}/\partial\mathbf{Q}_i$ is nonzero at body surfaces only. In this case, the terms $\partial\mathcal{J}/\partial\mathcal{L}$, $\partial\mathcal{J}/\partial\mathcal{D}$ and $\partial p_i/\partial\mathbf{Q}_i$ are straightforward to compute, but $\partial\mathcal{L}/\partial p_i$ and $\partial\mathcal{D}/\partial p_i$ require more careful derivation. The lift and drag coefficients are computed by first summing the forces on all the surfaces, and resolving into the appropriate directions:

$$\begin{aligned} \mathcal{L} &= \mathbf{F} \cdot \hat{\mathbf{L}} \\ \mathcal{D} &= \mathbf{F} \cdot \hat{\mathbf{D}} \end{aligned} \quad (\text{E.2})$$

where

$$\begin{aligned} \hat{\mathbf{L}} &= \hat{\mathbf{i}} \sin \alpha + \hat{\mathbf{j}} \cos \alpha \\ \hat{\mathbf{D}} &= \hat{\mathbf{i}} \cos \alpha - \hat{\mathbf{j}} \sin \alpha \end{aligned} \quad (\text{E.3})$$

are directions of lift and drag respectively, and

$$\mathbf{F} = \sum_{\Delta} p_{\text{avg}} \cdot \mathbf{N} \quad (\text{E.4})$$

is the sum of all pressure forces on the body. Here p_{avg} is the average pressure at the corners at each triangular element (Figure E.1), obtained by simple averaging of the three corners of

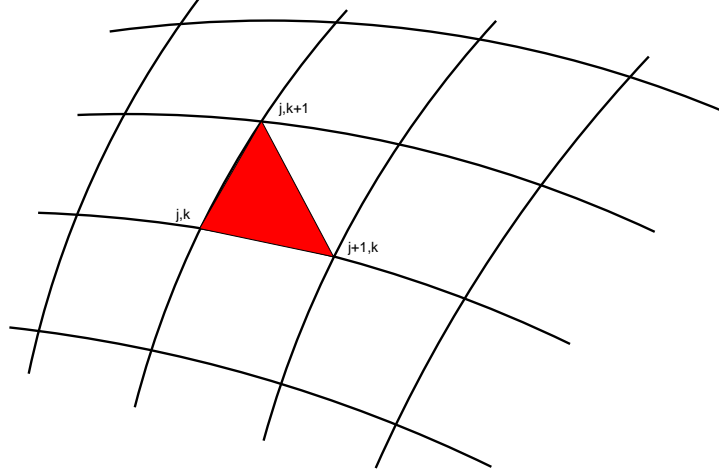


Figure E.1: Triangular elements used to compute body forces.

the triangular element, and $\mathbf{N} = \mathbf{n} \cdot A_{\Delta}$ is the area-weighted vector normal to the surface of the triangular element. Substituting this expression for \mathbf{F} into \mathcal{L} and \mathcal{D} and taking the partial derivative with respect to p_i results in the equation:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \sum_{\Delta} \left(\frac{dp_{\text{avg}}}{dp_i} \mathbf{N} \cdot \hat{\mathbf{L}} \right) \quad (\text{E.5})$$

$$\frac{\partial \mathcal{D}}{\partial p_i} = \sum_{\Delta} \left(\frac{dp_{\text{avg}}}{dp_i} \mathbf{N} \cdot \hat{\mathbf{D}} \right) \quad (\text{E.6})$$

Since the average pressure p_{avg} is only a simple average of the corners, $dp_{\text{avg}}/dp_i = 1/3$ for every triangular element that includes the node i in its corner, and 0 everywhere else. Thus it is possible to loop through every surface point and insert the pressure contribution to all related parts of the vector.

E.2 Derivation of $\partial \mathcal{J} / \partial \alpha$

For angle-of-attack design variables, the derivation of $\frac{\partial \mathcal{J}}{\partial \alpha}$ can be done in a similar fashion as $\frac{\partial \mathcal{J}}{\partial \mathbf{Q}}$, by using the chain rule:

$$\frac{\partial \mathcal{J}}{\partial \alpha} = \frac{\partial \mathcal{J}}{\partial \mathcal{L}} \frac{\partial \mathcal{L}}{\partial \alpha} + \frac{\partial \mathcal{J}}{\partial \mathcal{D}} \frac{\partial \mathcal{D}}{\partial \alpha} \quad (\text{E.7})$$

Since lift and drag coefficients are only explicitly related to α in $\hat{\mathbf{L}}$ and $\hat{\mathbf{D}}$ (E.2), the expression is simple to derive by hand. For geometric design variables, finite differencing is used.

E.3 Derivation of $\partial\mathbf{R}/\partial\alpha$

For angle of attack design variables, the complex step method is used by adding a complex perturbation to α :

$$\hat{\alpha} = \alpha + ih$$

with a step size of $h = 10^{-20}$. In residual calculations, α is only explicitly referred to in free-stream properties. Thus complex free-stream properties are set using the new α and the residual vector is computed. The imaginary part contains the solution $\partial\mathbf{R}/\partial\alpha$. For geometric design variables where regridding is required, finite differencing is used.

Appendix F

Additional Optimization Results

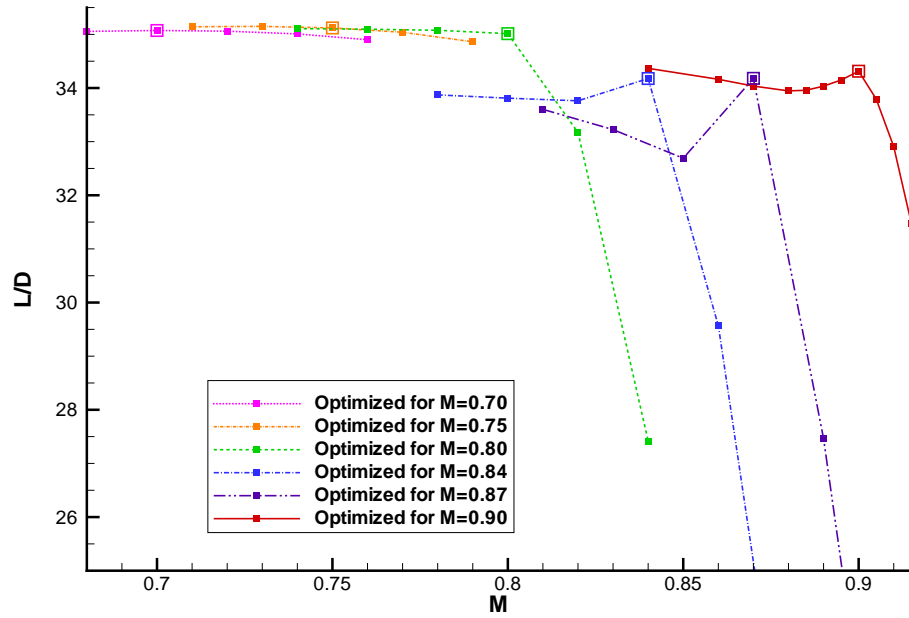
To demonstrate the robustness of the optimizer for design problems in a range of Mach numbers, additional results are presented for five single-point optimization cases for the ONERA M6 wing at various Mach numbers. For these cases, the lift-constrained drag minimization objective function (2.5) with the same targets and weights on lift and drag as in Section 8.1 are reused:

$$\begin{aligned} C_L^* &= 0.308 & \omega_L &= 100.0 \\ C_D^* &= 0.00746 & \omega_D &= 1.0 \end{aligned}$$

The same volume grid, surface parameterization, design variables, and volume constraint are also reused from Section 8.1. The operating Mach numbers are $M_1 = 0.70$, $M_2 = 0.75$, $M_3 = 0.80$, $M_4 = 0.84$ and $M_5 = 0.87$. At least 100 optimization iterations are performed for each case, until the objective function and gradient L_2 -norm have reduced by at least two orders of magnitude.

The drag divergence for each optimized geometry is computed at a fixed lift coefficient of $C_L = 0.307$. The lift-to-drag ratios of the optimized wings are shown in Figure F.1 and in Table F.1. The optimization results for $M = 0.90$ (Section 8.1) are also shown for comparison. In every case, the final L/D ratio is found to be nearly independent of the operating Mach number. However, the minimum induced drag predicted by lifting-line theory (2.6) cannot be achieved in all of these cases. This may indicate that more flexibility in the geometry parameterization may be needed. In particular, additional local twist control may be desirable. It is also difficult to determine exactly how much numerical error is present.

The wing sections of the optimized wings are shown in Figure F.2. The final sweep angles for all optimized wings are compared in Table F.1. In general, it is found that at lower Mach numbers, wings have lower sweep angles.

Figure F.1: Drag divergence of various optimized wings at $C_L = 0.307$

M	Final L/D	Λ_{final}
0.70	35.05	30.1°
0.75	35.09	30.7°
0.80	35.01	29.0°
0.84	34.17	31.7°
0.87	34.22	34.4°
0.90	34.35	44.3°

Table F.1: Final L/D ratios and sweep angles of various optimized wings

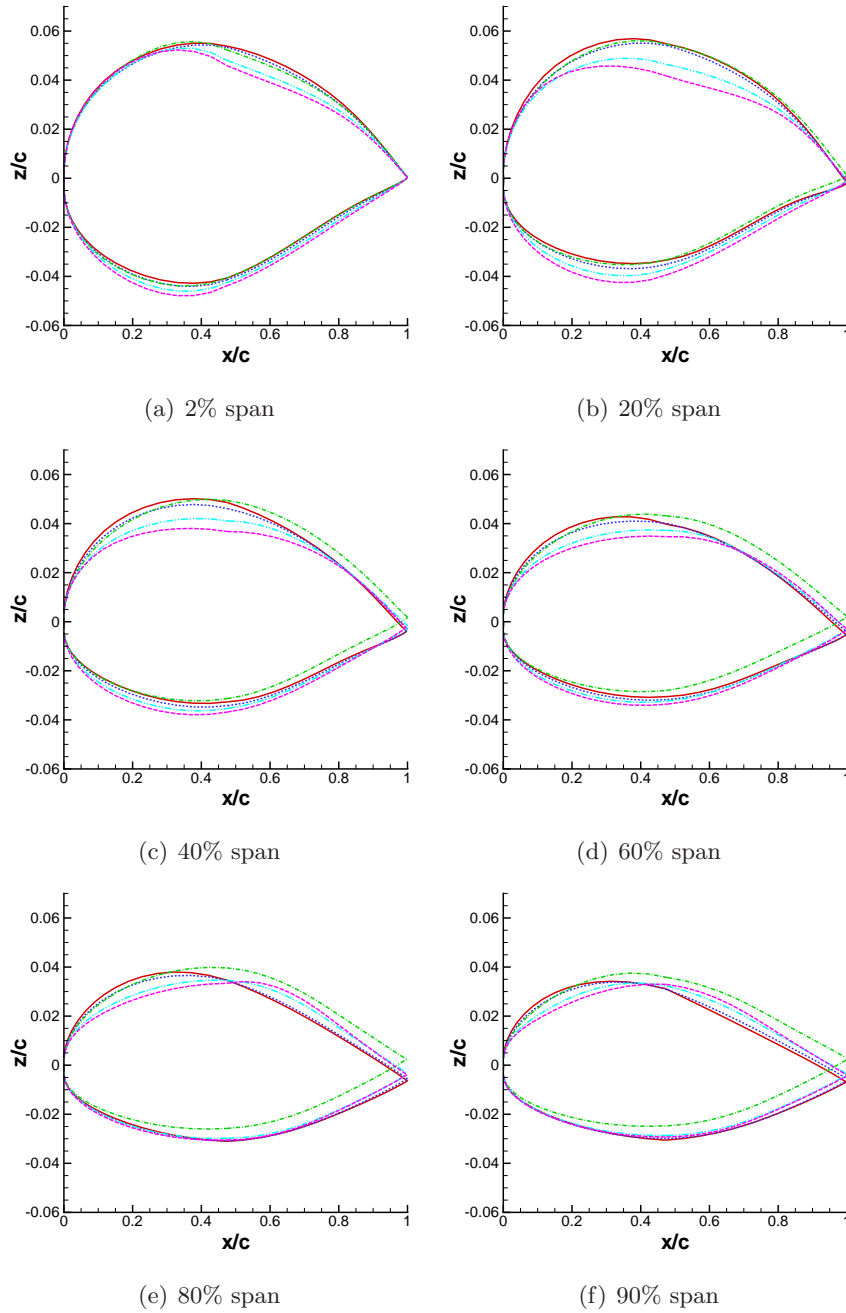


Figure F.2: Comparison of wing section shapes of single-point optimized wings at various spanwise locations