Optimal Shape Design Of Aerodynamic Configurations: A Newton-Krylov Approach

by

Marian Nemec

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

Abstract

Optimal Shape Design Of Aerodynamic Configurations: A Newton-Krylov Approach

Marian Nemec

<marian@oddjob.utias.utoronto.ca>

Doctor of Philosophy

Graduate Department of Aerospace Science and Engineering

University of Toronto

2003

Optimal shape design of aerodynamic configurations is a challenging problem due to the nonlinear effects of complex flow features such as shock waves, boundary layers, and separation. A Newton–Krylov algorithm is presented for aerodynamic design using gradient-based numerical optimization. The flow is governed by the two-dimensional compressible Navier–Stokes equations in conjunction with a one-equation turbulence model, which are discretized on multi-block structured grids. The discrete-adjoint method is applied to compute the objective function gradient. The adjoint equation is solved using the preconditioned generalized minimal residual (GMRES) method. A novel preconditioner is introduced, and together with a complete differentiation of the discretized Navier–Stokes and turbulence model equations, this results in an accurate and efficient evaluation of the gradient. The gradient is obtained in just one-fifth to one-half of the time required to converge a flow solution. Furthermore, fast flow solutions are obtained using the same preconditioned GMRES method in conjunction with an inexact-Newton approach. Optimization constraints are enforced through a penalty formulation, and the resulting unconstrained problem is solved via a quasi-Newton method. The performance of the new algorithm is demonstrated for several design examples that include lift enhancement, where the optimal position of a flap is determined within a high-lift configuration, lift-constrained drag minimization at multiple transonic operating points, and the computation of a Pareto front based on competing objectives. In all examples, the gradient is reduced by several orders of magnitude, indicating that a local minimum has been obtained. Overall, the results show that the new algorithm is among the fastest presently available for aerodynamic shape optimization and provides an effective approach for practical aerodynamic design.

Acknowledgements

I would like to thank David Zingg for excellent teaching, guidance, and supervision. His passion for research and understanding of fundamental principles are a constant source of motivation. I would also like to thank my committee members, Jorn Hansen, Clinton Groth, Tom Pulliam, and Marius Paraschivoiu, for many insightful discussions and suggestions.

To the staff and students at UTIAS, especially the CFD group, thank you for a professional, yet warm and enjoyable atmosphere. Most notably, Stan De Rango who helped me understand the many subtle details of flow solvers, and Todd Chisholm for the debates on Jacobians and algorithms. The system administration work of Mike Sullivan and Jason Lassaline is much appreciated.

To my family, I would like to express my gratitude for their support and understanding. Mélanie, thank you for your patience and encouragement. You provided a balance and a good perspective on things. Grand merci!

Financial support from the Natural Sciences and Engineering Research Council of Canada, the Government of Ontario, Bombardier Aerospace, Pratt & Whitney Canada, and the University of Toronto is gratefully acknowledged.

Marian Nemec

University of Toronto Institute for Aerospace Studies
November 2, 2002

CONTENTS

List of Tables

List of Figures

List of Symbols

Alphanumeric

$\mathcal{A}$	flow Jacobian matrix
$\mathcal{F}$	vector of continuous flow equations
$\mathcal{G}$	gradient vector
$\mathcal{H}$	approximation to inverse of Hessian matrix
$\mathcal{J}$	objective function (scalar)
$\mathcal{M}$	GMRES preconditioning matrix
$\mathcal{P}r$	laminar Prandtl number
$\mathcal{P}r_t$	turbulent Prandtl number
$\mathcal{R}e$	Reynolds number
I	identity matrix
a	speed of sound
A, B	inviscid flux Jacobians
B	B-spline basis functions
C	constraint equations
c	chord length of airfoil
$C_{\mathrm{L}}, C_{\mathrm{D}}$	lift and drag coefficients
$C^*_{\mathrm{L}}, C^*_{\mathrm{D}}$	target lift and drag coefficients
C_{p}	pressure coefficient
C^*_{p}	target pressure coefficient
d	B-spline knot sequence
d_{w}	distance to closest wall
E	inviscid flux vector in x-direction
e	total energy
E_v	viscous flux vector in x-direction
e_n	nth unit vector
F	inviscid flux vector in y-direction
F_v	viscous flux vector in y-direction
h	finite-difference stepsize
$h(x_j)$	airfoil thickness at location x_j

J	metric Jacobian
K	viscous flux Jacobian
M	Mach number
m	number of GMRES search directions
N_{B}	number of grid blocks
N_{c}	number of constraint equations
N_{D}	number of design variables
N_{F}	number of flow variables
N_{m}	number of design points
p	pressure
Q	vector of conservative flow variables
R	vector of discrete residual equations
S	vorticity
s	search direction vector
t	time
U, V	contravariant velocities
u, v	x, y-components of velocity, respectively
w	parameter vector for B-spline curve
w_i	user specified weight for multi-point optimization
X	vector of design variables
x, y	Cartesian coordinate directions
$X^{\mathrm{c}}, Y^{\mathrm{c}}$	Cartesian coordinates of B-spline control points
$x_{\mathrm{a}}, y_{\mathrm{a}}$	Cartesian coordinates of airfoil surface

Greek

α	angle of attack
β	stepsize for quasi-Newton update
Δt	time step
γ	specific heat ratio
κ_2	second-difference dissipation constant
κ_4	fourth-difference dissipation constant
μ	dynamic laminar viscosity
μ_t	turbulent eddy viscosity
ν	kinematic laminar viscosity, $\nu = \mu/\rho$

ν_t	kinematic eddy viscosity, $\nu_t = \mu_t/\rho$
Ω	feasible region of design space
ω_{D}	user specified weight for drag coefficient
ω_{L}	user specified weight for lift coefficient
ω_{P}	user specified weight for gap and overlap constraints
ω_{T}	user specified weight for thickness constraint
ϕ	artificial dissipation constant for $\mathcal{M}$
ρ	density
σ	spectral radius of the inviscid flux Jacobian matrix
τ	shear stress
$\tilde{\nu}$	turbulence model working variable
Υ	pressure switch
ξ, η	general curvilinear coordinates
ζ	degree of convergence for inexact-Newton method

Superscripts

$\bar{}$	dimensional variable
T	transpose
$\hat{}$	transformation to curvilinear coordinates

Subscripts

∞	freestream quantity
M	discrete variables for mean flow
T	discrete variable for turbulence model

Abbreviations

AF	Approximate Factorization
BFGS	Broyden–Fletcher–Goldfarb–Shanno
CAD	Computer Aided Design
GMRES	Generalized Minimal Residual
ILU	Incomplete Lower Upper
NK	Newton Krylov

Chapter 1

INTRODUCTION

1.1 Motivation

THE goal of aerodynamic design is to accurately and efficiently determine surface shapes that attain optimal aerodynamic performance. Among the most important applications are the design of external surfaces of aircraft, especially wings, and the design of components for gas-turbine engines, such as compressors and turbines. Although our focus is on aerodynamic design, it should be noted that optimal shape design problems, where the optimization of a performance criterion depends on the shape of a boundary, occur in many areas of engineering. Examples include the design of structures, such as plates and shells, and the problem of minimal hydrodynamic resistance.[1]

The primary challenge of aerodynamic design is the underlying complex nature of the flow. For cruise configurations of commercial aircraft, the flow is compressible, usually transonic, and may contain features such as shock waves, shock-induced boundary-layer separation, and boundary-layer transition. At take-off and landing, the use of configurations with multiple elements, for example slats and flaps, at high angles of attack causes additional difficulties. An excellent overview of high-lift aerodynamics is provided by Smith [159]. The dominant

[1] First studied by Newton over three hundred years ago.

flow features include regions of separated flow, confluent boundary layers and wakes, and even regions of supercritical flow where compressibility effects are important. Such flow features have a strong influence on the aerodynamic performance of a configuration due to their nonlinear effects. Consequently, the nonlinear effects must be carefully controlled by the shape of the configuration in order to realize optimal performance. Note that the design problem inherently involves multiple operating conditions.

The design of practical configurations with optimal aerodynamic properties requires the consideration of not only aerodynamics, but also several other disciplines, namely, structures, acoustics, and controls [161]. The aero-structural coupling is perhaps most significant, since an optimal aerodynamic design may incur excessive weight penalties [109, 129]. Furthermore, the design is constrained by numerous engineering requirements [60], such as the desirable fuel tank volume within the wing. Jameson and Vassberg [88] present an insightful example of multidisciplinary design based on requirements for a small race plane.

Although aerodynamic design is a complex task, the incentive for the development of effective design strategies is substantial. The potential benefits include safer and more efficient aircraft and shorter design-cycle times. For example, an efficient high-lift configuration can significantly improve the aerodynamic performance of an aircraft, as well as provide weight savings and reductions in mechanical complexity of the high-lift system [172]. For cruise configurations, the benefits include low drag and improved fuel burn, an increase in the drag-divergence Mach number, and also sonic-boom reductions for the design of quiet supersonic platforms [85, 4].

Experimental methods, using wind tunnel and flight testing, provide the basis of the traditional "cut and try" approach for the design of new aerodynamic shapes. This approach alone, however, is too expensive, which has motivated the continual development of sophisticated computational methods for flow simulation [104] and established the field of computational fluid dynamics. These methods, along with performance gains in computer technology, complement and in some instances even replace the use of the wind tunnel during the design process. In this setting, the experimental and computational techniques are analysis tools that provide reliable estimates of aerodynamic performance for given configurations and operating conditions. A good physical insight of the designer is required to select and evolve the candidate aerodynamic shapes and provide an overall control for the design process.

The fundamental question that the design process seeks to answer is the following:

What is the aerodynamic shape that attains the design objectives?

The "cut and try" approach does not address this question directly, resulting in an inefficient design process. Consequently, a significant research effort has been devoted to the development of computational methods for the solution of the design problem. The resulting methods can

be divided into two categories, namely, inverse design methods and numerical optimization methods.

Inverse design methods, first introduced in 1945 by Lighthill [102], are an established approach for the determination of an airfoil shape that attains a given pressure distribution. For example, an experienced designer is able to specify a lift-preserving pressure distribution that is shock-free for transonic flow conditions, thereby achieving significant drag reductions. Inverse methods were also used to design the well-known Liebeck high-lift airfoils [101]. An advantage of this approach is its low computational cost. Giles and Drela [61] developed an inverse design method using the two-dimensional, coupled Euler and boundary layer equations with a computational cost equivalent to the solution of just one analysis problem. Although inverse design methods have been applied to three-dimensional problems [55], their primary limitation is the specification of desirable pressure distributions that lead to optimal designs, especially for problems with multiple operating conditions and turbulent and separated flow.

Numerical optimization methods provide a more general approach for solving design problems. The creativity and insight of an experienced designer are required to reduce the design problem to a well-posed optimization problem. This involves the definition of objective functions that specify the goals of the optimization, design variables that determine the aerodynamic shape, as well as constraints that qualify a feasible region of the design space. Note that for practical problems, it is very likely that the objectives are competing and that changes in the specification of the optimization problem occur as the design evolves. Typically, the problem is cast as a minimization, where the objective functions include lift, drag, and moment functionals. Inverse design can be considered a subset of numerical optimization by defining an objective function that represents the difference between the target and the actual pressure distributions.

Once an aerodynamic shape optimization problem is defined, a numerical optimization method coupled with a suitable flow analysis tool (flow solver) is used to find a solution. The goal of the optimization is to determine a set of design variables that satisfies the objectives within the feasible region of the design space. This approach provides an effective design strategy, since the selection of an optimal configuration is based on a systematic and potentially fast evaluation of candidate designs. Furthermore, the effects of nonlinear flow features, multiple operating conditions, and multidisciplinary interactions can be intrinsic in the formulation of the optimization problem.

The main difficulty in the realization of such general optimization algorithms is computational cost, which is due to the need for repeated evaluation of the objective function, and hence the flow equations. Objective function gradients can be used to accelerate the convergence of the optimization problem; however, fast evaluation of accurate gradients is a challenging task. The goal of this thesis is to develop an accurate and robust algorithm for complex aerodynamic

shape optimization problems that provides an efficient evaluation of the objective function and gradient.

An overview of algorithms for aerodynamic shape optimization problems is given in the following section, which is divided into two parts. In the first part, Subsection 1.2.1, we provide an outline of the techniques used for the essential components of the optimization problem. In Subsection 1.2.2, we summarize methods used for the evaluation of the gradient. The objectives of this thesis are stated in Section 1.3, which also provides an outline of the document.

1.2 Review of Aerodynamic Shape Optimization Algorithms and Applications

1.2.1 Essential Components

The aforementioned, essential components of an aerodynamic shape optimization problem are objective functions and constraints, design variables[2], a flow solver, and a numerical optimization method. These components should be selected carefully, since they have a direct impact on the accuracy and efficiency of the optimization. Jou *et al.* [89], Giles [60], and Drela [38] present important factors that influence the selection of the objective functions and constraints for practical problems. These factors include limitations of the flow solver that can be exploited by the optimization procedure and the consideration of multiple operating conditions.

Samareh [154, 153], Oyama [136], and Reuther *et al.* [146] provide excellent summaries of shape parameterization techniques that can be used to define design variables. Overall, it is important that the selected parameterization technique provides sufficient flexibility in order to realize truly optimal designs; yet, it is also desirable that the number of parameters required to define the shape is small in order to ensure a reasonable convergence rate of the optimization. Additional desirable qualities include local shape control, as well as smooth shape perturbations.

The simplest choice of design variables, which does not require an explicit parameterization technique, is the location of surface nodes in the computational grid. Jameson [84] and Mohammadi [113] have successfully used this approach in conjunction with smoothing operators. Unfortunately, the number of design variables becomes prohibitively large for three-dimensional, turbulent flow problems.

The most promising shape parameterization approaches are the shape-function, polynomial and spline approaches. The shape-function approach was introduced by Hicks and Henne [75]. For recent publications that contain a detailed description of this approach see [48, 146]. An

[2] Also referred to as design parameters or controls in the context of control theory or genes in the context of genetic algorithms.

example of the polynomial approach is PARSEC [160, 59, 136], where practical airfoil parameters such as the trailing-edge angle, airfoil thickness and curvature are used to define the airfoil shape. Spline approaches, such as basis splines (B-splines) and nonuniform rational B-splines (NURBS) [32, 53], have received considerable attention [20, 8, 98] since they can be tailored to provide local and smooth airfoil shape control and are easily incorporated into the CAD environment.

Closely associated with the design variables is a grid-perturbation algorithm, which adjusts the computational grid relative to the surface shape defined by the design variables. Although this task could be accomplished by a grid generator, present-day grid generation tools for complex geometries are computationally expensive and typically not fully automated. Therefore, grid-perturbation algorithms are often used to modify a baseline grid, which is generated only once at the start of the optimization. For two- and simple three-dimensional problems using structured grids, algebraic grid-perturbation strategies work well. Burgreen and Baysal [19] use an algebraic strategy that preserves the distance to the outer grid boundary and relocates the nodes in the normal direction proportional to the distance from the surface. A similar strategy is used by Reuther *et al.* [145]. However, they note that this strategy fails for three-dimensional problems where multiple surfaces require simultaneous adjustment. A new strategy is introduced based on transfinite interpolation [167]. Grid-perturbation algorithms for unstructured grids generally use a force equilibrium of springs approach, but safeguards are required for viscous-grid perturbations [8, 47, 42]. A promising strategy for large surface deformations based on a modified linear elasticity theory is proposed by Nielsen and Anderson [132].

The accuracy of the optimization ultimately depends on the modelling of the flow, and hence, the flow solver. Accurate modelling of the nonlinear flow effects described in Section 1.1 requires the solution of the compressible, Reynolds-averaged, Navier–Stokes equations combined with a suitable turbulence model. Two-dimensional flow problems for single and multi-element airfoils are considered in this work. Detailed reviews of two-dimensional flow solvers and results for transonic cases are provided by Holst [78] and for high-lift cases see Rumsey and Ying [149], Klausmeyer and Lin [95], and Fejtek [54]. With the exception of stall and post-stall conditions, the flow solvers provide accurate estimates of aerodynamic performance. The algorithm developed in this work is based on two established solvers, namely, ARC2D [143] and TORNADO [121, 67]. TORNADO is a generalization of the approximate-factorization algorithm of ARC2D for multi-block structured grids. Both solvers have been a subject of detailed accuracy studies for a wide range of flow conditions and airfoil configurations, see for example [107, 78, 182, 122, 54, 120, 34, 149].

The flow solver has also a significant influence on the efficiency of the optimization. The repeated evaluations of the objective functions required during the optimization demand fast

flow solutions. Detailed reviews of effective flow solvers for the Navier–Stokes equations are given by Nelson [119], Pueyo [141], and Walsh [176]. Among the fastest algorithms are the Newton–Krylov solvers [175, 9, 7, 142, 183, 22]. For example, promising results are presented by Pueyo and Zingg [142], who used the preconditioned generalized minimal residual (GMRES) Krylov subspace method [152] in conjunction with an inexact-Newton strategy. A critical component in this approach is a fast solution of the linear system at each Newton iteration, which is provided by the preconditioned GMRES method. For the aerodynamic shape optimization problem, such Newton–Krylov solvers are very appealing, since they not only provide fast solutions to the flow equations, but the preconditioned GMRES method can also be used to compute objective function gradients.

Numerical optimization methods, which control the solution of the optimization problem, can be divided into the following four categories: 1) classical direct search methods, 2) stochastic methods, 3) gradient-based methods, and 4) fully-coupled methods. An insightful summary of direct search methods is provided by Lewis *et al.* [99]. Among the most well-known direct search methods is the simplex method of Nelder and Mead. Duvigneau and Visonneau [42] investigate the performance of this method for aerodynamic and hydrodynamic shape optimization problems, including the optimization of a high-lift configuration. The flow is governed by the two-dimensional incompressible Navier–Stokes equations. Although this approach provides a robust optimization method, its convergence to the optimal solution is slow and requires a large number of flow evaluations.

Simulated annealing [94] and genetic algorithms [68] are good examples of stochastic methods. The latter, in particular, have received considerable attention for application in aerodynamic shape optimization problems, see for example Giannakoglou [59], Marco *et al.* [108], Obayashi [134], and Oyama [136]. The fundamental concepts of genetic algorithms are based on the process of natural selection. These algorithms are capable of finding a global optimum, are well-suited for problems with multiple and non-smooth objectives, and can be used with categorical design variables. The primary disadvantage of genetic algorithms is high computational cost. Tse and Chan [170] and Holst and Pulliam [79] provide optimization examples where up to 10,000 flow solutions are required to reach convergence.

Gradient-based methods [66, 36, 133] are potentially the most effective methods for aerodynamic shape optimization problems, since significant design improvements can be obtained in a relatively few evaluations of the objective function and gradient. However, these methods require a smooth design space[3] and inherently converge to a local optimum. The gradient can be used directly to determine a search direction for the optimization process, which is the case

[3]Recently, Moreau and Aeyels [115] examined the optimization of discontinuous objective functions using the concept of generalized gradients.

for the classic method of steepest descent. Alternatively, the gradient in conjunction with the objective function value can be used to approximate the "curvature" of the design space, which leads to the much more efficient quasi-Newton methods and conjugate gradient methods. In 1974, Hicks, Murman, and Vanderplaats [76] were the first to apply gradient-based methods to aerodynamic shape optimization problems. They used the method of feasible directions, which is based on conjugate gradients, to optimize airfoil shapes in transonic flow governed by the small-disturbance equation. Since this pioneering work, the application of gradient-based methods to aerodynamic shape optimization problems has been an active area of research.

Although steepest descent, quasi-Newton, and conjugate gradient methods were originally developed in the context of unconstrained optimization, when combined with other techniques these methods are also effective for constrained problems. The aerodynamic shape optimization problem can be formulated as an unconstrained problem by a careful selection of objective functions and design variables [27, 84, 165]; however, constraints that represent structural limitations, such as volume or thickness requirements, are usually necessary.

The most popular approaches for constrained problems include the projection of the search direction into the allowable design space [37, 48, 86], the use of the Kreisselmeier-Steinhauser function [181, 8, 131, 11] and other penalty methods [113], and the use of methods based on sequential quadratic programming (SQP) [66, 146, 162, 91]. Melvin *et al.* [112], see also [89], use the SQP approach, but introduce an approximate Hessian formulation. Jameson and Vassberg [87] compare a number of gradient-based methods using a model problem with as many as 8,000 design variables. It should be emphasized that one of the main challenges for effective implementation of gradient-based methods is an accurate and efficient computation of the gradient. Methods for gradient computation are summarized in the following subsection, which also includes example applications.

Fully-coupled methods[4] use a Lagrangian formulation for the optimization problem and attempt to solve the corresponding coupled system of nonlinear equations that define the first-order optimality conditions. Excellent descriptions of this approach are provided by Gunzburger [71], Biros and Ghattas [16], and Shenoy *et al.* [156]. Note that the gradient-based methods solve the same system of nonlinear equations, but in a decoupled form. Consequently, the flow equations and the gradient must be evaluated repeatedly, which is computationally expensive. The main advantage of fully-coupled methods is that the repeated flow and gradient evaluations are avoided; however, the coupled system of nonlinear equations is approximately twice the size of the flow equations. In addition, the system size increases linearly with the number of objectives and operating conditions considered in the optimization problem.

[4]Also referred to as all-at-once and one-shot methods.

Frank and Shubin [57] compare the fully-coupled and gradient-based methods for an inverse design problem using a quasi-one-dimensional nozzle flow. They conclude that the fully-coupled approach is less robust but considerably more efficient than gradient-based methods. A similar problem is also considered by Feng and Pulliam [56], and Gatsis and Zingg [58]. Iollo *et al.* [82] apply a pseudo-time method to solve the fully-coupled system and examine an inverse design problem using the two-dimensional Euler equations. They report that the computational cost of the optimization problem is only three to four times greater than the analysis problem. Biros and Ghattas [16] also consider an inverse design problem, but for a laminar incompressible flow. Their fully-coupled method appears to converge five time faster when compared with conventional SQP methods.

We conclude this subsection with a brief overview of approximation techniques. These techniques are combined with numerical optimization methods in order to obtain reductions in the computational cost of the optimization. The main idea is to approximate the design space with a simpler, low-cost model. On the basis of this model, a numerical optimization method is used to find an approximate optimal solution. The model can be subsequently refined and the procedure repeated.

Dadone and Grossman [29, 30] present a progressive optimization procedure that uses grid-sequencing and loosely-converged flow and gradient evaluations. Their results indicate that the amount of computational work required for the convergence of the optimization is equivalent to just four flow solutions. Alexandrov *et al.* [3] consider variable-fidelity physics models for gradient-based optimization of high-lift configurations. This model-management approach can yield up to five-fold savings in computational work relative to the high-fidelity model alone. Greenman and Roth [69] use an artificial neural network in conjunction with genetic and gradient-based algorithms to optimize the positions and deflections of slats and flaps. Ahn *et al.* [2] construct a response surface based on quadratic polynomials and consider airfoil optimization problems in transonic flow governed by the Navier–Stokes equations. Otto *et al.* [135] present a Bayesian-validated surrogate approach that estimates the accuracy of the approximation and provides error bounds for the optimal solution. LeGresley and Alonso [97] apply proper orthogonal decomposition to construct a reduced-order model for the Euler equations. The resulting model is evaluated for inverse design problems. Chung and Alonso [24] investigate a Cokriging method that uses objective function and gradient values to approximate the design-space surface.

1.2.2 Numerical Methods for Gradient Computation

Finite-difference schemes are a classic approach for the computation of objective function gradients [76, 75, 27, 52]. An advantage of this approach is that its implementation is straightforward. Unfortunately, there are two significant disadvantages. First, for practical problems the computational cost of repeated finite-difference gradient calculations is prohibitive. Second, finite-difference schemes are prone to accuracy errors due to their dependence on the step size. Recently, Anderson *et al.* [6] and Martins *et al.* [110] presented improved accuracy results for finite-difference gradient calculations using an approach based on complex variables [106]. Although this approach is insensitive to the step size, it requires even greater computational resources. In this work, we have not implemented the complex variable approach; however, we found the finite-difference gradients very useful for testing and validation studies.

Two significantly better ways of computing the gradient are the direct, or flow-sensitivity, method and the adjoint method [138]. Pironneau [139] provides an insightful overview of the development of these methods that includes a discussion of theoretical results on the existence of solutions, techniques for numerical implementation, and practical applications. Both methods have been applied to design problems governed by the steady Navier–Stokes equations and can be subdivided into the continuous [84, 86, 162, 171, 117, 118, 166] and the discrete approach [10, 8, 5, 96, 157, 49, 45, 90, 62, 180]. For the continuous approach, the adjoint (or sensitivity) equation is derived from the continuous flow equations and then discretized, while for the discrete approach, the adjoint equation is derived directly from the discrete flow equations. Gunzburger [71], and Giles and Pierce [63] provide detailed reviews of both approaches. Furthermore, the adjoint and flow-sensitivity methods have been also applied to control problems for unsteady flow[5] [72, 74, 83], and the adjoint method is a promising approach for error analysis [65, 174].

The main advantage of the adjoint method over the flow-sensitivity method is that the cost of the gradient computation is virtually independent of the number of design variables. However, flow-sensitivities can be useful for design problems that contain constraints dependent on the flow variables [180]. In addition, it may be advantageous to implement both methods since the resulting information can be used to accelerate the convergence of the design problem by constructing better approximations of the Hessian matrix [37, 89, 157].

Jameson *et al.* [86] derived the viscous adjoint terms for the continuous approach for laminar and turbulent flows on structured grids. Although this formulation neglects the linearization of laminar and turbulent viscosities, it has been successfully applied to a number of aerodynamic shape optimization problems, including two-dimensional high-lift configurations [92, 93] with

[5] Similar unsteady adjoint applications are also frequent in meteorology [168, 177].

the Baldwin–Lomax and the one-equation Spalart–Allmaras turbulence models [164]. Nadarajah and Jameson [118] obtained inaccurate gradients using the continuous adjoint approach when considering objective functions based on skin-friction drag. In order to overcome this problem they suggest replacing the viscous continuous adjoint boundary condition with the discrete adjoint boundary condition.

Anderson and Venkatakrishnan [8] analyzed both the continuous- and discrete-adjoint methods for unstructured grids and implemented the discrete approach for viscous design problems. Anderson and Bonhaus [5] extended this work to turbulent flows by differentiating the Spalart–Allmaras turbulence model by hand. They report accurate gradients for turbulent design cases. Nielsen and Anderson [131] apply the same strategy to three-dimensional, turbulent flow design problems and also demonstrate excellent gradient accuracy. Furthermore, their results show the influence of various simplifying assumptions in the linearization of the discretized governing equations, such as the assumption of constant turbulent viscosity and a linearization based on first-order discretization. They conclude that most of these simplifying assumptions result in significant gradient errors. Similar results are obtained by Kim *et al.* [90]. Automatic differentiation can be used to differentiate the governing flow equations [114, 157, 6]. However, the resulting gradient computation is usually not as efficient and extensive modifications of the flow solver code may be required in order to control memory requirements.

In the adjoint and flow-sensitivity methods, the computational cost of the gradient calculation is dominated by the solution of the large linear system of equations that arises from the flow Jacobian matrix. A popular approach to solve the adjoint and flow-sensitivity equations is to use the same scheme that solves the governing flow equations, for example the explicit and point-implicit multistage Runge–Kutta schemes coupled with multigrid [86, 49, 62], the approximate-factorization scheme [96], and also the lower-upper symmetric-Gauss–Seidel (LU-SGS) scheme [90]. The preconditioned GMRES method has been used to solve the discrete sensitivity equation for laminar flows [44] and also to solve the discrete adjoint equation in conjunction with a backward-Euler time-marching scheme with multigrid for turbulent flows [8, 131, 132]. Generally, the computational effort required to converge the adjoint equation sufficiently in order to obtain accurate gradients is approximately equivalent to one to two flow solutions; however, for the discrete adjoint method this effort may be significantly increased if memory limitations prohibit the storage of the flow Jacobian matrix [130, 90].

Incomplete gradients [114, 163] are a promising approach for reducing the cost of gradient computations. This approach is based on the assumption that for small changes in shape, the flow solution remains almost unchanged. Consequently, only grid nodes close to the shape boundary of interest are considered during the gradient computation. A disadvantage of incomplete gradients is that they introduce accuracy errors in the gradient. This error can have

an adverse effect on the convergence of the optimization [31, 15].

1.3 Objectives

The objectives of this thesis are two-fold:

1. Develop a gradient-based algorithm for aerodynamic shape optimization problems that are governed by the compressible, two-dimensional Navier–Stokes equations in conjunction with the Spalart-Allmaras turbulence model. This objective can be further subdivided as follows:

   - Establish a framework for gradient-based optimization of single- and multi-element airfoil configurations. This includes methods for airfoil shape parameterization, grid-perturbation, flow solution, gradient computation, and an optimization procedure.
   - Examine the use of the preconditioned GMRES method to solve the discrete adjoint and flow-sensitivity equations for gradient computations.
   - Accelerate the convergence rate of the flow solver by applying the same preconditioned GMRES method in conjunction with an inexact-Newton strategy.

2. Evaluate and characterize the new algorithm in a practical aerodynamic design context by applying it to several representative design examples, including inverse design, maximization of lift-to-drag ratio, lift-enhancement of high-lift configurations, lift-constrained drag minimization at multiple transonic operating points, and the computation of Pareto fronts based on competing objectives. Factors under consideration include the accuracy and efficiency of the gradient computation, the efficiency of the flow solver, the overall efficiency of the optimization procedure, and global and local optimal solutions.

Chapters 2 and 3 address the first objective, where we present the governing equations for the formulation of the optimization problem and the numerical algorithm, respectively. Thereafter, we focus on the second objective. The algorithm is validated in Chapter 4 and design examples are presented in Chapter 5. The development and application of the new aerodynamic shape optimization algorithm is also presented in [123, 125, 127, 124, 126, 128, 184], which provide additional results.

Chapter 2

GOVERNING EQUATIONS

2.1 Problem Formulation

The aerodynamic shape optimization problem consists of determining values of design variables X, such that the objective function $\mathcal{J}$ is minimized

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

subject to constraint equations C_j

$$C_j(X, Q) \leq 0 \qquad j = 1, \ldots, N_c \tag{2.2}$$

where the vector Q denotes the conservative flow variables and N_c denotes the number of constraint equations. The flow variables are forced to satisfy the governing flow equations within a feasible region of the design space Ω

$$\mathcal{F}(X, Q) = 0 \qquad \forall \ X \in \Omega \tag{2.3}$$

which implicitly defines $Q = f(X)$. The following sections provide a detailed description of Eqs. 2.1–2.3 in order to clearly define the aerodynamic shape optimization problem.

2.2 Design Variables

The vector of design variables, X, primarily contains parameters that control the shape of the airfoil. Depending on the problem of interest, additional design variables may include the angle of attack, and the horizontal and vertical translation design variables that control the position of slats and flaps in multi-element configurations. The element's relative deflection angle within a configuration is kept constant.

B-splines are used to parameterize the airfoil shape. The following development, which describes the construction of a B-spline curve and the initial airfoil shape approximation, is based on the work of de Boor [32] and Hoschek [80], with additional information provided in [53, 148]. The parametric representation of an airfoil shape with a B-spline curve is given by

$$x_{\mathrm{a}}(w_j) = \sum_{i=1}^{n+1} X_i^{\mathrm{c}} B_{i,k}(w_j) \tag{2.4}$$

$$y_{\mathrm{a}}(w_j) = \sum_{i=1}^{n+1} Y_i^{\mathrm{c}} B_{i,k}(w_j) \tag{2.5}$$

where $(x_{\mathrm{a}}, y_{\mathrm{a}})$ are the Cartesian coordinates of the airfoil surface, $B_{i,k}$ are the B-spline basis functions of order k (degree $k-1$), $(X_i^{\mathrm{c}}, Y_i^{\mathrm{c}})$ are the coordinates of the B-spline control points, and $n+1$ is the total number of control points. Cubic B-splines, $k = 4$, are used for all test cases. The B-spline basis functions are defined by the Cox–de Boor recurrence relation

$$\begin{aligned} B_{i,1}(w_j) &= \begin{cases} 1 & \text{if } d_i \le w_j < d_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ B_{i,k}(w_j) &= \frac{w_j - d_i}{d_{i+k-1} - d_i} B_{i,k-1}(w_j) + \frac{d_{i+k} - w_j}{d_{i+k} - d_{i+1}} B_{i+1,k-1}(w_j) \end{aligned} \tag{2.6}$$

where the vector d represents a nonuniform knot sequence given by

$$d_i = \begin{cases} 0 & 1 \le i \le k \\ \frac{n-k+2}{2}\left[1 - \cos\left(\frac{i-k}{n-k+2}\pi\right)\right] & k+1 \le i \le n+1 \\ n-k+2 & n+2 \le i \le n+k+1 \end{cases} \tag{2.7}$$

Note that the knots are clustered near the end points of the curve in order to provide greater fidelity near the leading and trailing edges of the airfoil. For single-element airfoils, Eq. 2.7 is used to obtain the knot sequence for the lower surface and the same sequence is then reused for the upper surface of the airfoil. For complex multi-element configurations, where the computational grid contains multiple cluster points, the cosine knot distribution, see Eq. 2.7, is replaced by a linear distribution given by $i - k$. This distribution can be adjusted by the user in order to provide a detailed control over the initial location of the B-spline control points. An efficient algorithm for the evaluation of Eq. 2.6 is presented by de Boor [32].

The distance along the B-spline curve is represented by the parameter value w_j for each point j on the surface of the airfoil. The initial values of w are given by

$$w_1 = 0$$

$$w_j = \frac{(n-k+2)}{L_{\mathrm{T}}} \sum_{m=1}^{j-1} \sqrt{L_m} \qquad j = 2, \ldots, N \tag{2.8}$$

where N denotes the total number of airfoil surface points. The segment length between airfoil surface points is represented by L_m and L_{T} is given by

$$L_{\mathrm{T}} = \sum_{m=1}^{N-1} \sqrt{L_m} \tag{2.9}$$

The multiplication by the factor $n-k+2$ in Eq. 2.8 ensures that the maximum value of the parameter vector is equal to the maximum value of the knot vector. Eq. 2.8 is referred to as the centripetal chord length parameterization. This parameterization provides better estimates of w_j for curves with regions of high curvature than the more traditional chord length parameterization [53], which is obtained by removing the square root from Eqs. 2.8 and 2.9. It is important to note that the parameter vector remains constant throughout the optimization procedure.

At the onset of the optimization procedure, it is necessary to determine the location of the B-spline control points that best approximate the initial airfoil shape. Assume that the initial airfoil surface is defined by a set of points $P_j = P(x_j^*, y_j^*)$. A linear least-squares formulation is used to find the location of the control points $D(X_i^{\mathrm{c}}, Y_i^{\mathrm{c}})$ such that the distance between the data point P_j and the corresponding B-spline curve point $C_j = C[x_{\mathrm{a}}(w_j), y_{\mathrm{a}}(w_j)]$ is minimized

$$\min_D \sum_{j=1}^{N} \|P_j - C_j\| \tag{2.10}$$

The obtained control points $(X_i^{\mathrm{c}}, Y_i^{\mathrm{c}})$ and the initial parameter vector w define a curve C for which most of the vectors $P_j - C_j$ are not perpendicular to the tangent C_j'. This is a consequence of the fact that the initial parameter vector w only approximates the location of the data points. A procedure to improve the values of w is proposed by Hoschek [80] and is based on a Newton update

$$\overline{w}_j = w_j + \Delta r_j \frac{d_{n+1} - d_1}{\ell} \tag{2.11}$$

where $\Delta r_j = (P_j - C_j)C_j'/\|C_j'\|$, and ℓ is the length of the control polygon defined by the control points. The parameter correction update is permissible if $\|P_j - C[x_{\mathrm{a}}(\overline{w}_j), y_{\mathrm{a}}(\overline{w}_j)]\| < \|P_j - C_j\|$, otherwise the update is reduced by a factor of two. The least-squares problem, Eq. 2.10, is re-evaluated with the new parameter vector in order to obtain a better set of control points. This

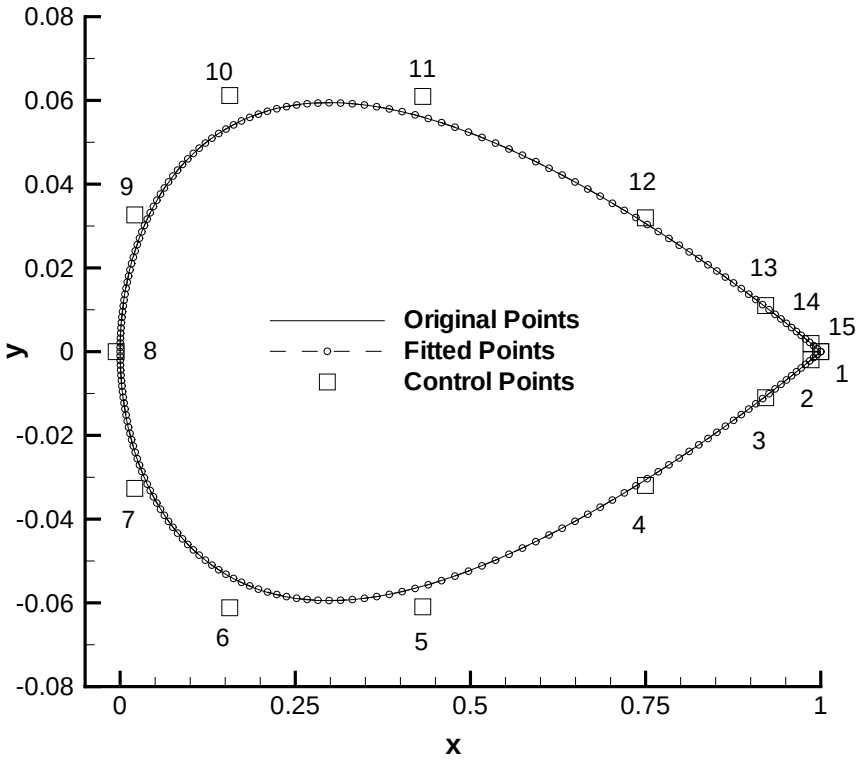


Figure 2.1: B-spline curve and control points for the NACA 0012 airfoil

procedure typically converges within a few hundred iterations, although for some cases a few thousand iterations are required. The iterations do not require significant computational effort. In addition, the B-spline curve is forced to pass through the leading- and trailing-edge points in order to maintain the orientation of the chord line.

An example is shown in Fig. 2.1, where cubic B-splines constructed from 15 control points are used to approximate the NACA 0012 airfoil. By increasing the number of control points, the accuracy of the B-spline curve is improved. For the test cases considered in this work, the maximum error between the B-spline curve and the original data points is below $5 \times 10^{-4}c$. Numerical experiments performed on a number of airfoil shapes suggest that 25 control points are sufficient to reach an error tolerance of $8 \times 10^{-5}c$. Trépanier *et al.* [169] show that for errors below this threshold, the pressure distribution for the B-spline approximation is very close to the pressure distribution for the original airfoil. Note that the manufacturing tolerance used by the aircraft industry is typically assumed to be $2 \times 10^{-4}c$. Figure 2.2 shows an example for a multi-element configuration, where a B-spline curve is fitted over the upper surface of the main element and flap for the NLR 7301 airfoil.

The coordinates of the B-spline control points are used as design variables and we only allow displacements in the vertical direction. The control points associated with leading and

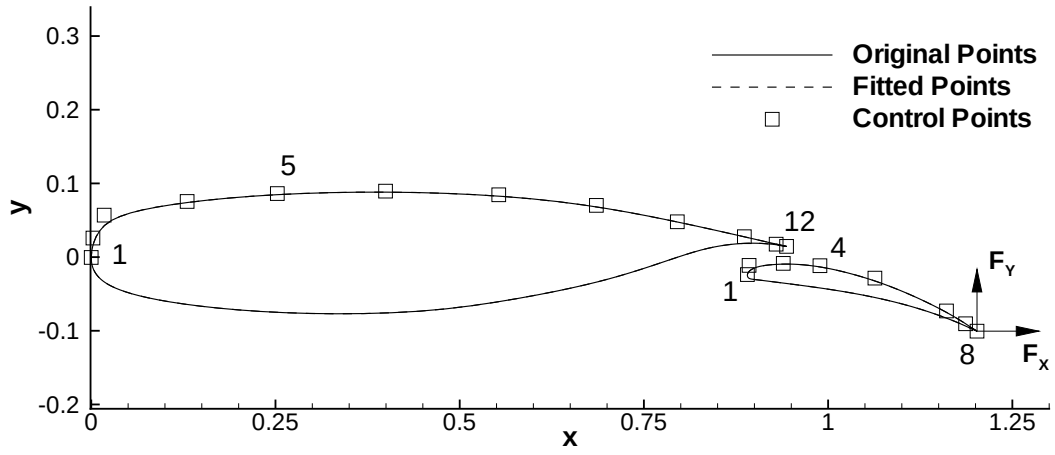


Figure 2.2: B-spline curves, control points, and flap translation design variables for the NLR 7301 configuration

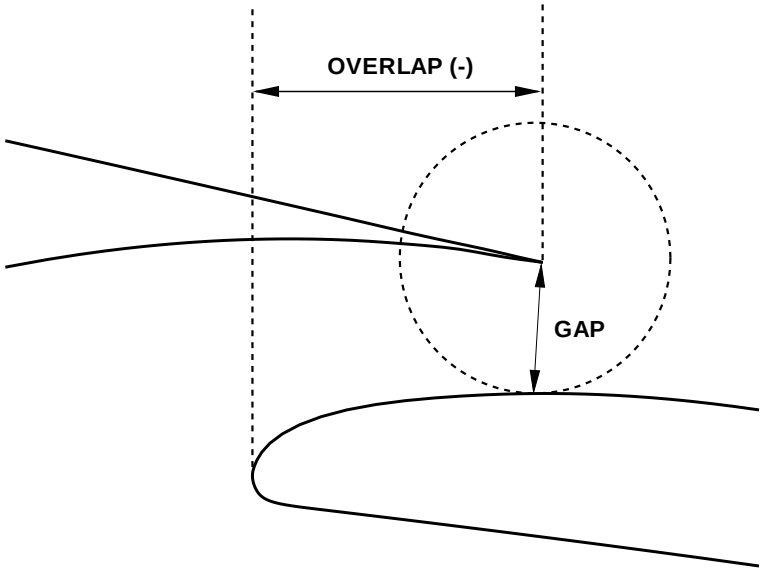


Figure 2.3: Definition of gap and overlap distances

trailing edges remain fixed. In addition to the control points, the design variables may also include the horizontal and vertical translation of the high-lift elements within a multi-element configuration. An example is shown in Fig. 2.2, where the translation design variables are labeled as F_x and F_y. These variables control the gap and overlap distances in the slot region of the configuration as defined in Fig. 2.3.

2.3 Objective Functions

The inverse design, lift-constrained drag minimization, lift enhancement, and maximization of lift-to-drag ratio optimization problems are considered. For the inverse design problem the objective function is given by

$$\mathcal{J} = \frac{1}{2}\int_{\mathcal{B}} (C_{\mathrm{p}} - C_{\mathrm{p}}^*)^2 \mathrm{d}s \tag{2.12}$$

where C_{p}^* represents the target pressure distribution which is user specified and $\mathcal{B}$ denotes the airfoil boundary. By minimizing $\mathcal{J}$, the optimizer finds the shape of the airfoil that, in the least-squares sense, best matches the target pressure distribution.

For the lift-constrained drag-minimization and lift-enhancement problems, the objective function has the form

$$\mathcal{J} = \begin{cases} \omega_{\mathrm{L}} \left(1 - \frac{C_{\mathrm{L}}}{C_{\mathrm{L}}^*}\right)^2 + \omega_{\mathrm{D}} \left(1 - \frac{C_{\mathrm{D}}}{C_{\mathrm{D}}^*}\right)^2 & \text{if } C_{\mathrm{D}} > C_{\mathrm{D}}^* \\ \omega_{\mathrm{L}} \left(1 - \frac{C_{\mathrm{L}}}{C_{\mathrm{L}}^*}\right)^2 & \text{otherwise} \end{cases} \tag{2.13}$$

where C_{D}^* and C_{L}^* represent the target drag and lift coefficients, respectively. The weights ω_{D} and ω_{L} are user-specified constants. We find this formulation of the objective function particularly useful, since it provides an intuitive approach for the selection of weights and additional terms, such as the moment coefficient, can be readily included. For the maximization of lift-to-drag ratio problem we use

$$\mathcal{J} = \frac{C_{\mathrm{D}}}{C_{\mathrm{L}}} \tag{2.14}$$

The weighted-sum method is used for multi-point optimization problems,

$$\mathcal{J}_{\mathrm{m}} = \sum_{i=1}^{N_{\mathrm{m}}} w_i \, \mathcal{J}_i \tag{2.15}$$

where N_{m} denotes the number of design points (typically defined by freestream Mach numbers), and w_i represents a user-assigned weight for each design point. The strengths and weaknesses of the weighted-sum method are well illustrated by Drela [38] for a number of aerodynamic shape optimization problems, including the design of cruise configurations for transonic flow.

2.4 Constraints

The constraint equations, Eq. 2.2, primarily represent airfoil thickness constraints that are used to ensure feasible designs. For multi-element configurations, it is also necessary to constrain the gap and overlap distances in order to prevent collisions among the elements and to ensure a reasonable computational grid (further discussed in Section 3.5). Additional constraints, that are useful for practical design, include the leading-edge radius and the trailing-edge angle. Note that the constraints are a function of only the design variables, i.e. $C_j(X) \leq 0$. For example, the thickness constraints are given by

$$h^*(x_j) - h(x_j) \leq 0 \tag{2.16}$$

where $h^*(x_j)$ represents the minimum allowable thickness at location x_j and $h(x_j)$ represents the current airfoil thickness.

2.5 Flow Equations

The governing flow equations, Eq. 2.3, are the compressible Navier–Stokes equations in conjunction with a one-equation turbulence model. These equations are expressed in generalized curvilinear coordinates in order to facilitate the use of finite differences. The high-Reynolds number aerodynamic flows under consideration allow the use of the thin-layer approximation.

2.5.1 Navier–Stokes Equations

For a two-dimensional flow with density ρ, velocities (u, v) in Cartesian coordinates (x, y), and total energy e, the compressible Navier–Stokes equations are given by

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = \mathcal{R}e^{-1}\left(\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y}\right) \tag{2.17}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.18}$$

is the vector of conservative dependent flow variables. The convective flux vectors are given by

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix} \quad \text{and} \quad F = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ v(e+p) \end{bmatrix} \tag{2.19}$$

The viscous flux vectors are given by

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \varphi_1 \end{bmatrix} \quad \text{and} \quad F_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \varphi_2 \end{bmatrix} \tag{2.20}$$

with

$$\begin{aligned} \tau_{xx} &= (\mu + \mu_t)(4u_x - 2v_y)/3 \\ \tau_{xy} &= (\mu + \mu_t)(u_y - v_x) \\ \tau_{yy} &= (\mu + \mu_t)(-2u_x + 4v_y)/3 \\ \varphi_1 &= u\tau_{xx} + v\tau_{xy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\partial_x(a^2) \\ \varphi_2 &= u\tau_{xy} + v\tau_{yy} + (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\partial_y(a^2) \end{aligned} \tag{2.21}$$

Pressure, p, is related to the flow variables by the equation of state for a perfect gas

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{2.22}$$

and the speed of sound, a, is given by

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{2.23}$$

Note that Eqs. 2.17–2.23 are written in a non-dimensional form. The non-dimensional variables are obtained by using the following scaling parameters

$$x = \frac{\bar{x}}{c}, \quad y = \frac{\bar{y}}{c}, \quad \rho = \frac{\bar{\rho}}{\rho_\infty}, \quad u = \frac{\bar{u}}{a_\infty}, \quad v = \frac{\bar{v}}{a_\infty}, \quad e = \frac{\bar{e}}{\rho_\infty a_\infty^2}, \quad t = \frac{\bar{t}\, a_\infty}{c} \tag{2.24}$$

where the bar symbol denotes dimensional variables, ∞ refers to freestream quantities and c is the chord length of the airfoil. The ratio of specific heats, γ, is 1.4 for air under the conditions of interest here.

Sutherland's law is used to relate the dynamic viscosity, μ, to temperature

$$\mu = \frac{\bar{\mu}}{\mu_\infty} = \frac{(\bar{T}/T_\infty)^{3/2}(T_\infty + S^*)}{\bar{T} + S^*} = \frac{a^3(1 + S^*/T_\infty)}{a^2 + S^*/T_\infty} \tag{2.25}$$

where T_∞ denotes the freestream temperature which is assumed to be 460.0°R and the constant S^* is 198.6°R for air. Turbulent eddy viscosity is denoted by μ_t, $\mathcal{R}e$ is the Reynolds number, $\mathcal{P}r$ is the laminar Prandtl number, and $\mathcal{P}r_t$ is the turbulent Prandtl number. The non-dimensional, laminar Prandtl number is given by

$$\mathcal{P}r = \frac{\bar{c}_p \bar{\mu}}{\bar{\kappa}_t} \tag{2.26}$$

where $\bar{\kappa}_t$ denotes the thermal conductivity and $\bar{c_p}$ is the specific heat at constant pressure. The laminar and turbulent Prandtl numbers are assumed to be constant and are set to 0.72 and 0.90, respectively. The scaling for the Reynolds number is given by

$$\mathcal{R}e = \frac{\rho_\infty \, c \, a_\infty}{\mu_\infty} \tag{2.27}$$

The boundary conditions for the Navier–Stokes equations are discussed in Section 3.2.1.

2.5.2 Turbulence Model

The dynamic eddy viscosity, μ_t in Eq. 2.21, accounts for the effects of turbulence. The Spalart–Allmaras turbulence model [164] is used to determine the value of μ_t. This one-equation transport model, written in non-dimensional and non-conservative form, is given by

$$\frac{\partial \tilde{\nu}}{\partial t} + u\frac{\partial \tilde{\nu}}{\partial x} + v\frac{\partial \tilde{\nu}}{\partial y} = \frac{c_{b1}}{\mathcal{R}e}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu} + \frac{1}{\sigma \mathcal{R}e}\left\{\left(1 + c_{b2}\right)\nabla \cdot \left[\left(\nu + \tilde{\nu}\right)\nabla\tilde{\nu}\right] - c_{b2}\left(\nu + \tilde{\nu}\right)\nabla^2\tilde{\nu}\right\}$$
$$- \frac{1}{\mathcal{R}e}\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d_{\mathrm{w}}}\right)^2 + \mathcal{R}e f_{t1}\Delta U^2 \tag{2.28}$$

where $\tilde{\nu}$ is the non-dimensional working variable. The freestream kinematic laminar viscosity is used to render $\tilde{\nu}$ non-dimensional

$$\tilde{\nu} = \frac{\bar{\tilde{\nu}}}{\nu_\infty} = \bar{\tilde{\nu}}\frac{\rho_\infty}{\mu_\infty} \tag{2.29}$$

The kinematic eddy viscosity, $\nu_t = \mu_t/\rho$, is obtained from

$$\nu_t = \tilde{\nu} f_{v1} \tag{2.30}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.31}$$

and

$$\chi = \frac{\tilde{\nu}}{\nu} \tag{2.32}$$

The production term $\tilde{S}$ is given by

$$\tilde{S} = S \; \mathcal{R}e + \frac{\tilde{\nu}}{\kappa^2 d_{\mathrm{w}}^2} f_{v2} \tag{2.33}$$

where

$$S = \left|\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right| \tag{2.34}$$

is the magnitude of the vorticity, d_{w} is the distance to the closest wall and

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.35}$$

The destruction function f_w is given by

$$f_w = g\left[\frac{1+c_{w3}^3}{g^6+c_{w3}^6}\right]^{\frac{1}{6}} \tag{2.36}$$

where

$$g = r + c_{w2}(r^6 - r) \tag{2.37}$$

and

$$r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d_{\mathrm{w}}^2} \tag{2.38}$$

The functions f_{t1} and f_{t2} control the laminar-turbulent transition locations. In this work, the flow is assumed to be fully turbulent. Therefore, the functions f_{t1} and f_{t2} are set to zero. For a definition of these functions see [164]. The remaining parameters are constants given by

$$\begin{aligned} c_{b1} &= 0.1355 & c_{b2} &= 0.622 \\ c_{w1} &= c_{b1}/\kappa^2 + (1+c_{b2})/\sigma & \kappa &= 0.41 \\ c_{w2} &= 0.3 & c_{w3} &= 2.0 \\ c_{v1} &= 7.1 & \sigma &= \frac{2}{3} \end{aligned} \tag{2.39}$$

The turbulence model boundary conditions are discussed in Section 3.2.1.

2.5.3 The Thin-Layer Approximation and Coordinate Transformation

For attached or mildly-separated aerodynamic flows at high-Reynolds numbers, the Navier–Stokes equations can be simplified by using the thin-layer approximation. For such flows, the viscous derivatives in the streamwise direction are much smaller than the viscous derivatives in the normal direction. The thin-layer approximation therefore neglects the viscous streamwise derivatives.

The grids that are used to discretize the physical domain around the airfoil are constructed such that the grid lines follow the contours of the airfoil in one direction and are locally normal to the airfoil in the other direction. These directions are referred to as ξ and η, respectively. A general transformation

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x,y,t) \\ \eta &= \eta(x,y,t) \end{aligned} \tag{2.40}$$

is used to map the curvilinear grid into a computational domain where the spacing is uniform and equal to one, as illustrated in Fig. 2.4. For further details regarding the thin-layer approximation and the generalized curvilinear coordinate transformation see [143, 167].

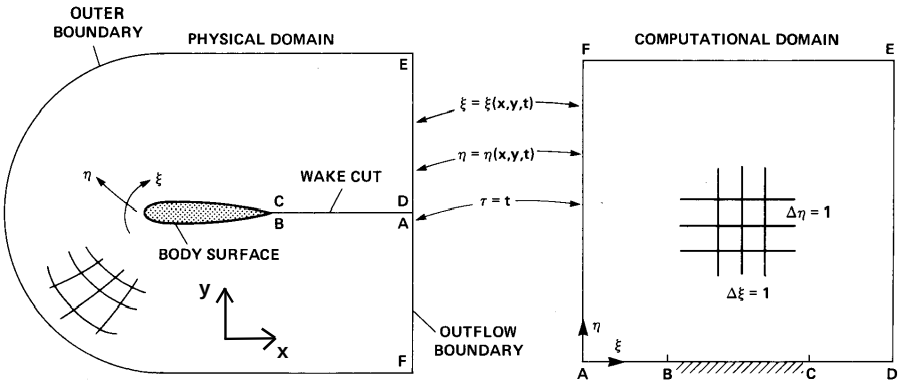


Figure 2.4: Curvilinear coordinate transformation (used with permission from T. H. Pulliam [143])

The thin-layer Navier–Stokes equations, written in generalized curvilinear coordinates, are given by

$$\frac{\partial \widehat{Q}}{\partial \tau} + \frac{\partial \widehat{E}}{\partial \xi} + \frac{\partial \widehat{F}}{\partial \eta} = \mathcal{R}e^{-1}\frac{\partial \widehat{S}}{\partial \eta} \tag{2.41}$$

where

$$\widehat{Q} = J^{-1}Q = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.42}$$

The convective flux vectors are

$$\widehat{E} = J^{-1}\begin{bmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U - \xi_t p \end{bmatrix}, \quad \widehat{F} = J^{-1}\begin{bmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V - \eta_t p \end{bmatrix} \tag{2.43}$$

with

$$U = \xi_t + \xi_x u + \xi_y v, \quad V = \eta_t + \eta_x u + \eta_y v \tag{2.44}$$

the contravariant velocities. The variable J represents the metric Jacobian of the transformation

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi) \tag{2.45}$$

The viscous flux vector is

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x(um_1 + vm_3 + m_4) + \eta_y(um_2 + vm_3 + m_5) \end{bmatrix} \tag{2.46}$$

with

$$\begin{aligned} m_1 &= (\mu + \mu_t)(4\eta_x u_\eta - 2\eta_y v_\eta)/3 \\ m_2 &= (\mu + \mu_t)(\eta_y u_\eta + \eta_x v_\eta) \\ m_3 &= (\mu + \mu_t)(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \\ m_4 &= (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\eta_x \partial_\eta(a^2) \\ m_5 &= (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\eta_y \partial_\eta(a^2) \end{aligned} \tag{2.47}$$

The Spalart–Allmaras turbulence model without the laminar-turbulent transition functions, written in generalized curvilinear coordinates, is given by

$$\frac{\partial \tilde{\nu}}{\partial \tau} + U\frac{\partial \tilde{\nu}}{\partial \xi} + V\frac{\partial \tilde{\nu}}{\partial \eta} = \frac{1}{\mathcal{R}e}\left\{ c_{b1}\tilde{S}\tilde{\nu} - c_{w1} f_w \left(\frac{\tilde{\nu}}{d_{\mathrm{w}}}\right)^2 + \frac{1}{\sigma}\left[(1 + c_{b2})\, T_1 - c_{b2} T_2\right] \right\} \tag{2.48}$$

where

$$\begin{aligned} T_1 = \xi_x \frac{\partial}{\partial \xi}\left[(\nu + \tilde{\nu})\, \xi_x \frac{\partial \tilde{\nu}}{\partial \xi}\right] + \eta_x \frac{\partial}{\partial \eta}\left[(\nu + \tilde{\nu})\, \eta_x \frac{\partial \tilde{\nu}}{\partial \eta}\right] \\ + \xi_y \frac{\partial}{\partial \xi}\left[(\nu + \tilde{\nu})\, \xi_y \frac{\partial \tilde{\nu}}{\partial \xi}\right] + \eta_y \frac{\partial}{\partial \eta}\left[(\nu + \tilde{\nu})\, \eta_y \frac{\partial \tilde{\nu}}{\partial \eta}\right] \end{aligned} \tag{2.49}$$

and

$$T_2 = (\nu + \tilde{\nu})\left[\xi_x \frac{\partial}{\partial \xi}\left(\xi_x \frac{\partial \tilde{\nu}}{\partial \xi}\right) + \eta_x \frac{\partial}{\partial \eta}\left(\eta_x \frac{\partial \tilde{\nu}}{\partial \eta}\right) + \xi_y \frac{\partial}{\partial \xi}\left(\xi_y \frac{\partial \tilde{\nu}}{\partial \xi}\right) + \eta_y \frac{\partial}{\partial \eta}\left(\eta_y \frac{\partial \tilde{\nu}}{\partial \eta}\right)\right] \tag{2.50}$$

Note that all terms containing mixed derivatives have been neglected.

Chapter 3

The Newton–Krylov Algorithm

The aerodynamic shape optimization problem defined by Eqs. 2.1–2.3 is cast as an unconstrained problem. This is accomplished by lifting the side constraints, Eq. 2.2, into the objective function $\mathcal{J}$ using a penalty method. Furthermore, the constraint imposed by the flow equations, Eq. 2.3, is satisfied at every point within the feasible design space, and consequently these equations do not explicitly appear in the formulation of the optimization problem.

For an unconstrained minimization problem, the well-known necessary and sufficient conditions for a design variable vector X^* to be a local minimizer of $\mathcal{J}$ can be stated as follows. It is necessary that the gradient $\mathcal{G}$ of the objective function equal zero and that the Hessian of the objective function is at least positive semi-definite at X^*. The sufficient condition states that the Hessian is required to be positive definite at X^*. For further details see [36, 133]. These optimality conditions can be satisfied only when the gradient is based on the discrete form of Eqs. 2.1–2.3 [71, 117, 63].

The discrete form of the objective functions and constraints is presented in Section 3.1 in conjunction with penalty methods. The discrete form of the flow equations is presented in Sub-

section 3.2.1 and the derivation of the gradient, using the discretized flow equations, is given in Section 3.3. An optimizer based on the quasi-Newton method is used to solve the aerodynamic shape optimization problem. At each step of the optimization procedure, the algorithm requires the value of the objective function, which is provided by the solution of the flow equations, and also the objective function gradient. For a given number of objective function and gradient evaluations, the overall efficiency of the optimization procedure is dominated by the time required to solve the flow equations and compute the gradient. In Subsections 3.2.2 and 3.3.2, we provide a description of a fast strategy for the evaluation of the flow equations and the gradient, respectively. The optimizer is discussed in Section 3.4, and a grid-perturbation strategy that accommodates the airfoil shape and position changes throughout the optimization procedure is presented in Section 3.5. Overall, this chapter provides a detailed description of all components of a new aerodynamic shape optimization algorithm, which we refer to as the Newton–Krylov algorithm.[1]

3.1 Objectives with Constraints

A penalty method is used to combine the objective function with the constraint equations. For example, the formulation for the thickness constraints is given by

$$\mathcal{J} = \mathcal{J}_{\mathrm{d}} + \omega_{\mathrm{T}} \sum_{j=1}^{N_{\mathrm{c}}} C_j \tag{3.1}$$

where $\mathcal{J}_{\mathrm{d}}$ denotes the design objectives defined by Eqs. 2.13 and 2.14, and ω_{T} is a user specified constant. The constraint equations C_j, based on Eq. 2.16, are cast using a quadratic penalty term

$$C_j = \begin{cases} \left[1 - h(x_j)/h^*(x_j)\right]^2 & \text{if } h(x_j) < h^*(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

A similar formulation is used to enforce the lower and upper bounds for the gap and overlap distances, where we use an additional user specified constant ω_{P}. It is important to realize that the purpose of the constraints is two-fold: 1) to prevent the occurrence of infeasible shapes, such as airfoil surface cross-over, during the optimization, and 2) to enforce engineering structural limitations, such as minimum airfoil thickness. The former constraints are usually inactive for the final design, while the latter constraints are typically active. The penalty formulation works well for the design cases presented in this work, but it should be noted that there are well-known weaknesses of penalty methods [66, 133], and more sophisticated strategies for solving constrained problems are given in [48, 37, 181, 133].

[1] The name 'Newton–Krylov' is based on the fact that the optimizer, the flow solver, and the gradient computation algorithm make use of Newton's method and/or a Krylov subspace method.

For the inverse design problem, the objective function given by Eq. 2.12 is replaced by the following discrete form

$$\mathcal{J} = \frac{1}{2}\sum_{j=1}^{N_{\mathrm{A}}}(C_{\mathrm{p}_j} - C^*_{\mathrm{p}_j})^2 \tag{3.3}$$

where N_{A} denotes the number of nodes on the airfoil. Constraints are not required for the inverse design problem since the shape modifications are relatively small for the design cases considered.

3.2 Flow Analysis

Structured grids are used to discretize the physical domain surrounding the airfoil. All grids are generated with the multi-block grid-generation tool AMBER2d [119]. For single-element airfoils, single-block C-topology grids are used, as shown in Fig. 3.1(a), while for multi-element airfoils, H-topology grids are used, as shown in Fig. 3.1(b).

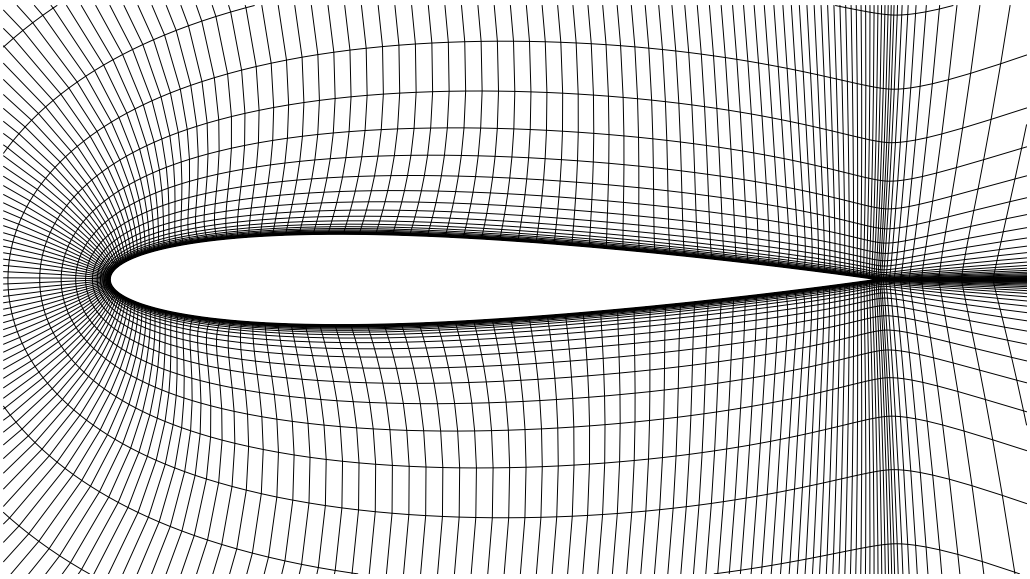
The spatial discretization of the flow equations, Eq. 2.3, is presented in the following section. The algorithm used to determine the steady-state solution of the discretized flow equations is discussed in Subsection 3.2.2.

3.2.1 Spatial Discretization

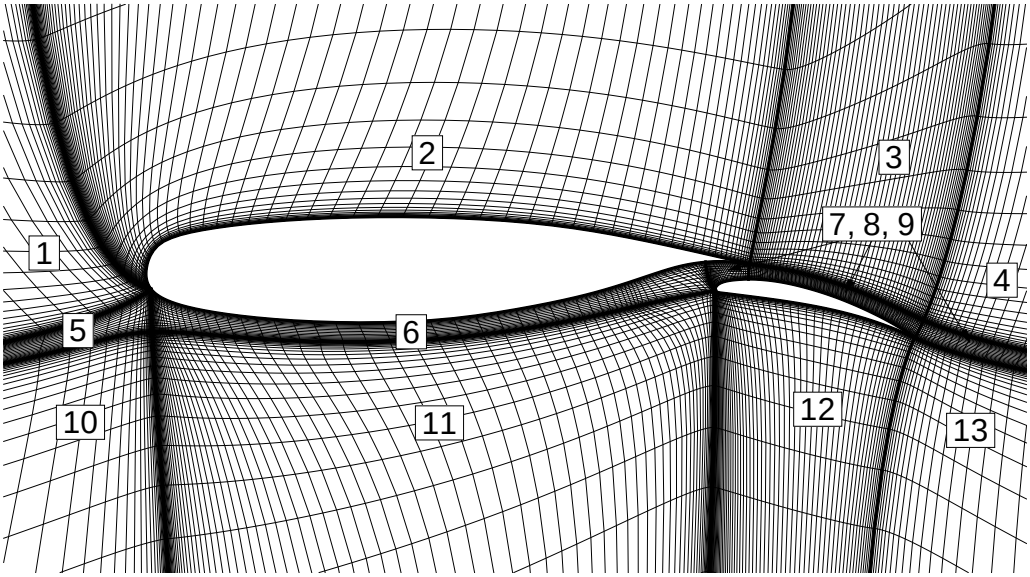
Interior Scheme

The spatial discretization of the thin-layer Navier–Stokes equations is the same as that used in ARC2D [143] for C-topology grids and TORNADO [121] for H-topology grids. The discretization for the inviscid fluxes consists of second-order centered-difference operators with second- and fourth-difference scalar artificial dissipation given by

$$\frac{\partial \widehat{E}}{\partial \xi} \approx \frac{\widehat{E}_{j+1,k} - \widehat{E}_{j-1,k}}{2} - \nabla_\xi A_{\mathrm{D}} \tag{3.4}$$



(a) C-topology grid for the NACA 0012 airfoil



(b) H-topology grid for the NLR 7301 airfoil (block numbering indicated)

Figure 3.1: Examples of structured grids for single- and multi-element airfoils

where

$$
\begin{aligned}
A_{\mathrm{D}} &= d^{(2)}_{j+\frac{1}{2},k}\,\Delta_\xi\, Q_{j,k} - d^{(4)}_{j+\frac{1}{2},k}\,\Delta_\xi \nabla_\xi \Delta_\xi\, Q_{j,k} && (3.5)\\
d^{(2)}_{j+\frac{1}{2},k} &= 2\left(\epsilon\,\sigma\,J^{-1}\right)_{j+\frac{1}{2},k} && (3.6)\\
d^{(4)}_{j+\frac{1}{2},k} &= \max\left[0, 2\,\kappa_4\left(\sigma\,J^{-1}\right)_{j+\frac{1}{2},k} - d^{(2)}_{j+\frac{1}{2},k}\right] && (3.7)\\
\sigma_{j,k} &= |U| + a\sqrt{\xi_x^2 + \xi_y^2} && (3.8)\\
\epsilon_{j,k} &= \kappa_2\left[0.5\Upsilon^*_{j,k} + 0.25\left(\Upsilon^*_{j-1,k} + \Upsilon^*_{j+1,k}\right)\right] && (3.9)\\
\Upsilon^*_{j,k} &= \max\left(\Upsilon_{j+1,k}, \Upsilon_{j,k}, \Upsilon_{j-1,k}\right) && (3.10)\\
\Upsilon_{j,k} &= \frac{|p_{j+1,k} - 2p_{j,k} + p_{j-1,k}|}{|p_{j+1,k} + 2p_{j,k} + p_{j-1,k}|} && (3.11)
\end{aligned}
$$

The operators Δ_ξ and ∇_ξ represent first-order forward and backward difference operators

$$
\begin{aligned}
\Delta_\xi\,(\cdot) &= (\cdot)_{j+1,k} - (\cdot)_{j,k}\\
\nabla_\xi\,(\cdot) &= (\cdot)_{j,k} - (\cdot)_{j-1,k}
\end{aligned} \qquad (3.12)
$$

and κ_2 and κ_4 are constants. Typical values of κ_2 and κ_4 are 1.0 and 0.01, respectively. The scalar coefficient σ is the spectral radius of the flux Jacobian matrix. The term $\Upsilon_{j,k}$ is a pressure switch to control the use of first-order dissipation near shock waves. Values at half nodes are calculated using arithmetic averages given by

$$
(\cdot)_{j+\frac{1}{2},k} = \frac{(\cdot)_{j+1,k} + (\cdot)_{j,k}}{2} \qquad (3.13)
$$

Near computational boundaries the third-order dissipation stencil $Q_{j+2,k} - 4Q_{j+1,k} + 6Q_{j,k} - 4Q_{j-1,k} + Q_{j-2,k}$ is modified to a one-sided first-order stencil [143]. For example, at the first interior node, the stencil $-2Q_{j-1} + 5Q_j - 4Q_{j+1} + Q_{j+2}$ is used, while at the last interior node, the stencil $-2Q_{j+1} + 5Q_j - 4Q_{j-1} + Q_{j-2}$ is used. The computation of grid metrics involves the same centered-difference operator as the inviscid fluxes and first-order one-sided operators are used at the boundaries. Analogous terms appear in the η direction; however, Eq. 3.10 is not used.

The viscous terms in Eq. 2.41 are in the form

$$
\partial_\eta\left(\alpha_{j,k}\,\partial_\eta \beta_{j,k}\right) \qquad (3.14)
$$

The discretization of this term is accomplished by the following conservative three-point stencil

$$
\nabla_\eta\left(\alpha_{j,k+\frac{1}{2}}\,\Delta_\eta \beta_{j,k}\right) = \alpha_{j,k+\frac{1}{2}}(\beta_{j,k+1} - \beta_{j,k}) - \alpha_{j,k-\frac{1}{2}}(\beta_{j,k} - \beta_{j,k-1}) \qquad (3.15)
$$

All values at half nodes are calculated using Eq. 3.13, including laminar and turbulent viscosities.

The Spalart–Allmaras turbulence model is discretized as described in [164, 67]. It is convenient to rewrite the model, Eq. 2.48, in the following steady-state form

$$J^{-1}\left[M(\tilde{\nu}) - P(\tilde{\nu}) + D(\tilde{\nu}) - N(\tilde{\nu})\right] = 0 \tag{3.16}$$

where

$$M(\tilde{\nu}) = U\frac{\partial\tilde{\nu}}{\partial\xi} + V\frac{\partial\tilde{\nu}}{\partial\eta} \qquad \text{convective term} \tag{3.17}$$

$$P(\tilde{\nu}) = \frac{c_{b1}}{\mathcal{R}e}\tilde{S}\tilde{\nu} \qquad \text{production term} \tag{3.18}$$

$$D(\tilde{\nu}) = \frac{c_{w1}f_w}{\mathcal{R}e}\left(\frac{\tilde{\nu}}{d_{\mathrm{w}}}\right)^2 \qquad \text{destruction term} \tag{3.19}$$

$$N(\tilde{\nu}) = \frac{1}{\sigma\,\mathcal{R}e}\left[(1+c_{b2})\,T_1 - c_{b2}T_2\right] \qquad \text{diffusion term} \tag{3.20}$$

and the factor of J^{-1} is introduced in order to improve the scaling of the flow Jacobian matrix, see Subsection 3.2.2.

A first-order upwind discretization is used for the convective term, Eq. 3.17, given by

$$M(\tilde{\nu})_{j,k} = \frac{1}{2}(U_{j,k} + |U_{j,k}|)(\tilde{\nu}_{j,k} - \tilde{\nu}_{j-1,k}) + \frac{1}{2}(U_{j,k} - |U_{j,k}|)(\tilde{\nu}_{j+1,k} - \tilde{\nu}_{j,k}) \tag{3.21}$$

for the ξ direction.[2] The diffusion term, Eq. 3.20, has the same form as the viscous terms, Eq. 3.14, and therefore the same three-point stencil given by Eq. 3.15 is used. The computation of vorticity is given by

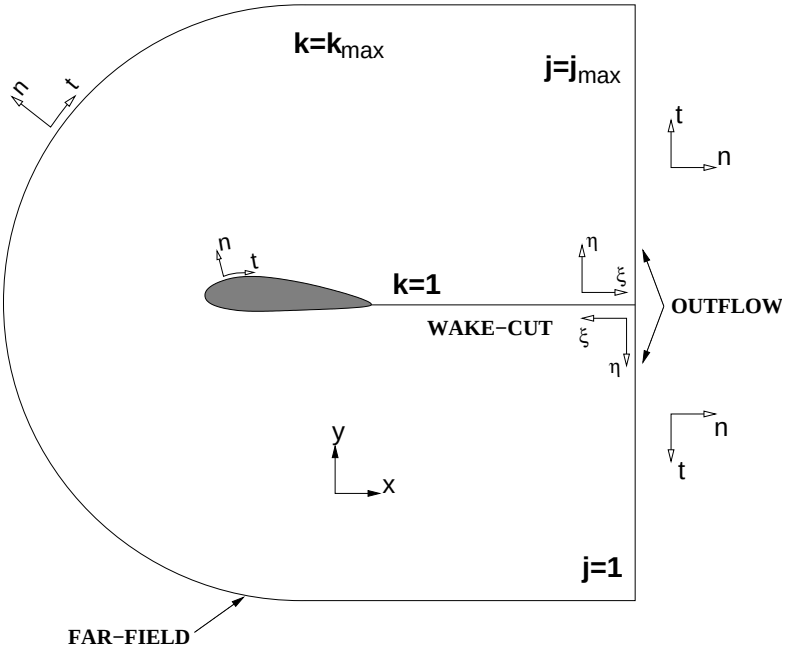
$$S = \frac{1}{2}\,|(v_{j+1,k} - v_{j-1,k})(\xi_x)_{j,k} + (v_{j,k+1} - v_{j,k-1})(\eta_x)_{j,k} \\ -(u_{j+1,k} - u_{j-1,k})(\xi_y)_{j,k} - (u_{j,k+1} - u_{j,k-1})(\eta_y)_{j,k}| \tag{3.22}$$

Boundary Conditions and Block Interfaces

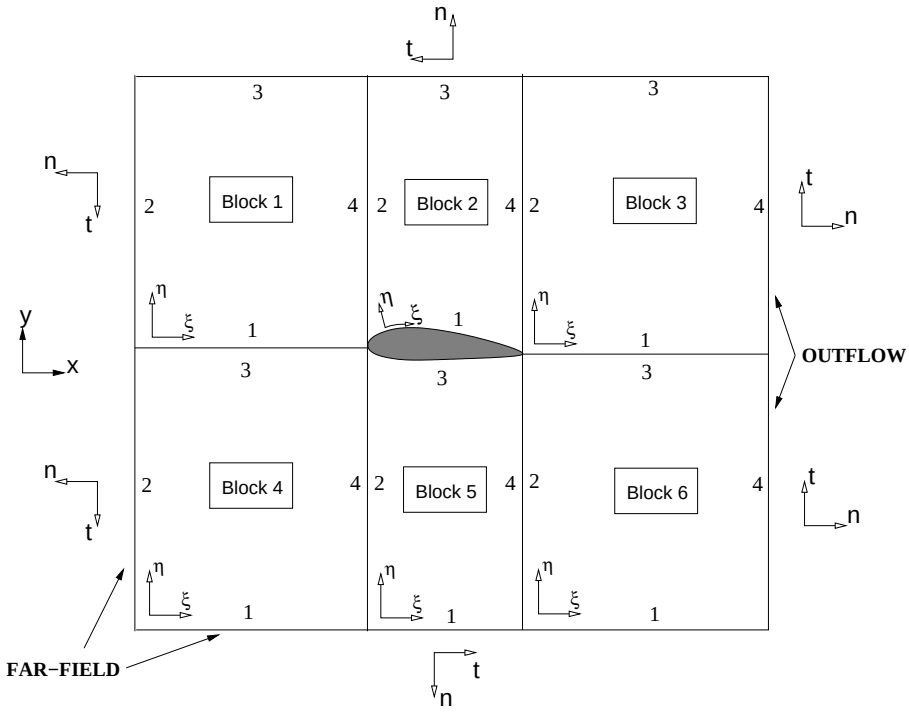
The boundaries of the physical domain consist of the airfoil body, far-field and outflow boundaries. The location of these boundaries is indicated in Fig. 3.2 for C- and H-topology grids. In addition, special consideration is required at interior boundaries such as the wake-cut and block interfaces. Examples of the boundary equations are provided specifically for C-topology domains; however, the modifications required for H-topologies only involve the index values, and the tangent and normal directions as shown in Fig. 3.2.

At the airfoil body, the no-slip boundary condition is applied. Furthermore, the boundary is considered to be adiabatic and the gradient of pressure normal to the body is set to zero [77]. Combining the adiabatic condition and a zero normal-pressure gradient with the perfect gas

[2] An analogous term appears in the η direction.



(a) C-topology domain



(b) H-topology domain (with block and side numbering)

Figure 3.2: Grid boundaries with normal and tangential directions indicated (modified and used with permission from S. De Rango [33])

law results in a zero normal-density gradient. For the turbulence model equation, the working variable $\tilde{\nu}$ is set to zero. These boundary conditions can be expressed as

$$\rho_{j,1} - \rho_{j,2} = 0 \tag{3.23}$$

$$(\rho u)_{j,1} = 0 \tag{3.24}$$

$$(\rho v)_{j,1} = 0 \tag{3.25}$$

$$p_{j,1} - p_{j,2} = 0 \tag{3.26}$$

$$\tilde{\nu}_{j,1} = 0 \tag{3.27}$$

where the index j loops over the nodes on the surface of the airfoil, see Fig. 3.2(a).

At the far-field boundaries, locally one-dimensional Riemann invariants are used

$$R_1 = V_n - \frac{2a}{\gamma - 1} \tag{3.28}$$

$$R_2 = V_n + \frac{2a}{\gamma - 1} \tag{3.29}$$

where V_n denotes the normal velocity component, as shown in Fig. 3.2 and defined in Appendix A. These invariants are associated with the two characteristic speeds $\lambda_1 = V_n - a$ and $\lambda_2 = V_n + a$, respectively. Three additional equations are required, two for the Navier–Stokes equations and one for the turbulence model. The choice of equations which works well in practice [143, 164] is based on the tangential velocity component V_t (for definition see Appendix A), entropy $S = \ln(p/\rho^\gamma)$, and the variable $\tilde{\nu}$ given by

$$R_3 = V_t \tag{3.30}$$

$$R_4 = \frac{\rho^\gamma}{p} \tag{3.31}$$

$$R_5 = \tilde{\nu} \tag{3.32}$$

Depending on the sign of the corresponding characteristic speed, these variables are either extrapolated from the interior values or they are set to the freestream values.

For subsonic inflow, $V_n < 0$, $\lambda_1 < 0$ and $\lambda_2 > 0$. Therefore, the Riemann invariant R_1 is determined from freestream conditions and R_2 is determined by extrapolation from the interior. For subsonic outflow, $V_n > 0$, $\lambda_1 < 0$ and $\lambda_2 > 0$. As in the case for subsonic inflow, R_1 is determined from freestream conditions and R_2 is extrapolated. For inflow, R_3, R_4 and R_5 are set to freestream conditions. For outflow, they are extrapolated from the interior. Zero-order extrapolation [77] is used for all values. The formulas for the computation of the flow variables from the boundary equations are provided in Appendix A.

Overall, the subsonic inflow far-field boundary equations for C-topology grids, see Fig. 3.2(a),

are given by

$$\left(V_n - \frac{2a}{\gamma - 1}\right)_{j,k_{\max}} - \left(V_n - \frac{2a}{\gamma - 1}\right)_{\infty} = 0 \tag{3.33}$$

$$\left(V_n + \frac{2a}{\gamma - 1}\right)_{j,k_{\max}} - \left(V_n + \frac{2a}{\gamma - 1}\right)_{j,k_{\max}-1} = 0 \tag{3.34}$$

$$\left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max}} - S_\infty = 0 \tag{3.35}$$

$$(V_t)_{j,k_{\max}} - (V_t)_\infty = 0 \tag{3.36}$$

$$\tilde{\nu}_{j,k_{\max}} - \tilde{\nu}_\infty = 0 \tag{3.37}$$

where the variable $\tilde{\nu}_\infty$ is set to 1×10^{-3}. For subsonic outflow, the last three equations are replaced by

$$\left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max}} - \left(\frac{\rho^\gamma}{p}\right)_{j,k_{\max-1}} = 0 \tag{3.38}$$

$$(V_t)_{j,k_{\max}} - (V_t)_{j,k_{\max-1}} = 0 \tag{3.39}$$

$$\tilde{\nu}_{j,k_{\max}} - \tilde{\nu}_{j,k_{\max-1}} = 0 \tag{3.40}$$

At the outflow boundary, the use of Riemann invariants is inappropriate due to the entropy gradients in the wake of the airfoil. Practical experience indicates that zero-order extrapolation of the flow variables provides good results. Hence, the outflow boundary equations at $j = 1$ for C-topology grids are given by

$$\rho_{1,k} - \rho_{2,k} = 0 \tag{3.41}$$

$$(\rho u)_{1,k} - (\rho u)_{2,k} = 0 \tag{3.42}$$

$$(\rho v)_{1,k} - (\rho v)_{2,k} = 0 \tag{3.43}$$

$$p_{1,k} - p_{2,k} = 0 \tag{3.44}$$

$$\tilde{\nu}_{1,k} - \tilde{\nu}_{2,k} = 0 \tag{3.45}$$

and similar equations apply at $j = j_{\max}$.

The proximity of the far-field boundary may significantly affect the accuracy of the numerical solution for lifting airfoils [143, 182]. This undesirable effect is moderated by using the far-field circulation correction as described in Appendix A.

For H-topology grids, the block interfaces are averaged in the normal direction. For example, consider the interface between blocks 1 and 5 in Fig. 3.1(b), the residual equation for the average boundary condition is

$$Q^{(1)}_{j,1} - \frac{1}{2}\left(Q^{(1)}_{j,2} + Q^{(5)}_{j,k_{\max}-1}\right) = 0 \tag{3.46}$$

where the superscript denotes the block number. This expression is used for the first three conservative flow variables and also the turbulence model variable, but for the energy equation the pressures are averaged instead. A similar equation is used along the wake-cut for C-topology grids. Block interfaces in the streamwise direction are overlapped. The governing equations are solved for each node at the block interface. Two columns of halo points are used for temporary data storage.

The leading- and trailing-edge points require additional consideration since each of these points maps to four distinct points in the computational domain for H-topology grids. Dual-value leading- and trailing-edge points are used. For example, at the trailing edge the boundary equations are applied from the upper and lower surfaces and these values are then copied to the downstream blocks. A dual-value trailing edge is also used for C-topology grids.

3.2.2 Flow Solver

The spatial discretization of the governing equations leads to a nonlinear system of equations

$$R(\widehat{Q}, X) = 0 \tag{3.47}$$

where $\widehat{Q}$ is the discrete vector of conservative dependent flow variables[3] with dimension N_{F}. For a grid with N_{B} blocks, the total number of flow variables is given by $N_{\mathrm{F}} = 5 \sum_{i=1}^{N_{\mathrm{B}}} (j_{\max} \times k_{\max})_i$. Hence, at each node (j, k) within the computational domain

$$\widehat{Q}_{j,k} = \left(J^{-1}Q\right)_{j,k} = J_{j,k}^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \\ \tilde{\nu} \end{bmatrix}_{j,k} = \begin{bmatrix} \widehat{Q}_{\mathrm{M}} \\ \widehat{Q}_{\mathrm{T}} \end{bmatrix}_{j,k} \tag{3.48}$$

The notation $\widehat{Q}_{\mathrm{M}}$ and $\widehat{Q}_{\mathrm{T}}$ is introduced for convenience. The symbol $\widehat{Q}_{\mathrm{M}}$ denotes the flow variables for the mean flow, i.e. the Navier–Stokes equations, and $\widehat{Q}_{\mathrm{T}}$ denotes the turbulence model variable. Similarly, the residual equations, Eq. 3.47, consist of the Navier–Stokes equations, R_{M}, and the Spalart–Allmaras turbulence model equation, R_{T}. Note that the residual equations include the interior, boundary, and wake-cut or block interface equations. Although R is written as a function of the design variables, see Eq. 3.47, we emphasize that during a flow solution the design variables, and consequently the computational grid, are constants.

A classic method for the solution of Eq. 3.47 is Newton's method

$$\mathcal{A}^{(n)} \, \Delta\widehat{Q}^{(n)} = -R^{(n)} \tag{3.49}$$

[3]For the remainder of this document, $\widehat{Q}$ denotes the discrete flow variable vector instead of the continuous flow variable vector denoted in Chapter 2.

where

$$\mathcal{A} = \frac{\partial R}{\partial \widehat{Q}} \tag{3.50}$$

is the flow Jacobian matrix evaluated at $\widehat{Q}^{(n)}$ and

$$\widehat{Q}^{(n+1)} = \widehat{Q}^{(n)} + \Delta\widehat{Q}^{(n)} \tag{3.51}$$

The flow Jacobian matrix is a large sparse matrix with dimensions $N_{\mathrm{F}} \times N_{\mathrm{F}}$. The matrix is non-symmetric and usually ill-conditioned. The crux of an effective implementation of Newton's method consists of two problems:

1. An efficient solution of the linear system of equations represented by Eq. 3.49
2. An efficient startup procedure in order to provide the initial vector $\widehat{Q}^{(0)}$

The following two subsections describe the algorithms that address these two problems. We use the term "outer iterations" to denote the iterations associated with the Newton updates, i.e. Eq. 3.51, while the term "inner iterations" is used to denote the iterations required for the solution of the linear system given by Eq. 3.49.

Preconditioned GMRES Algorithm

The solution of Eq. 3.49 is obtained using the generalized minimal residual (GMRES) Krylov subspace method [152, 150]. The restarted version of the GMRES algorithm is used, denoted by GMRES(m), where m represents the number of search directions. The matrix-vector products at each iteration of GMRES are approximated with first-order forward differences. Consequently, the flow Jacobian matrix does not need to be formed explicitly. This implementation of the algorithm is referred to as matrix-free GMRES. Preconditioning of the flow Jacobian is required in order to cluster its eigenvalue spectrum around unity, and therefore *significantly* improve the efficiency of the GMRES algorithm. The preconditioning matrix is applied from the right side, such that Eq. 3.49 becomes

$$\mathcal{A}\mathcal{M}^{-1}\mathcal{M}\Delta\widehat{Q} = -R \tag{3.52}$$

where $\mathcal{M}$ represents the preconditioning matrix. Although the preconditioning matrix can be applied from the left or both sides, an advantage of right preconditioning is that the residual of the linear system is not affected by the preconditioner. Saad [150] indicates that the choice of the side for preconditioning should not have a significant impact on the convergence rate of GMRES unless $\mathcal{M}$ is ill-conditioned. An outline of the preconditioned, matrix-free GMRES algorithm is provided in Appendix B.

The matrix $\mathcal{M}$ is decomposed using incomplete LU factorization [111] with a level of fill k, such that

$$\mathcal{M} = \tilde{L} \cdot \tilde{U} \approx L \cdot U = \mathcal{A} \tag{3.53}$$

This is referred to as ILU(k) preconditioning where the level of fill controls the accuracy of the incomplete factorization. Higher fill levels allow more non-zero entries during the Gaussian elimination process, resulting in better approximations of the $L \cdot U$ factors. This gain in accuracy is balanced by the increase in the computational effort and storage requirements. Saad [150] provides a detailed overview of various ILU algorithms.

The choice of $\mathcal{M}$ is a critical aspect of effective ILU(k) preconditioners. The existence and stability of the ILU(k) factorization has been established only for M-matrices[4] [111], even though the factorization has been applied to a much wider spectrum of problems [23, 21]. Considering Eq. 3.52, a natural choice for the preconditioner is the flow Jacobian matrix. However, since this matrix is non-symmetric and non-diagonally dominant, the ILU(k) factorization may generate inaccurate and unstable $\tilde{L} \cdot \tilde{U}$ factors [50, 51, 18]. Broad criteria for a "good" preconditioner are a matrix that is easier to invert than $\mathcal{A}$, while its inverse remains a good approximation of $\mathcal{A}^{-1}$ [142, 23]. In order to describe the formation of $\mathcal{M}$, where such criteria are fulfilled, it is necessary to present a detailed description of the flow Jacobian. Note that the flow Jacobian matrix is also used for the computation of the objective function gradient, discussed in Subsection 3.3.2.

Consider a *coarse* H-topology grid surrounding the NACA 0012 airfoil as shown in Fig. 3.3. The six blocks are separated such that the block boundaries are clearly visible. The leading edge of the airfoil is located at nodes 21, 26, 100, and 105. The trailing edge of the airfoil is located at nodes 46, 51, 125, and 130. The ordering of nodes in Fig. 3.3 is referred to as natural ordering. This ordering proceeds in the normal direction in order obtain a smaller matrix bandwidth.[5] The corresponding entries in the flow Jacobian matrix are shown in Fig. 3.4. These entries are consistent with the residual equations presented in Subsection 3.2.1, and each entry represents a 5×5 block matrix consistent with Eq. 3.48. For example, the complete interior scheme (including artificial dissipation) is first applied at node 13, which corresponds to the 9 entries shown on line 13 in Fig. 3.4. Similarly, the reader can verify the treatment of boundaries and

[4]Matrix B is an M-matrix if [150]:

1. $b_{i,i} > 0$ for $i = 1, \ldots, n$
2. $b_{i,j} \leq 0$ for $i \neq j, \quad i, j = 1, \ldots, n$
3. B is non-singular
4. $B^{-1} \geq \varnothing$, where $\varnothing$ denotes a zero matrix

[5]For practical grids (especially C-topology grids), the number of nodes in the streamwise direction usually exceeds the number of nodes in the normal direction.

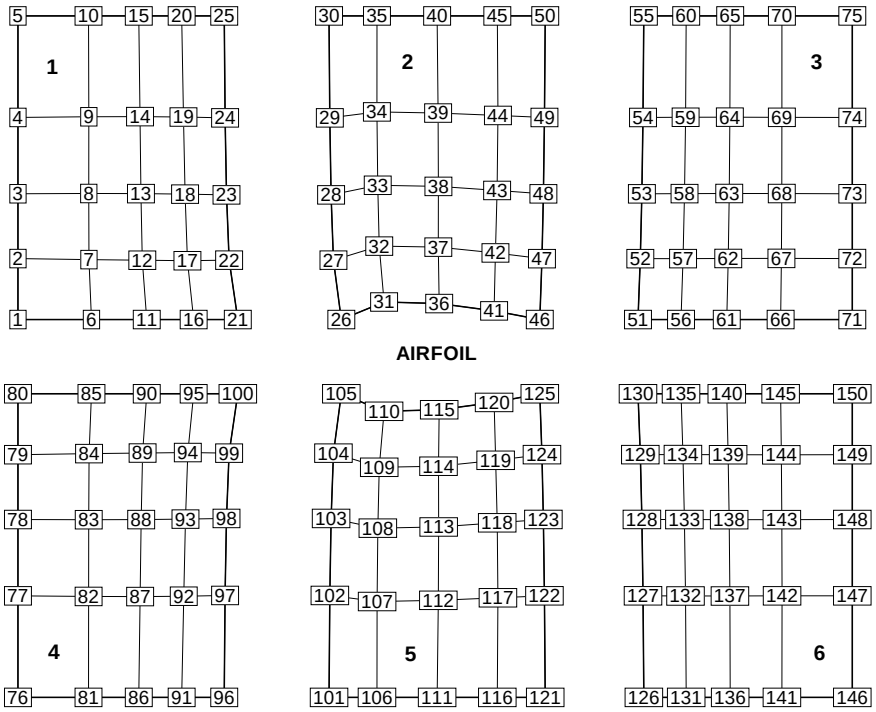


Figure 3.3: Natural node order for H-topology grids

block interfaces. At streamwise block interfaces, we solve the Navier–Stokes equations on both sides of the interface instead of using slave conditions.[6] Although the Navier–Stokes equations contribute slightly more fill to the Jacobian matrix, as shown in Fig. 3.4, scaling problems are avoided. The 5×5 blocks within the flow Jacobian matrix can be represented by

$$\begin{pmatrix} \begin{bmatrix} \\ \dfrac{\partial R_{\mathrm{M}}}{\partial \widehat{Q}_{\mathrm{M}}} \\ 4 \times 4 \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \dfrac{\partial R_{\mathrm{M}}}{\partial \widehat{Q}_{\mathrm{T}}} \\ 4 \times 1 \\ \\ \end{bmatrix} \\ \begin{bmatrix} \\ \dfrac{\partial R_{\mathrm{T}}}{\partial \widehat{Q}_{\mathrm{M}}} \\ 1 \times 4 \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \dfrac{\partial R_{\mathrm{T}}}{\partial \widehat{Q}_{\mathrm{T}}} \\ 1 \times 1 \\ \\ \end{bmatrix} \end{pmatrix} \tag{3.54}$$

where the contributions from the Navier–Stokes and turbulence model equations are clearly distinguished.

Pueyo [141] derived the flow Jacobian for the Navier–Stokes equations discretized on single-block C-topology grids. The spatial discretization of the Navier–Stokes equations in [141] is identical to the spatial discretization presented in Subsection 3.2.1. In his derivation, the lami-

[6] An example slave condition for node 28 in Fig. 3.3 is given by $Q_{28} - Q_{23} = 0$.

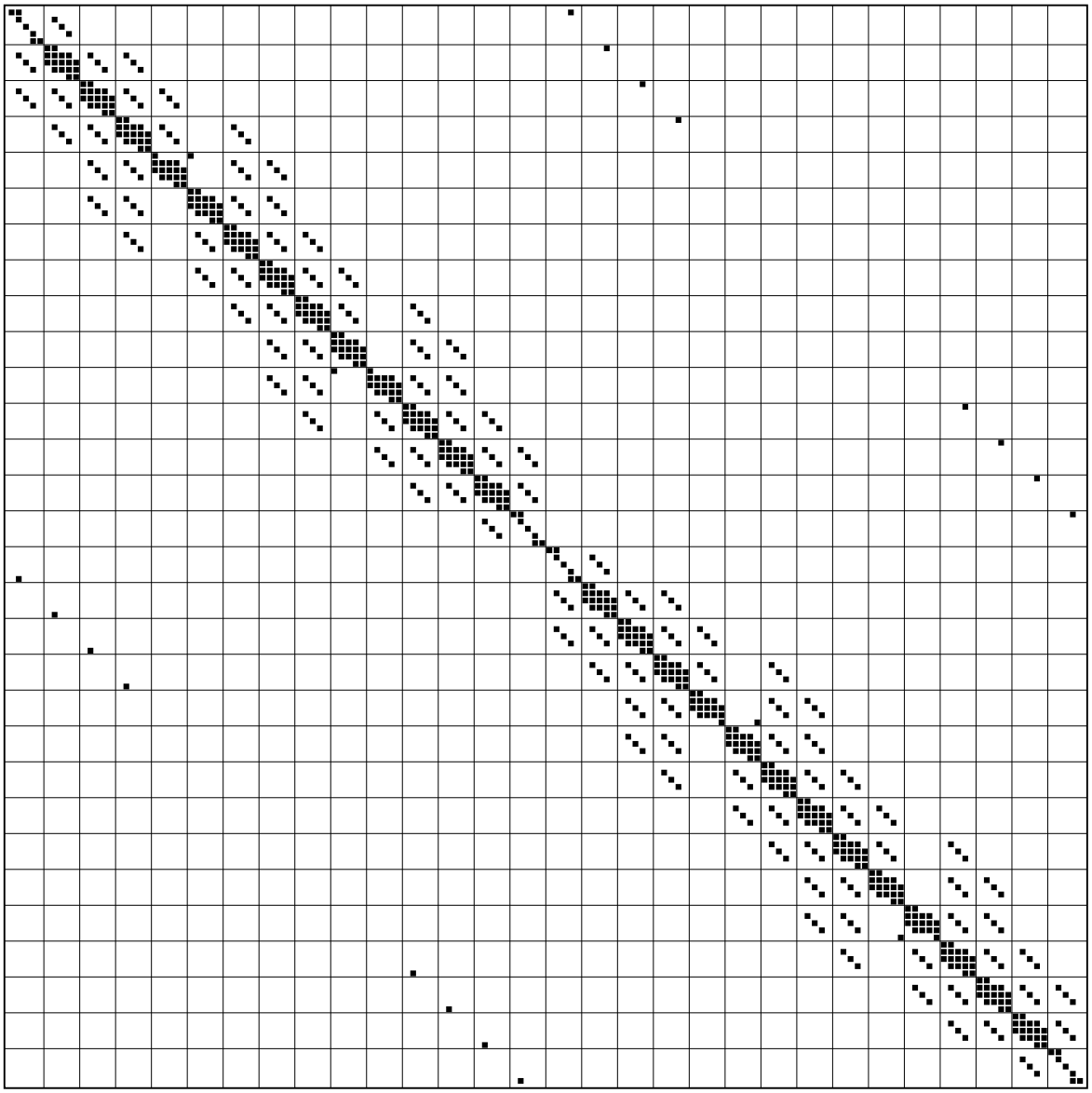


Figure 3.4: Entries for the flow Jacobian matrix based on Fig. 3.3 (Each entry denotes a 5×5 block matrix.)

nar viscosity, turbulent viscosity, the vortex strength due to the far-field circulation correction, and the artificial-dissipation coefficients, i.e. Eqs. 3.6 and 3.7, are treated as constants with respect to the flow variables. Furthermore, the residual equation for the turbulence model is not required since the effects of turbulence were modelled with an algebraic turbulence model. Hence, the flow Jacobian blocks are reduced to 4×4 matrices represented by an approximation to the term $\partial R_{\mathrm{M}} / \partial \widehat{Q}_{\mathrm{M}}$. For example, at an interior node the four rows of the flow Jacobian with the corresponding nine block entries are given by

$$\delta_{\xi}^{(\mathrm{a})} \widehat{A} - \delta_{\xi}^{(\mathrm{s})} \widehat{D}_{\xi} + \delta_{\eta}^{(\mathrm{a})} \widehat{B} - \delta_{\eta}^{(\mathrm{s})} \widehat{D}_{\eta} - \mathcal{R}e^{-1} \delta_{\eta}^{(\mathrm{v})} \widehat{K} \tag{3.55}$$

where the matrices $\widehat{A}$, $\widehat{B}$, and $\widehat{K}$ are the flux Jacobians, defined by

$$\widehat{A} = \frac{\partial \widehat{E}}{\partial \widehat{Q}}, \quad \widehat{B} = \frac{\partial \widehat{F}}{\partial \widehat{Q}}, \text{ and } \quad \widehat{K} = \left. \frac{\partial \widehat{S}}{\partial \widehat{Q}} \right|_{(\mu+\mu_t)=\text{const.}}$$

and $\widehat{D}$ represents a diagonal matrix with the contribution of the artificial-dissipation coefficients based on Eqs. 3.6 and 3.7. The flux Jacobians are provided in [144, 143, 141]. The operator $\delta^{(\mathrm{a})}$ denotes the antisymmetric centered-difference stencil, while $\delta^{(\mathrm{s})}$ denotes the symmetric second- and fourth-difference artificial-dissipation stencils, see Eq. 3.4, and $\delta^{(\mathrm{v})}$ denotes the stencil for the viscous flux vector, see Eq. 3.15.

Pueyo [141] also derived the Jacobian matrices associated with the differentiation of the boundary conditions. It is important to pivot and scale the entries within the flow Jacobian that correspond to the blocks for the far-field and body boundary conditions. These entries are not of the same order of magnitude as the entries for the interior nodes, which can cause poor convergence of GMRES.

We extend the work of Pueyo [141] to include the Spalart–Allmaras turbulence model, Eq. 3.16. The differentiation of laminar viscosity, turbulent viscosity, and the artificial-dissipation coefficients is also included. Due to the complexity of the pressure switch equations, Eqs. 3.9–3.11, the coefficient ϵ in Eq. 3.9 is treated as a constant with respect to the flow variables. An additional complication is the differentiation of the far-field circulation correction, presented in Subsection 3.2.1 and Appendix A. In particular, the computation of the vortex strength leads to the coupling of airfoil surface nodes and far-field boundary nodes. Although this contribution can be differentiated as described by Korivi *et al.* [96], in this work we treat the vortex strength as a constant with respect to the flow variables as well. It is important to realize that these two simplifications do not affect the steady-state flow solution. Furthermore, their impact on the convergence rate of the flow equations is relatively minor since they only influence the preconditioning matrix. These simplifications, however, do affect the accuracy of the objective function gradient, which is discussed in Subsection 3.3.2 and Section 4.3.

Overall, the differentiation of the residual equations is a tedious but relatively straightforward process. One of the difficulties is the differentiation of the absolute value function, which is used for the evaluation of the spectral radius, Eq. 3.8, the upwind discretization of convective terms for the turbulence model, Eq. 3.21, and the computation of vorticity, Eq. 3.22. The derivative of the absolute value function is assumed to be

$$\frac{\mathrm{d}|b|}{\mathrm{d}b} = \begin{cases} 1 & \text{if } b \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.56}$$

The preconditioning matrix, $\mathcal{M}$, is identical to the flow Jacobian matrix except for the treatment of the artificial-dissipation coefficients, Eqs. 3.6 and 3.7. First, these coefficients are assumed to be constant with respect to the flow variables. Second, the matrix $\mathcal{M}$ is formed with only second-difference dissipation, but the second-difference coefficient is combined with the fourth-difference coefficient as follows,

$$d_l^{(2)} = d_r^{(2)} + \phi\, d_r^{(4)} \tag{3.57}$$

where the subscript r denotes the contribution from the right-hand side, and the subscript l denotes the resulting left-hand side value used in the preconditioner. Fast convergence is obtained with the value of ϕ set to 6.0, which has been determined through numerical experiments. We emphasize that this modification does not affect the steady-state solution.

As a result of Eq. 3.57, the preconditioning matrix contains 5 blocks per interior node, as shown in Fig. 3.5. This provides a significant reduction in the computational effort and storage requirements for the ILU(k) factorization. Furthermore, Eq. 3.57 improves the diagonal dominance of the preconditioning matrix. This approach is similar to the "diagonal shift" strategy suggested by Chow and Saad [23]. Pueyo and Zingg [142] discuss an inviscid-flow example where they demonstrate that an ILU(0) preconditioner based on this approximate-flow Jacobian provides a better estimate of $\mathcal{A}^{-1}$ than an ILU(0) preconditioner based on the exact-flow Jacobian. The present preconditioning matrix can be regarded as a compromise between two commonly used approaches [131, 132]: 1) a Jacobian matrix based on the first-order upwind discretization of the flow equations and 2) the exact flow Jacobian. This novel intermediate preconditioner is significantly more effective than either of these approaches, as discussed further in Chapter 4.

The block structure of the preconditioning matrix is well suited for a block ILU(k) factorization. Practical experience suggests that a block ILU(k) factorization is generally superior to a scalar ILU(k) factorization in terms of performance [21]. The approach that is actually implemented is based on a scalar ILU(k) factorization, but any zero entries within a block are treated as non-zeroes. This approach is referred to as block-fill ILU(k) [BFILU(k)] [142].

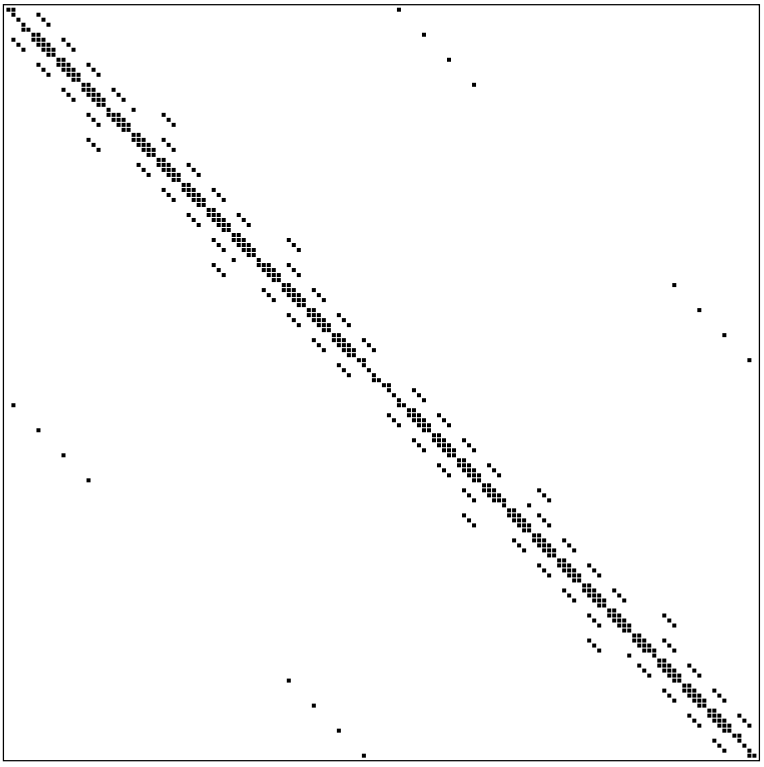


Figure 3.5: Block entries for the preconditioning matrix before ILU(k), based on Fig. 3.3

The ordering of the grid nodes can have a significant impact on the quality of the ILU(k) factorization, and consequently, the convergence of the GMRES algorithm [39, 40, 14]. The natural ordering, shown in Fig. 3.3, leads to entries in the preconditioning matrix which are far from the main diagonal (large bandwidth), as shown in Fig. 3.5. In a complete LU factorization, the remote matrix entries contribute many additional non-zeroes. This has motivated the development of ordering (and reordering) algorithms for sparse-direct solvers that minimize the computational cost of the factorization.

For ILU(k) factorization, the ordering algorithms also provide a reduction in computational cost, except for ILU(0) where the fill pattern is fixed by the original matrix. More importantly, experimental evidence suggests that the ordering algorithms have a significant influence on the quality of the incomplete factorization, since the factorization depends on the "local" values of the matrix coefficients [40, 50]. Studies by Benzi *et al.* [14] show that the greatest benefits of reordering occur for non-symmetric matrices. Overall, many studies [103, 39, 40, 14, 142, 17, 175] indicate that the most efficient and consistent preconditioners are obtained using the reverse Cuthill–McKee (RCM) reordering algorithm [28] (see also [150]). The RCM algorithm, based on a symmetric graph of the preconditioning matrix, is used for all cases presented in this work.

The RCM algorithm is sensitive to the initial ordering of the grid nodes and the selec-

tion of the root node [184]. For single-block C-topology grids, we follow the work of Pueyo and Zingg [142] and use the double-bandwidth ordering to determine the initial node order. Referring to Fig. 3.2(a), the ordering begins at node ($j = 1,\ k = k_{\max}$) and proceeds upward across the wake-cut. Unfortunately, the double-bandwidth ordering is difficult to generalize for multi-element H-topology grids. For these cases, the initial node ordering is the reverse of the natural ordering, as shown in Fig. 3.6(a). Note that both the double-bandwidth and the reverse orderings number the nodes in an "upwind" direction, which provides superior performance. Numerical experiments performed on simple H-topology grids indicate that the performance of preconditioners based on the reverse initial ordering is comparable to the double-bandwidth initial ordering. An example of an RCM reordering is shown in Fig. 3.6(b). The entries in the preconditioning matrix resulting from an RCM reordering are shown in Fig. 3.7. The bandwidth of the preconditioning matrix has been significantly reduced.

The exact solution of Eq. 3.49 is computationally expensive and is not necessary for a rapid convergence of Newton's method from a sufficiently good initial guess. This relaxation of accuracy on the solution of the linear system of equations leads to inexact-Newton methods [35]

$$\left|\left| R^{(n)} + \mathcal{A}^{(n)}\, \Delta\widehat{Q}^{(n)} \right|\right| \leq \zeta^{(n)} \left|\left| R^{(n)} \right|\right| \tag{3.58}$$

where the parameter $\zeta^{(n)} \in (0, 1)$ controls the degree of convergence of the linear system at each outer iteration. If $\zeta^{(n)} = 0$ for all n, the linear system is solved exactly at every outer iteration and Newton's method is recovered.

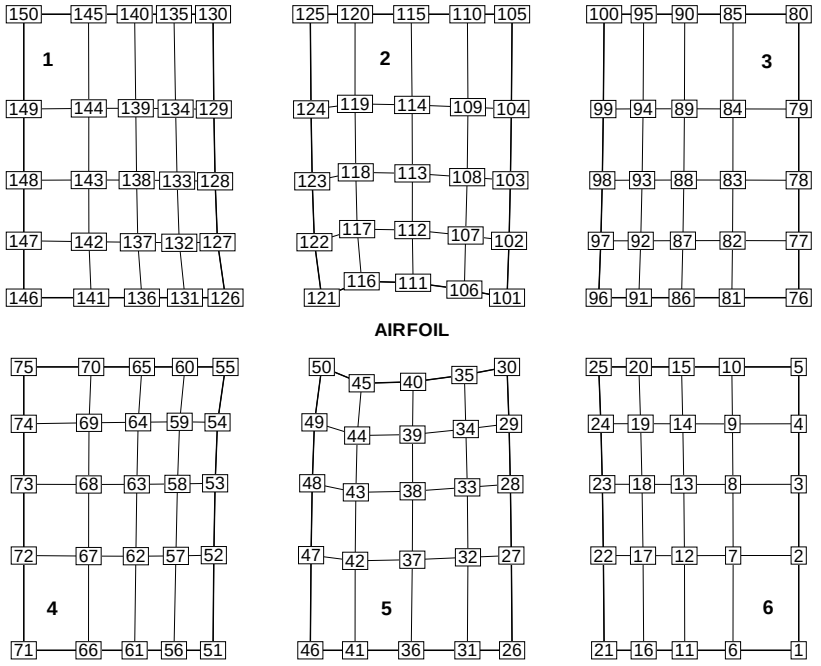
It is possible to choose non-zero values of $\zeta^{(n)}$ and obtain superlinear or even quadratic convergence [35]. However, such choice of $\zeta^{(n)}$ is not necessarily optimal in terms of overall computational efficiency due to the strong possibility of oversolving the linear problem, especially for early outer iterations [43]. A popular approach is to select $\zeta^{(n)}$ based on local residual values such that a linear convergence rate is obtained, but oversolving is avoided [116, 155, 17, 142]. We adopt an even simpler strategy from [142], which still converges linearly, and is summarized as follows:

1. $\zeta^{(n)} = 0.5$ for the first 10 outer iterations

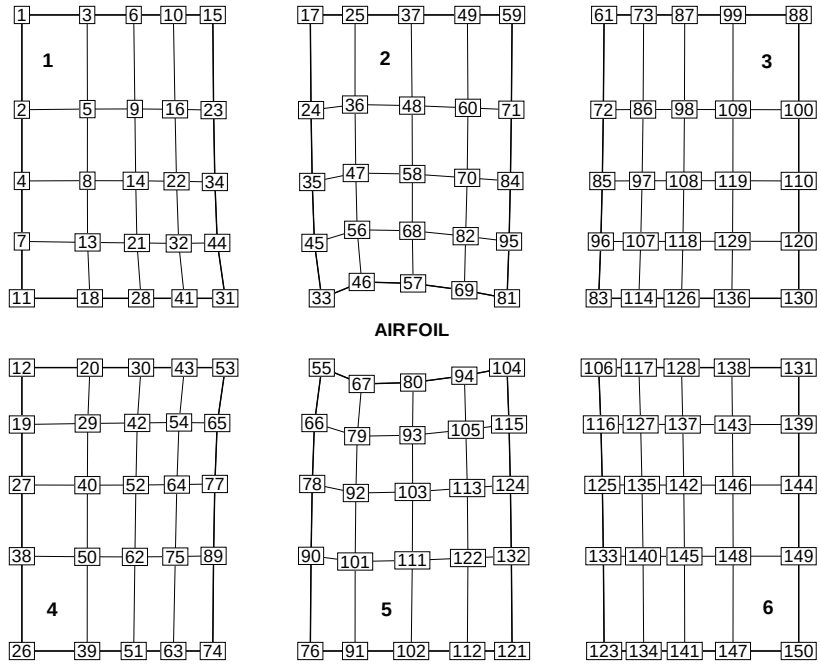
2. $\zeta^{(n)} = 0.1$ for any remaining outer iterations

We select the number of GMRES search directions, m, such that these exit criteria are satisfied without performing GMRES restarts. On the basis of numerical experiments and the work of Pueyo and Zingg [142][7], the value of m equal to 40 provides robust and efficient performance. If for any outer iteration the number of GMRES search directions reaches 40, we perform a

[7] Pueyo and Zingg use GMRES(20) and allow one restart for a maximum total of 40 inner iterations.



(a) Reverse initial ordering



(b) RCM reordering based on reverse initial ordering

Figure 3.6: Node ordering strategy for H-topology grids

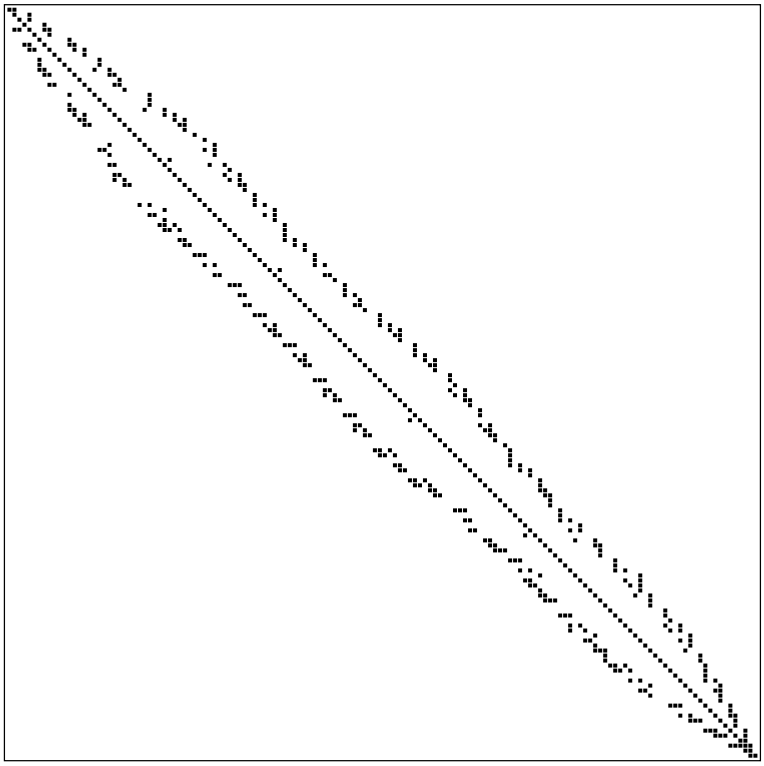


Figure 3.7: Block entries for RCM-reordered preconditioning matrix, before ILU(k)

Newton update, i.e. no GMRES restarts are allowed. Furthermore, the preconditioning matrix is formed only on the first outer iteration and is kept frozen for the rest of the outer iterations. Pueyo and Zingg [142] demonstrate that this strategy provides approximately a factor of two saving in computational time.

Startup Algorithm

A startup algorithm is required in order to ensure global convergence[8] of Newton's method. Backtracking line-search algorithms based on strategies for optimization problems and the implicit-Euler time-marching method are among the most commonly used startup algorithms [36, 13, 155, 175, 7, 9, 22]. In this work, the implicit-Euler time-marching method in conjunction with approximate factorization provides an efficient startup algorithm.

The application of the implicit-Euler time-marching method to the unsteady Navier–Stokes equations, Eq. 2.41, results in

$$\left[\frac{\mathrm{I}}{\Delta t} + \mathcal{A}\right] \Delta\widehat{Q}^{(n)} = -R^{(n)} \tag{3.59}$$

where Δt denotes a time step. Note that as Δt approaches infinity, the Newton equation,

[8] In the context of numerical methods for nonlinear equations, global convergence refers to a method that converges to some solution from almost *any* starting point [36].

Eq. 3.49, is recovered. Although startup algorithms based on an increasing time step can provide an effective relaxation strategy [175, 9, 22], the present algorithm is motivated by the results obtained in [142] and is based on a constant time step[9], but further simplifications are made to the implicit (left-hand) side of Eq. 3.59.

The present algorithm uses the approximate-factorization approach of ARC2D in diagonal form [143] in conjunction with a similar approximate-factorization approach for the turbulence model equation [164, 67]. Hence, the Navier–Stokes and turbulence model equations are solved in a loosely-coupled manner and the boundary conditions are treated only on the explicit (right-hand) side. Considering the Navier–Stokes equations, when the approximate-factorization method of Beam and Warming [12] is applied to Eq. 3.59, the equations take on the following form

$$\left[\mathrm{I} + \Delta t\, \delta_\xi^{(\mathrm{a})}\, \widehat{A}\right] \left[\mathrm{I} + \Delta t\, \delta_\eta^{(\mathrm{a})}\, \widehat{B} - \Delta t\, \mathcal{R}e^{-1} \delta_\eta^{(\mathrm{v})}\, \widehat{K}\right] \Delta\widehat{Q}^{(n)} = -\Delta t\, R^{(n)} \tag{3.60}$$

where the contribution from the artificial-dissipation terms has been omitted for simplicity, see Eq. 3.55. The computational complexity of Eq. 3.60 is further reduced by applying the diagonal form of Pulliam and Chaussee [144]. The eigenvalue decomposition of the inviscid flux-Jacobian matrices is given by

$$\Lambda_\xi = T_\xi^{-1} \widehat{A}\, T_\xi \tag{3.61}$$

$$\Lambda_\eta = T_\eta^{-1} \widehat{B}\, T_\eta \tag{3.62}$$

where the matrices Λ_ξ and Λ_η are diagonal matrices whose elements are the eigenvalues of the inviscid flux Jacobians. The matrix T_ξ has the eigenvectors of $\widehat{A}$ as columns, and T_η has the eigenvectors of $\widehat{B}$ as columns. Substituting the eigenvalue decomposition into Eq. 3.60 and factoring out the eigenvector matrices results in the final form of the equation

$$\begin{aligned} T_\xi \left[\mathrm{I} + \Delta t\, \delta_\xi^{(\mathrm{a})}\, \Lambda_\xi - \Delta t\, \delta_\xi^{(\mathrm{s})}\, \widehat{D}_\xi\right] T_\xi^{-1} T_\eta \\ \left[\mathrm{I} + \Delta t\, \delta_\eta^{(\mathrm{a})}\, \Lambda_\eta - \Delta t\, \delta_\eta^{(\mathrm{s})}\, \widehat{D}_\eta - \Delta t\, \delta_\eta^{(\mathrm{v})}\, (\lambda_\nu)\, \mathrm{I}\right] T_\eta^{-1} \Delta\widehat{Q}^{(n)} = -\Delta t\, R^{(n)} \end{aligned} \tag{3.63}$$

where λ_ν approximates the contribution from the viscous-flux Jacobian and is defined as

$$\lambda_\nu = \frac{\mu}{J\mathcal{R}e}(\eta_x^2 + \eta_y^2)\delta_\eta \left(\frac{J}{\rho}\right) \tag{3.64}$$

The convergence rate of the startup algorithm is accelerated with a spatially varying time step [143] given by

$$\Delta t = \frac{\Delta t_{\mathrm{ref}}}{1 + \sqrt{J}} \tag{3.65}$$

[9]The time step is spatially varying, see Eq. 3.65.

in order to account for the widely varying cell dimensions in the grid. A typical value of Δt_{ref} is 5.0.

A similar approximate-factorization approach is used for the Spalart–Allmaras turbulence model, Eq. 3.16. However, specific modifications of the implicit terms as well as a subiteration scheme are required in order to ensure that $\tilde{\nu}$ remains positive during startup, see Spalart and Allmaras [164] for details. In practice, the subiteration scheme is required only for cases with regions of large flow separation.

For multi-block grids, the blocks are solved independently during startup [119, 121]. At streamwise block interfaces, two columns of halo points are used to exchange boundary information between blocks at each iteration. The updates (including the intermediate update) are averaged such that the interface becomes transparent at steady-state. At normal block interfaces, the average boundary condition, Eq. 3.46, is used.

A reduction in the L_2 norm of the initial residual vector by three orders of magnitude is required in order to obtain a good initial guess for Newton's method. Note that the residual vector contains the contribution from all flow variables. For simple cases, especially single-element airfoils, a two order of magnitude reduction is usually sufficient. Generally, for cases when the startup tolerance is not sufficient, Newton's method fails to converge due to negative values of $\tilde{\nu}$, the turbulence model working variable. Grid sequencing can be used to accelerate the startup algorithm. It should be noted that the approximate-factorization algorithm has been used as an effective solver [143, 121, 67]. Performance comparisons between the approximate-factorization and Newton–Krylov flow solvers are provided in Section 4.2.

Final Algorithm

Overall, the inexact-Newton–Krylov flow solver can be summarized as follows:

- Initial residual is reduced by three orders of magnitude using the startup algorithm.
- Eq. 3.49 is solved using matrix-free GMRES(40) with no restarts. The matrix-vector products required at each GMRES iteration are formed with first-order finite-differences.
- The preconditioner is a block-fill incomplete LU decomposition (BFILU) of an approximate-flow-Jacobian matrix. Right preconditioning is used, and the preconditioner is stored in a modified-sparse-row (MSR) format [150, 151]. The level of fill for most cases is 2 [BFILU(2)], but difficult multi-element cases may require BFILU(4).
- Reverse Cuthill–McKee reordering of grid nodes based on an initial double-bandwidth ordering for single-element airfoils and a reverse ordering for multi-element airfoils.
- The preconditioner is frozen after the first outer iteration.

- The GMRES convergence tolerance is set to 0.5 for the first 10 outer iterations and 0.1 for any remaining outer iterations.

3.3 Discrete Gradients

Three methods for the computation of the objective function gradient have been implemented and are described below.

3.3.1 Finite-Difference Gradients

A centered-difference formula is used to compute the objective function gradient, $\mathcal{G}$, given by

$$\mathcal{G}_n = \frac{\mathcal{J}\left[X + he_n, Q(X + he_n)\right] - \mathcal{J}\left[X - he_n, Q(X - he_n)\right]}{2h} \qquad n = 1, \ldots, N_{\mathrm{D}} \tag{3.66}$$

where N_{D} denotes the number of design variables, e_n denotes the nth unit vector, and

$$h = \max\left(\epsilon \cdot |X_n|, 1 \times 10^{-8}\right) \tag{3.67}$$

A typical value of ϵ is 1×10^{-6}. An advantage of this approach is that the flow solver can be treated as a "black-box" subroutine. Unfortunately, as discussed in Section 1.2.2, for practical problems, the computational cost of repeated finite-difference gradient calculations is prohibitive since two flow solutions per design variable are required. Furthermore, the accuracy of finite-difference formulas is sensitive to the stepsize. A large stepsize introduces a significant truncation error, while a small stepsize is prone to subtractive cancellation errors.

3.3.2 Adjoints and Sensitivities

The gradient, $\mathcal{G}$, of the objective function $\mathcal{J}\left[X, Q(X)\right]$ is given by

$$\mathcal{G} = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} + \frac{\partial \mathcal{J}}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} \tag{3.68}$$

where we reduce the vector of design variables, X, to a scalar in order to clearly distinguish between partial and total derivatives. For problems with multiple design variables, it may be helpful to note that $\mathcal{G}$ and $\partial\mathcal{J}/\partial X$ are $[1 \times N_{\mathrm{D}}]$ row vectors, $\partial\mathcal{J}/\partial Q$ is a $[1 \times N_{\mathrm{F}}]$ row vector, and $\mathrm{d}Q/\mathrm{d}X$ is a $[N_{\mathrm{F}} \times N_{\mathrm{D}}]$ matrix, where N_{F} represents the number of flow variables.

Throughout this development, we assume that the implicit function $Q(X)$ is sufficiently smooth. Theoretically, this assumption is violated at flow discontinuities, for example shock waves, and leads to the requirement for additional boundary conditions. In practice, the smoothing of the discontinuities by the underlying spatial discretization of the flow equations provides a $Q(X)$ that is sufficiently smooth [71, 64, 63].

The difficulty in Eq. 3.68 is the evaluation of the term $\mathrm{d}Q/\mathrm{d}X$, referred to as the flow sensitivities. Evaluation of the partial derivatives, $\partial\mathcal{J}/\partial X$ and $\partial\mathcal{J}/\partial Q$, is relatively straightforward and is described at the end of this section. In order to compute the flow sensitivities, differentiate Eq. 3.47 with respect to the design variables

$$\frac{\mathrm{d}R}{\mathrm{d}X} = \frac{\partial R}{\partial X} + \frac{\partial R}{\partial \widehat{Q}}\frac{\mathrm{d}\widehat{Q}}{\mathrm{d}X} \tag{3.69}$$

and realize that $\frac{\mathrm{d}R}{\mathrm{d}X} = 0$, since for any design variable Eq. 3.47 is always satisfied. Furthermore, note that $\partial\widehat{Q}/\partial Q = J^{-1}\mathrm{I}$, where I is the identity matrix, and consequently Eq. 3.69 is simplified to the following large sparse system of linear equations

$$\frac{\partial R}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} = -\frac{\partial R}{\partial X} \tag{3.70}$$

We emphasize that the residual vector R contains the contribution from all grid nodes, including boundary conditions. The direct, or flow-sensitivity, method results from solving Eq. 3.70 for the flow sensitivities $\mathrm{d}Q/\mathrm{d}X$ and using these values in Eq. 3.68 to obtain the gradient.

In order to formulate the discrete-adjoint method, substitute Eq. 3.70 into Eq. 3.68 to obtain

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} - \frac{\partial \mathcal{J}}{\partial Q}\left(\frac{\partial R}{\partial Q}\right)^{-1}\frac{\partial R}{\partial X} \tag{3.71}$$

From the triple-product term in Eq. 3.71, define the following intermediate problem

$$\psi^{\mathrm{T}} = \frac{\partial \mathcal{J}}{\partial Q}\left(\frac{\partial R}{\partial Q}\right)^{-1} \tag{3.72}$$

where ψ is a $[N_{\mathrm{F}} \times 1]$ column vector. Post-multiplication of both sides by $(\partial R/\partial Q)$ and applying the transpose operator results in the following linear system of equations

$$\frac{\partial R}{\partial Q}^{\mathrm{T}}\psi = \frac{\partial \mathcal{J}}{\partial Q}^{\mathrm{T}} \tag{3.73}$$

This is known as the adjoint equation[10], and the vector ψ represents the adjoint variables. Substituting ψ into Eq. 3.71, the expression for the gradient becomes

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} - \psi^{\mathrm{T}}\frac{\partial R}{\partial X} \tag{3.74}$$

Note that Eq. 3.70 must be solved for each design variable, while Eq. 3.73 is independent of the design variables. If a sparse-direct solver is used to solve Eq. 3.70, then the LU factorization can be reused with different right-hand-side vectors. Unfortunately, direct solvers are computationally expensive for practical aerodynamic problems. A straightforward implementation of

[10]This terminology arises from the fact that the adjoint equation denotes one of the optimality conditions of a stationary Lagrangian for constrained minimization. From the viewpoint of linear algebra, the flow-sensitivity and adjoint equations can be referred to as the primal and dual equations, respectively.

iterative solvers leads to re-solving Eq. 3.70 for each design variable, which is computationally expensive as well. See [158] for modifications to iterative solvers that focus on linear systems with multiple right-hand sides; however, even with these solvers the computational overhead is still significant.

The GMRES method from the flow solver is adopted to solve both the adjoint and flow-sensitivity equations. We use right preconditioning with the preconditioner based on the approximate-flow-Jacobian matrix described in Subsection 3.2.2. Fast adjoint and flow-sensitivity solutions are obtained with BFILU(6) and GMRES(85), and these settings are further discussed in Chapter 4. For the flow-sensitivity equation, we use matrix-free GMRES with second-order accurate finite-differences, see Appendix B. In addition to memory savings, the matrix-free approach is easier to implement, since an accurate differentiation of cumbersome functions in the residual equations, such as the pressure switch, Eqs. 3.9–3.11, and the far-field circulation correction, is "automatically" provided. Due to the transpose on the left-hand-side of Eq. 3.73, the matrix-free approach is not possible for the adjoint equation. The flow Jacobian matrix is formed as described in Subsection 3.2.2 and is stored explicitly.

For the inverse design objective function the term $\partial\mathcal{J}/\partial Q$ is evaluated analytically, while for the remaining objective functions it is evaluated using centered differences. The remaining terms in Eqs. 3.68 and 3.74, namely the objective function sensitivity $\partial\mathcal{J}/\partial X$ and the residual sensitivity $\partial R/\partial X$, are also evaluated using centered differences. The use of centered differences for the evaluation of the partial derivative terms is not computationally expensive. For example, the centered-difference formula for the residual sensitivities is given by

$$\frac{\partial R}{\partial X}_n = \frac{R(X + he_n, Q) - R(X - he_n, Q)}{2h} \qquad n = 1, \ldots, N_{\mathrm{D}} \tag{3.75}$$

where h is defined by Eq. 3.67. Comparing Eqs. 3.75 and 3.66, it is important to realize that Eq. 3.75 involves two evaluations of the residual vector per design variable and *not* two flow solutions.

Note that the evaluation of residual sensitivities includes the evaluation of grid sensitivities[11], since the design variables do not explicitly appear in the residual equations except for the angle of attack design variable. The computational cost of the gradient calculation could be reduced by neglecting grid sensitivities for grid points sufficiently far from the airfoil. This approach, however, can introduce substantial error in the gradient calculation [131]. Since the computational cost of the grid-perturbation procedure (see Section 3.5) and of the residual evaluation is only a small fraction of the overall gradient calculation, we evaluate the residual sensitivities at every node in the domain. Recently, Hansen *et al.* [73] introduced a promising approach for the computation of solution sensitivities due to shape variations that avoids the

[11] $\frac{\partial R}{\partial X} = \frac{\partial R}{\partial N}\frac{\partial N}{\partial X}$, where $\partial N/\partial X$ represents the sensitivity of the grid node locations, N, to the design variables

computation of grid sensitivities. The approach is referred to as the fixed basis function finite element approach, and it has been applied to structural shape optimization problems.

3.4 Optimizer

The optimizer used to solve the aerodynamic shape optimization problem can have a significant impact on the efficiency of the optimization procedure, see [87] for model problem examples. Note that by using the penalty method to incorporate side constraints, the optimization problem is cast as an unconstrained problem. Consequently, the goal of the optimizer is to drive a suitable norm of the gradient vector to zero. The unconstrained problem is solved using the BFGS[12] quasi-Newton algorithm in conjunction with a backtracking line search. A brief description of the algorithm is provided below. For further details see [36, 133, 140].

The quasi-Newton algorithm, at each iteration p, determines a search-direction vector s using the relation

$$s_p = -\mathcal{H}_p \, \mathcal{G}_p \tag{3.76}$$

where $\mathcal{H}$ represents an approximation to the inverse of the Hessian matrix and $\mathcal{G}$ denotes the gradient vector of the objective function $\mathcal{J}$. Once the search-direction vector is known, the update of the design variables is given by

$$X_{p+1} = X_p + \beta_p \, s_p \tag{3.77}$$

where β represents a stepsize length determined by a line-search procedure. Note that if the stepsize equals unity and $\mathcal{H}_p$ is the inverse of the exact Hessian matrix, then Eqs. 3.76 and 3.77 represent Newton's method driving the gradient to zero. Since the calculation of the exact Hessian matrix is not practical, the BFGS secant update is used to generate a sequence of matrices $\mathcal{H}_p$ that approximate the inverse of the exact Hessian matrix with increasing accuracy. The BFGS update is given by

$$\mathcal{H}_{p+1} = \mathcal{H}_p - \frac{\mathcal{H}_p \upsilon_p \, (\mathcal{H}_p \upsilon_p)^{\mathrm{T}}}{\upsilon_p^{\mathrm{T}} \mathcal{H}_p \upsilon_p} + \frac{\delta_p \, \delta_p^{\mathrm{T}}}{\delta_p^{\mathrm{T}} \upsilon_p} + \upsilon_p^{\mathrm{T}} \mathcal{H}_p \upsilon_p \left(r \, r^{\mathrm{T}} \right) \tag{3.78}$$

where

$$r = \frac{\delta_p}{\delta_p^{\mathrm{T}} \upsilon_p} - \frac{\mathcal{H}_p \upsilon_p}{\upsilon_p^{\mathrm{T}} \mathcal{H}_p \upsilon_p} \tag{3.79}$$

and

$$\begin{aligned} \delta_p &= X_{p+1} - X_p \\ \upsilon_p &= \mathcal{G}_{p+1} - \mathcal{G}_p \end{aligned} \tag{3.80}$$

[12] Discovered independently by Broyden, Fletcher, Goldfarb, and Shanno.

The initial matrix $\mathcal{H}_0$ is set equal to the identity matrix and the initial stepsize is scaled by the L_2 norm of the gradient. This results in the scaling of the initial search vector to unity and the first descent direction is equivalent to the steepest-descent direction. After the first step but before the first update of $\mathcal{H}$, the value of $\mathcal{H}_0$ is set to

$$\mathcal{H}_0 = \frac{v_p^{\mathrm{T}} s_p}{v_p^{\mathrm{T}} v_p} \mathrm{I} \tag{3.81}$$

This equation approximates an eigenvalue of the inverse of the initial exact Hessian matrix and has been found to work well in practice [133].

It is well known that a stepsize of $\beta = 1$, i.e. a full-Newton step, results in an overall superlinear convergence of the algorithm. However, due to the inherent nonlinear nature of the problem, a full-Newton step may cause divergence of the algorithm during the initial iterations. This leads to a backtracking line-search procedure that systematically reduces the stepsize until an acceptable value is found. The acceptable stepsize value is required to satisfy the strong Wolfe conditions[13] [133, 36]

$$\begin{aligned} \mathcal{J}\left[X_p + \beta s_p, Q(X_p + \beta s_p)\right] &\leq \mathcal{J}\left[X_p, Q(X_p)\right] + c_1 \beta \mathcal{G}\left[X_p, Q(X_p)\right]^{\mathrm{T}} s_p \\ \left|\mathcal{G}\left[X_p + \beta s_p, Q(X_p + \beta s_p)\right]^{\mathrm{T}} s_p\right| &\leq c_2 \left|\mathcal{G}\left[X_p, Q(X_p)\right]^{\mathrm{T}} s_p\right| \end{aligned} \tag{3.82}$$

where $c_1 = 1 \times 10^{-4}$ and $c_2 = 0.9$. These conditions ensure that the updated matrix $\mathcal{H}$ remains positive definite and that a sufficient decrease in the objective function value is obtained for a sufficiently large stepsize.

When the stepsize does not satisfy Eqs. 3.82, the value of β is reduced by using the following line-search algorithm. At each step of the line search, the objective function value and the gradient are required in order to construct a local cubic interpolant for the one-dimensional function

$$\phi(\beta) = \mathcal{J}\left[X_p + \beta s_p, Q(X_p + \beta s_p)\right] \tag{3.83}$$

The minimum of the cubic interpolant is either at its endpoints or in the interior [133], where it is given by

$$\beta_{k+1} = \beta_k - (\beta_k - \beta_{k-1}) \left[\frac{\phi'(\beta_k) + r_2 - r_1}{\phi'(\beta_k) - \phi'(\beta_{k-1}) + 2r_2}\right] \tag{3.84}$$

with

$$r_1 = \phi'(\beta_{k-1}) + \phi'(\beta_k) - 3\frac{\phi(\beta_{k-1}) - \phi(\beta_k)}{\beta_{k-1} - \beta_k} \tag{3.85}$$

$$r_2 = \sqrt{r_1^2 - \phi'(\beta_{k-1})\phi'(\beta_k)} \tag{3.86}$$

The subscript k denotes a line-search iteration and the prime notation denotes differentiation with respect to β. It should be emphasized that this line-search algorithm is well suited for

[13] Also referred to as the Armijo-Goldstein-Wolfe conditions.

problems where the computation of the gradient is less expensive than an objective function evaluation.

The line-search algorithm also includes educated guesses for adjusting the stepsize when the line minimum has been bracketed and when the stepsize is too close to the endpoints [133, 36]. The line search is limited to a maximum of twenty steps in order to prevent unnecessary iterations when the objective function values become indistinguishable due to small changes in the stepsize. After each update of the design variables, but before a flow solution, the airfoil geometry and computational grid are checked for extreme side constraint violations. These include the detection of airfoil surface cross-over, as well as gap and overlap distances exceeding 15% of the set limits, see Section 3.5. If such violations are detected, the stepsize is reduced by a factor of two. Although the optimization problem is cast as an unconstrained problem, these checks are necessary in order to avoid flow solutions for defective geometries and grids, when the design variable update steps too far in a constrained direction prior to activating a penalty term. This is especially important during the initial iterations of the optimizer.

At the onset of the optimization, the initial value of the design variables is used to scale the optimization problem [66] given by

$$X = L^{-1} X_0 \tag{3.87}$$

where L is a diagonal matrix containing values of the initial design variables, X_0. The stopping criterion for the optimization is based on the scaled L_2 norm of the gradient. We require a reduction of four orders of magnitude in the L_2 norm of the gradient in order to ensure convergence to a local optimum.

3.5 Grid-Perturbation Strategy

As the shape and position of an airfoil evolve during the optimization process, the location of the grid nodes is adjusted from the baseline configuration to conform to the new configuration. This could be accomplished by a grid-generation tool, which could generate the required grid at each iteration of the optimization procedure and also during the computation of the gradient. Unfortunately, the generation of grids suitable for Navier–Stokes problems is computationally expensive; the process typically requires some user supervision and is difficult to linearize. Therefore, an algebraic grid-perturbation strategy is used instead. This grid-perturbation strategy is summarized below for the relocation of grid nodes in the normal direction. An analogous formulation holds for the streamwise direction. First, only airfoil shape changes are considered, and then the strategy is extended to also include changes in position.

Given a displacement of a B-spline control point in the vertical direction, the grid-perturbation strategy preserves the location of the outer boundary. The interior nodes along a normal grid

line are positioned as determined by

$$y_k^{\text{new}} = y_k^{\text{old}} + \Delta y\,(1 - S_k) \qquad k = 1, \ldots, k_{\max} - 1 \tag{3.88}$$

where Δy represents the airfoil shape change. S_k is the normalized arclength distance given by

$$S_1 = 0$$

$$S_k = \frac{1}{L_{\text{g}}} \sum_{i=2}^{k} L_i \qquad k = 2, \ldots, k_{\max} - 1 \tag{3.89}$$

where L_i is the length of a segment between nodes k and $k-1$. L_{g} is the grid-line length from the body to the outer boundary given by

$$L_{\text{g}} = \sum_{i=2}^{k_{\max}} L_i \tag{3.90}$$

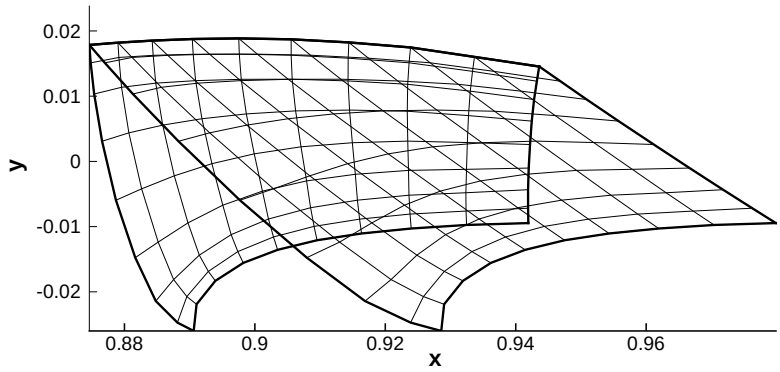
This grid-perturbation strategy is similar to one of the strategies outlined in [19, 145]. The strategy works well even for large shape changes in conjunction with both C- and H-topology grids.

When the optimization problem involves the horizontal and vertical translation design variables for multi-element configurations, the use of Eq. 3.88 can introduce significantly skewed grid cells. For example, consider a block in the slot region of the NLR 7301 airfoil, i.e. block 7 in Fig. 3.1(b). This block contains the leading-edge portion of the upper surface of the flap and the trailing edge of the main airfoil. Figure 3.8(a) shows the original and modified blocks using Eq. 3.88 for a flap displacement in the positive x-direction. The leading edge of the flap for the baseline configuration is located at $(0.89, -0.025)$. Note the poor orthogonality of the grid lines near the body as well as block interfaces. In order to improve the quality of the modified grid, we introduce a new grid-perturbation strategy given by

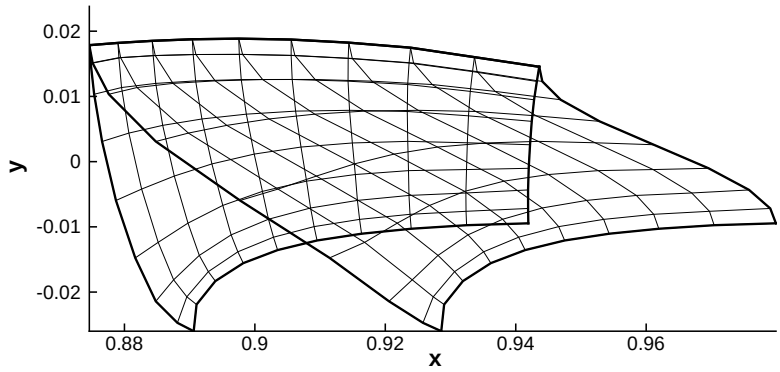
$$y_k^{\text{new}} = y_k^{\text{old}} + \frac{\Delta y}{2}\left[1 + \cos\left(\pi S_k\right)\right] \qquad k = 1, \ldots, k_{\max} - 1 \tag{3.91}$$

Fig. 3.8(b) shows the modified block resulting from the new grid-perturbation strategy. The orthogonality of grid lines near the body is significantly improved. Note that for a given horizontal and vertical flap displacement, blocks 6 to 9 and 11 to 13 in Fig. 3.1(b) require node adjustment. Changes in the airfoil shape do not affect blocks upstream of the leading edge and downstream of the trailing edge.

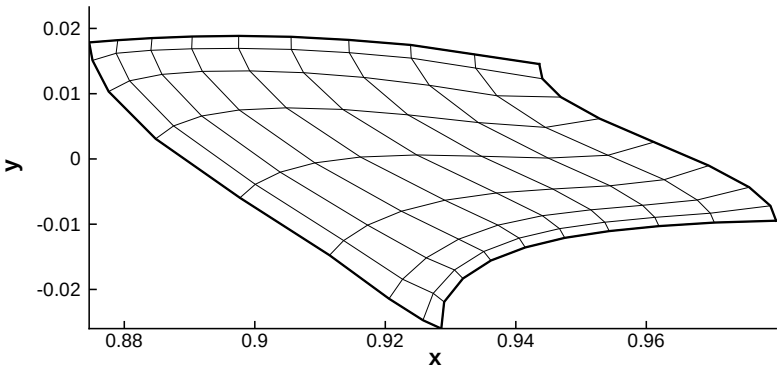
Elliptic smoothing [119, 167] can be applied to the modified grid, which is shown in Fig. 3.8(c). The interior grid lines become "more" orthogonal across block boundaries. However, elliptic smoothing is computationally expensive and the resulting accuracy benefits are marginal. Therefore, we do not apply elliptic smoothing to the perturbed grids.



(a) Grid perturbation using Eq. 3.88



(b) Grid perturbation using Eq. 3.91



(c) Elliptic smoothing for grid based on Eq. 3.91

Figure 3.8: Grid-perturbation strategy due to horizontal element translation

In the present work, we use Eq. 3.88 for C-topology grids, while Eq. 3.91 is used for H-topology grids. An example that examines the influence of the grid-perturbation strategy on the accuracy of the computed aerodynamic coefficients is presented in Subsection 4.5. For excessively large element displacements, the grid-perturbation strategy may generate overlapped grid cells. As discussed in Section 2.4, constraint equations are introduced in order to limit the extent of the displacements. Practical limits of the displacements are somewhat case dependent, but in general, the maximum changes for the gap and overlap distances should not exceed $\pm 0.5\%c$ and $\pm 1\%c$, respectively, when both translational design variables are active.

Chapter 4

VALIDATION

4.1 Overview

The solution of the aerodynamic shape optimization problem is controlled by the optimizer, as described in Section 3.4. The optimizer requires inputs from two components: 1) the flow solver described in Subsection 3.2.2, and 2) the gradient computation algorithm described in Section 3.3. The output from the optimizer, which is the vector of updated design variables, is processed by the grid-perturbation algorithm described in Section 3.5. These three components form a critical part of the Newton–Krylov algorithm, and consequently, we carefully validate each component in Sections 4.2-4.5. The validation of the gradient computation is divided into two sections. We establish the accuracy of the gradient in Section 4.3, and we present the efficiency of the gradient computation in Section 4.4.

All validation and design examples (presented in Chapter 5) are based on one of the following three airfoil geometries: 1) the single-element NACA 0012 airfoil [1] shown in Fig. 3.1(a), 2) the single-element RAE[1] 2822 airfoil [26], and 3) the two-element NLR[2] 7301 configuration [173] shown in Fig. 3.1(b). For the RAE 2822 airfoil, we use the standard coordinates instead of the

[1] Royal Aircraft Establishment
[2] National Aerospace Laboratory, The Netherlands

measured coordinates. The NLR 7301 airfoil and flap configuration is representative of take-off settings. The experimentally measured flap position is used, given by an overlap of $-5.3\%c$, a gap of $2.4\%c$, and a deflection angle of $19.75°$.

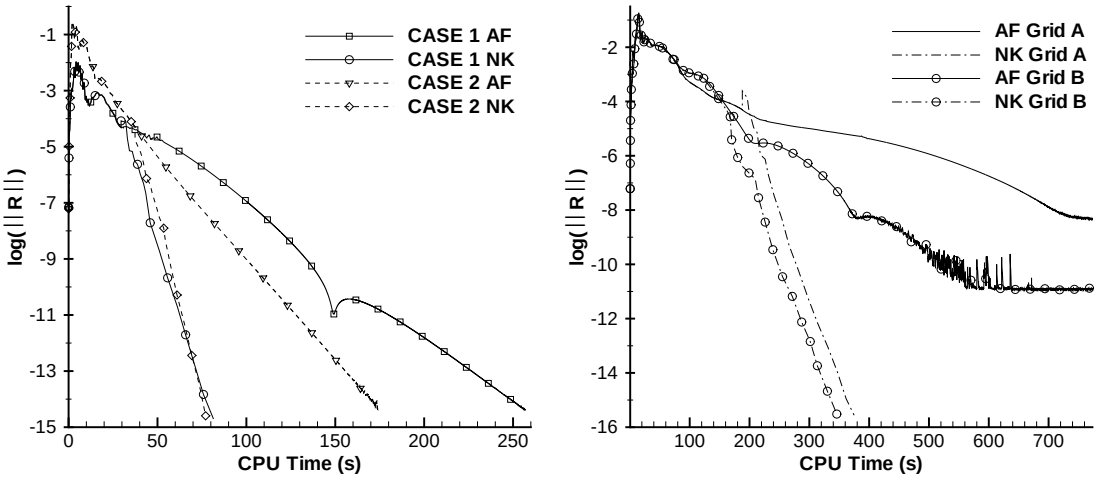
All single-element airfoil cases are computed on a 265×53 grid for subsonic flows, while a 257×57 grid is used for transonic flows. For these grids, the distance to the outer boundary is $24c$, the off-wall spacing is $2 \times 10^{-6}c$, the leading-edge clustering is $5 \times 10^{-4}c$, and the trailing-edge clustering is $2 \times 10^{-3}c$. For all multi-element airfoil cases, the grids consist of approximately 31,000 nodes. The off-wall spacing is $2 \times 10^{-6}c$, the distance to the outer boundary is $12c$, the spacing at the H-topology grid stagnation points is $2 \times 10^{-5}c$, and the trailing-edge clustering is $2 \times 10^{-3}c$. The selected grids are similar to those used for detailed accuracy studies presented in [185, 34] and provide sufficient numerical accuracy for the test cases considered here. The circulation correction is not used unless explicitly stated. The dissipation coefficients κ_2 and κ_4 in Eq. 3.4 are set to 1.0 and 0.01 for transonic flow, and 0.0 and 0.01 for subsonic flow, respectively. Hence, the pressure switch is not used for subsonic flow. The CPU times reported in the following sections are obtained on a 667 MHz Alpha 21264 processor (SPECfp 2000 rating of 562 peak) with 2 GBytes of memory.

4.2 Flow-Solver Performance

A fast solution of the flow equations is an important component of an effective design algorithm, since an evaluation of the objective function is required at each iteration of the optimizer. The accuracy of the flow solver, i.e. the spatial discretization presented in Subsection 3.2.1, has been established in previous studies [107, 182, 121, 67, 120, 34], and therefore, we focus only on the speed-up of the convergence rate. The performance of the flow solver is examined for the following analysis test problems:

1. NACA 0012 airfoil at $M_\infty = 0.3$, $\alpha = 6°$ and $\mathcal{R}e = 2.88 \times 10^6$.
2. RAE 2822 airfoil at $M_\infty = 0.729$, $\alpha = 2.31°$ and $\mathcal{R}e = 6.5 \times 10^6$.
3. NLR 7301 airfoil and flap at $M_\infty = 0.25$, $\alpha = 8°$ and $\mathcal{R}e = 2.51 \times 10^6$.

All cases are assumed to be fully turbulent. For case 1, the flow is subsonic and fully attached. The freestream conditions for Case 2 are given by Holst [78]. This case features transonic flow with a moderate-strength shock wave on the upper surface of the airfoil. For case 3, the flow is subsonic and the freestream conditions are based on [173], but here we consider a slightly larger Mach number. For all cases, the preconditioner constant ϕ, see Eq. 3.57, is set to 6.0. For cases 1 and 2, the level of fill in the preconditioner is set to 2, while for the multi-element case 3, the fill level is set to 4.



(a) Single-element airfoil cases 1 and 2 (b) Multi-element airfoil case 3

Figure 4.1: Performance of the Newton–Krylov flow solver

Fig. 4.1 shows that the Newton–Krylov flow solver (denoted as NK) is approximately two to three times faster than the approximate-factorization flow solver (denoted as AF). For many cases, this speed-up can be even larger. Initially, the convergence rate of both flow solvers is identical, since approximate-factorization is used as a startup procedure for the Newton–Krylov flow solver. Figure 4.1(b) compares the performance of the multi-block Newton–Krylov and approximate-factorization flow solvers for case 3 on two grids, labeled as Grid A and B. Grid A is the default grid generated by AMBER2d [119, 178] using the parameters specified at the beginning of this chapter[3]. The only difference between Grids A and B is that the aspect ratio is manually reduced by a factor of ten at block interfaces in the wake region of the main element. As shown in Fig. 4.1(b), the performance of the Newton–Krylov flow solver is fairly independent of minor variations in the grid quality.

The storage requirements of the Newton–Krylov flow solver are approximately ten times greater than the storage requirements of the approximate-factorization flow solver. This is consistent with results presented by Pueyo [141]. For design problems, however, the dominant storage requirements are due to the gradient computation (discussed in Section 4.4), and consequently, this memory can be reused for the analysis problem.

One of the main difficulties associated with Newton's method is the startup procedure, as discussed in Subsection 3.2.2. This startup procedure can be quite expensive, as shown in Fig. 4.1(a) for case 2 and Fig. 4.1(b) for case 3, where the startup time takes almost half of the flow solve time. The Newton–Krylov flow solver is particularly well suited for the design problem since once we obtain the solution for the initial airfoil shape, we "warm-start" the

[3]For readers familiar with AMBER2d, the usual clustering adjustments at the far-field boundaries are applied.

remaining flow solves. If the stepsizes during the line-search procedure are sufficiently small, the startup procedure is not necessary. The warm started flow solves typically converge in 2/3 of the original flow solve time.

4.3 Gradient Accuracy

The finite-difference gradient provides a benchmark that is used to establish the accuracy of the gradient computation using the flow-sensitivity and adjoint methods. The following representative test cases are considered: 1) the subsonic inverse design problem, 2) the lift-constrained transonic drag minimization problem, and 3) the subsonic lift-enhancement problem. For the computation of the finite-difference gradient, the flow solution is converged 14 orders of magnitude. The flow-sensitivity and adjoint equations are converged 8 orders of magnitude. The finite-difference stepsize constant, denoted by ϵ in Eq. 3.67, is set to 1×10^{-6}. Numerical experiments revealed that when ϵ is in the range of 1×10^{-4} to 1×10^{-8}, the variation in the computed value of the finite-difference gradient is very small. Similar results are also obtained by Wong [179].

For case 1, the freestream conditions are $M_\infty = 0.3$, $\alpha = 6^\circ$, and $\mathcal{R}e = 2.88 \times 10^6$. The NACA 0012 pressure distribution at the given freestream conditions is used as the target pressure distribution. The initial pressure distribution is obtained by replacing the NACA 0012 leading edge with the RAE 2822 leading edge, which modifies the location of B-spline control points numbered 6, 7, 9 and 10 in Fig. 2.1.

The gradient of the inverse design objective function (Eq. 3.3) with respect to the design variables associated with the four control points is calculated using centered differences, the adjoint method, and the flow-sensitivity method. The calculated gradient values are shown in Table 4.1, where the agreement between the finite-difference, adjoint, and matrix-free flow-sensitivity gradients is very good.

For case 2, the freestream conditions are from Holst [78], given by $M_\infty = 0.7$, $C_l = 0.4728$, and $\mathcal{R}e = 9 \times 10^6$. The initial airfoil geometry is the NACA 0012 airfoil. We compute the gradient of the objective function, Eq. 2.13, with respect to control points 9, 10, 11 and 12 (see Fig. 2.1), as well as the angle of attack (α). The target drag value, C_{D}^*, is set equal to 0.0112, which represents a 30% reduction from the initial drag value. No thickness constraints are imposed and the values of ω_{L} and ω_{D} in Eq. 2.13 are set to 2.0 and 1.0, respectively. Table 4.2 shows that there is some error in the adjoint gradients relative to the finite-difference gradients. This is due to the treatment of the pressure switch, Eqs. 3.9-3.11, as a constant with respect to the flow variables during the differentiation of the residual equations. Note that the pressure switch was not used for case 1. The agreement between the matrix-free flow-sensitivity gradients

Table 4.1: Gradient accuracy for case 1

Control Point #	Finite Difference	Adjoint (% Diff.)[b]	S-MF[a] (% Diff.)[b]
6	-127.82	0	-0.008
7	618.91	0	0
9	-2093.8	0	0
10	-526.58	-0.002	0

[a] matrix-free flow-sensitivity

[b] % Diff $= (\mathcal{G} - \mathcal{G}_{\text{FD}})/\mathcal{G}_{\text{FD}} \times 100$

Table 4.2: Gradient accuracy for case 2

Control Point #	Finite Difference	Adjoint (% Diff.)	S-MF[a] (% Diff.)
9	22.465	-7.0	0.004
10	29.592	4.1	-0.01
11	-16.428	-6.2	0.006
12	2.4066	-8.9	0.47
α	0.57486	0.52	0.06

[a] matrix-free flow-sensitivity

and the finite-difference gradients is not quite as good as for case 1, but remains excellent.

In order to demonstrate that the differences in Table 4.2 are due to the treatment of the pressure switch, we perform the following numerical experiment. We first converge the flow solver and store the pressure switch values, and then we reuse these values when we compute the two neighbouring states in the centered-difference gradient calculation. Hence, during the finite-difference gradient calculation the pressure switch is treated as a constant with respect to the flow variables. The results are summarized in Table 4.3, where the values obtained from the adjoint method agree well with the finite-difference and flow-sensitivity values. The minor differences that appear in Table 4.3 are due to the differentiation of the absolute value function, see Eq. 3.56.

The effect of the far-field circulation correction on the gradient accuracy is shown in Table 4.4. We perform a similar numerical experiment to that above; however, we freeze both the pressure switch and all absolute value functions such that we completely isolate the error

Table 4.3: Case 2 with frozen pressure switch

Control Point #	Finite Difference	Adjoint (% Diff.)	S-MF[a] (% Diff.)
9	20.895	-0.03	-0.005
10	30.803	-0.06	-0.02
11	-15.416	-0.03	0.04
12	2.1940	-0.01	-0.02
α	0.57798	-0.02	0.03

[a] matrix-free flow-sensitivity

Table 4.4: Case 2 with circulation correction

Control Point #	Finite Difference	Adjoint (% Diff.)	S-MF[a] (% Diff.)
9	17.954	-0.1	0
10	27.992	-0.1	0
11	-12.370	-0.7	-0.008
12	2.7021	1.7	0
α	0.54509	0.3	0

[a] matrix-free flow-sensitivity

contribution from the far-field circulation correction in the adjoint equation. The agreement between finite-differences and flow sensitivities is very good, and the error in the adjoint gradients is small. These results suggest that treating the vortex strength as a constant with respect to the flow variables has a relatively small effect on gradient accuracy.

For case 3, the freestream conditions are $M_\infty = 0.25$, $\alpha = 4^\circ$, and $\mathcal{R}e = 2.51 \times 10^6$. The initial airfoil is the two-element NLR 7301 configuration. We compute the gradient of the objective function, Eq. 2.13, with respect to control point 5 on the main airfoil (denoted as 5M), control point 4 on the flap (denoted as 4F), and the horizontal and vertical flap displacements (denoted as F_x and F_y, respectively), see Fig. 2.2. The target drag coefficient, C_{D}^*, is set equal to the initial drag coefficient (0.04298), while the target lift coefficient, C_{L}^*, is set equal to 2.2. The initial lift coefficient for this case is 2.149. The values of ω_{L} and ω_{D} in Eq. 2.13 are both set to 1.0 and there are no side constraints. The pressure switch and the circulation correction are not used. Table 4.5 shows that there is a good agreement between the finite-difference, adjoint,

Table 4.5: Gradient accuracy for case 3

Control Point #	Finite Difference ($\times 10^{-1}$)	Adjoint (% Diff.)	S-MF[a] (% Diff.)
5M	-0.12280	0.02	-0.07
4F	-0.85325	-0.19	-0.07
F_x	-0.25908	0.06	-0.02
F_y	-0.33627	-0.05	-0.05

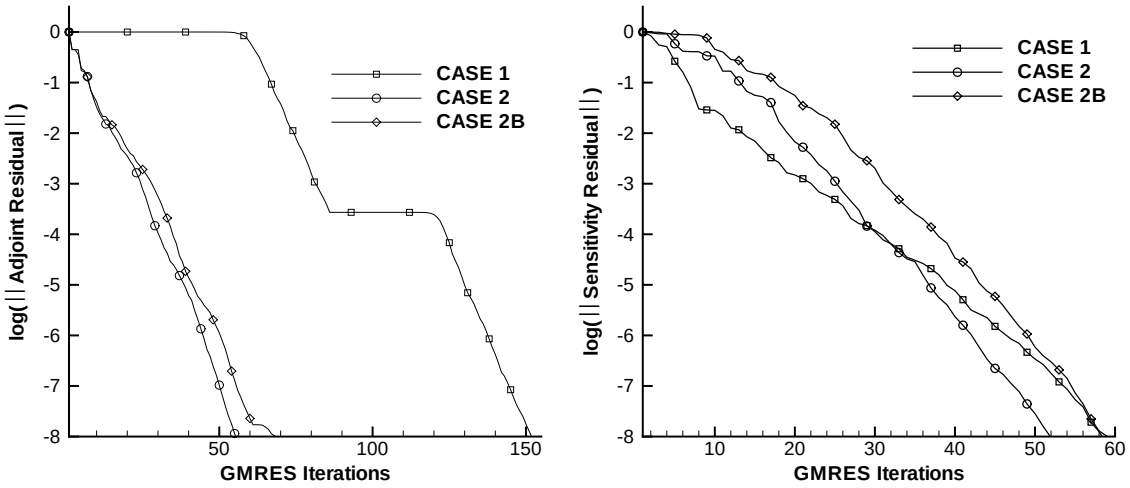
[a] matrix-free flow-sensitivity

and flow-sensitivity gradients.

4.4 Gradient Computation Efficiency

The critical parameters for efficient gradient computations using the preconditioned GMRES algorithm described in Subsections 3.2.2 and 3.3.2 are the BFILU level-of-fill, the number of GMRES search directions, and the value of ϕ in Eq. 3.57. The residual of the adjoint and flow-sensitivity equations should be reduced by three orders of magnitude in order to obtain gradients of sufficient accuracy [146, 130, 91]. For example, Nielsen [130] demonstrates that for such partially-converged adjoint solutions, the error in the gradient is below 0.01%. Hence, for a given BFILU level-of-fill, we select the number of search directions such that the adjoint and flow-sensitivity equations are sufficiently converged, yet GMRES restarts are avoided. The convergence of GMRES may stagnate after a restart [150], leading to an increase in the gradient computation time. However, the avoidance of restarts is balanced by the storage and work requirements of GMRES. The storage requirements increase linearly with the number of search directions, while work increases quadratically.

On the basis of our experience over a wide range of design problems, BFILU(6) in conjunction with GMRES(85) and $\phi = 3$ provides an efficient algorithm. For design problems based on multi-element airfoils, we use $\phi = 6$, which improves the robustness of GMRES, but the number of GMRES iterations increases by approximately 20%. Figure 4.2 shows the convergence histories of the adjoint and flow-sensitivity equations for the inverse design problem, labeled as case 1, and the lift-constrained drag minimization problem, labeled as case 2, both introduced in the previous subsection. The adjoint equation for the inverse design problem takes more iterations to converge than the drag minimization problem, and note that GMRES is forced to perform a restart on iteration 86 as shown in Fig. 4.2(a). The flow-sensitivity equation converges well for both problems as shown in Fig. 4.2(b), where the convergence history of only the first design



(a) Adjoint equation (b) Flow-sensitivity equation (matrix-free)

Figure 4.2: GMRES convergence for cases 1 and 2

variable is shown. The convergence histories of the remaining design variables are similar. The reason for the slower convergence of the inverse design adjoint equation is not fully understood. However, our experience suggests that the slower convergence rate is not due to the given flow conditions, and not every inverse design adjoint equation suffers from the slower convergence behaviour. Further insight is provided by the next test case.

Fig. 4.3 shows the convergence of the adjoint and flow-sensitivity equations for case 3, the lift-enhancement problem introduced in the previous subsection. The residuals of the flow-sensitivity equation are shown for each design variable. Fig. 4.3 highlights the influence of different right-hand-side vectors on the convergence of GMRES. Note that the initial guess for the adjoint and flow-sensitivity solution vectors is zero, and the pressure switch is not used. The left-hand side of the flow-sensitivity equation, Eq. 3.70, is the flow Jacobian matrix. This matrix remains the same for each design variable. For the adjoint equation, Eq. 3.73, the left-hand side is the transpose of the flow Jacobian matrix. The transpose operator does not change the eigenvalues of the flow Jacobian. Although the flow-sensitivity equation converges faster than the adjoint equation, the flow-sensitivity equation must be solved for each design variable, resulting in a much longer gradient computation time. Note that the asymptotic convergence rates are roughly equal, as might be expected.

The storage requirements for the adjoint and flow-sensitivity methods are summarized in Table 4.6 in terms of the double precision numbers required for the GMRES search directions, the preconditioner, and the flow Jacobian matrix. The integer arrays needed for the CSR and MSR matrix storage formats [151] are not included. The adjoint method requires the storage of the flow Jacobian matrix, a BFILU(6) preconditioner, and 85 GMRES search directions. This

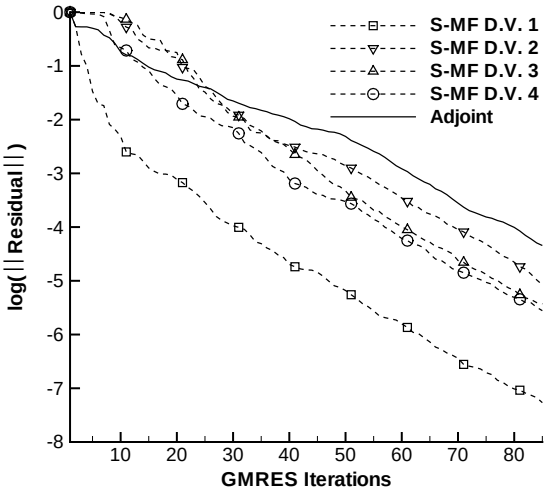


Figure 4.3: GMRES convergence for case 3 (S-MF denotes matrix-free flow-sensitivity, D.V. denotes design variable)

results in total memory requirements of $1.0 + 1.86 + 1.96 = 4.82$ expressed in terms of the non-zeros in the flow Jacobian matrix, see Table 4.6. For the flow-sensitivity equation, the storage of the flow Jacobian matrix is not required due to the matrix-free implementation of GMRES. We also provide the storage requirements for the BFILU(2) preconditioner that is used for the solution of the flow equations. Overall, the storage requirements of the present algorithm for optimization problems using the adjoint method allow the use of grids with roughly 100,000 nodes on a workstation with 2 GBytes of memory.

The values of the fill level in the BFILU decomposition and the number of search directions can be significantly reduced for the objective function defined by Eq. 2.13. For example, consider case 2B shown in Figs. 4.2(a) and 4.2(b), which is the lift-constrained drag minimization problem, evaluated with BFILU(4) and GMRES(60). There is only a minimal reduction in performance, while the storage requirements are reduced by 26%.

The convergence times for the adjoint solver are summarized in Table 4.7, where T_{PREC} refers to the time required to form the preconditioner, T_{GMRES} refers to the time required to reduce the adjoint residual by three orders of magnitude, and flow solve refers to the time required for a Newton–Krylov flow solve to converge ten orders of magnitude. In all design examples in this work (see Chapter 5), we converge the flow solution at least ten orders of magnitude, and therefore, the flow solve times in Table 4.7 provide a good reference. Overall, for case 1 the time to calculate the gradient is just over one-third of the flow solve time, while for cases 2 and 3 it is close to one-sixth of the flow solve time. These test cases provide a good indication of the variation in performance of the GMRES algorithm.

In order to further elucidate the results in Table 4.7, we use case 3 as an example and plot

Table 4.6: Storage requirements for case 2

Matrix	Number of non-zero elements[a]
BFILU(0)	0.56
BFILU(2)	0.99
BFILU(4)	1.43
BFILU(6)	1.86
GMRES(60)	1.39
GMRES(85)	1.96

[a] normalized by the flow Jacobian matrix, 3172675 non-zeros

Table 4.7: Adjoint solver convergence times[a]

Case	T_{PREC}	T_{GMRES}	Total	Flow Solve
1	4.2	22.9	27.1	75.2
2	4.2	7.0	11.2	65.0
2B	2.6	6.6	9.2	65.0
3	9.2	36.1	45.3	245.0

[a] CPU time measured in seconds

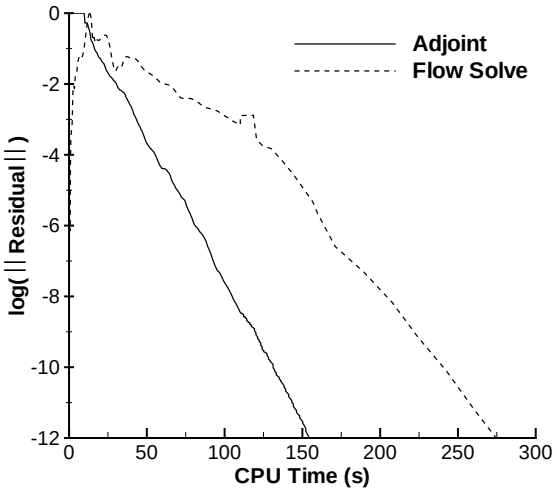


Figure 4.4: Comparison of adjoint and flow solve convergence times for case 3

the convergence history of the adjoint and flow equations, as shown in Fig. 4.4. The time for the formation of the preconditioning matrices is included in Fig. 4.4.[4] Comparison of Fig. 4.4 with similar plots presented by Elliott and Peraire [46], Nielsen [130], Reuther *et al.* [147], and Kim *et al.* [91] indicates that the preconditioned GMRES algorithm is among the fastest gradient computation algorithms currently available.

[4]In Fig. 4.4, the "flat step" in the convergence of the flow solver after a three order-of-magnitude decrease in the residual indicates the formation time of the preconditioner. For the adjoint equation, this time is indicated at the start of the convergence history.

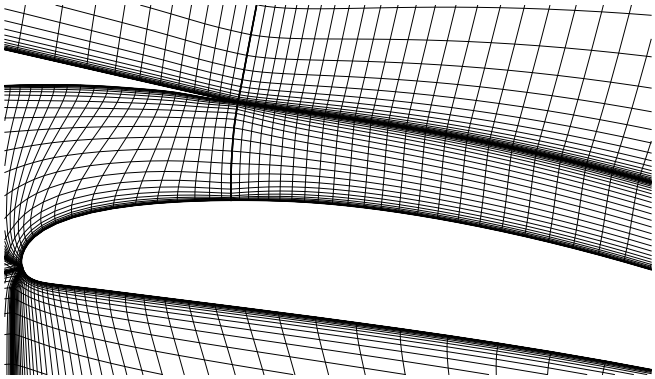
Table 4.8: Accuracy comparison of grid-perturbation strategies

Grid	C_L	C_D
Baseline	2.149	0.04298
Perturbation using Eq. 3.88	2.154	0.04310
Perturbation using Eq. 3.91	2.155	0.04303
New	2.155	0.04303

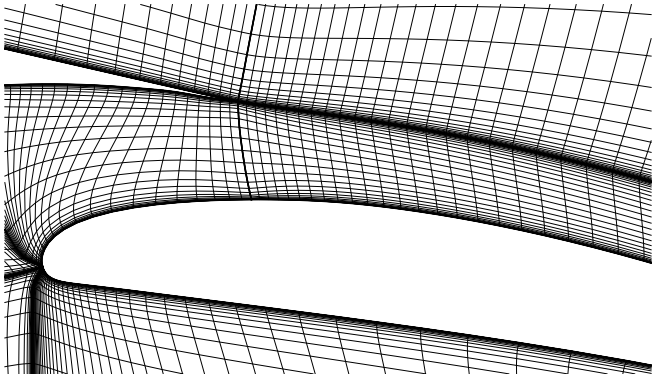
4.5 Comparison of Grid-Perturbation Strategies

The two grid-perturbation strategies presented in Section 3.5 are evaluated for design problems that involve the horizontal and vertical translation design variables. Both strategies work well for problems that involve only airfoil shape changes. The accuracy of the computed aerodynamic coefficients for the two strategies is examined for case 3, introduced in Section 4.3.

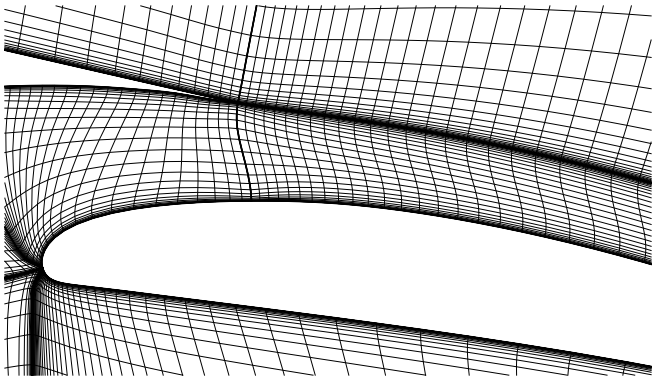
The baseline configuration is the NLR 7301 airfoil and flap, with a gap of $2.4\%c$ and an overlap of $-5.3\%c$. Figure 4.5(a) shows the grid details near the trailing edge of the main element and the leading edge of the flap. The flap is re-positioned to an overlap of $-4.8\%c$ and the same gap, and the baseline grid is adjusted to the new flap position using Eqs. 3.88 and 3.91. Figures 4.5(b) and 4.5(c) show the resulting grid modifications. We also generate a new grid for the re-positioned flap that is used as a benchmark to estimate the solution errors resulting from the grid-perturbation strategies. Table 4.8 shows the values of the aerodynamic coefficients for the baseline configuration, the two grid-perturbation strategies, and the new grid. The grid resulting from Eq. 3.91 provides improved estimates of aerodynamic performance.



(a) Original grid



(b) Grid-perturbation using Eq. 3.88



(c) Grid-perturbation using Eq. 3.91

Figure 4.5: Grid distortion resulting from a flap displacement

Chapter 5

Design Examples

... an optimal allocation of resources is not attained in any given society when it is still possible to make at least one individual better off in his/her own estimation while keeping others as well off as before in their own estimation ...

Vilfredo Pareto (1848-1923)

5.1 Overview

Having validated the individual components of the Newton–Krylov algorithm in Chapter 4, the performance of this algorithm is now demonstrated for a number of representative design problems. In Sections 5.2-5.4, single-element airfoil design problems are examined for each objective function presented in Section 2.3, namely inverse design, lift-constrained drag minimization, and maximization of lift-to-drag ratio. The optimization of high-lift configurations is presented in Section 5.5. In the final Sections, 5.6 and 5.7, we consider optimization problems that involve multiple objectives and operating points.

The grids introduced in Section 4.1 are used for all design examples, except for the multi-objective problem discussed in Section 5.6. The adjoint method is used for all gradient computations. The flow-sensitivity method is used only once, in Section 5.3, for a comparison study.

Table 5.1: Preconditioner parameter summary

	Gradient		Flow Solver	
Configuration	ϕ^a	Fill Level	ϕ^a	Fill Level
single-element	3.0	6	6.0	2
multi-element	6.0	6	6.0	4

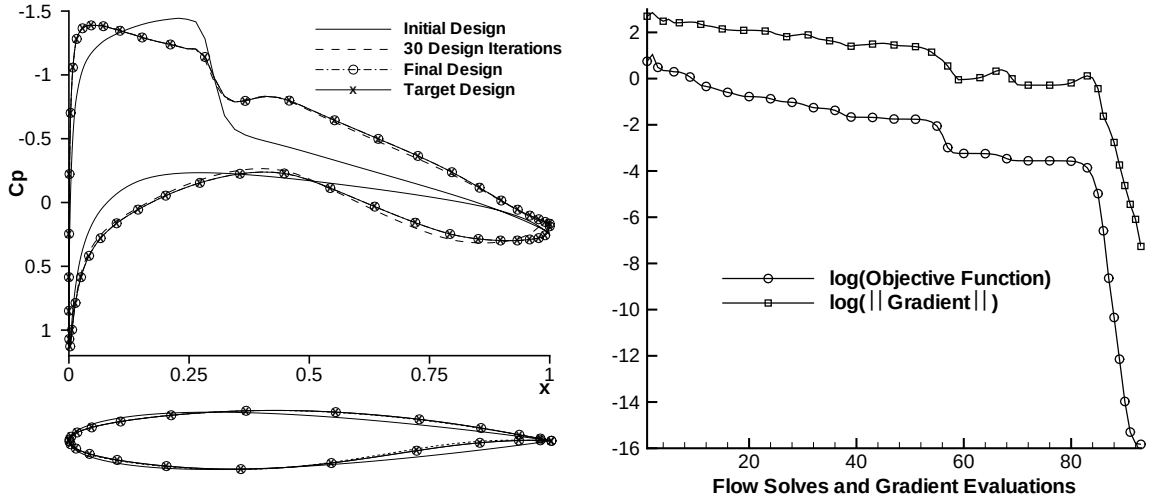
[a] see Eq. 3.57

The flow equations are converged at least ten orders of magnitude, while the adjoint equation is converged at least three orders of magnitude. The dissipation coefficients κ_2 and κ_4 in Eq. 3.4 are set to 1.0 and 0.01 for transonic flow, and 0.0 and 0.01 for subsonic flow, respectively, unless stated otherwise. The circulation correction is not used. A summary of parameters used for the GMRES preconditioning matrices is provided in Table 5.1.

5.2 Inverse Design

A transonic inverse design problem, see Eq. 3.3, is presented for the following initial conditions. The initial pressure distribution corresponds to the NACA 0012 airfoil and the target pressure distribution corresponds to a B-spline approximation of the RAE 2822 airfoil at $M_\infty = 0.7$, $\alpha = 3^\circ$ and $\mathcal{R}e = 9 \times 10^6$. In other words, the goal of the optimization is to obtain the RAE 2822 airfoil from the symmetric NACA 0012 airfoil. The initial shape is described with 15 B-spline control points of which 12 are used as design variables. The control point at the leading edge and the two control points at the trailing edge (points labeled 1, 8, and 15 in Fig. 2.1) are kept constant during the optimization.

Figure 5.1(a) shows the initial pressure distribution corresponding to the NACA 0012 airfoil, the target pressure distribution corresponding to the RAE 2822 airfoil, and the final design pressure distribution, as well as the corresponding airfoil shapes. Also shown in the figure are the pressure distribution and airfoil shape after 30 design iterations, which are very close to the target. The optimization convergence history is summarized in Fig. 5.1(b), where the abscissa includes the line-search iterations. Note that 90 flow solves and gradient evaluations are required to reduce the L_2 norm of the gradient by 10 orders of magnitude, although plotting accuracy is achieved within 60 design iterations. In terms of CPU time, plotting accuracy is achieved in approximately 1.5 hours. The errors in the accuracy of the adjoint gradient due to the treatment of the pressure switch, see Subsection 4.3, do not appear to significantly affect the convergence of this design example. The next design example examines the issue of gradient accuracy in greater detail.



(a) Pressure distributions and airfoil shapes (b) Optimization convergence history

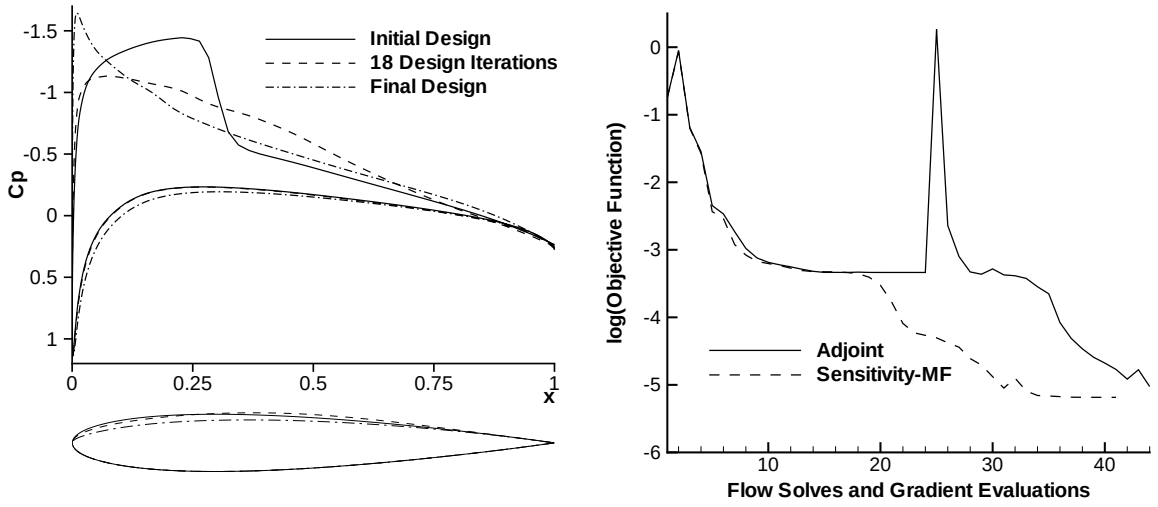
Figure 5.1: Inverse design problem (12 design variables)

5.3 Lift-Constrained Drag Minimization

A lift-constrained transonic drag minimization problem is solved using the adjoint and flow-sensitivity methods in order to examine the impact of gradient accuracy on the optimization problem. The problem setup is identical to case 2 presented in Subsection 4.3, and involves 5 design variables including the angle of attack.

Figure 5.2(a) indicates that after 18 design iterations the upper surface shock is eliminated. We only plot the adjoint results since those obtained with the flow-sensitivity method are very similar. The values of C_{L}, C_{D} and α at this stage are 0.4713, 0.01144 and 2.83, respectively, for the adjoint method and 0.4724, 0.01144, and 2.84, respectively, for the flow-sensitivity method. At this point, the adjoint method does not appear to be significantly affected by the errors in its gradient calculation. The convergence of the objective function stalls during the subsequent search direction for the adjoint method, as shown in Fig. 5.2(b). The BFGS algorithm is restarted using the steepest descent direction, and the adjoint method catches up to the matrix-free flow-sensitivity method after 40 design iterations. The values of C_{L}, C_{D} and α are 0.4727, 0.01123, and 3.33, respectively, for the adjoint method and 0.4726, 0.01123, and 3.28, respectively, for the matrix-free flow-sensitivity method. The pressure distribution and the airfoil shape are shown in Fig. 5.2(a). The L_2 norm of the flow-sensitivity and adjoint gradients is reduced by 4.5 and 3.5 orders of magnitude, respectively.

The stalling of the adjoint method appears to be case dependent and we find that for most design problems the two methods have very similar convergence histories. The corrupted search direction that causes the stall in the optimization procedure is mostly a result of the triggering



(a) Pressure distributions and airfoil shapes

(b) Comparison of adjoint and flow-sensitivity methods

Figure 5.2: Lift-constrained drag minimization (5 design variables)

of the pressure switch in subsonic regions of the flow, for example near the trailing edge of the airfoil. The fluctuating pressure-switch values confuse the line-search algorithm and provide poor updates of the Hessian matrix. The restarting of the optimization procedure with the steepest-descent direction works well in practice, and we do not use the pressure switch for design problems with subsonic flow.

5.4 Maximization of Lift-to-Drag Ratio

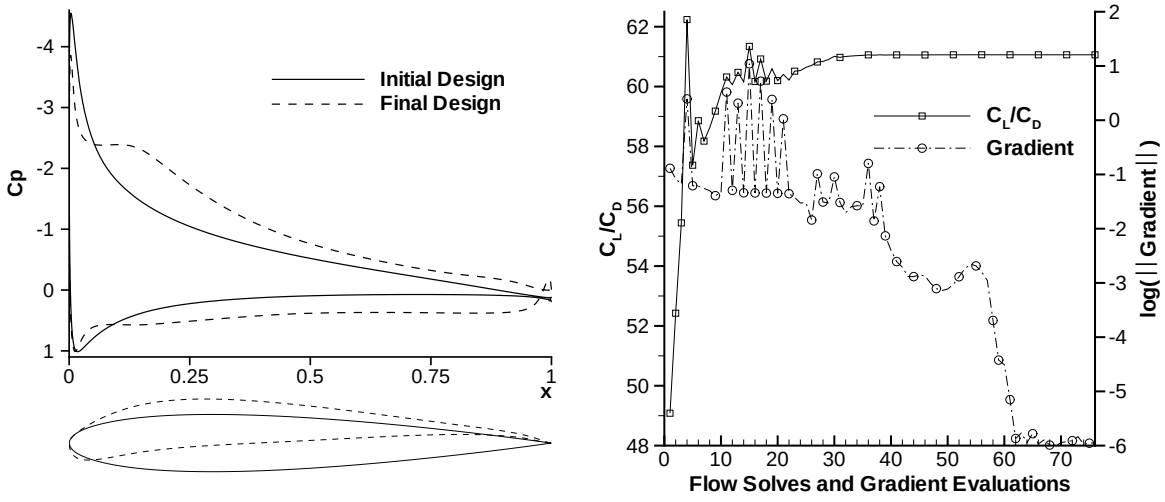
For the maximization of the lift-to-drag ratio, Eq. 2.14, the following subsonic design problem is considered. The initial airfoil is the NACA 0012 and the freestream conditions are $M_\infty = 0.25$, $\alpha = 9^\circ$, and $\mathcal{R}e = 2.88 \times 10^6$. The airfoil shape is described with 15 B-spline control points of which 10 are used as design variables. The angle of attack is fixed during the optimization. In addition, after a few trial-and-error optimization runs, five thickness constraints are specified. These are summarized in Table 5.2, where minimum thickness denotes the minimum allowable thickness below which the penalty function, Eq. 3.2, is activated. The value of ω_T in Eq. 3.2 is set to 0.05. This ensures that the objective function and the thickness penalty are of the same order of magnitude. The residual of the adjoint equation is reduced by eight orders of magnitude. This case is similar to one of the cases considered by Liebeck [101]; however, it should be emphasized that in the present work the flow is assumed to be fully turbulent.

Figure 5.3(a) shows the initial pressure distribution corresponding to the NACA 0012 airfoil, the final design pressure distribution, and the corresponding airfoil shapes. The lift coefficient

Table 5.2: Thickness constraints for the max C_L/C_D problem

Constraint	Location[a]	Minimum Thickness[a]	Final Thickness[a]
1	6.0	8.0	8.14
2	25.0	10.0	9.85
3	60.0	5.5	5.48
4	80.0	2.0	2.13
5	90.0	1.0	0.998

[a] % of airfoil chord



(a) Pressure distributions and airfoil shapes

(b) Optimization convergence history

Figure 5.3: Maximization of C_L/C_D (10 design variables)

is increased from 0.967 to 1.49, while the drag coefficient increases from 0.01969 to 0.02443. Correspondingly, the lift-to-drag ratio increases from 49.1 to 61.1, and the L_2 norm of the gradient is reduced by five orders of magnitude in 62 design iterations as shown in Fig. 5.3(b). The oscillations in the L_2 norm of the gradient are due to the activation of thickness constraints, especially during the line searches. Furthermore, Fig. 5.3(b) shows that the lift-to-drag ratio is converged to engineering accuracy within 35 design iterations and this is also true for the objective function (not shown), which includes the thickness penalty term. The coefficient of skin friction indicates that the flow is separated on the upper surface of the airfoil over the last 1.5%c. The final airfoil thicknesses are shown in Table 5.2. Three of the five thickness constraints are active, but the thickness violations are relatively small.

5.5 Optimization of High-Lift Configurations

The performance of the multi-block Newton–Krylov algorithm is examined for the optimization of high-lift configurations. The first design example, in Subsection 5.5.1, considers the problem of finding an optimal flap position for the two-element NLR 7301 configuration. The issue of a local versus a global minimum is also addressed. In the second design example, Subsection 5.5.2, we perform a constrained optimization for the shape and position of the flap using the NLR 7301 configuration, such that the cruise airfoil shape is not modified.

5.5.1 Flap Position Optimization

The goal is to determine the optimal gap and overlap distances for the NLR 7301 configuration, such that the modified configuration achieves a higher lift coefficient while maintaining the same (or lower) drag coefficient as the original configuration. The freestream conditions are $M_\infty = 0.25$, $\alpha = 4^\circ$, and $\mathcal{R}e = 2.51 \times 10^6$. The dissipation coefficients κ_2 and κ_4 in Eq. 3.4 are set to 0.0 and 0.015, respectively. The resulting initial values of C_L and C_D are 2.145 and 0.04720. The objective function is given by Eq. 2.13, where we set $C_\mathrm{L}^* = 2.180$ and C_D^* equal to the initial drag coefficient. The weights ω_L and ω_D are set to 1.0. The design variables are the horizontal and vertical displacements of the trailing edge of the flap, as indicated in Fig. 2.2. The gap and overlap limits are set to $\pm 0.5\%c$ and $\pm 1.0\%c$, respectively, based on the initial configuration. The weight associated with the gap and overlap constraints is set to 0.05.

Table 5.3 and Fig. 5.4 summarize the results. Within a few flow and gradient evaluations, the flap reaches the maximum allowable overlap distance of approximately $-4.3\%c$, at which point the overlap penalty function becomes active. The optimization converges to the design #1 configuration, shown in Fig. 5.4. A new grid is generated for this configuration, and the corresponding values of C_L and C_D are given in Table 5.3. The optimization is restarted from

Table 5.3: Gap-overlap optimization summary

Design	C_L	C_D	G[a]	O[b]
NLR 7301	2.145	0.04720	2.40	-5.31
#1	2.165	0.04687	1.99	-4.28
#2	2.173	0.04677	1.95	-3.30
Final	2.175	0.04675	2.02	-2.68
Target	2.180	$\leq$ 0.0472		

[a] Gap (%c)

[b] Overlap (%c)

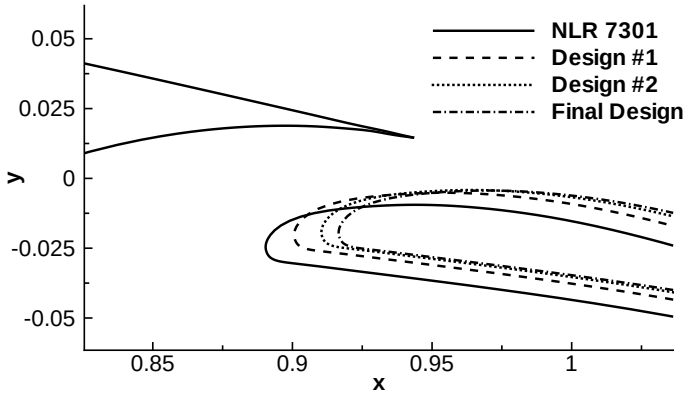


Figure 5.4: Flap position summary

the new grid with the same objective function. This procedure is continued until convergence to the final design is obtained (see Fig. 5.4), where the gap and overlap constraints are no longer active. Note that the drag objective is satisfied for all the designs. Consequently, the optimization is purely attempting to maximize the lift coefficient. Overall, a 1.4% increase in the value of the lift coefficient is obtained. This is achieved by an increased loading on the main element as well as the flap, as shown in Fig. 5.5.

Example convergence histories for the intermediate design #2 and final configurations are shown in Fig. 5.6. The oscillations in the L_2 norm of the gradient for design #2 are due to the presence of the gap and overlap constraints. The norm of the gradient is reduced by several orders of magnitude, which indicates that the optimization converged to a local minimum.

Given that the target value of the lift coefficient is not achieved at the final design configuration (see Table 5.3), it is somewhat surprising that further design improvements cannot be realized by further extending the effective chord of the configuration. The convergence of the gradient in Fig. 5.6 indicates that a local optimum has been found, but a global optimum

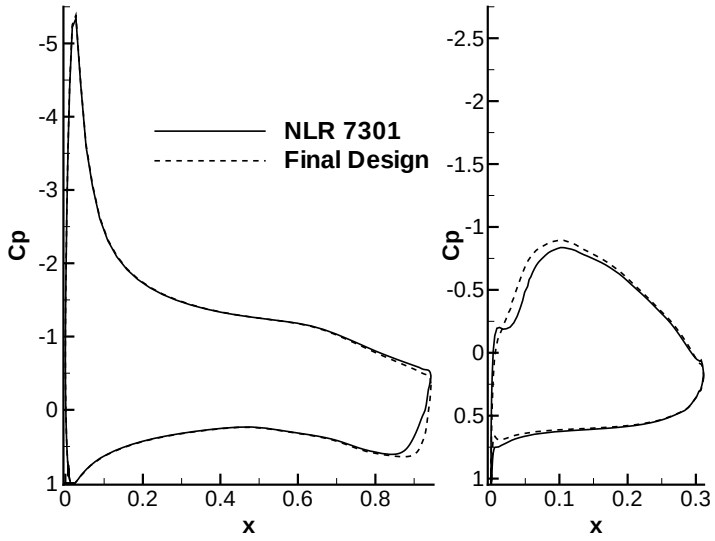


Figure 5.5: C_p distribution for main element and flap

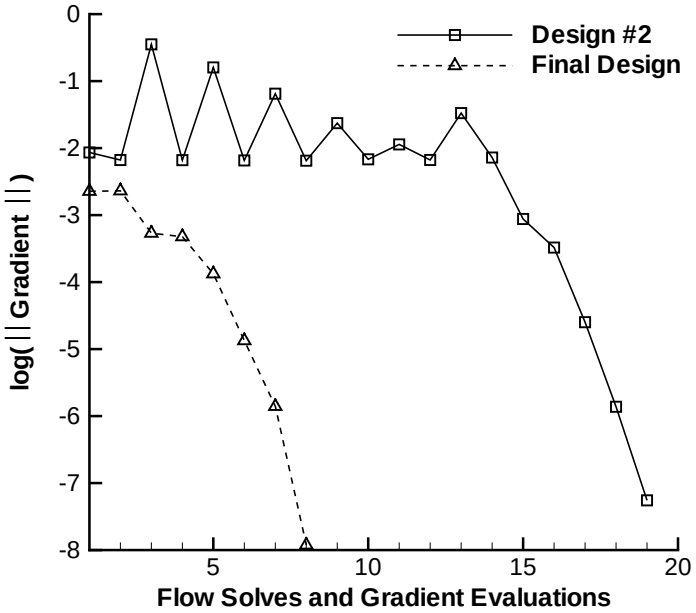


Figure 5.6: Convergence histories for gap-overlap optimization

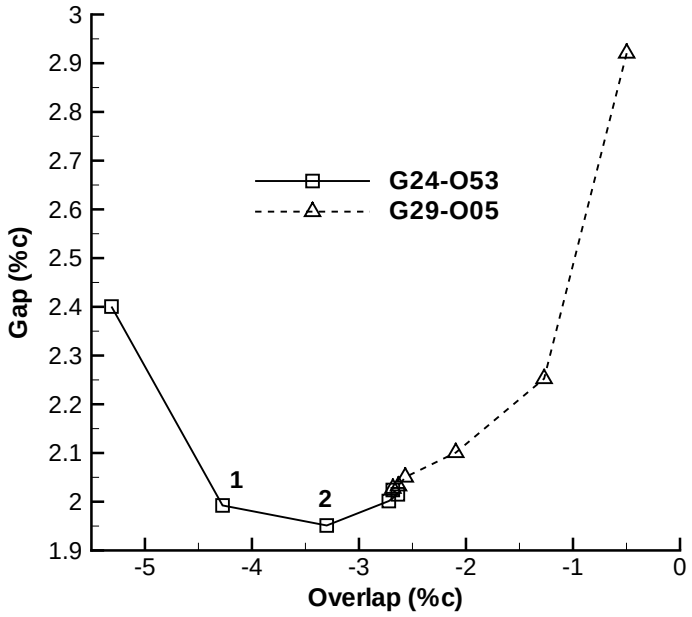


Figure 5.7: Convergence to optimal gap-overlap distances from two distinct initial conditions

is not guaranteed. In order to verify the uniqueness of the optimal solution, the optimization is restarted from a different initial condition. The flap is re-positioned to a gap of 2.9%c and an overlap of $-0.5\%c$, i.e. the leading edge of the flap is almost aligned with the trailing edge of the main element. Figure 5.7 shows that the optimization converges to the same optimal solution. The data labeled "G24-O53" show the convergence to the optimal solution from the original configuration, with designs #1 and #2 indicated, while the data labeled "G29-O05" show the convergence to the same optimal solution from the new initial conditions.

5.5.2 Flap Shape and Position Optimization

The flap position optimization example presented in Subsection 5.5.1 is expanded to include flap shape changes. As shown in Fig. 5.8, a B-spline curve is fitted over a portion of the upper surface of the flap, such that the cruise (flap-stowed) configuration is not affected by the shape modifications. The design variables consist of the four shaded control points, as well as the horizontal and vertical flap displacements.

The objective function and all the optimization parameters remain unchanged from Subsection 5.5.1 except for the target lift coefficient, which is increased to 2.2. Table 5.4 and Fig. 5.9 summarize the results. The optimization is started from the optimal gap and overlap values obtained previously, which is denoted as the initial configuration in Table 5.4 and Fig. 5.9. For the final design, the gap distance remains approximately constant, but the thickness of the flap increases considerably near the leading edge, as shown in Fig. 5.9. The gradient convergence

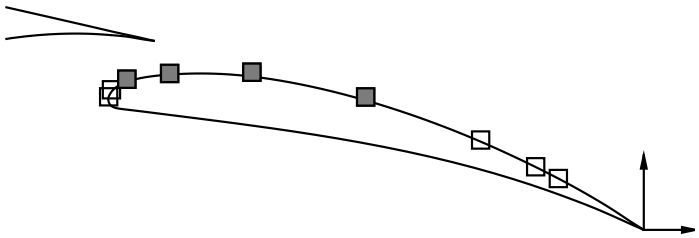


Figure 5.8: Flap shape and position design variables

Table 5.4: Flap optimization summary

Design	C_{L}	C_{D}	G[a]	O[b]
NLR 7301	2.145	0.04720	2.40	-5.31
Initial	2.175	0.04675	2.02	-2.68
Final	2.2	0.04723	2.06	-2.40
Target	2.2	$\leq$ 0.0472		

[a] Gap (%c)

[b] Overlap (%c)

history is shown in Fig. 5.10, where the optimization converges within 25 flow and gradient evaluations. A new grid is generated for the final configuration, and the corresponding values of the lift and drag coefficients are provided in Table 5.4. Overall, the final design achieves a 2.5% increase in the lift coefficient value, while almost no drag penalty is incurred when compared with the original NLR 7301 configuration. Comparison of Figs. 5.5 and 5.11 shows that the loading on both elements is significantly increased as a result of the shape design variables.

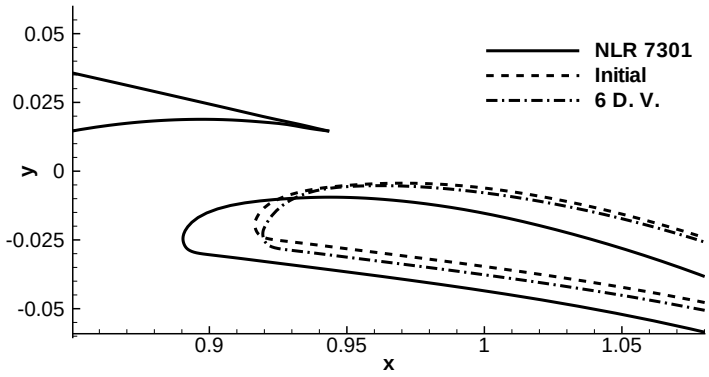


Figure 5.9: Flap shape and position summary (D.V. denotes design variable)

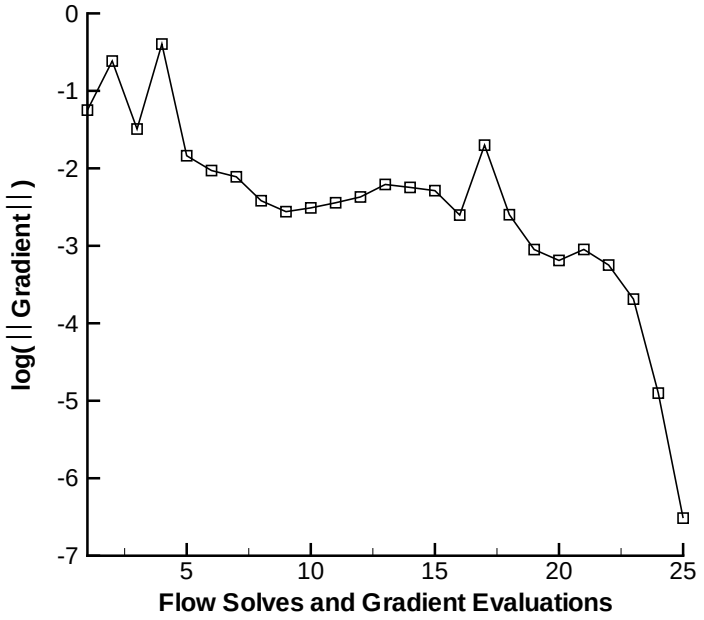


Figure 5.10: Gradient convergence history

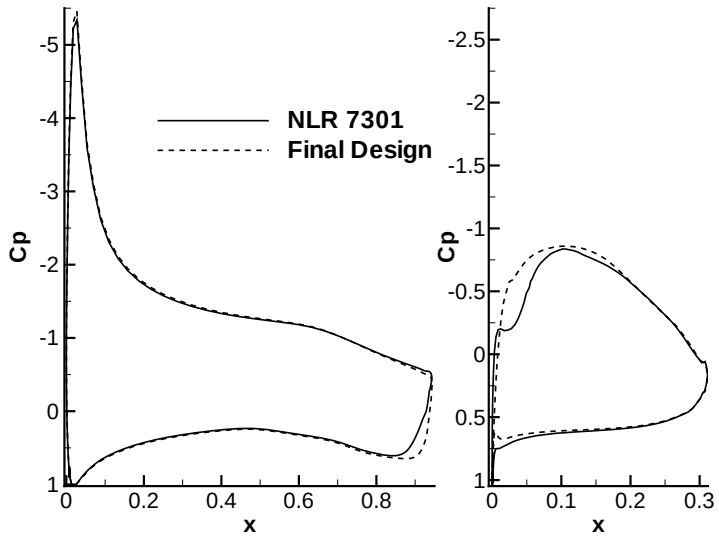


Figure 5.11: C_p distribution for main element and flap

5.6 Multi-Objective Design

The design examples considered so far are based on a single objective function in conjunction with a few constraints. Since practical design problems usually involve multiple objectives, we investigate the performance of the Newton–Krylov algorithm for the computation of a set of non-inferior solutions that is based on two competing objective functions. In particular, we consider the design of an airfoil shape to achieve specified lift and drag coefficients using the following two objectives:

$$\mathcal{J}_{\mathrm{L}} = \left(1 - \frac{C_{\mathrm{L}}}{C_{\mathrm{L}}^*}\right)^2 \tag{5.1}$$

$$\mathcal{J}_{\mathrm{D}} = \left(1 - \frac{C_{\mathrm{D}}}{C_{\mathrm{D}}^*}\right)^2 \tag{5.2}$$

The target lift and drag coefficients are chosen such that for a given set of design variables and constraints, the two objectives cannot be satisfied simultaneously. The objectives are competing, since a reduction in drag will typically result in a reduction in lift due to the decrease in the thickness and camber of the airfoil. Consequently, this problem does not have a unique solution. Instead, we seek to find a set of non-inferior solutions, where an improvement in one of the objectives results in a degradation of the other. Such solution sets are referred to as Pareto fronts. This concept was first introduced in the field of sociology and the original definition is briefly stated in the opening quote of this chapter.

There are numerous techniques to solve multi-objective problems [25, 108]. The technique used here is the weighted-sum method. The vector of the objective functions is converted to a scalar by assigning a weight to each objective and then forming a sum of the objectives. The resulting objective function is similar to Eq. 3.1 and is given by

$$\mathcal{J} = \omega_{\mathrm{L}}\,\mathcal{J}_{\mathrm{L}} + (1 - \omega_{\mathrm{L}})\,\mathcal{J}_{\mathrm{D}} + \omega_{\mathrm{T}} \sum_{j=1}^{N_{\mathrm{c}}} C_j \tag{5.3}$$

where $\omega_{\mathrm{T}} = 1.0$. Although the use of this technique is problematic for poorly scaled problems and non-convex Pareto boundaries, the implementation of this technique is straightforward and it can provide considerable insight into multi-objective problems.

The results are presented for the following transonic design example. The freestream conditions are $M_\infty = 0.7$ and $\mathcal{R}e = 9 \times 10^6$. The dissipation coefficients κ_2 and κ_4 in Eq. 3.4 are set to 0.0 and 0.01, respectively. The computational grid consists of 201×45 nodes. Additional grid parameters are given in Section 4.1. We specify a target lift coefficient of 0.55 and a target drag coefficient of 0.0095. The initial airfoil is the NACA 0012 airfoil. The airfoil shape is described with 15 B-spline control points and we use 10 control points as design variables, as

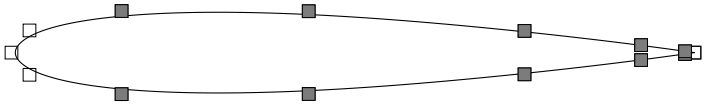


Figure 5.12: B-spline control points and design variables (shaded control points)

Table 5.5: Thickness constraints

T. C. no.	1	2	3
Location (%c)	25.0	92.0	99.0
Thickness (%c)	11.8	0.9	0.2

shown in Fig. 5.12. The angle of attack is also a design variable, resulting in a total of 11 design variables. In addition, we specify three thickness constraints as summarized in Table 5.5.

The computed Pareto front is shown in Fig. 5.13, where the trade-off between the competing objectives is clearly captured. The label T.C. denotes the thickness-constraint penalty value. Also shown are two sample airfoil shapes obtained at the end-points of the front. Aerodynamic coefficients for a few selected solutions are provided in Table 5.6.

Example convergence histories for different values of ω_{L} are shown in Fig. 5.14(a). The first optimal solution is obtained for $\omega_{\mathrm{L}} = 0.9$, which requires approximately 130 flow and gradient evaluations. As discussed in Section 5.4, the oscillations in the L_2 norm of the gradient are mainly due to the activation of thickness constraints during the line searches. The solutions for the remaining values of ω_{L} are computed in decreasing order by warm-starting the optimization from the previous solution. The warm-started solutions are typically obtained in 65 to 90 flow and gradient evaluations, as indicated in Fig. 5.14(a). An example convergence history of the objective function is shown in Fig. 5.14(b) for $\omega_{\mathrm{L}} = 0.9$. The values of the objective function are plotted at the end of each search direction, i.e. when the line-search exit criteria are satisfied. Note that within 25 flow and gradient evaluations, the objective function is converged to engineering accuracy. Hence, the computation of Pareto fronts using such partially converged

Table 5.6: Aerodynamic coefficients for selected Pareto optimal solutions

ω_{L}	C_{L}	C_{D}	α
0.9	0.5440	0.01204	0.264
0.7	0.5291	0.01187	0.222
0.5	0.5074	0.01169	0.167
0.3	0.4693	0.01145	0.0906
0.1	0.3681	0.01099	-0.0557

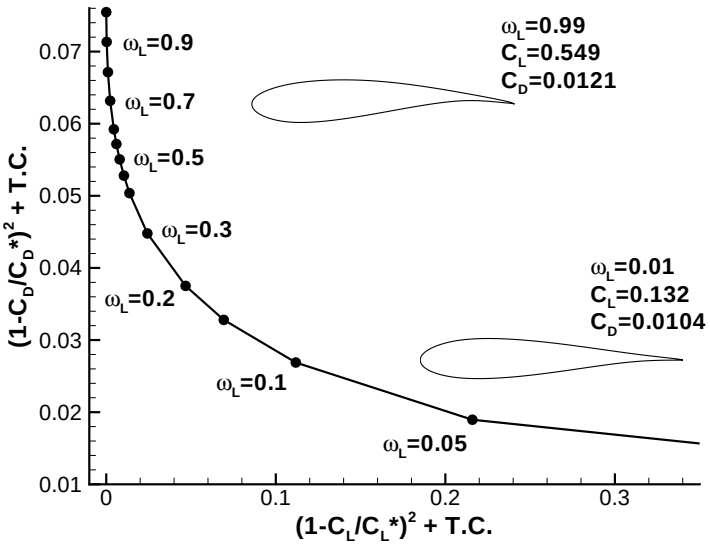
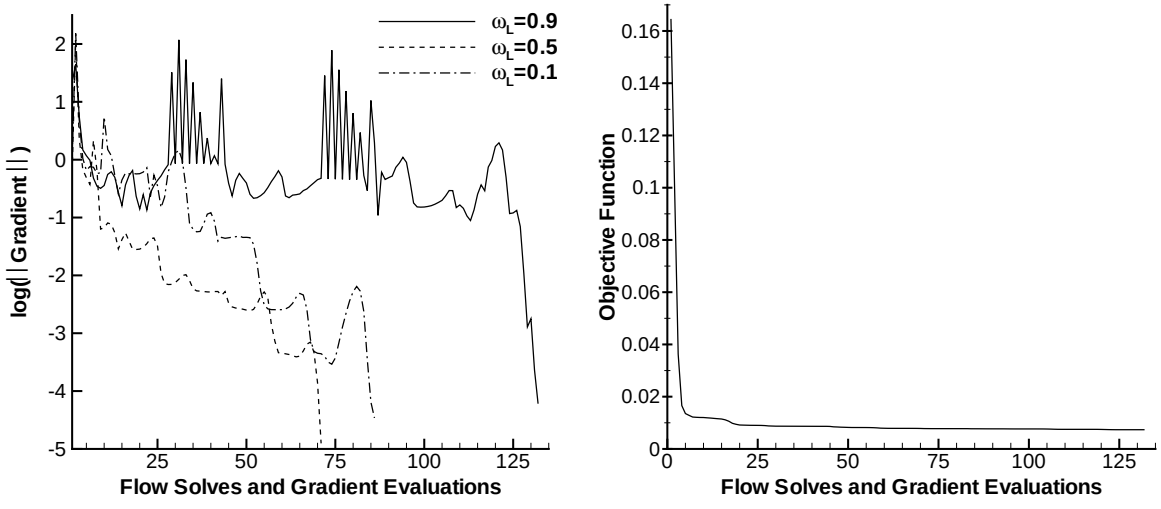


Figure 5.13: Pareto front (11 design variables)



(a) Gradient history

(b) Objective function history for $\omega_L = 0.9$

Figure 5.14: Convergence histories for selected Pareto front solutions

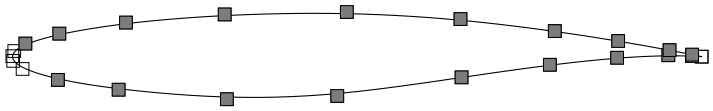


Figure 5.15: Control points and design variables (shaded) for the RAE 2822 airfoil

solutions can significantly reduce CPU time requirements.

Since the Pareto front shown in Fig. 5.13 has been obtained using a gradient-based method, the convergence to a true, or global, Pareto front is not guaranteed. In [128], we confirm that the Pareto front obtained by the Newton–Krylov algorithm is a global front. This is accomplished by solving the present multi-objective problem using a genetic algorithm developed by Holst and Pulliam [79], which is specifically tailored to capture global fronts.

5.7 Multi-Point Design

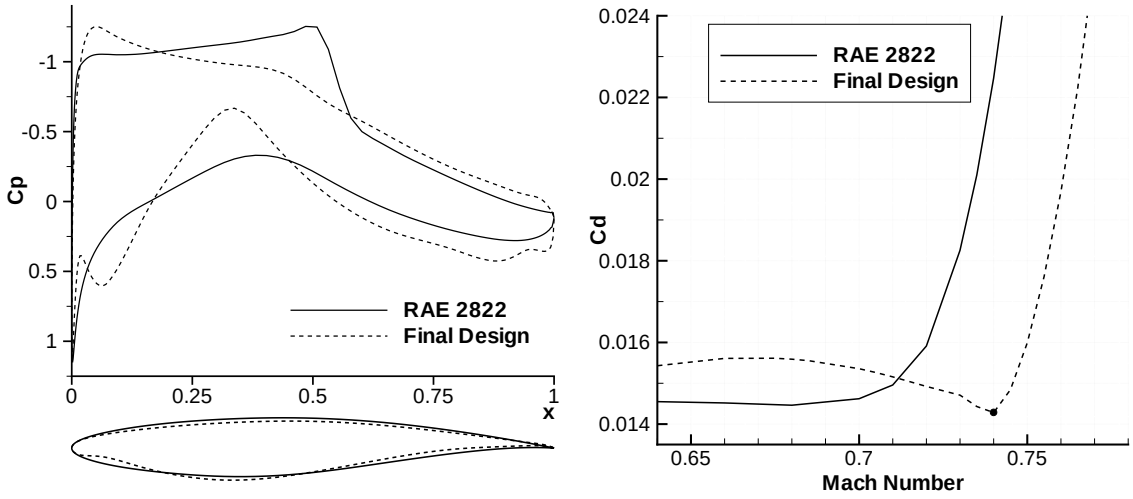
In order to investigate the performance of the Newton–Krylov algorithm for multi-point optimization problems, the design of a low-drag airfoil for transonic flight conditions at a specified lift coefficient is considered. This example is based on one of the cases studied by Drela [38], where the AGARD Case 13a [26] is used for initial conditions. The objective function is given by Eq. 2.13, with the target drag coefficient, C_{D}^{*}, set to 0.013, and the target lift coefficient, C_{L}^{*}, set to 0.733. The Reynolds number is 2.7×10^{6} and the initial angle of attack is 2°. The artificial dissipation coefficients κ_2 and κ_4 in Eq. 3.4 are set to 0.0 and 0.02, respectively, in order to avoid stalling during the line-search iterations (discussed previously in Section 5.3). The initial airfoil is the RAE 2822 airfoil. The airfoil shape is described with 25 control points, and we use 19 control points as design variables, as well as the angle of attack. The B-spline control points together with the active design variables are shown in Fig. 5.15. The values of ω_{L} and ω_{D} are set to 1.0 and 0.1, respectively. In addition, three airfoil thickness constraints are specified, as summarized in Table 5.7. The constraint at $35\%c$ represents the initial airfoil thickness, while the constraints near the trailing edge are used to prevent airfoil surface cross-over. The value of ω_{T} is set to 1.0 in order to maintain the initial airfoil thickness at $35.0\%c$.

First, we consider a one-point optimization problem for the design Mach number of 0.74. Figure 5.16(a) shows the initial and final pressure distributions and the corresponding airfoil shapes. The shock wave on the upper surface of the airfoil is eliminated, and the airfoil thickness at $35\%c$ is maintained, as indicated in Table 5.7. Figure 5.16(b) shows the values of the drag coefficient over a range of Mach numbers for $C_{\mathrm{L}} = 0.733$. The drag coefficient is reduced by 36.4% at the design Mach number. However, for Mach numbers below 0.71 the optimized airfoil performs significantly worse than the original airfoil. This is a well-known weakness of

Table 5.7: Thickness constraints for multi-point design

			Final Thickness[a]		
Constraint	Location[a]	Minimum Thickness[a]	One-point	Two-point	Four-point
1	35.0	12.04	11.97	11.97	11.88
2	96.0	0.5	0.5	0.5	0.5
3	99.0	0.12	0.12	0.12	0.1201

[a] % of airfoil chord



(a) C_p distribution and airfoil shapes ($M = 0.74$) (b) Drag coefficient at $C_L = 0.733$

Figure 5.16: One-point optimization (20 design variables)

optimizations based on a single-operating point. The resulting designs are generally not robust, i.e. their performance for mildly perturbed operating conditions is not satisfactory.

The convergence histories of the objective function and gradient are shown in Fig. 5.17, where the values are plotted at the end of each search direction. Due to the larger number of design variables, the L_2 norm of the gradient is only reduced by two orders of magnitude after 180 design iterations (flow solves and gradient evaluations). Note that after 40 design iterations the changes in the objective function value are relatively small. Significant computational effort is required to reduce the gradient by an additional two orders of magnitude. This is achieved in approximately 800 design iterations[1], which is not practical and also not necessary. In Table 5.8, we provide the percent differences between the final (converged) values of C_L and C_D and a few selected values obtained during the optimization. The final values of C_L and C_D are 0.7287 and

[1] In an effort to limit numerical errors in the BFGS Hessian approximation and also due to implementation issues, the optimization is restarted from the steepest-descent direction every 185 design iterations.

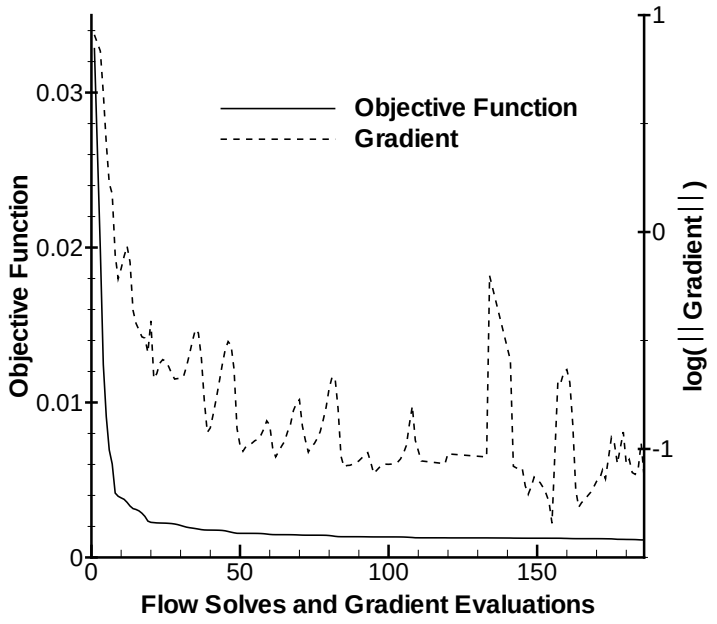


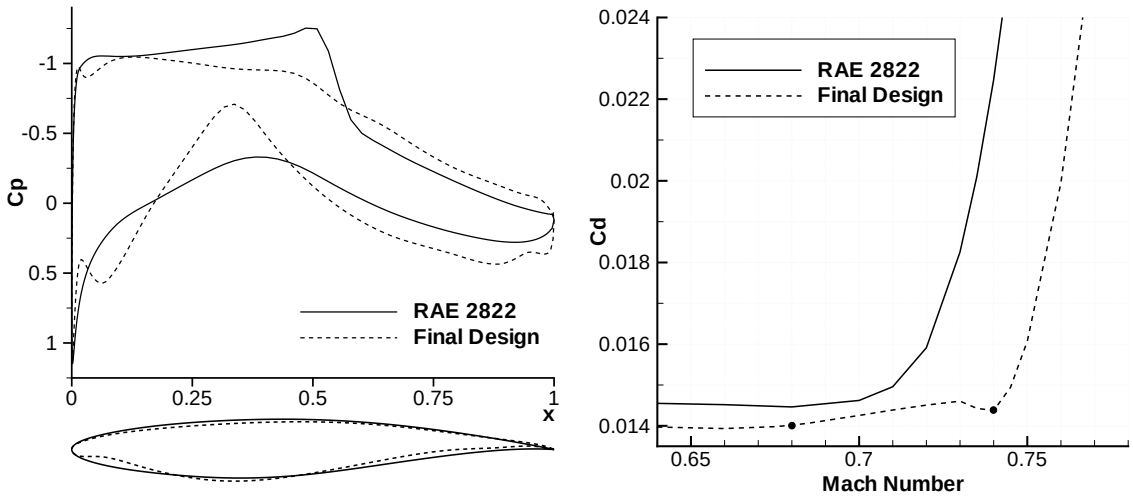
Figure 5.17: Objective function and gradient convergence histories for one-point design

Table 5.8: Convergence of aerodynamic coefficients, one-point design

Search Directions	Flow and Gradient Evaluations	C_L (% Diff.)	C_D (% Diff.)
0	1	-16.7	7.7
10	12	-1.9	6.5
20	22	-0.8	4.2
40	42	-0.5	2.8
150	181	-0.2	0.7

0.01423, respectively. Hence, within 40 design iterations the agreement between the partially converged values of the aerodynamic coefficients and the final values is very good. For example, at 40 design iterations the value of C_D is 0.01463, which is within 4 counts of the final value.

Next, we consider two- and four-point optimization problems such that the performance of the airfoil is improved over a range of Mach numbers. For the two-point problem, the selected design Mach numbers are 0.68 and 0.74 (after a few trial-and-error optimizations). The weights assigned to each design Mach number for the weighted-sum method, Eq. 2.15, are 1.0 and 2.0, respectively. Note that the same target lift and drag coefficients from the one-point optimization are used for all operating points and each operating point adds an additional angle of attack as a design variable. Figure 5.18(a) shows the initial and final pressure distributions and the corresponding airfoil shapes for the design point $M = 0.74$. Again, the upper-surface shock



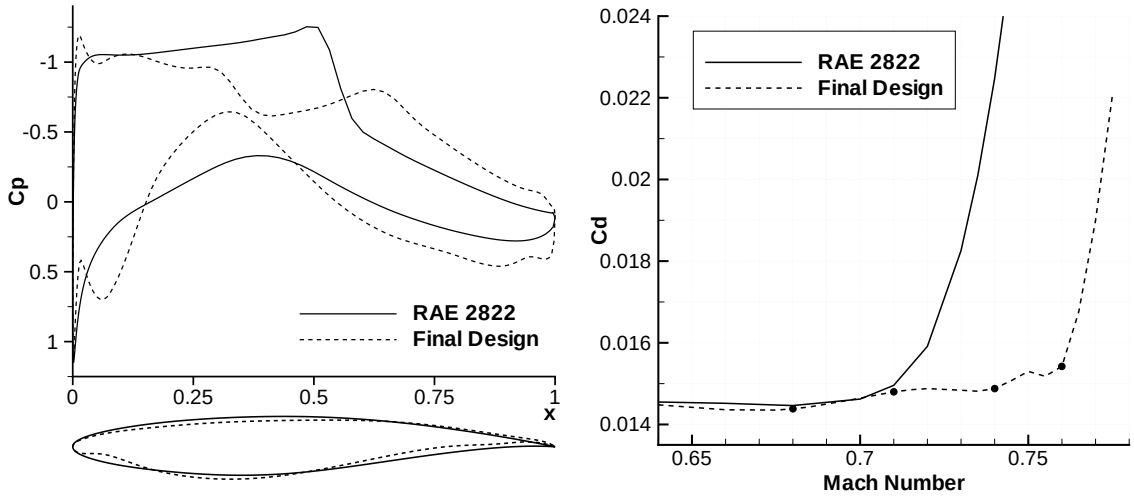
(a) C_p distribution and airfoil shapes ($M = 0.74$) (b) Drag coefficient at $C_L = 0.733$

Figure 5.18: Two-point optimization (21 design variables)

wave is eliminated and the final thicknesses at the constrained locations match the one-point design, as shown in Table 5.7. Figure 5.18(b) shows a significant improvement in the values of the drag coefficient over the Mach number range of interest for $C_L = 0.733$ when compared with the initial RAE 2822 airfoil and the one-point optimization design shown in Fig. 5.16(b). The drag coefficient is reduced by 36.3% at the design Mach number of 0.74, which is very close to the reduction obtained for the one-point optimization problem.

For the four-point problem, the selected design Mach numbers are 0.68, 0.71, 0.74 and 0.76 with associated weights of 1.0, 1.0, 2.0 and 3.0, respectively. Figure 5.19(a) shows the initial and final pressure distributions and the airfoil shapes for the design point $M = 0.74$. Unlike the one- and two-point optimization problems, the current pressure distribution indicates a weak shock on the upper surface of the airfoil. The thicknesses at the constrained locations are summarized in Table 5.7. Figure 5.19(b) shows the values of the drag coefficient over a range of Mach numbers for $C_L = 0.733$. When compared with the initial RAE 2822 airfoil, the new design achieves significantly lower drag values for Mach numbers above 0.71. The drag coefficient is reduced by 34.2% at $M = 0.74$, which is only slightly less than the reduction obtained for the one- and two-point optimization problems. More importantly, the drag-divergence Mach number is increased by 7.0% when compared with the initial RAE 2822 airfoil.

Figure 5.20 shows the objective function and gradient convergence histories, where design iterations denote the number of objective function and gradient evaluations. It is important to realize that each design iteration represents four flow and gradient evaluations. The convergence of the objective function is slightly slower than for the one-point optimization shown in Fig. 5.17, but again the largest changes in the objective function value occur within 40 design iterations.



(a) C_{p} distribution and airfoil shapes ($M = 0.74$) (b) Drag coefficient at $C_{\mathrm{L}} = 0.733$

Figure 5.19: Four-point optimization (23 design variables)

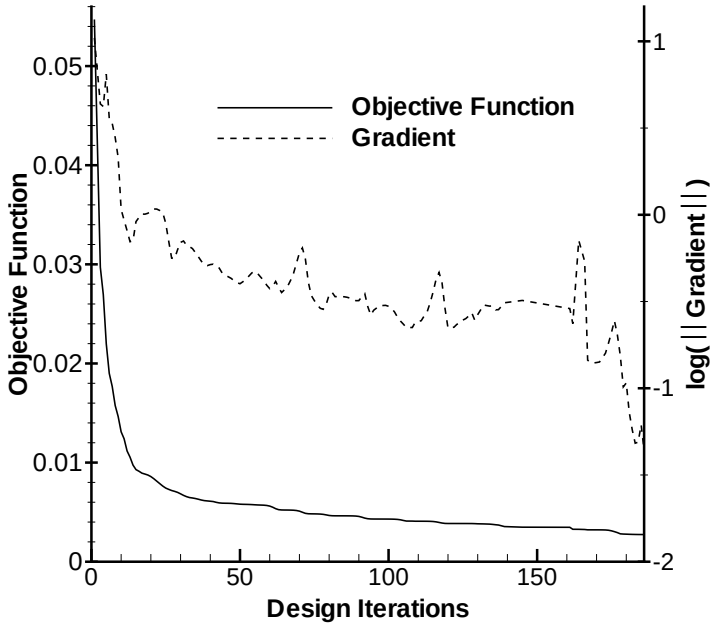


Figure 5.20: Objective function and gradient convergence histories for four-point design

Chapter 6

Conclusions and Recommendations

A Newton–Krylov algorithm for the design of single- and multi-element airfoil configurations has been developed. The algorithm consists of four modules: 1) design variables and grid perturbation, 2) flow solver, 3) gradient computation, and 4) optimizer. The design variables are based on a B-spline parameterization of the airfoil. A novel algebraic grid-perturbation strategy is introduced. The discretized Navier–Stokes and turbulence model equations are differentiated by hand and the adjoint method is used to compute the objective function gradient. The preconditioned GMRES method is applied to solve not only the flow equations, where it is used in conjunction with an inexact-Newton method, but also the adjoint equation. The accuracy of the gradient is verified by comparison with gradients based on the finite-difference and flow-sensitivity methods. The optimization problem is cast as an unconstrained problem by using quadratic penalty functions. A BFGS quasi-Newton optimizer is used to solve the unconstrained problem.

A detailed evaluation of the algorithm has been performed with emphasis on the accuracy and efficiency of the gradient computation and the efficiency of the flow solver. The results

demonstrate that the new algorithm provides a highly efficient approach for aerodynamic design problems governed by the Navier–Stokes equations. Use of the Navier–Stokes equations is particularly relevant for high-lift applications, where viscous effects and boundary-layer separation can be dominant. Key reasons for the effectiveness of the algorithm, which can also be considered the main research contributions, are the introduction of a novel preconditioning strategy for GMRES and the accurate differentiation of the residual equations. The preconditioning strategy has been optimized for both the flow solution and the gradient evaluation.

On the basis of the validation cases presented in Chapter 4 and design examples in Chapter 5, we provide the following conclusions that characterize the performance of the new algorithm:

1. The gradient computed using the adjoint method shows excellent accuracy for subsonic cases with some error relative to finite-difference and flow-sensitivity gradients in transonic cases. The error is primarily caused by treating the pressure switch used for shock capturing and the vortex strength associated with the far-field circulation correction as constants during the differentiation of the residual equations. The small error does not significantly affect the final design. However, this error can cause the line search to stall, as discussed in the lift-constrained drag minimization example. The storage of the flow Jacobian matrix required by the adjoint method results in a memory usage increase of 20% when compared with the flow-sensitivity method.

2. Robust and efficient computation of the gradient is obtained with GMRES(85) using a BFILU(6) preconditioner. A critical component of the gradient computation is the preconditioner. The preconditioner is based on an approximation of the flow Jacobian matrix. The approximation involves the artificial dissipation coefficients in conjunction with the parameter ϕ, which was set to 3.0 for single-element cases and 6.0 for multi-element cases. The preconditioner based on the approximate flow Jacobian matrix has lower memory requirements, yet provides faster convergence than a preconditioner based on the exact flow Jacobian matrix. The gradient is computed in one-fifth to one-half of the time of a warm-started flow solution, which is significantly faster than results reported in literature. Note that the adjoint residual is reduced at least three orders of magnitude in order to ensure sufficiently accurate gradients and that the computational cost of the adjoint method is independent of the number of design variables.

3. Fast convergence of the flow equations is obtained using 40 GMRES search directions and BFILU(4). The preconditioner is based on the same approximate flow Jacobian matrix used for the gradient computation, with the parameter ϕ set to 6.0. The initial guess required by Newton's method is provided by an approximate-factorization flow solver, which is also used to benchmark the performance of the new solver. The convergence of the

new solver is on average three times faster for a wide range of test cases, including subsonic and transonic flow conditions, as well as single- and multi-element airfoils. The new solver is well suited for optimization problems due to the rapid convergence of Newton's method from nearby solution states.

4. The present B-spline parameterization technique is particularly effective for multi-element configurations, where only local geometry modifications are desired. Typically, twelve B-spline control points for each airfoil surface are necessary in order to obtain a close approximation of the original airfoil and provide sufficient shape flexibility. This results in a relatively compact design space.

5. The novel grid-perturbation strategy provides improved estimates of aerodynamic performance for design examples that involve the optimization of the relative position of flaps and slats. However, for sufficiently large element displacements, the grid-perturbation strategy generates highly skewed and possibly overlapped grid cells. This breakdown is automatically detected and a grid generator is used to create a new grid.

6. The penalty formulation, which is used to impose performance and structural constraints, performed well for all cases studied here. However, it should be noted that the number of constraints considered is relatively small, i.e. less than ten constraints.

7. The weighted-sum method used for the Pareto front computation and the multi-point optimization problems provides a simple and effective approach. A notable difficulty of this approach is the selection of weights associated with each operating point, such that a robust design is obtained for a range of operating conditions.

8. For all design examples, the L_2 norm of the gradient is reduced by several orders of magnitude, which indicates convergence to a local optimum. The number of design iterations (objective function and gradient evaluations) required to reach the optimal solution depends on the number of design variables and the objective function. Objectives that involve the drag coefficient converge more slowly than objectives based purely on the lift coefficient. For design examples with approximately ten design variables, the gradient is reduced by at least four orders of magnitude within 100 design iterations. The slowest convergence is obtained for the multi-point optimization example. This example involved 20 design variables and required approximately 800 design iterations to reduce the norm of the gradient by four orders of magnitude. However, the design examples clearly demonstrate that such a large number of design iterations is *not* necessary in order to reach engineering accuracy. The greatest improvements in the objective function value occur

within just 10 design iterations, and the results are converged to engineering accuracy within 50 design iterations for all examples considered.

9. Design examples that involve the optimization of high-lift configurations represent the most important practical application of the new algorithm. The use of gradient-based methods for high-lift design is demonstrated to be advantageous. This is due to the fact that relatively few flow solutions are required in order to obtain significant design improvements, which is a consequence of the accurate and efficient evaluation of the gradient.

10. Convergence to only local optimal solutions is a well-known limitation of gradient-based methods. We verify the uniqueness of the optimal solution in two design examples, namely, the flap position optimization and the Pareto front computation. In the former example, two distinct initial conditions are used and we show that within a practical region of the design space the objective function is unimodal. In the latter example, a genetic algorithm specifically tailored for the computation of global Pareto fronts is used to verify the uniqueness of the gradient-based front.

The conclusions suggest a number of improvements that could enhance the capabilities of the present algorithm. These improvements, as listed below, could be implemented and evaluated within the established framework:

- For optimization problems that are dominated by transonic flow with strong shocks or supersonic flow, a differentiation of the pressure switch would provide better convergence characteristics and accuracy of the optimization. Differentiation of the current pressure switch, Eqs. 3.9-3.11, is problematic, since it contains a combination of several discontinuous functions. Furthermore, the differentiation would result in an increased bandwidth of the flow Jacobian. Perhaps an appropriate simplification of the pressure switch, or the use of a simpler limiter function that is already available in the flow solver [122], would resolve this issue.

- Modifications of the grid-perturbation strategy such that the location of neighbouring block boundaries is adjusted as the multi-element configuration changes could increase the range of motion for flaps and slats and improve the quality of the modified grid. Grid-topology changes appear to be a limitation of the present multi-block approach. This motivates the development of Newton–Krylov algorithms for unstructured and chimera grids, as well as the development of automatic grid-generation tools.

- Matrix or CUSP dissipation [122], a higher-order algorithm [34], and the ability to specify (or predict) boundary-layer transition could be used to improve the accuracy of the flow

solver and consequently, the accuracy of the optimization. The implementation of these techniques such that the efficiency of the Newton–Krylov flow solver and the gradient computation algorithm is maintained presents a significant research challenge.

- Advanced optimizers, such as SQP methods, could improve the convergence and robustness of the optimization and provide a more efficient treatment of constraints. The use of grid sequencing could also provide significant convergence acceleration. Note that the lift constraint could be eliminated by forcing the flow solution to match the desired lift by adjusting the angle of attack within the flow solver at each iteration of the optimization.

- Further development of methods for problems with multiple objectives and operating points is required. For multi-point problems, the profile optimization method recently introduced by Li *et al.* [100] and the probabilistic approach suggested by Huyse *et al.* [81] are promising robust design techniques. An investigation of goal-attainment methods and the use of concepts from game theory [137] could provide insight into multi-objective problems.

- Newton–Krylov algorithms offer a promising approach for multidisciplinary optimization problems, in particular aero-structural optimization, since an efficient synthesis of solvers based on computational structural mechanics and computational fluid dynamics could be achieved, for example see [105].

- Assessments of the physical models and the required grid refinement are necessary in order to establish the validity of the optimal designs. For example, an investigation of the ability to quantitatively predict aerodynamic performance due to a change in the position of a flap or a slat is required. Additional areas of interest include shape optimization subject to flow with regions of large separation, for instance the flow at post-stall conditions, and internal-flow problems.

- The algorithm should be extended to three-dimensional problems. A critical factor is the large memory requirement of Krylov methods. The use of parallel processing techniques, such as Schwarz methods [70] and parallel ILU preconditioners [41], should be investigated.

As a final thought, we would like to emphasize that although the Newton–Krylov algorithm provides a strong foundation for effective aerodynamic shape optimization tools, the algorithm addresses only a fraction of the design problem. For example, the realization that vortex generators can in some instances provide a performance advantage or the decision of how many high-lift elements are optimal, requires the insight and skill of an experienced aerodynamicist. Perhaps concepts from topology optimization and even artificial intelligence may guide the development of future aerodynamic design algorithms.

REFERENCES

[1] I. H. Abbott and A. E. Von Doenhoff, *Theory of Wing Sections, Including a Summary of Airfoil Data*, Dover Publications, New York, 1959.

[2] J. Ahn, H.-J. Kim, D.-H. Lee, and O.-H. Rho, *Response surface method for airfoil design in transonic flow*, Journal of Aircraft, 38 (2001), pp. 231–238.

[3] N. M. Alexandrov, E. J. Nielsen, R. M. Lewis, and W. K. Anderson, *First-order model management with variable-fidelity physics applied to multi-element airfoil optimization*, AIAA Paper 2000–4886, September 2000.

[4] J. J. Alonso, I. M. Kroo, and A. Jameson, *Advanced algorithms for design and optimization of quiet supersonic platforms*, AIAA Paper 2002–0144, Reno, NV, Jan. 2002.

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

[6] 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 Journal, 39 (2001), pp. 56–63.

[7] W. K. Anderson, R. D. Rausch, and D. L. Bonhaus, *Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids*, Journal of Computational Physics, 128 (1996), pp. 391–408.

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

[9] T. J. Barth and S. W. Linton, *An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation*, AIAA Paper 95–0221, Jan. 1995.

[10] O. Baysal and M. E. Eleshaky, *Aerodynamic sensitivity analysis methods for the compressible Euler equations*, Journal of Fluids Engineering, 113 (1991), pp. 681–688.

[11] O. Baysal and K. Ghayour, *Continuous adjoint sensitivities for optimization with general cost functionals on unstructured meshes*, AIAA Journal, 39 (2001), pp. 48–55.

[12] R. M. Beam and R. F. Warming, *An implicit factored scheme for the compressible Navier–Stokes equations*, AIAA Journal, 16 (1978), pp. 393–402.

[13] S. Bellavia and B. Morini, *A globally convergent Newton–GMRES subspace method for systems of nonlinear equations*, SIAM Journal on Scientific Computing, 23 (2001), pp. 940–960.

[14] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1652–1670.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Gradient convergence in gradient methods with errors*, SIAM Journal on Optimization, 10 (2000), pp. 627–642.

[16] G. Biros and O. Ghattas, *Parallel Newton–Krylov methods for pde-constrained optimization*, in Proceedings of SC99, Portland, OR, Nov. 1999.

[17] M. Blanco and D. W. Zingg, *Fast Newton–Krylov method for unstructured grids*, AIAA Journal, 36 (1998), pp. 607–612.

[18] A. M. Bruaset, A. Tveito, and R. Winther, *On the stability of relaxed incomplete LU factorizations*, Mathematics of Computation, 54 (1990), pp. 701–719.

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

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

[21] A. Chapman, Y. Saad, and L. Wigton, *High-order ILU preconditioners for CFD problems*, International Journal for Numerical Methods in Fluids, 33 (2000), pp. 767–788.

[22] T. Chisholm and D. W. Zingg, *A fully coupled Newton-Krylov solver for turbulent aerodynamic flows*, Paper 333, ICAS 2002, Toronto, ON, Sept. 2002.

[23] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, Journal of Computational and Applied Mathematics, 86 (1997), pp. 387–414.

[24] H.-S. Chung and J. J. Alonso, *Using gradients to construct Cokriging approximation models for high-dimensional design optimization problems*, AIAA Paper 2002–0317, Reno, NV, Jan. 2002.

[25] C. A. C. Coello, *A comprehensive survey of evolutionary-based multiobjective optimization techniques*, Knowledge and Information Systems, 1 (1999), pp. 269–308. Also see www.lania.mx/∼ccoello/EMOO/EMOObib.html.

[26] P. H. Cook, M. A. MacDonald, and M. C. P. Firmin, *Aerofoil RAE 2822 - pressure distributions, and boundary-layer and wake measurements.* AGARD-AR-138, May 1979.

[27] G. B. Cosentino and T. L. Holst, *Numerical optimization design of advanced transonic wing configurations*, Journal of Aircraft, 23 (1986), pp. 192–199.

[28] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in 24th National Conference of the Association for Computing Machinery, no. ACM P-69, New York, 1969, Brandon Press, pp. 157–172.

[29] A. Dadone and B. Grossman, *Progressive optimization of inverse fluid dynamic design problems*, Computers & Fluids, 29 (2000), pp. 1–32.

[30] ——, *Fast convergence of viscous airfoil design problems*, AIAA Journal, 40 (2002), pp. 1997–2005.

[31] A. Dadone, B. Mohammadi, and N. Petruzzelli, *Incomplete sensitivities and bfgs methods for 3D aerodynamic shape design*, Tech. Rep. 3633, Institut National De Recherche En Informatique Et En Automatique (INRIA), France, March 1999.

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

[33] S. De Rango, *Higher-Order Spatial Discretization for Turbulent Aerodynamic Flows*, PhD thesis, University of Toronto, 2001.

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

[35] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[36] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, N.J., 1983.

[37] M. Drela, *Design and optimization method for multi-element airfoils*, AIAA Paper 93–0969, 1993.

[38] ——, *Pros & cons of airfoil optimization*, in Frontiers of Computational Fluid Dynamics 1998, D. A. Caughey and M. M. Hafez, eds., Singapore, 1998, World Scientific, pp. 363–381.

[39] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[40] L. C. Dutto, *The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier–Stokes equations*, International Journal for Numerical Methods in Engineering, 36 (1993), pp. 457–497.

[41] L. C. Dutto and W. G. Habashi, *Parallelization of the ILU(0) preconditioner for CFD problems on shared-memory computers*, International Journal for Numerical Methods in Fluids, 30 (1999), pp. 995–1008.

[42] R. Duvigneau and M. Visonneau, *Shape optimization of incompressible and turbulent flows using the simplex method*, AIAA Paper 2001–2533, Anaheim, CA, Sept. 2001.

[43] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[44] M. E. Eleshaky and O. Baysal, *Discrete aerodynamic sensitivity analysis on decomposed computational domains*, Computers & Fluids, 23 (1994), pp. 595–611.

[45] J. Elliott and W. Herling, *A chimera approach to aerodynamic shape optimization for the compressible, high-Re Navier Stokes equations*, AIAA Paper 2000–4729, Long Beach, CA, Sept. 2000.

[46] J. Elliott and J. Peraire, *Aerodynamic optimization on unstructured meshes with viscous effects*, AIAA Paper 97–1849, June 1997.

[47] ——, *Practical 3D aerodynamic design and optimization using unstructured meshes*, AIAA Journal, 35 (1997), pp. 1479–1485.

[48] ——, *Constrained, multipoint shape optimisation for complex 3D configurations*, Aeronautical Journal, 102 (1998), pp. 365–376.

[49] ——, *Progress towards a 3D aerodynamic shape optimization tool for the compressible high-Re Navier Stokes equations discretized on unstructured meshes*, AIAA Paper 98–2897, June 1998.

[50] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Mathematics of Computation, 47 (1986), pp. 191–217.

[51] ——, *Relaxed and stabilized incomplete factorizations for non-self-adjoint linear systems*, BIT, 29 (1989), pp. 890–915.

[52] S. Eyi, K. D. Lee, S. E. Rogers, and D. Kwak, *High-lift design optimization using Navier–Stokes equations*, Journal of Aircraft, 33 (1996), pp. 499–504.

[53] G. E. Farin, *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*, Academic Press, San Diego, 4th ed., 1997.

[54] I. Fejtek, *Summary of code validation results for a multiple element airfoil test case*, AIAA Paper 97–1932, June 1997.

[55] 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, Anaheim, CA, June 2001.

[56] D. Feng and T. H. Pulliam, *An all-at-once reduced hessian SQP scheme for aerodynamics design optimization*, Tech. Rep. 95–24, RIACS, 1995.

[57] P. D. Frank and G. R. Shubin, *A comparison of optimization-based approaches for a model computational aerodynamics design problem*, Journal of Computational Physics, 98 (1992), pp. 74–89.

[58] J. Gatsis and D. W. Zingg, *A fully-coupled algorithm for aerodynamic design optimization*, in 48th Annual CASI Conference, Toronto, ON, 2001, pp. 227–236.

[59] K. C. Giannakoglou, *Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence*, Progress in Aerospace Sciences, 38 (2002), pp. 43–76.

[60] M. B. Giles, *Aerospace design: A complex task*, in Inverse Design and Optimization Methods, 1997-05, von Karman Institute For Fluid Dynamics, Brussels, Belgium, April 1997.

[61] M. B. Giles and M. Drela, *Two-dimensional transonic aerodynamic design method*, AIAA Journal, 25 (1987), pp. 1199–1206.

[62] M. B. Giles, M. C. Duta, and J.-D. Müller, *Adjoint code developments using the exact discrete approach*, AIAA Paper 2001–2596, June 2001.

[63] M. B. Giles and N. A. Pierce, *An introduction to the adjoint approach to design*, Flow, Turbulence and Combustion, 65 (2000), pp. 393–415.

[64] ——, *Analytic adjoint solutions for the quasi-one-dimensional Euler equations*, Journal of Fluid Mechanics, 426 (2001), pp. 327–345.

[65] ——, *Adjoint error correction for integral outputs*, in Lecture Notes in Computer Science, Springer-Verlag, 2002. To appear, see http://web.comlab.ox.ac.uk/oucl/work/mike.giles/.

[66] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, Toronto, 1981.

[67] P. Godin, D. W. Zingg, and T. E. Nelson, *High-lift aerodynamic computations with one- and two-equation turbulence models*, AIAA Journal, 35 (1997), pp. 237–243.

[68] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, Mass., 1989.

[69] R. M. Greenman and K. R. Roth, *Minimizing computational data requirements for multi-element airfoils using neural networks*, Journal of Aircraft, 36 (1999), pp. 777–784.

[70] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *High-performance parallel implicit CFD*, Parallel Computing, 27 (2001), pp. 337–362.

[71] M. D. Gunzburger, *Introduction into mathematical aspects of flow control and optimization*, in Inverse Design and Optimization Methods, 1997-05, von Karman Institute For Fluid Dynamics, Brussels, Belgium, April 1997.

[72] ——, *Adjoint equation-based methods for control problems in incompressible, viscous flows*, Flow, Turbulence and Combustion, 65 (2000), pp. 249–272.

[73] J. S. Hansen, Z. S. Liu, and N. Olhoff, *Shape sensitivity analysis using a fixed basis function finite element approach*, Structural and Multidisciplinary Optimization, 21 (2001), pp. 177–195.

[74] J.-W. He, R. Glowinski, R. Metcalfe, A. Nordlander, and J. Periaux, *Active control and drag optimization for flow past a circular cylinder*, Journal of Computational Physics, 163 (2000), pp. 83–117.

[75] R. M. Hicks and P. A. Henne, *Wing design by numerical optimization*, Journal of Aircraft, 15 (1978), pp. 407–412.

[76] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats, *An assessment of airfoil design by numerical optimization*, tech. rep., NASA TM X-3092, July 1974.

[77] C. Hirsch, *Numerical Computation of Internal and External Flows*, vol. 2, John Wiley & Sons, England, 1994.

[78] T. L. Holst, *Viscous transonic airfoil workshop compendium of results*, AIAA Paper 87–1460, Honolulu, HI, June 1987.

[79] T. L. Holst and T. H. Pulliam, *Aerodynamic shape optimization using a real-number-encoded genetic algorithm*, AIAA Paper 2001–2473, Anaheim, CA, June 2001.

[80] J. Hoschek, *Intrinsic parametrization for approximation*, Computer Aided Geometric Design, 5 (1988), pp. 27–31.

[81] L. Huyse, S. Padula, R. M. Lewis, and W. Li, *Probabilistic approach to free-form airfoil shape optimization under uncertainty*, AIAA Journal, 40 (2002), pp. 1764–1772.

[82] A. Iollo, G. Kuruvila, and S. Ta'asan, *Pseudo-time method for optimal shape design using the euler equations*, NASA CR–198205, Aug. 1995. Also ICASE report 95–59.

[83] A. Iollo and L. Zannetti, *Optimal control of a vortex trapped by an airfoil with a cavity*, Flow, Turbulence and Combustion, 65 (2000), pp. 417–430.

[84] A. Jameson, *Aerodynamic design via control theory*, Journal of Scientific Computing, 3 (1988), pp. 233–260. Also ICASE report 88–64.

[85] ——, *A perspective on computational algorithms for aerodynamic analysis and design*, Progress in Aerospace Sciences, 37 (2001), pp. 197–243.

[86] A. Jameson, N. A. Pierce, and L. Martinelli, *Optimum aerodynamic design using the Navier-Stokes equations*, Theoretical and Computational Fluid Dynamics, 10 (1998), pp. 213–237.

[87] A. Jameson and J. C. Vassberg, *Studies of alternative numerical optimization methods applied to the Brachistochrome problem*, Computational Fluid Dynamics Journal, 9 (2000), pp. 281–296.

[88] ——, *Computational fluid dynamics for aerodynamic design: Its current and future impact*, AIAA Paper 2001–0538, Reno, NV, Jan. 2001.

[89] W. H. Jou, W. P. Huffman, D. P. Young, R. G. Melvin, M. B. Bieterman, C. L. Hilmes, and F. T. Johnson, *Practical considerations in aerodynamic design optimization*, AIAA Paper 95–1730, 1995.

[90] C. S. Kim, C. Kim, and O. H. Rho, *Sensitivity analysis for the Navier–Stokes equations with two-equation turbulence models*, AIAA Journal, 39 (2001), pp. 838–845.

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

[92] S. Kim, J. J. Alonso, and A. Jameson, *Two–dimensional high–lift aerodynamic optimization using the continuous adjoint method*, AIAA Paper 2000–4741, Sept. 2000.

[93] ——, *Design optimization of high–lift configurations using a viscous continuous adjoint method*, AIAA Paper 2002–0844, Jan. 2002.

[94] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

[95] S. M. Klausmeyer and J. C. Lin, *Comparative results from a CFD challenge over a 2D three-element high-lift airfoil*, NASA TM 112858, May 1997.

[96] V. M. Korivi, A. C. Taylor III, P. A. Newman, G. W. Hou, and H. E. Jones, *An approximately factored incremental strategy for calculating consistent discrete aerodynamic sensitivity derivatives*, Journal of Computational Physics, 113 (1994), pp. 336–346.

[97] P. A. LeGresley and J. J. Alonso, *Investigation of non-linear projection for pod based reduced order models for aerodynamics*, AIAA Paper 2001–0926, Reno, NV, Jan. 2001.

[98] J. Lépine, F. Guibault, J. Y. Trépanier, and F. Pépin, *Optimized nonuniform rational B-spline geometrical representation for aerodynamic design of wings*, AIAA Journal, 39 (2001), pp. 2033–2041.

[99] R. M. Lewis, V. Torczon, and M. W. Trosset, *Direct search methods: Then and now*, NASA CR–2000–210125, May 2000. Also ICASE Report No. 2000–26.

[100] W. Li, L. Huyse, and S. Padula, *Robust airfoil optimization to achieve drag reduction over a range of mach numbers*, Structural and Multidisciplinary Optimization, 24 (2002), pp. 38–50.

[101] R. H. Liebeck, *A class of airfoils designed for high lift in incompressible flow*, Journal of Aircraft, 10 (1973), pp. 610–617.

[102] M. J. Lighthill, *A new method of two-dimensional aerodynamic design*, R&M 2112, Aeronautical Research Council, June 1945.

[103] W.-H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill–McKee and the reverse Cuthill–Mckee ordering algorithms for sparse matrices*, SIAM Journal on Numerical Analysis, 13 (1976), pp. 198–213.

[104] H. Lomax, T. H. Pulliam, and D. W. Zingg, *Fundamentals of Computational Fluid Dynamics*, Springer–Verlag, Berlin, Heidelberg, 2001.

[105] E. Lund, H. Møller, and L. A. Jakobsen, *Shape optimization of fluid-structure interaction problems using two-equation turbulence models*, AIAA Paper 2002–1478, Denver, CO, April 2002.

[106] J. N. Lyness and C. B. Moler, *Numerical differentiation of analytic functions*, SIAM Journal on Numerical Analysis, 4 (1967), pp. 202–210.

[107] C. M. Maksymiuk and T. H. Pulliam, *Viscous transonic airfoil workshop results using ARC2D*, AIAA Paper 87–0415, Reno, NV, Jan. 1987.

[108] N. Marco, J.-A. Désidéri, and S. Lanteri, *Multi-objective optimization in CFD by genetic algorithms*, Tech. Rep. 3686, Institut National De Recherche En Informatique Et En Automatique (INRIA), France, April 1999. Also see www.lania.mx/∼ccoello/EMOO/EMOObib.html.

[109] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther, *High-fidelity aero-structural design optimization of a supersonic business jet*, AIAA Paper 2002–1483, April 2002.

[110] J. R. R. A. Martins, I. M. Kroo, and J. J. Alonso, *An automated method for sensitivity analysis using complex variables*, AIAA Paper 2000–0689, Jan. 2000.

[111] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[112] R. G. Melvin, W. P. Huffman, D. P. Young, F. T. Johnson, C. L. Hilmes, and M. B. Bieterman, *Recent progress in aerodynamic design optimization*, International Journal for Numerical Methods in Fluids, 30 (1999), pp. 205–216.

[113] 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.

[114] ——, *Practical applications to fluid flows of automatic differentiation for design problems*, in Inverse Design and Optimization Methods, 1997-05, von Kármán Institute For Fluid Dynamics, Brussels, Belgium, April 1997.

[115] L. Moreau and D. Aeyels, *Optimization of discontinuous functions: A generalized theory of differentiation*, SIAM Journal on Optimization, 11 (2000), pp. 53–69.

[116] B. Morini, *Convergence behaviour of inexact Newton methods*, Mathematics of Computation, 68 (1999), pp. 1605–1613.

[117] S. Nadarajah and A. Jameson, *A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization*, AIAA Paper 2000–0667, Jan. 2000.

[118] ——, *Studies of the continuous and discrete adjoint approach to viscous automatic aerodynamic optimization*, AIAA Paper 2001–2530, June 2001.

[119] T. Nelson, *Numerical Solution of the Navier–Stokes Equations for High-Lift Airfoil Configurations*, PhD thesis, University of Toronto, 1994.

[120] T. E. Nelson, P. Godin, S. De Rango, and D. W. Zingg, *Flow computations for a three-element airfoil system*, Canadian Aeronautics and Space Journal, 45 (1999), pp. 132–139.

[121] T. E. Nelson, D. W. Zingg, and G. W. Johnston, *Compressible Navier–Stokes computations of multielement airfoil flows using multiblock grids*, AIAA Journal, 32 (1994), pp. 506–511.

[122] M. Nemec and D. W. Zingg, *Aerodynamic computations using the convective-upstream split pressure scheme with local preconditioning*, AIAA Journal, 38 (2000), pp. 402–410.

[123] ——, *Aerodynamic shape optimization using the discrete adjoint method*, in 48th Annual CASI Conference, Toronto, ON, 2001, pp. 203–213.

[124] ——, *A Newton–Krylov algorithm for complex aerodynamic design*, 10th CFD Society of Canada Conference, Windsor, ON, June 2001.

[125] ——, *Towards efficient aerodynamic shape optimization based on the Navier–Stokes equations*, AIAA Paper 2001–2532, Anaheim, CA, June 2001.

[126] ——, *From analysis to design of high-lift configurations using a Newton–Krylov algorithm*, Paper 173, ICAS 2002, Toronto, ON, Sept. 2002.

[127] ——, *Newton–Krylov algorithm for aerodynamic design using the Navier–Stokes equations*, AIAA Journal, 40 (2002), pp. 1146–1154.

[128] M. Nemec, D. W. Zingg, and T. H. Pulliam, *Multi-point and multi-objective aerodynamic shape optimization*, AIAA Paper 2002–5548, Atlanta, GA, Sept. 2002.

[129] J. C. Newman III, A. C. Taylor III, R. W. Barnwell, P. A. Newman, and G. J.-W. Hou, *Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations*, Journal of Aircraft, 36 (1999), pp. 87–96.

[130] E. J. Nielsen, *Aerodynamic Design Sensitivities on an Unstructured Mesh Using the Navier Stokes Equations and a Discrete Adjoint Formulation*, PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[131] E. J. Nielsen and W. K. Anderson, *Aerodynamic design optimization on unstructured meshes using the Navier Stokes equations*, AIAA Journal, 37 (1999), pp. 1411–1419.

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

[133] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer–Verlag, New York, 1999.

[134] S. Obayashi, *Aerodynamic optimization with evolutionary algorithms*, in Inverse Design and Optimization Methods, Lecture Series 1997-05, R. A. Van den Braembussche and M. Manna, eds., Brussels, Belgium, 1997, von Karman Institute for Fluid Dynamics.

[135] J. C. Otto, M. Paraschivoiu, and S. Y. A. T. Patera, *Computer-simulation surrogates for optimization: Application to trapezoidal ducts and axisymmetric bodies*, in Proceedings of the Forum on CFD for Design and Optimization, San Francisco, 1995, ASME International Mechanical Engineering Conference and Exposition.

[136] A. Oyama, *Wing Design Using Evolutionary Algorithms*, PhD thesis, Tohoku University, 2000.

[137] J. Periaux, H. Q. Chen, B. Mantel, M. Sefrioui, and H. T. Sui, *Combining game theory and genetic algorithms with application to ddm-nozzle optimization problems*, Finite Elements in Analysis and Design, 37 (2001), pp. 417–429.

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

[139] ——, *Optimal shape design by local boundary variations*, in Optimal Shape Design, A. Cellina and A. Ornelas, eds., Springer–Verlag, Berlin, Heidelberg, 2000.

[140] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77, The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 2nd ed., 1992.

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

[142] A. Pueyo and D. W. Zingg, *Efficient Newton–Krylov solver for aerodynamic computations*, AIAA Journal, 36 (1998), pp. 1991–1997.

[143] 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.

[144] T. H. Pulliam and D. S. Chaussee, *A diagonal form of an implicit approximate factorization algorithm*, Journal of Computational Physics, 39 (1981), pp. 347–363.

[145] J. J. Reuther, J. J. Alonso, M. J. Rimlinger, and A. Jameson, *Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on distributed memory parallel computers*, Computers & Fluids, 28 (1999), pp. 675–700.

[146] 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, 36 (1999), pp. 51–60.

[147] ——, *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2*, Journal of Aircraft, 36 (1999), pp. 61–74.

[148] D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics*, McGraw–Hill Publishing Co., New York, 1990.

[149] C. L. Rumsey and S. X. Ying, *Prediction of high lift: Review of present CFD capability*, Progress in Aerospace Sciences, 38 (2002), pp. 145–180.

[150] Y. Saad, *Iterative Methods For Sparse Linear Systems*, International Thomson Publishing Company, Boston, 1996.

[151] ——, *Sparskit: a basic tool kit for sparse matrix computations*, tech. rep., http://www-users.cs.umn.edu/~saad/software.html, 2002.

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

[153] J. A. Samareh, *Novel multidisciplinary shape parametrization approach*, Journal of Aircraft, 38 (2001), pp. 1015–1023.

[154] ——, *Survey of shape parametrization techniques for high-fidelity multidisciplinary shape optimization*, AIAA Journal, 39 (2001), pp. 877–883.

[155] J. N. Shadid, R. S. Tuminaro, and H. F. Walker, *An inexact Newton method for fully coupled solution of the Navier–Stokes equations with heat and mass transport*, Journal of Computational Physics, 137 (1997), pp. 155–185.

[156] A. Shenoy, M. Heinkenschloss, and E. M. Cliff, *Airfoil design by an all-at-once method*, IJCFD, 11 (1998), pp. 3–25.

[157] L. L. Sherman, A. C. Taylor III, L. L. Green, P. A. Newman, G. W. Hou, and V. M. Korivi, *First- and second-order aerodynamic sensitivity derivatives via automatic differentiation with incremental iterative methods*, Journal of Computational Physics, 129 (1996), pp. 307–331.

[158] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM Journal on Scientific Computing, 16 (1995), pp. 917–933.

[159] A. M. O. Smith, *High-lift aerodynamics*, Journal of Aircraft, 12 (1975), pp. 501–530.

[160] H. Sobieczky, *Parametric airfoils and wings*, in Recent Development of Aerodynamic Design Methodologies– Inverse Design and Optimization, Braunschweig/Wiesbaden, Germany, 1999, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, pp. 72–74.

[161] J. Sobieszczanski-Sobieski and R. T. Haftka, *Multidisciplinary aerospace design optimization: Survey of recent developments*, AIAA Paper 96–0711, Reno, NV, Jan. 1996.

[162] B. I. Soemarwoto and T. E. Labrujère, *Airfoil design and optimization methods: Recent progress at NLR*, International Journal for Numerical Methods in Fluids, 30 (1999), pp. 217–228.

[163] O. Soto and R. Löhner, *CFD shape optimization using an incomplete-gradient adjoint formulation*, International Journal for Numerical Methods in Engineering, 51 (2001), pp. 735–753.

[164] P. R. Spalart and S. R. Allmaras, *A one-equation turbulence model for aerodynamic flows*, AIAA Paper 92–0439, Jan. 1992.

[165] C. Sung and J. H. Kwon, *Accurate aerodynamic sensitivity analysis using adjoint equations*, AIAA Journal, 38 (2000), pp. 243–250.

[166] ——, *Aerodynamic design optimization using the Navier–Stokes and adjoint equations*, AIAA Paper 2001–0266, Jan. 2001.

[167] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation, Foundations and Applications*, Elsevier Science Publishing Co., New York, 1985.

[168] J. Thuburn and T. W. N. Haine, *Adjoints of nonoscillatory advection schemes*, Journal of Computational Physics, 171 (2001), pp. 616–631.

[169] J. Y. Trépanier, J. Lépine, and F. Pépin, *An optimized geometric representation for wing profiles using NURBS*, Canadian Aeronautics and Space Journal, 46 (2000), pp. 12–19.

[170] D. Tse and L. Chan, *Transonic airfoil design optimization using soft computing methods*, Canadian Aeronautics and Space Journal, 46 (2000), pp. 65–73.

[171] E. Turgeon, D. Pelletier, and J. Borggaard, *A continuous sensitivity equation approach to optimal design in mixed convection*, AIAA Paper 99–3625, Norfolk, VA, June 1999.

[172] C. P. van Dam, *The aerodynamic design of multi-element high-lift systems for transport airplanes*, Progress in Aerospace Sciences, 38 (2002), pp. 101–144.

[173] B. Van Den Berg, *Boundary layer measurements on a two-dimensional wing with flap*, National Aerospace Laboratory NLR TR 79009 U, Amsterdam, The Netherlands, Jan. 1979.

[174] D. A. Venditti and D. L. Darmofal, *Grid adaptation for functional outputs: Application to two-dimensional inviscid flows*, Journal of Computational Physics, 176 (2002), pp. 40–69.

[175] V. Venkatakrishnan and D. J. Mavriplis, *Implicit solvers for unstructured meshes*, Journal of Computational Physics, 105 (1993), pp. 83–91.

[176] P. C. Walsh, *Adaptive Solution of Viscous Aerodynamic Flows Using Unstructured Grids*, PhD thesis, University of Toronto, 1998.

[177] Z. Wang and K. Droegemeier, *The adjoint Newton algorithm for large-scale unconstrained optimization in meteorology applications*, Computational Optimization and Applications, 10 (1998), pp. 283–320.

[178] A. R. Wilkinson and D. W. Zingg, *AMBER 2d User's Guide*, 1993.

[179] P. Wong, *Aerodynamic optimization using the flow sensitivity approach*, MASc thesis, University of Toronto, 2001.

[180] P. Wong and D. W. Zingg, *Aerodynamic optimization using the flow sensitivity approach*, in 48th Annual CASI Conference, Toronto, ON, 2001, pp. 243–251.

[181] G. A. Wrenn, *An indirect method for numerical optimization using the Kreisselmeier-Steinhauser function*, NASA CR–4220, March 1989.

[182] D. W. Zingg, *Grid studies for thin-layer Navier-Stokes computations of airfoil flowfields*, AIAA Journal, 30 (1992), pp. 2561–2564.

[183] D. W. Zingg, S. De Rango, and A. Pueyo, *Advances in algorithms for computing aerodynamic flows*, in Frontiers of Computational Fluid Dynamics 2002, D. A. Caughey and M. M. Hafez, eds., Singapore, 2002, World Scientific.

[184] D. W. Zingg, M. Nemec, and T. Chisholm, *A Newton–Krylov algorithm for aerodynamic analysis and design*, keynote paper, International Conference on Computational Fluid Dynamics (ICCFD) 2, Sydney, AU, July 2002.

[185] D. W. Zingg, S. D. Rango, M. Nemec, and T. H. Pulliam, *Comparison of several spatial discretizations for the Navier–Stokes equations*, Journal of Computational Physics, 160 (2000), pp. 683–704.

APPENDICES

Appendix A

Boundary Conditions

For C-topology grids, the normal and tangential velocity components at the far-field boundary ($k = k_{\max}$, see Fig. 3.2) are given by

$$V_n = \frac{\eta_x\, u + \eta_y\, v}{\sqrt{\eta_x^2 + \eta_y^2}} \tag{A.1}$$

$$V_t = \frac{\eta_y\, u - \eta_x\, v}{\sqrt{\eta_x^2 + \eta_y^2}} \tag{A.2}$$

For H-topology grids, the normal and tangential velocity components (see Fig. 3.2(b) and for further details see [33, 119]) are given by

$$V_n = \phi(\tilde{\kappa_x} u + \tilde{\kappa_y} v) \begin{cases} \phi = 1 & \text{sides 3 and 4} \\ \phi = -1 & \text{sides 1 and 2} \end{cases} \tag{A.3}$$

$$V_t = \phi(\tilde{\kappa_y} u - \tilde{\kappa_x} v) \begin{cases} \phi = 1 & \text{sides 1 and 4} \\ \phi = -1 & \text{sides 2 and 3} \end{cases} \tag{A.4}$$

The metric terms are defined by

$$\tilde{\kappa_x} = \begin{cases} \dfrac{\xi_x}{\sqrt{\xi_x^2+\xi_y^2}} & \text{sides 2 and 4} \\ \dfrac{\eta_x}{\sqrt{\eta_x^2+\eta_y^2}} & \text{sides 1 and 3} \end{cases} \quad \text{and} \quad \tilde{\kappa_y} = \begin{cases} \dfrac{\xi_y}{\sqrt{\xi_x^2+\xi_y^2}} & \text{sides 2 and 4} \\ \dfrac{\eta_y}{\sqrt{\eta_x^2+\eta_y^2}} & \text{sides 1 and 3} \end{cases} \tag{A.5}$$

The flow variables at the far-field boundary are computed using the following expressions:

$$V_n = \frac{1}{2}(R_1 + R_2) \tag{A.6}$$

$$a = \frac{1}{4}(\gamma - 1)(R_2 - R_1) \tag{A.7}$$

$$\rho = (\frac{1}{\gamma}\, a^2\, R_4)^{\frac{1}{\gamma-1}} \tag{A.8}$$

$$p = \frac{1}{\gamma}\,\rho\, a^2 \tag{A.9}$$

and the velocities are obtained from

$$u = \begin{cases} -\tilde{\kappa_x} V_n + \tilde{\kappa_y} V_t & \text{side 1} \\ -\tilde{\kappa_x} V_n - \tilde{\kappa_y} V_t & \text{side 2} \\ \tilde{\kappa_x} V_n - \tilde{\kappa_y} V_t & \text{side 3} \\ \tilde{\kappa_x} V_n + \tilde{\kappa_y} V_t & \text{side 4} \end{cases} \quad \text{and} \quad v = \begin{cases} -\tilde{\kappa_y} V_n - \tilde{\kappa_x} V_t & \text{side 1} \\ -\tilde{\kappa_y} V_n + \tilde{\kappa_x} V_t & \text{side 2} \\ \tilde{\kappa_y} V_n + \tilde{\kappa_x} V_t & \text{side 3} \\ \tilde{\kappa_y} V_n - \tilde{\kappa_x} V_t & \text{side 4} \end{cases} \tag{A.10}$$

Circulation Correction

A compressible potential vortex solution is added as a perturbation to the free-stream velocity [143]

$$u_f = u_\infty + \frac{\beta\Gamma \sin(\theta)}{2\pi r[1 - M_\infty^2 \sin^2(\theta - \alpha)]} \tag{A.11}$$

$$v_f = v_\infty - \frac{\beta\Gamma \cos(\theta)}{2\pi r[1 - M_\infty^2 \sin^2(\theta - \alpha)]} \tag{A.12}$$

where $\Gamma = \frac{1}{2} M_\infty\, c\, C_\mathrm{L}$, $\beta = \sqrt{1 - M_\infty^2}$, and r and θ are the polar coordinates to the point of application at the far-field boundary relative to the quarter-chord point on the airfoil chord line. The speed of sound is also corrected to enforce constant free-stream enthalpy as follows:

$$a_f^2 = (\gamma - 1)\left(H_\infty - \frac{1}{2}(u_f^2 + v_f^2)\right) \tag{A.13}$$

Appendix B

GMRES

In order to solve the linear system of equations

$$\mathcal{A}x = b \tag{B.1}$$

the GMRES algorithm finds at every iteration m an approximate solution of the form $x_m \in \{x_0 + K_m\}$, such that the L_2 norm of the residual $r_m = b - \mathcal{A}x_m$ is minimized. The initial guess is denoted by x_0 and K_m is a Krylov subspace given by

$$K_m = \text{span}\{v_1, \mathcal{A}v_1, \mathcal{A}^2 v_1, ..., \mathcal{A}^{m-1} v_1\} \tag{B.2}$$

The vector v_1 is formed from the initial guess

$$v_1 = \frac{r_0}{\|r_0\|_2} = \frac{b - \mathcal{A}x_0}{\|b - \mathcal{A}x_0\|_2} \tag{B.3}$$

For further details see [150].

The SPARSKIT [151] implementation of GMRES is used in conjunction with the ILUK(P) preconditioner. The only modification performed to the GMRES algorithm was to simplify the first iteration of the algorithm since the initial guess was always zero. The following is an outline of the preconditioned GMRES algorithm:

1. *Start*: Choose x_0, compute $r_0 = b - \mathcal{A}x_0$ and $v_1 = r_0/\|r_0\|_2$

2. *Iterate*: For $j = 1, \ldots, m$ do

   Preconditioning: $z_j = \mathcal{M}^{-1} v_j$
   $w_j = \mathcal{A} z_j$
   $h_{i,j} = (w_j, v_i), i = 1, 2, \ldots, j$
   $\hat{v}_{j+1} = w_j - \sum_{i=1}^{j} h_{i,j} v_j$
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$
   $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$

3. *Solve the least-squares problem:*

$$\min_{y_m} ||\beta\, e_1 - \bar{H}_m\, y_m||_2$$

   where $\beta = ||r_0||$, e_1 denotes the first column of an $m \times m$ identity matrix, and $\bar{H}_m$ is a $(m+1) \times m$ upper-Hessenberg matrix containing the $h_{i,j}$ coefficients. Check exit condition.

4. *Form the approximate solution*:

   Apply preconditioner:

$$u_m = \mathcal{M}^{-1} (V_m\, y_m)$$

   where V_m is a $N_\mathrm{F} \times m$ matrix with columns $v_1, \ldots, v_m$.
   Form $x_m = x_0 + u_m$

5. *Restart*:

   Compute $r_m = b - \mathcal{A} x_m$, if satisfied then Stop,
   else $x_0 \leftarrow x_m$ and Goto 1

Note that the GMRES algorithm requires only matrix-vector products. This is indicated in step 2 of the above algorithm, where we require the product $\mathcal{A}z$, or without preconditioning $\mathcal{A}v$. For the Newton–Krylov flow solver, the matrix vector products are approximated using first-order finite differences

$$\mathcal{A} v \simeq \frac{R(\widehat{Q} + hv) - R(\widehat{Q})}{h} \tag{B.4}$$

where h denotes the stepsize. The stepsize is given by

$$h \;=\; \sqrt{\varepsilon_m} \,/\, ||v||_2 \tag{B.5}$$

where ε_m is the value of machine zero, which for the computer hardware used in this work is 2.2×10^{-16}. This strategy is based on the work of Pueyo and Zingg [142]. For the solution of the flow-sensitivity equation, the user can select first- or second-order finite differences. The second-order finite-differences require more computational effort and we find that the gain in accuracy does not have a significant effect on the convergence of the optimization.