Efficient Homotopy Continuation Algorithms with Application to Computational Fluid Dynamics

by

David A. Brown

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

Efficient Homotopy Continuation Algorithms with Application to Computational Fluid Dynamics

David A. Brown
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2016

New homotopy continuation algorithms are developed and applied to a parallel implicit finite-difference Newton-Krylov-Schur external aerodynamic flow solver for the compressible Euler, Navier-Stokes, and Reynolds-averaged Navier-Stokes equations with the Spalart-Allmaras one-equation turbulence model. Many new analysis tools, calculations, and numerical algorithms are presented for the study and design of efficient and robust homotopy continuation algorithms applicable to solving very large and sparse nonlinear systems of equations. Several specific homotopies are presented and studied and a methodology is presented for assessing the suitability of specific homotopies for homotopy continuation.

A new class of homotopy continuation algorithms, referred to as monolithic homotopy continuation algorithms, is developed. These algorithms differ from classical predictor-corrector algorithms by combining the predictor and corrector stages into a single update, significantly reducing the amount of computation and avoiding wasted computational effort resulting from over-solving in the corrector phase. The new algorithms are also simpler from a user perspective, with fewer input parameters, which also improves the user's ability to choose effective parameters on the first flow solve attempt. Conditional convergence is proved analytically and studied numerically for the new algorithms.

The performance of a fully-implicit monolithic homotopy continuation algorithm is evaluated for several inviscid, laminar, and turbulent flows over NACA 0012 airfoils and ONERA M6 wings. The monolithic algorithm is demonstrated to be more efficient than the predictor-corrector algorithm for all applications investigated. It is also demonstrated to be more efficient than the widely-used pseudo-transient continuation algorithm for all inviscid and laminar cases investigated, and good performance scaling with grid refinement is demonstrated for the inviscid cases. Performance is also demonstrated to be better or comparable to the pseudo-transient continuation algorithm for all turbulent cases investigated. Stability of the new algorithms is also investigated.

Acknowledgements

First and foremost I would like to thank my supervisor Professor David Zingg for his constant support and mentor-ship during the past five years. Countless times throughout the PhD when faced with challenging technical problems he has been an invaluable source of advice and has often shown great insight in proposing research objectives and guiding the thesis direction. Most importantly, his exceptional work ethic, integrity, and attention to detail has greatly inspired my own development as a researcher and has had a major impact on the quality of my thesis work.

I would also like to thank my doctoral committee members Professor Clinton Groth, Professor Prasanth Nair, and Professor Z. J. Wang for their efforts with the written thesis document. Their comments and suggestions have with no uncertainty improved the quality of the final written account and I have also appreciated their genuine interest in my work and enjoyed discussing it with them at my final oral examination as well as on other occasions.

Computational aerodynamics is a complex multi-disciplinary subject, requiring knowledge of numerical methods, physics, computers, programming, and often other technical subjects. When my own knowledge was lacking, I often turned to my colleagues, and their assistance has been invaluable. Some specific people that I would like to thank are Michal Osusky for sharing his expertise of the flow solver on many occasions; Howard Buckley for his assistance with many computer-related subjects, such as Fortran 90, MPI, and version control; Scott Northrup for his help on several occasions with questions related to the SciNet system; our system administrators Marc Charest and Boone Tensuda for keeping our computers running (most of the time); David Del Rey Fernández for assisting with the analysis of some of the discrete operators; and Pieter David Boom for assistance with the analysis of the matrix-free monolithic homotopy algorithm presented in the thesis. Over the years, I have also had many relevant technical discussions with David Del Rey Fernández, Jenmy Zhang, Nick Holt, and Ramy Rashad, which contributed to my understanding of many concepts covered in this thesis.

I would also like to acknowledge Chris Lee for providing the 2D inviscid grid and Michal Osusky for providing all turbulent and laminar grids used in this thesis. The 3D inviscid grid was created by Jason Hicken.

Finally, the thesis would not have been possible without the moral support of the friends and colleagues who I met at UTIAS. These individuals have contributed as much to the thesis through their friendship as any of the technical contributions listed above. I am especially grateful to Pakeeza Hafeez, who has in recent years become my greatest distraction as well as my greatest source of inspiration.

Contents

List of Tables

List of Figures

List of Algorithms

List of Symbols

a	sound speed; alt. pseudo-transient continuation parameter
a,b	pseudo-transient continuation parameters used to determine the reference time step
$\mathcal{B}_r$	ball of radius r
c	reference chord length, alt. homotopy curv
$\mathrm{C_D}$	drag coefficient for three-dimensional flow
$\mathrm{C_d}$	drag coefficient for two-dimensional flow
$\mathrm{C_L}$	lift coefficient for three-dimensional flow
$\mathrm{C_l}$	lift coefficient for two-dimensional flow
$\mathcal{D}^{(2)}$,$\mathcal{D}^{(4)}$	dissipation operators
D	number of spatial dimensions
$\mathcal{D}$	operator representing a numerical approximation to a directional derivative
e	energy
E,F,G	inviscid flux operators
E_v,F_v,G	inviscid flux operators
$\mathcal{G}$	homotopy system residual
h	step-length
$\mathcal{H}$	homotopy deformation residual
$\mathcal{H}^*$	dynamic inverse of $\mathcal{H}$
$\mathcal{I}$	identity matrix
J	metric Jacobian
m	number of iterations between preconditioner updates
Ma	Mach number
N	total number of equations in the CFD problem
p	pressure
$\mathbf{q}$	vector of discrete state variables
r	parameter for λ-parametrization
$\mathcal{R}$	flow residual
Re	Reynolds number
s	arclength parameter
t	time; alt. tangent vector
T	temperature
u, v, w	fluid velocity in x,y,z
wu	TauBench work unit

x, y, z	Cartesian spatial coordinates
α	angle of attack
β	parameter appearing in the convergence condition for the dynamic inverse
δ	parameter used in ϵ calculation
ϵ	step size for finite-difference matrix-vector products; alt. pressure sensor
γ	heat capacity, alt. parameter appearing in the monolithic homotopy continuation algorithms
κ	homotopy scaling parameter; alt. curvature
$\kappa^{(2)}$,$\kappa^{(4)}$	dissipation coefficients
λ	homotopy continuation parameter
μ	viscosity; alt. homotopy scaling parameter
μ_{a}	baseline component of homotopy scaling parameter μ
μ_{u}	user-supplied component of homotopy scaling parameter μ
ρ	fluid density
σ	spectral radius
σ_{dlf}	dissipation lumping factor
τ_{l}	linear solver tolerance
$\tilde{\nu}$	turbulence model working variable
ξ, η, ζ	curvilinear coordinates
$\mathbb{C}$	set of complex numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^N$	set of real-valued N-dimensional vectors where $N \in \mathbb{Z}$
$\mathbb{Z}$	set of integers

List of Acronyms

AMVP	Approximate Matrix-Vector Product
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy number
CHC	Convex Homotopy Continuation
CPU	Central Processing Unit
FDMVP	Finite-Difference Matrix-Vector Product
GCROT(m, k)	Generalized Conjugate Residual with inner Orthogonalization and outer Truncation
GHC	Global Homotopy Continuation
(F)GMRES	(Flexible) Generalized Minimal RESidual
ILU(p)	Incomplete Lower-Upper factorization with fill level p
INP	Inexact Newton Phase
MFMH	Matrix-Free Monolithic Homotopy
MH	Monolithic Homotopy
ODE	Ordinary Differential Equation
PC	Predictor-Corrector
PL	Piecewise-Linear
PTC	Pseudo-Transient Continuation
RANS	Reynolds Averaged Navier-Stokes
RK4	Fourth-Order Runge-Kutta
SA	Spalart Allmaras turbulence model
SAT	Simultaneous Approximation Term
SBP	Summation By Parts

Notation

Scalar-valued variables and constants are generally italicized. They can be Roman or Greek, upper- or lower-case, and can contain subscripts or superscripts. Some examples are γ, λ_k, a, C_κ. Vector-valued variables are bold-faced and non-italic. For example: $\mathbf{q}$, $\mathbf{u}$, $\mathbf{v}$. Operators, both linear and nonlinear, are usually typeset with caligraphic font. Some examples are $\mathcal{A}$, $\mathcal{R}$, $\mathcal{H}$.

Superscripts, when not indicating exponentiation, are placed in parentheses and generally refer to the iteration index of a fixed point method such as Newton's method or pseudo-transient continuation. Subscripts usually refer to the iterate of the homotopy curve tracing algorithm, with some exceptions which will be clear from the context. In the case where a specific component of a vector-valued variable is referenced, a subscript in square brackets is used. Variables may take any combination of these subscripts. For example, $\mathbf{q}_k^{(n)}$ refers to the n-th Newton iterate at the k-th homotopy curve-tracing step, whereas $\mathbf{u}_{[i]}$ is the i-th component of the vector $\mathbf{u}$. When not applied in the context of an iterative algorithm, subscripts can indicate partial differentiation. For example, $\mathbf{q}_x(x,y) \equiv \frac{\partial}{\partial x}\mathbf{q}(x,y)$. In some cases, the subscript is used simply to distinguish a variable from similar variables. For example, S_r and S_c are row- and column-scaling operators, respectively. The usage of the subscript will either be clear from context or explained in the text.

The operator Δ is interpreted as a forward difference operator. The differencing is applied to whichever index is present and should be clear from the context. As an example, $\Delta\mathbf{q}^{(n)} \equiv \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)}$.

Differentiation is sometimes indicated with dots above the dependent variable and is taken with respect to the current parametrization. For example, $\dot{\mathbf{q}}(s) \equiv \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{q}(s)$ and $\ddot{\mathbf{q}}(s) \equiv \frac{\mathrm{d}^2}{\mathrm{d}s^2}\mathbf{q}(s)$. For higher-order derivatives, or for general derivatives of order n, the unfortunate notation $\overset{(n)}{\mathbf{q}}(s) \equiv \frac{\mathrm{d}^n}{\mathrm{d}s^n}\mathbf{q}(s)$ is used instead of the more common bracketed superscript notation in order to avoid confusion with other uses of the bracketed superscript.

Chapter 1

Introduction

1.1 Computational Aerodynamics

Accurate estimates of lift and drag coefficients for wings under different operating conditions are important in the design of wing shapes. Due to the complexity and nonlinearity of the partial differential equations governing fluid flow, closed form solutions to the flow field around a wing have not been obtained, and there is no evidence to suggest that such solutions will be obtained in the foreseeable future. In the absence of analytical solutions, quantities of engineering relevance, such as lift and drag, can be estimated based on experimental wind tunnel testing. This process can be expensive and time-consuming, especially if many wing shapes are to be investigated. With the ever-increasing power and availability of high-performance computing systems, computational fluid dynamics (CFD) has become an increasingly common alternative for obtaining reliable estimates of any relevant engineering quantities. These numerical calculations can usually be performed in less time and at less monetary cost than could be achieved by wind-tunnel experiments.

Most CFD methodologies can be described as follows. Consider a continuous finite physical domain representing the region in which the fluid flow is to be modelled. This continuous domain is approximated with a finite number of discrete points, and this set of points is referred to as a *grid* or a *mesh*. Points on the grid are referred to as *nodes*. The state variables (e.g. density, pressure, and velocity components for compressible flow) are interpreted discretely by considering only values at each grid point. The partial differential equations of interest, the compressible Navier-Stokes equations for example, are then represented discretely at each set of points. The vast array of discretization strategies that exist in the literature are far too broad to encompass in the scope of this thesis, but generally fall into three categories: finite difference, finite volume, and finite element. The expression for the discrete derivatives of any order at any spatial point will contain state values at neighbouring nodes, and hence the discrete flow equations form a fully coupled system of algebraic equations, which is nonlinear since the partial differential equations from which they are derived are nonlinear. The vector of discrete flow equations, when evaluated at a given value of the state, is referred to as the *flow residual*.

Though the amount of computational effort required to accurately solve the flow equations numerically is by most standards considered very high, the cost has become increasingly more manageable due to rapid and ongoing advances in computer technology over past decades as well as major research efforts over the years in developing more efficient discretization techniques and also more efficient numerical

algorithms for solving nonlinear systems of equations.

While the cost of CFD is made more manageable by improvements in computer architecture, the demand for CFD is also increasing, as are the size and complexity of the systems to which CFD is applied, and the accuracy to which a solution is desired. One might argue that the demand for CFD has increased consistent with the CPU power available and will continue to increase as faster and more sophisticated computer systems become available. As CFD continues to be pushed to the limit of the available computational resources, there will continue to be an incentive to investigate new methodologies to reduce the computational time needed to obtain CFD flow solutions.

One way to improve the accuracy of a CFD discretization is to refine the mesh - that is, to use more nodes in the computational domain. However, the CPU cost of solving the discrete flow equations, unless combined with multi-grid, is expected to increase super-linearly with the number of grid points. There are more computationally efficient techniques for achieving a more accurate solution than simply refining the grid uniformly. For example, the mesh could be refined locally in regions where the refinement is needed. This process can be automatic; see, for example, Nemec et al. [119]. This can be especially useful for unsteady problems where the level of grid refinement needed to capture the flow physics can change locally with time.

Currently, higher-order accurate spatial discretizations are widely studied by the CFD community for the potential efficiency improvement over the usual second-order accurate discretizations. The order of accuracy is the rate at which the error reduces as the grid spacing is reduced. For example, if the error is proportional to the grid spacing to the power p, then the scheme is said to be pth order accurate. The theoretical order of the scheme is generally not observed unless the grid spacing is sufficiently small. Higher order methods are currently a major area of CFD research, where high order generally refers to orders greater than two. Some important examples of higher-order schemes are Harten et al. [57], Lele [93], Bassi and Rebay [8], Liu et al. [97], Huynh [70], and Del Rey Fernández et al. [29]. A recent paper by Wang et al. [167] includes more literature review and studies comparing the relative efficiency of different higher-order methods.

The reason to study grid adaptation or higher-order methods is because they can potentially lead to *efficiency* improvements in the sense that a solution can potentially be obtained with the same accuracy in less CPU time. Similarly, any improvements to the performance of the numerical algorithm for solving the nonlinear system of equations can be seen as an improvement to the efficiency of the CFD algorithm, and are not mutually exclusive of grid adaptation or higher-order methods.

Perhaps the most well-known convergence acceleration algorithm is multigrid, which systematically uses sets of coarser grids to accelerate the convergence of iterative schemes [142]. Originally conceived for explicit methods [5, 72, 89, 109, 122, 169], convergence acceleration can also be achieved for implicit schemes [75, 110]. However, the effectiveness of multigrid can depend on the flow conditions. For example, it is often less effective for transonic cases, as discussed by Eriksson and Rizzi [42].

Jespersen and Buning [76] developed a convergence acceleration procedure aimed at improving the convergence of slowly converging explicit flow solves. Their method uses eigenvalue and eigenvector estimates with an extrapolation procedure to estimate the converged solution, in effect eliminating the extreme eigenvalues responsible for the slow convergence. Hafez et al. [55] investigated this procedure, along with a similar procedure, for accelerating the convergence of some transonic cases. Dagan [27] later applied this convergence acceleration technique to implicit solvers. Further contributions to the methodology for explicit flow solvers were provided by Eyi [43].

The multigrid and eigenvalue convergence acceleration methods can be applied to an iterative algorithm for solving nonlinear systems of equations. The iterative algorithm itself can be designed efficiently or inefficiently. In addition, the algorithm may fail to converge to the solution in some cases. How reliably the iterative algorithm converges to the flow solution is referred to as *robustness* and is equally important as the algorithm efficiency. This is especially true when using a higher-order discretization, since the discrete flow equations tend to be more difficult to solve than their second-order counterparts.

A popular fixed-point algorithm for solving nonlinear algebraic systems of equations is Newton's method because it can give quadratic convergence for reasonable CPU cost. However, Newton's method is unlikely to converge unless a suitable starting iterate is provided. Obtaining a suitable starting point which leads to convergence of the algorithm is known as *globalization* and is generally performed with a more reliable but slower converging algorithm known as a numerical *continuation* algorithm.

Some examples of continuation algorithms are line search [40], trust region [114], mesh sequencing [25, 82], pseudo-transient continuation (PTC) [78], and homotopy methods [1]. More description and examples of literature concerning these methods can be found in the review paper by Knoll and Keyes [81]. Of particular interest to us is the PTC method. This method imitates physical time marching. It is simple to implement and to use, greatly improves the conditioning of the Jacobian in the continuation phase, is usually quite robust, and can give q-super-linear convergence [78]. As a result, PTC has seen widespread use in modern CFD practice. If an alternative continuation algorithm is proposed, it is necessary to demonstrate some performance benefit of the new algorithm over PTC, in order to show that the new algorithm is of practical value.

1.2 Homotopy Continuation

The premise of homotopy continuation is that a second system of equations is defined, called the *homotopy system*, which exists in the same real vector space as the system of equations of interest, which in the present context is the discrete flow equations. The homotopy system should either have a known solution or be easy to solve and, when added to the flow residual, should improve the stability and conditioning of the linear system. The solution to this system of equations is then deformed to the solution to the discrete flow equations. If the deformation is continuous then it is called a *homotopy*. This deformation can also be interpreted as a curve existing in the same real vector-space as the discrete flow equations. Homotopy continuation algorithms are based on approximately tracing this curve from the solution to the homotopy system to the solution to the discrete flow equations.

Since the homotopy can be constructed to have a continuous analogue which is essentially independent of mesh refinement, the homotopy continuation method has the potential for linear performance scaling with mesh refinement, whereas the step size for the pseudo-transient method is limited by the mesh spacing through the Courant-Friedrichs-Lewy (CFL) number [142] and so CPU time is expected to increase super-linearly. Additionally, the homotopy should remain fundamentally unchanged if a higher-order scheme is employed, potentially avoiding many of the stability issues associated with time-marching for higher order problems. With increased demand on CFD for solving larger and more complex problems, and the growing application of higher-order accurate discretization schemes, this is a highly appropriate time to be investigating the potential benefits of homotopy continuation.

Some historical context of homotopy continuation methods is given by Allgower and Georg [1]: "Their

use can be traced back at least to such venerated works as those of Poincaré [133] (1881), Klein [80] (1882), and Bernstein [10] (1910). Leray and Schauder [96] (1934) refined the tool and presented it as a global result in topology viz. the homotopy invariance of degree." Lahaye [85] (1934) is perhaps the first to use homotopy methods to solve nonlinear scalar equations, as well as nonlinear systems of equations [86] (1948). Ortega and Rheinboldt [126] (1970) provide a framework for the implementation of homotopy methods as numerical algorithms. Contributions and refinements to these algorithms over the next two decades are presented in Allgower and Georg [1] (1990). Allgower and Georg [2] provide an extensive literature review and bibliography pertaining to homotopy continuation methods prior to 1992.

Several researchers have applied homotopy continuation to CFD problems. Carey and Krishnan [21] investigated using the Reynolds number as the continuation parameter to solve a simple incompressible viscous lid-driven cavity problem at high Reynolds number. Wales et al. [166] (2012) have applied homotopy continuation to two-dimensional turbulent flow around a NACA 0012 airfoil. By treating the angle of attack as the continuation parameter, the authors were able to acquire flow solutions for dynamically unstable flow at high Reynolds number and high angle of attack. These flow solutions may have been difficult or impossible to achieve using PTC since PTC is not globally or even locally convergent for dynamically unstable solutions.

Some recent studies have also been performed to study the multiplicity of solutions which are not necessarily unstable. Jameson et al. [74] (2014) studied the multiplicity of inviscid flow solutions over several airfoils by sweeping the Mach number in some cases and angle of attack in other cases. While the Mach number sweep was not actually performed using a homotopy method, the solution sweep does result in a homotopy and could have been formed using homotopy continuation. The angle of attack sweep appears to have been performed by homotopy continuation using the angle of attack as the continuation parameter, though the authors do not identify it as such. A similar study has been performed by the same authors which included some wing geometries [132] (2014). Lee et al. [91] (2015) later performed a similar study for an airfoil exhibiting multiple solutions using a Mach number sweep. In this case the solution arcs were determined using homotopy continuation with the Mach number as the continuation parameter.

There has also been some research interest in using the critical point detection properties of homotopy methods in CFD; for example, Riley and Winters [145], Sanchez et al. [151], Winters [171], and Winters and Cliffe [172]. As an example of some of this work, Riley and Winters [145] (1990) studied the problem of rotating a side-heated porous cavity to a bottom-heated porous cavity. Since the side-heated problem has a unique solution but the bottom-heated problem has multiple solutions, bifurcation points are encountered when deforming the solution to the side-heated problem to the solution to the bottom-heated problem. The objective of this analysis was to gain insight into the physical mechanism allowing a system with multiple solutions to evolve into a system with a unique solution. The angle of rotation was treated as the continuation parameter for this homotopy. Many examples where homotopy continuation has been used to study systems with multiple solutions exist outside of CFD. Some examples are the study of resistive circuits [92, 164] and stochastic games [11]. The book by Morgan [115] deals exclusively with finding all roots of polynomials using homotopy methods.

In contrast to the objectives of previous researchers, the aim of this thesis is to develop homotopy continuation as an efficient and robust globalization methodology for modern Newton-Krylov CFD flow solvers. Hicken and Zingg [64] applied homotopy continuation in two ways: by varying the boundary

conditions and using a second-difference artificial dissipation operator. It is often necessary to apply some form of numerical dissipation to the discrete flow residual for stability of the numerical scheme [99]. The numerical dissipation scheme used by Hicken and Zingg [64] is based on the artificial dissipation operators developed by Jameson et al. [73] and later refined by Pulliam [139].

The dissipation operator used by Hicken and Zingg [64] is a second-difference Laplacian-like operator with first-order spatial accuracy and is not applied to second-order accurate discretizations, other than locally near shocks, because it would reduce the order of accuracy of the scheme to first order. However, the operator has a highly stabilizing affect on the flow residual and has a narrower stencil than the fourth-difference dissipation operator that would normally be applied to a second-order code. Later in this thesis, when preconditioning methods are discussed, it will become clear why these properties make the second-difference operator a more logical choice for the homotopy system than the fourth-difference operator.

The homotopy continuation method of Hicken et al. [61], referred to by the authors as *dissipation-based continuation*, can be described as follows. The flow residual is constructed as usual but is augmented with an additional numerical dissipation operator. Since, relative to the unmodified discrete flow equations, the preconditioned Jacobian matrix of the augmented system is much better conditioned and has its eigenvalues shifted well to the right (or left, depending on convention) of the imaginary axis, the augmented system of equations is easier to solve with Newton's method than the original and most common iterative linear solvers, including Krylov solver, are expected to converge in fewer iterations. This problem is solved to some tolerance using a Newton-Krylov approach. The amount of dissipation is reduced and the next subproblem is solved, using the solution to the recent subproblem to initialize. This process is repeated until eventually, after solving several subproblems, the additional dissipation is removed entirely and the flow equations are solved using a Newton-Krylov method.

The dissipation-based continuation algorithm was investigated by Hicken et al. [61] for Euler, laminar Navier-Stokes, and Reynolds-averaged Navier-Stokes (RANS) cases for a Newton-Krylov finite-difference flow solver. With the exception of some low Mach number laminar cases, all of the cases were two-dimensional flows. The algorithm was found to be competitive with PTC for all three flow types, particularly for the inviscid case, though we have found that the performance of PTC could have been improved considerably by tuning some of the parameters. Recently, Hao et al. [56] have applied a homotopy continuation method to a high-order WENO discretization, apparently independently of the work of Hicken et al. Yu and Wang [174] have recently performed some preliminary investigation of homotopy continuation methods for applications to a discrete Galerkin flow solver based on the work of Hicken et al. [61] and Hicken and Zingg [64], as well as some of our preliminary work [16].

1.3 Thesis Objectives

The goal of the current research programme is to develop efficient and robust homotopy continuation algorithms with superior performance relative to other globalization methods currently employed by the CFD community. However, we do not describe this as the thesis objective.

Though there is a wide array of homotopy literature available, it has primarily been applied to very small scale problems where the numerical efficiency has not been of major concern. In particular, there has been little consideration for applications to large sparse systems. For example, many homotopy

researchers have assumed that matrix inverses can readily be formed analytically or have assumed that a $\mathcal{QR}$ factorization can easily be computed. Many researchers who have used homotopy continuation to solve problems on a larger scale have used simple algorithms without developing sophisticated or efficient tools since the application of homotopy in these cases has usually been for the study of systems with multiple or unstable solutions and it was not important that the algorithm perform competitively with other globalization methods. Much of the research effort in the preparation of this thesis was simply in developing the capability of performing many of the calculations that exist in the homotopy literature in a way that is both possible and computationally efficient for steady CFD problems.

Though it is important to compare the efficiency of the new algorithms to PTC, which is accomplished in Chapter 8, the performance data reported in this comparison are subject to many external factors. Some obvious factors affecting performance are the specific processors and compilers used and the coding efficiency of the implementation. Moreover, relative performance may vary by application, the equations being solved, the discretization scheme, mesh refinement, choice of linear solver or preconditioner, linear system scaling, etc. These factors are all active research areas; it is possible that the most efficient linear solver for our applications may not have been developed yet. As such, it is not sufficient to give a quantitative assessment of algorithm performance specifically limited to the flow solver technologies and computer hardware that we currently have access to; it is more important to develop an understanding of what aspects of the algorithms affect performance the most and in what way. To this end, we have included throughout the thesis many studies and analyses aimed at developing an understanding of the performance of the algorithms - how the algorithm parameters and design decisions in constructing the algorithms and homotopies will affect convergence times and robustness.

In anticipation that the work presented in this thesis will be subject to improvements and adaptations made by future researchers, and in consideration of the current state of flux of the technologies currently at our disposal for CFD analysis, we now state our main objectives in writing this thesis:

- Construct realizable homotopies which are applicable to CFD problems;
- Develop, present, and study efficient tools for constructing and analyzing homotopy continuation algorithms;
- Develop and assess metrics for analyzing the suitability of a homotopy for homotopy continuation;
- Characterize the homotopies developed for different applications, flow conditions, and grid properties such as refinement and topology;
- Develop and study new homotopy continuation algorithms which are more efficient than those in the literature;
- Develop a quantitative understanding of how various tools, parameters, and design decisions affect the performance of the homotopy continuation algorithms;
- Quantitatively assess the performance of the new tools and algorithms; and
- Quantitatively compare the new homotopy continuation algorithms with PTC using a modern CFD flow solver and modern computer hardware.

Chapter 2

Governing Equations and Spatial Discretization

The equations studied are the compressible Euler equations, the Navier-Stokes equations, and the Reynolds-averaged Navier-Stokes (RANS) equations with Spalart-Allmaras (SA) turbulence model. The flows considered are external aerodynamic flows around three-dimensional wings or two-dimensional airfoils. The governing equations are discretized spatially using a finite-difference discretization with summation-by-parts (SBP) operators and simultaneous approximation terms (SATs) to weakly enforce the boundary conditions on the domain boundaries and to couple the system across block interfaces [22, 30, 44, 83, 125, 159, 160]. All spatial derivatives are discretized with second-order accuracy except for the advection term of the SA model for which a first-order upwinding scheme is used. The discretization for the Euler equations was implemented by Hicken and Zingg [63] and Hicken [60] and extended to include the viscous terms and RANS equations by Osusky and Zingg [130] and Osusky [128].

2.1 Euler and Navier-Stokes Equations

The three-dimensional Euler and Navier-Stokes equations are given in non-dimensional form by

$$\partial_t q + \partial_x E + \partial_y F + \partial_z G = 0 \text{ (Euler)}, \tag{2.1}$$

$$\partial_t q + \partial_x E + \partial_y F + \partial_z G = \frac{1}{\mathrm{Re}} \left(\partial_x E_v + \partial_y F_v + \partial_z G_v \right) \text{ (Navier} - \text{Stokes)}, \tag{2.2}$$

$$\mathrm{Re} = \frac{\rho_\infty a_\infty l}{\mu_\infty} \text{ (Reynolds number)}, \tag{2.3}$$

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\left(e+p\right) \end{bmatrix}, \; F = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\left(e+p\right) \end{bmatrix}, \; G = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w\left(e+p\right) \end{bmatrix} \text{ (Inviscid fluxes)}, \tag{2.4}$$

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \; F_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \; G_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix} \text{ (Viscous fluxes)}, \tag{2.5}$$

$$q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \text{ (State vector)}. \tag{2.6}$$

In the above equations: ρ is the density, a is the speed of sound, e is the energy, p is the pressure, l is the mean chord length, μ is the viscosity, $\mathbf{u} = (u, v, w)$ are the Cartesian velocity components, $\tau = \tau(u, v, w, \mu)$ are the viscous stresses, Re is the Reynolds number, and $\mu = \mu(a)$ is the viscosity and is given by Sutherland's law:

$$\mu = \frac{a^3 (1 + S^*/T_\infty)}{a^2 + S^*/T_\infty}, \tag{2.7}$$

where $S^* = 198.6°\text{R}$ is Sutherland's constant and the subscript ∞ indicates the free-stream value of a quantity. Assuming that the flow behaves as an ideal gas, the pressure variable can be written in terms of energy and velocity:

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right), \tag{2.8}$$

where $\gamma \in \mathbb{R}$ is the heat capacity ratio and is taken as 1.4 for air. This additional algebraic equation is used to reduce the effective number of variables to five for the Euler and Navier-Stokes equations. The turbulent viscosity $\mu_t = \mu_t(\rho, \mu, \tilde{\nu})$, where $\tilde{\nu}$ is the turbulence variable, is added to μ in the case of turbulent flows. Explicit expressions for E_v, F_v, and G_v are omitted here but are given by Osusky and Zingg [130]. The density is non-dimensionalized by ρ_∞, the velocities by a_∞, the viscosity by μ_∞, the temperature by T_∞, and the spatial coordinates by l.

2.2 The Spalart-Allmaras Turbulence Model

Turbulence is by nature an unsteady and chaotic phenomenon that cannot be captured by steady simulation. However, if all that is desired are average values of some functionals such as the lift and drag coefficients $\mathrm{C_L}$ and $\mathrm{C_D}$, then the turbulent fluctuations can be averaged over a period of time to obtain a steady system of equations, known as the Reynolds-Averaged Navier-Stokes (RANS) equations. Estimates for the time-averaged values of these functionals can be calculated from the solution to the RANS equations.

The modeling of the additional terms that arise from the time-averaging process has been the subject of much research over the last century. Many turbulence models and their derivations are provided by Wilcox [170]. The turbulence model used in this thesis is the Spalart-Allmaras [156] turbulence model (SA model) in its original form, which is currently one of the most commonly used models for external aerodynamic flows. This model is classified as a *one-equation* turbulence model because one additional partial differential equation is coupled to the mean-flow equations. The original SA turbulence model is

given by

$$\frac{\partial\tilde{\nu}}{\partial t}+u\frac{\partial\tilde{\nu}}{\partial x}+v\frac{\partial\tilde{\nu}}{\partial y}+w\frac{\partial\tilde{\nu}}{\partial z}=\frac{c_{b1}}{\mathrm{Re}}\left(1-f_{t2}\right)\tilde{S}\tilde{\nu}+\frac{1+c_{b2}}{\sigma\mathrm{Re}}\nabla\cdot\left[\left(\nu+\tilde{\nu}\right)\nabla\tilde{\nu}\right]-\frac{c_{b2}}{\sigma\mathrm{Re}}\left(\nu+\tilde{\nu}\right)\nabla^{2}\tilde{\nu}$$
$$-\frac{1}{\mathrm{Re}}\left[c_{w1}f_{w}-\frac{c_{b1}}{\kappa^{2}}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^{2}+\mathrm{Re}f_{t1}\Delta U^{2}, \tag{2.9}$$

where ν is the kinematic viscosity, and $\tilde{\nu}$ is referred to simply as the *turbulence variable*. This equation is highly nonlinear and many of the terms above are functions of $\tilde{\nu}$ or other state variables. More details of the SA model and the discretization used in this thesis, including boundary conditions, are available from Osusky and Zingg [130] or Osusky [128].

2.3 Transformation to a Computational Coordinate System

The discrete flow equations are evaluated on a structured time-independent mesh in a Cartesian coordinate system (x, y, z). Rather than discretize the flow equations directly on the physical coordinate system, the flow equations are discretized on a much simpler "computational" coordinate system and then transformed to physical space using a coordinate transformation. The computational coordinate system is denoted:

$$\xi=\xi\left(x,y,z\right),\ \eta=\eta\left(x,y,z\right),\ \zeta=\zeta\left(x,y,z\right). \tag{2.10}$$

The coordinates (ξ, η, ζ) are orthogonal and satisfy $\Delta\xi = \Delta\eta = \Delta\zeta = 1$ everywhere in the domain. This simplifies the discretization process considerably but makes the basic equations more complicated. The Navier-Stokes equations in transformed coordinates are given by

$$\frac{1}{\mathrm{Re}}\left(\partial_{\xi}\hat{E}_{v}+\partial_{\eta}\hat{F}_{v}+\partial_{\zeta}\hat{G}_{v}\right)=\partial_{t}\hat{\mathbf{q}}+\partial_{\xi}\hat{E}+\partial_{\eta}\hat{F}+\partial_{\zeta}\hat{G}, \tag{2.11}$$

$$\hat{\mathbf{q}}=\mathcal{J}^{-1}\mathbf{q},$$

$$\hat{E}=\mathcal{J}^{-1}\left(\xi_{x}E+\xi_{y}F+\xi_{z}G\right),\ \hat{F}=\mathcal{J}^{-1}\left(\eta_{x}E+\eta_{y}F+\eta_{z}G\right),\ \hat{G}=\mathcal{J}^{-1}\left(\zeta_{x}E+\zeta_{y}F+\zeta_{z}G\right),$$

$$\hat{E}_{v}=\mathcal{J}^{-1}\left(\xi_{x}E_{v}+\xi_{y}F_{v}+\xi_{z}G_{v}\right),\ \hat{F}_{v}=\mathcal{J}^{-1}\left(\eta_{x}E_{v}+\eta_{y}F_{v}+\eta_{z}G_{v}\right),\ \hat{G}_{v}=\mathcal{J}^{-1}\left(\zeta_{x}E_{v}+\zeta_{y}F_{v}+\zeta_{z}G_{v}\right),$$

$$\mathcal{J}=\left(x_{\xi}y_{\eta}z_{\zeta}+y_{\xi}z_{\eta}x_{\zeta}+z_{\xi}x_{\eta}y_{\zeta}-x_{\xi}z_{\eta}y_{\zeta}-y_{\xi}x_{\eta}z_{\zeta}-z_{\xi}y_{\eta}x_{\zeta}\right)^{-1}\ \text{(Metric Jacobian)}. \tag{2.12}$$

2.4 Discretization using Summation-by-Parts Operators

Unless otherwise specified, all computations presented in this thesis were performed using second-order accurate SBP operators to discretize the spatial derivatives. This does not apply to the advection term of the SA model equation, for which a first-order upwinding scheme is used for stability. The second-order accurate SBP operator for the first derivative is expressed as

$$D_{1}=H^{-1}\Theta, \tag{2.13}$$

$$H = h \begin{bmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{bmatrix}, \; \Theta = \frac{1}{2} \begin{bmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{bmatrix},$$

where h is the step size in the appropriate spatial direction. Since a computational coordinate system is used for the discretization, h is identically equal to 1.

The operator D_1 represents a discrete derivative in a given coordinate direction. For example,

$$\partial_\xi \hat{E} \rightarrow D_{1\xi} \hat{E}. \tag{2.14}$$

This is the SBP operator used for the inviscid flux terms. The second derivatives are similarly modeled using second-order accurate compact SBP operators. Explicit expressions for all SBP operators used in the discretization are given by Osusky [128].

2.5 Boundary Condition Enforcement using Simultaneous Approximation Terms

SBP operators such as D_1 are applied on each block in the computational domain. Boundary conditions are imposed on all block sides, whether domain boundaries or block interfaces, using SATs. In contrast to many boundary condition treatments, the SAT boundary enforcement does not result in any additional equations but instead takes the form of an additional term (called the SAT) which is added to the residual at each boundary node. Since the boundary condition does not need to be satisfied exactly using this method, the boundary conditions are said to be enforced *weakly*.

The SAT approach minimizes the amount of information that needs to be communicated between processors when the equations are parallelized on a distributed memory architecture using a message passing interface (MPI). Additionally, the fact that this discretization does not require the formation of derivatives across block interfaces reduces the continuity requirement for meshes at interfaces, simplifying mesh generation for complex geometries. Only C^0 continuity is necessary for grid lines at interfaces, facilitating grid construction.

The SATs presented in this section are for the inviscid flux terms only; additional SATs are applied when discretizing the Navier-Stokes or RANS-SA equations. The inviscid flux SATs are given by

$$\sigma_{\text{inv}}(\mathbf{q})^+ = -H_{\text{b}}^{-1} \mathcal{J}^{-1} \hat{A}_\xi^+ (\mathbf{q}_{\text{b}} - \mathbf{q}_{\text{tar}}), \text{ (high side SAT)} \tag{2.15}$$

$$\sigma_{\text{inv}}(\mathbf{q})^- = -H_{\text{b}}^{-1} \mathcal{J}^{-1} \hat{A}_\xi^- (\mathbf{q}_{\text{b}} - \mathbf{q}_{\text{tar}}), \text{ (low side SAT)} \tag{2.16}$$

$$\hat{A}_\xi^+ = \frac{\hat{A}_\xi + |\hat{A}_\xi|}{2}, \; \hat{A}_\xi^- = \frac{\hat{A}_\xi - |\hat{A}_\xi|}{2}, \; \hat{A}_\xi = \frac{\partial \hat{E}}{\partial (\mathcal{J}^{-1}\mathbf{q})},$$

where $\mathbf{q}_{\text{b}}$ is the state components at a boundary node, $\mathbf{q}_{\text{tar}}$ is the target value of $\mathbf{q}_{\text{b}}$ at that node, and H_{b} is the boundary value of H and is equal to $\frac{1}{2}$. When the SAT is applied to a block interface, $\hat{A}$ is evaluated at the average of the states at the two coincident interface nodes. The term $|\hat{A}|$ is defined as

follows:

$$|\hat{A}| = X|\hat{\Lambda}|X^{-1}, \tag{2.17}$$

$$|\hat{\Lambda}| = \begin{bmatrix} \hat{\lambda}_1 & 0 & 0 & 0 & 0 \\ 0 & \hat{\lambda}_2 & 0 & 0 & 0 \\ 0 & 0 & \hat{\lambda}_3 & 0 & 0 \\ 0 & 0 & 0 & \hat{\lambda}_4 & 0 \\ 0 & 0 & 0 & 0 & \hat{\lambda}_5 \end{bmatrix},$$

$$\begin{aligned} \hat{\lambda}_1 &= \max\left(\left|U_n + a\sqrt{\xi_x^2 + \eta_y^2 + \zeta_z^2}\right|, V_n\left(|U_n| + a\sqrt{\xi_x^2 + \eta_y^2 + \zeta_z^2}\right)\right), \\ \hat{\lambda}_2 &= \max\left(\left|U_n - a\sqrt{\xi_x^2 + \eta_y^2 + \zeta_z^2}\right|, V_n\left(|U_n| + a\right)\right), \\ \hat{\lambda}_3 = \hat{\lambda}_4 = \hat{\lambda}_5 &= \max\left(|U_n|, V_l\left(|U_n| + a\sqrt{\xi_x^2 + \eta_y^2 + \zeta_z^2}\right)\right), \end{aligned}$$

$$U_n = \xi_x u + \eta_y v + \zeta_z w.$$

For subsonic flows $V_n = \frac{1}{40}$, and for transonic flows $V_n = \frac{1}{4}$. The constant V_l is fixed at $\frac{1}{40}$ for all flows.

Nodes at interfaces are treated as two spatially coincident nodes: one corresponding to the current block and one corresponding to the adjacent block. The state values at a boundary point on the current block and a neighbouring block are considered as different variables. The SATs on a block interface of the current block weakly enforce equality between the state values at the coincident nodes and a similar expression exists on the neighbouring block. Though this weak enforcement can result in some discontinuity at block interfaces, the overall expression is conservative and stable. Since these interface SATs are the only terms on the current block containing state values at neighbouring blocks in the expression, these are the only terms in the discretization that require inter-processor communication when the blocks are distributed across multiple CPUs.

The value of $\mathbf{q}_{\text{tar}}$ depends on which block boundary the SAT is applied. The four types of block boundaries are solid surfaces, symmetry planes, far-field boundaries, and block interfaces. For solid surface boundaries, $\mathbf{q}_{\text{tar}}$ is set to enforce the condition that the velocity normal to the surface is zero. For steady inviscid flows, Hicken [60] indicates that an enthalpy condition can also be enforced. However, this condition is not enforced in the current work. For viscous flows, in addition to the inviscid SATs, zero velocity is also enforced on all solid surfaces and the wall is assumed to be adiabatic (zero temperature gradient in the direction normal to the surface). If the flow is turbulent, the turbulence variable $\tilde{\nu}$ is also enforced as zero.

For symmetry planes, the condition of zero normal velocity is enforced for all flows. If the flow is viscous, zero gradients are also enforced normal to the boundary for density, the non-normal velocity components, and pressure. If the flow is turbulent, the gradient of $\tilde{\nu}$ is also enforced as zero.

For far-field (free-stream) conditions, $\mathbf{q}_{\text{tar}}$ is set to the far-field values. This applies on all far-field boundaries, including both upstream and downstream boundaries. If the flow is viscous then an additional zero gradient condition is enforced for the viscous fluxes. If the flow is also turbulent then an additional zero gradient condition is enforced for $\tilde{\nu}$.

For interfaces, $\mathbf{q}_{\text{tar}}$ is set to the value of $\mathbf{q}$ at the coincident node on the neighbouring block. If the flow is viscous then a similar condition is enforced for the viscous fluxes. For turbulent flows, a zero gradient condition is additionally enforced for $\tilde{\nu}$.

Specific details of the viscous and SA model SATs are given by Osusky and Zingg [130] and Osusky [128].

2.6 Artificial Dissipation

Artificial dissipation is needed to limit the production of high frequency modes when attempting to solve nonlinear convection-dominated problems using a time-marching method. The scalar dissipation model developed by Jameson et al. [73] and refined by Pulliam [139] is used for all flow solves in this thesis. While artificial dissipation introduces some numerical error into the solution, only a small amount of dissipation is usually needed to stabilize the numerical algorithm. The matrix dissipation model of Swanson and Turkel [161] is an alternative dissipation model which introduces less error but can lead to a more expensive and less robust flow solution algorithm. Both dissipation models are described by Pulliam and Zingg [142].

Since dissipation operators are essentially of the form of an undivided difference operator multiplied by the grid spacing to some power, the contribution from any dissipation operator vanishes in the limit as the grid spacing vanishes, so the error introduced by the dissipation operator is reduced on finer grids. Dissipation must be an even difference and will vanish at a rate one order less. The second-difference dissipation operator, for example, will have a first-order convergence rate, meaning that the error is proportional to the grid spacing. Using this dissipation operator will reduce the order of accuracy of a second-order method to first-order. Since the dissipation operator must be an even difference, a fourth-difference operator (which has a third-order convergence rate) must be used for second-order accurate codes if the convergence rate is to be preserved.

For simplicity, the dissipation operators given below are for a one-dimensional scalar problem. The extension to multiple variables is easily made because the same dissipation operator is used for each flow variable. When working in more than one spatial dimension, an additional operator of the same form is applied in each additional spatial direction. The operators are applied to the state variable and added to the discrete flow residual.

In this section, the subscripts will denote the grid index in the ξ-direction. The fourth difference dissipation operator in the ξ-direction is denoted $\mathcal{D}_\xi^{(4)} : \mathbb{R}^N \to \mathbb{R}^N$. The total dissipation applied by this operator is

$$\mathcal{D}_\xi^{(4)}(\mathbf{q}) = \Delta_\xi^T C_\xi^{(4)} \Delta_\xi B \Delta_\xi^T \Delta_\xi \mathbf{q} \tag{2.18}$$

$$\Delta_\xi = \begin{bmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & 0 \end{bmatrix},$$

$$C_\xi^{(4)} = \begin{bmatrix} \kappa^{(4)}\sigma_{\left[\frac{1}{2}\right]} & & & \\ & \kappa^{(4)}\sigma_{\left[\frac{3}{2}\right]} & & \\ & & \ddots & \\ & & & \kappa^{(4)}\sigma_{\left[N-\frac{3}{2}\right]} \end{bmatrix},$$

$$\Delta_\xi \in \mathbb{R}^{(N-1)\times N},\ B \in \mathbb{R}^{N\times N},\ C_\xi^{(4)} \in \mathbb{R}^{(N-1)\times(N-1)},$$

$$\sigma_{\left[i+\frac{1}{2}\right]} = \frac{1}{2}\left(\sigma_{[i]} + \sigma_{[i+1]}\right),\ \ \sigma_{[i]} = \left[\mathcal{J}^{-1}\left(|U_n| + a\sqrt{\xi_x^2 + \eta_y^2 + \zeta_z^2}\right)\right]_{[i]},$$

where $\kappa^{(4)} \in \mathbb{R}$ is a parameter that allows the user to control the amount of dissipation added to the residual.

Dissipation is also added for shock capturing. If the flow solution is expected to include a shock, a second-difference dissipation operator is included in addition to the usual fourth-difference operator and the fourth-difference dissipation coefficient $\kappa^{(4)}$ is modified. The total dissipation added to the flow residual is

$$\mathcal{D}_\xi^{(2)}(\mathbf{q}) + \mathcal{D}_\xi^{(4)}(\mathbf{q}) = \Delta_\xi^T C_\xi^{(2)} \Delta_\xi \mathbf{q} + \Delta_\xi^T C_\xi^{(4)} \Delta_\xi B \Delta_\xi^T \Delta_\xi \mathbf{q}, \tag{2.19}$$

$$C_\xi^{(2)}(\mathbf{q}) = \begin{bmatrix} \kappa^{(2)}\sigma_{\left[\frac{1}{2}\right]}\epsilon_{\left[\frac{1}{2}\right]} & & & \\ & \kappa^{(2)}\sigma_{\left[\frac{3}{2}\right]}\epsilon_{\left[\frac{3}{2}\right]} & & \\ & & \ddots & \\ & & & \kappa^{(2)}\sigma_{\left[N-\frac{3}{2}\right]}\epsilon_{\left[N-\frac{3}{2}\right]} \end{bmatrix},$$

$$\epsilon_{\left[i+\frac{1}{2}\right]} = \frac{1}{2}\left(\epsilon_{[i]} + \epsilon_{[i+1]}\right),$$

$$\epsilon_{[i]} = \frac{1}{4}\left(\Upsilon^*_{[i-1]} + 2\Upsilon^*_{[i]} + \Upsilon^*_{[i+1]}\right),$$

$$\Upsilon^*_{[i]} = \max\left(\Upsilon_{[i-1]}, \Upsilon_{[i]}, \Upsilon_{[i+1]}\right),$$

$$\Upsilon_{[i]} = \frac{\left|p_{[i-1]} - 2p_{[i]} + p_{[i+1]}\right|}{\left|p_{[i-1]} + 2p_{[i]} + p_{[i+1]}\right|},$$

where σ is referred to as the *spectral radius*. The "pressure sensor" ϵ, developed by Jameson et al. [73], is used for shock capturing and will only be active near the shock. This does not produce a "true" shock, which should take the form of a discontinuity in several flow variables with respect to the spatial variables, but instead leads to an approximation of the shock by smearing it over several grid nodes. Values of $\epsilon_{[i]}$, $\Upsilon^*_{[i]}$, and $\Upsilon_{[i]}$ at block boundaries are extrapolated from the adjacent interior nodes. When the second-difference dissipation is included, the fourth-difference dissipation coefficient $\kappa^{(4)}$ is modified at each node according to the following formula:

$$\kappa^{(4)}_{\left[i+\frac{1}{2}\right]} \leftarrow \max\left(0, \kappa^{(4)}_{\left[i+\frac{1}{2}\right]} - \kappa^{(2)}_{\left[i+\frac{1}{2}\right]}\epsilon_{\left[i+\frac{1}{2}\right]}\right).$$

Chapter 3

Jacobian-Free Newton-Krylov-Schur Flow Solver

The flow solver is a parallel implicit Jacobian-free Newton-Krylov-Schur algorithm for three-dimensional compressible external aerodynamic flows. Topics covered in this chapter include the linear solver and preconditioner, the approach for approximating matrix-vector products, linear system scaling, and an overview of the globalization methods studied in the thesis.

3.1 Overview of the Jacobian-Free Newton-Krylov Methodology

The objective of the Newton-Krylov-Schur algorithm is to solve the discrete flow equations, which form a nonlinear algebraic system of equations of the form

$$\mathcal{R}(\mathbf{q}) = \mathbf{0}, \tag{3.1}$$

where $\mathcal{R}(\mathbf{q})$, $\mathcal{R} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is known as the *residual*. The approach taken to converge the flow residual is to use a quasi-Newton algorithm. As discussed in the introduction, Newton's method is unlikely to converge if the starting point $\mathbf{q}_0$ is too distant from the solution and so when Newton's method is used a globalization method is also needed. The starting point for the globalization phase is usually constructed by setting the state variables to free-stream values everywhere in the domain. The continuation algorithm proceeds until a certain termination criterion is met, where different globalization methods can use different termination criteria. Once the globalization phase is terminated, the inexact Newton method is used to converge the flow residual to a specified tolerance.

The Newton-Krylov methodology is currently popular among the CFD community but it has not always been the case. Before the development of efficient linear solver and preconditioning methods, explicit methods were quite popular. A well-known example of an explicit solver is that of MacCormack [103] (1969), who solved the compressible Navier-Stokes equations using a fully explicit method with second-order accuracy in both time and space based on the earlier work of Lax and Wendroff [90] (1960). The slow convergence of explicit methods was greatly accelerated using the multigrid method, which began to appear over the next two decades [12, 72, 105, 109, 122]. For steady flows, it is common

to use a spatially-varying time step and a multi-stage method to solve the equations efficiently [142]. However, explicit methods suffer from very slow convergence rates on fine grids and are especially inefficient for turbulent flows, where the meshes typically contain very high aspect ratio cells to accurately resolve the viscous boundary layer.

Implicit methods were less popular in the early years of CFD partly due to the high memory requirements. Implicit algorithms were also not as competitive as they are in modern practice due to the high computational cost of solving the linear systems of equations prior to the development of efficient Krylov solvers for non-symmetric systems of equations. Solving the linear system is a simple task for one-dimensional problems, since the Jacobian is a narrow-stencil banded matrix, but in two or three dimensions the bandwidth increases dramatically due to the more complex equation coupling at each grid node. The approximate factorization algorithm was introduced by Beam and Warming [9] for hyperbolic systems to approximate the Jacobian for a two-dimensional system by the multiplication of two matrices which take the form of the Jacobian of a one-dimensional system. This method was applied to the Navier-Stokes equations by Steger [158] in the well-known ARC2D algorithm, with further improvements by Pulliam and Chaussee [141] and Pulliam [140]. However, the convergence rate of this method is still limited by forming the Jacobian approximately.

Krylov subspace methods were first introduced as direct methods in the 1950s [59] but gained popularity when they were introduced as iterative methods by Reid [143]. A history of Krylov methods is given by Golub and O'Leary [53]. The development of Krylov linear solvers such as the Generalized Minimal Residual (GMRES) method developed by Saad and Schulz [149] was an important milestone in the development of implicit flow solvers, as partially converged solutions to linear systems of equations could be achieved in much less time than with previous iterative solvers such as SSOR or direct solvers such as LU decomposition with Gaussian elimination. However, solving the linear system inexactly affects the performance of Newton's method, hence the branding inexact Newton. The consequence of under-solving the linear system is that the convergence rate can be degraded below q-quadratic. It can still be possible to achieve q-super-linear or even q-quadratic convergence by adapting the residual tolerance for the linear solution algorithm based on the nonlinear residual [31, 41]. However, it is not necessarily cost-effective to do so as the linear solver tolerances can become quite conservative.

An additional benefit of Krylov linear solvers is that the Jacobian matrix is not explicitly needed in the algorithm. What is needed is the matrix-vector products, which can be approximated without explicitly forming the matrix. This is quite useful because the full Jacobian is very costly to form in terms of data storage and it is also laborious to implement and prone to programming errors. Methods for forming the Jacobian-vector products without explicitly forming the Jacobian matrix are discussed later in this chapter. Since a smaller-stencil matrix is still often formed for building the preconditioner, as is done in the current work, the algorithm is more appropriately referred to as Jacobian-free rather than matrix-free.

Some important early examples demonstrating the efficiency of the Newton-Krylov methodology for CFD problems are Venkatakrishnan et al. [165], Barth and Linton [7], and Anderson et al. [3]. A literature survey of Jacobian-free Newton Krylov methods prior to 2004 is given by Knoll and Keyes [81]. Much additional discussion and literature review of Jacobian-free Newton-Krylov methods and preconditioning techniques is given by Pueyo [136], Nelson and Zingg [118], Chisholm [23], or Gatsis [45].

The flow solver used in the current study has its roots in the implicit solver ARC2D. The original ARC2D of Pulliam [140] used the algebraic turbulence model of Baldwin and Lomax [6]. Godin et

al. [52] (1997) compared the one-equation SA model to the two-equation Menter [112] model using the ARC2D algorithm and found that the SA model generally gave better predictions for non-separated flows. Pueyo [136] and Pueyo and Zingg [137, 138] (1998) employed the same structured finite-difference discretization philosophy of the traditional ARC2D of Pulliam [140] with the major difference that a preconditioned Newton-Krylov algorithm was used to solve the discrete system of equations. As with Pulliam's ARC2D algorithm, Pueyo also used the algebraic turbulence model of Baldwin and Lomax. Pueyo demonstrated that the performance of the Newton-Krylov algorithm scaled better with mesh refinement than the ARC2D algorithm [136].

The Newton-Krylov approach was later applied to the two-dimensional thin-layer Navier-Stokes equations with SA turbulence model by Nemec and Zingg [120] and to the Euler equations on three-dimensional multi-block grids by Nichols and Zingg [123], both using the finite-difference discretization of ARC2D. Chisholm [23] and Chisholm and Zingg [24, 25] implemented a two-dimensional finite-difference Newton-Krylov algorithm with SA turbulence model. A three-dimensional unstructured finite-volume turbulent flow solver was developed by Wong and Zingg [173] who also used the SA one-equations turbulence model.

The flow solver which we have adopted for our studies is the parallel implicit Newton-Krylov-Schur flow solver of Hicken [60] and Hicken and Zingg [63]. The main difference with this flow solver and previous flow solvers is the use of the SBP-SAT discretization as well as the parallelization, for which Schur and Schwarz with block ILU(p) were both found to be efficient [60].

The original flow solver of Hicken was second-order accurate and included the Euler equations only. The flow solver was augmented to the RANS equations by Osusky and Zingg [130, 131], Osusky et al. [129], and Osusky [128]. Dias and Zingg [34] and Dias [33] included third- and fourth-order accurate capabilities to the Euler equations and found that some efficiency gains were possible though a limited amount of testing was performed for practical flow problems. The Navier-Stokes equations were augmented with third- and fourth-order capabilities by Holt [68]. Third- and fourth-order capabilities are currently being implemented for the RANS-SA equations by Shen and Zingg [154].

Based on the work of Martins et al. [106, 107] and Kenway et al. [79], Zhang et al. [176] have recently incorporated the capability of coupling the aerodynamic analysis with the elastic structural deflections of the wing, known as aerostructural analysis, by solving the flow and structures equations iteratively using a block Gauss-Jacobi iterative method. They are currently working on implementing a fully implicit monolithic approach to solving the aerostructural system, an approach which could potentially benefit from homotopy continuation in the future. Despite these recent advances, application of the homotopy continuation algorithms is presently limited to purely aerodynamic analysis using a second-order accurate discretization.

3.2 Inexact Newton Method

Consider a nonlinear algebraic system of equations, represented by

$$\mathcal{F}(\mathbf{q}) = \mathbf{0}, \tag{3.2}$$

$$\mathcal{F} : \mathbb{R}^N \to \mathbb{R}^N, \ \mathbf{q} \in \mathbb{R}^N.$$

The update due to Newton's method, when applied to this system of equations, is calculated by solving the linear system of equations

$$\nabla \mathcal{F}^{(n)}\left(\mathbf{q}\right) \Delta \mathbf{q}^{(n)} = -\mathcal{F}\left(\mathbf{q}^{(n)}\right), \tag{3.3}$$

$$\Delta \mathbf{q}^{(n)} \equiv \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)},$$

where $\nabla \mathcal{F}^{(n)} : \mathbb{R}^N \to \mathbb{R}^N$ is the Jacobian of $\mathcal{F}\left(\mathbf{q}\right)$, defined as

$$\nabla \mathcal{F}_{[i,j]}\left(\mathbf{q}\right) \equiv \frac{\partial \mathcal{F}_{[i]}\left(\mathbf{q}\right)}{\partial \mathbf{q}_{[j]}}, \tag{3.4}$$

which can be represented by a square matrix. Since the linear system is being solved to some relative tolerance $\tau_{\text{l}}^{(n)} \in \mathbb{R}$, the actual Newton step is taken inexactly, and the update $\Delta\mathbf{q}$ does not satisfy equation (3.3) but does satisfy the inequality

$$\left\| \mathcal{F}\left(\mathbf{q}^{(n)}\right) + \nabla \mathcal{F}^{(n)}\left(\mathbf{q}\right) \Delta \mathbf{q}^{(n)} \right\| \leq \tau_{\text{l}}^{(n)} \left\| \mathcal{F}\left(\mathbf{q}^{(n)}\right) \right\|. \tag{3.5}$$

3.3 Physicality Constraints

If either the pressure p or density ρ becomes negative, the flow becomes non-physical and the flow residual cannot be evaluated as it would contain complex-valued entries. Consistent with Hicken and Zingg [63], if an intermediate iteration results in negative p or ρ then the solution update $\Delta\mathbf{q}$ is damped by a factor of 0.9 repeatedly until a physical state vector is obtained. This mechanism can trigger in some of the more difficult flow solves, especially for some high Mach number transonic cases and usually in the globalization phase, but does not normally trigger for most flow solves.

3.4 The Linear Solvers

Solving linear systems of the form $\mathcal{A}\mathbf{x} = \mathbf{b}$ for unknown $\mathbf{x} \in \mathbb{R}^N$ given $\mathcal{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{b} \in \mathbb{R}^N$ is a task which must be performed at every iteration of an implicit flow solver. With the exception of the matrix-free monolithic homotopy algorithms developed in Chapter 7, the homotopy algorithms also require solutions to linear systems of equations in this form. In this thesis, the linear system is solved using a flexible variant of GMRES called FGMRES [146, 148, 149]. The use of FGMRES instead of GMRES is due to the Schur preconditioner and is explained in Section 3.5.

GMRES belongs to a class of iterative methods known as Krylov subspace methods. An iterative linear solver can produce an approximate solution in less time than a direct solver would take to produce an exact solution (to machine precision). In the current implementation, linear iterations are performed until the residual $\|\mathbf{b} - \mathcal{A}\mathbf{x}\| / \|\mathbf{b}\|$ is below some user-specified tolerance τ_{l}, where τ_{l} is typically around 0.01. It is only necessary to compute the residual explicitly at the end of iterations because one of the quantities calculated in the GMRES algorithm is equal to this value, and hence the convergence criterion can be checked at no cost.

At each linear iteration, an additional vector is added to the Krylov subspace. The computational cost of calculating this vector is one matrix-vector product as well as several vector operations. The number of vector operations increases at each linear iteration and, if enough linear iterations are performed, can eventually become more expensive than the residual evaluation.

To alleviate both the memory and computational cost per linear iteration, GMRES can be restarted after a fixed number of iterations using the partially converged solution as the initial guess. While this can reduce the cost of the linear iterations and the memory requirements, it can significantly impact the convergence rate and frequent restarting can be computationally inefficient [148]. In the current implementation, restarting is not used. Instead, if a maximum number of iterations are reached, the GMRES algorithm is terminated and the partially-converged Newton update is applied, even if the relative residual is not below the tolerance τ_l.

In addition to FGMRES, a flexible variant of the linear solver GCROT is available to solve the linear system. This linear solver was developed by de Sturler [28] and was modified to GCROT(m, k) by Hicken and Zingg [66] for application to gradient-based optimization [65]. GCROT(m, k) uses a linear solver internally, in this case FGMRES. The advantage of GCROT(m, k) over FGMRES is that it allows the internal linear solver to essentially be restarted but without suffering the severe convergence penalty normally incurred when restarting. While this linear solver is not typically used as part of the Newton-Krylov algorithm, it is useful for several homotopy analysis applications.

3.5 Preconditioning

The convergence rate of Krylov solvers depends heavily on the preconditioner. However, good preconditioners can be very expensive to form in terms of both CPU time and data storage and so the quality and efficiency of the preconditioner can have a very significant impact on the overall flow solver performance.

All parallel preconditioners discussed in this section make use of an ILU(p) decomposition. The term ILU(p) refers to an *incomplete lower-upper* factorization with fill level p and is detailed by Saad [148]. In general, exact decomposition of a matrix into a lower and upper triangular matrix $\mathcal{A} = \mathcal{L}\mathcal{U}$ is very expensive and leads to the formation of two dense matrices. The basic concept of ILU(p) is that matrices $\mathcal{L}$ and $\mathcal{U}$ can be approximated by sparse approximations $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{U}}$, saving both CPU time and memory. It is then straightforward to invert the triangular matrices and perform the computation $\tilde{\mathcal{U}}^{-1}\tilde{\mathcal{L}}^{-1}$ to form an approximation to $\mathcal{A}^{-1}$. Inverse matrices are not formed explicitly in practice; if a quantity $\mathbf{y} = \mathcal{L}^{-1}\mathbf{x}$ is needed then this can be obtained by solving the linear system $\mathcal{L}\mathbf{y} = \mathbf{x}$ for $\mathbf{y}$.

The ILU(p) technique can be used to form a right or left preconditioner. One advantage of using a right preconditioner is that the residual is the same as the unpreconditioned system, which means that no additional algebra needs to be performed to check the convergence criterion of the linear solver. In this thesis, right preconditioning is used. Instead of solving $\mathcal{A}\mathbf{x} = \mathbf{b}$, the system

$$\mathcal{A}\tilde{\mathcal{U}}^{-1}\tilde{\mathcal{L}}^{-1}\tilde{\mathbf{x}} = \mathbf{b} \tag{3.6}$$

is solved for $\tilde{\mathbf{x}}$, and $\mathbf{x}$ is calculated from

$$\mathbf{x} = \tilde{\mathcal{U}}^{-1}\tilde{\mathcal{L}}^{-1}\tilde{\mathbf{x}}. \tag{3.7}$$

The number of nonzeros in the ILU(p) decomposition is based on the location (but not the magnitude) of non-zeros in the matrix and also the fill level p, where the smallest fill level $p = 0$ will add no additional non-zeros to the initial matrix graph. The preconditioner is formed based on a first-order approximate Jacobian matrix in order to reduce the CPU time and memory requirements of the ILU(p) factorization. In the current study, a block version of ILU(p) is used where the small block matrices at each node

(5 by 5 for the Euler and Navier-Stokes equations, 6 by 6 for the RANS-SA equations) are treated as block elements. A more detailed description and analysis of serial preconditioners for a two-dimensional Newton-Krylov flow solver including block ILU(p) is given by Pueyo [136] or especially Gatsis [45].

The first-order preconditioner matrix used to form the ILU(p) factorization is constructed to use information from only the current grid point and nearest neighbours. For three-dimensional grids, this consists of a total of seven points for interior nodes. In order to limit the stencil to include only nearest neighbours, several approximations need to be made. The third-order dissipation operator is approximated with a first-order dissipation operator multiplied by a "lumping factor," the cross-derivative terms are ignored in the discretization of the viscous terms, and the pressure sensor, if present, is treated as constant. As a matter of convenience and to save on floating point operations, the spectral radius in the dissipation operator is also treated as constant. While this is not necessary to maintain the smaller stencil size, we have found from experimentation that including the linearization of the spectral radius in the preconditioner matrix produces no noticeable benefit.

The dissipation lumping factor $\sigma_{\text{dlf}} \in \mathbb{R}$ is due to Pueyo and Zingg [138]. This is a modification to the coefficient in front of the first-order dissipation term in the preconditioner matrix to approximately capture the contributions of the third-order dissipation to the Jacobian. The modification is applied to $\kappa^{(2)}$ in the preconditioner matrix only:

$$\kappa^{(2)} \leftarrow \kappa^{(2)} + \sigma_{\text{dlf}} \kappa^{(4)}. \tag{3.8}$$

Previous studies have found that values of σ_{dlf} in the range of 4 to 6 are suitable for the Euler equations [123, 34]. For the three-dimensional RANS-SA equations, Osusky and Zingg [130] suggest $\sigma_{\text{dlf}} = 8$ for scalar dissipation and $\sigma_{\text{dlf}} = 12$ for matrix dissipation. For a similar two-dimensional flow solver using the same dissipation models and turbulence model, Chisholm and Zingg [25] have suggested $\sigma_{\text{dlf}} = 5$ for scalar dissipation and $\sigma_{\text{dlf}} = 12$ for matrix dissipation. The optimal value of σ_{dlf} can be case-dependent. For simplicity, $\sigma_{\text{dlf}} = 7$ is used for all cases in this thesis.

Building a preconditioner based on an ILU(p) factorization of the global system has previously been found to be too costly to be competitive for parallel computations [63]. An alternative is to use a simple Schwartz preconditioner, which essentially ignores the interface SAT terms which couple the subdomains local to each process and applies a block ILU(p) preconditioner to the local subdomains. A second alternative is to use the Schur complement method. The approximate Schur preconditioner used in this thesis is based on the work of Saad and Sosonkina [150] with some efficiency improvements by Hicken and Zingg [63]. The basic premise of this method is that the coupling terms between subdomains are approximately solved for and eliminated. Since the subdomains have been decoupled, the global ILU(p) factorization of the resulting approximation to the global system can be constructed by individually calculating the ILU(p) decompositions on the subdomains.

The Schur complement method is more complicated to implement than the Schwartz method and is also more computationally expensive to form. The relative performance of the two preconditioners is grid-dependent but they generally give similar performance when only a few processors are used [63]. However, when the number of processors reaches the order of 100 or more, the Schur complement method has been found to be noticeably more efficient for both inviscid and viscous flows [62]. The Schur preconditioner is used for all linear solves in this thesis based on these past findings.

When using the Schur complement method, the relaxation in solving the Schur complement system inexactly results in slight inconsistency in the factored preconditioner between successive iterations of

the linear solver. This is the reason the flexible linear solver FGMRES must be used in place of GMRES as mentioned in the previous section.

3.6 Linear System Scaling

Scaling of the linear system can have a dramatic impact on the performance of the linear solver. The condition number of a matrix $\mathcal{A} \in \mathbb{R}^N \times \mathbb{R}^N$ is defined as the ratio of the largest to smallest singular value of $\mathcal{A}$, where the singular values of $\mathcal{A}$ are defined as the square root of the eigenvalues of $\mathcal{A}^T\mathcal{A}$. However, since the eigenvalues of the matrix are very difficult to obtain, an easier rule of thumb to attempt to improve the conditioning of the linear system is to attempt to make the row and column norms of similar magnitude. Of course this must be done such that the solution to the original system of equations can still be recovered. This can be accomplished by row and column scaling.

Another effect of the row scaling is that it can distribute emphasis on certain equations in the iterative linear solver. If some residual entries are smaller, then these entries will be emphasized less during linear iterations and the error associated with the corresponding solution components may be reduced by less on completion of the linear iterations. This reasoning provides incentive to apply scaling based on the residual vector.

Row and column scaling operations can be represented by diagonal matrices. For a general system of equations of the form $\mathcal{A}\mathbf{x} = \mathbf{b}$, $\mathcal{A} \in \mathbb{R}^N \times \mathbb{R}^N$, $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{b} \in \mathbb{R}^N$, the system can be rewritten to include row and column scaling operations:

$$\mathcal{S}_\mathrm{r}\mathcal{A}\mathcal{S}_\mathrm{c}\mathcal{S}_\mathrm{c}^{-1}\mathbf{x} = \mathcal{S}_\mathrm{r}\mathbf{b}, \tag{3.9}$$

where $\mathcal{S}_\mathrm{r} \in \mathbb{R}^N \times \mathbb{R}^N$ and $\mathcal{S}_\mathrm{c} \in \mathbb{R}^N \times \mathbb{R}^N$ are diagonal row and column scaling matrices respectively. The following procedure is used to calculate the solution of the system $\mathcal{A}\mathbf{x} = \mathbf{b}$:

1. Set $\tilde{\mathcal{A}} = \mathcal{S}_\mathrm{r}\mathcal{A}\mathcal{S}_\mathrm{c}$ and $\tilde{\mathbf{b}} = \mathcal{S}_\mathrm{r}\mathbf{b}$;
2. Solve $\tilde{\mathcal{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ for $\tilde{\mathbf{x}}$;
3. Compute $\mathbf{x} = \mathcal{S}_\mathrm{c}\tilde{\mathbf{x}}$.

3.6.1 Scaling of the Euler and Navier-Stokes Equations

Considering the Euler and Navier-Stokes equations, the non-dimensional flow variables are of order unity throughout most of the domain, but the residual also contains geometric terms which depend on the local grid spacing, including the inverse geometric Jacobian $\mathcal{J}^{-1}$, which approximates the cell volume corresponding to each grid node. The geometric Jacobian can vary by orders of magnitude throughout the physical domain. Chisholm and Zingg [25] alleviated this issue by applying a row scaling factor of $\mathcal{J}$, which they referred to as *inherent scaling*. Osusky and Zingg [130] instead apply a factor of $\mathcal{J}^{(D-1)/D}$, where D is the number of spatial dimensions (either 2 or 3).

The inherent scaling is not the only row scaling factor that is applied. Each component equation (mass, x-momentum, y-momentum, etc.) is scaled by the residual norm associated with that component. To clarify, if the conservation of mass equations for all grid nodes are arranged into a single vector, then the L^2-norm of this vector is the factor which each entry of the residual vector corresponding to the conservation of mass equation is divided by. This scaling factor helps to equalize the row norms associated with each component equation and is referred to as *equation scaling*.

The final scaling is a factor of the total residual norm which is applied to the entire matrix. Since this factor does not affect the ratio of the maximum to minimum singular value, this factor does not affect performance of the linear solver. It is provided so that the residual norm is relatively consistent for different flow solves on different meshes.

The combination of the two residual scalings was proposed by Chisholm and Zingg [25] who referred to it as *auto-scaling*. Both the component residual norm and the residual norm are calculated from the residual vector before the inherent scaling is applied. As mentioned, no column scaling is normally applied to the Euler or laminar Navier-Stokes equations.

3.6.2 Scaling of the RANS-SA Equations

The inherent scaling for the RANS-SA equations was implemented by Osusky and Zingg [130] based on the earlier work of Chisholm and Zingg [25]. It is similar in philosophy to the scaling for the Euler and Navier-Stokes equations with the following differences:

- An additional factor of $\mathcal{J}^{-1}$ is applied to the SA model since it is not normally present in this component equation.
- Since the turbulence variable $\tilde{\nu}$ can take values on the order of 10^3 and the mean flow variables are of order unity as a result of the non-dimensionalization, a column scaling factor of 10^{-3} is applied to the columns corresponding to $\tilde{\nu}$.
- A row scaling factor of 10^3 is applied to the rows of the SA model equation.
- The auto-scaling is calculated after the inherent scaling is applied.[1]

The scaling for the RANS-SA equations can be summarized in the notation of equation (3.9):

$$\mathcal{S}_\mathrm{r} = \mathrm{diag}\left(\mathcal{S}_\mathrm{r}^{(1)}, \ldots, \mathcal{S}_\mathrm{r}^{(N/6)}\right), \quad \mathcal{S}_\mathrm{c} = \mathrm{diag}\left(\mathcal{S}_\mathrm{c}^{(1)}, \ldots, \mathcal{S}_\mathrm{c}^{(N/6)}\right)$$

$$\mathcal{S}_\mathrm{r}^{(i)} = \mathrm{diag}\left(\mathcal{J}^{2/3}, \mathcal{J}^{2/3}, \mathcal{J}^{2/3}, \mathcal{J}^{2/3}, \mathcal{J}^{2/3}, 10^3\mathcal{J}^{-1/3}\right), \quad \mathcal{S}_\mathrm{c}^{(i)} = \mathrm{diag}\left(1, 1, 1, 1, 1, 10^{-3}\right).$$

The RANS-SA scaling is applied differently for the homotopy algorithm. There are three differences:

- The column scaling is (partly) replaced with explicit variable scaling of $\tilde{\nu}$, as explained in Section 5.7.
- The auto-scaling is calculated before the inherent scaling is applied.
- The row scaling factor of 10^3 is not applied.

The reason the auto-scaling is calculated before the inherent scaling is applied is due to practical programming considerations. In the PTC codes, the auto-scaling is applied directly to the residual vector whereas the inherent scaling is stored in vector format and applied to the residual only after it has been re-cast as a vector in the proper format. The homotopy algorithms generally involve more manipulations of the residual vector than PTC and sometimes involve the solution to linear systems of equations where

[1] This actually makes the row scaling factor of 10^3 redundant because it will cancel when the component residual norm is applied.

the matrix is not the Jacobian of the right-hand side vector. Consequently, it can be inconvenient to work with $\mathcal{R}(\mathbf{q})$ after the auto-scaling has been applied, particularly in the Jacobian-vector product estimations.

Applying the inherent scaling before the auto-scaling seems not to have a significant impact on algorithm performance. However, a consequence is that the residual norm that is tracked to assess convergence is several orders of magnitude lower. Inspection of the convergence histories of C_L and C_D for some test cases indicates that the first 10^{-5} drop in the PTC residual norm results in approximately the same error reduction as a 10^{-3} drop in the homotopy residual norm. The difference of two orders of magnitude seems to be consistent throughout the rest of the flow solve, so that a total drop of 10^{-12} in the PTC residual norm results in approximately the same error reduction as a 10^{-10} drop in the homotopy residual norm.

3.6.3 Alternative Scaling

As an alternative to the row- and column-scaling procedure of the previous section, which will be referred to as geometric scaling, the linear system can be scaled by applying a row and column normalization procedure. The procedure adopted here is taken from Saad [147] who applied this scaling to test linear preconditioner and linear solver performance for a large suite of linear systems arising in a wide array of CFD applications. First, the row scaling is applied by dividing each row by the L^2-norm of that row. The column scaling is then applied to the row-normalized matrix by dividing each of its columns by the L^2-norm of that column. Since the column scaling is applied second, only the columns of the scaled matrix will actually be normalized and not the rows.

This normalization generally produces a very well-conditioned matrix. However, this can be a disadvantage for the outer nonlinear algorithm because each component of the residual will be emphasized to the same extent in the Krylov solver regardless of how much error it actually represents in the nonlinear problem. As a result, the practical effect of using the normalization scaling of this section in a fixed-point method such as Newton's method compared to the geometric scaling is that each linear solve will complete in fewer iterations but will improve the nonlinear residual less.

While the overall CPU cost of either scaling method is comparable, it can be less intuitive to select solver parameters when using the normalization scaling. For example, since more nonlinear iterations are used, the time step update used by PTC, which is discussed in Section 3.9.1, must be considerably more conservative. Because flow solver parameters are less intuitive with this scaling it is not generally used with the PTC algorithm, though we include it in some of the analysis and results related to the monolithic homotopy algorithm. The reasoning for this is included in the appropriate sections.

3.7 Matrix-Vector Products

Consider a nonlinear algebraic system of equations $\mathcal{F}(\mathbf{q})$, $\mathcal{F}: \mathbb{R}^N \to \mathbb{R}^N$ with Jacobian $\nabla\mathcal{F}(\mathbf{q}): \mathbb{R}^N \to \mathbb{R}^N$. All of the linear systems which need to be solved in the Newton-Krylov algorithm, the pseudo-transient continuation algorithm, and the homotopy continuation algorithms have a Jacobian as the matrix which needs to be inverted. In the FGMRES linear solver, the only appearance of the matrix is in the matrix-vector product calculation, and any other appearance of the Jacobian matrix in any of these algorithms is also in the form of a matrix-vector product. As such, it is not necessary to form and store the Jacobian in any of these algorithms if it is possible to form the Jacobian-vector products some

other way.

One way to approximate the Jacobian-vector products is to use a finite-differencing method (FDMVPs). This is explained by defining the Jacobian through the mapping that it provides rather than through its matrix representation given previously as equation (3.4).

Definition 3.1. *Let* $\mathbf{u} \in \mathbb{R}^{N_1}$, $\mathbf{v} \in \mathbb{R}^{N_1}$, *and let* $\mathcal{F} : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ *be at least* C^1 *differentiable. The* Jacobian *of* $\mathcal{F}(\mathbf{u})$ *is an operator* $\nabla\mathcal{F}(\mathbf{u}) : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ *such that*

$$(\nabla\mathcal{F}(\mathbf{u})\,\mathbf{v})_{[i]} \equiv \sum_j \frac{\partial}{\partial \mathbf{u}_{[j]}} \mathcal{F}_{[i]}(\mathbf{u})\,\mathbf{v}_{[j]}. \tag{3.10}$$

It can be seen from equation (3.10) that the mapping $\nabla\mathcal{F}(\mathbf{u})\,\mathbf{v}$ represents the directional derivative of $\mathcal{F}(\mathbf{u})$ in the direction of $\mathbf{v}$ and with "speed" $\|\mathbf{v}\|$. Hence the mapping $\nabla\mathcal{F}(\mathbf{u})\,\mathbf{v}$ can be written in the Fréchet sense using forward-differencing:

$$\nabla\mathcal{F}(\mathbf{u})\,\mathbf{v} = \lim_{\epsilon\to 0} \frac{\mathcal{F}(\mathbf{u}+\epsilon\mathbf{v}) - \mathcal{F}(\mathbf{u})}{\epsilon}, \tag{3.11}$$

$\epsilon \in \mathbb{R}$, or, similarly, with backwards-differencing:

$$\nabla\mathcal{F}(\mathbf{u})\,\mathbf{v} = \lim_{\epsilon\to 0} \frac{\mathcal{F}(\mathbf{u}) - \mathcal{F}(\mathbf{u}-\epsilon\mathbf{v})}{\epsilon}. \tag{3.12}$$

A first-order approximation to the Jacobian-vector product can be formed by applying either equation (3.11) or (3.12) with a small but finite ϵ and ignoring the limit. These approximations will be represented by operators $\mathcal{D}^+ : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ and $\mathcal{D}^- : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$:

$$\mathcal{D}^+\mathcal{F}(\mathbf{u})\,\mathbf{v} \equiv \frac{\mathcal{F}(\mathbf{u}+\epsilon\mathbf{v}) - \mathcal{F}(\mathbf{u})}{\epsilon}, \tag{3.13}$$

$$\mathcal{D}^-\mathcal{F}(\mathbf{u})\,\mathbf{v} \equiv \frac{\mathcal{F}(\mathbf{u}) - \mathcal{F}(\mathbf{u}-\epsilon\mathbf{v})}{\epsilon}. \tag{3.14}$$

It is also possible, and sometimes practical, to form a second-order accurate approximation

$$\mathcal{D}\mathcal{F}(\mathbf{u})\,\mathbf{v} \equiv \frac{\mathcal{F}(\mathbf{u}+\epsilon\mathbf{v}) - \mathcal{F}(\mathbf{u}-\epsilon\mathbf{v})}{2\epsilon}. \tag{3.15}$$

While the second-order approximation incurs less truncation error than the first-order approximations for sufficiently small ϵ, it also comes at twice the computational cost, assuming that the residual at the unperturbed state $\mathcal{F}(\mathbf{q})$ has no cost because it has been calculated and stored previously. We have found repeatedly that there is no noticeable effect on accuracy when using the centred difference compared to the forward difference, suggesting that truncation error is not significant at the values of ϵ typically used, and hence the first-order difference given by equation (3.11) is always the option chosen when FDMVPs are used, unless otherwise specified.

The parameter ϵ should be chosen to balance between the error in the finite-differencing approximation and rounding error. Following the approach of Nielsen et al. [124], as well as Hicken and Zingg [63] and Osusky and Zingg [130], the following expression is used:

$$\epsilon = \sqrt{\frac{N\delta}{\mathbf{v}^T\mathbf{v}}}, \tag{3.16}$$

where $\delta \in \mathbb{R}$ is typically taken around 10^{-12}, which is about 10^3 times machine precision when using double-precision arithmetic. Since $\mathbf{v}$ is used in this formula instead of the vector $\mathbf{w} \in \mathbb{R}^N$ defined by $\mathbf{w}_{[j]} \equiv \mathbf{v}_{[j]}/\mathbf{u}_{[j]}$, this formula assumes that the flow variables have been non-dimensionalized and are of order unity.

Another way to approximate the matrix-vector products is to recycle the smaller-stencil approximate Jacobian used to form the preconditioner which was discussed in Section 3.5. Matrix-vector products using the approximate Jacobian will be referred to as approximate matrix-vector products (AMVPs).

AMVPs are less accurate than FDMVPs, but they are cheaper to form. As a consideration for parallelization on a distributed-memory architecture using a message-passing interface (MPI), AMVPs can easily be implemented using non-blocking communication whereas FDMVPs cannot, further reducing the relative cost of the AMVPs. FDMVPs are typically used in the inexact Newton portion of the flow solver since the additional accuracy has a bigger impact on the convergence rate in this phase. Our experience is that the AMVPs can still be used to converge to machine precision in the inexact Newton phase but are not an efficient choice due to the very slow convergence rate. For flow solves globalized with PTC, the accuracy of the Jacobian matrix is less important and we have generally found that shorter convergence times can be achieved using AMVPs while maintaining algorithm robustness.

Matrix-vector products can also be approximated using the complex step method. The notion of estimating real derivatives using complex arithmetic dates back to Lyness and Moler [102] and Lyness [101]. The complex step method was first presented by Squire and Trapp [157] and was applied to CFD algorithms soon after; for example, Newman et al. [121] or Martins et al. [108]. The derivative approximation is given by the expression

$$\nabla \mathcal{F}(\mathbf{u})\,\mathbf{v} \approx \mathrm{Im}\frac{\mathcal{F}(\mathbf{u} + i\epsilon\mathbf{v})}{\epsilon}, \tag{3.17}$$

where $i \equiv \sqrt{-1}$, $i \in \mathbb{C}$. Furthermore, $\mathbf{u}, \mathbf{v} \in \mathbb{C}^N$ for this calculation but are assumed to have imaginary component $\mathbf{0}$; e.g. $\mathbf{u} = \mathbf{u}_r + i\mathbf{u}_i$, $\mathbf{u}_i = \mathbf{0}$, $\mathbf{u}_r, \mathbf{u}_i \in \mathbb{R}^N$. This expression can be derived using either a Taylor polynomial around $\mathbf{u}$ in the direction $i\mathbf{v}$ [157, 121, 108] or, as shown by Martins et al. [108], by applying the Cauchy-Riemann equations

$$\mathcal{D}_{\mathbf{u}_r}\mathrm{Re}\left(\mathcal{F}(\mathbf{u})\right) = \mathcal{D}_{\mathbf{u}_i}\mathrm{Im}\left(\mathcal{F}(\mathbf{u})\right), \;\; \mathcal{D}_{\mathbf{u}_i}\mathrm{Re}\left(\mathcal{F}(\mathbf{u})\right) = -\mathcal{D}_{\mathbf{u}_r}\mathrm{Im}\left(\mathcal{F}(\mathbf{u})\right). \tag{3.18}$$

The advantage of using equation (3.17) over the FDMVPs to approximate the matrix-vector products is that there is no subtractive cancellation error and so ϵ can be taken very small, for example $\epsilon = 10^{-30}$, which can provide a very accurate estimate of the matrix-vector product. The disadvantage is that the cost of computing each matrix-vector product is substantially higher due to the need for complex arithmetic. Due to its cost, complex step matrix-vector products are not used in any of the flow solver algorithms, though they are used in some of the analysis.

3.8 Tensor-Vector Products

Tensor-vector products are not required by any part of the Newton-Krylov algorithm but are used with some of the homotopy analysis tools. Tensors are formed on subsequent applications of the operator ∇ to the Jacobian. These operators will need to be approximated using finite-differencing.

To introduce the concept, the Hessian operator $\nabla^2\mathcal{F}(\mathbf{u})$ is defined through the mapping that it

provides.

Definition 3.2. *Let* $\mathbf{u} \in \mathbb{R}^{N_1}$, $\mathbf{v}_1 \in \mathbb{R}^{N_1}$, $\mathbf{v}_2 \in \mathbb{R}^{N_1}$, *and let* $\mathcal{F} : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ *be at least* C^2 *differentiable. The* Hessian *of* $\mathcal{F}(\mathbf{u})$ *is an operator* $\nabla^2\mathcal{F}(\mathbf{u}) : \mathbb{R}^{N_1} \times \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ *such that*

$$\left(\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2]\right)_{[i]} = \sum_k \sum_j \frac{\partial^2}{\partial \mathbf{u}_{[j]} \partial \mathbf{u}_{[k]}} \mathcal{F}_{[i]}(\mathbf{u}) \mathbf{v}_{1[j]} \mathbf{v}_{2[k]}, \tag{3.19}$$

where the notation $\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2]$ *indicates that* $\nabla^2\mathcal{F}(\mathbf{u})$ *operates on the vector pair* $[\mathbf{v}_1, \mathbf{v}_2]$.

The Hessian is easily verified to be a bilinear operator with the property

$$\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2] = \nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_2, \mathbf{v}_1]. \tag{3.20}$$

In addition, the following remark is easily verified by applying Definition 3.1 twice.

Remark 3.1. *Let* $\mathbf{u} \in \mathbb{R}^{N_1}$, $\mathbf{v}_1 \in \mathbb{R}^{N_1}$, $\mathbf{v}_2 \in \mathbb{R}^{N_1}$, *and let* $\mathcal{F} : \mathbb{R}^{N_1} \to \mathbb{R}^{N_2}$ *be at least* C^2 *differentiable. Then*

$$\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2] = \nabla\left[\nabla\mathcal{F}(\mathbf{u})\mathbf{v}_1\right]\mathbf{v}_2. \tag{3.21}$$

Like the Jacobian, the Hessian also represents a directional derivative [134]; it can be interpreted as the directional derivative of $\mathcal{F}(\mathbf{u})$ in the direction of $\mathbf{v}_1$ differentiated for a second time in the direction of $\mathbf{v}_2$. As such, it can also be approximated by successively applying the Fréchet definition of the directional derivative. In accordance with Remark 3.1 the following approximation can be constructed by successively applying first-order approximations to the first derivative:

$$\begin{aligned}
\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2] &\approx \mathcal{D}^+\left[\mathcal{D}^-\mathcal{F}(\mathbf{u})\mathbf{v}_1\right]\mathbf{v}_2 \\
&= \frac{\mathcal{D}^+\mathcal{F}(\mathbf{u})\mathbf{v}_2 - \mathcal{D}^+\mathcal{F}(\mathbf{u} - \epsilon_1\mathbf{v}_1)\mathbf{v}_2}{\epsilon} \\
&= \frac{\mathcal{F}(\mathbf{u} + \epsilon_2\mathbf{v}_2) - \mathcal{F}(\mathbf{u} - \epsilon_1\mathbf{v}_1 + \epsilon_2\mathbf{v}_2) - \mathcal{F}(\mathbf{u}) + \mathcal{F}(\mathbf{u} - \epsilon_1\mathbf{v}_1)}{\epsilon_1\epsilon_2}.
\end{aligned} \tag{3.22}$$

Similarly, the second-order approximation to the first derivative can be used:

$$\begin{aligned}
&\nabla^2\mathcal{F}(\mathbf{u})[\mathbf{v}_1, \mathbf{v}_2] \approx \mathcal{D}\left[\mathcal{D}\mathcal{F}(\mathbf{u})\mathbf{v}_1\right]\mathbf{v}_2 \\
&= \frac{\mathcal{F}(\mathbf{u} + \epsilon_1\mathbf{v}_1 + \epsilon_2\mathbf{v}_2) - \mathcal{F}(\mathbf{u} - \epsilon_1\mathbf{v}_1 + \epsilon_2\mathbf{v}_2) - \mathcal{F}(\mathbf{u} + \epsilon_1\mathbf{v}_1 - \epsilon_2\mathbf{v}_2) + \mathcal{F}(\mathbf{u} - \epsilon_1\mathbf{v}_1 - \epsilon_2\mathbf{v}_2)}{2^2\epsilon_1\epsilon_2}.
\end{aligned} \tag{3.23}$$

This method for calculating second-order directional derivatives can easily be extended to directional derivatives of any order. The algorithms developed for this purpose can be found in Appendix D.

Comparing the first- and second-order approximations (3.22) and (3.23), the second-order approximation will incur less truncation error for sufficiently small ϵ but requires an additional residual evaluation, assuming that $\mathcal{F}(\mathbf{u})$ has previously been computed.

The complex step method can also be extended to directional derivatives of any degree. The second-

order accurate approximation to the Hessian-vector product using the complex step method is

$$\nabla^2_{\mathbf{u}_r}\mathcal{F}(\mathbf{u})[\mathbf{v}_1,\mathbf{v}_2] \approx -\mathcal{D}^2_{\mathbf{u}_i}\mathcal{F}(\mathbf{u})[\mathbf{v}_1,\mathbf{v}_2]$$
$$= -\mathrm{Re}\left(\frac{\mathcal{F}(\mathbf{u}+i\epsilon_1\mathbf{v}_1+i\epsilon_2\mathbf{v}_2)-\mathcal{F}(\mathbf{u}-i\epsilon_1\mathbf{v}_1+i\epsilon_2\mathbf{v}_2)-\mathcal{F}(\mathbf{u}+i\epsilon_1\mathbf{v}_1-i\epsilon_2\mathbf{v}_2)+\mathcal{F}(\mathbf{u}-i\epsilon_1\mathbf{v}_1-i\epsilon_2\mathbf{v}_2)}{2^2\epsilon_1\epsilon_2}\right), \tag{3.24}$$

which has been derived using the equations that result from differentiating the Cauchy-Riemann equations (3.18).

Unlike the first derivative approximation, the complex step approximation to the second derivative suffers from subtractive cancellation errors, as do the higher derivatives. In fact the complex step approximations to the second derivative can actually be less accurate and more expensive than the approximation made using finite-differences [88]. Lai and Crassidis [88] investigated accuracy improvements by considering directional derivatives in a general complex direction and found that it is possible to develop more accurate approximations using this method than the finite-difference method. However, these complicated methods are not considered here.

3.9 Globalization Methods

The continuation methods used to globalize Newton's method are presented. While the pseudo-transient continuation algorithm is simple and is given a full description in this section, the homotopy continuation algorithms are only introduced at a fundamental level, to be discussed in detail in later chapters.

3.9.1 Pseudo-Transient Continuation

Pseudo-transient continuation (PTC) is a pseudo-time-marching method. This means that it is an imitation of physical time-marching, though time-accuracy is not required. The update formula is given by the implicit Euler method with local time linearization [99]:

$$\left(\mathcal{T}^{(n)}+\nabla\mathcal{R}^{(n)}\right)\Delta\mathbf{q}^{(n)} = -\mathcal{R}\left(\mathbf{q}^{(n)}\right), \tag{3.25}$$

where $\mathcal{T}^{(n)} = \frac{1}{\Delta t}\mathcal{I}$, and $\mathcal{I}$ is the identity matrix. Since time-accuracy is not required in the context of globalization, Δt can take large values and vary spatially. In this study, Δt is evolved according to:

$$\Delta t^{(n)}_{[i]} = \frac{\Delta t^{(n)}_{\mathrm{ref}}\mathcal{J}_{[i]}}{1+\mathcal{J}^{\frac{1}{D}}_{[i]}},\ \Delta t^{(n)}_{\mathrm{ref}} = a\,(b)^{m\left\lfloor\frac{n}{m}\right\rfloor}, \tag{3.26}$$

where $\mathcal{J}$ is the geometric Jacobian resulting from the coordinate transformation on the mesh, i is the grid point index, D is the number of spatial dimensions (either 2 or 3 in this thesis), and $\lfloor\cdot\rfloor$ is the floor operator[2]. The floor operator is present in the formula in the case where we choose to update the preconditioner every m iterations instead of every iteration. Typical values of a and b are discussed in the results section as their assignment is typically case-dependent.

[2] $\lfloor x\rfloor = y$, where y is the largest integer less than or equal to x.

PTC is terminated and the inexact Newton phase (INP) initiated when the relative residual

$$\mathcal{R}_{\text{rel}}^{(n)} \equiv \frac{\left\|\mathcal{R}\left(\mathbf{q}^{(n)}\right)\right\|}{\left\|\mathcal{R}\left(\mathbf{q}^{(0)}\right)\right\|} \tag{3.27}$$

is reduced below some user-specified tolerance τ_{rel}. Typical values of τ_{rel} are $\tau_{\text{rel}} \approx 10^{-1}$ to 10^{-2} for the Euler equations and $\tau_{\text{rel}} \approx 10^{-3}$ to 10^{-4} for the RANS-SA equations.

When the PTC method is used for globalization, the inexact Newton update is also modified to take the form of the implicit Euler update (3.26). The time step matrix is carried over from the last iteration of the PTC phase but the reference time step Δt_{ref} is replaced with the successive evolution relaxation used by Hicken and Zingg [63] and Osusky and Zingg [130], which was adapted from Mulder and van Leer [116]. The update is given as

$$\Delta t_{\text{ref}}^{(n)} = \max\left[\alpha\left(R_d^{(n)}\right)^{-\beta}, \Delta t_{\text{ref}}^{(n-1)}\right], \tag{3.28}$$

$$\alpha = a b^{m\left\lfloor \frac{n_{\text{Newt}}}{m} \right\rfloor}\left(R_d^{(n_{\text{Newt}})}\right)^{\beta},\ R_d^{(n)} \equiv \frac{\left\|\mathcal{R}^{(n)}\right\|}{\left\|\mathcal{R}^{(0)}\right\|},$$

where n_{Newt} is the first inexact Newton iteration. This update formula tends to increase Δt_{ref} rapidly in the inexact Newton phase so that the contribution of $\mathcal{T}$ quickly becomes small, so this modification can be seen as a short transition region between the pseudo-transient phase and inexact Newton phase.

A pseudo-code of the PTC algorithm is provided as Algorithm 3.1.

Algorithm 3.1: Pseudo-transient continuation (PTC)

Initialize: Choose a starting guess for $\mathbf{q}$ (e.g. free-stream)
while $\|\mathcal{R}\| > \tau_{\text{rel}}$ *and* $\|\mathcal{R}\| > \tau_{\text{abs}}$ **do**
  Get $\mathcal{R}$ and $\|\mathcal{R}\|$
  Set the reference time step $\Delta t_{\text{ref}} \leftarrow a \cdot b^{\lfloor n/m \rfloor}$
  Set the time step diagonal matrix $\mathcal{T}[:] \leftarrow \left(1 + \mathcal{J}^{1/D}[:]\right) / \left(\Delta t_{\text{ref}} \mathcal{J}[:]\right)$
  Form and factor the preconditioner based on the matrix $\mathcal{T} + \nabla\mathcal{R}$
  Solve the linear system $[\mathcal{T} + \nabla\mathcal{R}]\,\Delta\mathbf{q} = -\mathcal{R}$ inexactly and update $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
end
Inexact Newton Phase: Solve $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ to some tolerance using the inexact Newton method

3.9.2 Convex Homotopy Continuation

Consider some nonlinear system of equations $\mathcal{R}(\mathbf{q}) = \mathbf{0}$, $\mathcal{R} : \mathbb{R}^N \to \mathbb{R}^N$. The basic concept of homotopy continuation is that some other system of equations is defined which is easier to solve and the solution to this "easier" system is then deformed to the solution of $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ by varying some continuation parameter $\lambda \in \mathbb{R}$.

Consider the so-called *convex homotopy* [1] which is defined as the (presumably) continuous solution $\mathbf{q}(\lambda)$ to

$$\mathcal{H}(\mathbf{q}, \lambda) = (1 - \lambda)\,\mathcal{R}(\mathbf{q}) + \lambda\mathcal{G}(\mathbf{q}) = \mathbf{0}, \tag{3.29}$$

$$\mathcal{H} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N,\ \mathcal{G} : \mathbb{R}^N \to \mathbb{R}^N,\ \mathcal{R} : \mathbb{R}^N \to \mathbb{R}^N,\ \lambda \in \mathbb{R}.$$

A continuation method, called convex homotopy continuation (CHC), can be developed from this homotopy by discretizing in λ to form a sequence of nonlinear equations:

$$\mathcal{H}(\mathbf{q}, \lambda_k) = (1 - \lambda_k)\,\mathcal{R}(\mathbf{q}) + \lambda_k \mathcal{G}(\mathbf{q}) = \mathbf{0}, \tag{3.30}$$

$$k \in [0, m],\ \lambda_k \in \mathbb{R},\ \lambda_0 = 1,\ \lambda_m = 0,\ \lambda_{k+1} < \lambda_k.$$

Solving $\mathcal{H}(\mathbf{q}, \lambda) = \mathbf{0}$ for sequentially increasing k is referred to as *traversing*.

3.9.3 Global Homotopy Continuation

Global homotopy originated as a globally-convergent generalization of Newton's method, which was studied by Branin [13] and later Smale [155]. This method was referred to as a global Newton method and the connection to homotopy continuation was later made by Keller [77]. This led to the notion of global homotopy continuation (GHC). Global homotopy continuation is performed by sequentially solving

$$\mathcal{H}(\mathbf{q}, \lambda_k) = \mathcal{R}(\mathbf{q}) - \lambda_k \mathcal{R}(\mathbf{q}_0) = \mathbf{0}, \tag{3.31}$$

$$k \in [0, m],\ \lambda_k \in \mathbb{R},\ \lambda_0 = 1,\ \lambda_m = 0,\ \lambda_{k+1} < \lambda_k,$$

$$\mathcal{H} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N,\ \mathcal{R} : \mathbb{R}^N \to \mathbb{R}^N.$$

The vector $\mathbf{q}_0 \in \mathbb{R}^N$ can be any vector of choice. Some logical selections of $\mathbf{q}_0$ are the far-field conditions or an approximation to the solution which may have been obtained previously.

It is easily verified that the global homotopy system (3.31) can be written as the convex system (3.30) with homotopy system $\mathcal{G}(\mathbf{q}) = \mathcal{R}(\mathbf{q}) - \mathcal{R}(\mathbf{q}_0)$. Thus, GHC can be regarded as a special case of CHC.

3.10 Metrics for Comparing the Performance of Continuation Algorithms

The desired properties of a continuation algorithm are:

1. The CPU cost should be low;
2. The algorithm should be robust; and
3. User control should be as simple and intuitive as possible.

The first two items are typically competing objectives because there are typically some input parameters which, when adjusted, can make the algorithm converge in less time but will reduce the likelihood that the flow solution will be reached successfully, or vice versa. Constructing a fair comparison between algorithms must account for this.

Performance comparisons in this thesis are normally performed by running test cases over a wide range of Mach numbers and angles of attack. This is done to take into account algorithm robustness and also to avoid bias if one algorithm tends to perform better under certain operating conditions - for example, one algorithm might perform better at low Mach number. Some mild tuning of algorithm parameters is also performed to ensure that one algorithm is not under-performing simply because non-competitive parameters were chosen. However, the same set of parameters is used for all test cases in the

suite so that individual cases are not tuned and to keep the robustness comparison fair. The robustness is assessed for each algorithm in terms of the number of test cases which failed. The time to complete the flow solve is compared between algorithms where the algorithm parameters have been chosen such that a similar level of robustness is maintained for all algorithms in the comparison.

The CPU time taken to reduce the flow residual $\|\mathcal{R}(\mathbf{q})\|$ below a certain relative or absolute tolerance, when measured in seconds, depends on the hardware used to run the codes, the code compilers, and how efficiently the algorithm has been coded, none of which are the focus of this study. It is desirable to attempt to make the performance comparison independent of these artifacts.

One way to attempt to remove processor dependence is to consider relative residual evaluations instead of CPU time. A relative residual evaluation is the amount of time taken to compute the flow residual $\mathcal{R}(\mathbf{q})$ a single time. This is one of several approaches that has been taken for previous flow solver performance studies as carried out by Osusky and Zingg [131] or Brown et al. [14]. However, this metric depends on the cost of evaluating the residual so it is not always suitable for comparing the performance of different codes or flow solver performance under different spatial discretizations.

Another method for measuring flow solver performance is the TauBench system [35]. The TauBench code roughly simulates the CPU cost of running the DLR-developed flow solver Tau, which is a three-dimensional hybrid multigrid solver for the RANS equations. A benchmark timing factor is generated for the TauBench codes by specifying a grid size, number of processors, and number of iterative steps. While this does not account for specific processor dependence, all cases in this thesis were run on the SciNet general purpose cluster on computational hardware of consistent make and model and, in our experience, CPU time varies by less than 3% when comparing timing for the same flow solve on different computational nodes.

The TauBench benchmark that was used is a 2.50×10^5 node mesh run in serial with 10 iterative steps. This benchmark was chosen for consistency with other researchers who have employed this metric [14, 15, 167]. From running the TauBench code with these settings four times on the SciNet general purpose cluster and taking the average, we have calculated 9.571s as the Taubench benchmark, which we refer to as one *work unit* (wu).

Chapter 4

Homotopy Design and Analysis

This chapter presents some analysis of homotopy, including important concepts and calculations which can be used directly in the homotopy continuation algorithms. Several analysis tools are also presented which are used to assess the suitability of a homotopy system for use in a homotopy continuation algorithms or to evaluate the performance of a given homotopy continuation algorithm. The homotopies introduced in this section are investigated numerically from numerous perspectives.

4.1 Geometric Interpretation of Homotopies

If both $\mathcal{R}(\mathbf{q})$ and $\mathcal{G}(\mathbf{q})$ are continuous, then the solution points $\mathbf{q}_k$ of $\mathcal{H}(\mathbf{q}, \lambda_k) = \mathbf{0}$, when ordered sequentially, form a continuous curve in $\mathbb{R}^N$. If $\mathcal{H}(\mathbf{q}, 1) = \mathbf{0}$ and $\mathcal{H}(\mathbf{q}, 0) = \mathbf{0}$ have unique solutions then the solution to each $\mathcal{H}(\mathbf{q}, \lambda_k) = 0$ is (usually) unique [113] by the Implicit Function Theorem [67], which is given in Appendix A.[1] When considering convex homotopies,[2] since $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ has a unique solution by assumption, it remains to construct $\mathcal{G}(\mathbf{q})$ to have a unique solution as well.

The most intuitive way to interpret the homotopy curve is to consider the parametric curve $\mathbf{q} \in \left\{\mathcal{H}^{-1}(\mathbf{0}) \,|\lambda \in [0,1]\right\}$, $\mathbf{q} \in \mathbb{R}^N$, $(\lambda) \mapsto \mathbf{q}(\lambda)$. This homotopy curve will be referred to as having λ parametrization. However, it will often be useful to treat the homotopy curve as being of the form $c(s) \in \left\{\mathcal{H}^{-1}(\mathbf{0}, \lambda) \,|\lambda \in [0,1]\right\}$, $(s) \mapsto c(s)$, $c : \mathbb{R} \to \mathbb{R}^N \times \mathbb{R}$, where $c(s)$ is of the form $(\mathbf{q}(s); \lambda(s))$ and $s \in \mathbb{S}$, $\mathbb{S} \subset \mathbb{R}$, by which it is meant that s takes values in a proper subset of $\mathbb{R}$. The homotopy curve is always defined implicitly. It is never explicitly available in any parametric form and is written out explicitly in parametric form for analysis only.

There are unlimited ways in which the curve can be parametrized but the most common is to use an arclength parametrization:

Definition 4.1. *Let $c : \mathbb{R} \to \mathbb{R}^N \times \mathbb{R}$, $(s) \mapsto c(s)$ be a C^1-differentiable curve of the form $(\mathbf{q}(s); \lambda(s))$ with parameter $s \in \mathbb{S}$, $\mathbb{S} \subset \mathbb{R}$. The parametrization defined implicitly by*

$$\dot{c}(s) \cdot \dot{c}(s) = 1 \tag{4.1}$$

is called an arclength parametrization of $c(s)$ [84].

[1] The uniqueness property might not be satisfied if $\nabla_{\mathbf{q}} \mathcal{H}(\mathbf{q})$ is singular at any point on the curve, in which case bifurcation points can arise.

[2] Convex homotopies were introduced in Section 3.9.2.

A formal definition of the λ parametrization can also be given in this form.

Definition 4.2. *Let $c : \mathbb{R} :\to \mathbb{R}^N \times \mathbb{R}$, $(r) \mapsto c(r)$ be a C^1-differentiable curve of the form $(\mathbf{q}(r); \lambda(r))$ with parameter $r \in \mathbb{R}$, $r \in [0, 1]$. The parametrization defined by*

$$\dot{\lambda}(r) = -1 \tag{4.2}$$

is called a λ parametrization of $c(r)$.

From this point forward, the notation $c(s)$ indicates that the curve has an arclength parametrization and $c(r)$ indicates that the curve has a λ parametrization. When using an arclength parametrization, distance is measured along the curve, including λ as a curve variable. This can be a more relevant way to measure distance for some numerical tools such as step-length adaptation. However, both parametrizations can be useful for analysis.

4.2 Convex Homotopy System Design Objectives

The suitability of a given $\mathcal{G}(\mathbf{q})$ as a homotopy system depends on how well each of the following criteria are met:

1. The matrix $\nabla_{\mathbf{q}}\mathcal{G}(\mathbf{q})$ is well-conditioned and definite[3], and improves the conditioning and definiteness when added to the flow residual;
2. The solution to $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ is known or easily obtainable;
3. The solution to $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ is unique;
4. The homotopy connecting $\mathcal{G}^{-1}(\mathbf{0})$ and $\mathcal{R}^{-1}(\mathbf{0})$ should exhibit modest curvature.

Numerical techniques for tracing the curve and determining the sequence λ_k are of critical importance to the efficiency of homotopy continuation algorithms. During traversing, the curve is approximated numerically, and information will typically be limited to local information of the curve and usually the vector tangent to the curve. Hence, if the tangent vector does not change dramatically for small changes in λ then the curve will be easier to trace numerically. In other words, it is desirable that the curve should exhibit low *curvature*.

For convex homotopy, since curvature is inherently a result of the interaction of the nonlinearities of the systems $\mathcal{R}(\mathbf{q})$ and $\mathcal{G}(\mathbf{q})$, an analytical study of how to choose $\mathcal{G}$ such that the homotopy curve exhibits low curvature is a challenging problem. While the current analysis focuses on numerical studies of the curves produced by specific homotopy systems and is lacking in analytical work, this is only due to the challenging nature of this problem and our hope is that some analysis can be performed in the future to better understand the interaction between $\mathcal{R}$ and $\mathcal{G}$ and how this affects the homotopy.

4.3 Metrics for Evaluating Curve Traceability

Several definitions are given for different types of curvature:

[3]The definition of a definite matrix is given as Definition A.1 in Appendix A.

Definition 4.3. *Let $c : \mathbb{R} \to \mathbb{R}^N \times \mathbb{R}$, $(s) \mapsto c(s)$ be a C^2-differentiable curve of the form $c(\mathbf{q}(s), \lambda(s))$ with arclength parameter $s \in \mathbb{S}$, $\mathbb{S} \subset \mathbb{R}$. The total curvature [84] $\kappa : \mathbb{R} \to \mathbb{R}$, $(s) \mapsto \kappa(s)$ is defined as*

$$\kappa(s) = \sqrt{\ddot{c}(s) \cdot \ddot{c}(s)}, \tag{4.3}$$

and the partial curvature $\kappa_{\mathbf{q}} : \mathbb{R} \to \mathbb{R}$ is defined as

$$\kappa_{\mathbf{q}}(s) = \sqrt{\ddot{\mathbf{q}}(s) \cdot \ddot{\mathbf{q}}(s)}. \tag{4.4}$$

Definition 4.4. *Let $c : \mathbb{R} \to \mathbb{R}^N \times \mathbb{R}$, $(r) \mapsto c(r)$ be a C^2-differentiable curve of the form $c(\mathbf{q}(r), \lambda(r))$ with parameter $r \in \mathbb{R}$ such that $\dot{\lambda}(r) = -1$. The total curvature with λ parametrization $\kappa_r : \mathbb{R} \to \mathbb{R}$, $(r) \mapsto \kappa(r)$ is defined as*

$$\kappa_r(r) = \sqrt{\ddot{\mathbf{q}}(r) \cdot \ddot{\mathbf{q}}(r)}. \tag{4.5}$$

While it is clear that curvature is an important metric for evaluating curve traceability, the apparent curvature, and hence traceability, also depends on the parametrization. For example, if it is assumed that traversing is performed with constant step-length, where step-length is measured with respect to an arclength parametrization, then it is important to consider $\kappa_{\mathbf{q}}$. However, if it is assumed that the curve is traced with constant $\Delta\lambda$ then it is more relevant to consider κ_r. Since the use of λ as the parameter controlling the deformation is a matter of convenience and not performance, curve tracing algorithms are generally designed to attempt to maintain a relatively consistent Δs. However, in practice, λ is the parameter which is updated and the actual Δs associated with a given $\Delta\lambda$ can be determined with limited accuracy. Thus, both $\kappa_{\mathbf{q}}$ and κ_r are relevant performance metrics for the continuation algorithm.

Since $\kappa_{\mathbf{q}}$ is (mostly) independent of the parametrization, this is a useful tool for assessing the suitability of a homotopy system for use in a continuation algorithm under the assumption that the curve can be re-parametrized. However, it is not a good metric for directly comparing two homotopies because it assumes both curves are being traced with the same step size Δs and does not take into account that, under this condition, more steps would be needed if the curve is longer in the arclength sense.

To establish a more appropriate metric, consider the Taylor expansion around some s_0:

$$\mathbf{q}(s_0 + \Delta s) = \mathbf{q}(s_0) + \Delta s \dot{\mathbf{q}}(s_0) + \frac{1}{2}\Delta s^2 \ddot{\mathbf{q}}(s_0) + \mathcal{O}\left(\Delta s^3\right). \tag{4.6}$$

If a predictor is formed using only $c(s)$ and $\dot{c}(s)$, then, neglecting $\mathcal{O}\left(\Delta s^3\right)$ terms, the norm of the error $e \in \mathbb{R}$ resulting from the curvature is

$$e = \sqrt{\left(\frac{1}{2}\Delta s^2\right)^2 \ddot{\mathbf{q}}(s) \cdot \ddot{\mathbf{q}}(s)} = \frac{1}{2}\Delta s^2 \kappa_{\mathbf{q}}. \tag{4.7}$$

If it is assumed that the curve is always traced with the same number n_s of equally spaced (in Δs) steps then $s_{\text{tot}} = n_s \Delta s$ and hence equation (4.7) becomes

$$e = \frac{1}{2n_s^2} s_{\text{tot}}^2 \kappa_{\mathbf{q}}. \tag{4.8}$$

The actual value of n_s is irrelevant for comparison so $s_{\text{tot}}^2 \kappa_{\mathbf{q}}$ is the traceability metric considered when assuming an arclength parametrization. This quantity should be plotted against s/s_{tot}.

While the traceability metrics presented in this section are useful for comparing homotopies on a given mesh under certain flow conditions, the curvature can scale in complicated ways with mesh size, local grid refinement, and the state variables $\mathbf{q}$. Hence the traceability metrics should not, in general, be used to compare traceability across meshes or under different flow conditions. An exception can be made if the mesh is refined in a consistent way; such a study is performed in Section 4.9.6.

4.4 Some Specific Homotopy Systems

The homotopy systems presented in this section are obvious choices in that they easily satisfy the first three requirements listed above. That is, they are very well-conditioned, $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ can be solved easily, and they can be constructed to have a unique solution. The curvature profiles are investigated numerically.

4.4.1 Diagonal Operator

A linear operator $\mathcal{G}(\mathbf{q})$ with the property $(\mathcal{G}(\mathbf{q}))_{[i]} = g_i \mathbf{q}_{[i]}$, $g_i > 0$, $g_i \in \mathbb{R}$ is a suitable homotopy system because the Jacobian of this system is a diagonal positive definite matrix. As such, the Jacobian is nonsingular, can be inverted trivially, and will improve the diagonal dominance of the Jacobian when added to the discrete flow equations. Because the Jacobian is diagonal, this homotopy system will be referred to as the "Diagonal" operator.

By analogy to the Jacobian formed when applying a pseudo-transient method using equation (3.26), the equation blocks of this homotopy system for the three-dimensional RANS equations take the form:

$$\mathcal{G}_{(i)}(\mathbf{q}) = \left(\frac{1 + \mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1 + \mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1 + \mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1 + \mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1 + \mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, 1 + \mathcal{J}_{[i]}^{\frac{1}{D}} \right) \mathbf{q}_{(i)}, \tag{4.9}$$

where $\mathcal{J}$ is the metric Jacobian as given by equation (2.12). The rounded brackets in the subscript (i) indicate the sub-vector corresponding to the ith grid node and the square brackets $[j]$ indicate the jth component of the vector. The parameter D is the number of spatial dimensions and is equal to 3 for three-dimensional flows. For two-dimensional flows, D is set to 2, and the equation block size is reduced by one by deleting the fourth entry of equation (4.9). This homotopy system only differs from the pseudo time operator of the pseudo-transient method in that it does not contain a factor of the reference time step Δt_{ref}. As such, adding the Jacobian of this homotopy system to the flow Jacobian will have the effect of increasing the positive definiteness (shifting the eigenvalues further to to the right of the imaginary axis) and improving the conditioning, as can be seen from the eigenvalue analysis of Hicken and Zingg [61].

While this operator has an easily invertible Jacobian, the solution to $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ is $\mathbf{q} = \mathbf{0}$, which includes zero density and is non-physical. This is obviously a poor choice of homotopy system so it is preferable to apply this operator as a warm-started homotopy operator as described in Section 4.4.3. This formulation allows for any specified value of $\mathbf{q}$, such as far-field conditions, to satisfy the modified homotopy system without affecting the system Jacobian.

4.4.2 Dissipation Operator

Recall the artificial dissipation operators introduced in Section 2.6 which are added to the discrete flow equations for numerical stability and shock capturing. Adding a dissipation operator residual of this form to the flow residual increases diagonal dominance and definiteness of the Jacobian of the combined system. This can directly improve the convergence rate of GMRES and can also improve the convergence rate of preconditioned GMRES because it can improve the effectiveness of the ILU(p) preconditioner [148]. Because the eigenvalues are shifted away from the imaginary axis convergence of the nonlinear system can also be enhanced [61].

Since the homotopy system is used only for continuation, it does not affect the flow solution accuracy so it is preferable to use the smaller-stencil second-difference (first-order) dissipation operator, without the pressure sensor. The Jacobian of this operator will fit within the three-point stencil used in the preconditioner matrix, allowing for the Jacobian to be preconditioned more effectively as it will be represented more accurately by the block ILU(p) preconditioner.

Recall the forward differencing operator $\Delta_\xi : \mathbb{R}^N \to \mathbb{R}^{N-1}$ from equations (2.18) and (2.19) which is used in the construction of the second-difference dissipation operator $\mathcal{D}^{(2)} : \mathbb{R}^N \to \mathbb{R}^N$. That Δ_ξ defines a mapping from $\mathbb{R}^N$ to $\mathbb{R}^{N-1}$ is sufficient to identify that the range of $\mathcal{D}^{(2)}$ is spanned by at most $N-1$ linearly independent vectors. Therefore $\mathcal{D}^{(2)}$ is singular and the deformation will not be a regular homotopy. This can be remedied by augmenting $\mathcal{D}^{(2)}$ with pseudo boundary conditions.

Pseudo boundary conditions for $\mathcal{D}^{(2)}$ are formulated using the SAT approach to be consistent with the application of the boundary conditions for the discrete flow equations. The scalar one-dimensional version of the operator is analyzed for clarity of presentation. By analogy to the diffusion equation, the operator, including boundary conditions, is assumed to be of the following form:

$$\mathcal{G}(\mathbf{q}) = \mathcal{D}^{(2)}\mathbf{q} + \Sigma(\mathbf{q}), \tag{4.10}$$

$$\Sigma(\mathbf{q}) = \operatorname{diag}\left(\sigma_{\mathrm{L}}\left(\mathbf{q}_{[1]} - q_{\mathrm{L}}\right), 0, \ldots, 0, \sigma_{\mathrm{R}}\left(\mathbf{q}_{[N]} - q_{\mathrm{R}}\right)\right),$$

$$\Sigma : \mathbb{R}^N \to \mathbb{R}^N,\ \sigma_{\mathrm{L}}, \sigma_{\mathrm{R}}, q_{\mathrm{L}}, q_{\mathrm{R}} \in \mathbb{R}.$$

At a domain boundary, q_{L} and q_{R} are boundary conditions. At block interfaces, they are the flow values at the same point in physical space but corresponding to the adjacent block. Discussed in the following paragraphs are necessary conditions on the scalars σ_{L} and σ_{R} which ensure that the Jacobian of the dissipation operator is regular and definite, assuming that the spectral radius is constant and positive.

Since $\mathcal{D}^{(2)}$ is not an SBP operator, it is not straightforward to derive the stability condition on the SATs using the usual energy method. However, by treating the spectral radius as constant, it is possible to determine some conditions on the SATs by analysis of the pseudo-linear operator representing the Jacobian. For analysis, this operator is most conveniently expressed by giving the expression for the rows in the one-dimensional case:[4]

$$\mathcal{G}^{(l)}_{[i,:]} = \begin{cases} \left(\sigma_{\mathrm{L}} + d_{\left[\frac{3}{2}\right]}, -d_{\left[\frac{3}{2}\right]}, 0, 0, \ldots, 0\right) & i = 1, \\ \operatorname{trid}_i\left(-d_{\left[i-\frac{1}{2}\right]}, d_{\left[i-\frac{1}{2}\right]} + d_{\left[i+\frac{1}{2}\right]}, -d_{\left[i+\frac{1}{2}\right]}\right) & i = 2, \ldots, N-1, \\ \left(0, \ldots, 0, 0, -d_{\left[N-\frac{1}{2}\right]}, d_{\left[N-\frac{1}{2}\right]} + \sigma_{\mathrm{R}}\right) & i = N, \end{cases} \tag{4.11}$$

[4] In this context, $\operatorname{trid}_i(x, y, z)$ refers to the ith row of a matrix with x at the $i-1$st entry, y at the ith entry, z at the $i+1$st entry, and zeros everywhere else.

$$d_{i+\frac{1}{2}} = \frac{1}{2}\left(d_i + d_{i+1}\right),\ d_i = \frac{1}{\Delta x_i}\left(|u_i| + a_i\right).$$

Consider the subtraction of the sum of the absolute values of the off-diagonal elements of the matrix from the absolute value of the diagonal elements:

$$\left|\mathcal{G}^{(l)}_{[i,i]}\right| - \sum_{j\neq i}\left|\mathcal{G}^{(l)}_{[i,j]}\right| = \begin{cases} \sigma_{\mathrm{L}} & i = 1, \\ 0 & i = 2,\ldots,N-1, \\ \sigma_{\mathrm{R}} & i = N. \end{cases} \tag{4.12}$$

Since $\mathcal{G}^{(l)}$ is easily verified to be irreducible, the necessary and sufficient condition for $\mathcal{G}^{(l)}$ to be irreducibly row-diagonally dominant[5] is

$$\sigma_{\mathrm{R}} \geq 0,\ \sigma_{\mathrm{L}} \geq 0,\ \sigma_{\mathrm{L}} + \sigma_{\mathrm{R}} > 0. \tag{4.13}$$

Since irreducibly row-diagonally dominant systems are nonsingular [148], condition (4.13) is also a sufficient condition for non-singularity of $\mathcal{G}^{(l)}$.

By symmetry, and comparison with the well-known diffusion operator, it is inferred that the conditions for well-posedness are given by $\sigma_{\mathrm{L}} = d_1$ and $\sigma_{\mathrm{R}} = d_N$ for the scalar one-dimensional version of the operator. The extension to three-dimensional vector-valued systems is accomplished by similarly constructing the SATs in each direction and for each equation.

The boundary conditions q_{L} and q_{R} do not affect conditioning of the linear system and can be chosen based on other criteria. One benefit to setting q_{L} and q_{R} to far-field conditions at all domain boundaries is that $\mathcal{G}\left(\mathbf{q}_{\mathrm{ff}}\right) = \mathbf{0}$, where $\mathbf{q}_{\mathrm{ff}} \in \mathbb{R}^N$ is the vector consisting of the far-field values in all corresponding elements. Use of "flow-imitative" boundary conditions is also considered, where the boundary conditions are set at the solid surfaces, far-field boundaries, and symmetry planes which imitate the SATs corresponding to the flux terms of $\mathcal{R}\left(\mathbf{q}\right)$.

4.4.3 Warm-Started Homotopy Systems

The concept of "warm-starting" a nonlinear algorithm applies to fixed-point methods such as Newton's method or PTC where a good initial guess $\mathbf{q}_0$ can reduce the total number of iterations taken by the algorithm and thus reduce the total CPU time needed for convergence. Global homotopy continuation can naturally take an initial guess $\mathbf{q}_0$ but convex homotopies cannot because $\mathbf{q}_0$ will not correspond to a point on the curve. However, an effect similar to warm-starting can be obtained in the context of convex homotopy for a given homotopy system $\mathcal{G}$ by constructing a modified homotopy system

$$\mathcal{G}^*\left(\mathbf{q}\right) = \mathcal{G}\left(\mathbf{q}\right) - \mathcal{G}\left(\mathbf{q}_0\right) \tag{4.14}$$

since $\mathbf{q} = \mathbf{q}_0$ is a solution to $\mathcal{G}^*\left(\mathbf{q}\right) = \mathbf{0}$.

4.5 Tangent Vector

A formal definition of the tangent vector $t \in \mathbb{R}^{N+1}$ is given in Allgower and Georg [1]. This definition is adopted here but with a different convention for the third condition:

[5] A formal definition of an irreducibly row-diagonally dominant matrix is given in Appendix A.

Definition 4.5. *Let* $\mathcal{A} \in \mathbb{R}^{N\times(N+1)}$ *with* $\operatorname{rank}(\mathcal{A}) = N$. *The unique vector* $t \in \mathbb{R}^{N+1}$ *satisfying the three conditions:*

(1) $\mathcal{A}t = \mathbf{0}$; *(2)* $\|t\| = 1$; *(3)* $\det \begin{pmatrix} \mathcal{A} \\ t^T \end{pmatrix} < 0$;

is called the tangent vector induced by $\mathcal{A}$.

The first condition in Definition 4.5 gives the tangent direction, the second condition gives its magnitude. The third condition fixes the orientation of the tangent to ensure that the curve is traversed in a consistent direction. The convention adopted by Allgower and Georg [1] is to fix a positive orientation. However, it is more convenient for us to fix a negative orientation since we are assuming that $\mathcal{A}$ is positive definite. This is explained in Section 4.5.1.

There are several ways in which the tangent vector can be calculated. The first condition states that the tangent induced by $\mathcal{A}$ is in the kernel of $\mathcal{A}$. The kernel of $\mathcal{A}$ can be determined using either a singular value decomposition [54] or a $\mathcal{QR}$ factorization. Since the singular value decomposition is expensive and requires the formation of dense matrices it is not considered here.

Consider, for $\mathcal{A} \in \mathbb{R}^{N\times(N+1)}$, the $\mathcal{QR}$ decomposition of $\mathcal{A}^T$:

$$\mathcal{A}^T = \bar{\mathcal{Q}} \begin{pmatrix} \bar{\mathcal{R}} \\ 0^T \end{pmatrix}, \tag{4.15}$$

where $\bar{\mathcal{Q}} \in \mathbb{R}^{(N+1)\times(N+1)}$ is an orthonormal matrix, and $\bar{\mathcal{R}} \in \mathbb{R}^{N\times N}$. Using $\bar{\mathcal{Q}}^{-1} = \bar{\mathcal{Q}}^T$ and performing some minor algebra:

$$\mathcal{A}\bar{\mathcal{Q}} = \begin{pmatrix} \bar{\mathcal{R}}^T & 0 \end{pmatrix}. \tag{4.16}$$

Assuming that $\dim(\ker(\mathcal{A})) = 1$, then $\bar{\mathcal{R}}$ is nonsingular. Hence, the tangent vector t can be extracted from the last column of $\bar{\mathcal{Q}}$.

The three most common methods for calculating a $\mathcal{QR}$ decomposition are Householder reflections [69], Givens rotations [51, 127], or the modified Gram-Schmidt method [36], all of which can be parallelized for large sparse matrices. Out of the three, Givens rotations makes the best use of sparsity [127], particularly when considering the version of Gentleman [46]. However, the tools to compute a full $\mathcal{QR}$ decomposition of the Jacobian matrix are not readily available to most CFD flow solvers and from experimentation for some simple one-dimensional homotopy problems we have found these methods to be too slow to be used in a cost-competitive algorithm.

A more efficient method for calculating the tangent is presented here. This method is a special case of the method presented by Rheinboldt [144], who did not give consideration for sparse matrices. For the special case of homotopies as presented in this thesis, the tangent vector induced by $\nabla\mathcal{H}(c(s))$ is given by

$$t = \frac{\tau}{\|\tau\|}, \quad \tau = \begin{pmatrix} \mathbf{z} \\ -1 \end{pmatrix}, \quad \mathbf{z} = \left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q},\lambda)\right]^{-1} \frac{\partial}{\partial\lambda}\mathcal{H}(\mathbf{q},\lambda), \tag{4.17}$$

$$t \in \mathbb{R}^{N+1}, \quad \mathbf{z} \in \mathbb{R}^{N}, \quad \tau \in \mathbb{R}^{N+1},$$

where $\frac{\partial}{\partial\lambda}\mathcal{H}(\mathbf{q},\lambda) = \mathcal{G}(\mathbf{q}) - \mathcal{R}(\mathbf{q})$ for convex homotopy and $\frac{\partial}{\partial\lambda}\mathcal{H}(\mathbf{q},\lambda) = -\mathcal{R}(\mathbf{q}_0)$ for global homotopy. Two derivations are provided for equation (4.17), as each illustrates some key features of this important equation.

4.5.1 Derivation of Equation (4.17) by an Algebraic Method

This derivation is similar to Rheinboldt [144] or Allgower and Georg [1] and emphasizes that the tangent calculation has been constructed to preserve sparsity and, to some degree, linear system conditioning.

Let $\mathbf{x} \in \mathbb{R}^{N+1}$ satisfy

$$\nabla \mathcal{H}\left(c\left(s\right)\right) \mathbf{x} = \nabla \mathcal{H}\left(c\left(s\right)\right) \mathbf{y} \tag{4.18}$$

for some nonzero $\mathbf{y} \in \mathbb{R}^{N+1}$. If

$$\tau = \mathbf{y} - \mathbf{x}, \tag{4.19}$$

then $\tau \in \mathbb{R}^{N+1}$ is in the kernel of $\nabla \mathcal{H}\left(c\left(s\right)\right)$. This is easily verified:

$$\nabla \mathcal{H}\left(c\left(s\right)\right) \tau = \nabla \mathcal{H}\left(c\left(s\right)\right) \left[\mathbf{y} - \mathbf{x}\right] = \mathbf{0}. \tag{4.20}$$

Clearly the tangent vector given by

$$t = \pm \frac{\tau}{\|\tau\|} \tag{4.21}$$

satisfies the first two conditions of Definition 4.5 and the sign can be chosen so that the tangent is oriented in the direction of traversing.

While the only condition given so far for $\mathbf{y} \in \mathbb{R}^{N+1}$ is that it should be nonzero, some choices of nonzero $\mathbf{y}$ may cause the under-determined system given by equation (4.18) to have no solutions and some choices may result in a linear system which is poorly conditioned or expensive to solve due to the presence of dense rows or columns. The additional equation needed to make the linear system (4.18) fully determined will affect the solution to the linear system of equations but not the final value of the tangent vector, so long as the linear system is nonsingular and τ is nonzero.

For simplicity, let $\mathbf{y} = \mathbf{e}_i$, where $\mathbf{e}_i$ satisfies $\mathbf{e}_i \cdot \mathbf{x} = \mathbf{x}_{[i]}$, e.g. $\mathbf{e}_i$ is a vector containing a 1 at the ith position and zeros elsewhere. If $\mathbf{e}_i \cdot \mathbf{x} = 0$ is used as the final equation then $\mathbf{e}_i \cdot \tau = \mathbf{e}_i \cdot \mathbf{e}_i - 0 > 0$, so $\tau \neq \mathbf{0}$, and hence the solution to the linear system is nonzero.

Some authors [1, 144] have advocated setting i to the maximum element of the tangent vector from a previous iteration to attempt to make $\mathbf{e}_i$ as parallel as possible to the kernel of $\nabla \mathcal{H}\left(c\left(s\right)\right)$. If $\mathbf{e}_i$ is parallel to the kernel of $\nabla \mathcal{H}\left(c\left(s\right)\right)$ then it must be normal to all of the row vectors of $\nabla \mathcal{H}\left(c\left(s\right)\right)$, so it is intuitive that selecting i this way is more likely to result in a well-conditioned matrix. For large sparse systems, however, there is significant benefit to choosing $i = N + 1$. In this case, $\mathbf{e}_i \cdot \mathbf{x} = 0$ can be expressed simply as $\mathbf{x}_{[i]} = 0$, making it unnecessary to solve for $\mathbf{x}_{[N+1]}$ and allowing for the $N + 1$st column of $\nabla \mathcal{H}\left(c\left(s\right)\right)$ to be deleted, which is critical since this column corresponds to $\frac{\partial}{\partial \lambda} \mathcal{H}\left(c\left(s\right)\right)$ and is expected to be dense. The resulting matrix $\nabla_{\mathbf{q}} \mathcal{H}\left(c\left(s\right)\right)$ is nonsingular by the regularity assumption and should be relatively well-conditioned, assuming that the homotopy system Jacobian is well-conditioned.

Choosing $i = N + 1$, the system (4.18) reduces to:

$$\nabla_{\mathbf{q}} \mathcal{H}\left(c\left(s\right)\right) \mathbf{z} = \frac{\partial}{\partial \lambda} \mathcal{H}\left(c\left(s\right)\right), \tag{4.22}$$

where $\mathbf{z} \in \mathbb{R}^N$ is obtained by deleting the $N + 1$st entry of $\mathbf{x} \in \mathbb{R}^{N+1}$. Furthermore, combining equations (4.19) and (4.21) gives

$$t = \pm \frac{\tau}{\|\tau\|}, \qquad \text{where} \quad \tau = \begin{pmatrix} \mathbf{z} \\ -1 \end{pmatrix}. \tag{4.23}$$

It is intuitive that the sign of t should be chosen to be positive, since this will give $\lambda_{k+1} < \lambda_k$. If it is assumed that $\nabla\mathcal{H}(c(s))$ is positive definite then it can be shown analytically that the orientation of the tangent is fixed:

$$\begin{aligned}
\det\begin{pmatrix} \nabla_{\mathbf{q}}\mathcal{H}(c(s)) & \frac{\partial}{\partial\lambda}\mathcal{H}(c(s)) \\ \mathbf{z}^T & -1 \end{pmatrix} &= \det\begin{pmatrix} \nabla_{\mathbf{q}}\mathcal{H}(c(s)) & \nabla_{\mathbf{q}}\mathcal{H}(c(s))\,\mathbf{z} \\ \mathbf{z}^T & -1 \end{pmatrix} \\
&= \det\begin{pmatrix} \nabla_{\mathbf{q}}\mathcal{H}(c(s)) & 0 \\ \mathbf{z}^T & \mathbf{z}\cdot\mathbf{z}+1 \end{pmatrix} \det\begin{pmatrix} \mathcal{I} & \mathbf{z} \\ \mathbf{0}^T & -1 \end{pmatrix} \\
&= -(1+\mathbf{z}\cdot\mathbf{z})\det(\nabla_{\mathbf{q}}\mathcal{H}(c(s))) \\
&< 0. \qquad (4.24)
\end{aligned}$$

So, as it turns out, by the convention of Allgower and Georg [1], the curves traced by our algorithms are all traced with negative orientations.

This analysis also indicates that if $\det(\nabla_{\mathbf{q}}\mathcal{H}(c(s))) < 0$ then the negative sign must be chosen in front of the tangent vector, in which case $\dot{\lambda}(s)$ must become positive. This is an interesting result because it shows that if $\nabla\mathcal{G}(\mathbf{q}(s))$ is positive definite, which by construction is normally the case, then any homotopy between the solutions to $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ and $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ where $\mathcal{R}(\mathbf{q})$ has the property $\det(\nabla\mathcal{R}(\mathbf{q})) < 0$ must violate the regularity assumption and contain at least one bifurcation. While not all unstable systems have the property $\det(\nabla\mathcal{R}(\mathbf{q})) < 0$, this result does not lend much confidence in the ability of the algorithm to converge to unstable flow solutions in the absence of bifurcation point detection capability.

4.5.2 Derivation of Equation (4.17) through Differential Geometry

This derivation emphasizes the relationship between the tangent vector and its parametrization. Consider the curve defined implicitly by the homotopy:

$$\mathcal{H}(c(s)) = \mathbf{0}, \qquad (4.25)$$

where the curve $c(s) = (\mathbf{q}(s); \lambda(s))$, $c : \mathbb{R} \to \in \mathbb{R}^N \times \mathbb{R}$ has an arclength parametrization. Differentiating both sides of equation (4.25) with respect to arclength parameter s gives:

$$\nabla\mathcal{H}(c(s))\,\dot{c}(s) = \mathbf{0}, \qquad (4.26)$$

which can also be written:

$$\nabla_{\mathbf{q}}\mathcal{H}(c(s))\,\dot{\mathbf{q}}(s) + \dot{\lambda}(s)\frac{\partial}{\partial\lambda}\mathcal{H}(c(s)) = \mathbf{0}. \qquad (4.27)$$

Rearranging:

$$\nabla_{\mathbf{q}}\mathcal{H}(c(s))\left[\frac{-1}{\dot{\lambda}(s)}\dot{\mathbf{q}}(s)\right] = \frac{\partial}{\partial\lambda}\mathcal{H}(c(s)). \qquad (4.28)$$

As before, let $\mathbf{z} \in \mathbb{R}^N$ satisfy equation (4.22). Then

$$\dot{\mathbf{q}} = -\dot{\lambda}\mathbf{z} \qquad (4.29)$$

and

$$\dot{c}(s) \cdot \dot{c}(s) = \dot{\mathbf{q}}(s) \cdot \dot{\mathbf{q}}(s) + \dot{\lambda}(s)\,\dot{\lambda}(s) = \dot{\lambda}^2 [\mathbf{z} \cdot \mathbf{z} + 1]. \tag{4.30}$$

Notice that $\sqrt{\mathbf{z} \cdot \mathbf{z} + 1}$ is equal to $\|\tau\|$, as defined in equation (4.17). Since $\dot{c}(s) \cdot \dot{c}(s) = 1$ by the definition of the arclength parameter s, equation (4.30) can be used to obtain an equation for $\dot{\lambda}(s)$:

$$\dot{\lambda}(s) = \frac{-1}{\sqrt{\mathbf{z} \cdot \mathbf{z} + 1}} = \frac{-1}{\|\tau\|}, \tag{4.31}$$

where the negative sign has been included to force a negative orientation for $\dot{\lambda}(s)$. Substituting this back into equation (4.28) gives the expression for $\dot{\mathbf{q}}(s)$:

$$\dot{\mathbf{q}}(s) = -\dot{\lambda}\mathbf{z} = \frac{\mathbf{z}}{\|\tau\|}. \tag{4.32}$$

The combination of equations (4.31) and (4.32) completes the derivation and is consistent with equation (4.17).

The main purpose of providing this second derivation is to demonstrate the dependence of the tangent on the scaling of the elements of the $\mathbf{q}$ vector, and to interpret this dependence in a meaningful way. It is clear, for example, from equation (4.31) that increasing the magnitude of individual elements of $\mathbf{q}$ through scaling will reduce the size of $\dot{\lambda}(s)$ in the normalized tangent vector. Scaling of variables does not change the direction of the tangent in any meaningful way (the tangent is coordinate-independent [84]), but it does redefine the arclength parametrization, the effect being that the length measured along the curve is more sensitive to variables that have greater magnitude. While the mean-flow equations have already been scaled to have consistent magnitude, the turbulence variable $\tilde{\nu}$ can vary by several orders of magnitude throughout the flow field and can be significantly larger than the mean-flow variables; hence some additional scaling should be applied for effective curve tracing.

The arclength parameter is also affected by the scaling of λ relative to $\mathbf{q}$. If λ takes large values relative to $\mathbf{q}$, then traversing the homotopy curve with constant step length, for example, would result in nearly constant $\Delta\lambda$, and if λ takes on very small values relative to $\mathbf{q}$ then λ would be nearly ignored in determining the step size. These are important considerations for numerical curve tracing. Since these effects are completely ignored in the formulation of convex and global homotopy, equations (3.30) and (3.31) are modified in Section 5.5 to attempt to calibrate λ with respect to $\mathbf{q}$ more appropriately.

4.5.3 Validation of the Tangent Calculation

The validation is performed for inviscid flow over the two-dimensional NACA 0012 airfoil at Mach 0.3 and an angle of attack of 1°. Grid Ne is used[6], which has an H topology and consists of 15390 nodes divided evenly into 18 blocks. The second-difference dissipation operator with far-field boundary conditions is used as the homotopy system.

Since the tangent vector cannot be calculated analytically, finite-differencing is used for the validation. The solution is needed at two consecutive points along the curve, which are obtained using the predictor-corrector method of Chapter 5 - the details of the predictor-corrector method are not important for this analysis. The change in the arclength Δs is estimated by taking the standard norm of the difference in

[6] A list of all grids used in this thesis are catalogued in Appendix B

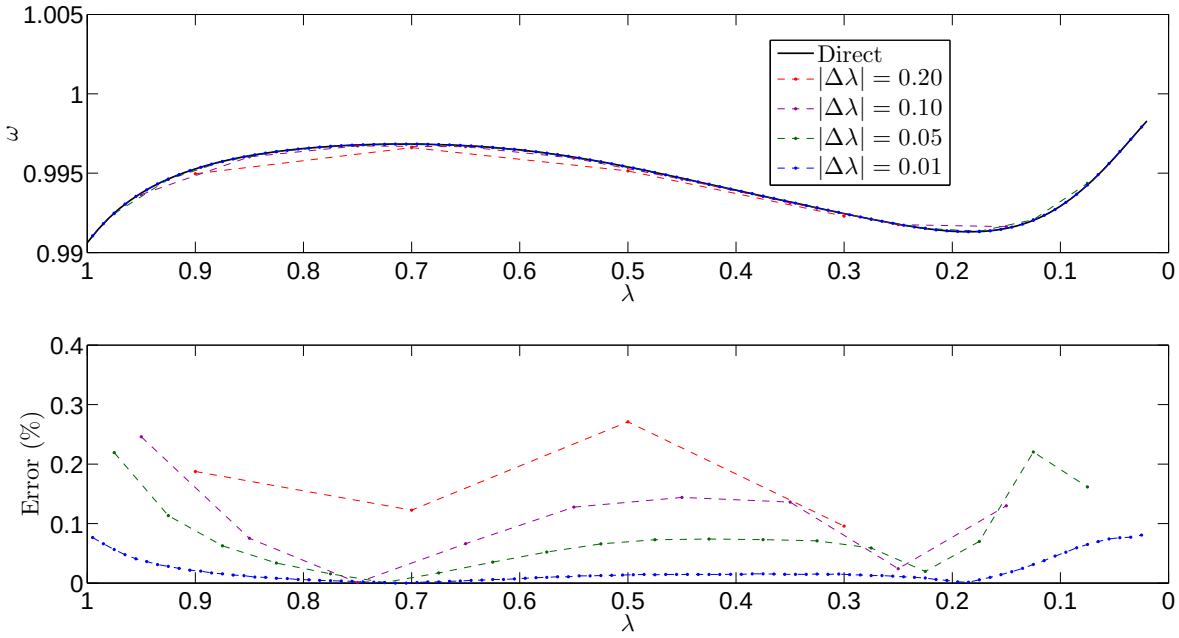


Figure 4.1: Comparison of ω calculated using the direct and finite-difference (FD) method with varying step size $|\Delta\lambda|$ on the inviscid NACA 0012 airfoil at Mach 0.3 and an angle of attack of 1°

solution between these points:

$$\Delta s_i \approx \Delta s_i^* = \sqrt{(\mathbf{q}_{i+1} - \mathbf{q}_i) \cdot (\mathbf{q}_{i+1} - \mathbf{q}_i) + (\lambda_{i+1} - \lambda_i)^2}. \tag{4.33}$$

A centred-difference estimate of the tangent vector is constructed by the equation

$$\dot{c}\left(s_{i+\frac{1}{2}}\right) \approx \dot{c}^*\left(s_{i+\frac{1}{2}}\right) = \frac{1}{\Delta s_i^*}\left(c\left(s_{i+1}\right) - c\left(s_i\right)\right). \tag{4.34}$$

Since $\dot{c}(s) \cdot \dot{c}(s) = 1$, this check is analogous to comparing $\left|\dot{\lambda}(s)\right|$ using the direct or finite-differencing method.

The validation is performed by determining whether the finite-difference estimates of $\omega \equiv \sqrt{\dot{\mathbf{q}}(s) \cdot \dot{\mathbf{q}}(s)}$ converge to the ω values obtained from the direct method in the limit of the finite-difference step size $|\Delta\lambda|$ going to 0. The finite-difference approximation is interpreted as representing the derivative at $s_{i+\frac{1}{2}} \approx \frac{1}{2}(s_{i+1} + s_i)$ so when the error is calculated the values of ω calculated from the direct method are interpolated at these points.

The curve points are solved for quite accurately in this study by enforcing an absolute tolerance of $\|\mathcal{H}(c(s))\| < 10^{-8}$ in the corrector phase of the predictor-corrector method and by solving the linear systems appearing in the tangent calculation to a relative tolerance of 10^{-8} using GCROT(m, k). Finite-differencing is used to form all matrix-vector products in this calculation. The comparison is shown in Figure 4.1, which demonstrates that the finite-difference estimates do converge to the direct tangent estimate in the limit of $|\Delta\lambda|$ vanishing.

4.6 Higher Derivatives of Implicitly-Defined Curves

Higher curve derivatives can be used for analysis of homotopies, as well as in the construction of higher order predictors. While higher curve derivatives have been calculated in the past for homotopies of very small scale [134, 153], direct application of these methods would require expensive dense matrix operations which would be prohibitively expensive for the scale of the homotopies studied in this thesis. In this section the derivation and validation of an efficient new method for calculating curve derivatives of any order is presented.

4.6.1 The Curvature Vector

As with the tangent vector, the curvature vector will depend on the parametrization. Carrying over from the tangent calculation, an arclength parametrization is assumed. The derivation begins by differentiating both sides of equation (4.26), which gives

$$\nabla \mathcal{H}\left(c\left(s\right)\right) \ddot{c}\left(s\right) + \nabla^2 \mathcal{H}\left(c\left(s\right)\right)\left[\dot{c}\left(s\right), \dot{c}\left(s\right)\right] = \mathbf{0}. \tag{4.35}$$

For clarity of presentation, let $\mathbf{w}_2 = \nabla^2 \mathcal{H}\left(c\left(s\right)\right)\left[\dot{c}\left(s\right), \dot{c}\left(s\right)\right]$ and notice that $\mathbf{w}_2 \in \mathbb{R}^N$ can be approximated by equation (3.22). Equation (4.35) can be expanded to

$$\nabla_{\mathbf{q}} \mathcal{H}\left(c\left(s\right)\right) \ddot{\mathbf{q}}\left(s\right) + \ddot{\lambda} \frac{\partial}{\partial \lambda} \mathcal{H}\left(c\left(s\right)\right) = -\mathbf{w}_2. \tag{4.36}$$

Equation (4.22) can be used to simplify:

$$\nabla_{\mathbf{q}} \mathcal{H}\left(c\left(s\right)\right) \left[\ddot{\mathbf{q}} + \ddot{\lambda}\mathbf{z}\right] = -\mathbf{w}_2. \tag{4.37}$$

Let

$$\mathbf{z}_2 = \ddot{\mathbf{q}} + \ddot{\lambda}\mathbf{z}, \tag{4.38}$$

$\mathbf{z}_2 \in \mathbb{R}^N$. It is possible to solve the linear system (4.37) for $\mathbf{z}_2$. However, an additional equation will be needed to retrieve all $N+1$ initial unknowns. As with the tangent calculation, this equation comes from the parametrization. Differentiating both sides of the arclength definition given by equation (4.1) gives the new equation

$$\ddot{c}\left(s\right) \cdot \dot{c}\left(s\right) = 0 \tag{4.39}$$

which, when expanded, can be written in terms of $\ddot{\mathbf{q}}$, $\ddot{\lambda}$, and the vector $\mathbf{z}$ previously calculated during the tangent calculation:

$$\ddot{\mathbf{q}} \cdot \mathbf{z} - \ddot{\lambda} = 0. \tag{4.40}$$

To solve for $\ddot{\lambda}$, take the dot product $\mathbf{z}_2 \cdot \mathbf{z}$ and use equations (4.38) and (4.40):

$$\mathbf{z}_2 \cdot \mathbf{z} = \ddot{\mathbf{q}} \cdot \mathbf{z} + \ddot{\lambda}\mathbf{z} \cdot \mathbf{z} = \ddot{\lambda}\left(\mathbf{z} \cdot \mathbf{z} + 1\right). \tag{4.41}$$

This expression is rearranged to obtain

$$\ddot{\lambda} = \frac{\mathbf{z}_2 \cdot \mathbf{z}}{\mathbf{z} \cdot \mathbf{z} + 1}. \tag{4.42}$$

Finally, equation (4.38) is used to retrieve the vector $\ddot{\mathbf{q}}$:

$$\ddot{\mathbf{q}} = \mathbf{z}_2 - \ddot{\lambda}\mathbf{z}. \tag{4.43}$$

4.6.2 Validation of the Curvature Calculation

As with the validation for the tangent calculation, the test case is inviscid flow over the two-dimensional NACA 0012 airfoil on grid Ne. Two cases are investigated: the first is a subsonic case at Mach 0.3 and an angle of attack of 1°, the second is a transonic case at Mach 0.8 and an angle of attack of 3°. In both cases, the second-order dissipation operator is used as the homotopy system.

The backwards-difference estimate of the second derivative is obtained by dividing the difference in the tangent vector calculated at the current point and immediately previous point along the curve by Δs^*:

$$\ddot{c}\left(s_{i+\frac{1}{2}}\right) \approx \ddot{c}^*\left(s_{i+\frac{1}{2}}\right) = \frac{1}{\Delta s_i^*}\left(\dot{c}\left(s_{i+1}\right) - \dot{c}\left(s_i\right)\right). \tag{4.44}$$

The validation is performed by determining whether the finite-difference estimates of $\kappa_{\mathbf{q}}$ converge to the $\kappa_{\mathbf{q}}$ values obtained from the direct method in the limit of the finite-difference step size $|\Delta\lambda|$ going to 0. From Figure 4.2, it is apparently the case. To explain the smaller $\kappa_{\mathbf{q}}$ values for the transonic case, it is because the arclength parametrization causes $\kappa_{\mathbf{q}}$ to be proportional to $1/\sqrt{\mathbf{q}\cdot\mathbf{q}}$ and this case is at a higher Mach number.

Note that once the finite-difference step size $|\Delta\lambda|$ becomes sufficiently small, the error in the finite-difference approximation to $\kappa_{\mathbf{q}}$ will begin to increase with decreasing step size. This can be explained by considering the error vectors $\mathbf{e}\left(s_i\right), \mathbf{e}\left(s_{i+1}\right) \in \mathbb{R}^N$ associated with the tangent vectors estimated from the direct method. These error vectors are independent of the estimated step size $\Delta s^* = s_{i+1} - s_i$. Then the curvature approximation can be written as

$$\ddot{c}^*\left(s\right) = \frac{1}{\Delta s^*}\left(\dot{c}\left(s_{i+1}\right) + \mathbf{e}\left(s_{i+1}\right) - \dot{c}\left(s_i\right) - \mathbf{e}\left(s_i\right)\right). \tag{4.45}$$

As Δs^* goes to 0, $\frac{1}{\Delta s^*}\left(\dot{c}\left(s_{i+1}\right) - \dot{c}\left(s_i\right)\right)$ approaches $\ddot{c}\left(s\right)$ but, since $\mathbf{e}\left(s\right)$ does not decrease with Δs, $\frac{1}{\Delta s^*}\left|\mathbf{e}\left(s_{i+1}\right) - \mathbf{e}\left(s_i\right)\right|$ will grow as Δs^* decreases and will eventually become larger than the truncation error.

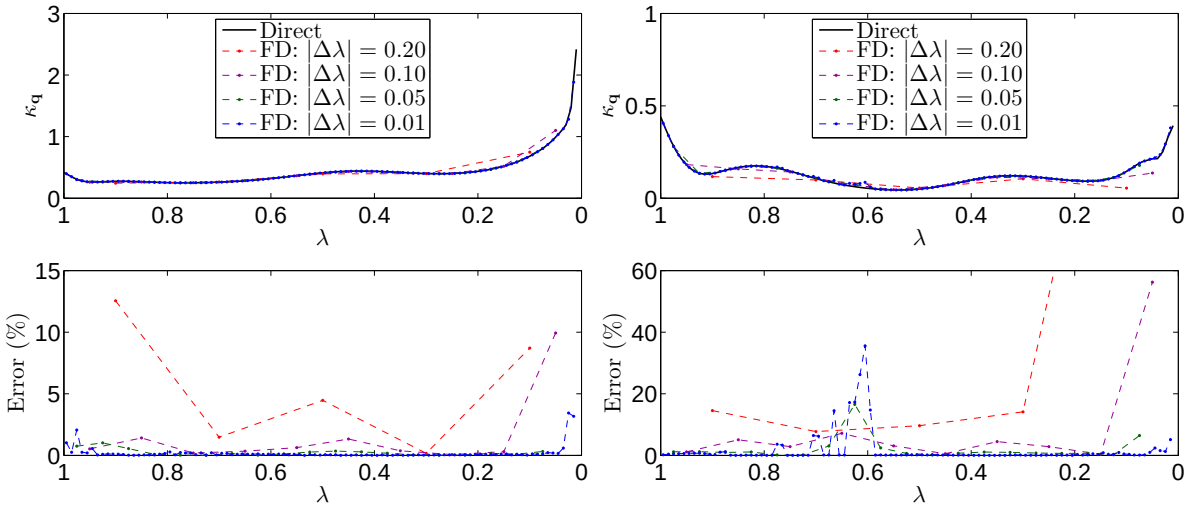
4.6.3 Curve Derivatives of Order n

The derivative of order n can be derived from the derivatives up to order $n-1$ in much the same way as the second derivative is derived from the first. The first step is to approximate the nth derivative of $\mathcal{H}\left(c\left(s\right)\right)$. This is given by Faá de Bruno's formula [135]:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{H}\left(c\left(s\right)\right) = \sum \frac{n!}{\prod_{j=1}^{n} j!^{m_j} m_j!} \nabla^{\sum_{j=1}^{n} m_j} \mathcal{H}\left(c\left(s\right)\right) \prod_{j=1}^{n} \left[\overset{(j)}{c}\left(s\right)\right]^{m_j}, \tag{4.46}$$

where the outer summation is taken over all n-tuples of non-negative integers $\{m_1, \ldots, m_n\}$ such that

$$\sum_{j=1}^{n} j m_j = n \tag{4.47}$$



(a) Subsonic: Mach 0.3, angle of attack 1°

(b) Transonic: Mach 0.8, angle of attack 3°

Figure 4.2: Comparison of $\kappa_{\mathbf{q}}$ calculated using the direct and finite-difference (FD) method with varying step size $|\Delta\lambda|$ on the inviscid NACA 0012 airfoil at subsonic and transonic conditions

and the notation which seems to indicate the product $\left[\overset{(j)}{c}(s)\right]^{m_j}$ for all j is intended to indicate that $\overset{(j)}{c}(s)$ appears with multiplicity m_j as input to $\nabla^{\sum_{j=1}^{n} m_j}\mathcal{H}(c(s))$.

Since $\mathcal{H}(c(s)) = \mathbf{0}$, therefore $\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{H}(c(s)) = \mathbf{0}$. Let

$$\mathbf{w}_n = \frac{\mathrm{d}}{\mathrm{d}s}\mathcal{H}(c(s)) - \nabla\mathcal{H}(c(s))\overset{(n)}{c}(s), \tag{4.48}$$

$\mathbf{w}_n \in \mathbb{R}^N$. Then

$$\nabla\mathcal{H}(c(s))\overset{(n)}{c}(s) = -\mathbf{w}_n. \tag{4.49}$$

Note that $\mathbf{w}_n$ is not a function of $\overset{(n)}{c}(s)$, so it is possible to approximate it using a generalized algorithm for the discrete directional derivative operators $\mathcal{D}$ on $\left(\dot{c}(s), \ldots, \overset{(n-1)}{c}(s)\right)$. Directional derivatives of any order can be calculated using Algorithm D.1 or D.3 for the general case. If all directions of the directional derivative are the same (that is, all input vectors in the tensor-vector product are the same) then Algorithm D.2 or D.4 can be used to reduce the number of residual evaluations needed.

Using equation (4.22), the under-determined system (4.49) can be compacted to the fully determined system

$$\nabla_{\mathbf{q}}\mathcal{H}(c(s))\,\mathbf{z}_n = -\mathbf{w}_n, \tag{4.50}$$

where

$$\mathbf{z}_n = \overset{(n)}{\mathbf{q}} + \overset{(n)}{\lambda}\mathbf{z}, \tag{4.51}$$

$\mathbf{z}_n \in \mathbb{R}^N$. The linear system (4.50) can be solved numerically for $\mathbf{z}_n$.

The additional equation needed to solve for $\overset{(n)}{\mathbf{q}}$ and $\overset{(n)}{\lambda}$ comes from differentiating the arclength

definition given by equation (4.1) $n-1$ times. This expression is obtained using the general Leibniz rule:

$$0 = \frac{\mathrm{d}^{n-1}}{\mathrm{d}s^{n-1}}\dot{c}\cdot\dot{c} = \sum_{k=0}^{n-1}\binom{n-1}{k}\overset{(k+1)}{c}\cdot\overset{(n-k)}{c}. \tag{4.52}$$

Solving for $\overset{(n)}{c}\cdot\dot{c}$ gives

$$\overset{(n)}{c}\cdot\dot{c} = \begin{cases} -\sum_{k=1}^{\frac{n-3}{2}}\binom{n-1}{k}\overset{(k+1)}{c}\cdot\overset{(n-k)}{c} - \frac{1}{2}\binom{n-1}{\frac{n-1}{2}}\overset{((n+1)/2)}{c}\cdot\overset{((n+1)/2)}{c} & n \text{ is odd} \\ -\sum_{k=1}^{\frac{n}{2}-1}\binom{n-1}{k}\overset{(k+1)}{c}\cdot\overset{(n-k)}{c} & n \text{ is even,} \end{cases} \tag{4.53}$$

which can be evaluated numerically.

Taking the dot product of both sides of equation (4.51) with $\dot{\mathbf{q}}$ gives

$$\mathbf{z}_n\dot{\mathbf{q}} = \overset{(n)}{\mathbf{q}}\cdot\dot{\mathbf{q}} + \overset{(n)}{\lambda}\mathbf{z}\cdot\dot{\mathbf{q}} = \overset{(n)}{c}\cdot\dot{c} - \overset{(n)}{\lambda}\dot{\lambda} + \overset{(n)}{\lambda}\mathbf{z}\cdot\dot{\mathbf{q}}. \tag{4.54}$$

This can be rearranged and simplified using equations (4.31) and (4.32) to give:

$$\overset{(n)}{\lambda} = \frac{\mathbf{z}_n\cdot\dot{\mathbf{q}} - \overset{(n)}{c}\cdot\dot{c}}{\sqrt{\mathbf{z}\cdot\mathbf{z}+1}}. \tag{4.55}$$

This is substituted into equation (4.51) to calculate $\overset{(n)}{\mathbf{q}}$:

$$\overset{(n)}{\mathbf{q}} = \mathbf{z}_n - \overset{(n)}{\lambda}\mathbf{z}. \tag{4.56}$$

The higher order derivative calculation is summarized as Algorithm 4.1. The calculation can alternatively be represented as an $N+1$ by $N+1$ system of equations:

$$\begin{pmatrix} \nabla_{\mathbf{q}}\mathcal{H}\left(c\left(s\right)\right) & \mathbf{z} \\ \dot{\mathbf{q}}\left(s\right) & \dot{\lambda}\left(s\right) \end{pmatrix}\begin{pmatrix} \overset{(n)}{\mathbf{q}}\left(s\right) \\ \overset{(n)}{\lambda}\left(s\right) \end{pmatrix} = \begin{pmatrix} -\mathbf{w}_n \\ x_n \end{pmatrix}, \tag{4.57}$$

where $x_n \in \mathbb{R}$ is calculated from the right-hand side of equation (4.53). Notice that the $N+1$st column and $N+1$st row are both dense. A procedure for solving a sparse linear system with a dense row and column appended is presented in Appendix C. This procedure involves two linear solves using the sparse sub-matrix $\nabla_{\mathbf{q}}\mathcal{H}\left(c\left(s\right)\right)$, whereas the procedure presented in this section requires only one linear solve because it has been possible to recycle the solution to the linear solve from the tangent calculation.

4.6.4 Validation of Higher Curve Derivative Calculations

The test case is again inviscid flow over the two-dimensional NACA 0012 airfoil on grid Ne at Mach 0.3 and angle of attack of 1° using the second-order dissipation operator as the homotopy system. The backwards-difference estimate of the nth derivative is obtained using the $n-1$st derivative:

$$\overset{(n)}{c}\left(s_{i+\frac{1}{2}}\right) \approx \overset{(n)}{c}{}^{*}\left(s_{i+\frac{1}{2}}\right) = \frac{1}{\Delta s_i^*}\left(\overset{(n-1)}{c}\left(s_{i+1}\right) - \overset{(n-1)}{c}\left(s_i\right)\right). \tag{4.58}$$

Algorithm 4.1: High-order curve derivative calculation with arclength parametrization for curve derivative of order n

Data: n, $\mathbf{q}$, λ, $\mathcal{H}(\mathbf{q},\lambda)$, $\nabla\mathcal{H}(\mathbf{q},\lambda)$
Result: $\dot{\mathbf{q}}(s)$, ..., $\overset{(n)}{\mathbf{q}}(s)$, $\dot{\lambda}(s)$, ..., $\overset{(n)}{\lambda}(s)$
Calculate $\dot{c}(s)$
for $d = 2 : n$ **do**
  Calculate ϵ for $\overset{(d-1)}{c}(s)$
  Calculate $\mathbf{w}_n$ from equations (4.48) and (4.46)
  Solve $\nabla_{\mathbf{q}}\mathcal{H}\mathbf{z}_n = -\mathbf{w}_n$
  Calculate $\overset{(d)}{\lambda}(s)$ from equation (4.55)
  Calculate $\overset{(d)}{\mathbf{q}}(s)$ from equation (4.56)
end

Defining

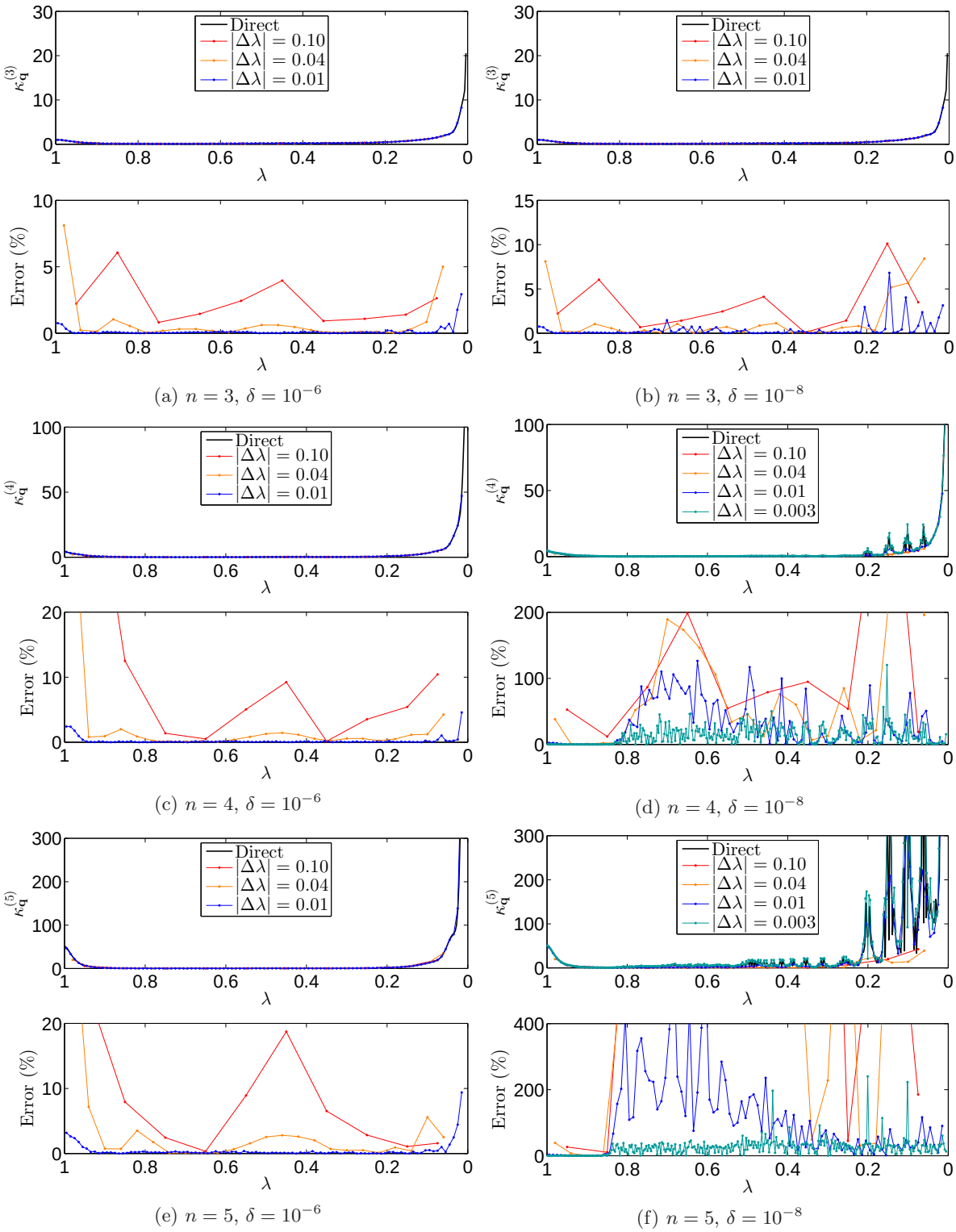
$$\kappa_{\mathbf{q}}^{(n)} \equiv \sqrt{\overset{(n)}{\mathbf{q}} \cdot \overset{(n)}{\mathbf{q}}}, \tag{4.59}$$

$\kappa_{\mathbf{q}}^{(n)} \in \mathbb{R}$, the validation is performed by determining whether the backwards-difference estimates of $\kappa_{\mathbf{q}}^{(n)}$ converge to the $\kappa_{\mathbf{q}}^{(n)}$ values obtained from the direct method in the limit of the finite-difference step size $|\Delta\lambda|$ going to 0.

It is apparent from Figures 4.3a), c), and e) that the higher order derivative calculations have been implemented correctly. However, it is also observed that the calculation is sensitive to the value of δ. Figure 4.3 shows the $\kappa_{\mathbf{q}}^{(n)}$ values for both $\delta = 10^{-6}$ and $\delta = 10^{-8}$, where δ refers to the δ value used in the $\mathbf{w}_n$ calculations only; $\delta = 10^{-12}$ is used for the matrix-vector products in all linear solves in both cases.

While it is clear that the accuracy of the calculation is sensitive to δ, it is also clear from Figures 4.3b), d), and f) that the $\overset{(n)}{c}(s)$ calculation is sensitive to the accuracy of $\overset{(n-1)}{c}(s)$ as the numerical errors can be seen to become greatly exaggerated when propagated to the next higher derivative. Since this is observed for both the direct and finite-difference estimates of $\overset{(n)}{c}(s)$, this may be a property of $\overset{(n)}{c}(s)$ and not the direct calculation method presented here.

It is generally expected when estimating directional derivatives with finite-difference approximations such as equation (3.23) that the calculation will be accurate for a certain range of δ. When δ is too large, truncation error will dominate and when δ is too small, rounding error will dominate. When plotting error versus δ, a "V" pattern is thus expected. One might infer then from Figure 4.3, for which the $\kappa_{\mathbf{q}}^{(n)}$ calculation has been performed using double precision arithmetic, that $\delta = 10^{-8}$ is too small and that rounding error is dominating the calculation for this value of δ. However, when increasing the arithmetic precision from double precision (64 bit) to quadruple precision (128 bit), this is revealed not to be the case. A comparison of the error in the curvature calculation using double and quadruple precision arithmetic for several values of δ is shown in Figure 4.4. It is apparent from the figure that the higher derivative calculations for δ values as small as 10^{-12} are in very close agreement for the same calculation performed in double and quadruple precision, indicating that rounding error for the double precision calculations is not very significant for these values of δ. While the calculation seems accurate for $\delta = 10^{-6}$, it becomes inaccurate for smaller δ in the range $10^{-8} \geq \delta \geq 10^{-12}$. As δ is decreased beyond 10^{-12}, the accuracy of the calculation improves for the quadruple precision calculation but rounding error begins to dominate for the double precision calculation. For the quadruple precision calculation,



(a) $n = 3$, $\delta = 10^{-6}$

(b) $n = 3$, $\delta = 10^{-8}$

(c) $n = 4$, $\delta = 10^{-6}$

(d) $n = 4$, $\delta = 10^{-8}$

(e) $n = 5$, $\delta = 10^{-6}$

(f) $n = 5$, $\delta = 10^{-8}$

Figure 4.3: Comparison of $\kappa_{\mathbf{q}}^{(n)}$ for $n = 3$, $n = 4$, and $n = 5$ calculated using the direct and finite-difference (FD) method with different step size $|\Delta\lambda|$ on the inviscid NACA 0012 airfoil at Mach 0.3 and angle of attack of 1°

the error eventually begins to increase again around 10^{-20}.

4.6.5 Curve Derivatives with λ Parametrization

The curve derivatives can also be calculated with a λ parametrization. Recall that λ parametrization is defined implicitly by the condition $\dot{\lambda}(r) = -1$ previously indexed as equation (4.2). Successively differentiating both sides of this equation yields the additional conditions:

$$\overset{(n)}{\lambda}(r) = 0 \tag{4.60}$$

for all $n > 1$. Differentiating $\mathcal{H}(c(r)) = \mathbf{0}$ and using condition (4.2) gives the expression for the first derivative:

$$\nabla_{\mathbf{q}}\mathcal{H}(c(r))\,\dot{\mathbf{q}}(r) = \frac{\partial}{\partial\lambda}\mathcal{H}(c(r)). \tag{4.61}$$

Note the useful property

$$\dot{\mathbf{q}}(r) = \sqrt{\mathbf{z}\cdot\mathbf{z}+1}\,\dot{\mathbf{q}}(s) \tag{4.62}$$

which allows for easy conversion between $\dot{\mathbf{q}}(s)$ and $\dot{\mathbf{q}}(r)$.

The derivation for the higher derivatives for this parametrization proceeds in much the same way as the derivation for the higher derivatives with respect to the arclength parametrization. Differentiating $\mathcal{H}(c(r)) = \mathbf{0}$ n times gives an expression for $\frac{\mathrm{d}}{\mathrm{d}r}\mathcal{H}(c(r))$ analogous to equation (4.46). Define

$$\mathbf{w}'_n = \frac{\mathrm{d}}{\mathrm{d}r}\mathcal{H}(c(r)) - \nabla\mathcal{H}(c(r))\overset{(n)}{\mathbf{q}}(r), \tag{4.63}$$

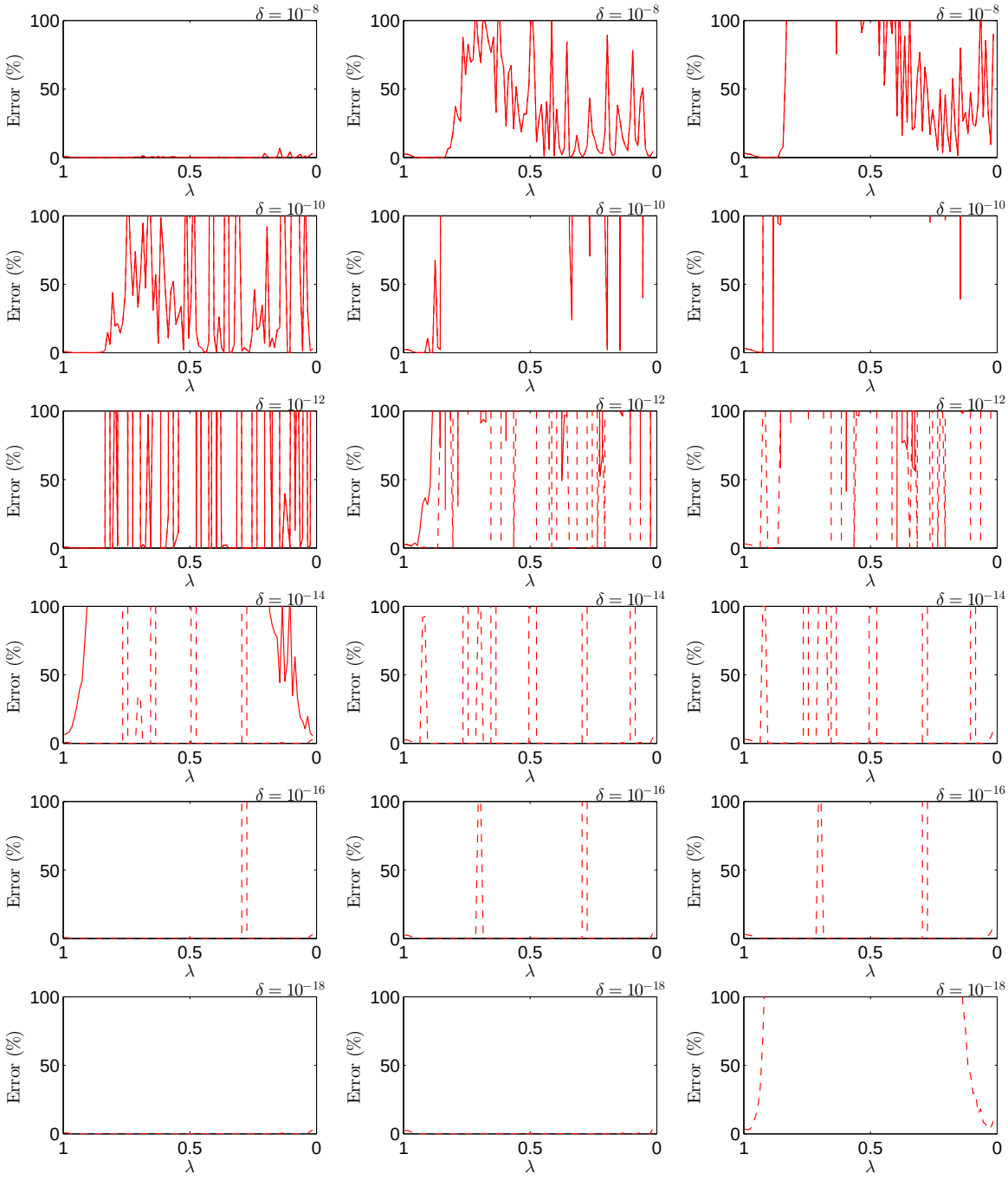
$\mathbf{w}'_n \in \mathbb{R}^N$, where the prime distinguishes $\mathbf{w}'_n$ from $\mathbf{w}_n$. Since $\frac{\mathrm{d}}{\mathrm{d}r}\mathcal{H}(c(r)) = \mathbf{0}$, the expression for the nth derivative of $\mathbf{q}(r)$ is given by

$$\nabla\mathcal{H}(c(r))\overset{(n)}{\mathbf{q}}(r) = -\mathbf{w}'_n. \tag{4.64}$$

The $\overset{(n)}{\mathbf{q}}(r)$ calculation is summarized as Algorithm 4.2, where the vector $\mathbf{w}'_n$ can be evaluated by applying Algorithms D.1 through D.4 without modification.

Algorithm 4.2: High order curve derivative calculation with λ parametrization for curve derivative of order n

Data: n, $\mathbf{q}$, λ, $\mathcal{H}(\mathbf{q},\lambda)$, $\nabla\mathcal{H}(\mathbf{q},\lambda)$
Result: $\dot{\mathbf{q}}(r)$, …, $\overset{(n)}{\mathbf{q}}(r)$, $\dot{\lambda}(r)$, …, $\overset{(n)}{\lambda}(r)$
Calculate $\dot{c}(r)$
for $d = 2 : n$ **do**
  Calculate ϵ for $\overset{(d-1)}{c}(r)$
  Calculate $\mathbf{w}'_n$ from equations (4.63) and (4.46)
  Solve $\nabla_{\mathbf{q}}\mathcal{H}\overset{(n)}{\mathbf{q}}(r) = -\mathbf{w}'_n$
end



(a) $n = 3$ (b) $n = 4$ (c) $n = 5$

Figure 4.4: Comparison of the error in the direct and finite-difference estimates of $\kappa_{\mathbf{q}}^{(n)}$ calculated using double precision (solid line) and quadruple precision (dashed line) at $|\Delta\lambda| = 0.01$ for an inviscid NACA 0012 airfoil at Mach 0.3 and angle of attack of 1°; if only the solid line is visible it is because the two lines overlap; otherwise, if a line is not visible it is because the error is in excess of 100%

4.6.6 Practical Considerations for Calculating High Derivatives of Curves

The primary cost in terms of CPU time is in forming the $\mathbf{w}_n$ vector and solving the linear system given by equation (4.49). The linear system is solved using the usual preconditioned FGMRES algorithm and is straightforward to implement. A consideration which can reduce CPU time is that the matrix on the left-hand side of the equation is the same for any n, so the preconditioner only needs to be formed once. The focus of this section will be on the formation of $\mathbf{w}_n$ since this calculation can be done in a variety of different ways with very significant impact on accuracy, data storage, and CPU cost.

The $\mathbf{w}_n$ calculation potentially requires numerous directional derivatives and their coefficients to be computed. The directional derivatives are identified and the coefficients are calculated using equation (4.46) with condition (4.47) and ignoring the $\nabla\mathcal{H}\left(c\left(s\right)\right)\overset{(n)}{c}\left(s\right)$ term. While this is relatively straightforward, the complexity of the summation in equation (4.46) provides some challenges in terms of data allocation and establishing a logical indexing. The indexing system that we have developed comes from noticing that the sum of the orders of the derivatives is always equal to n and recalling that the input vectors to the directional derivatives commute. This means that the total number of directional derivatives needed to construct $\mathbf{w}_n$ is equal to the number of different integer combinations which can be summed to make n, the order of the summands being irrelevant. This value, denoted $\mathcal{P}\left(n\right)$, is called the *partition function* and any non-ordered integer set whose sum equals n is called a *partition* [4]. The partition function can be evaluated using the recursive equation:

$$\mathcal{P}\left(n\right)=\sum_{k>0}\left(-1\right)^{k-1}\left[\mathcal{P}\left(n-g_k^-\right)+\mathcal{P}\left(n-g_k^+\right)\right], \tag{4.65}$$

$$g_k^-=\frac{k\left(3k-1\right)}{2},\ g_k^+=\frac{k\left(3k+1\right)}{2},$$

$$\mathcal{P}(1)=0,\ \mathcal{P}(0)=0,\ \mathcal{P}(k)=0\ \text{ for }\ k<0,$$

$$\mathcal{P}:\mathbb{Z}\to\mathbb{Z},\ g_k^-\in\mathbb{Z},\ g_k^+\in\mathbb{Z},$$

where the summation is terminated when the condition $n>g_k^-$ is met.

The partitions are arranged anti-lexicographically using algorithm ZS1 of Zoghbi and Stojmenović [177], the parts of the partition representing the multiplicity of each derivative. As an example, the indexing matrix generated from the partitioning algorithm is shown paired with the coefficient vector generated from equation (4.46) for $n=4$:

$$\begin{pmatrix}4&0&0&0\\3&1&0&0\\2&2&0&0\\2&1&1&0\\1&1&1&1\end{pmatrix},\quad\begin{pmatrix}1\\4\\3\\6\\1\end{pmatrix}.$$

Since the order of the tensor for the jth term is equal to the number of non-zero entries in the jth row, this matrix and vector combination contains enough information to construct $\mathbf{w}_4$. Ignoring the top line, which corresponds to $\mathcal{DH}\left(c\left(s\right)\right)\overset{(n)}{c}\left(s\right)$, the expression for $\mathbf{w}_4$ is

$$\mathbf{w}_4=4\mathcal{D}^2\mathcal{H}\left[\dddot{c},\dot{c}\right]+3\mathcal{D}^2\mathcal{H}\left[\ddot{c},\ddot{c}\right]\ +6\mathcal{D}^3\mathcal{H}\left[\ddot{c},\dot{c},\dot{c}\right]+\mathcal{D}^4\mathcal{H}\left[\dot{c},\dot{c},\dot{c},\dot{c}\right].$$

n	5	10	15
$(n+1)\mathcal{P}(n)$	42	462	2816
n^n	3125	10^{10}	4.38×10^{17}

Table 4.1: Number of integer values stored for the partition-based storage scheme compared to a simple index-based storage scheme for calculating $\mathbf{w}_n$

The number of integers needed to be stored using the partition-based method is only $(n+1)\mathcal{P}(n)$, whereas a more naive scheme which allocates for every combination of $j \leq n$ would allocate for n^n. Table 4.1 shows that partition-based memory allocation and indexing becomes necessary as n becomes large.

While the number of residual evaluations required to evaluate equation (4.48) grows at a rapid rate, the cost is still quite low for practical values of n. Mackens [104] investigated a method for reducing this cost and also the complexity of the calculation. The method comes from the observation that

$$\left(\frac{\mathrm{d}}{\mathrm{d}s}\right)^n \mathcal{H}\left(c(s) + \sum_{j=1}^{n-1} \frac{\Delta s^j}{j!} c^{(j)}(s)\right)_{|\Delta s=0} = -\mathbf{w}_n. \tag{4.66}$$

The cost benefit of using equation (4.66) in place of equation (4.48) is that only a single directional derivative with a single direction must be computed for each derivative, resulting in much fewer residual evaluations.

The disadvantage of Mackens' method is that a single step size is applied and takes exponent k when it is the coefficient of the kth derivative vector. We have concerns that this might cause severe problems with accuracy for our applications. Mackens [104] showed the error for calculations of derivatives up to $n = 4$ using this method for a very low-dimensional homotopy in $\mathbb{R}^2$ and found that the rounding error grew significantly as n was increased. Syam and Siyyam [162] performed a similar study and the same trend is seen in their second test case, their first case being a simple scalar case. Since the direction vectors in the directional derivatives can vary by 8 orders of magnitude or more for our applications it seems highly unlikely that the directional derivatives can be constructed with sufficient accuracy using this method. The significant increase in $\kappa_{\mathbf{q}}^{(n)}$ near $\lambda = 0$ for larger n may be due to the increased nonlinearity of $\mathcal{R}(\mathbf{q})$ relative to $\mathcal{G}(\mathbf{q})$.

The next topic which is addressed is whether second-order or first-order first derivative approximations should be used to construct the tensor-vector products. From experimentation with both we have not seen any accuracy improvement for the second-order approximation when calculating curve derivatives of up to the third derivative but some slight improvement seems apparent at higher derivatives. The cost measured in number of residual evaluations of forming the tensor-vector products for $\nabla^n \mathcal{H}$ is 2^n when using the second-order accurate method and $2^n - 1$ when using the first-order accurate method. Since the additional cost of using the second-order approximation is between modest and negligible, especially when considering the cost of solving the linear system, we opt to use the second-order accurate tensor-vector product estimates.

Another consideration which is important for efficiency is special consideration for the case where all direction vectors to the directional derivative are in the same direction. Though the generalized Algorithm D.3 could be used for all directional derivatives, the number of residual evaluations needed to evaluate the tensor-vector product in this way is 2^n, whereas Algorithm D.4, which takes advantage

Method	n							
	2	3	4	5	6	7	8	9
1a	3	10	28	66	154	334	723	1515
1b	2	6	16	40	92	214	473	1009
2a	4	12	32	72	164	348	744	1544
2b	2	8	18	46	100	228	492	1038
Mackens	2	3	4	5	6	7	8	9

Table 4.2: Cost of evaluating $\mathbf{w}_n$ using various methods, where cost is measured in number of residual evaluations; the methods are:
1a) First-order accurate, general tensor-vector products only (Algorithm D.1);
1b) First-order accurate, making use of single-direction directional derivatives (Algorithms D.1 and D.2);
2a) Second-order accurate, general tensor-vector products only (Algorithm D.3);
2b) Second-order accurate, making use of single-direction directional derivatives (Algorithms D.3 and D.4)

of all directions being the same, requires only $n + \mathrm{mod}\,(n, 2)$ residual evaluations. This is particularly significant when calculating the high curve derivatives because the highest order tensor-vector product appearing in the $\mathbf{w}_n$ calculation is always of this form.

The cost measured in number of residual evaluations needed to calculate $\mathbf{w}_n$ is shown in Table 4.2 for the various methods. The data in this table reinforce the recommendations made in this section. If high derivatives are desired then clearly there is cost benefit to Mackens' method if the accuracy issues can be overcome. There is also room for significant cost reduction if the tensor-vector product is fully generalized to account for any common direction vectors. For example, $\nabla^3\mathcal{H}\,(c)\,[\ddot{c}, \dot{c}, \dot{c}]$ could be made more efficient by considering that two of the direction vectors are the same, reducing the cost from 8 residual evaluations to 6. While the cost reduction is modest for this example, it becomes very significant for higher order tensors and occurs often in the $\mathbf{w}_n$ calculation.

4.7 μ-Scaling

The μ-scaling was developed as a means to distribute the curvature effects more evenly across the homotopy curve. This scaling is equivalent to a global re-parametrization with respect to λ and does not affect the curve other than through a coordinate transformation. However, making the curvature more consistent throughout traversing can improve the performance of the homotopy continuation algorithms.

Consider the following modified version of the convex homotopy given by equation (3.30):

$$\mathcal{H}\,(\mathbf{q}, \lambda_k) = (1 - \lambda_k)\,\mathcal{R}\,(\mathbf{q}) + \lambda_k \mu \mathcal{G}\,(\mathbf{q}) = \mathbf{0}, \tag{4.67}$$

$$k \in [0, m]\,,\ \lambda_k \in \mathbb{R},\ \lambda_0 = 1,\ \lambda_m = 0,\ \lambda_{k+1} < \lambda_k,\ \mu \in \mathbb{R},$$

$$\mathcal{H} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N,\ \mathcal{G} : \mathbb{R}^N \to \mathbb{R}^N,\ \mathcal{R} : \mathbb{R}^N \to \mathbb{R}^N.$$

The effects of μ can be understood by recognizing that the deformation (4.67) is equivalent to (3.30) under the following change of coordinates:

$$\lambda \leftarrow \frac{\lambda\mu}{1 - \lambda + \mu\lambda}. \tag{4.68}$$

When λ is close to 0, this change of coordinates results approximately in $\lambda \leftarrow \mu\lambda$. So, if $\mu > 1$, the equivalent step in the λ-direction will be larger near $\lambda = 0$. In effect, $\mu > 1$ compresses the domain near $\lambda = 0$ and stretches it near $\lambda = 1$, and $\mu < 1$ has the reverse effect.

In practice, it is convenient to apply μ-scaling as two components:

$$\mu = \mu_{\mathrm{a}}\mu_{\mathrm{u}},\ \ \mu_{\mathrm{a}}, \mu_{\mathrm{u}} \in \mathbb{R}, \tag{4.69}$$

where μ_{a} is a benchmark value which may be determined from experience or using a numerical algorithm. A numerical algorithm for determining μ_{a} based on the performance of the predictor-corrector algorithm is presented in Section 5.6. The parameter μ_{u} is user-supplied to allow for some user control over the μ-scaling. However, the μ-scaling is only intended for some high level calibration and μ_{u} is rarely set to a value other than unity. If a homotopy continuation algorithm fails, we usually use the $\mathrm{C_l}$ and $\mathrm{C_d}$ values calculated along the curve trajectory to assess whether the curvature is poorly distributed. If the continuation algorithm fails early on and is accompanied by large changes in the functional values, then choosing $\mu_{\mathrm{u}} > 1$ may improve performance. If the continuation algorithm fails closer to $\lambda = 0$ and the functional values are relatively unchanging throughout the early traversing process, then $\mu_{\mathrm{u}} < 1$ may help convergence. Generally μ_{u} must be changed by at least a factor of 5 in order to see any significant change to algorithm performance.

A more quantitative way to re-distribute curvature is to use step-length adaptation. Step-length adaptation is equivalent to local re-parameterization and is based on information collected during traversing. Step-length adaptation is specific to the continuation algorithm and will be discussed for each homotopy continuation algorithm presented.

The values of μ_{a} determined by the algorithm (to be presented in Section 5.6 for the convex homotopy with the dissipation operator are approximately 0.7 for inviscid flows, 0.5 for laminar flow, and 0.1 for RANS cases, each obtained from a subsonic case on the ONERA M6 using grids Me1, MlHH, and MtHH. The global homotopy can also include μ-scaling, in which case it takes on the following form:

$$\left[1 - \lambda_k \left(1 - \mu\right)\right] \mathcal{R}\left(\mathbf{q}\right) - \lambda_k \mu \mathcal{R}\left(\mathbf{q}_0\right). \tag{4.70}$$

4.8 Surrogate Curves

Since the curves representing the homotopies exist in higher dimensional real space and cannot be visualized, one-dimensional surrogate curves can be used to assess the performance of the homotopy continuation algorithms. For example, the values of the lift and drag coefficients calculated along the curve can be used as one-dimensional surrogates for the curve. These surrogates are not expected to give a realistic impression of the features of the curve such as curvature but can be used to roughly evaluate the effectiveness of various curve tracing or curve prediction tools. Since the lift and drag coefficients take on simple scalar values it is possible for one to be close to the correct value even if the error is high. This is less likely to occur with two surrogate curves, so when the surrogate curve approach is used to evaluate performance, at least two surrogate curves are normally considered.

4.9 Some Numerical Studies of Homotopies

The purpose of the studies presented in this section is to show examples of surrogate curves, demonstrate the use of curvature profiling, characterize the homotopies presented in this chapter for some specific test cases, and demonstrate the effects of μ-scaling on the curvature profiles.

4.9.1 Surrogate Curves for some 1D Inviscid Homotopies

A preliminary study was performed to investigate convex and global homotopies for subsonic and transonic cases. The reason a one-dimensional case was studied is because even very difficult homotopy curves can be traced due to the low cost and the use of a direct solver using an LU decomposition in Matlab avoids possible failure of the Krylov solver.

The case studied is compressible air flowing through a converging-diverging nozzle. This physics problem is studied and solved analytically in several textbooks, including that of Pulliam and Zingg [142]. The shape of the nozzle $S(x)$ is given by

$$S(x) = \begin{cases} 1 + 1.5\left(1 - \frac{x}{5}\right)^2 & 0 \leq x \leq 5 \\ 1 + 0.5\left(1 - \frac{x}{5}\right)^2 & 5 \leq x \leq 10. \end{cases} \tag{4.71}$$

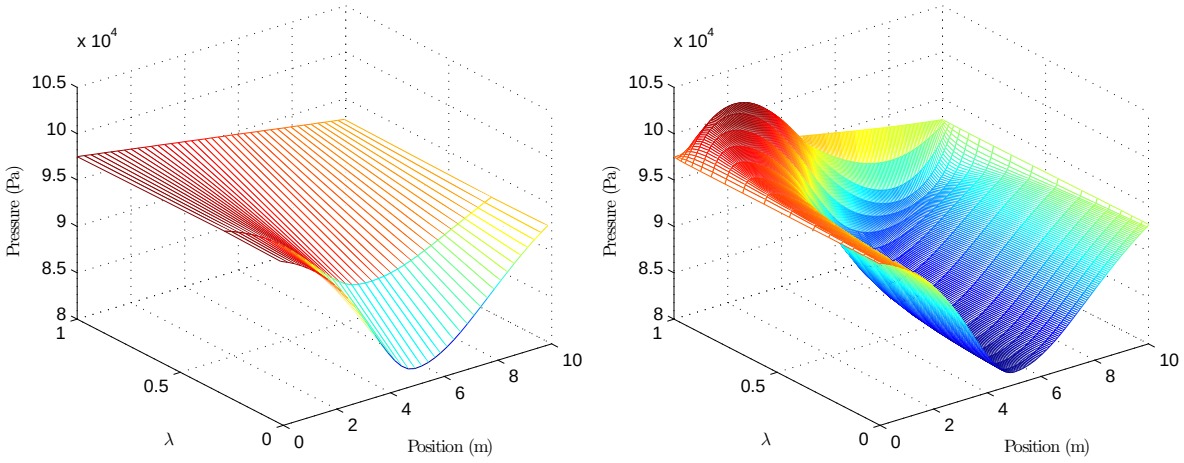
For the discretization we have used the SBP-SAT approach with 200 equally spaced nodes and an interface at the 120th node, where coupling across the interface is also enforced using the SAT approach. The implementation is in Matlab using an LU decomposition to solve the linear system. Flow cases for both the subsonic and transonic conditions of Pulliam and Zingg [142] are considered.

The homotopies considered are the convex homotopy with the dissipation operator with appropriate inlet/outlet boundary conditions and the global homotopy. The surrogate for the deformation in this case is the pressure profile over the spatial domain $x \in [0, 10]$, $x \in \mathbb{R}$. It can be seen from Figure 4.5 that the global homotopy is much more traceable than the convex homotopy with the dissipation operator for the subsonic case. However, the convex homotopy curve is far more traceable in the transonic case. For the global homotopy, the predictor-corrector method was found to diverge from the curve unless a very small step size (approximately $|\Delta\lambda| = 0.001$) was taken, even for this relatively simple problem.

4.9.2 Curvature Profiles for a 2D Inviscid Subsonic Flow

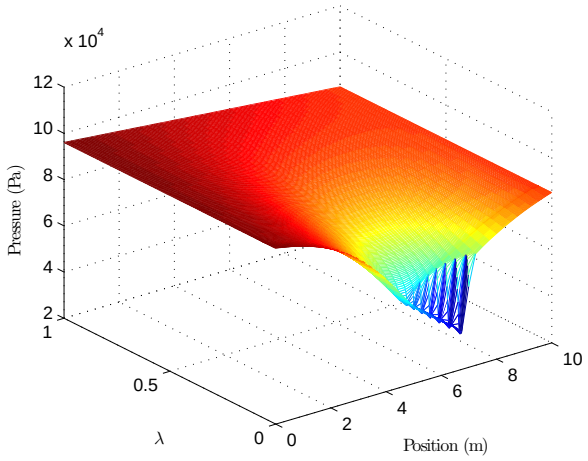
The test case for this study is an inviscid subsonic flow over the NACA 0012 airfoil on grid Ne, which is an 18-block, 15390 node grid. The operating conditions are Mach 0.3 and an angle of attack of 1°. Surrogate curves and curvature profiles are generated for the convex homotopy with the dissipation operator using each of far-field ("Diss - ff") and flow-imitative ("Diss - flow") boundary conditions, the convex homotopy with the diagonal operator ("Diag"), and also for global homotopy. The profiles are generated by accurately solving for points along the curve using a step size of $|\Delta\lambda| = 0.001$. The curvatures and C_l and C_d values are calculated and recorded at each point.

The profiles are shown in Figure 4.6. Clearly the lift and drag profiles give an inaccurate impression of the curve traceability. The traceability of the different homotopies can be compared by considering the curvature profiles with respect to different parametrizations. Under both arclength and λ parametrizations, the curve generated by the convex homotopy with the diagonal operator appears to exhibit the lowest traceability, and the curve generated by the global homotopy appears to exhibit the highest trace-

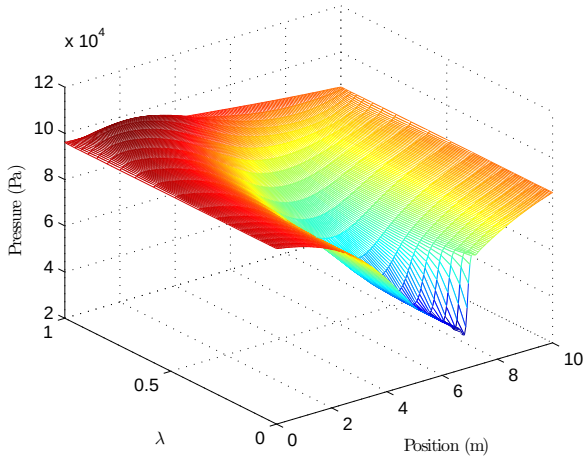


(a) Global, subsonic

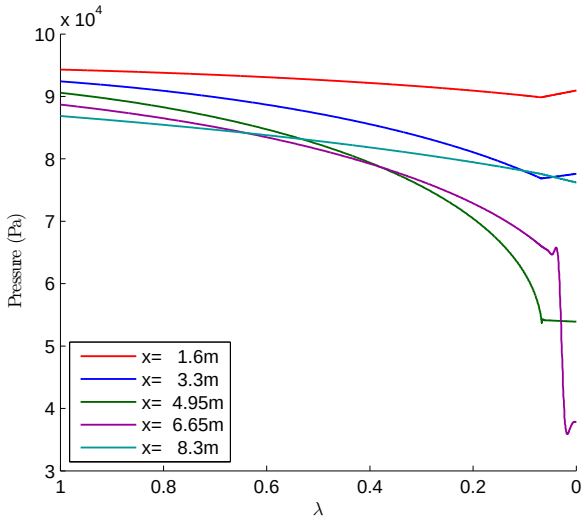
(b) Convex, subsonic



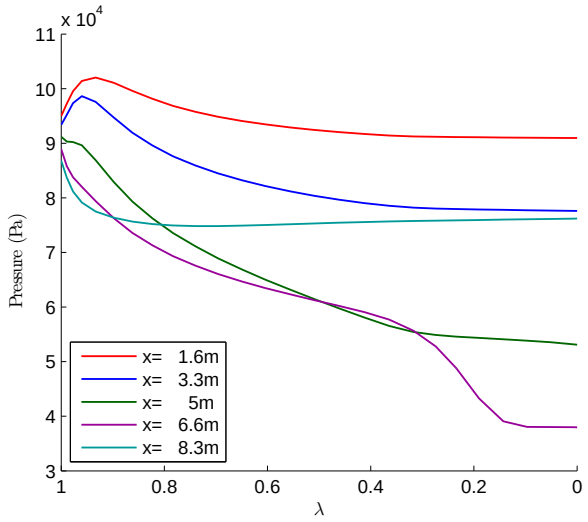
(c) Global, transonic



(d) Convex, transonic



(e) Global, transonic



(f) Convex, transonic

Figure 4.5: Surrogate curves for the global and convex homotopies with the dissipation homotopy system for a one-dimensional converging-diverging nozzle problem under subsonic and transonic conditions

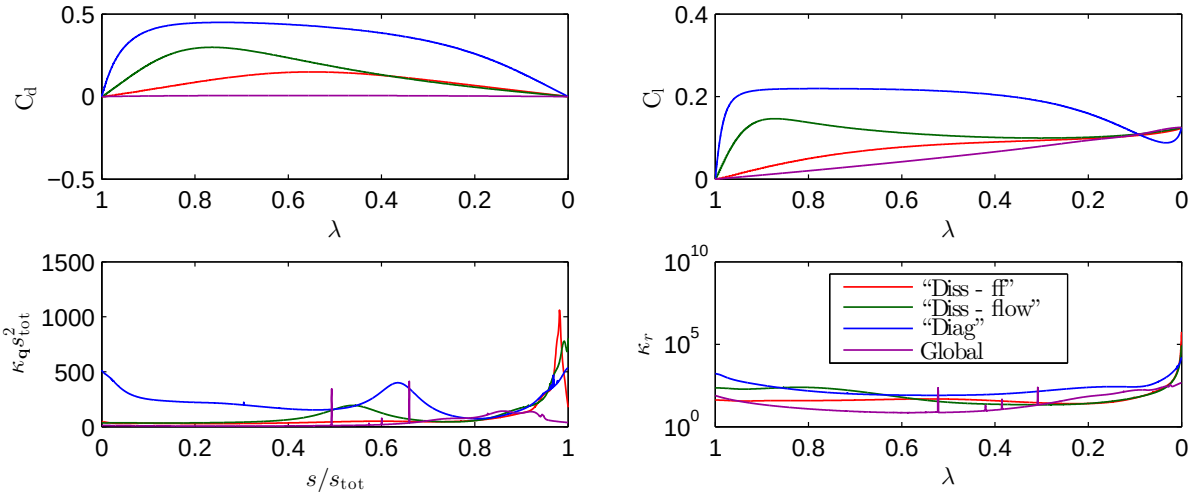


Figure 4.6: Surrogate curves and curvature profiles for some homotopies for inviscid flow on the NACA 0012 airfoil at Mach 0.3 and angle of attack of 1°; the homotopy systems are the convex homotopy with the dissipation operator using far-field boundary conditions ("Diss - ff"), dissipation operator using flow-imitative boundary conditions ("Diss - flow"), the diagonal operator ("Diag"), and global homotopy

ability. Neither boundary condition type for the dissipation operator definitively stands out as giving a curve with better traceability over the other. A notable feature of the curvature profiles from the deformations using the dissipation operators are the large curvature spikes near $s/s_{\text{tot}} = 1$. These spikes indicate that the curve becomes increasingly difficult to trace near $\lambda = 0$. The spikes appearing in the global homotopy curvature profile are indicative of inaccuracy in the $\mathbf{w}_n$ or tangent calculation.

4.9.3 Curvature Profiles for a 2D Transonic Inviscid Flow

This study is analogous to the previous one except that it is performed at the transonic conditions of Mach 0.8 and angle of attack 3°. The surrogate curves and curvature profiles are shown in Figure 4.7. The profiles could not be generated for the global homotopy because the curve-tracing algorithm failed for this case, even when using a very small step size. As with the subsonic example, the traceability of the convex homotopy with the diagonal operator is lower than the traceability using the dissipation operator.

4.9.4 Accuracy Study of the Curvature Calculation

The inaccuracy noticed in the curvature calculations of the previous study is investigated. The error was especially pronounced for the convex homotopy under the diagonal operator. This study varies from the previous accuracy study because the inaccuracy is observed for $\kappa_{\mathbf{q}}$ so the error cannot have propagated from previous $\mathbf{w}_k$ calculations because no $\mathbf{w}_k$ vectors have been formed yet other than $\mathbf{w}_2$. The sources investigated for this error are the tangent, the linear system, and the tensor-vector products.

The tangent and the linear system solved for the curvature calculation are subject to the same sources of error: rounding error in the linear solver, error resulting from inexactly solving the linear system, and

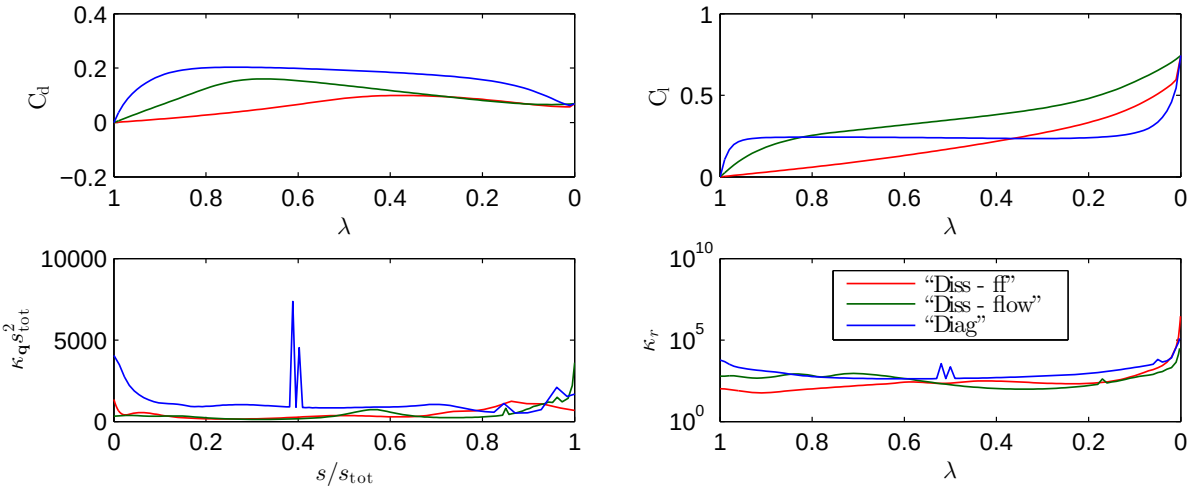


Figure 4.7: Surrogate curves and curvature profiles for some homotopies for inviscid flow on the NACA 0012 airfoil at Mach 0.8 and angle of attack of 3°; the homotopy systems are the convex homotopy with the dissipation operator using far-field boundary conditions ("Diss - ff"), dissipation operator using flow-imitative boundary conditions ("Diss - flow"), and the diagonal operator ("Diag")

inaccuracy in the Jacobian-vector product approximations. If the error is caused by inaccuracy in the directional derivative estimations, then it should be sensitive to the value of δ used in calculating the finite-differencing step size ϵ.

Figure 4.8 shows a comparison of the curvature profiles for the convex homotopy under the diagonal homotopy system for the transonic NACA 0012 case using different values of δ in the tensor-vector product. The linear systems appearing in the tangent and curvature calculation are solved to a relative tolerance of 10^{-8} using GCROT(m, k). The sweep is performed twice, once using finite-difference Jacobian-vector products with $\delta = 10^{-12}$ in the linear systems and once using the complex step method. The complex step data is barely visible in the figure because it very nearly overlaps with the finite difference data, indicating that any inaccuracy incurred by using the finite-differencing method to approximate the Jacobian-vector products in the tangent calculation has minimal impact on the inaccuracy observed in the $\kappa_{\mathbf{q}}$ calculation. The effect of changing the value of δ has a much more significant impact however. It is clear that the tensor-vector products are sensitive to this value. Except for two points along the curve for the $\delta = 10^{-12}$ case where the error was high, the extreme values $\delta = 10^{-12}$ and $\delta = 10^{-6}$ appear to generally result in less error in $\kappa_{\mathbf{q}}$ than the intermediate values, which is consistent with the observations of Section 4.6.4.

4.9.5 A Demonstration of the Effects of μ-Scaling

As an example illustrating the effect of μ-scaling, the C_l and C_d surrogate curves, as well as the curvature profile, are shown in Figure 4.9 for the inviscid NACA 0012 test case at Mach number 0.3 and angle of attack 1°. It can be seen from the C_l and C_d surrogate curves that using $\mu_u > 1$ has the effect of stretching the curve near $\lambda = 1$ and compressing it near $\lambda = 0$, while choosing $\mu_u < 1$ has the reverse effect. This is also reflected by the s vs. λ plot, where it can be seen that more of the arclength s is

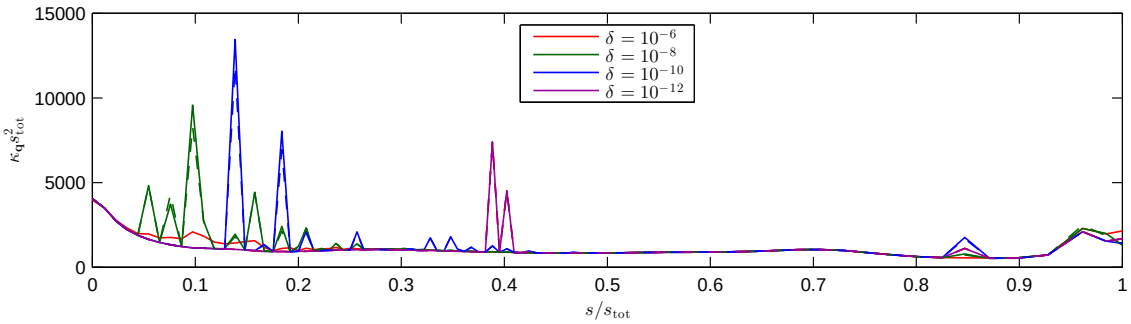


Figure 4.8: Effect of δ on the accuracy of the curvature calculation for the convex homotopy under the diagonal operator for the inviscid NACA 0012 test case at Mach 0.8 and angle of attack of 3°; the Jacobian-vector products appearing in the linear system in the tangent and curvature calculations are estimated using finite-differencing (solid lines) or the complex step method (dashed lines)

traversed early on when using smaller μ_{u}, or by the $\kappa_{\mathbf{q}}$ vs. λ plot, which shows how the partial curvature $\kappa_{\mathbf{q}}$ has been redistributed. The following plot, which shows $\kappa_{\mathbf{q}} s_{\text{tot}}^2$ vs. s/s_{tot}, shows much less dramatic redistribution of the curvature, indicating that the value of μ has much less impact on the performance of a continuation method which takes constant Δs than a continuation method which takes constant $\Delta\lambda$. The plot of κ_r vs. λ provides a metric for how difficult it would be to trace each of the three curves if constant $\Delta\lambda$ is used. Ideally, μ is chosen to make the κ_r profile is as flat as possible. Since $\mu_{\text{u}} = 1$ appears to provide a fairly consistent κ_r profile throughout traversing, the final subplot shows that the value of $\mu_{\text{a}} = 0.7$ produced by the minimization algorithm (5.17) is effective for this case.

4.9.6 Curvature Profiles for a 3D Inviscid Flow and Effect of Mesh Refinement

The purpose of this study is two-fold: to profile the homotopies for three-dimensional inviscid flows and to investigate the effects of grid refinement on the homotopy. For global homotopy, the homotopy residual has a continuous counterpart in physical space which is obtained by taking the limit of the mesh spacing vanishing. As such, the homotopy is expected to converge to a grid-converged value in the limit of the mesh spacing vanishing in much the same way as the flow solution does. Furthermore, since the flow residual is second-order accurate, the global homotopy should also be second-order accurate in the same sense as the flow residual; that is, the error in the homotopy on a current mesh calculated relative to its continuous counterpart is expected to be proportional to the grid spacing squared. Since both the dissipation operator and diagonal operator presented in this chapter have continuous counterparts which depend explicitly on the grid metrics, these homotopies can potentially have a high dependence on the mesh spacing, even if the mesh is already fine enough to give a very accurate flow solution.

The surrogate curves and curvature profiles are generated for inviscid flow over the ONERA M6 wing at Mach 0.4 and angle of attack 3°. The grid consists of 1.9208×10^6 nodes divided evenly into 32 blocks and is indexed as grid Me1. The fine version of the mesh is generated by doubling the number of nodes in each direction, the location of the new nodes determined by interpolation from a B-spline parametrization of the grid [65, 175], which is important in order to fit the ONERA M6 wing smoothly on the fine mesh. Each block is then split evenly into 8 blocks, resulting in 256 blocks in total. The grid is indexed as grid Me2.

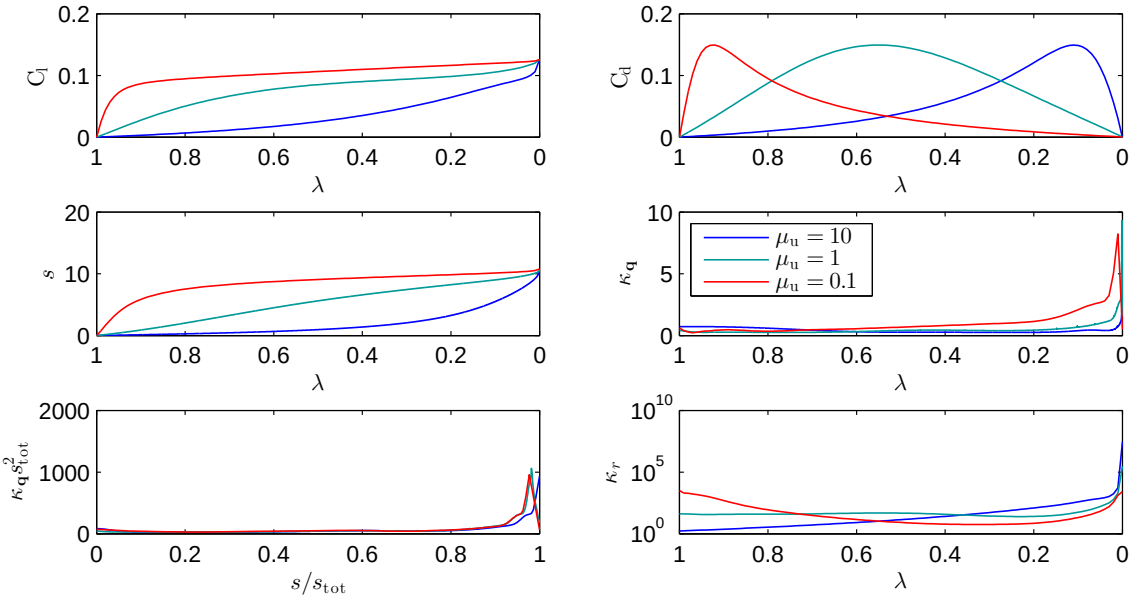


Figure 4.9: Effect of μ_u on the homotopy using the dissipation operator for the inviscid NACA 0012 test case at Mach 0.3 and angle of attack 1°

Since there are 8 times more points on the fine mesh than the original, the $\kappa_{\mathbf{q}} s_{\text{tot}}^2$ and κ_r values will naturally increase by a factor of $\sqrt{8}$. The reason for this can be seen clearly from the equations. For example, if instead the nodes had increased by a factor of two (consider, for example, the 1D case), then the additional elements appended to the $\mathbf{q}$ vector are very close in size to the original values, so $\kappa_{r,\text{fine}} \sim \sqrt{\dot{\mathbf{q}}(r) \cdot \dot{\mathbf{q}}(r) + \dot{\mathbf{q}}(r) \cdot \dot{\mathbf{q}}(r)} = \sqrt{2}\kappa_r$. This factor is simply due to having more state elements in the approximation to the continuous-space counterpart of the homotopy and does not indicate decreased traceability on the finer mesh. To account for this effect, $\kappa_{\mathbf{q}} s_{\text{tot}}^2/\sqrt{N}$ and $\kappa_r/\sqrt{N}$ are used as traceability metrics instead of the usual $\kappa_{\mathbf{q}} s_{\text{tot}}^2$ and κ_r. Note that this modification to the traceability metrics is only applicable to uniform mesh refinement and is not suitable for comparing traceability on different meshes in general.

Figure 4.10 shows the comparison of the surrogate curves and curvature profiles on the original mesh and finer mesh. It appears from the data that the profiles for all three convex homotopies have been affected noticeably but not dramatically by the mesh refinement. The surrogate curves for the global homotopy match very closely on both the original mesh and finer mesh but there is some significant difference in the curvature profile. By comparing the $\kappa_{\mathbf{q}}$ and κ_r values to their finite-difference estimates, it appears that the discrepancy may simply be inaccuracy in the curvature calculation on the fine mesh. This also emphasizes the need to improve the accuracy of the tensor-vector product estimations.

4.9.7 Curvature Profiles for a 3D Laminar Flow and Effect of Grid Topology

This test case is laminar flow over the ONERA M6 wing at Mach 0.3, angle of attack 1°, and Reynolds number 1×10^3. Two grids are used, indexed as grids MlHH and MlHC. Grid MlHH consists of 2.11×10^6 nodes divided evenly into 48 blocks; grid MlHH consists of 1.88×10^6 nodes divided evenly into 16 blocks. Though the grids are of similar size and refinement, the first one has an H-H topology whereas the second one has an H-C topology. The global homotopy could not be converged for this case.

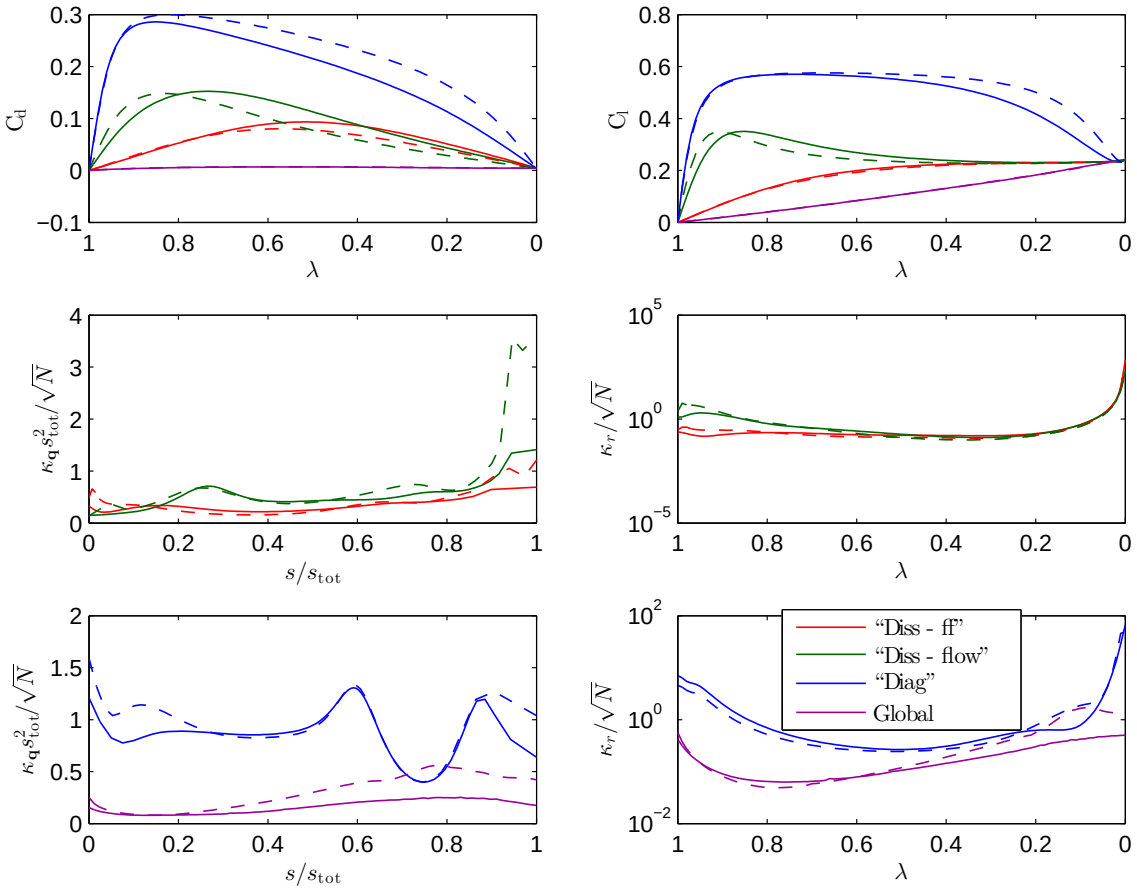


Figure 4.10: Curvature profiles and the effects of mesh refinement on several homotopies for the inviscid ONERA M6 test case at Mach 0.4 and angle of attack of 3°; the dashed lines represent the data for the finer grid

The homotopies on the two grids are compared in Figure 4.11. The homotopies appear to be quite similar on the two grids based on the functional evolution as well as the curvature profiles. Hence, algorithm performance is expected to be similar for both grid topologies.

4.9.8 Curvature Profiles for a 2D Turbulent Flow

This study is a curvature profiling for turbulent flow over the NACA 0012 airfoil using the RANS-SA equations. Grid Nt is used, which consists of 19200 nodes divided evenly into 8 blocks. The Reynolds number based on the chord length for the test case is 4×10^6, the Mach number is 0.4, and the angle of attack is 1°. The profiles are shown in Figure 4.12. The accuracy again appears to be a problem, but it is more prevalent in the finite-difference differencing approximation which used $|\Delta\lambda| = 0.001$, which is apparently small enough to be very sensitive to error propagation from the tangent vector, as discussed previously.

The curvature profiles are complicated in this case. The s/s_{tot} vs. λ plot is helpful to put the curvature profiles into the proper context. From an arclength perspective, most of the curve is traced once λ becomes small. For the diagonal case, at $\lambda = 0.05$, s/s_{tot} is still only 0.04, indicating that only

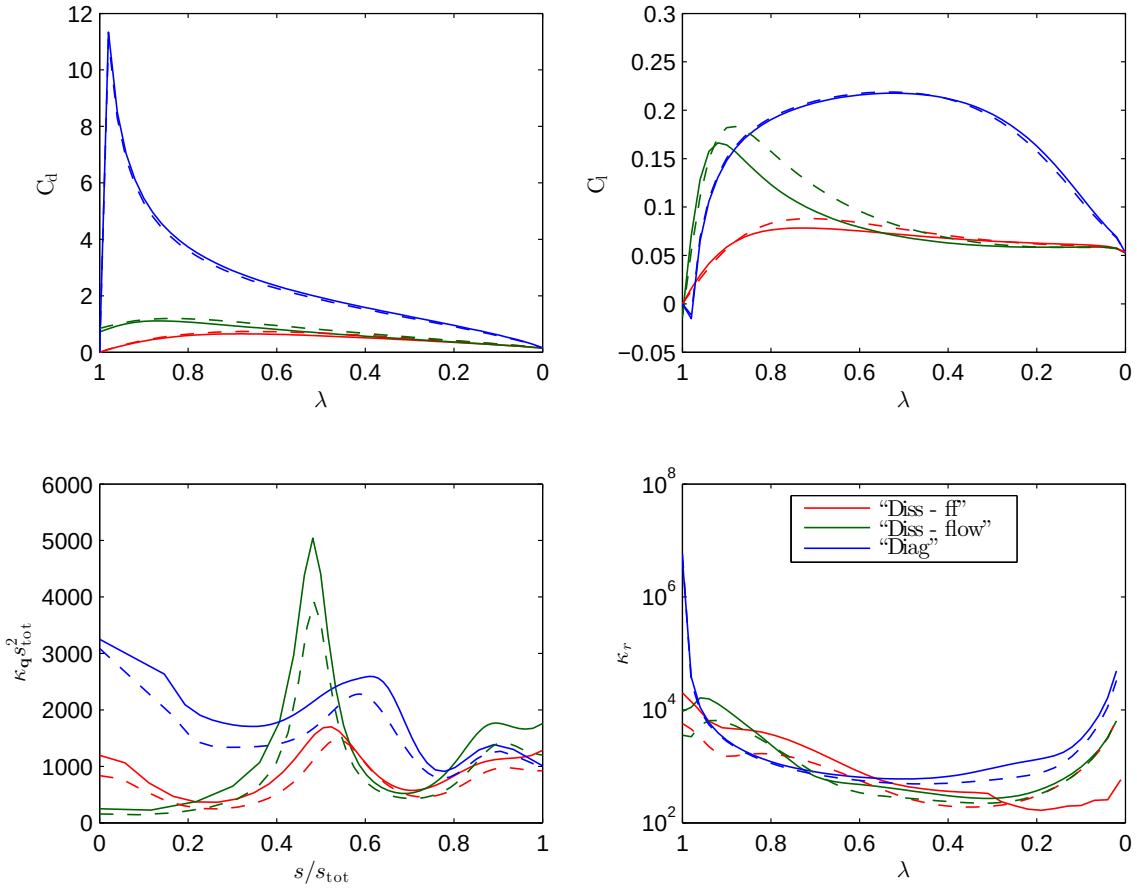


Figure 4.11: Curvature profiles and the effects of grid topology on several homotopies for the laminar ONERA M6 test case at Mach 0.3, angle of attack 1°, and Reynolds number 1×10^3; the solid lines correspond to grid MlHH and the dashed lines correspond to grid MlHC

4% of the curve has been traversed, measured in arclength units. For small λ, the curvature would thus appear much lower moving along the curve than to an outside observer using the λ coordinate system as the curvature has been stretched out along the length of the curve. A more practical assessment of the curve tracing error might be the plot on the bottom left of the figure, which shows $\kappa_{\mathbf{q}} \left(\Delta s \Big|_{|\Delta\lambda|=0.01} \right)^2$ vs. s/s_{tot}, providing an estimate of the relative predictor error incurred by a continuation algorithm using a constant step size of $|\Delta\lambda| = 0.01$.

The deformation for this turbulent case is clearly very imbalanced. This issue cannot be addressed by a simple change of coordinates.

4.9.9 Curvature Profile for a 3D Turbulent Flow

This case is a three-dimensional turbulent flow over the ONERA M6 wing. The study is performed using mesh MtHH, which has an H-H topology and consists of 2.33×10^6 nodes split evenly into 192 blocks. The flow conditions are Mach 0.4, angle of attack 3°, and Reynolds number 1×10^6. The data is shown in Figure 4.13. Though a sizable error might be assumed for this calculation based on past

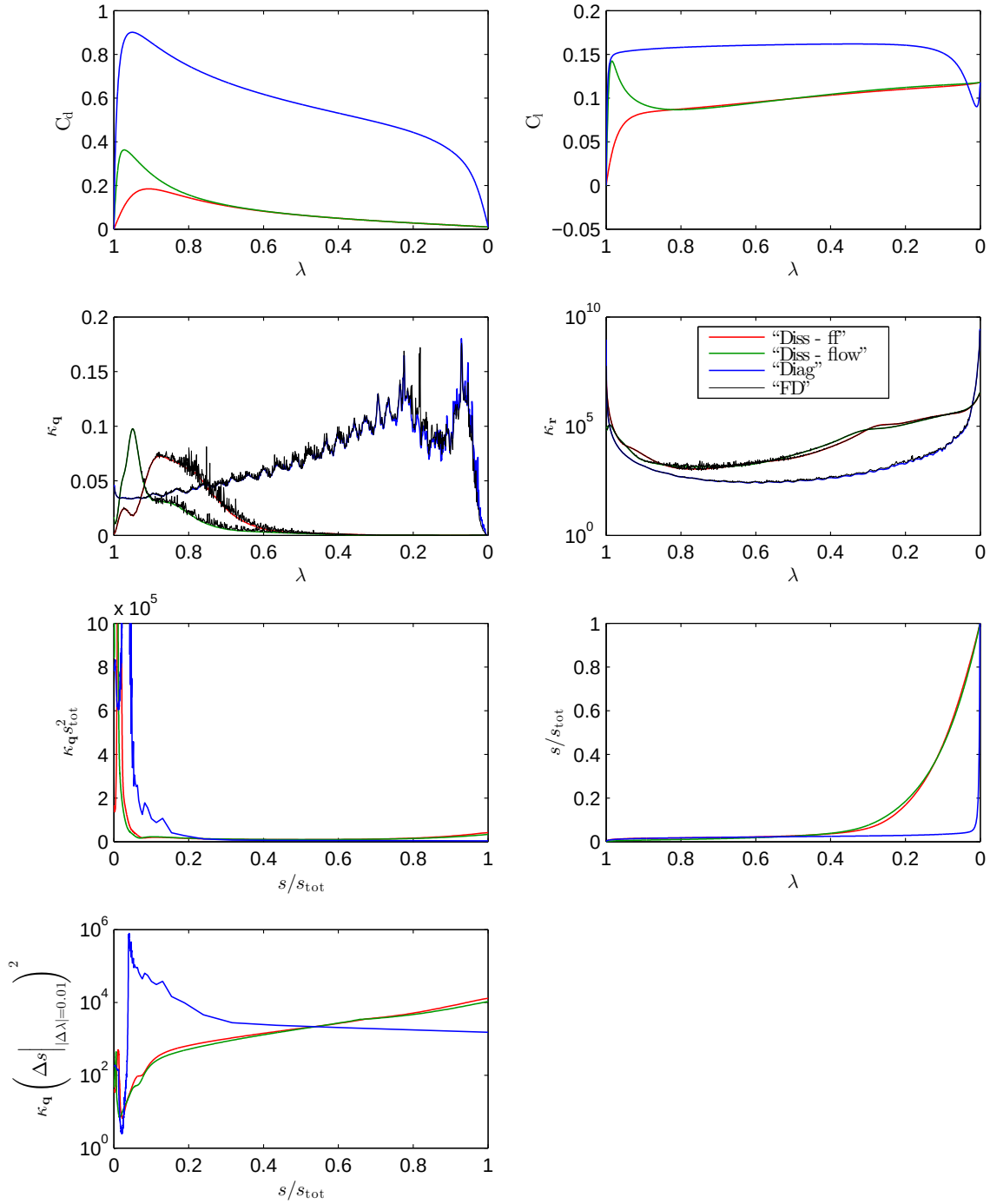


Figure 4.12: Curvature profiles of several homotopies including comparison to the finite-difference (FD) approximations for the turbulent NACA 0012 test case at Mach 0.4, Reynolds number 4×10^6, and angle of attack of 1°

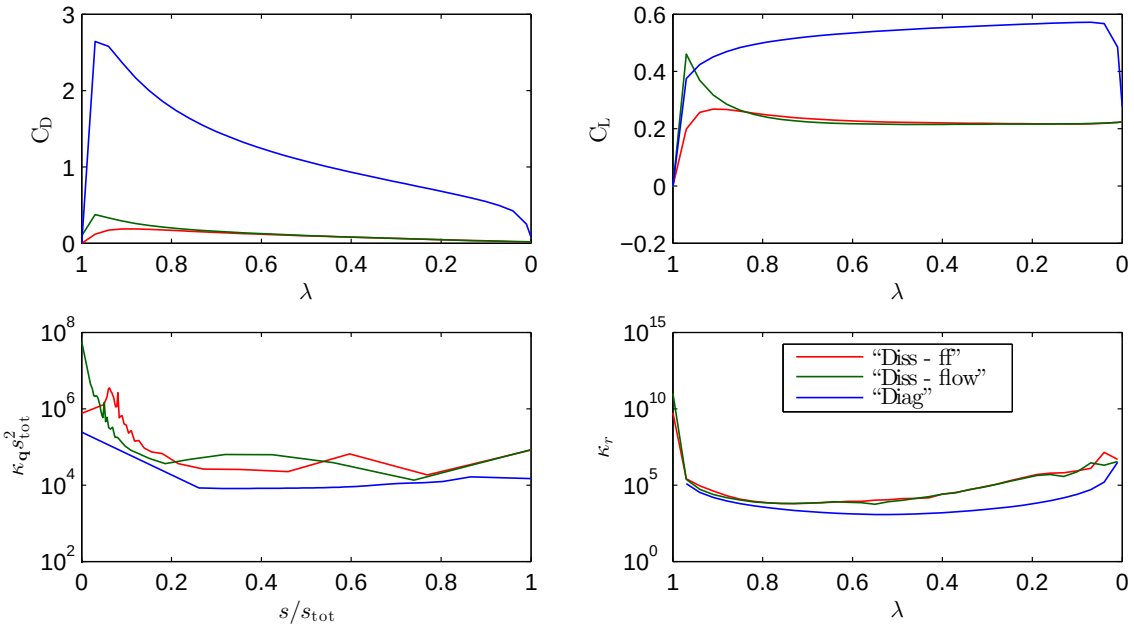


Figure 4.13: Curvature profiles of several homotopies for the turbulent ONERA M6 test case at Mach 0.4, Reynolds number 10^6, and angle of attack of 3°

experience, the general trend in the curvature profiles can still be observed. The curvature profiles for this case appear similar to what was recorded for the turbulent NACA 0012 case. The surrogate curve profiles also appear similar to what was recorded for the NACA 0012 case.

Chapter 5

Predictor-Corrector Algorithm

To our knowledge, there are two classes of homotopy continuation algorithm which have been studied prior to our work: the predictor-corrector (PC) algorithm, which approximately traces the continuous curve defined by the homotopy, and piecewise-linear (PL) methods.

5.1 Piecewise-Linear Algorithms

Piecewise-linear continuation algorithms originated with the work of Lemke and Howson [95] and Lemke [94] and were developed further by Eaves [37, 38] and Eaves and Scarf [39].

The basic idea of PL methods is that the vector space $\mathbb{R}^N$ which contains the homotopy curve is imagined to be filled contiguously with simplices. If the homotopy curve enters a simplex then it must also exit the same simplex at another location. The neighbouring simplex in which the curve will also be present can be determined by determining by which face of the current simplex the curve exits and the curve is tracked in this fashion.

A drawback to the PL algorithm is that the residual $\mathcal{H}(\mathbf{q}, \lambda)$ must be evaluated at the vertices of each simplex and arranged in a dense matrix. Since a simplex in $\mathbb{R}^N$ has $N + 1$ vertices, this is an exceedingly expensive task for homotopies in high-dimensional space. Such methods are not studied in this thesis.

5.2 Overview of the Predictor-Corrector Method

As the name suggests, the predictor-corrector algorithm consists of two phases: the predictor phase and the corrector phase. The two phases are applied repeatedly until traversing is complete.

The objective of the predictor phase is to obtain a suitable starting guess for the $k+1$st sub-problem using the estimated solution at the k-th sub-problem, a trajectory, and a distance (step-length) to travel along that trajectory. The simplest and most common predictors are Euler predictors. In this case, the predictor update at the kth step is given by

$$\mathbf{u}_{k+1}^{(0)} = \mathbf{u}_k^{(p_k)} + h_k \mathbf{d}_k, \tag{5.1}$$

Algorithm 5.1: Homotopy continuation based on the predictor-corrector (PC) framework

Initialize: Set $\lambda = 1$ and solve $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ if necessary
Predictor iterations: while $\lambda > 0$ **do**
  Corrector iterations: while $\|\mathcal{H}\|$ *is above some (relative) tolerance* **do**
    Get $\mathcal{H}$, $\|\mathcal{H}\|$, and $\frac{\partial}{\partial\lambda}\mathcal{H}$
    Form and factor the preconditioner based on the matrix $\nabla\mathcal{H}$
    Solve the linear system $\nabla\mathcal{H}\Delta\mathbf{q} = -\mathcal{H}$ and take a Newton step
  end
  Calculate the predictor direction
  Calculate the step-length
  Take a predictor step, updating λ and $\mathbf{q}$
end
Inexact Newton Phase: Solve $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ to some tolerance using the inexact Newton method

$$h_k \in \mathbb{R}_+,\ \mathbf{u}_k \in \mathbb{R}^{N+1},\ \mathbf{d}_k \in \mathbb{R}^{N+1},$$

where $\mathbf{u}_k = [\mathbf{q}_k; \lambda_k]$, h_k is the step-length, $\mathbf{d}_k$ is the step direction, and $p_k \in \mathbb{Z}$ is the number of iterations required to converge the k-th sub-problem.

The objective of the corrector phase is to solve the nonlinear sub-problem at λ_k. The sub-problem $\mathcal{H}(\mathbf{q}_k, \lambda_k) = \mathbf{0}$ is solved inexactly using the inexact Newton method. Newton iterations are performed until the relative residual $\left\|\mathcal{H}\left(\mathbf{q}_k^{(n)}, \lambda_k\right)\right\| / \left\|\mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k\right)\right\|$ is reduced below some user-defined tolerance $\mu_k \in \mathbb{R}$. Typically, $\mu_k \in [0.1, 0.5]$. A pseudo-code of the algorithm is provided as Algorithm 5.1.

5.3 The Predictor Step Direction

Obtaining a suitable starting guess for each corrector phase sub-problem will reduce the number of iterations required to meet the corrector phase residual μ_{k+1} and improve the success rate when applying Newton's method.

5.3.1 Embedding Algorithms

The simplest way to apply the predictor step is to simply use the final iterate of the $k+1$st sub-problem as the initial guess for the kth sub-problem, updating only λ. This is called an *embedding* algorithm and can be applied by using equation (5.1) with direction vector $\mathbf{d} = (0, 0, \ldots, 0, -1)$. This results in the predictor update $\lambda_{k+1} \leftarrow \lambda_k - h_k$, and $\mathbf{q}_{k+1}^{(0)} \leftarrow \mathbf{q}_k^{(p_k)}$.

5.3.2 Predictors Based on the Tangent Vector

The predictor step can be applied using equation (5.1) with direction vector $\mathbf{d}$ set as some approximation to the tangent vector. The tangent vector can be calculated accurately using the direct algebraic method presented in Section 4.5. The only sources of error in this calculation are the error due to solving the linear system inexactly, any inaccuracy in the matrix-vector products used in the linear solver, and rounding error. This predictor can be quite effective. However, as the method requires the solution to

a linear system of equations, the CPU cost is also significant.

The secant method is an approximation to the tangent vector using first-order backwards finite-differencing. This calculation has been presented as equation (4.34) but is repeated here in the notation consistent with equation (5.1):

$$\mathbf{d}_k = \frac{1}{\left\| \mathbf{u}_k^{(p_k)} - \mathbf{u}_{k-1}^{(p_{k-1})} \right\|} \left(\mathbf{u}_k^{(p_k)} - \mathbf{u}_{k-1}^{(p_{k-1})} \right). \tag{5.2}$$

Though this approximation to the tangent is considerably less accurate than the direct method presented in Section 4.5, it comes at very little cost. In some applications, using the secant approximation can be more cost-effective than the direct tangent, though it also comes at the price of reduced robustness.

5.3.3 Higher-Order Predictors

A higher order predictor can be constructed using the Taylor polynomial

$$c(s + \Delta s) = c(s) + \sum_{j=1}^{n} \frac{(\Delta s)^j}{j!} c^{(j)}(s) \tag{5.3}$$

centred at the current corrector point. The derivatives needed in the construction of such a predictor can be calculated from the procedure presented in Section 4.6. It is also possible to apply the predictor using a λ parametrization, in which case the Taylor polynomial is in terms of $\Delta\lambda$ instead of Δs. However, using an arclength parametrization for the predictor is more convenient if the step-length adaptation is also based on an arclength parametrization, which it typically is.

Applying a predictor in the form of a Taylor polynomial is an appealing prospect because even though it comes at the cost of additional linear solves the Jacobian matrices associated with these linear solves are shifted closer to $\lambda = 1$ and hence should be better conditioned. As an example, consider a scenario where building a predictor from curve derivatives of order 4 allows for double the step size compared to using a predictor built on the tangent vector. Then if each corrector phase requires on average 3 linear solves then the number of linear solves will be the same in either case but the algorithm using the higher order predictor will be cheaper because the additional linear systems will be closer to $\lambda = 1$ and are expected to be better conditioned. Additionally, each additional linear solve required to build the fourth derivative has the same Jacobian matrix, so the same preconditioner can be reused for each additional linear solve without incurring any performance penalty.

An additional challenge not yet addressed which arises when constructing higher order predictors based on the Taylor polynomial (5.3) is that the expression is in terms of the change in arclength Δs, and it is sometimes necessary to perform the predictor update for a specific $\Delta\lambda$, not Δs. However, this is easily solved by recognizing that a lower bound for Δs can be obtained if the λ-component of the tangent predictor equation is solved for Δs with a given $\Delta\lambda$:

$$\Delta s \geq \frac{\Delta\lambda}{\dot{\lambda}(s)}. \tag{5.4}$$

Since high order predictors will result in high order polynomials in Δs which may contain multiple solutions, Newton's method is not recommended for solving for Δs as only one particular solution is of

interest and Newton's method cannot be relied on to converge to it. Instead, a simple bisection method can be used, setting the value for Δs at the left end of the search interval by equation (5.4) and then increasing Δs until $\lambda < \lambda_{\text{tar}}$, $\lambda_{\text{tar}} \in \mathbb{R}$ to determine an appropriate Δs to use as the right endpoint of the interval. The bisection method is then used to solve for Δs, iterating until

$$\left| \lambda - \lambda_{\text{tar}} + \sum_{j=1}^{n} \frac{(\Delta s)^j}{j!} \overset{(j)}{\lambda}(s) \right| < \mu_{\text{tar}}, \tag{5.5}$$

where $\mu_{\text{tar}} \in \mathbb{R}$ is some tolerance and is set to 10^{-4} in this study. If equation (5.4) is not satisfied or a suitable right interval point cannot be found then this is an indication that λ_{tar} corresponds to a point on the curve which is outside of the radius of convergence of the Taylor series and so the Taylor polynomial cannot be used to predict this point.

There are other ways in which higher-order predictors can be formed. It is possible to use polynomial extrapolation [1] such as Newton or Lagrange extrapolation, which use previous solution points, or Hermite extrapolation, which additionally requires previous tangent calculations. Such methods are not considered here; they are not expected to be very effective because the curve points are generally not solved very accurately and the step sizes taken are relatively large. For these reasons, it should not be expected that future points can be predicted accurately from previous data points and such methods already suffer from stability problems. Lundberg and Poore [100] have advocated the use of a variable-order Adams-Bashforth method for certain applications. However, our experimentation with second-order Adams-Bashforth has not demonstrated any benefit for our applications.

Another class of predictors which have appeared in the literature, most commonly in the field of computational structural mechanics but also for simple fluid mechanics problems, are known as *asymptotic numerical methods* [26, 87, 163]. The objective is to build a predictor based on a polynomial of the form:

$$\mathbf{q}(a) = \mathbf{q}_k + a\tilde{\mathbf{q}}_{(1)} + a^2\tilde{\mathbf{q}}_{(2)} + \ldots + a^n\tilde{\mathbf{q}}_{(n)}, \tag{5.6}$$

$$\lambda(a) = \lambda_k + a\tilde{\lambda}_{(1)} + a^2\tilde{\lambda}_{(2)} + \ldots + a^n\tilde{\lambda}_{(n)}, \tag{5.7}$$

where $a \in \mathbb{R}$ is usually an estimate of the arclength parameter, and the vectors $\tilde{\mathbf{q}}_{(j)} \in \mathbb{R}^N$ and $\tilde{\lambda}_{(j)} \in \mathbb{R}$ are unknowns which must be solved for as part of the methodology. This formulation requires that the system $\mathcal{H}$ can be written in terms of polynomial operators such as linear, quadratic, and possibly higher order. This can be done for the discrete Euler and laminar Navier-Stokes equations if the viscosity and spectral radius in the dissipation operators are treated as constant for the predictor calculation. However, the RANS-SA model equation is highly nonlinear and cannot be written in this form. For this reason, such predictors are not considered here.

5.3.4 Predictor Performance Analysis

The prediction capabilities of the Taylor polynomial given by equation (5.3) are investigated. The test case is inviscid flow over the two-dimensional NACA 0012 airfoil at Mach 0.3 and angle of attack of 1°. Grid Ne is used, which has an H topology and consists of 15390 nodes divided evenly into 18 blocks. The homotopy system applied is the dissipation operator with far-field boundary conditions.

Derivatives of up to order 9 are generated at $\lambda = 0.9$, 0.7, and 0.4, and the residual is calculated

along the trajectory predicted by the Taylor polynomial. To ensure that the derivatives are calculated as accurately as possible, GCROT is used to solve the linear systems which appear in the tangent and curvature calculations to a relative tolerance of 10^{-10}. However, the actual final relative residual is not actually this low due to the use of FDMVPs in the linear solvers, which limits the accuracy of the matrix-vector products. Note that the predictor based on the first derivative ($n = 1$) is equivalent to the tangent predictor applied using equation (5.1).

As shown in Figure 5.1a, the higher order predictors predict the curve extremely well in a small radius Δs around s_0, usually corresponding to $|\Delta\lambda|$ values between 0.1 and 0.2, but the benefit of the higher order predictor quickly diminishes outside of this radius. When the predictor step is large, the higher order predictor can often be seen to give a worse prediction than even the tangent ($n = 1$) predictor. This is expected behaviour for a Taylor polynomial since the approximation is only valid within some radius. However, the various sources of rounding and truncation error from the derivative calculations may also be inhibiting performance.

A second study has also been performed to investigate how sensitive the predictor is to error. This study is analogous to the previous one but the linear systems were only solved to a relative residual of 10^{-2}. The predicted trajectory is compared to that of the previous study in Figure 5.1b. This study is necessary for assessing the viability of the predictor as part of a continuation algorithm since solving each linear system to a relative residual of 10^{-10} is too expensive to be used in a cost-competitive algorithm.

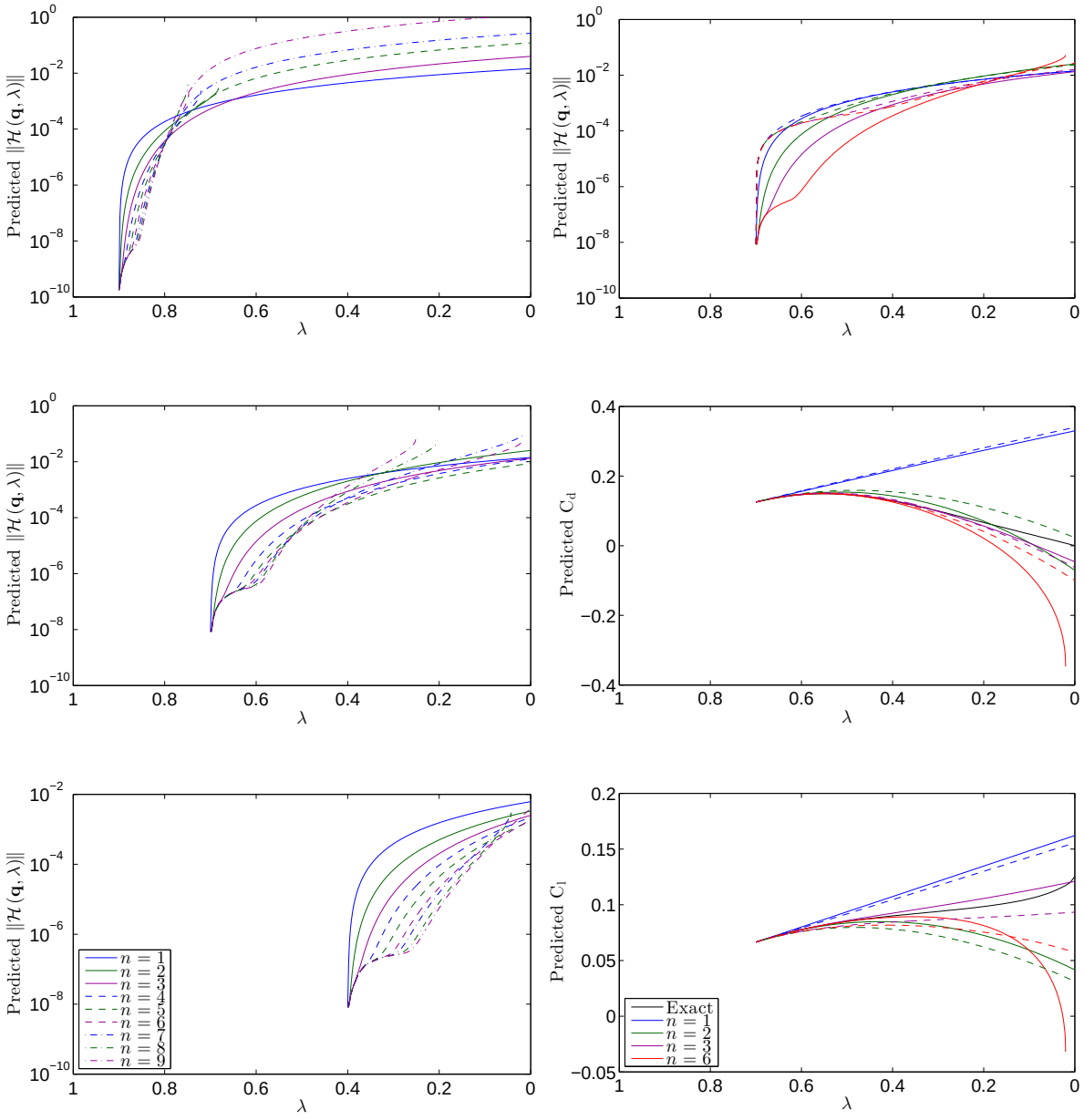
While $\|\mathcal{H}(\mathbf{q}, \lambda)\| < 10^{-6}$ can no longer be maintained near s_0 when the accuracy of the derivative approximations is relaxed, the curve is still predicted quite well. Comparing the predictions of the lift coefficient C_l and drag coefficient C_d surrogate curves, also shown in Figure 5.1b, the performance degradation of the predictor when relaxing the accuracy of the curve derivatives is not as severe as it would appear when only considering the residual. While being able to maintain $\|\mathcal{H}(\mathbf{q}, \lambda)\| < 10^{-6}$ along the predicted trajectory is remarkable, this level of accuracy is unnecessary for effective continuation, and the high-order polynomial predictor is still effective when the linear solver tolerance is relaxed.

It is apparent from the C_l and C_d plots in Figure 5.1b that the curve can be predicted reasonably well within a larger radius with higher order predictors than with a tangent predictor and at a reasonable cost increase. However, the radius for which the higher order predictor is valid remains unpredictable. When considering the suitability of these predictors as part of a continuation algorithm, robustness is a concern.

5.4 Step-length Adaptation

Step-length adaptation algorithms are used to adjust the step-length automatically during traversing. The step-length can either be reduced to improve robustness or increased to increase the rate of traversing. These methods do not provide an explicit expression for the step-length. Rather, adjustments are applied to the current step-length value based on collected data.

In conformity with the notation of Allgower and Georg [1], the output of the step-length adaptation algorithm is denoted as the factor $f \in \mathbb{R}$. Once f is determined, the condition $f_{\min} \leq f \leq f_{\max}$ is enforced. The local step-length $h_k = \frac{h_{k-1}}{f}$ is then calculated and the condition $h_{\min} \leq h \leq h_{\max}$ is enforced. These upper and lower bounds on f and h are user-supplied. However, since h is generally taken as a measure of arclength, values of $h_{\max}$ and $h_{\min}$ are not intuitive and so $|\Delta\lambda|_{\min} \leq |\Delta\lambda| \leq |\Delta\lambda|_{\max}$



(a) $\tau_l = 10^{-10}$ (b) $\tau_l = 10^{-10}$ (solid) and $\tau_l = 10^{-2}$ (dashed)

Figure 5.1: Residual at the predicted state from Taylor polynomials of order n calculated at $\lambda = 0.9$, 0.7, and 0.4 for the inviscid subsonic NACA 0012 case; the effect of the linear solver tolerance τ_l is investigated

is enforced instead.

The method of *asymptotic expansions*, first presented by Georg [47], is a relatively simple method based on some of the physical properties of consecutive predictor and corrector points. A slightly modified version of the algorithm presented by Allgower and Georg [1] is applied here using two criteria: the distance $\delta \in \mathbb{R}$ between the predictor point and the corrector point and the angle $\phi \in \mathbb{R}$ between consecutive tangent vectors. Explicit expressions for these quantities are given by

$$\delta = \left\| \mathbf{q}_k^{(p_k)} - \mathbf{q}_k^{(0)} \right\|, \tag{5.8}$$

$$\phi = \arccos\left(\dot{c}_k^{(p_k)}(s) \cdot \dot{c}_{k-1}^{(p_{k-1})}(s) \right). \tag{5.9}$$

Using the method of asymptotic expansions [1], the factor f is calculated as

$$f = \max\left\{ \sqrt{\frac{\delta}{\tilde{\delta}}}, \frac{\phi}{\tilde{\phi}} \right\}. \tag{5.10}$$

A method for determining suitable values for $\tilde{\delta}$ and $\tilde{\phi}$ is presented in Section 5.5.

Another option for step-length adaptation is Newton-Kantorovich methods [32], which adapt the step-length based on error models for Newton's method. This type of algorithm requires as input a target number of Newton iterations per corrector phase and makes adjustments at each corrector phase based on how many corrector steps were actually applied and an estimate of the error at the second-to-last corrector iteration. Though this method requires the solution to two nonlinear algebraic equations, both are scalar equations with a single solution and can easily be solved using the bisection method at negligible computational cost and with guaranteed convergence.

Some experimentation with the Newton-Kantorovich method of Den Heijer and Rheinboldt [32] has revealed some practical limitations: Newton's method is usually not converged far enough in the corrector phase for the error models to be accurate, and the target number of iterations is often difficult for the user to predict. Hence, this method is not included as part of the step-length adaptation.

5.5 κ-Scaling

The scaling of $|\Delta\lambda|$ relative to $\|\Delta\mathbf{q}\|$ is an important consideration for step-length adaptation, as it affects how much $\Delta\mathbf{q}$ is emphasized relative to $\Delta\lambda$.

Consider a predictor step-length $\Delta\lambda$ and a following corrector step of magnitude $\left\|\Delta\mathbf{q}_k^{(0)}\right\|$. The objective is to develop a relationship between these two quantities and θ, which is the angle between the direction of the current corrector iterate $\Delta\mathbf{q}_k^{(0)}$ and a corrector step along shortest line connecting the current point to the curve, denoted $\Delta\mathbf{q}_k^{(0_c)}$. The problem is formulated in this way because θ is an intuitive quantity that we might propose to be around 5°.

The relationship works out to the following:

$$\Delta\lambda = \pm \left\|\Delta\mathbf{q}_k^{(0)}\right\| \cot\theta. \tag{5.11}$$

The derivation is provided in Appendix E.

Equation (5.11) is rather informative. In the case of $\left\|\Delta\mathbf{q}_k^{(0)}\right\| \gg |\Delta\lambda|$, θ_k evaluates to approximately $\pm\frac{\pi}{2}$, and equation (E.10) further indicates that $\left\|\Delta\mathbf{u}^{(0_c)}\right\| \approx |\Delta\lambda|$. Hence, if the state variables are scaled too large with respect to λ, then a corrector iteration applied in the minimum-norm sense (instead of in the constant-λ sense described in Section 5.2) will act in the opposite (considering equation (E.9)) direction of an embedding step and with the same magnitude. In other words, the first corrector step will actually reverse the previous embedding step under these circumstances. Though utilizing a minimum-norm corrector is not cost-effective for our intended applications, the impact on step-length adaptation is clear.

To correct this problem, λ is scaled by a factor κ, which can be determined from a numerical algorithm. Considering equation (5.11), the following construction is used:

$$|\Delta\lambda|_{\text{ideal}} = \|\Delta\mathbf{q}_{\text{ref}}\| \cot\theta_{\text{ref}},\ \theta_{\text{ref}} \in \mathbb{R}, \tag{5.12}$$

where θ_{ref} can be user-supplied. The scaling is implemented by modifying the CHC equation (4.67) or GHC equation (4.70) to the following:

$$\mathcal{H}(\mathbf{q}, \lambda_k) = (\kappa - \lambda_k)\,\mathcal{R}(\mathbf{q}) + \lambda_k \mu \mathcal{G}(\mathbf{q}) = \mathbf{0},\ \text{(Convex)} \tag{5.13}$$

$$\mathcal{H}(\mathbf{q}, \lambda_k) = [\kappa - \lambda_k(1-\mu)]\,\mathcal{R}(\mathbf{q}) - \lambda_k \mu \mathcal{R}(\mathbf{q}_0) = \mathbf{0},\ \text{(Global)} \tag{5.14}$$

$$k \in [0, m],\ \lambda_k, \mu, \kappa \in \mathbb{R},\ \lambda_0 = \kappa,\ \lambda_m = 0,\ \lambda_{k+1} < \lambda_k,$$

$$\kappa = \frac{|\Delta\lambda|_{\text{ideal}}}{|\Delta\lambda|} = C_\kappa \kappa_{\text{f}} \cot\theta_{\text{ref}},\ C_\kappa = \frac{\|\Delta\mathbf{q}\|}{|\Delta\lambda|}, \tag{5.15}$$

$$\mathcal{H} : \mathbb{R}^{N+1} \to \mathbb{R}^N,\ \mathcal{G} : \mathbb{R}^N \to \mathbb{R}^N,\ \mathcal{R} : \mathbb{R}^N \to \mathbb{R}^N, C_\kappa, \kappa_{\text{f}}, \theta_{\text{ref}}, \mu, \Delta\lambda \in \mathbb{R},$$

where the scaling factor κ_{f} has been supplied for additional user input since θ_{ref} may be unintuitive, and $\Delta\lambda$ is the change in λ that accompanies $\Delta\mathbf{q}$. Generally, this formula is intended to be applied without user intervention, since user intervention is highly unintuitive and fine-tuning of κ is not necessary.

To use equation (5.15), an estimate for C_κ is needed. Clearly this factor will be problem-dependent. One possible solution is to calibrate the flow solver by running several test cases, and correlating C_κ to the flow variables and grid parameters according to some target value for θ_{ref}. Some numerical testing has shown that the relationship between C_κ and Mach number is fairly linear and that C_κ is relatively insensitive to angle of attack and Reynolds number.

The analysis provided in this section also gives a method for calibrating the target distance $\tilde{\delta}$ needed for the distance to the curve criterion used for step-length adaptation. Since the length of a corrector step after an embedding step with step size $|\Delta\lambda|$ is given by equation (E.10), then the interpolated distance at some reference predictor length $|\Delta\lambda|_{\text{ref}}$ for the standard constant-λ corrector is given by

$$\tilde{\delta} = C_\kappa \delta_f |\Delta\lambda|_{\text{ref}}, \tag{5.16}$$

where $|\Delta\lambda|_{\text{ref}} \in [0, 1]$ can be taken as the initial step-length, and $\delta_f \in \mathbb{R}$ is a user-supplied factor. It is far more intuitive to choose this factor than to choose $\tilde{\delta}$ directly.

Though the homotopy has been modified such that λ now takes values in the interval $[0, \kappa]$ instead of $[0, 1]$, it is more intuitive to think of λ as taking values within the interval $[0, 1]$, so this convention is

adopted throughout the remainder of this thesis.

5.6 An Algorithm for Estimating the μ-Scaling Parameter

In Section 4.7, the application and effects of μ-scaling were discussed, in which $\mu = \mu_{\mathrm{a}}\mu_{\mathrm{u}}$, where μ_{a} is a benchmark value and μ_{u} is a user-supplied value and is unity by default. In this section is presented a numerical algorithm for determining a suitable value of μ_{a}.

The value of μ_{a} can be optimized for a particular flow solve by first creating a discrete estimate of the curve using the predictor-corrector method with a relatively small and constant $|\Delta\lambda|$ and some initial guess for μ_{a}. The corrector phase sub-problems are solved to a tight tolerance to attempt to establish the curve with some reasonable accuracy. When the curve tracing is complete, minimization is applied to the H^2-norm of some quantification of predictor performance, integrating in the λ domain.[1] For example, the derivative of the starting error with respect to λ can be used to quantify the predictor performance. The minimization problem is:

$$\min_{\mu} r\left(\mu_{\mathrm{a}}\right), \tag{5.17}$$

where

$$r\left(\mu\right) = \left\| \frac{\mathrm{d}e}{\mathrm{d}\lambda} \right\|^2_{2,[0,1]} = -\int_1^0 \left[\frac{de}{d\lambda}\right]^2 d\lambda \rightarrow \sum_k \left[\frac{e^{(0)}_{k+1} - e^{(p_k)}_k}{\lambda_{k+1} - \lambda_k}\right]^2 |\Delta\lambda| \approx \sum_k \frac{\left[e^{(0)}_{k+1}\right]^2}{|\lambda_{k+1} - \lambda_k|}, \tag{5.18}$$

$$e^{(n)}_k \in \mathbb{R},\ r : \mathbb{R} \rightarrow \mathbb{R}.$$

The error $e^{(n)}_k \in \mathbb{R}$ is defined as the magnitude of the difference in $\mathbf{q}^{(n)}_k$ and the actual curve value at λ_k. It can be estimated as $e^{(n)}_k \approx \left\| \mathbf{q}^{(n)}_k - \mathbf{q}^{(p_k)}_k \right\|$.

To minimize $r\left(\mu_{\mathrm{a}}\right)$, it would appear that $r\left(\mu_{\mathrm{a}}\right)$ would need to be calculated for a range of μ_{a}, requiring for the curve to be traced numerous times. However, this is not the case. Considering equation (4.68), the value of $r\left(\mu_{\mathrm{a}}\right)$ can in fact be determined for any μ_{a} by tracing the curve a single time and applying the change of coordinates (4.68) to λ. Of course, for large changes in μ_{a}, the Riemann sum approximation to the H^2-norm is less accurate, so if μ_{a} changes by a large amount (e.g. factor of 5 or more) then the curve should be re-traced and the minimization algorithm repeated with an updated value of μ_{a}.

5.7 Scaling Considerations for the RANS-SA Equations

As discussed in Section 3.6.2, the turbulence variable $\tilde{\nu}$ present in the Spalart-Allmaras model can take values similar in magnitude to the mean flow equations in some regions of the domain but can be roughly 10^3 times greater in other regions. This can adversely affect the conditioning of the linear system and hence linear solver performance. Previous researchers [25, 130] have mitigated this effect by applying a factor of 10^3 to the columns of the Jacobian corresponding to $\tilde{\nu}$, and correcting the solution when the

[1] Though the obvious procedure would be to choose μ_{a} to even out the κ_r profile as much as possible, the μ-scaling was developed before the curvature calculation was possible and so the algorithm does not make use of the curvature calculation.

linear solve is complete. The same effect could have been achieved by applying a scaling factor of 10^{-3} to $\tilde{\nu}$ directly in the flow equations.

As discussed in Section 5.3, the scaling of the flow variables affects step-length adaptation and it is important to re-scale $\tilde{\nu}$ for this purpose as well. For this reason, it has been more convenient to take the approach of scaling $\tilde{\nu}$ directly. However, we have found that the factor of 10^{-3} de-emphasizes $\tilde{\nu}$ too much for the step-length adaptation and that a factor of 10^{-1} is sufficient. An additional column scaling factor of 10^{-2} is applied when solving any linear systems of equations.

Chapter 6

Monolithic Homotopy Continuation

The motivation for the monolithic homotopy (MH) continuation algorithm comes from the dynamic inversion principle. Dynamic inversion [48, 50, 49] is a process which was developed for tracking implicitly-defined trajectories to which the inverse mapping is not analytically available and is parameter-dependent. This method was developed for applications in the field of robotics, where the parameter is time.

The problem studied by Getz and Marsden [50] is a robotic manipulator where the target trajectory is defined implicitly by a nonlinear system of equations. Assume that the target trajectory of the robotic manipulator can be represented as a continuous curve in $\mathbb{R}^N$ and consider the problem of attempting to trace this trajectory by applying a control law to the manipulator. Since real time is consumed while estimating the state and computing the control law, the control law calculated at time t will in fact be applied at time $t + \Delta t$. Hence, what is needed is not an estimate of the current state, but an estimate of some future state.

The MH method was developed by applying the dynamic inversion principle in the context of homotopy. The initial homotopy system at $\lambda = 1$ is first solved exactly or to some relative tolerance, applying Newton's method if necessary. In contrast to the PC method, the MH algorithm consists of a single phase in which both λ and $\mathbf{q}$ are updated at each iteration. The update vector consists of predictor and corrector components which are calculated simultaneously by solving a single linear system of equations with the same matrix which appears in the linear systems for the PC method. The motivation behind the development of this algorithm is to reduce the total number of linear systems that need to be solved, thereby reducing the total CPU time required for traversing, without compromising algorithm stability.

Since corrector iterations are not being performed at each point on the curve for the MH algorithm, smaller step sizes will have to be taken but at reduced cost since only one linear system must be solved. However, since the step sizes are smaller, the algorithm is more flexible regarding step-length adaptation, and more control is possible over the traversing process, with the potential for additional improvements in efficiency. As a further benefit, the MH algorithm also requires fewer input parameters, simplifying user control.

6.1 Convergence of Scalar Time-ODEs to Implicitly-Defined Trajectories

The analysis here is based on ideas from Getz [48]. Consider a scalar ordinary differential equation (ODE) in time taking the form

$$\dot{q}(t) + f(q(t), t) = 0, \tag{6.1}$$

$$q(t) \in \mathbb{R},\ t \in \mathbb{R},\ f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}.$$

Suppose that the function $f(q(t), t)$ is continuous and differentiable for all $t \in \mathbb{D}$, $\mathbb{D} \subset \mathcal{R}$ and that for all $t \in \mathbb{D}$ there is a unique $q_s(t) \in \mathbb{R}$ satisfying $f(q_s(t), t) = 0$. Then $f(q(t), t)$ cannot change sign anywhere on $q(t) > q_s(t)$ or on $q(t) < q_s(t)$.

For the ODE to converge to the trajectory $q_s(t)$, $q(t)$ should decrease with time if it is greater than $q_s(t)$, and $q(t)$ should increase with time if it is less than $q_s(t)$. More succinctly:

$$\begin{cases} \dot{q}(t) < 0 \text{ when } q(t) > q_s(t), \\ \dot{q}(t) > 0 \text{ when } q(t) < q_s(t). \end{cases} \tag{6.2}$$

Since it is clear from equation (6.1) that $f(q(t), t) > 0 \Leftrightarrow \dot{q}(t) < 0$ and $f(q(t), t) < 0 \Leftrightarrow \dot{q}(t) > 0$, the ODE will converge if $f(q(t), t)$ satisfies the following conditions:

$$\begin{cases} f(q(t), t) > 0 \text{ when } q(t) > q_s(t), \\ f(q(t), t) < 0 \text{ when } q(t) < q_s(t). \end{cases} \tag{6.3}$$

6.2 Convergence of Vector-Valued Time-ODEs to Implicitly-Defined Trajectories

Consider the vector-valued ODE:

$$\dot{\mathbf{q}}(t) + \mathcal{F}(\mathbf{q}(t), t) = \mathbf{0}, \tag{6.4}$$

$$\mathbf{q}(t) \in \mathbb{R}^N,\ t \in \mathbb{R},\ \mathcal{F} : \mathbb{R}^N \to \mathbb{R}^N.$$

Assume that the function $\mathcal{F}(\mathbf{q}(t), t)$ is continuous and differentiable in all elements for all $t \in \mathbb{D}$, $\mathbb{D} \subset \mathbb{R}$ and that there exists a unique trajectory $\mathbf{q}_s(t) \in \mathbb{R}^N$ satisfying $\mathcal{F}(\mathbf{q}_s(t), t) = \mathbf{0}$ for all $t \in \mathbb{D}$. A sufficient (but overly restrictive) property which will ensure that the ODE given by equation (6.4) converges to $\mathbf{q}_s(t)$ is that for any perturbation $\Delta\mathbf{q}(t) \in \mathbb{R}^N$ on the solution, components of the perturbation which are positive will cause the corresponding component of $\mathcal{F}(\mathbf{q}(t), t)$ to become positive, and components of the perturbation which are negative will cause the corresponding component of $\mathcal{F}(\mathbf{q}(t), t)$ to become negative. This is expressed by the component-wise equation:

$$\begin{cases} \mathcal{F}_{[i]}(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t) > 0 \text{ for } \Delta\mathbf{q}_{[i]}(t) > 0, \\ \mathcal{F}_{[i]}(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t) = 0 \text{ for } \Delta\mathbf{q}_{[i]}(t) = 0, \\ \mathcal{F}_{[i]}(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t) < 0 \text{ for } \Delta\mathbf{q}_{[i]}(t) < 0. \end{cases} \tag{6.5}$$

This equation can also be expressed in a way that includes a set of parameters

$$\{\beta_i | \beta_i \in \mathbb{R},\, \beta_i > 0,\, i = 1, \ldots, N\},$$

which will be useful for analysis later:

$$\begin{cases} \mathcal{F}_{[i]}\left(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t\right) \geq \beta_i \Delta\mathbf{q}_{[i]}(t) & \text{for } \Delta\mathbf{q}_{[i]}(t) > 0, \\ \mathcal{F}_{[i]}\left(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t\right) = \beta_i \Delta\mathbf{q}_{[i]}(t) & \text{for } \Delta\mathbf{q}_{[i]}(t) = 0, \\ \mathcal{F}_{[i]}\left(\mathbf{q}_s(t) + \Delta\mathbf{q}(t), t\right) \leq \beta_i \Delta\mathbf{q}_{[i]}(t) & \text{for } \Delta\mathbf{q}_{[i]}(t) < 0. \end{cases} \tag{6.6}$$

For any of the three conditions in the above equation, multiplying the left and right sides of the inequality by $\Delta\mathbf{q}_{[i]}(t)$ results in the same expression:

$$\Delta\mathbf{q}_{[i]} \mathcal{F}_{[i]}\left(\mathbf{q}_s + \Delta\mathbf{q}, t\right) \geq \beta_i \Delta\mathbf{q}_{[i]}^2. \tag{6.7}$$

Performing summation over all i:

$$\Delta\mathbf{q}^T \mathcal{F}\left(\mathbf{q}_s + \Delta\mathbf{q}, t\right) \geq \beta \left\|\Delta\mathbf{q}\right\|^2, \tag{6.8}$$

$$\beta \in \mathbb{R},\ \beta = \min_i \beta_i.$$

It is shown by the conclusion of Section 6.3 that the condition (6.8) is sufficient for stability in time. Though this condition is a consequence of condition (6.5), it is much less of a restriction on $\mathcal{F}$ and it is not necessary that $\mathcal{F}$ actually satisfy condition (6.5). Deriving this inequality from condition (6.5) was done only to give some intuitive understanding of its meaning. The existence of $\beta > 0$ such that equation (6.8) is satisfied will ensure that the ODE (6.4) converges asymptotically to the solution curve of $\mathcal{F}(\mathbf{q}, t) = \mathbf{0}$. In some cases, such a β may only exist in some ball of radius r around the curve. As shown later in this chapter, the value of β also determines the rate of convergence.

This presentation of time-convergence is useful for the arguments presented in this chapter but it is generally difficult to know if a given $\mathcal{F}(\mathbf{q}, t)$ satisfies condition (6.8). This condition can be written in a more familiar form using the Taylor expansion:

$$\Delta\mathbf{q}^T \mathcal{F}\left(\mathbf{q}_s + \Delta\mathbf{q}, t\right) = \Delta\mathbf{q}^T \mathcal{F}\left(\mathbf{q}_s, t\right) + \Delta\mathbf{q}^T \nabla_{\mathbf{q}} \mathcal{F}\left(\mathbf{q}_s, t\right) \Delta\mathbf{q} + \mathcal{O}\left(\left\|\Delta\mathbf{q}\right\|^2\right). \tag{6.9}$$

Since $\mathcal{F}(\mathbf{q}_s, t) = \mathbf{0}$, and ignoring the $\mathcal{O}\left(\left\|\Delta\mathbf{q}\right\|^2\right)$ terms, substituting equation (6.9) into condition (6.8) gives

$$\Delta\mathbf{q}^T \nabla_{\mathbf{q}} \mathcal{F}\left(\mathbf{q}_s, t\right) \Delta\mathbf{q} \geq \beta \left\|\Delta\mathbf{q}\right\|^2. \tag{6.10}$$

Thus if condition (6.8) holds then the Jacobian of $\mathcal{F}(\mathbf{q}_s, t)$ must be positive definite, which is a more intuitive and familiar concept than condition (6.8).

6.3 Dynamic Inversion Principle

Since for general $\mathcal{F}(\mathbf{q}(t), t)$ equation (6.8) may not hold, it can be useful to introduce an operator $\mathcal{F}^*(\mathbf{q}(t), t)$, $\mathcal{F}^* : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ such that $\mathcal{F}^* \mathcal{F}(\mathbf{q}(t), t)$ does satisfy equation (6.8). If $\mathcal{F}^*$ is also

constructed such that $\mathcal{F}^*(\mathbf{w},t)=\mathbf{0}\Leftrightarrow\mathbf{w}=\mathbf{0}$ then $\mathcal{F}^*\mathcal{F}(\mathbf{q}(t),t)=\mathbf{0}\Leftrightarrow\mathcal{F}(\mathbf{q}(t),t)=\mathbf{0}$, and hence the ODE $\dot{\mathbf{q}}(t)+\mathcal{F}^*\mathcal{F}(\mathbf{q}(t),t)=\mathbf{0}$ converges asymptotically to the solution to $\mathcal{F}(\mathbf{q}(t),t)=\mathbf{0}$.

The operator $\mathcal{F}^*$ is called the *dynamic inverse* of $\mathcal{F}$. In contrast to an inverse operator $\mathcal{F}^{-1}$, which allows for $\mathcal{F}(\mathbf{q}(t),t)=\mathbf{0}$ to be solved directly, the dynamic inverse allows for the conversion of a dynamically unstable ODE of the form $\dot{\mathbf{q}}(t)+\mathcal{F}(\mathbf{q}(t))=\mathbf{0}$ into a stable ODE with the same solution. In the context of convex homotopy, if $\mathcal{H}$ is positive definite for all λ then $\mathcal{H}^*=\mathcal{I}$ is a suitable dynamic inverse because it will result in a stable ODE. However, alternative forms of $\mathcal{H}^*$ can give better curve tracing performance, so the general form of the dynamic inverse is assumed for the analysis.

Since convex homotopy continuation involves integration in the decreasing direction of the ODE parameter λ, it is necessary to define the *reverse mode dynamic inverse* which, when applied to $\mathcal{H}$, stabilizes the ODE $\mathcal{H}^*\mathcal{H}(\mathbf{q},\lambda)$ when integrating in the direction of decreasing λ. An appropriate first step in the development of the monolithic homotopy continuation algorithm is to provide formal definitions of forward and reverse mode dynamic inversion in the context of homotopy.

Definition 6.1. *Let $\mathbf{q}_s(\lambda)$ be a regular homotopy defined implicitly by $\mathcal{H}(\mathbf{q},\lambda)=\mathbf{0}$, $\mathcal{H}:\mathbb{R}^N\times\mathbb{R}\to\mathbb{R}^N$. Let $\mathcal{H}^*:\mathbb{R}^N\times\mathbb{R}\to\mathbb{R}^N$ be continuous in λ and Lipschitz continuous on the ball $\mathcal{B}_r=\left\{\Delta\mathbf{q}\in\mathbb{R}^N\middle|\ \|\Delta\mathbf{q}\|\le r\right\}$, $r>0$. Then $\mathcal{H}^*$ is called a **forward dynamic inverse** of $\mathcal{H}$ on $\mathcal{B}_r$ if there exists fixed $\beta\in\mathbb{R}$, $0<\beta<\infty$, such that*

$$\Delta\mathbf{q}^T\mathcal{H}^*\left(\mathcal{H}\left(\mathbf{q}_s(\lambda)+\Delta\mathbf{q},\lambda\right),\lambda\right)\ge\beta\|\Delta\mathbf{q}\|^2 \tag{6.11}$$

*for all $\Delta\mathbf{q}\in\mathcal{B}_r$. Similarly, $\mathcal{H}^*$ is called a **reverse mode dynamic inverse** of $\mathcal{H}$ on $\mathcal{B}_r$ if there exists fixed $\beta\in\mathbb{R}$, $0<\beta<\infty$, such that*

$$\Delta\mathbf{q}^T\mathcal{H}^*\left(\mathcal{H}\left(\mathbf{q}_s(\lambda)+\Delta\mathbf{q},\lambda\right),\lambda\right)\le-\beta\|\Delta\mathbf{q}\|^2 \tag{6.12}$$

for all $\Delta\mathbf{q}\in\mathcal{B}_r$.

Remark 6.1. *If $\mathcal{H}^*$ is a (forward or reverse mode) dynamic inverse of $\mathcal{H}$ with constant β, then for any $\gamma\in\mathbb{R}$, $\gamma>0$, $\gamma\mathcal{H}^*$ is a (forward or reverse mode) dynamic inverse of $\mathcal{H}$ with constant $\gamma\beta$.*

Remark 6.2. *If $\mathcal{H}^*$ is a forward dynamic inverse of $\mathcal{H}$, then $-\mathcal{H}^*$ is a reverse mode dynamic inverse of $\mathcal{H}$.*

The dynamic inverse $\mathcal{H}^*$ of Definition 6.1 is a local state- and parameter-dependent approximation to the inverse of the nonlinear system of equations $\mathcal{H}(\mathbf{q},\lambda)$. Theorem 6.1 to follow presents the predictor portion $\mathcal{E}$, which is a local state- and parameter-dependent approximation to $\dot{\mathbf{q}}_s(\lambda)$, the rate of change of actual curve values $\mathbf{q}_s$ with respect to the parameter λ.

Theorem 6.1. *Let $\mathbf{q}_s(\lambda)$ be a regular homotopy defined implicitly by $\mathcal{H}(\mathbf{q},\lambda)=\mathbf{0}$. Assume that $\mathcal{H}^*:\mathbb{R}^N\times\mathbb{R}\to\mathbb{R}^N;\ (\mathbf{w},\lambda)\mapsto\mathcal{H}^*(\mathbf{w},\lambda)$ is a reverse mode dynamic inverse of $\mathcal{H}(\mathbf{q},\lambda)$ on $\mathcal{B}_r=\left\{\Delta\mathbf{q}\in\mathbb{R}^N\middle|\ \|\Delta\mathbf{q}\|\le r\right\}$, $r>0$, $0<\beta<\infty$. Let $\mathcal{E}:\mathbb{R}^N\times\mathbb{R}\to\mathbb{R}^N;\ (\mathbf{q},\lambda)\mapsto\mathcal{E}(\mathbf{q},\lambda)$ be locally Lipschitz in $\mathbf{q}$ and piecewise continuous in λ. Assume that for some fixed $\omega\in(0,\infty)$, $\mathcal{E}(\mathbf{q},\lambda)$ satisfies*

$$-\frac{1}{2}\omega\|\Delta\mathbf{q}\|^2\le\Delta\mathbf{q}^T\left[\mathcal{E}\left(\mathbf{q}_s+\Delta\mathbf{q},\lambda\right)+\dot{\mathbf{q}}_s(\lambda)\right]\le\frac{1}{2}\omega\|\Delta\mathbf{q}\|^2 \tag{6.13}$$

for all $\Delta\mathbf{q} \in \mathcal{B}_r$. *Let* $\mathbf{q}'_s(\lambda)$ *denote the solution to the system*

$$-\dot{\mathbf{q}} = \gamma \mathcal{H}^* \left(\mathcal{H}(\mathbf{q}, \lambda), \lambda \right) + \mathcal{E}(\mathbf{q}, \lambda), \tag{6.14}$$

where $\gamma \in \mathbb{R}$, $\gamma > 0$ *(see Remark 6.1). Consider now some* $\lambda_k \in \mathbb{R}$ *such that*

$$\mathbf{q}_s(\lambda_k) - \mathbf{q}'_s(\lambda_k) \in \mathcal{B}_r. \tag{6.15}$$

Then

$$\|\mathbf{q}'_s(\lambda) - \mathbf{q}_s(\lambda)\| \leq \|\mathbf{q}'_s(\lambda_k) - \mathbf{q}_s(\lambda_k)\| \, e^{-(\gamma\beta - \omega)|\lambda_k - \lambda|} \tag{6.16}$$

for all $\lambda < \lambda_k$.

Proof.
The proof is similar to that of Getz [48], pp. 25-26. Let $\mathbf{z}(\lambda) = \mathbf{q}'_s(\lambda) - \mathbf{q}_s(\lambda)$, $\mathbf{z} \in \mathbb{R}^N$. Let $V(\mathbf{z}(\lambda)) = \frac{1}{2}\|\mathbf{z}(\lambda)\|^2$, $V : \mathbb{R}^N \to \mathbb{R}$. Then

$$\begin{aligned}
\dot{\mathbf{z}} &= \dot{\mathbf{q}}'_s - \dot{\mathbf{q}}_s \\
&= -\gamma \mathcal{H}^* \left(\mathcal{H}(\mathbf{q}'_s, \lambda) \lambda \right) - \mathcal{E}(\mathbf{q}'_s, \lambda) - \dot{\mathbf{q}}_s \\
&= -\gamma \mathcal{H}^* \left(\mathcal{H}(\mathbf{q}_s + \mathbf{z}, \lambda) \right) - \mathcal{E}(\mathbf{q}_s + \mathbf{z}, \lambda) - \dot{\mathbf{q}}_s.
\end{aligned} \tag{6.17}$$

Differentiating $V(\mathbf{z}(\lambda))$ with respect to $-\lambda$:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}(-\lambda)} V(\mathbf{z}(\lambda)) &= -\mathbf{z}^T \dot{\mathbf{z}} \\
&= \mathbf{z}^T \left[\gamma \mathcal{H}^* \left(\mathcal{H}(\mathbf{q}_s + \mathbf{z}, \lambda) \right) + \mathcal{E}(\mathbf{q}_s + \mathbf{z}, \lambda) + \dot{\mathbf{q}}_s \right] \\
&\leq -\gamma\beta \|\mathbf{z}\|^2 + \omega \|\mathbf{z}\|^2.
\end{aligned} \tag{6.18}$$

So, for $\mathbf{z} \in \mathcal{B}_r$,

$$\frac{\mathrm{d}}{\mathrm{d}(-\lambda)} V(\mathbf{z}(\lambda)) \leq -2(\gamma\beta - \omega) V(\mathbf{z}(\lambda)). \tag{6.19}$$

For an initial condition $-\lambda = -\lambda_k$, then by the Comparison Theorem A.2:

$$V(\mathbf{z}(\lambda)) \leq V(\mathbf{z}(\lambda_k)) \, e^{-2(\gamma\beta - \omega)(\lambda_k - \lambda)}, \;\; \lambda < \lambda_k. \tag{6.20}$$

Taking the square root of both sides of equation (6.20) returns equation (6.16) as required. □

The ODE (6.14) will converge to the homotopy curve asymptotically as long as $\gamma\beta > \omega$ and with an upper bound on the convergence rate as given by equation (6.16). To achieve a high convergence rate, it is desired that $\gamma\beta - \omega$ should be as large as possible. An explicit value for the parameter β is generally unavailable but depends on the definiteness of $\mathcal{H}^*\mathcal{H}(\mathbf{q}, \lambda)$. The parameter ω is a measure of the quality of the predictor term $\mathcal{E}$ and is 0 if $\mathcal{E}$ is exact, which is possible in the continuous case if $\mathcal{E}$ is set analytically to $-\dot{\mathbf{q}}_s$, as can be verified by inspection of equation (6.13). The final parameter γ is an additional degree of freedom introduced in the definition of the dynamic inverse and can be chosen by the user.

Since it is desired for rapid convergence of the continuous ODE (6.14) that $\gamma\beta - \omega$ should be as large as possible, the best choice of γ is to make γ arbitrarily large. However, this is only possible in the

continuous case where it is assumed that $\mathcal{H}^*$ and $\mathcal{E}$ are updated continuously with $\mathbf{q}$ and λ. For the discrete case, it is not possible to trace the curve accurately with arbitrarily large γ unless $|\Delta\lambda|$ takes on arbitrarily small values. Analysis for the discrete case and the relationship between γ and $|\Delta\lambda|$ is covered in Section 6.7.

6.4 Construction of $\mathcal{E}$

The quantity $\dot{\mathbf{q}}_s(\lambda)$ is the tangent vector with λ-parametrization and can easily be derived by directly differentiating $\mathcal{H}(\mathbf{q}_s(\lambda), \lambda) = \mathbf{0}$ and rearranging:

$$\dot{\mathbf{q}}_s(\lambda) = -\left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s(\lambda), \lambda)\right]^{-1} \frac{\partial}{\partial\lambda}\mathcal{H}(\mathbf{q}_s(\lambda), \lambda). \tag{6.21}$$

A suitable choice for the operator $\mathcal{E}$ can be obtained by setting $\mathcal{E}$ to $-\dot{\mathbf{q}}_s$, as per equation (6.13):

$$\mathcal{E}(\mathbf{q}, \lambda) = \left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}'_s(\lambda), \lambda)\right]^{-1} \frac{\partial}{\partial\lambda}\mathcal{H}(\mathbf{q}'_s(\lambda), \lambda). \tag{6.22}$$

For the special case of convex homotopy, the specific form of this equation is

$$\mathcal{E}(\mathbf{q}, \lambda) = \left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}'_s(\lambda), \lambda)\right]^{-1} \left[\mathcal{G}(\mathbf{q}'_s) - \mathcal{R}(\mathbf{q}'_s)\right]. \tag{6.23}$$

6.5 Construction of $\mathcal{H}^*$

As mentioned previously, a stable ODE can be constructed for the homotopies studied in this thesis by taking the reverse mode dynamic inverse $\mathcal{H}^*$ as the identity operator. However, the ODE that would result from equation (6.14) would resemble an explicit time-marching method and a high convergence rate would not be expected.

A more efficient alternative might be to use the inverse Jacobian evaluated near the curve, which is a forward dynamic inverse [49]. The most practical location to evaluate the inverse Jacobian, of course, is at the current point generated by the algorithm. It is shown in Section 6.7 that this results in a Newton-like update. While using an inverse Jacobian as the dynamic inverse requires for a linear system of equations to be solved, this contributes no additional cost when constructing $\mathcal{E}$ according to equation (6.22) because this term also contains an inverse Jacobian and the two linear solves can be combined into one in the MH update expression.

We now show, following the approach of Getz and Marsden [49], that the inverse Jacobian evaluated near the curve is a dynamic inverse of $\mathcal{H}$. Assume that the continuation algorithm has arrived at some point $\mathbf{q}_s + \Delta\mathbf{q}$, $\Delta\mathbf{q} \in \mathcal{B}_r \subset \mathbb{R}^N$, $r > 0$. The residual evaluated at this point $\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)$ can be represented by taking the Taylor expansion at $\mathbf{q}_s$:

$$\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) = \mathcal{H}(\mathbf{q}_s, \lambda) + \nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s, \lambda)\Delta\mathbf{q} + \mathcal{O}\left(\|\Delta\mathbf{q}\|^2\right). \tag{6.24}$$

Similarly expanding $\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda)$ at $\mathbf{q}_s$ gives

$$\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda) = \nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) + \mathcal{O}(\|\Delta\mathbf{q}\|). \tag{6.25}$$

Using equation (6.25) and $\mathcal{H}(\mathbf{q}_s, \lambda) = \mathbf{0}$, equation (6.24) becomes

$$\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) = \nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)\,\Delta\mathbf{q} + \mathcal{O}\left(\|\Delta\mathbf{q}\|^2\right). \tag{6.26}$$

Let $\mathcal{H}^* = [\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)]^{-1}$ be a candidate for a forward dynamic inverse of $\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)$. If it is a forward dynamic inverse, it must satisfy Definition 6.1. This is verified by applying $\Delta\mathbf{q}^T\mathcal{H}^*$ to both sides of equation (6.26):

$$\begin{aligned}\Delta\mathbf{q}^T\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) &= \Delta\mathbf{q}^T\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)\,\Delta\mathbf{q} + \Delta\mathbf{q}^T\mathcal{H}^*\mathcal{O}\left(\left\|\Delta\mathbf{q}^2\right\|\right) \\ &= \|\Delta\mathbf{q}\|^2 + \mathcal{O}\left(\|\Delta\mathbf{q}\|^3\right).\end{aligned} \tag{6.27}$$

The contribution of the $\mathcal{O}\left(\|\Delta\mathbf{q}\|^3\right)$ terms can be positive or negative. In either case, for sufficiently small $r > 0$, there exists fixed $0 < \beta < 1$ such that these combined terms are upper-bounded by $(1-\beta)\|\Delta\mathbf{q}\|^2$. Thus, for sufficiently small $r > 0$, there exists fixed $0 < \beta < 1$ such that

$$\Delta\mathbf{q}^T\mathcal{H}^*\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) \geq \|\Delta\mathbf{q}\|^2 - (1-\beta)\|\Delta\mathbf{q}\|^2 = \beta\|\Delta\mathbf{q}\|^2 \tag{6.28}$$

for all $\Delta\mathbf{q} \in \mathcal{B}_r$. Hence, the inverse Jacobian evaluated at $\mathbf{q}_s + \Delta\mathbf{q}$ is a forward dynamic inverse. Since a reverse-mode dynamic inverse is what is needed in the context of homotopy, then, by Remark 6.2, a suitable expression for $\mathcal{H}^*$ is

$$\mathcal{H}^* = -\left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda)\right]^{-1}. \tag{6.29}$$

6.6 Numerical Implementation

Substituting equations (6.29) and (6.23) into (6.14) gives the following ODE:

$$-\dot{\mathbf{q}}(\lambda) = \left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda)\right]^{-1}\left[-\gamma\mathcal{H}(\mathbf{q}, \lambda) + \mathcal{G}(\mathbf{q}) - \mathcal{R}(\mathbf{q})\right], \tag{6.30}$$

which can then be integrated numerically to obtain an update formula for the state. For example, using an Euler integration scheme:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + (\lambda_{k+1} - \lambda_k)\left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_k, \lambda_k)\right]^{-1}\left[\gamma_k\mathcal{H}(\mathbf{q}_k, \lambda_k) - \mathcal{G}(\mathbf{q}_k) + \mathcal{R}(\mathbf{q}_k)\right], \tag{6.31}$$

where selection of $\Delta\lambda_k$ and γ are discussed in Section 6.7.

Ideally, the monolithic homotopy algorithm is initiated at $\lambda_0 = 1$ with an exact solution for $\mathcal{G}(\mathbf{q}) = \mathbf{0}$, thus ensuring that condition (6.15) is met for $\lambda = 1$. Taking small enough $|\Delta\lambda|$ should ensure that $-\left[\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda)\right]^{-1}$ evaluated at the current value of $\mathbf{q}$ remains a dynamic inverse of $\mathcal{H}(\mathbf{q}, \lambda)$ and so there should always exist small enough $|\Delta\lambda|$ such that the MH algorithm is stable.

Once traversing is complete and λ is reduced to 0, the inexact Newton phase is initiated to solve the system $\mathcal{R}(\mathbf{q}) = \mathbf{0}$. A pseudo-code for the MH algorithm is provided as Algorithm 6.1.

Algorithm 6.1: Monolithic homotopy (MH) continuation

Initialize: Set $\lambda = 1$ and solve $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ if necessary
while $\lambda > 0$ **do**
  Get γ, $\mathcal{H}$, $\|\mathcal{H}\|$, and $-\gamma\mathcal{H} + \mathcal{G} - \mathcal{R}$
  Form and factor the preconditioner approximating the matrix $\nabla\mathcal{H}$
  Solve the linear system $\nabla\mathcal{H}^{-1}\left[-\gamma\mathcal{H} + \mathcal{G} - \mathcal{R}\right]$
  Determine $\Delta\lambda$ from step-length adaptation (Algorithm 6.2)
  Update λ and $\mathbf{q}$
end
Inexact Newton Phase: Solve $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ to some tolerance using the inexact Newton method

6.7 Predictor-Corrector Analogue and Selection of γ_k and $\Delta\lambda_k$

Recall equation (4.17), which is an analytical expression for the tangent vector $t \in \mathbb{R}^{N+1}$. For notational convenience, introduce $t_{\mathbf{q}} \in \mathbb{R}^N$, $t_\lambda \in \mathbb{R}$, such the $t = [t_{\mathbf{q}};\, t_\lambda]$. Since an Euler update for the PC method is given by

$$\begin{pmatrix} \mathbf{q} \\ \lambda \end{pmatrix}_{k+1}^{(0)} = \begin{pmatrix} \mathbf{q} \\ \lambda \end{pmatrix}_{k}^{(p_k)} + h_k \begin{pmatrix} t_{\mathbf{q}} \\ t_\lambda \end{pmatrix}_k, \tag{6.32}$$

where $h_k \in \mathbb{R}$ is the step-length, then h_k and $\Delta\lambda_k$ are correlated by

$$\Delta\lambda_k = -h_k / \|\tau_k\|. \tag{6.33}$$

Equation (6.33) can be used with equation (4.17) to develop an equation for $\Delta\lambda_k \mathbf{z}_k$:

$$\Delta\lambda_k \mathbf{z}_k = \Delta\lambda_k \|\tau_k\| t_{\mathbf{q}} = -h_k t_{\mathbf{q}}, \tag{6.34}$$

which can then be used to rewrite equation (6.31) as a combination of more familiar quantities:

$$\begin{aligned} \Delta\mathbf{q}_k &= \gamma_k \Delta\lambda_k \left[\nabla\mathcal{H}(\mathbf{q}_k, \lambda_k)\right]^{-1} \mathcal{H}(\mathbf{q}_k, \lambda_k) - \Delta\lambda_k \mathbf{z}_k \\ &= \gamma_k \Delta\lambda_k \left[\nabla\mathcal{H}(\mathbf{q}_k, \lambda_k)\right]^{-1} \mathcal{H}(\mathbf{q}_k, \lambda_k) + h_k t_{\mathbf{q}}. \end{aligned} \tag{6.35}$$

Numerically, the monolithic homotopy update formula is applied in the form of equation (6.31). However, writing this equation in the form of equation (6.35) reveals that each update iteration with $\gamma_k = 1/|\Delta\lambda_k|$ is analogous to a combination of a Newton iteration and a predictor step, both evaluated at the same state. When $|\Delta\lambda_k|$ is small, the corrector portion of the update will be emphasized and the algorithm will be more robust, but more steps will be required for traversing.

Since it may be desirable to adjust $\Delta\lambda_k$ after the linear solve, the following formula can be used for γ_k:

$$\gamma_k = \frac{1}{|\Delta\lambda_k|^*}, \tag{6.36}$$

where $|\Delta\lambda_k|^*$ is an approximation to $|\Delta\lambda_k|$. A suitable choice might be $|\Delta\lambda_k|^* = |\Delta\lambda_{k-1}|$, $k \geq 1$. This is the approach adopted in this thesis.

6.8 Step-length Adaptation

Though the MH method allows for more control over traversing, less information is available to do so, especially since it is not possible to distinguish between the predictor and corrector portion of the update. The distance to the curve is not available, and any change in orientation of the corrector portion of the update is not particularly meaningful, so that the angle between consecutive updates is not a useful metric. Furthermore, the residual is not a reliable metric, as will be made apparent in Section 6.9.3.

The only useful information comes from the update itself. If the norm of the state update $\|\Delta\mathbf{q}_k\|$ is large, then $|\Delta\lambda_k|$ should be reduced in order to attempt to maintain a consistent Δs. The proposed update formula is based on keeping $\|\Delta\mathbf{q}_k\|$ constant, calibrated from the first iteration. Thus, the user chooses an initial value of $|\Delta\lambda|$ to calibrate the step-length adaptation rather than choosing a target value of $\|\Delta\mathbf{q}\|$.

Denote the target value of $\|\Delta\mathbf{q}\|$ by $\|\Delta\mathbf{q}\|_{\text{tar}}$. This quantity can be set according to

$$\|\Delta\mathbf{q}\|_{\text{tar}} = \|\dot{\mathbf{q}}_0\| \, |\Delta\lambda_0| \,, \tag{6.37}$$

which can be used to calculate the numerical integration step

$$\Delta\lambda_k = -\frac{\|\Delta\mathbf{q}\|_{\text{tar}}}{\|\dot{\mathbf{q}}\|}. \tag{6.38}$$

The step size $|\Delta\lambda|$ is also restricted by upper and lower bounds:

$$\min\left\{|\Delta\lambda|_{\min}, f_{\min}\,|\Delta\lambda_{k-1}|\right\} \leq |\Delta\lambda_k| \leq \max\left\{|\Delta\lambda|_{\max}, f_{\max}\,|\Delta\lambda_{k-1}|\right\}, \tag{6.39}$$

where $|\Delta\lambda|_{\min}, f_{\min}, |\Delta\lambda|_{\max}, f_{\max} \in \mathbb{R}$ can be user-defined. For the current study, $f_{\min} = 1/3$ and $f_{\max} = 2$; $|\Delta\lambda|_{\min}$ and $|\Delta\lambda|_{\max}$ are adjusted according to the initial step size $|\Delta\lambda_0|$ but are generally not chosen to be overly restrictive. In addition, a maximum step size can be imposed at the final step to attempt to improve the quality of the initial guess for the inexact Newton phase at $\lambda = 0$. A detailed pseudo-code of the step-length adaptation is provided in Algorithm 6.2, including many different checks that are applied to $\Delta\lambda$.

6.9 Performance Investigation for Inviscid Flows

It is important to investigate how performance of the MH algorithm depends on several important flow solver parameters. This is especially true of the linear solver tolerances and the type of matrix vector products used, since these features can significantly impact CPU time and also the accuracy of the update. For all numerical studies in this chapter, the dissipation operator with far-field boundary conditions, as described in Section 4.4.2, is used as the homotopy system.

The inviscid test cases were performed on an ONERA M6 wing using grid Me1, which has an H-C topology with about 1.88 million nodes. Each parameter test involved a complete flow solve on 40 different operating conditions generated from every combination of Mach numbers between 0.2 and 0.9 varied in intervals of 0.1 with angle of attack varying between 0° and 12° varied in intervals of 3°. The solver parameters pertaining to the inexact Newton phase were kept consistent for all parameter studies.

Algorithm 6.2: Step-length adaptation for the MH continuation algorithm

Data: $|\Delta\lambda|_{\min}$, $f_{\min}$, $|\Delta\lambda|_{\max}$, $f_{\max}$, $\Delta\lambda_{\text{prev}}$, $|\Delta\lambda|_{\max,\text{final}}$, k
Result: $\Delta\lambda$
if $k = 0$ **then**
  Initialize $\Delta\lambda$ from the input value
  $\|\Delta\mathbf{q}\|_{\text{tar}} \leftarrow \|\dot{\mathbf{q}}\|\,|\Delta\lambda|$
else
  $\Delta\lambda \leftarrow -\|\dot{\mathbf{q}}\| \,/\, \|\Delta\mathbf{q}\|_{\text{tar}}$
  $f \leftarrow |\Delta\lambda| \,/\, |\Delta\lambda_{\text{prev}}|$
  $f \leftarrow \max\{\min\{f, f_{\max}\}, f_{\min}\}$
  $\Delta\lambda \leftarrow f\Delta\lambda_{\text{prev}}$
end
/* Check $|\Delta\lambda|$ against $|\Delta\lambda|_{\min}$ and $|\Delta\lambda|_{\max}$ */
if $|\Delta\lambda| < |\Delta\lambda|_{\min}$ *and* $\lambda > 0$ **then**
  $\Delta\lambda \leftarrow \Delta\lambda_{\min}$
else if $|\Delta\lambda| > |\Delta\lambda|_{\max}$ **then**
  $\Delta\lambda \leftarrow \Delta\lambda_{\max}$
end
/* Additional checks near $\lambda = 0$ */
$\lambda^* \leftarrow \lambda + \Delta\lambda$
$\lambda_{\min} \leftarrow 0.25\lambda$
if $\lambda^* < 0$ **then**
  /* Adjust $\Delta\lambda$ so that the Euler update will give exactly $\lambda = 0$ */
  $\Delta\lambda \leftarrow -\lambda$
  if $\lambda^* > |\Delta\lambda|_{\max,\text{final}}$ **then**
    /* Adjust $\Delta\lambda$ if it is the final step and exceeds $|\Delta\lambda|_{\max,\text{final}}$ */
    $\lambda_{\min} \leftarrow \min\left\{\lambda_{\min}, |\Delta\lambda|_{\max,\text{final}}\right\}$
    $\Delta\lambda \leftarrow \lambda_{\min} - \lambda$
  end
else if $\lambda^* < \lambda_{\min}$ **then**
  /* This logic exists to even out the final steps */
  $\Delta\lambda \leftarrow \lambda_{\min} - \lambda$
end

The relative linear solver tolerance for the inexact Newton phase was $\tau_l = 0.01$, and a maximum of 80 Krylov iterations were permitted in FGMRES.

Performance of the CHC-MH algorithm is benchmarked against the CHC-PC algorithm under suitable conditions: the predictor used an algebraic tangent calculation, all linear systems were solved to a relative tolerance of $\tau_l = 0.01$, AMVPs were used in the globalization phase with the preconditioner matrix being updated every 3 iterations, and the nonlinear subproblems were solved to a relative tolerance of 0.5.

6.9.1 Step Size when using FDMVPs

The first tests were conducted with $\Delta\lambda_k = \Delta\lambda_{k-1}$ to test how large the step size can be taken without loss of robustness. Tests were carried out at $|\Delta\lambda| \in \{0.02, 0.05, 0.1, 0.15, 0.2\}$, though the final step size was restricted to an upper bound of $|\Delta\lambda| \leq 0.05$ in all cases, which in some cases required some reduction in step size at the final iterate. The relative tolerance used in the linear solver was $\tau_l = 0.01$.

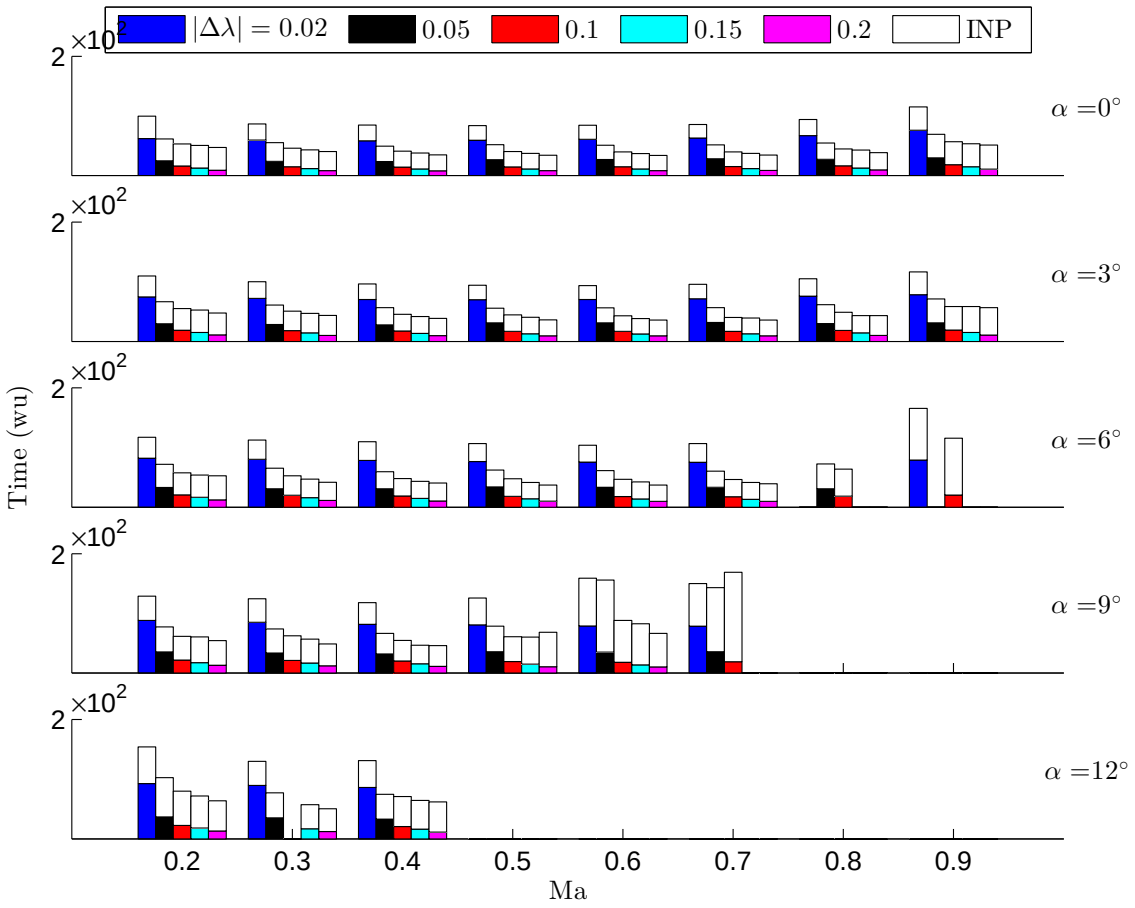


Figure 6.1: Performance of the CHC-MH algorithm with $\tau_l = 0.01$, finite-difference matrix-vector products, and constant $|\Delta\lambda|$.

The performance is summarized in Figure 6.1 and Table 6.1. From the figure and table, it would appear that an average step size of between 0.1 and 0.2 provide a reasonable compromise between robustness and CPU time for these inviscid cases when FDMVPs are used.

6.9.2 Step Size when using AMVPs

The same test was carried out using approximate matrix-vector products. This is an important test because the use of approximate matrix-vector considerably reduces linear solver time but also fundamentally changes the linear system being solved. The effect is that condition (6.13) is no longer satisfied exactly, so stability of the algorithm can no longer be assured.

To assess the viability of AMVPs in the globalization phase, cases were run for the same test suite using AMVPs in the globalization phase for $|\Delta\lambda| \in \{0.05, 0.1, 0.15, 0.2\}$. The results are summarized in Figure 6.2 and Table 6.1. Though the success rate reported in Table 6.1 gives the impression that robustness has actually improved slightly with increasing $|\Delta\lambda|$, it can be observed from Figure 6.2 the cost of the inexact Newton phase has increased for some of the more difficult flow solves with larger $|\Delta\lambda|$, which is due to inferior globalization in these cases. Based on these observations, and the perfor-

	FDMVPs		AMVPs	
$\|\Delta\lambda\|$	Success Rate	Rel. CPU Time	Success Rate	Rel. CPU Time
0.02	32/40	2.08	-	-
0.05	32/40	1.27	32/40	1.07
0.1	32/40	1.04	32/40	0.85
0.15	30/40	0.93	33/40	0.82
0.2	30/40	0.86	33/40	0.81

Table 6.1: Effects of $|\Delta\lambda|$ and matrix-vector products on the performance of the CHC-MH method. The relative CPU time is the CPU time taken by the CHC-MH method relative to the CPU time taken by the benchmark CHC-PC method, which had a success rate of 32/40.

mance data collected in Table 6.1, our assessment is that $|\Delta\lambda|$ values between 0.1 and 0.2 provide a good balance between robustness and CPU time for these inviscid cases. We also observe that using AMVPs in place of FDMVPs seems to reduce the CPU time without loss of robustness.

6.9.3 Linear Solver Tolerance

Some unexpected behaviour was observed regarding the relationship between the step size $|\Delta\lambda|$ and the homotopy residual $\|\mathcal{H}(\mathbf{q},\lambda)\|$. As $|\Delta\lambda|$ is reduced, it is expected that the curve should be traced more accurately and that the residual $\|\mathcal{H}(\mathbf{q},\lambda)\|$ evaluated along the trajectory should become smaller. However, it was consistently observed that $\|\mathcal{H}(\mathbf{q},\lambda)\|$ would grow to larger values during traversing for cases where smaller step sizes were taken.

To study this behaviour and how it relates to the actual tracking error, the C_L values evaluated at points along the trajectory generated by the MH algorithm were compared to the 'Exact' C_L values along the homotopy curve which were estimated using the PC algorithm. It is observed from Figure 6.3 that for smaller $|\Delta\lambda|$, the tracking error is consistently lower, but the homotopy residual norm $\|\mathcal{H}(\mathbf{q},\lambda)\|$ is generally higher. This behaviour was found to be linked to the linear solver tolerance. Figure 6.3b shows the same study performed using a linear tolerance of $\tau_l = 0.001$, in which $\|\mathcal{H}\|$ and the error in C_L are both consistently lower for smaller $|\Delta\lambda|$.

Though there does not appear to be any performance degradation directly linked to this unusual behaviour for this case, it is important to understand what has caused this behaviour. We conjecture that the unexpected trend that the tracking residual $\|\mathcal{H}\|$ increases as $|\Delta\lambda|$ decreases for sufficiently large τ_l is due to error propagation in the MH update. When a nonlinear update is performed, the error vector resulting from under-solving the linear system can be modeled as a random disturbance $\mathbf{e} \in \mathbb{R}^N$. For a generic Euler update:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + h\mathbf{d} + h\mathbf{e}, \tag{6.40}$$

where $\mathbf{d} \in \mathbb{R}^N$ is the exact update direction, and $\mathbf{d} + \mathbf{e}$ is the approximation to $\mathbf{d}$ resulting from under-solving the linear system of equations.

For the special case of the inexact Newton method, this error is incorporated into the updated value of $\mathbf{q}$, where it adds to the error in the nonlinear problem and is reduced in successive Newton iterations. As Newton's method converges and the step sizes become smaller, the error terms become smaller and the effect is reduced.

In the case of the MH method, the error vector at each iteration is not resolved using Newton's method. Though the MH update does contain a corrector component, the correction is always applied

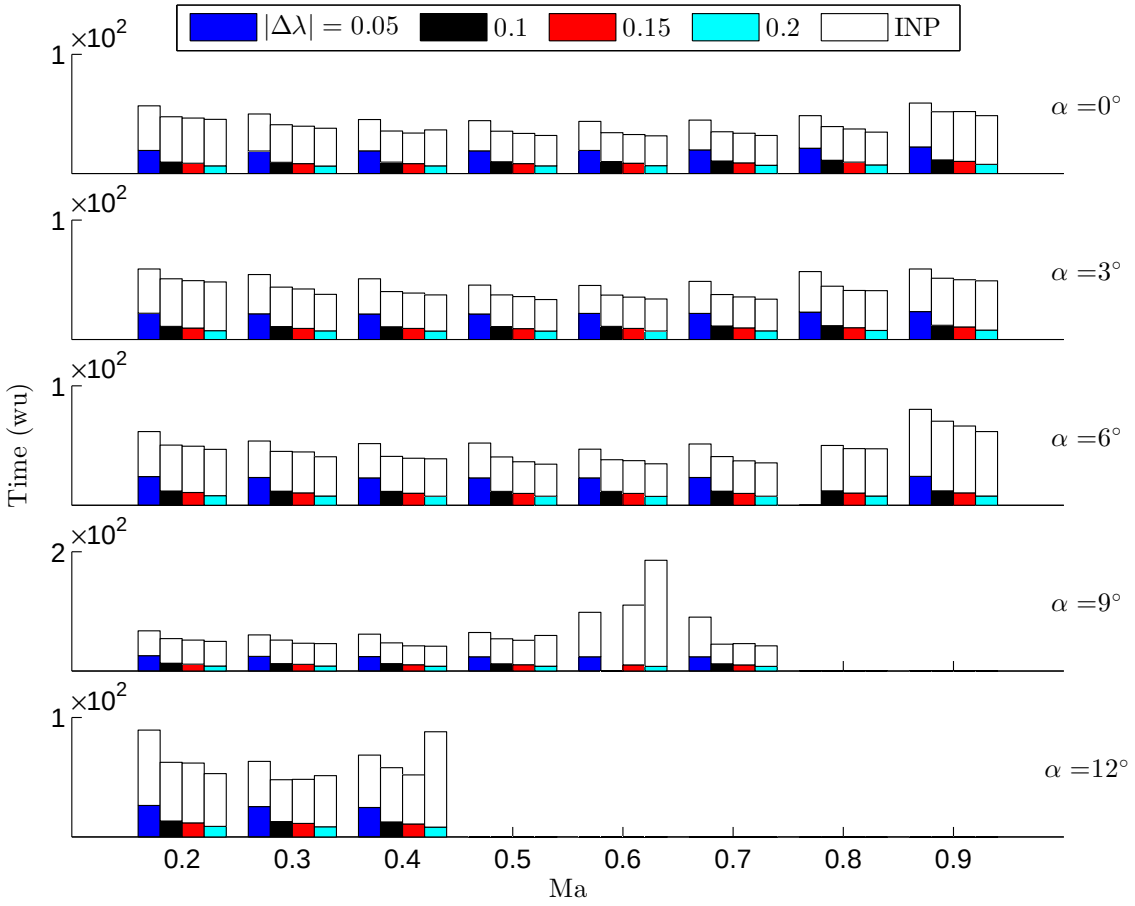


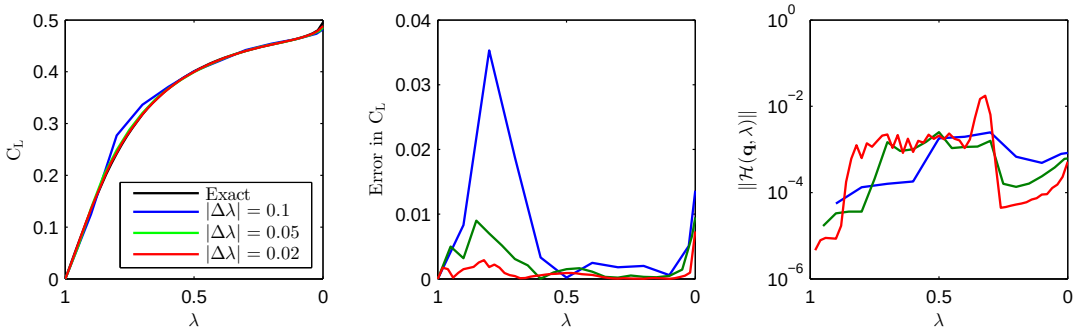
Figure 6.2: Performance of the CHC-MH algorithm with $\tau_l = 0.01$, approximate matrix-vector products, and constant $|\Delta\lambda|$.

to the following iterative step and does not fully resolve the error. The error vector from each iterate superimposes on the error vectors introduced at successive iterations.

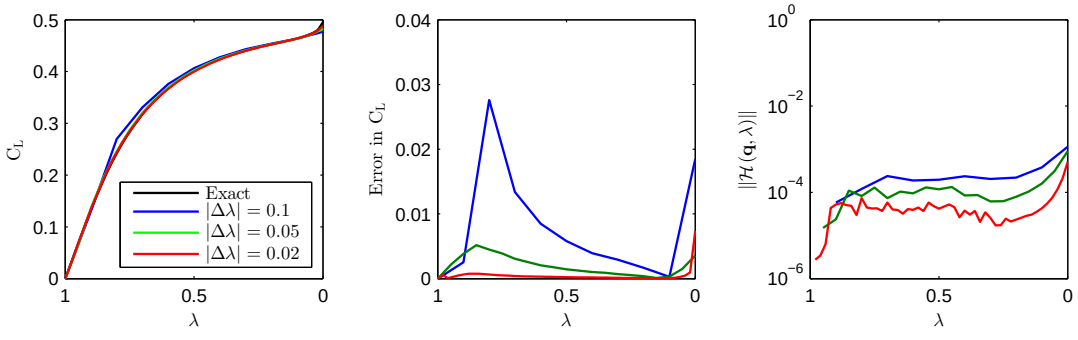
If the step size is smaller, then the magnitude of the error contributed by each iteration will be less, but the accumulation of a large number of unresolved error terms will appear in continuous space as high-frequency random error. The actual curve estimate at a given λ is equal to this error, along with the non-random curve-tracing error, superimposed on the exact solution. If $\Delta\mathbf{q}_e$ is the curve-tracing error occurring naturally from the algorithm and $\mathbf{q}_s$ is the exact solution to $\mathcal{H}(\mathbf{q}, \lambda) = \mathbf{0}$ at some λ, then $\|\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}_e + \mathbf{e}, \lambda)\|$ can be considerably larger than $\|\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}_e, \lambda)\|$ even if $\|\Delta\mathbf{q}_e\| \gg \|\mathbf{e}\|$ as a result of evaluating discrete derivatives using highly non-smooth data.

6.9.4 MH Algorithm with Step-length Adaptation

The step-length adaptation algorithm employed tends to result in average step sizes that are smaller than the initial step size. Since the constant step size of $|\Delta\lambda| = 0.1$ was previously found to be acceptable when step-length adaptation is inactive, the initial step sizes investigated with step-length adaptation are $|\Delta\lambda_0| \in \{0.1, 0.15, 0.2, 0.25, 0.3\}$. Approximate matrix-vector products are used in the linear solver with $\tau_l = 0.01$. A sample case for several values of $|\Delta\lambda_0|$ is shown in Figure 6.4.



(a) $\tau_l = 0.01$



(b) $\tau_l = 0.001$

Figure 6.3: Trajectory of the MH algorithm at Mach number 0.5 and angle of attack 6° with FDMVPs

$\lvert\Delta\lambda_0\rvert$	0.1	0.15	0.2	0.25	0.3
Success Rate	33/40	32/40	33/40	33/40	32/40
Rel. CPU Time	1.01	0.91	0.86	0.83	0.83

Table 6.2: Effects of $|\Delta\lambda_0|$ on the performance of the CHC-MH method with step-length adaptation. The linear solver uses AMVPs with $\tau_l = 0.01$. The relative CPU time is the CPU time taken by the CHC-MH method relative to the CPU time taken by the benchmark CHC-PC method, which had a success rate of 32/40.

The performance data is summarized in Table 6.2 and Figure 6.5. The step-length adaptation only moderately improves the performance for these inviscid cases, which may be expected since the curvature for these cases is moderate compared to the turbulent cases.

6.10 Performance Investigation for Turbulent Flows

In the previous section, a study was presented illustrating that the unintuitive behaviour that the homotopy residual can grow as the step size $|\Delta\lambda|$ is reduced can be related to the accuracy to which the linear system is solved. However, this did not result in major performance degradation for the inviscid cases.

This issue was found to be more problematic for turbulent flows, much more so on the finest mesh MtHC than on the two small turbulent meshes Nt and MtHH. This problem is investigated for the

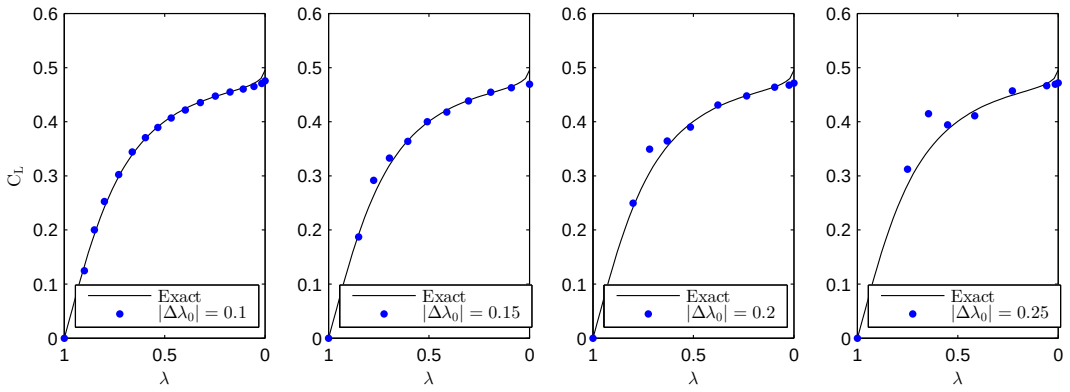


Figure 6.4: Tracking error history for the MH algorithm when applied to the inviscid ONERA M6 wing at Mach number 0.5 and angle of attack 6° with AMVPs, $\tau_l = 0.01$, and active step-length adaptation

Case	Linear solver tolerance	Matrix-vector product	Scaling
F2	10^{-2}	FDMVPs	Geometric
F3	10^{-3}	FDMVPs	Geometric
A2	10^{-2}	AMVPs	Geometric
A3	10^{-3}	AMVPs	Geometric
RC-F2	10^{-2}	FDMVPs	Row/column normalization

Table 6.3: List of studies performed with the MH method with constant $|\Delta\lambda|$ when solving the RANS-SA equations on grid MtHC at Mach 0.8, angle of attack 3°, and Reynolds number 1×10^7

case of three-dimensional transonic flow over the ONERA M6 wing at Reynolds number 1×10^7, Mach number 0.8, and angle of attack 3°. Grid MtHC is used, which consists of 3.68×10^7 nodes divided into 1024 blocks with a minimum off-wall distance of 8.00×10^{-7} mean chord units. The step size $|\Delta\lambda|$ was kept constant at a very small value of 0.005 in order to attempt to isolate the effects of certain solver parameters in the continuation phase. The parameters studied were the use of AMVPs versus FDMVPs, the tolerance used for the linear solver, and different scalings for the linear system. The combinations investigated are presented in Table 6.3.

The effects of the solver parameters on the tracking error are shown in Figure 6.6. As with the inviscid case, there is a clear correlation between the linear solver tolerance and $\|\mathcal{H}\|$. However, there was an additional effect observed here, that the tracking error became very large in the (approximate) region $0.6 < \lambda < 0.9$ regardless of whether the FDMVPs or AMVPs were used. This effect was greatly reduced when the linear solver tolerance was reduced to $\tau_l = 0.001$. This is an important observation, since such large deviation from the curve can lead to non-convergence of the algorithm.

Some important performance differences were observed when forming the matrix-vector products with the AMVPs or the FDMVPs. Though a similar error trend was observed throughout most of the traversing process between Cases F2 and A2, and similarly between Cases F3 and A3, there is significant discrepancy near $\lambda = 0$ where the FDMVPs begin to predict the C_L values much better. This can be attributed to the growth in the approximation error resulting from the use of the approximate Jacobian as the contribution from the flow residual becomes increasingly dominant. This observation is important because it affects the quality of the starting point for Newton's method and can greatly affect the success

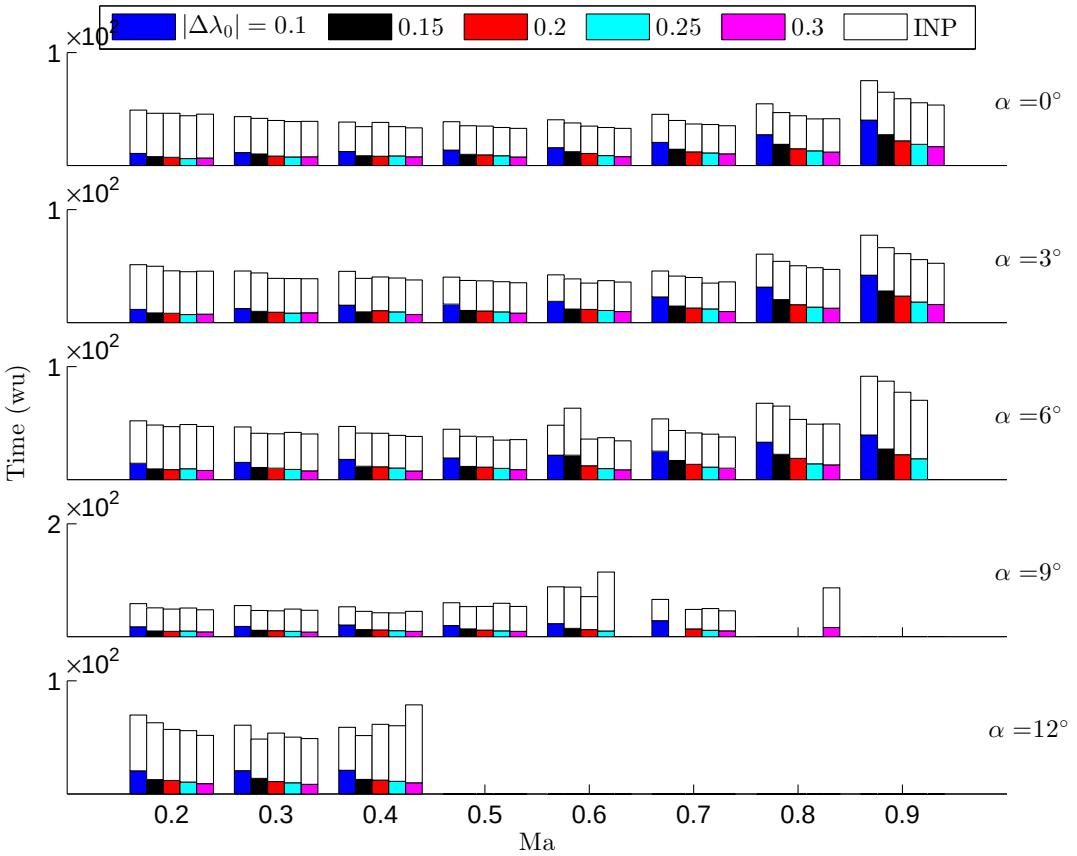


Figure 6.5: Performance of the MH algorithm with $\tau_l = 0.01$, AMVPs, and step-length adaptation; the value of $|\Delta\lambda_0|$ is investigated

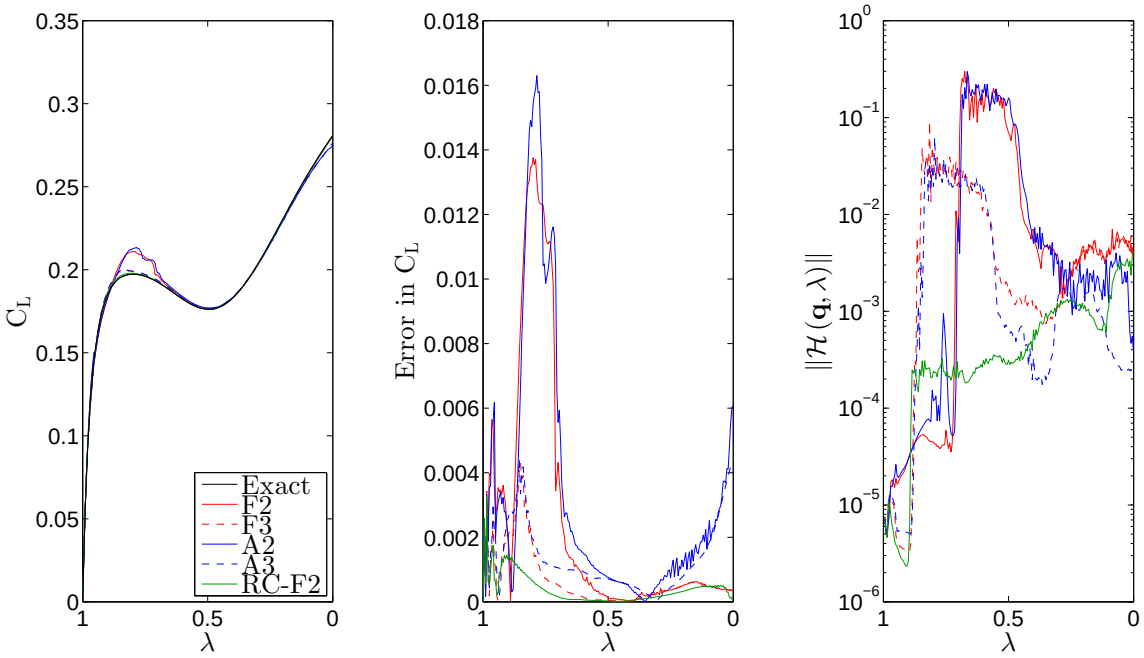


Figure 6.6: Tracking error history for the MH algorithm with constant $|\Delta\lambda|$ when applied to the 1024 block turbulent case at Mach 0.8, angle of attack 3°, and Reynolds number 1×10^7

rate in the inexact Newton phase. By comparison, the accuracy of the matrix-vector products is less important for PTC. The observations made here suggest that it may be advantageous to use FDMVPs during the globalization phase of turbulent cases when using CHC methods.

6.11 Stability Considerations

Stability of the new monolithic homotopy continuation algorithm was proved analytically in Section 6.3 under certain assumptions and conditions, including condition (6.13). Though it is clear that this condition is satisfied by setting $\mathcal{E}$ according to equation (6.23), it is not immediately clear if stability can be maintained if this requirement is relaxed. The exact equality of equation (6.23) is violated by numerical artifacts such as the error introduced by solving the linear system inexactly or with inexact matrix-vector products.

From the numerical tests which have been presented, it was found that the error introduced by solving the linear system inexactly can destabilize the algorithm. This is not apparent for the inviscid cases investigated or for cases on the two smaller turbulent meshes Nt or MtHH, only for the dense turbulent mesh MtHC. This is attributed to the additional steps required for traversing, and exacerbated by high curvature, resulting in more error propagation and inducing instabilities. It was found that using a more conservative linear tolerance or using the row/column normalization scaling instead of the usual geometric scaling improved the stability of the algorithm. These are very important practical considerations.

Chapter 7

Matrix-Free Monolithic Homotopy Continuation Algorithms

In Chapter 6, the general form of a monolithic homotopy continuation algorithm was presented and special forms of the operators $\mathcal{E}$ and $\mathcal{H}^*$ were presented involving $\nabla\mathcal{H}^{-1}$. This chapter presents several formulations of the monolithic algorithm which do not require any matrix inversion. These matrix-free monolithic homotopy (MFMH) continuation algorithms were developed so that homotopy curves generated for any stable homotopy function can be studied without building the Jacobian matrix. However, of more immediate interest, analysis of the matrix-free algorithms has provided additional insight into the monolithic homotopy continuation algorithms in general.

7.1 Matrix-Free Dynamic Inverse

Let $\mathcal{H} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$, $(\mathbf{q}, \lambda) \mapsto \mathcal{H}(\mathbf{q}, \lambda)$ be a regular homotopy which is Lipschitz continuous for all $0 \leq \lambda \leq 1$ and let $\mathcal{H}^*$ be a dynamic inverse of $\mathcal{H}(\mathbf{q}, \lambda)$. Applying $\Delta\mathbf{q}^T\mathcal{H}^*$ to both sides of equation (6.26) gives

$$\Delta\mathbf{q}^T\mathcal{H}^*\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) = \Delta\mathbf{q}^T\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)\Delta\mathbf{q} + \mathcal{O}\left(\|\Delta\mathbf{q}\|^3\right). \tag{7.1}$$

If $\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)$ is positive-definite, then

$$\Delta\mathbf{q}^T\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)\Delta\mathbf{q} > 0, \tag{7.2}$$

so there exists $0 < \beta < 1$ such that

$$\Delta\mathbf{q}^T\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda)\Delta\mathbf{q} \geq 2\beta\|\Delta\mathbf{q}\|^2. \tag{7.3}$$

For sufficiently small $r > 0$, there exists fixed $0 < \beta < 1$ such that the $\mathcal{O}\left(\|\Delta\mathbf{q}\|^3\right)$ terms are upper-bounded by $\beta\|\Delta\mathbf{q}\|^2$ for all $\Delta\mathbf{q} \in \mathcal{B}_r$. Thus, considering equations (7.1) and (7.3), as long as $\mathcal{H}^*\nabla_{\mathbf{q}}\mathcal{H}(\mathbf{q}, \lambda)$ is positive-definite, then for sufficiently small $r > 0$ there exists fixed $0 < \beta < 1$ such that

$$\Delta\mathbf{q}^T\mathcal{H}^*\mathcal{H}(\mathbf{q}_s + \Delta\mathbf{q}, \lambda) \geq 2\beta\|\Delta\mathbf{q}\|^2 - \beta\|\Delta\mathbf{q}\|^2 = \beta\|\Delta\mathbf{q}\|^2 \tag{7.4}$$

for all $\Delta\mathbf{q} \in \mathcal{B}_r$.

Since a real-valued matrix times its transpose is positive-definite, $\left[\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q},\lambda\right)\right]^T$ is a dynamic inverse of $\mathcal{H}\left(\mathbf{q},\lambda\right)$, regardless of whether or not $\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q},\lambda\right)$ is positive-definite. This formulation would avoid solving any linear systems of equations and could be useful for solving systems of equations whose Jacobian is indefinite. However, the method would not be matrix-free and is not studied in this thesis.

Consider as a candidate for $\mathcal{H}^*$ a diagonal positive-definite matrix $\mathcal{T}$. If the entries of $\mathcal{T}$ are all identical then clearly $\mathcal{T}\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q},\lambda\right)$ is positive-definite and so $\mathcal{T}$ is a dynamic inverse of $\mathcal{H}\left(\mathbf{q},\lambda\right)$ if and only if $\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q},\lambda\right)$ is positive-definite.

Remark 7.1. *By Remark 6.2, if $\mathcal{T}$ is a forward dynamic inverse of $\mathcal{H}$ then $-\mathcal{T}$ is a reverse-mode dynamic inverse of $\mathcal{H}$.*

By similarity to an explicit time marching method, the diagonal blocks of $\mathcal{T}$ are set according to the reciprocal of the CFL number:

$$\mathcal{T}_{(i,i)} = \left(\frac{1+\mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1+\mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1+\mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1+\mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, \frac{1+\mathcal{J}_{[i]}^{\frac{1}{D}}}{\mathcal{J}_{[i]}}, 1+\mathcal{J}_{[i]}^{\frac{1}{D}}\right). \tag{7.5}$$

Since the diagonal elements of $\mathcal{T}$ are not all identical when constructed in this way, there is no guarantee that equation (7.5) produces a dynamic inverse. However, provably stable versions of the continuation algorithm using this operator as the dynamic inverse are demonstrated to be convergent for the numerical examples investigated later in this chapter.

Accurate calculation of the tangent vector requires the Jacobian of $\mathcal{H}$ to be inverted. However, the tangent can be estimated using backwards differences. Using the diagonal matrix $\mathcal{T}$ as the dynamic inverse operator in conjunction with Remark 7.1 and applying first-order backwards-differencing to approximate the tangent vector gives the discrete update formula

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \left(\lambda_{k+1} - \lambda_k\right)\left[\gamma\mathcal{T}\mathcal{H}\left(\mathbf{q}_k,\lambda_k\right) + \frac{1}{\lambda_k - \lambda_{k-1}}\left(\mathbf{q}_k - \mathbf{q}_{k-1}\right)\right]. \tag{7.6}$$

Performing iterations according to equation (7.6) will be referred to as the single-stage MFMH continuation algorithm with Euler corrector and tangent predictor.

The use of backwards differencing to approximate the tangent vector violates condition (6.13), the consequence being that stability of the algorithm is no longer assured. It is shown later in this chapter that even though the algorithm can become unstable as a result of this approximation, it is possible to stabilize the algorithm using an explicit filter.

7.2 Two-Stage Formulation

Since the predictor and corrector portion of the update are independent, they can be separated. Unlike the matrix-present formulation, applying the predictor and corrector components separately incurs minimal additional computational cost. The benefit of this two-stage formulation is that information from the corrector iteration can be used to improve on the quality of the predictor.

The two-stage formulation of the MFMH algorithm with Euler corrector and secant predictor is given by

$$\begin{cases} \mathbf{q}_{k+\frac{1}{2}} = \mathbf{q}_k + \left(\lambda_{k+\frac{1}{2}} - \lambda_k\right)\gamma\mathcal{T}\mathcal{H}, \\ \mathbf{q}_{k+1} = \mathbf{q}_{k+\frac{1}{2}} + \frac{\lambda_{k+1}-\lambda_k}{\lambda_k-\lambda_{k-1}}\left(\mathbf{q}_{k+\frac{1}{2}} - \mathbf{q}_{k-\frac{1}{2}}\right). \end{cases} \tag{7.7}$$

This resembles a dual time marching method [142] where a single dual time step is applied as corrector using Euler's method. By this analogy, it is a logical design choice to set $\gamma = h_{\text{ref}} / \left|\lambda_{k+\frac{1}{2}} - \lambda_k\right|$ where $h_{\text{ref}} \in \mathbb{R}$ is a constant user-defined parameter. This allows for the dual time step size to be chosen independently of the parameter step size $|\Delta\lambda|$.

First-order backwards differencing is not the only estimate that can be used for the predictor. It can be useful to consider performance when using other predictors as well. A "rank 1" predictor refers to a secant predictor, a "rank 0" predictor refers to the case of $\mathcal{E} = 0$, and a "rank $\frac{1}{2}$" predictor refers to the case where the predictor alternates between the rank 0 and rank 1 predictor at successive iterations. A "rank 2" predictor indicates the use of a second degree Lagrange polynomial to extrapolate $\mathbf{q}_{k+1}$. Lagrange interpolation/extrapolation is covered by many textbooks, one example of which is Lomax et al. [99]. The formula for the point predicted using a second degree Lagrange polynomial extrapolation in the context of homotopy continuation is given below:

$$\mathbf{q}_{k+1} = \mathbf{q}_{k+\frac{1}{2}} l_k + \mathbf{q}_{k-\frac{1}{2}} l_{k-1} + \mathbf{q}_{k-\frac{3}{2}} l_{k-2}, \tag{7.8}$$

$$l_k = \frac{(\lambda_{k+1} - \lambda_{k-2})(\lambda_{k+1} - \lambda_{k-1})}{(\lambda_k - \lambda_{k-2})(\lambda_k - \lambda_{k-1})},\ l_{k-1} = \frac{(\lambda_{k+1} - \lambda_{k-2})(\lambda_{k+1} - \lambda_k)}{(\lambda_{k-1} - \lambda_{k-2})(\lambda_{k-1} - \lambda_k)},\ l_{k-2} = \frac{(\lambda_{k+1} - \lambda_{k-1})(\lambda_{k+1} - \lambda_k)}{(\lambda_{k-2} - \lambda_{k-1})(\lambda_{k-2} - \lambda_k)},$$
$$l_k,\ l_{k-1},\ l_{k-2} \in \mathbb{R}. \tag{7.9}$$

By analogy to a dual time marching method, the diagonal elements of $\mathcal{T}$ are assigned based on the reciprocal of the CFL number as given by equation (7.5). Since the use of the CFL number is applicable to the convection equation only, some additional calculations are needed to determine a more appropriate dual time constant for other operators. If $\mathcal{G}$ is the dissipation operator, then some analysis of the one-dimensional problem on a grid of constant grid spacing $\Delta x \in \mathbb{R}$ gives the relationship between the dual "time" steps $h_{\mathcal{G}}$ and $h_{\mathcal{R}}$ as

$$h_{\mathcal{G}} = \frac{\sqrt{\Delta x^2 + \sigma^2 h_{\mathcal{R}}^2} - \Delta x}{\sigma\mu}, \tag{7.10}$$

where $\sigma \in \mathbb{R}$ is the spectral radius as present in the dissipation operator. The local time step might then be calculated as

$$h_{\text{ref}} = \lambda h_{\mathcal{G}} + (1 - \lambda) h_{\mathcal{R}}. \tag{7.11}$$

There might be some benefit to assigning h_{ref} according to this formula. However, since it should not be assumed that explicit eigenvalue estimates will be available in general, the CFL number will be used for all studies in this chapter.

One detail that has so far been omitted from equation (7.7) is an update formula for λ_{k+1}. (Note that $\lambda_{k+\frac{1}{2}}$ is irrelevant when γ is set to the quantity given above.) Suppose that the initial value of $\Delta\lambda$

is supplied by the user. The formula resulting in constant $\Delta\lambda$ is

$$\lambda_{k+1} = 2\lambda_k - \lambda_{k-1}. \tag{7.12}$$

In the case where step-length adaptation is desired, the following update will ensure consistent $\|\Delta\mathbf{q}\|$ at each iteration:

$$\lambda_{k+1} = \lambda_k - \frac{\|\Delta\mathbf{q}\|_{\text{tar}}}{\left\| \frac{h_{\text{ref}}\mathcal{TH}(\mathbf{q}_k,\lambda_k)}{\lambda^*_{k+1}-\lambda_k} + \frac{\mathbf{q}_k + h_{\text{ref}}\mathcal{TH}(\mathbf{q}_k,\lambda_k) - \mathbf{q}_{k-1}}{\lambda_k - \lambda_{k-1}} \right\|}. \tag{7.13}$$

This equation must be solved iteratively until $\left|\lambda_{k+1} - \lambda^*_{k+1}\right|$ is below some tolerance. This can be accomplished by initializing λ^*_{k+1} from equation (7.12) and updating λ_{k+1} from equation (7.13), setting $\lambda^*_{k+1} \leftarrow \lambda_{k+1}$, and repeating the process. Setting λ_{k+1} in this way gives an update with $\|\mathbf{q}_{k+1} - \mathbf{q}_k\| \approx \|\Delta\mathbf{q}\|_{\text{tar}}$, where $\|\Delta\mathbf{q}\|_{\text{tar}} \in \mathbb{R}$ can be calibrated from the first iteration. The cost of this process, however, is significant. Some testing indicates that each iteration of this update formula may increase the cost of each MFMH iteration by as much as 15%.

Since the predictor portion of the MFMH update uses backwards differencing, a standard predictor-corrector algorithm is used to initialize the first interior traversing point. A rank 0 predictor is used from the first point and the corrector problem is solved at the second point by explicit time marching until a suitable user-defined tolerance is reached. If the distance between the curve points at the first two iterates is $\|\mathbf{q}_2 - \mathbf{q}_1\|$ and the first interior corrector problem is solved in n_2 iterations, then $\|\Delta\mathbf{q}\|_{\text{tar}}$ is taken as $\|\mathbf{q}_2 - \mathbf{q}_1\|/n_2$ and $\Delta\lambda_2$ is set to $\Delta\lambda_1/n_2$.

It may be possible to stabilize equation (7.6) using an explicit filter. The most successful explicit filter that we have applied is a λ-direction explicit kernel smoother. The filter uses the Nadaraya-Watson [117, 168] kernel weighted average, applied at the k-th iteration. The general formula for a smoothed point $\tilde{\mathbf{q}}$ at a given parameter value λ^* is

$$\tilde{\mathbf{q}}(\lambda^*) = \frac{\sum_{i=k+1-p}^{k+1} K_b(\lambda^*, \lambda_i)\,\mathbf{q}(\lambda_i)}{\sum_{i=k+1-p}^{k+1} K_b(\lambda^*, \lambda_i)}, \tag{7.14}$$

with Gaussian kernel function given by

$$K_b(\lambda^*, \lambda_i) = \exp\left(-\frac{(\lambda^* - \lambda_i)^2}{2b^2}\right), \tag{7.15}$$

where $K_b : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $b \in \mathbb{R}$, $b > 0$.

The smoothing is applied to the updated point $\mathbf{q}_{k+1}$ at $\lambda^* = \lambda_{k+1}$ after the update (7.6) is applied making use of p previous stages. The result is a general linear method [71] with coefficients depending on the parameter b, where $b = |\Delta\lambda_k|\, b_{\text{ref}}$, and $b_{\text{ref}} \in \mathbb{R}$, $b_{\text{ref}} > 0$ is a user input. Larger values of b_{ref} will result in more smoothing, the effect of which is increased stability at the cost of reduced curve-tracing accuracy. In this study, values in the range $0.5 \leq b_{\text{ref}} \leq 0.8$ were found to be suitable. For values of b_{ref} in this range, increasing the value of p beyond 2 has negligible effect.

A pseudo-code of the two-stage MFMH algorithm with explicit filter based on equation (7.6) is shown in Algorithm 7.1.

Algorithm 7.1: Two-stage matrix-free monolithic homotopy (MFMH) continuation with explicit Gaussian kernel filter

Initialize: Set $\lambda = 1$ and solve $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ if necessary. Take a step $\lambda \leftarrow \lambda + \Delta\lambda$ and solve $\mathcal{H}(\mathbf{q}, \lambda) = \mathbf{0}$ at the updated value of λ.

MFMH iterations: while $\lambda > 0$ **do**

- Calculate $\mathcal{H}$
- Set the diagonal matrix $\mathcal{T}[:] \leftarrow h_{\text{ref}}\mathcal{J}[:] / \left(1 + \mathcal{J}^{1/D}[:]\right)$
- Take the corrector portion of the update: $\mathbf{q}_{k+\frac{1}{2}} \leftarrow \mathbf{q}_k + h_{\text{ref}}\mathcal{T}\mathcal{H}$
- Step-length adaptation can be performed according to equation (7.13) if desired
- Perform a predictor step: $\mathbf{q}_{k+1} \leftarrow \mathbf{q}_{k+\frac{1}{2}} + \frac{\lambda_{k+1}-\lambda_k}{\lambda_k-\lambda_{k-1}}\left(\mathbf{q}_{k+\frac{1}{2}} - \mathbf{q}_{k-\frac{1}{2}}\right)$
- Smooth the update using the Gaussian kernel filter: $\mathbf{q}_{k+1} \leftarrow \frac{\mathbf{q}_{k+1}+\sum_{k-p+1}^{k} K_b(\lambda_{k+1},\lambda_i)\mathbf{q}_{i+\frac{1}{2}}}{1+\sum_{k-p+1}^{k} K_b(\lambda_{k+1},\lambda_i)}$

end

Inexact Newton Phase: Solve $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ to some tolerance using the inexact Newton method

7.3 A Stable Variant of Equation (7.6)

The main concern in equation (7.6) is that $\mathcal{E}(\mathbf{q}, \lambda)$ is approximated using a backwards difference formula. This invalidates condition (6.13) which is a condition for the stability proof. No continuous analogue can be made for the backwards-difference formulation, making analysis difficult. However, since increasing the curve-tracing error will worsen the backwards-difference approximation, it is intuitive that using the backwards-difference approximation in an iterative algorithm may result in cumulative error, which will cause instability regardless of $|\Delta\lambda|$.

In the case where no predictor is applied, eg. $\mathcal{E} = 0$, it can easily be shown that $V(\mathbf{z}(\lambda)) = \frac{1}{2}\|\mathbf{z}(\lambda)\|^2$ decays at a rate of

$$\frac{\mathrm{d}}{\mathrm{d}(-\lambda)} V(\mathbf{z}(\lambda)) \leq -\gamma\beta\|\mathbf{z}\|^2 + \mathbf{z}^T\dot{\mathbf{q}}_s. \tag{7.16}$$

Since the term $\mathbf{z}^T\dot{\mathbf{q}}_s$ is independent of γ and $|\Delta\lambda|$, it should be possible to achieve convergence if γ is sufficiently large. Since $|\Delta\lambda|$ will be taken inversely proportional to γ, the condition of requiring γ to be sufficiently large is equivalent to requiring that $|\Delta\lambda|$ be sufficiently small. So, numerically integrating equation

$$\dot{\mathbf{q}} = -\gamma\mathcal{T}\mathcal{H}(\mathbf{q}, \lambda) \tag{7.17}$$

can result in a stable algorithm for sufficiently small step size $|\Delta\lambda|$.

It is possible to integrate the stable variation of the MFMH algorithm given by equation (7.17) with either an Euler update or a more stable update. In this study, the standard fourth order Runge-Kutta (RK4) method is considered as an alternative to the explicit Euler update. This method can be found in numerous books such as that of Lomax et al. [99] and the update is given here in the context of

Algorithm 7.2: Single-stage matrix-free monolithic homotopy (MFMH) continuation

Initialize: Set $\lambda = 1$ and solve $\mathcal{G}(\mathbf{q}) = \mathbf{0}$ if necessary.
MFMH iterations: while $\lambda > 0$ **do**
  Calculate $\mathcal{H}$
  Set the diagonal matrix $\mathcal{T}[:] \leftarrow \mathcal{J}[:] / \left(1 + \mathcal{J}^{1/D}[:]\right)$
  Use equation (7.17) to update $\mathbf{q}$ applying explicit Euler or RK4 (7.18)
  Step-length adaptation can be performed similar to Algorithm 6.2 if desired
end
Inexact Newton Phase: Solve $\mathcal{R}(\mathbf{q}) = \mathbf{0}$ to some tolerance using the inexact Newton method

equation (7.17):

$$\begin{cases} \hat{\mathbf{q}}_{k+\frac{1}{2}} = \mathbf{q}_k - \frac{1}{2} h_{\text{ref}} \mathcal{T} \mathcal{H}\left(\mathbf{q}_k, \lambda_k\right), \\ \tilde{\mathbf{q}}_{k+\frac{1}{2}} = \mathbf{q}_k - \frac{1}{2} h_{\text{ref}} \mathcal{T} \mathcal{H}\left(\hat{\mathbf{q}}_{k+\frac{1}{2}}, \lambda_{k+\frac{1}{2}}\right), \\ \bar{\mathbf{q}}_{k+1} = \mathbf{q}_k - h_{\text{ref}} \mathcal{T} \mathcal{H}\left(\tilde{\mathbf{q}}_{k+\frac{1}{2}}, \lambda_{k+\frac{1}{2}}\right), \\ \mathbf{q}_{k+1} = \mathbf{q}_k - \frac{1}{6} h_{\text{ref}} \mathcal{T} \left[\mathcal{H}\left(\mathbf{q}_k, \lambda_k\right) + 2\left(\mathcal{H}\left(\hat{\mathbf{q}}_{k+\frac{1}{2}}, \lambda_{k+\frac{1}{2}}\right) + \mathcal{H}\left(\tilde{\mathbf{q}}_{k+\frac{1}{2}}, \lambda_{k+\frac{1}{2}}\right)\right) + \mathcal{H}\left(\bar{\mathbf{q}}_{k+1}, \lambda_{k+1}\right)\right], \end{cases} \tag{7.18}$$

where $\lambda_{k+\frac{1}{2}} = \lambda_k + \frac{1}{2}\Delta\lambda_k$.

A pseudo-code of the single-stage MFMH algorithm based on equation (7.17) is shown in Algorithm 7.2.

7.4 Performance Investigation for the Euler Equations

Test cases were performed in order to investigate the functionality of the algorithm and its various features. The test case is the two-dimensional NACA 0012 airfoil solved on mesh Ne. The flow is inviscid at Mach 0.3 and angle of attack 1°. The C_l and C_d values along the trajectory mapped by the MFMH algorithms are compared to accurate C_l and C_d curves generated using the PC algorithm.

7.4.1 Step Size $\Delta\lambda$

For the first study, the two-stage algorithm is employed with Euler corrector and rank 0 predictor, equivalent to the single stage algorithm with Euler parameter integration. No step-length adaptation is applied. The accuracy of the curve-tracing algorithm is investigated by varying the size of the constant step size $|\Delta\lambda|$ over the range of 1×10^{-4} to 1×10^{-7} with dual time constant $h_{\text{ref}} = 1 \times 10^{-5}$. The data collected from the study is shown in Figure 7.1.

The data shows the expected trend that the curve tracing accuracy is improved as the step size $|\Delta\lambda|$ is reduced. The curve is traced very inaccurately at $|\Delta\lambda| = 1 \times 10^{-4}$ and fairly accurately at 1×10^{-7}.

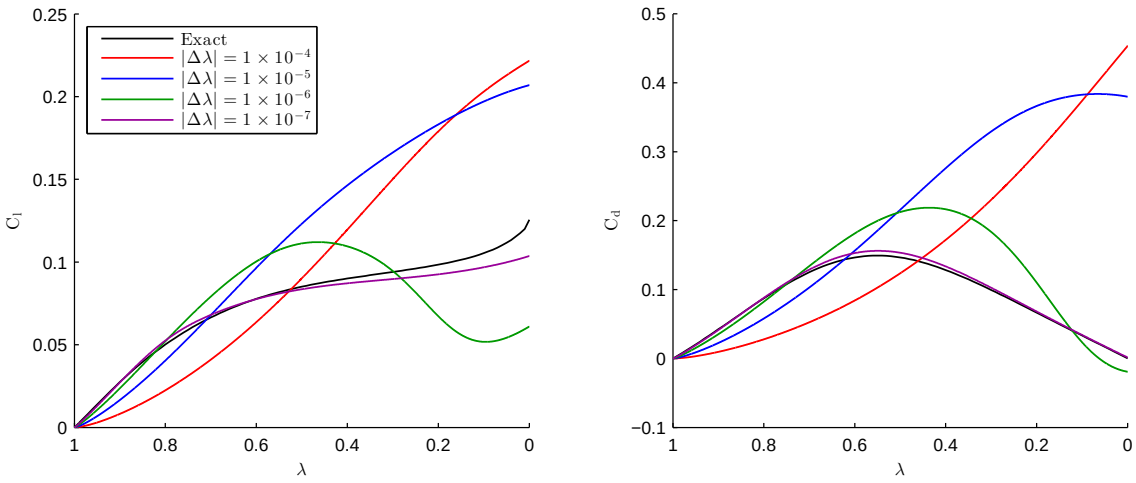


Figure 7.1: Two-stage MFMH algorithm with Euler corrector, rank 0 predictor, and no step-length adaptation; The effect of different $|\Delta\lambda|$ is investigated

7.4.2 Dual Time Step h_{ref}

For the second study, again the two-stage algorithm is employed with Euler corrector and rank 0 predictor with no step-length adaptation. The focus of this study is the effect of the reference dual "time" step h_{ref} on accuracy and stability. The reference step size h_{ref} was varied between 1×10^{-6} and 1×10^{-5} while $|\Delta\lambda|$ was varied between 1×10^{-4} and 1×10^{-7}. The data from the study is plotted in Figure 7.2.

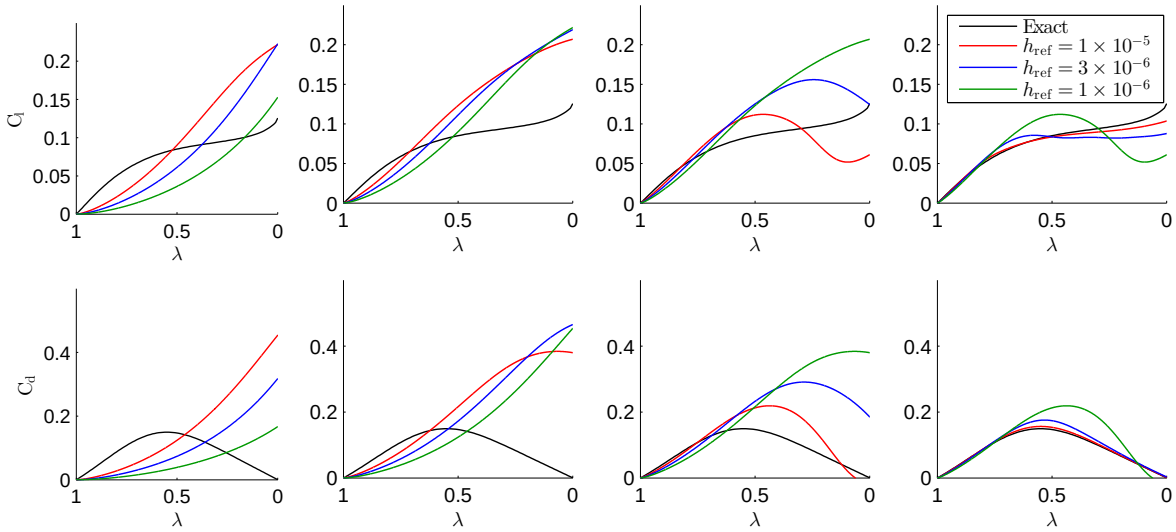
It was found that the algorithm becomes unstable for h_{ref} values around $h_{\text{ref}} = 3 \times 10^{-5}$ but demonstrates no symptoms of instability at $h_{\text{ref}} = 1 \times 10^{-5}$. It is apparent from the plots that the $h_{\text{ref}} = 1 \times 10^{-6}$ cases produce nearly identical curves to the $h_{\text{ref}} = 1 \times 10^{-5}$ case on the next larger $|\Delta\lambda|$. To clarify, taking ten times more integration steps at one tenth the reference dual step size h_{ref} has minimal impact on the curve-tracing accuracy. This is an important observation as it indicates that h_{ref} should be chosen as large as possible so long as the algorithm remains stable.

7.4.3 Predictor Variants for the Two-Stage Algorithm

The third study is an investigation of how the accuracy and stability are affected by predictor choice. The two-stage algorithm was used with Euler corrector and $h_{\text{ref}} = 1 \times 10^{-5}$. The data from this study is plotted in Figure 7.3.

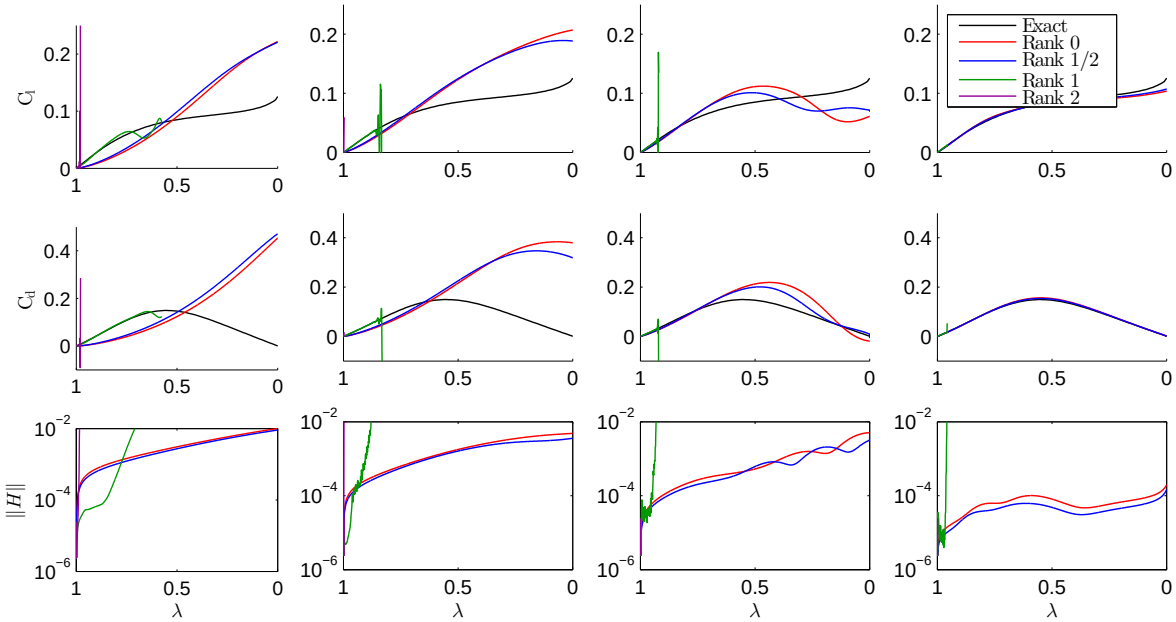
It is observed that the rank 1 predictor traces the curve much more accurately but results in instability early on. The fact that instability occurs earlier when $|\Delta\lambda|$ is made smaller suggests that the instability is more closely related to the number of iterations performed than the progress in λ. A similar issue was observed previously for the matrix-present MH algorithm in Chapter 6 and was identified as being related to the presence of high-frequency low-amplitude error which can accumulate as the iterations progress.

Using the rank $\frac{1}{2}$ predictor alleviates this instability but loses almost all of the additional accuracy incurred by the rank 1 predictor. The rank 2 predictor destabilizes even sooner than the rank 1 predictor. The use of an explicit filter to stabilize this method is investigated in the fifth study.



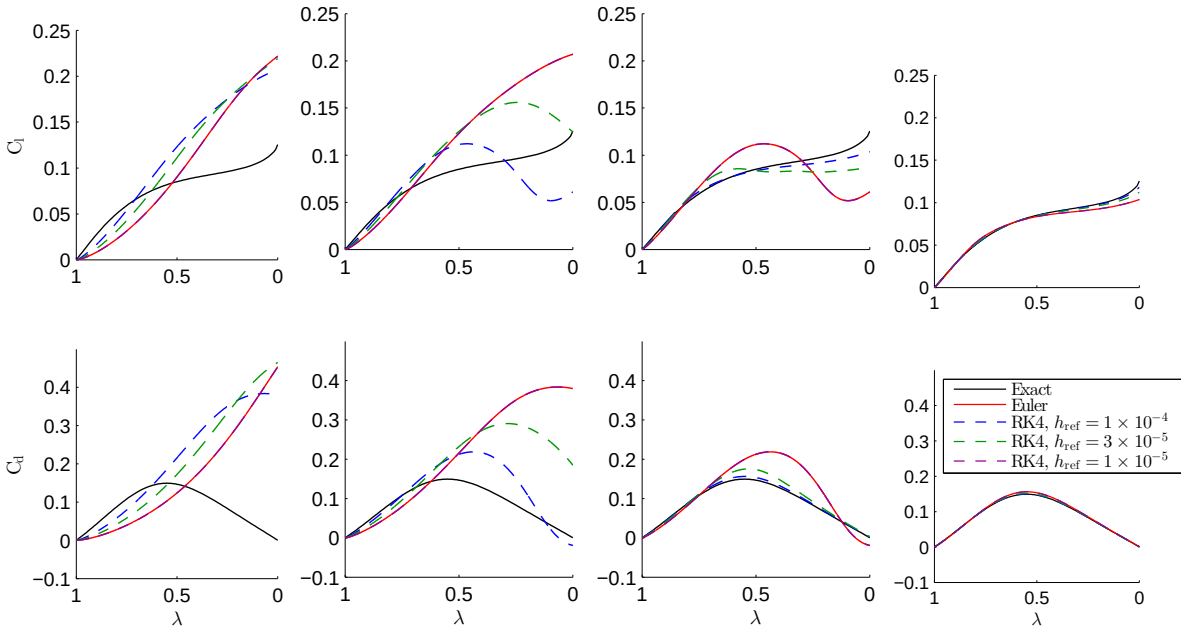
(a) $|\Delta\lambda| = 1 \times 10^{-4}$ (b) $|\Delta\lambda| = 1 \times 10^{-5}$ (c) $|\Delta\lambda| = 1 \times 10^{-6}$ (d) $|\Delta\lambda| = 1 \times 10^{-7}$

Figure 7.2: Two-stage MFMH algorithm with Euler corrector, rank 0 predictor, and no step-length adaptation; the effect of h_{ref} is investigated at several values of $|\Delta\lambda|$



(a) $|\Delta\lambda| = 1 \times 10^{-4}$ (b) $|\Delta\lambda| = 1 \times 10^{-5}$ (c) $|\Delta\lambda| = 1 \times 10^{-6}$ (d) $|\Delta\lambda| = 1 \times 10^{-7}$

Figure 7.3: Two-stage MFMH algorithm with Euler corrector, $h_{\text{ref}} = 1 \times 10^{-5}$, and no step-length adaptation; the effect of different rank predictors is investigated



(a) $|\Delta\lambda| = 1 \times 10^{-4}$ (b) $|\Delta\lambda| = 1 \times 10^{-5}$ (c) $|\Delta\lambda| = 1 \times 10^{-6}$ (d) $|\Delta\lambda| = 1 \times 10^{-7}$

Figure 7.4: Single stage MFMH algorithm with no step-length adaptation; the effectiveness of RK4 parameter integration is compared to Euler with $h_{\text{ref}} = 1 \times 10^{-5}$. Note that the trajectory of the RK4 algorithm with $h_{\text{ref}} = 1 \times 10^{-5}$ is visually indistinguishable from that of the Euler algorithm

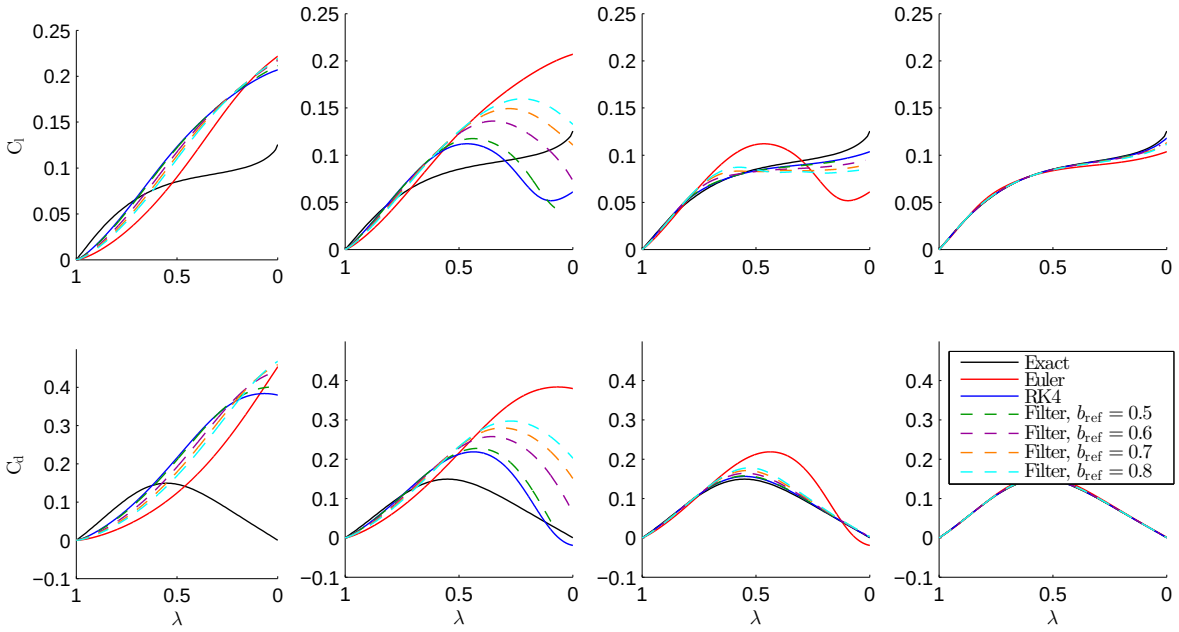
7.4.4 Runge-Kutta Integration for the Single-Stage Algorithm

For the fourth study, the efficiency gained by augmenting the single-stage algorithm given by equation (7.17) with RK4 dual parameter integration is investigated. The performance investigation encompasses several h_{ref} in the range of 1×10^{-4} to 1×10^{-5} for constant step size $|\Delta\lambda|$ ranging from 1×10^{-4} to 1×10^{-7}. The data from this study is shown in Figure 7.4.

It was found that the RK4 algorithm gave a nearly identical trajectory as the explicit Euler corrector when $h_{\text{ref}} = 1 \times 10^{-5}$ was used for both algorithms. The advantage of RK4 comes from being able to use a larger h_{ref} values. It was found that h_{ref} could be increased by an order of magnitude before stability issues were encountered. Though RK4 comes at four times the cost for a given $|\Delta\lambda|$, a tenfold cost increase would be required to achieve the same accuracy increase using the explicit Euler update, and so RK4 is actually about 2.5 times more efficient based on these values.

7.4.5 Filter-Stabilized Two-Stage Algorithm

The Gaussian kernel filter-stabilized method with $p = 2$, rank 1 predictor and $h_{\text{ref}} = 1 \times 10^{-5}$ from equation (7.6) was compared to the explicit Euler method applied to equation (7.17) with $h_{\text{ref}} = 1 \times 10^{-5}$ and RK4 applied to equation (7.17) with $h_{\text{ref}} = 1 \times 10^{-4}$. Several values of b were investigated in the range $b = 0.5$ to $b = 0.8$. The data is shown in Figure 7.5.



(a) $|\Delta\lambda| = 1 \times 10^{-4}$ (b) $|\Delta\lambda| = 1 \times 10^{-5}$ (c) $|\Delta\lambda| = 1 \times 10^{-6}$ (d) $|\Delta\lambda| = 1 \times 10^{-7}$

Figure 7.5: Two-stage MFMH algorithm with rank 1 predictor including Gaussian kernel filtering and no step-length adaptation; the filtered algorithm has $h_{\text{ref}} = 1 \times 10^{-5}$ and is compared to RK4 at 1×10^{-4} and explicit Euler with rank 0 predictor at $h_{\text{ref}} = 1 \times 10^{-5}$

The filter-stabilized algorithm, when it completes without becoming unstable, provides significant accuracy improvement over the explicit Euler method but is less accurate than the RK4 method. However, the cost increase of the filter-stabilized algorithm is only about 17% whereas the cost increase of the RK4 algorithm is approximately 300%. It also appears that the minimum value of b_{ref} needed for stability increases as $|\Delta\lambda|$ decreases; the $b_{\text{ref}} = 0.5$ case failed when $|\Delta\lambda| = 1 \times 10^{-5}$, as did the $b_{\text{ref}} = 0.5$ and $b_{\text{ref}} = 0.6$ cases for both $|\Delta\lambda| = 1 \times 10^{-6}$ and $|\Delta\lambda| = 1 \times 10^{-7}$. The rank 2 predictor was also tested with the filter but either became unstable or converged with the same accuracy as the rank 1 case for all $\|\Delta\lambda\|$ that were tested.

7.4.6 Comparison of Algorithm Variants

Since the single-stage RK4 algorithm was less accurate but also less expensive than the two-stage explicitly filtered rank 1 predictor algorithm, the relative effectiveness of the algorithms is evaluated by plotting the error in C_l and C_d at the end of the continuation phase versus the CPU wall time, where the wall time is measured from the beginning of iterations until the end of the continuation phase. The data is plotted in Figure 7.6.

For this case study, the two-stage rank 1 filtered algorithm and the single-stage MS4 algorithm give comparable performance and either of these two algorithms is more efficient than the basic rank 0 predictor algorithm.

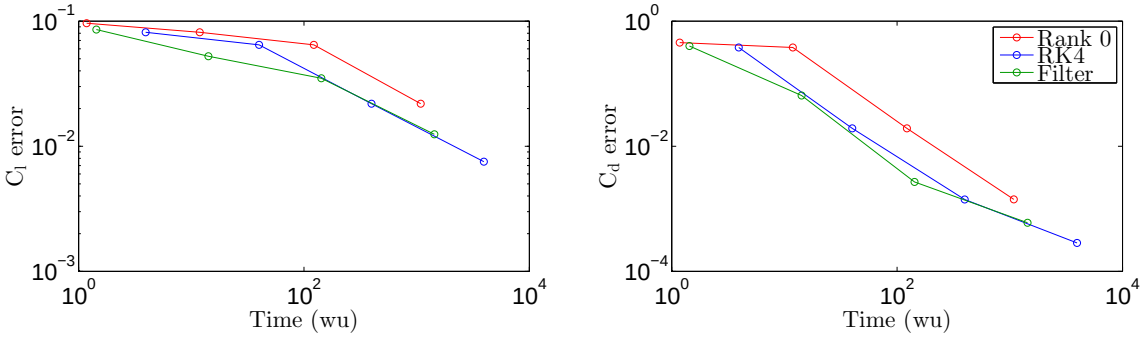


Figure 7.6: Globalization error versus work units for the rank 0 MFMH algorithm, the two-stage MFMH algorithm with RK4, and the single-stage rank 1 filtered MFMH algorithm

7.5 Summary

A matrix-free formulation of the monolithic homotopy continuation algorithm was developed. The original two-stage algorithm (7.6) was found to be unstable when using a backwards-difference approximation to the tangent. This stability issue could be addressed by applying an explicit filter.

An alternative stable matrix-free monolithic homotopy continuation algorithm could be constructed by considering a single stage version (7.17). The efficiency of this algorithm was improved by dual parameter integration using RK4, which allowed for a larger dual time step h_{ref} to be used, resulting in greater curve-tracing accuracy for a given $|\Delta\lambda|$ and improving the efficiency of the algorithm. Numerical testing of the two algorithms on a simple two-dimensional subsonic test case demonstrated comparable performance of the two algorithms.

Other convergence acceleration techniques, such as multigrid, are expected to be beneficial but were not investigated, and the efficiency of the method currently cannot compete with our previously-developed matrix-present monolithic homotopy continuation algorithm. The method can be of practical value to implementations where it is not desirable to implement any matrix inversion, either to reduce implementation time or to avoid the additional memory overhead. Of more immediate value, the analysis of the new algorithms has improved our understanding of monolithic homotopy continuation algorithms in general and may guide future development of these methods.

Chapter 8

Performance Studies

Performance studies are conducted to evaluate the performance of the new homotopy continuation algorithms compared to the established pseudo-transient continuation algorithm. Test suites are constructed over a wide range of subsonic and transonic conditions for inviscid, laminar, and turbulent flows. Parameters for each test suite are chosen to attempt to match robustness for all algorithms (e.g. a similar number of cases should converge for each algorithm) and then comparing the time to convergence for each algorithm, averaged over all cases that converged for all three algorithms. The range of test cases included in each test suite often includes several configurations which are dynamically unstable and not expected to converge.

For all homotopy cases, the convex homotopy with dissipation operator and far-field boundary conditions, as described in Section 4.4.2, is used as the homotopy system since this has generally given the most reliable performance. An Euler predictor is used in all instances of the CHC-PC algorithm and the algebraic method is used to calculate the tangent vector. Step-length adaptation, as described in this thesis, is applied to all homotopy continuation algorithms. Since testing is performed over a range of subsonic and transonic flow conditions in each study, and it is not always known which cases will be transonic and which will be subsonic, the second-difference dissipation with the pressure sensor is active for all cases.

A maximum wall time is imposed for each test suite. If the flow solve has failed to converge in the allotted time, the case is terminated and reported as failure. The maximum wall time used for each test suite is high and has rarely resulted in the termination of a case that appeared to be converging. A listing of the maximum wall times imposed for each test suite is included in Table 8.1.

In general, more conservative continuation parameters must be chosen for laminar flow cases than inviscid flow cases and even more conservative parameters must be chosen for turbulent flows than laminar. The parameters chosen for the pseudo-transient method are based on our experience and testing and are consistent with those used by previous authors [14, 25, 33, 34, 63, 130]. The parameters chosen for the homotopy methods are similarly based on our experience and testing. The algorithm parameters used for each test suite are listed in Table 8.1 for all three algorithms. For all algorithms and for all test suites, the entire suite has been executed for several combinations of parameters to ensure that no algorithm is significantly under-represented due to poor parameter selection, though fine-tuning of parameters has been avoided. Fine-tuning has also been avoided by virtue of the fact that a large number of test cases has been included in each test suite.

8.1 Data Presentation

Timing data is shown in the bar-graph figures accompanying each test suite, an example of which is Figure 8.1. All timing statistics are reported in TauBench work units, as explained in Section 3.10. If data for a particular case is missing from the figure then this indicates that that particular case failed to converge. The white portion of the bar-graph labeled INP indicates the portion of the flow solve corresponding to the inexact Newton phase whereas the coloured portion corresponds to the continuation phase. The total cost of the flow solve is the total height of the bar including both coloured and white portion.

In addition to timing statistics, sample convergence histories are given for each case. The convergence history is characterized with four plots, an example of which is Figure 8.2. Each marker on the plots represents a nonlinear iteration. The amount of time taken for each nonlinear iteration depends mainly on the number of Krylov iterations as well as the cost of the matrix-vector products. The dashed vertical lines indicate the switch to the inexact Newton phase and the solid vertical lines indicate that the flow solve has completed.

The top left plot shows the evolution of the % error in C_L with time for each continuation algorithm based on the final C_L value. Relative error values below $10^{-6}\%$ are not displayed. The top right plot shows the evolution of the residual for PTC and the evolution of λ for CHC-PC and CHC-MH. The residual for PTC is shown on the left vertical axis and λ for the CHC algorithms is shown on the right vertical axis. The plot on the bottom left shows the residual history, either $\mathcal{R}(\mathbf{q})$ for PTC or $\mathcal{H}(\mathbf{q}, \lambda)$ for CHC. In the inexact Newton phase, both residuals are $\mathcal{R}(\mathbf{q})$. However, the homotopy and pseudo-transient residuals are scaled differently for turbulent cases and cannot be directly compared, even in the inexact Newton phase. The final plot on the bottom right shows a comparison of the residuals of the two homotopy algorithms during traversing.

8.2 The High-Performance Computing System

All cases were run on the general purpose cluster on SciNet [98]. The general purpose cluster is an IBM iDataPlex cluster based on Intel's Nehalem architecture. At the time of writing this thesis, it consists of 3,780 IBM iDataPlex DX360M2 nodes. At eight Intel Xeon E5540 2.53GHz cores per node, this is a total of 30,240 cores, with 16GB RAM per node, averaging just under 2GB per core since about 2GB per node is reserved for the operating system. The cluster is connected with a combination of DDR and QDR InfiniBand.

8.3 Inviscid Cases

The inviscid cases are three-dimensional flows over the ONERA M6 wing. Performance testing is done on each of grids Me1 and Me2, where Me2 was generated by refining grid Me1 by a factor of two in each direction and also splitting the blocks once in each direction, resulting in eight times more grid nodes and blocks. The parameters used for the pseudo-transient flow solves on the coarser mesh Me1

Flow								
	Grid	Me1	Me2	MlHC	MlHH	Nt	MtHH	MtHC
	Equations	3D inviscid	3D inviscid	3D laminar	3D laminar	2D RANS-SA	3D RANS-SA	3D RANS-SA
	Ma	0.2, 0.3, ..., 0.9	0.2, 0.3, ..., 0.9	0.2, 0.3, ..., 0.9	0.2, 0.3, ..., 0.9	0.2, 0.3, ..., 0.9	0.4, 0.6, 0.75, 0.85	0.4, 0.6, 0.8, 0.9
	α (degrees)	0, 3,6, 9,12	0, 3, 6, 9, 12	0, 3, 6, 9, 12	0, 3, 6, 9, 12	0, 4, 8, 12	0, 1.5, 3, 4.5	0, 1.5, 3, 4.5
	Re	-	-	1000	1000	4.00×10^7	1.172×10^7	1.00×10^7
Max. wall time (wu)		533	752	470	188	31	564	2821
PTC	a	0.01	0.01	0.01	0.01	0.0001	0.001	0.0001
	b	1.4	1.4	1.5	1.5	1.35	1.1	1.1
	m	3	3	2	2	1	1	1
	ILU(p) fill	0	0	2	2	2	2	3
	MVPs	AMVPs	AMVPs	AMVPs	AMVPs	AMVPs	AMVPs	FDMVP
	τ_l	0.05	0.05	0.01	0.01	0.01	0.01	0.01
	τ_rel	0.05	0.05	0.001	0.001	0.0001	0.0001	10^{-5}
	Converged	32/40	21/40	24/40	28/40	31/32	16/16	13/16
	Rel. Time	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CHC-PC	$\lvert\Delta\lambda_0\rvert$	0.20	0.20	0.20	0.20	0.04	0.03	0.03
	ILU(p) fill	1	1	2	2	2	2	3
	MVP	AMVPs	AMVPs	AMVPs	AMVPs	AMVPs	AMVPs	FDMVPs
	τ_l	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	τ_s	0.5	0.5	0.1	0.1	0.5	0.5	0.1
	Converged	32/40	27/40	27/40	30/40	32/32	16/16	13/16
	Rel. Time	0.88	0.85	0.79	0.91	0.73	1.14	3.33
CHC-MH	$\lvert\Delta\lambda_0\rvert$	0.20	0.20	0.05	0.05	0.04	0.02	0.02
	ILU(p) fill	1	1	2	2	2	2	3
	MVP	AMVPs	AMVPs	AMVPs	AMVPs	AMVPs	FDMVPs	FDMVPs
	τ_l	0.01	0.01	0.01	0.01	0.01	0.01	0.03
	Converged	33/40	27/40	29/40	31/40	32/32	16/16	14/16
	Rel. Time	0.77	0.66	0.66	0.60	0.55	0.64	1.00
INP	τ_l	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	ILU(p) fill	2	2	2	2	2	3	3

Table 8.1: Summary of test case suites and performance statistics for all performance comparisons; τ_l is the relative linear solver tolerance; τ_rel is the relative tolerance required for globalization of PTC and τ_s is the relative tolerance to which the subproblems are converged for CHC-PC; Rel. Time (relative time) is calculated as follows: the average wall time to complete a flow solve is calculated for each algorithm, considering only the flow solves at the operating conditions that converged for *all three algorithms*; this average wall time is then divided by the average wall time taken by PTC

are similar to those obtained by Dias and Zingg [34] and Dias [33] through several parameter studies. For the predictor-corrector method on the original coarser mesh, it has been found from some testing that it is sufficient to solve the subproblems to a relative tolerance of only 0.5 and that a large initial step size of $|\Delta\lambda_0| = 0.2$ could be used. The same initial step size is used for the monolithic method. A listing of the algorithm parameters used for all three algorithms is provided in Table 8.1. For this test suite, and all test cases in this chapter, the convex homotopy is used with dissipation operator with far-field boundary conditions as the homotopy system.

Timing data for the complete flow solve for all three algorithms averaged over all test cases is shown in Figure 8.1 and Table 8.1 for the two test suites. On the coarser grid, the convex homotopy with the predictor-corrector method has performed only slightly better than the pseudo-transient method with a similar success rate and a 12% reduction in wall time. The monolithic homotopy algorithm has in turn performed slightly better than the predictor-corrector algorithm. On average, on the coarser grid, convergence of the algorithm is achieved in about 23% less wall time with the monolithic homotopy continuation algorithm than with pseudo-transient continuation.

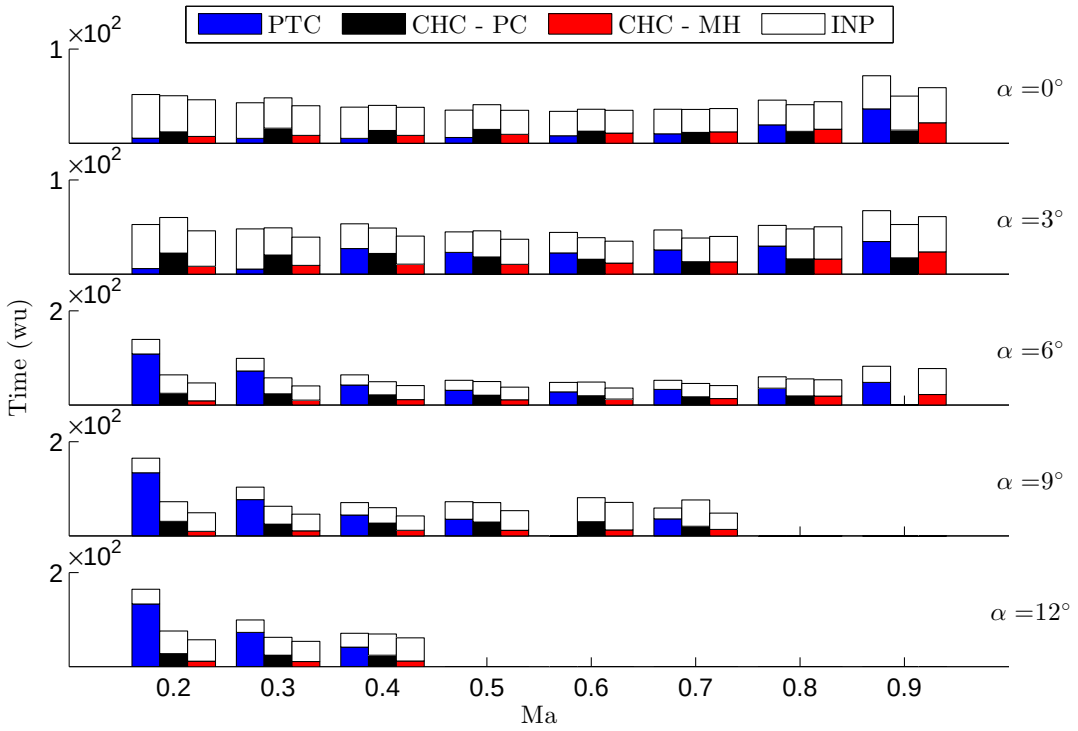
To investigate algorithm scalability, the parameters that have been determined to be suitable for the coarser mesh are used for the same test suite on the finer mesh. For the test suite on the finer mesh, the relative timing data of the homotopy algorithms improved slightly compared to the pseudo-transient algorithm, with the PC algorithm converging in 15% less time and the MH algorithm converging in 34% less time. However, the change in robustness is more significant, with the pseudo transient method converging in only 21/40 cases, down from 32/40 on the coarser mesh, while the PC algorithm converged in 26/40 and the MH algorithm converged in 27/40. Of course these statistics could be improved by parameter tuning, but that is not the purpose of the study.

The convergence histories for all three algorithms for a representative case from both test suites are shown in Figure 8.2. While it seems likely from this figure that the PTC algorithm would converge more efficiently if the switching tolerance for the inexact Newton phase were selected less conservatively, determining the most efficient switching tolerance a priori is impossible and so this over-solving is typical and expected for PTC.

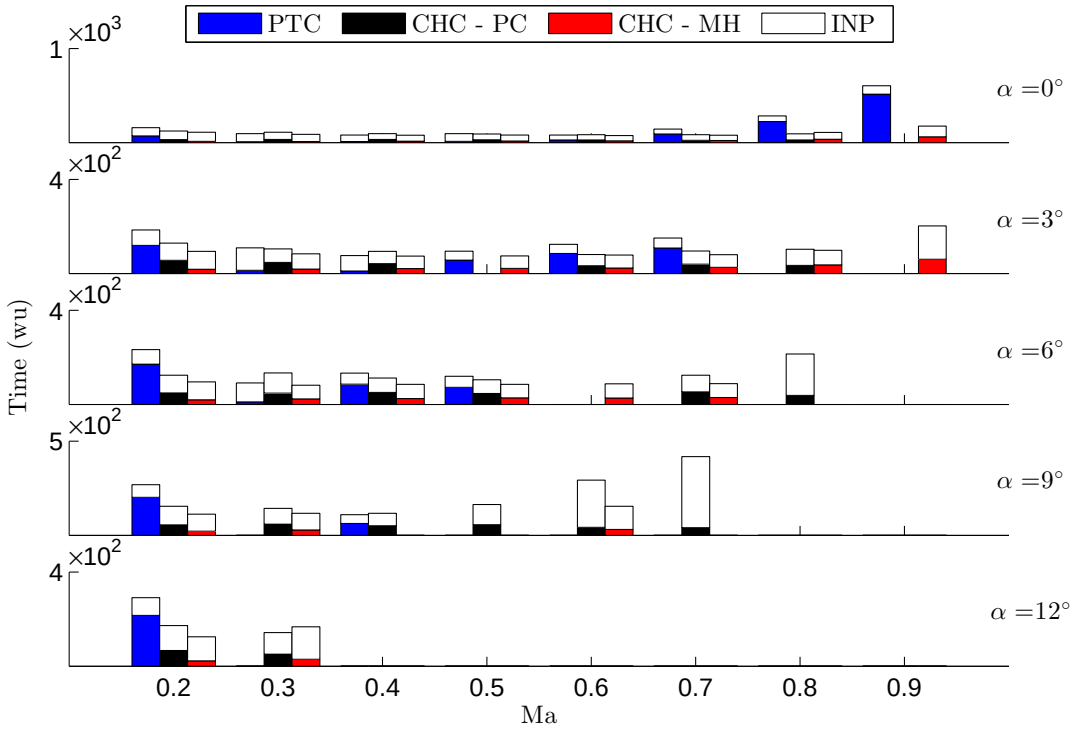
8.4 Laminar Cases

The laminar cases are three-dimensional flows over the ONERA M6 wing at Reynolds number 1000. Grids MlHC and MlHH are used. The parameters chosen for each algorithm are more conservative than the parameters chosen for the inviscid cases. The same algorithm parameters were used on both grids. The specific parameters used for all three algorithms on both grids are listed in Table 8.1.

The timing data for the two test suites is shown in Figure 8.3 and Table 8.1. Convergence histories for all three algorithms for a representative case are shown in Figure 8.4. Both the performance and relative performance of the different algorithms were slightly inconsistent on the two meshes. On average, for grid MlHC, the CHC-PC algorithm converged in 21% less time than PTC and the CHC-MH method converged in 34% less time than PTC. In addition, more cases were successfully converged using both homotopy continuation algorithms. On average, on grid MlHH, the CHC-PC algorithm converged in only 9% less wall time than PTC and the CHC-MH method converged in 40% less wall time than PTC. Again, more cases converged with the homotopy algorithms than with the pseudo-transient algorithm.

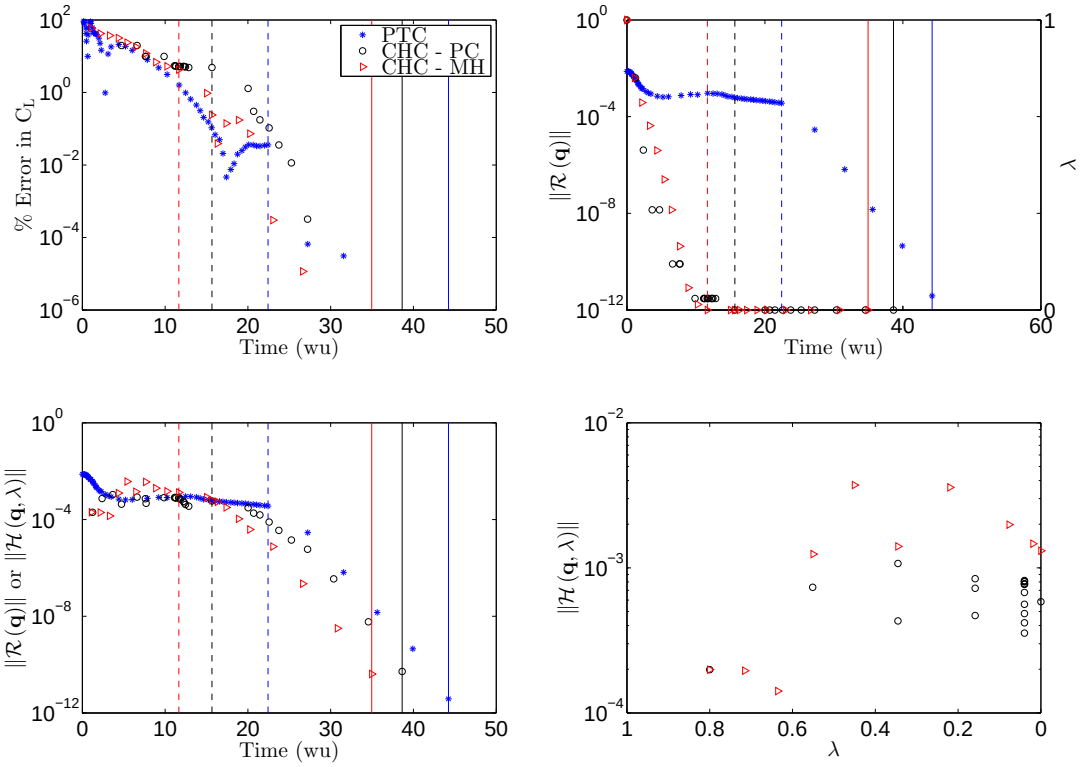


(a) Grid Me1

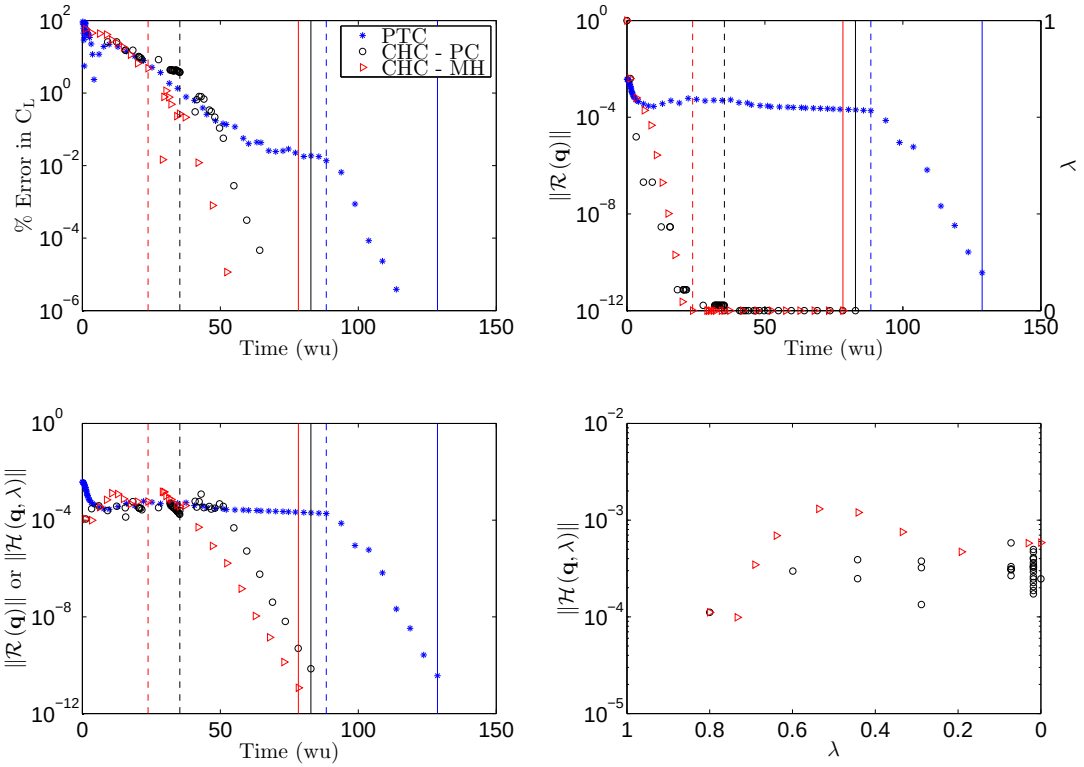


(b) Grid Me2

Figure 8.1: Performance comparison of several continuation algorithms for inviscid flows over the ONERA M6 wing



(a) Grid Me1



(b) Grid Me2

Figure 8.2: Convergence history for different continuation algorithms for inviscid flow over the ONERA M6 wing at Mach 0.6 and angle of attack 3°

Many of the mid-range Mach number and angle of attack cases did not converge for all algorithms. In many of these cases, the inexact Newton phase was reached and the residual was reduced by 5 or 6 orders of magnitude before stalling. While it is not clear why these cases are failing, these failures appear for all continuation algorithms and are not of major importance to the performance comparison.

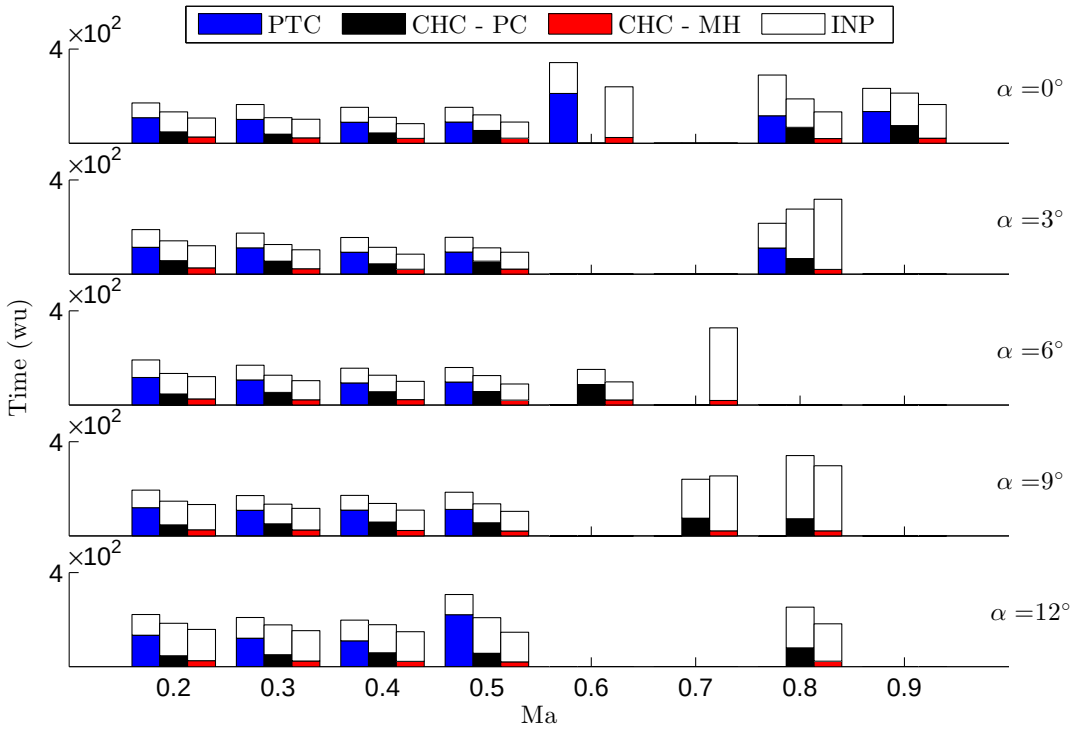
8.5 Turbulent Cases

Three turbulent test suites are investigated. One uses the NACA 0012 airfoil with grid Nt and is at Reynolds number 4×10^7. The other two are on the ONERA M6 wing. One is at Reynolds number 1.172×10^7 on grid MtHH and the other is at Reynolds number 1×10^7 on the finer grid MtHC. As with the previous studies, the parameters used for all three test suites are listed in Table 8.1.

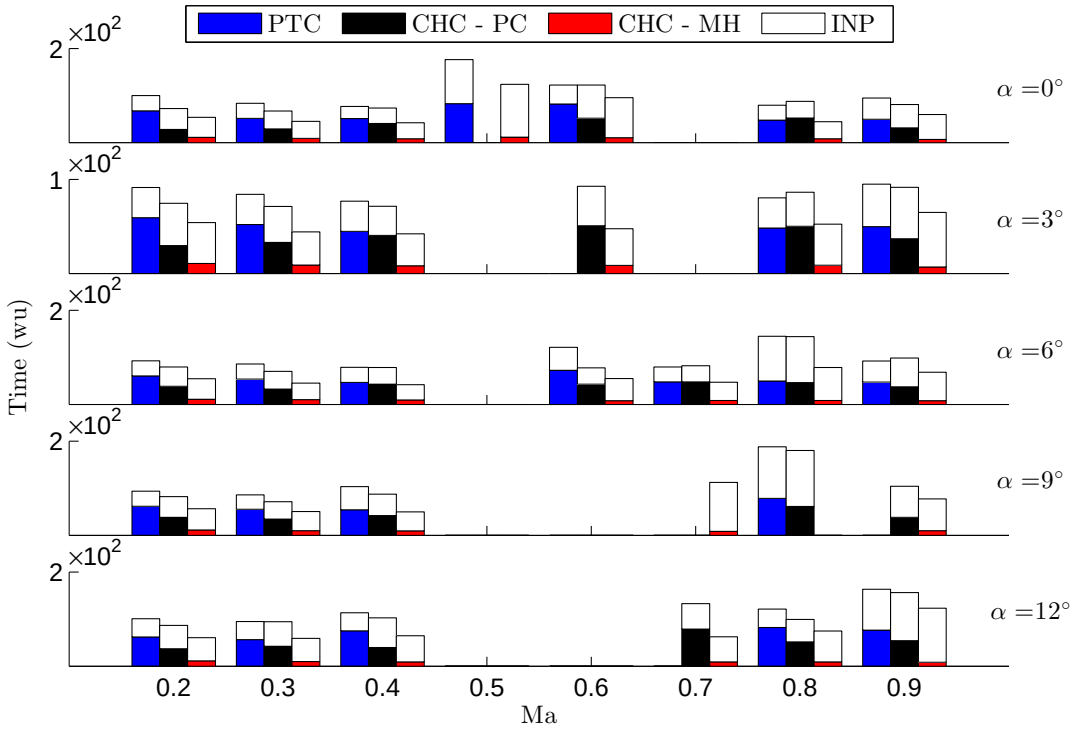
The performance data for the NACA 0012 test suite is shown in Figure 8.5 and Table 8.1. Convergence histories for all three algorithms for a representative case are shown in Figure 8.6. Flow solves were completed successfully across the entire range of input parameters investigated. On average, the CHC-PC algorithm converged in 27% less wall time than the PTC method and the CHC-MH algorithm converged in 45% less wall time than PTC, all with 100% success rate. It was found that the approximate matrix-vector products could be used for the CHC-MH algorithm without incurring a robustness penalty.

The performance data for the ONERA M6 test suite on grid MtHH is shown in Figure 8.7 and Table 8.1. Convergence histories for all three algorithms for a representative case are shown in Figure 8.8. The success rate of all three algorithms was also high in this case. The finite-differencing method was used to compute the matrix-vector products for the MH method. While it has been found that using the approximate matrix-vector products for this suite resulted in similar performance statistics, the deformation residual increased throughout traversing to the point where algorithm stability became a concern. From Figure 8.8, it appears that stability may still be a concern for these cases, even with the FDMVPs. While the need to use FDMVPs has implications on cost, the flow solves were on average still completed in 36% less wall time with the CHC-MH method than with PTC, whereas the CHC-PC method took 14% longer than PTC.

The turbulent test suite on the ONERA M6 wing with grid MtHC was the least successful for the homotopy methods. The timing data for this test suite is shown in Figure 8.9 and Table 8.1. Convergence histories for all three algorithms for a representative test case are shown in Figure 8.10. The CHC-PC method was found to be unreliable for this test suite unless a traceback method is employed, where the predictor phase is repeated if a subproblem fails to converge to the specified tolerance. Since this method is needed for convergence in many cases, it would appear that the step-length adaptation is not adequate to determine an appropriate step size for this case. Using this method is expensive and inefficient but at least the success rate of PTC could be matched. For the MH cases, the row/column normalization scaling was used instead of the geometric scaling based on the earlier observations that this improves the stability of the algorithm. Since using this scaling results in significantly more error reduction in each linear solve, but also increases the cost of each linear solve significantly, the linear solver tolerance was relaxed to 0.03 for the MH algorithm. On average, the CHC-MH flow solves took about the same wall time to converge than the PTC flow solves and was successful in one additional case.

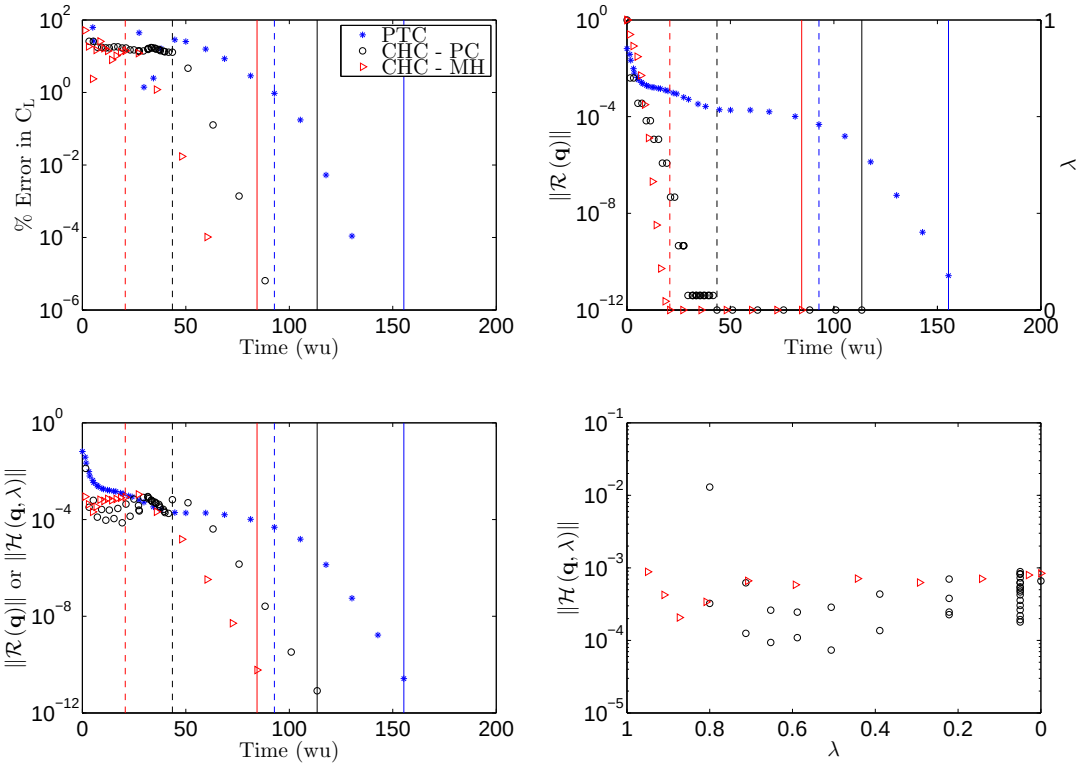


(a) Grid MlHC

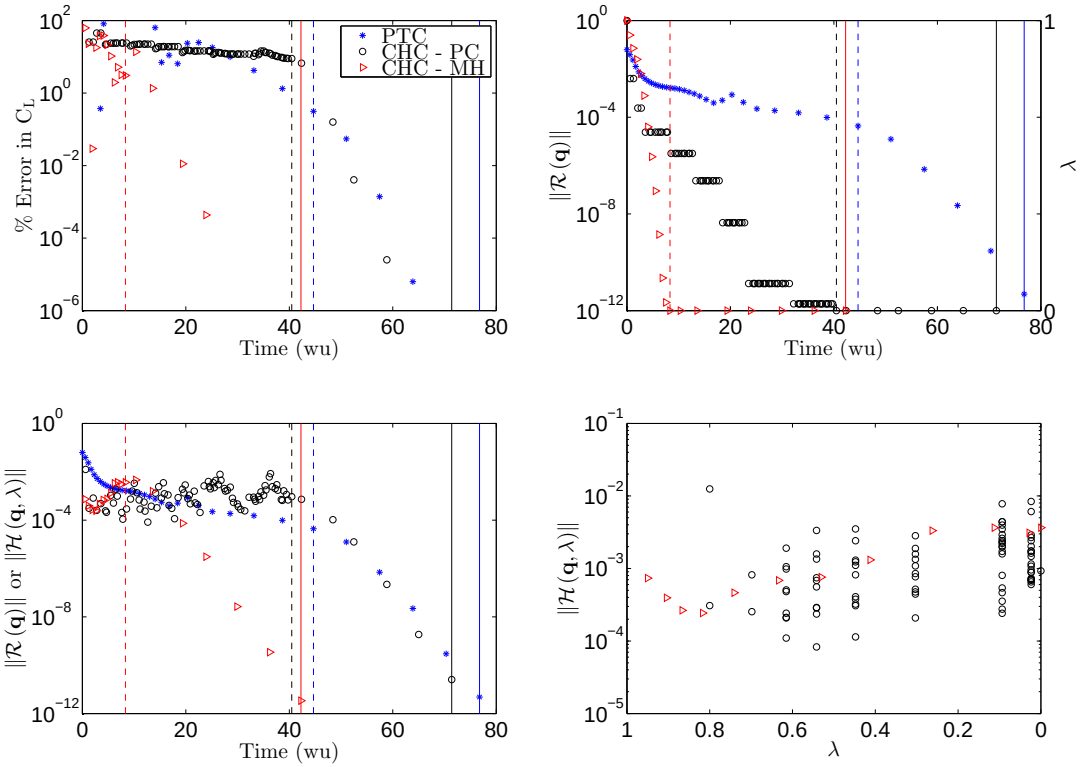


(b) Grid MlHH

Figure 8.3: Performance comparison of several continuation algorithms for laminar flows over the ONERA M6 wing



(a) Grid MlHC



(b) Grid MlHH

Figure 8.4: Convergence history for different continuation algorithms for laminar flow over the ONERA M6 wing at Mach 0.4 and angle of attack 3°

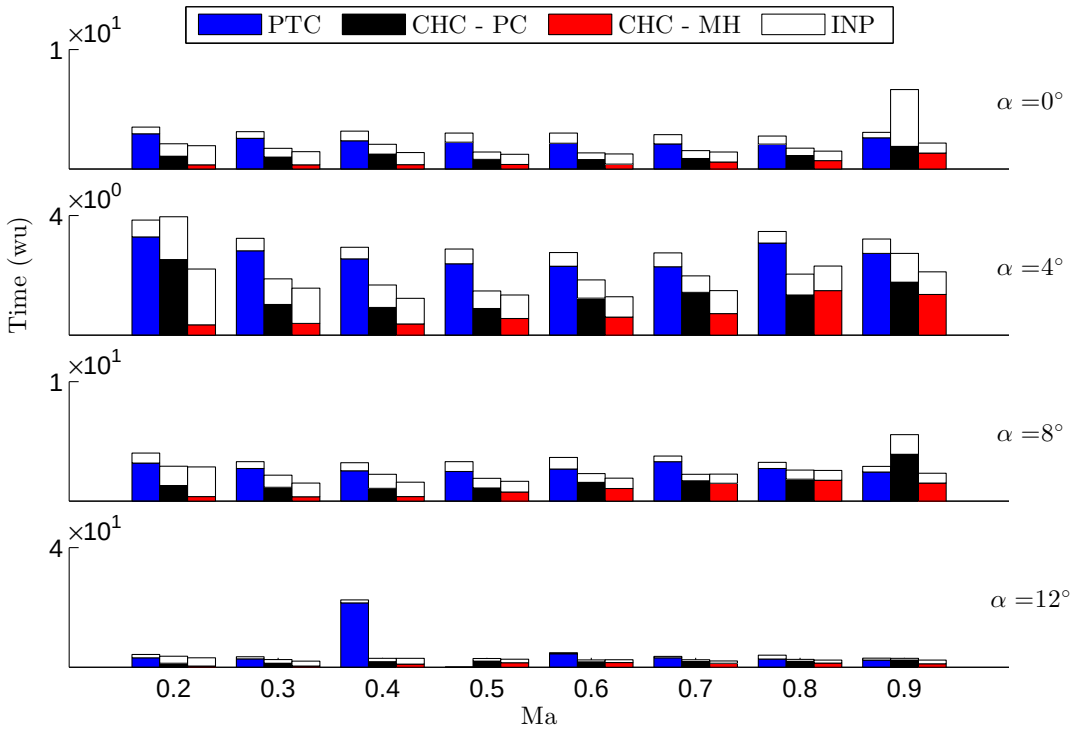


Figure 8.5: Performance comparison of several continuation algorithms for turbulent flows at Reynolds number 4×10^7 over the NACA 0012 airfoil on grid Nt

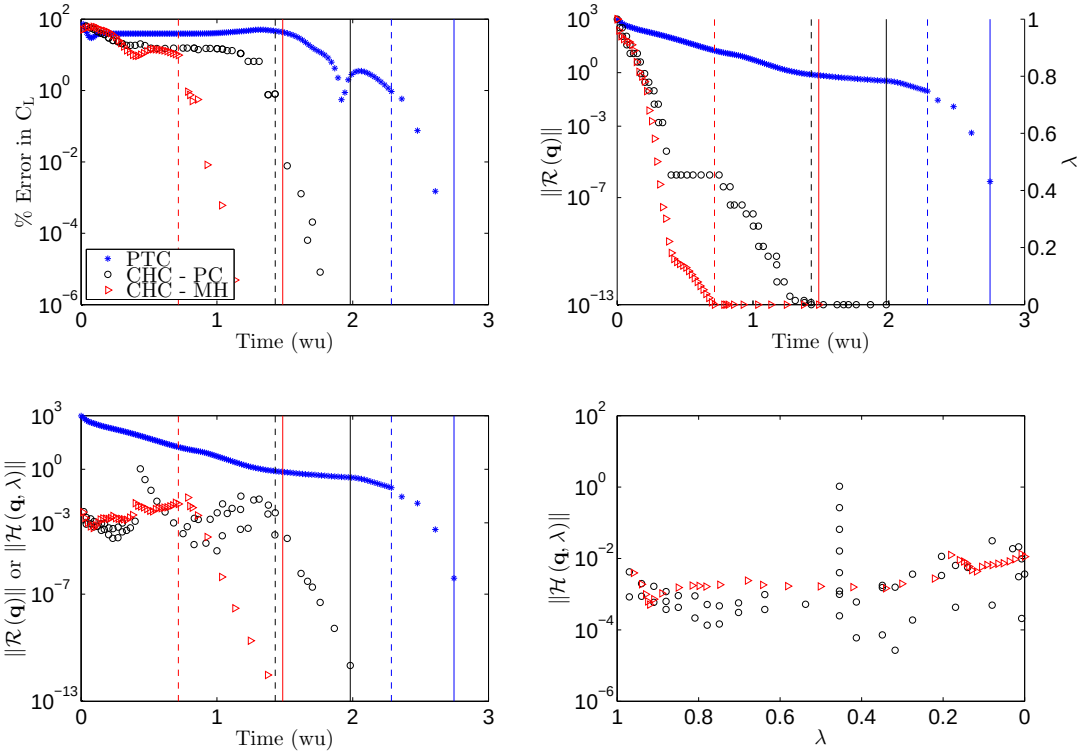


Figure 8.6: Convergence history for different continuation algorithms for turbulent flow over the NACA 0012 airfoil on grid Nt at Mach 0.7 and angle of attack 4°

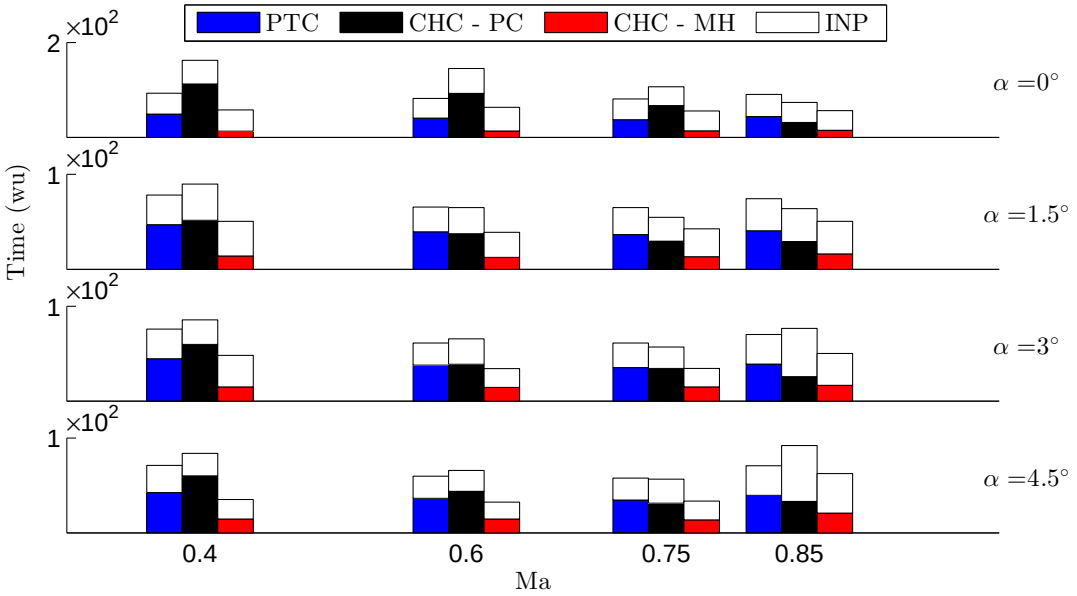


Figure 8.7: Performance comparison of several continuation algorithms for turbulent flows at Reynolds number 1.172×10^7 over the ONERA M6 wing on grid MtHH

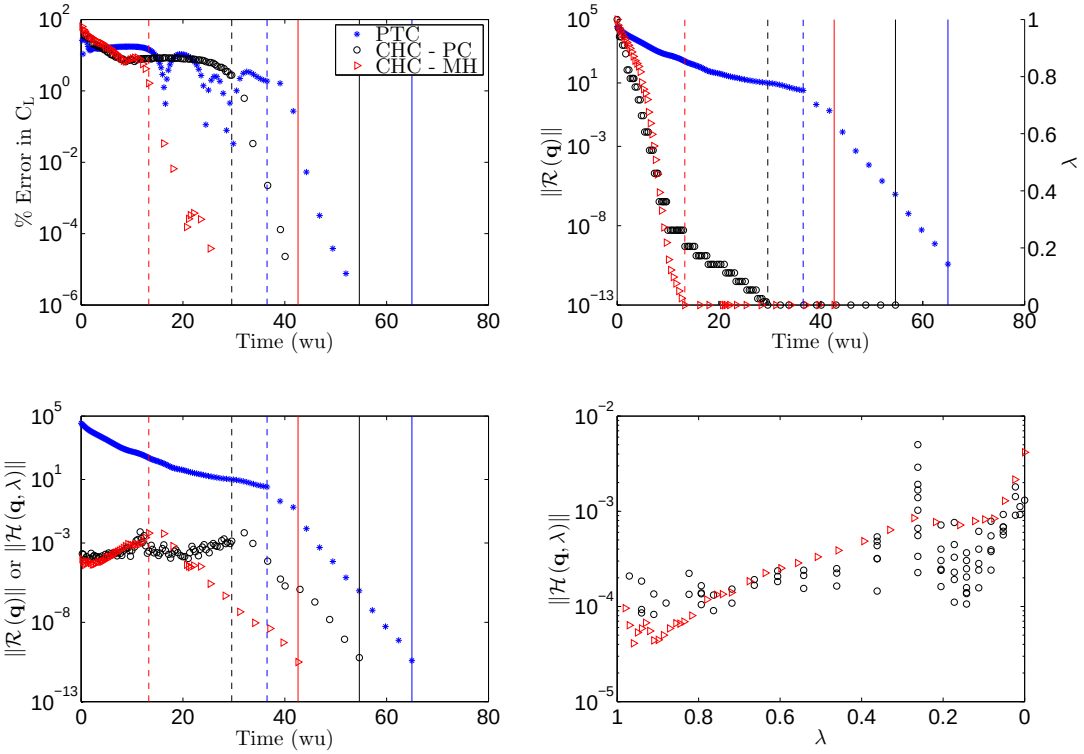


Figure 8.8: Convergence history for different continuation algorithms for turbulent flow over the ONERA M6 wing on grid MtHH at Mach 0.75 and angle of attack 1.5°

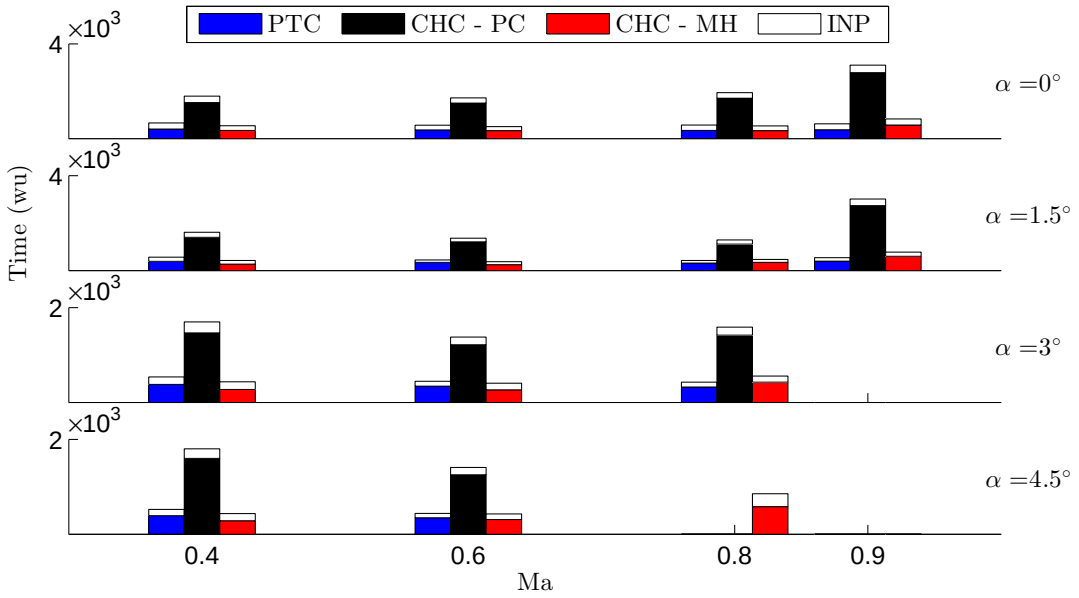


Figure 8.9: Performance comparison of several continuation algorithms for turbulent flows at Reynolds number 1×10^7 over the ONERA M6 wing on grid MtHC

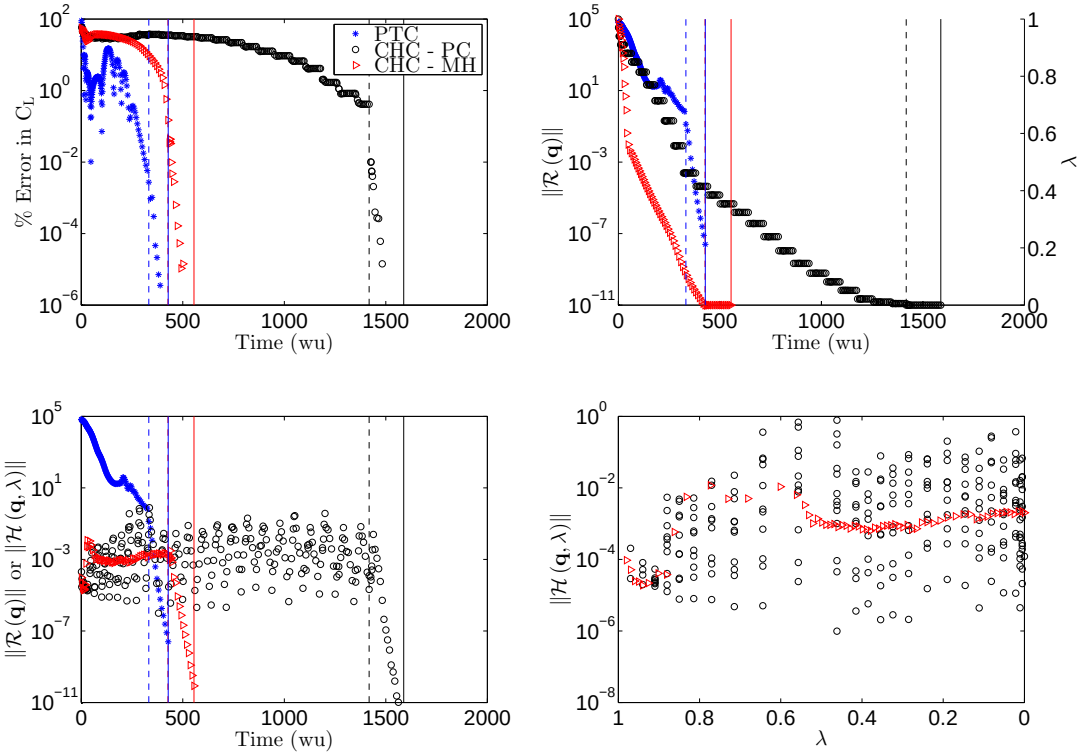


Figure 8.10: Convergence history for different continuation algorithms for turbulent flow over the ONERA M6 wing on grid MtHC at Mach 0.8 and angle of attack 3°

8.6 Summary

The test suites and statistics from the performance studies are summarized in Table 8.1. Except for the two three-dimensional turbulent test suites, the predictor-corrector homotopy method has exhibited improved performance over the pseudo-transient method. For all test suites investigated, the monolithic homotopy continuation method has demonstrated improved performance over the predictor-corrector algorithm. Except for the turbulent test suite on the largest three-dimensional mesh, the monolithic homotopy continuation algorithm has demonstrated improved performance over both the predictor-corrector algorithm and the pseudo-transient method.

The studies presented in this section not only give some quantification of performance but also serve as a parameter study for both PTC and the homotopy methods. The input parameters presented in Table 8.1 provide a guideline for selecting input parameters for the different algorithms for various grids and flow types. It was also found that approximate matrix-vector products could usually be used for the homotopy methods without incurring any significant stability penalty. However, the monolithic algorithm became unstable and unreliable for both of the three-dimensional RANS cases when the approximate matrix-vector products were used. This issue was mostly resolved by switching to the finite-difference matrix-vector products, though stability issues persist for the test suite on the largest RANS grid. For this test suite, the row/column normalization scaling was used in order to prevent the algorithm from becoming unstable. Unfortunately, the MH algorithm usually takes more time to converge when using this scaling.

Chapter 9

Summary, Conclusions, and Future Work

9.1 Thesis Summary

What we have chosen to provide in this section is a chronological account of the thesis. Summarizing the thesis in this way provides insight into why certain research directions were taken. It also emphasizes our original contributions as well as the current state of the art, including areas where further research efforts should be focused.

The first major deviation from the work of Hicken et al. [61] was in the independent treatment of the homotopy system. Since the dissipation operator does not have boundary conditions it is an underdetermined system of equations when treated independently. Augmenting the homotopy system with boundary conditions allowed for the homotopy to be initialized consistently at $\lambda = 1$ and improved the convergence rate of the preconditioned linear solver. Furthermore, this made the homotopy essentially independent of the blocking of the grid. The ability to define the homotopy more consistently on different grids improved our ability to study and analyze the homotopies.

The second important development was the implementation of the predictor-corrector method based on literature sources. This led to many important minor developments, including the development of a method for calculating the tangent vector suitable for large sparse systems, the development of what we have termed μ-scaling, and the numerical investigation of many step-length adaptation strategies. The investigation of step-length adaptation led to the development and study of what we have termed κ-scaling.

More progress was made in the implementation and investigation of a minimum-norm corrector which solves the sub-problems of the predictor-corrector method in a minimum-norm sense. While this was not found to be particularly useful for homotopies which do not feature bifurcations, and as such was not described in the thesis, it led to additional insight into why the κ-scaling is necessary and eventually a method was developed for acquiring suitable values of κ.

Applying the predictor-corrector algorithm to the RANS equations presented new challenges - step-length adaptation was performing poorly because the turbulence variables were dominating the arclength variable. This led to the realization that the step-length adaptation would not perform well unless variable scaling was applied explicitly to the turbulence variable. While the same effect could be achieved

by applying a factor to the homotopy calculations as appropriate, we thought it too complicated and error-prone to take this approach since it would have to be applied to any new methods that were developed in the future as well.

At this point, the algorithm was found to be significantly outperforming the dissipation-based continuation of Hicken and Zingg [61] and was outperforming PTC in most cases that we investigated, with the exception of some three-dimensional RANS cases. These milestones were presented by Brown and Zingg [16, 17].

The next major advancement was in the development of the monolithic homotopy continuation algorithm. The motivation for this work came from experience with the predictor-corrector method: noticing that solving the corrector stages was often inefficient and that the matrix for both the corrector phase and tangent calculation is the same. If the calculations could be combined, there was much potential for efficiency improvement. Since, to our knowledge, no such algorithm yet existed in the field of homotopy continuation, we turned to the field of robotics since we recognized that tracing an implicitly-defined curve is mathematically equivalent to tracing an implicitly-defined trajectory of a dynamical system, a problem that would be of practical interest to control theorists. We successfully developed and implemented this method, as well as a simple step-length adaptation strategy. Our numerical studies performed with this algorithm demonstrated performance improvements relative to the predictor-corrector algorithm which were consistent with our expectations. In addition, from the convergence proof, we were able to develop some notions of stability of the method and discovered that the accuracy of the tangent can have a major impact on stability. The new algorithm, as well as the performance studies, were presented by Brown and Zingg [18, 19].

After the development of the monolithic homotopy continuation algorithm, we realized that it may be possible to construct a stable continuation algorithm which does not require any linear systems to be inverted and hence no matrices would need to be constructed. The ability to construct homotopies without needing to form any matrices was appealing because it meant that we might at some future date be able to develop an algorithm to numerically construct and test homotopy systems and this matrix-free algorithm could be used to study the resulting homotopy.

Though we ultimately did not use the matrix-free method to analyze any new homotopies, we were able to gain some useful insight into the stability of the monolithic homotopy methods. We were not able to complete the stability proof for this matrix-free method - we could show that the corrector portion of the update was stabilizing but we were unable to establish stability if the predictor portion was not equal to the exact tangent vector, which could not be constructed without forming a matrix. We attempted using this algorithm anyways with a backwards-difference estimate of the tangent, which we found was unstable. We investigated the possibility of using a general linear method to stabilize the algorithm but since general linear methods are developed for fixed-point algorithms we could not see how to apply them in the context of curve tracing. Out of all of the methods that we attempted, the only method that we found which could stabilize the matrix-free algorithm, other than simply using a tangent-free version of the algorithm, was to use an explicit filter. What is surprising about this is that the resulting expression resembles a general linear method. It therefore seems that some general linear methods can be applied to the algorithm, though the only way that we are currently aware of for choosing the coefficients is from the explicit filter formulation.

This analysis gave some useful insight into the original monolithic homotopy method as well. We had already seen previously that under-solving the linear system in the tangent calculation could de-

stabilize the algorithm. The importance of accurately forming the tangent was thus reinforced. This is an important consideration for any researchers who will be implementing the monolithic homotopy continuation method in the future: that it may not be a very robust algorithm if it is not expected that the tangent can be approximated accurately.

The final important contribution came near the end of the thesis. It was realized early on that the nature of the homotopy will affect the effectiveness of the continuation algorithm and so it will be important to be able to analyze the properties of the homotopy. We found in particular that some homotopies encounter unidentified problems on some grids. As an example, we found that we could not include our homotopy continuation algorithms in an invited turbulent flow solver session [14] because the homotopy appeared to become non-physical on the provided mesh for reasons that are not currently understood.

An important quantity for profiling homotopies is the curvature. This was a calculation that we had unsuccessfully attempted on two previous occasions. However, after some study of differential geometry and newly deriving the tangent calculation from this perspective, the curvature calculation was relatively straightforward to derive. Unfortunately our inability to estimate the tensor-vector products reliably in double precision ultimately continues to be a problem in performing this calculation unreliably. Regardless, it has allowed us to finally characterize the traceability of some homotopies and gain some insight into the curvature distribution and how it varies with details such as mesh refinement and flow conditions. The derivation of the curvature calculation and some preliminary studies using this tool were subsequently published by Brown and Zingg [20].

Our newfound ability to calculate the curvature associated with homotopies opens up possibilities to construct algorithms using higher order curve derivatives - we are especially interested in attempting to augment the monolithic homotopy continuation algorithm with the second derivative. In addition, curvature is a suitable metric for performing optimization on the homotopy. As an example of how this could be achieved, a homotopy could be constructed by adding two homotopy systems in some ratio and the following optimization problem can be posed: what ratio of the two homotopy systems will give the minimum total curvature? Unfortunately the curvature calculation has only come at the end of the thesis and time does not permit us to explore these interesting ideas. In addition, the curvature will need to be calculated more accurately for an optimization algorithm to be reliable.

9.2 Conclusions

New monolithic homotopy continuation algorithms were developed and applied to a parallel implicit CFD flow solver for inviscid, laminar, and turbulent flows. When using the convex homotopy with the dissipation operator as the homotopy system, the new algorithms were found to be more efficient than the predictor-corrector algorithms prevalent in the literature. The new algorithms also demonstrated good performance relative to the pseudo-transient continuation algorithm common in implicit CFD solution methods.

Some homotopies were found to be unsuitable for continuation for certain grids, flow types, or operating conditions. For example, any cases that were not inviscid and subsonic were very difficult or impossible to solve with the global homotopy. It was also observed that the diagonal operator was less effective for viscous and, especially, turbulent cases due to the increased curvature and less balanced curvature profile. Stability could also become a concern with the monolithic homotopy method for longer

turbulent flow solves. While the algorithm can be stabilized by performing the update more accurately, the corresponding cost increase reduces the competitiveness of the algorithm.

With many studies showing good performance of the algorithm, and many research directions which can potentially lead to further improvements to algorithm efficiency, homotopy continuation strategies show value and potential for continued CFD applications.

9.3 Contributions

In summary, the main contributions of this thesis are

- The development of new homotopy continuation algorithms;
- The development of new tools for application to homotopy continuation algorithms;
- The development of new tools and analysis methodology for the study of homotopies; and
- A quantitative assessment of the performance of classical and newly developed homotopy continuation algorithms for the efficient solution of modern CFD problems.

9.4 Future Work and Recommendations

Additional research effort should be invested in the construction of homotopies for which the continuation algorithms will perform reliably on different grids and flow conditions. Though tools and methodologies have been developed for the numerical analysis of homotopies, what is lacking is a method for designing homotopy systems to give reduced curvature or other desired features. Also, the complete failure of the homotopy continuation algorithms on certain grids is an issue which must be addressed. We regard these research objectives as having highest priority, though they are also very challenging problems.

Additional studies can be performed with the higher derivative calculations. These can be used in continuation algorithms, optimization algorithms, and additional analysis. Additional research effort should also be invested in attempting to improve the numerical accuracy of these calculations.

Some specific recommendations for future research and development are listed and discussed.

- Homotopy continuation algorithm enhancements:
  - The higher derivative calculations could be integrated into the continuation algorithms. We do not consider it a priority to implement higher-order predictors for the predictor-corrector method because it seems unlikely that the inclusion of high-order predictors could improve the efficiency to the point where it is as efficient as the monolithic homotopy method.
  - The monolithic homotopy continuation algorithm could be generalized to include information from higher curve derivatives. We feel that there is potential for such methods to be more efficient and more stable than the single-derivative version presented in this thesis, as long as the higher derivatives can be calculated with sufficient accuracy.
  - We are interested in continuing the study of stability of the monolithic homotopy algorithms. For example, it may be possible to apply the filtration ideas of the matrix-free algorithm to the matrix-present algorithm.

  - Homotopies between two states are not unique. We have already discussed changing the homotopy by changing the homotopy system. Alternatively, the homotopy itself could be constructed differently. As an example, instead of defining the homotopy as the solution to $\mathcal{H}(\mathbf{q}, \lambda) = \mathbf{0}$ we might investigate a homotopy defined as the solution to an ODE such as

$$\lambda(1-\lambda)\mathcal{F}(x)\frac{\partial^2}{\partial\lambda^2}\mathbf{q} + \mathcal{H}(\mathbf{q}, \lambda) = \mathbf{0}$$

    to "smooth out" the deformation. It may however be challenging to design continuation algorithms around such a homotopy.

- High-order curve derivative calculations:
  - Currently we do not know of any method which can reliably estimate the tensor-vector products in double precision. One potential solution which we have not explored is the use of hypercomplex numbers.
  - More efficient ways to build the $\mathbf{w}_n$ vector could be investigated, since the cost can become prohibitive for high curve derivatives.
- Design of homotopy systems:
  - Some criteria could be developed to analytically estimate traceability of homotopies.
  - Combining the dissipation operator with the diagonal operator with some ratio yields a system which has a positive-definite Jacobian. This system is suitable as a homotopy system. Optimization can be used to find the ratio which minimizes some curvature-based metric, such as the maximum curvature or the line integral of the curvature over the length of the curve. Additional homotopy systems can be included as well. At the least, some useful insight may be gained from this study. For example, it may be informative to learn how flow-specific the result of the optimization is.
  - A method could be developed to automatically generate and test homotopy systems. The homotopies could be studied using the matrix-free monolithic homotopy continuation algorithm.
- Applications:
  - The relative performance of the algorithms was not investigated for higher order discretizations.
  - If desired, the homotopy continuation algorithms could be augmented with the capability of solving systems of equations with multiple solutions, unstable solutions, or solutions with singular flow Jacobian.

Appendix A

Supplemental Theorems and Definitions

Theorem A.1. (Implicit Function Theorem)*: Let* $f : \Re^n \times \Re^m \to \Re^n$ *be continuously differentiable, and define some point* $(\mathbf{x}_0, \mathbf{y}_0) \in \Re^n \times \Re^m$ *such that* $f(\mathbf{x}_0, \mathbf{y}_0) = \mathbf{c}$*. Then if the Jacobian* $\nabla_{\mathbf{x}} f(\mathbf{x}_0, \mathbf{y}_0)$ *is invertible then there exists an open set* A *containing* $\mathbf{x}_0$*, an open set* B *containing* $\mathbf{y}_0$*, and a unique continuously differentiable function* $g : B \to A$ *such that*

$$\{(\mathbf{g}(\mathbf{y}), \mathbf{y}) | \mathbf{y} \in B\} = \{(\mathbf{x}, \mathbf{y}) \in A \times B | f(\mathbf{x}, \mathbf{y}) = \mathbf{c}\}.$$

Theorem A.2. *Let* $f : \mathbb{R} \to \mathbb{R}$ *and* $g : \mathbb{R} \to \mathbb{R}$ *be Lipschitz continuous. Let* $x(t)$ *denote the solution to* $\dot{x} = f(x)$ *with* $x(0) = x_0$*; let* $y(t)$ *denote the solution to* $\dot{y} = g(y)$ *with* $y(0) = y_0 \geq x_0$*. Assume that for all* $x \in \mathbb{R}$*,* $f(x) \leq g(x)$*. Then for all* $t > 0$*,* $t \in \mathbb{R}$*,* $x(t) \leq y(t)$*.*

This theorem is given by Hartman [58] as Theorem 4.1 on page 26.

Definition A.1. *A real-valued matrix* $\mathcal{A} \in \mathbb{R}^N \times \mathbb{R}^N$ *is said to be positive definite if*

$$\mathbf{u}^T \mathcal{A} \mathbf{u} > 0$$

for all $\mathbf{u} \in \mathbb{R}^N$ *such that* $\mathbf{u} \neq \mathbf{0}$*. Similarly,* $\mathcal{A}$ *is said to be negative definite if*

$$\mathbf{u}^T \mathcal{A} \mathbf{u} < 0$$

for all $\mathbf{u} \in \mathbb{R}^N$ *such that* $\mathbf{u} \neq \mathbf{0}$*. A matrix which is either positive definite or negative definite is said to be definite.*

A real-valued matrix $\mathcal{A} \in \mathbb{R}^N \times \mathbb{R}^N$ *is said to be positive semi-definite if*

$$\mathbf{u}^T \mathcal{A} \mathbf{u} \geq 0$$

for all $\mathbf{u} \in \mathbb{R}^N$ *such that* $\mathbf{u} \neq \mathbf{0}$*. Similarly,* $\mathcal{A}$ *is said to be negative semi-definite if*

$$\mathbf{u}^T \mathcal{A} \mathbf{u} \leq 0$$

for all $\mathbf{u} \in \mathbb{R}^N$ *such that* $\mathbf{u} \neq \mathbf{0}$*. A matrix which is either positive semi-definite or negative semi-definite is said to be semi-definite.*

The definition of a positive definite matrix is given by Saad [148] as equation (1.47) in Section 1.11.

Definition A.2. *A matrix* $\mathcal{A}$ *is strictly irreducibly row-diagonally dominant if*

$$\sum_{i \neq j} \left|\mathcal{A}_{[i,j]}\right| \leq \left|\mathcal{A}_{[i,i]}\right| \; \forall i, \;\; \sum_{i \neq j} \left|\mathcal{A}_{[i,j]}\right| < \left|\mathcal{A}_{[i,i]}\right| \text{ for some } i,$$

and $\mathcal{A}$ *is irreducible. A matrix is said to be irreducible if its graph is connected, see Saad [148] Section 3.3.4 for more details.*

This definition is given by Saad [148] as Definition 4.5 in Section 4.2.3.

Appendix B

Grid Details

A grid indexing system is included to assist in identifying when two cases are run using the same grid. An intuitive index is used to catalog the grids based on the geometry and flow type, including additional identifiers as appropriate.

The NACA 0012 and ONERA M6 geometries are very common test cases for computational aerodynamics because of the simplicity of the geometry, the availability of experimental wind tunnel data, and the availability of other CFD data. See McCroskey [111] for a summary and assessment of experimental data for the NACA 0012 and Schmitt and Charpin [152] for experimental data for the ONERA M6.

Grid index	Geometry	Topology	Flow Type	Grid Nodes	Blocks	Min. OW Spacing
Ne	NACA 0012	H	2D inviscid	1.539×10^4	18	3.48×10^{-4}
Nt	NACA 0012	H	2D RANS	1.92×10^4	8	7.32×10^{-7}
Me1	ONERA M6	H-C	3D inviscid	1.9208×10^6	32	2.00×10^{-3}
Me2	ONERA M6	H-C	3D inviscid	1.53664×10^7	256	1.09×10^{-3}
MlHH	ONERA M6	H-H	3D laminar	2.111664×10^6	48	2.18×10^{-4}
MlHC	ONERA M6	H-C	3D laminar	1.882384×10^6	16	2.01×10^{-4}
MtHH	ONERA M6	H-H	3D RANS	2.336064×10^6	192	1.17×10^{-6}
MtHC	ONERA M6	H-C	3D RANS	3.6799488×10^7	1024	8.00×10^{-7}

Table B.1: Summary of grids used in this thesis; OW spacing is the off-wall spacing and is measured in chord units

Appendix C

Inversion of a Sparse Matrix with a Dense Row and a Dense Column

A method is derived for solving a linear system of the form

$$\begin{pmatrix} \mathcal{A} & \mathbf{v}_1 \\ \mathbf{v}_2^T & C \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ y_2 \end{pmatrix}, \tag{C.1}$$

where

$$\mathcal{A} \in \mathbb{R}^{N\times N},\ \mathbf{v}_1, \mathbf{v}_2, \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^N,\ C, x_2, y_2 \in \mathbb{R}.$$

It is assumed that $\mathcal{A}$ is sparse and invertible, whereas $\mathbf{v}_1$ and $\mathbf{v}_2$ are dense, making direct inversion of the augmented matrix expensive.

Expanding the first N rows of the linear system (C.1) gives the equation

$$\mathcal{A}\mathbf{x}_1 + x_2\mathbf{v}_1 = \mathbf{y}_1, \tag{C.2}$$

which can alternatively be written as

$$\mathcal{A}\left(\mathbf{x}_1 + x_2\tilde{\mathbf{v}}_1\right) = \mathbf{y}_1, \tag{C.3}$$

where $\tilde{\mathbf{v}}_1 \in \mathbb{R}^N$ is defined implicitly by

$$\mathcal{A}\tilde{\mathbf{v}}_1 = \mathbf{v}_1. \tag{C.4}$$

Let

$$\tilde{\mathbf{x}}_1 = \mathbf{x}_1 + x_2\tilde{\mathbf{v}}_1. \tag{C.5}$$

Then, taking the inner product of both sides with $\tilde{\mathbf{v}}_2$:

$$\mathbf{v}_2 \cdot \tilde{\mathbf{x}}_1 = \mathbf{v}_2 \cdot \left(\mathbf{x}_1 + x_2\tilde{\mathbf{v}}_1\right). \tag{C.6}$$

An additional equation is extracted from the $n+1$st row of the linear system (C.1):

$$\tilde{\mathbf{v}}_2 \cdot \mathbf{x}_1 + Cx_2 = \mathbf{y}_2, \tag{C.7}$$

which is used with equation (C.6) to obtain:

$$\mathbf{v}_2 \cdot \tilde{\mathbf{x}} = y_2 - Cx_2 + x_2 \mathbf{v}_2 \cdot \tilde{\mathbf{v}}_1. \tag{C.8}$$

This can be solved for x_2:

$$x_2 = \frac{\mathbf{v}_2 \cdot \tilde{\mathbf{x}}_1 - y_2}{\mathbf{v}_2 \cdot \tilde{\mathbf{v}}_1 - C}. \tag{C.9}$$

Substituting this back into equation (C.5):

$$\mathbf{x}_1 = \tilde{\mathbf{x}}_1 - x_2 \tilde{\mathbf{v}}_1, \tag{C.10}$$

which completes the derivation. The calculation is summarized in Algorithm C.1.

Algorithm C.1: Efficient method for the inversion of a sparse matrix with a dense row and a dense column; the linear system is assumed to be of the form of equation (C.1)

Solve the linear system $\mathcal{A}\tilde{\mathbf{v}}_1 = \mathbf{v}_1$ for $\tilde{\mathbf{v}}_1$
Solve the linear system $\mathcal{A}\tilde{\mathbf{x}}_1 = \mathbf{y}_1$ for $\tilde{\mathbf{x}}_1$
$x_2 \leftarrow \frac{\mathbf{v}_2 \cdot \tilde{\mathbf{x}}_1 - y_2}{\mathbf{v}_2 \cdot \tilde{\mathbf{v}}_1 - C}$
$\mathbf{x}_1 \leftarrow \tilde{\mathbf{x}}_1 - x_2 \tilde{\mathbf{v}}_1$

Appendix D

Algorithms for Calculating Directional Derivatives of Any Order

Algorithm D.1: First-order accurate nth directional derivative calculation with direction vectors specific to the order n curve derivative calculation; the variable n_t is the order of the tensor and the integer-valued vector $i_v \in \mathbb{Z}^{n_t}$ contains the orders of derivatives as input; for example, $\mathcal{D}^3\mathcal{H}\left(\mathbf{q}, \lambda\right)[\ddot{c}, \dot{c}, \dot{c}]$ is characterized by $n_t = 3$, $i_v = (2, 1, 1)$

Data: n_t, i_v, ϵ_1, $\ldots$, ϵ_{n-1}, $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ldots$, $\overset{(n-1)}{\mathbf{q}}$, λ, $\dot{\lambda}$, $\ldots$, $\overset{(n)}{\lambda}$
Result: $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
for $j = 1 : 2^{n_t}$ **do**
  $n_p \leftarrow 0$
  for $d = 1 : n_t$ **do**
    `/* The following logic ensures that every combination of input vectors is constructed */`
    if $\mod\left(j - 1, 2^d\right) + 1 > 2^{d-1}$ **then**
      $\mathbf{q} \leftarrow \mathbf{q} + (-1)^d \epsilon_{i_v(d)} \overset{(i_v(d))}{\mathbf{q}}$ and $\lambda \leftarrow \lambda + (-1)^d \epsilon_{i_v(d)} \overset{(i_v(d))}{\lambda}$
      $n_p \leftarrow n_p + 1$
    end
  end
  Evaluate $\mathcal{H}$ at the perturbed $\mathbf{q}$ and λ
  if $\mod\left(n_t, 4\right) \leq 1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + (-1)^{n_p} \mathcal{H}$
  else
    $\mathbf{w} \leftarrow \mathbf{w} + (-1)^{n_p+1} \mathcal{H}$
  end
  Reset $\mathbf{q}$ and λ back to their initial values
end
$\mathbf{w} \leftarrow \frac{1}{\epsilon_{v(1)} \cdots \epsilon_{i_v(n_t)}} \mathbf{w}$

Algorithm D.2: First-order accurate nth directional derivative calculation with direction vectors specific to the order n curve derivative calculation in the special case where all direction vectors are the same; the variable n_t is the order of the tensor

Data: n_t, i, ϵ_i, $\mathbf{q}$, $\overset{(i)}{\mathbf{q}}$, λ, $\overset{(i)}{\lambda}$
Result: $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
$C_{\mathcal{H},:} \leftarrow 0$
for $j = 1 : 2^{n_t}$ **do**
  $n_p \leftarrow 0$
  $C_\epsilon \leftarrow 0$
  /* Get the coefficients in front of ϵ_i and $\mathcal{H}$, denoted C_ϵ and $C_{\mathcal{H},:}$ respectively, for this term */
  for $d = 1 : n_t$ **do**
    if $\mod\left(j-1, 2^d\right) + 1 > 2^{d-1}$ **then**
      $C_\epsilon \leftarrow C_\epsilon + (-1)^{n_t+d}$
      $n_p \leftarrow n_p + 1$
    end
  end
  $k \leftarrow C_\epsilon + n_t + 1$
  $C_{\mathcal{H},k} \leftarrow C_{\mathcal{H},k} + (-1)^{n_p+1}$
end
for $k = 1 : 2n_t + 1$ **do**
  if $C_{\mathcal{H},k} \neq 0$ **then**
    $C_\epsilon \leftarrow k - n_t - 1$
    $\mathbf{q} \leftarrow \mathbf{q} + C_\epsilon \epsilon_i \overset{(i)}{\mathbf{q}}$ and $\lambda \leftarrow \lambda + C_\epsilon \epsilon_i \overset{(i)}{\lambda}$
    Evaluate $\mathcal{H}$ at the perturbed $\mathbf{q}$ and λ
    $\mathbf{w} \leftarrow \mathbf{w} + C_{\mathcal{H},k}\mathcal{H}$
    Reset $\mathbf{q}$ and λ back to their initial values
  end
end
$\mathbf{w} \leftarrow \frac{1}{\epsilon_i^{n_t}}\mathbf{w}$

Algorithm D.3: Second-order accurate nth directional derivative calculation with direction vectors specific to the order n curve derivative calculation; the variable n_t is the order of the tensor and the integer-valued vector $i_v \in \mathbb{Z}^{n_t}$ contains the orders of derivatives as input; for example, $\mathcal{D}^3\mathcal{H}(\mathbf{q}, \lambda)[\ddot{c}, \dot{c}, \dot{c}]$ is characterized by $n_t = 3$, $i_v = (2, 1, 1)$

Data: n_t, i_v, ϵ_1, ..., ϵ_{n-1}, $\mathbf{q}$, $\dot{\mathbf{q}}$, ..., $\overset{(n-1)}{\mathbf{q}}$, λ, $\dot{\lambda}$, ..., $\overset{(n)}{\lambda}$
Result: $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
for $j = 1 : 2^{n_t}$ **do**
  $C_i \leftarrow 1$
  for $d = 1 : n_t$ **do**
    /* The following logic ensures that every combination of input vectors is constructed */
    if $\mod\left(j - 1, 2^d\right) + 1 > 2^{d-1}$ **then**
      $C_\epsilon \leftarrow -1$
      $C_i \leftarrow -C_i$
    else
      $C_\epsilon \leftarrow 1$
    end
    $\mathbf{q} \leftarrow \mathbf{q} + C_\epsilon \epsilon_{i_v(d)} \overset{(i_v(d))}{\mathbf{q}}$ and $\lambda \leftarrow \lambda + C_\epsilon \epsilon_{i_v(d)} \overset{(i_v(d))}{\lambda}$
  end
  Evaluate $\mathcal{H}$ at the perturbed $\mathbf{q}$ and λ
  $\mathbf{w} \leftarrow \mathbf{w} + C_i \mathcal{H}$
  Reset $\mathbf{q}$ and λ back to their initial values
end
$\mathbf{w} \leftarrow \frac{1}{2^{n_t}\left(\epsilon_{v(1)} \cdots \epsilon_{i_v(n_t)}\right)} \mathbf{w}$

Algorithm D.4: Second-order accurate nth directional derivative calculation with direction vectors specific to the order n curve derivative calculation in the special case where all direction vectors are the same; the variable n_t is the order of the tensor

Data: n_t, i, ϵ_i, $\mathbf{q}$, $\overset{(i)}{\mathbf{q}}$, λ, $\overset{(i)}{\lambda}$
Result: $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
$C_{\mathcal{H},:} \leftarrow 0$
for $j = 1 : 2^{n_t}$ **do**
  $C_\epsilon \leftarrow 0$
  $C_i \leftarrow 1$
  /* Get the coefficients in front of ϵ_i and $\mathcal{H}$, denoted C_ϵ and $C_{\mathcal{H},:}$ respectively, for this term */
  for $d = 1 : n_t$ **do**
    if $\mod\left(j-1, 2^d\right) + 1 > 2^{d-1}$ **then**
      $C_\epsilon \leftarrow C_\epsilon - 1$
      $C_i \leftarrow -C_i$
    else
      $C_\epsilon \leftarrow C_\epsilon + 1$
    end
  end
  $k \leftarrow C_\epsilon + n_t + 1$
  $C_{\mathcal{H},k} \leftarrow C_{\mathcal{H},k} + C_i$
end
for $k = 1 : 2n_t + 1$ **do**
  if $C_{\mathcal{H},k} \neq 0$ **then**
    $C_\epsilon \leftarrow k - n_t - 1$
    $\mathbf{q} \leftarrow \mathbf{q} + C_\epsilon \epsilon_i \overset{(i)}{\mathbf{q}}$ and $\lambda \leftarrow \lambda + C_\epsilon \epsilon_i \overset{(i)}{\lambda}$
    Evaluate $\mathcal{H}$ at the perturbed $\mathbf{q}$ and λ
    $\mathbf{w} \leftarrow \mathbf{w} + C_{\mathcal{H},k}\mathcal{H}$
    Reset $\mathbf{q}$ and λ back to their initial values
  end
end
$\mathbf{w} \leftarrow \frac{1}{2^{n_t}\epsilon_i^{n_t}}\mathbf{w}$

Appendix E

Derivation of Equation (5.11)

Assume that the $k-1$st sub-problem has been solved to some suitable tolerance in p_{k-1} iterations. Rearranging the homotopy residual at the p_kth step:

$$\begin{aligned}\mathcal{H}\left(\mathbf{q}_{k-1}^{(p_{k-1})}, \lambda_{k-1}^{(p_{k-1})}\right) =& \mathcal{R}\left(\mathbf{q}_{k-1}^{(p_{k-1})}\right) + \lambda_{k-1}^{(p_{k-1})} \frac{\partial}{\partial \lambda_{k-1}^{(p_{k-1})}} \mathcal{H}\left(\mathbf{q}_{k-1}^{(p_{k-1})}, \lambda_{k-1}^{(p_{k-1})}\right) \approx \mathbf{0} \\ \Rightarrow \mathcal{R}\left(\mathbf{q}_{k-1}^{(p_{k-1})}\right) \approx& -\lambda_{k-1}^{(p_{k-1})} \frac{\partial}{\partial \lambda_{k-1}^{(p_{k-1})}} \mathcal{H}\left(\mathbf{q}_{k-1}^{(p_{k-1})}, \lambda_{k-1}^{(p_{k-1})}\right). \end{aligned} \tag{E.1}$$

For analysis, an embedding step $\mathbf{q}_k^{(0)} \leftarrow \mathbf{q}_{k-1}^{(p_{k-1})}$, $\lambda_k^{(0)} \leftarrow \lambda_{k-1}^{(p_{k-1})} + \Delta\lambda$ is applied from this point. For convex or global homotopy, the first corrector iteration following this embedding step is given by:

$$\mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k^{(0)}\right) = \mathcal{R}\left(\mathbf{q}_k^{(0)}\right) + \lambda_k^{(0)} \frac{\partial}{\partial \lambda_k^{(0)}} \mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k^{(0)}\right). \tag{E.2}$$

Using the fact that $\mathbf{q}_{k-1}^{(0)} = \mathbf{q}_k^{(p_{k-1})}$ to substitute equation (E.1) into equation (E.2):

$$\begin{aligned}\mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k^{(0)}\right) =& -\lambda_{k-1}^{(p_{k-1})} \frac{\partial}{\partial \lambda_{k-1}^{(p_{k-1})}} \mathcal{H}\left(\mathbf{q}_{k-1}^{(p_{k-1})}, \lambda_{k-1}^{(p_{k-1})}\right) + \lambda_k^{(0)} \frac{\partial \mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k^{(0)}\right)}{\partial \lambda_k^{(0)}} \\ =& \Delta\lambda_k \frac{\partial}{\partial \lambda_k^{(0)}} \mathcal{H}\left(\mathbf{q}_k^{(0)}, \lambda_k^{(0)}\right). \end{aligned} \tag{E.3}$$

From this point forward, the superscript (0) and the subscript k will be dropped in the interest of clarity.

Consider now the following linear system of equations:

$$\begin{aligned}\nabla\mathcal{H}\left(\mathbf{q}, \lambda\right)\mathbf{z} =& \frac{\partial}{\partial \lambda}\mathcal{H}\left(\mathbf{q}, \lambda\right) = \frac{1}{\Delta\lambda}\mathcal{H}\left(\mathbf{q}, \lambda\right) \\ \Rightarrow& \nabla\mathcal{H}\left(\mathbf{q}, \lambda\right)\Delta\lambda\mathbf{z} = \mathcal{H}\left(\mathbf{q}, \lambda\right). \end{aligned} \tag{E.4}$$

Equation (E.4) indicates that $\Delta\lambda\mathbf{z}$ is equal to $\Delta\mathbf{q}$, the solution to the linear system resulting from the first Newton iteration for a constant-λ corrector.

Let us now consider the direction of the line which minimizes the distance between a current corrector iterate and the homotopy curve. This vector can be estimated by treating λ as a variable and applying

a Newton update to the augmented system in the minimum-norm sense. A general Newton update for the augmented system is given by

$$\nabla\mathcal{H}\left(\mathbf{u}^{(n)}\right)\mathbf{z}^{(n)} = -\mathcal{H}\left(\mathbf{u}^{(n)}\right),\ e_i\mathbf{z}^{(n)} = 0, \tag{E.5}$$

where $e_i\mathbf{z} = \mathbf{z}[i]$. The minimum-norm Newton update is then obtained by removing the component of $\mathbf{z}^{(n)}$ that lies parallel to the tangent curve:

$$\Delta\mathbf{u}^{(n)} = \mathbf{z}^{(n)} - t\left(\nabla\mathcal{H}\left(\mathbf{u}^{(n)}\right)\right)\cdot\mathbf{z}^{(n)}t\left(\nabla\mathcal{H}\left(\mathbf{u}^{(n)}\right)\right). \tag{E.6}$$

Furthermore, choosing $i = N+1$ for e_i, the system (E.5) reduces to:

$$\nabla_{\mathbf{q}}\mathcal{H}\left(\mathbf{q}^{(n)}, \lambda_k^{(n)}\right)\mathbf{z}^{(n)} = -\mathcal{H}\left(\mathbf{q}^{(n)}, \lambda_k^{(n)}\right). \tag{E.7}$$

The direction of the distance-minimizing curve is thus estimated by solving equation (E.7) for $\mathbf{z}^{(n)}$ and substituting this vector into equation (E.6). The output vector $\Delta\mathbf{u}^{(n)}$ should then be normalized

Considering the algebraic method presented for the tangent calculation (see Section 5.3), the tangent vector is given by:

$$t = \frac{1}{\sqrt{\|\Delta\mathbf{q}\|^2 + \Delta\lambda^2}}\left[\Delta\mathbf{q}; \Delta\lambda\right]. \tag{E.8}$$

Substituting the expression for the tangent vector (E.8) into equation (E.6) and performing some algebra gives the modified corrector step:

$$\Delta\mathbf{q}^{(c)} = \Delta\mathbf{q}\left(\frac{\Delta\lambda^2}{\|\Delta\mathbf{q}\|^2 + \Delta\lambda^2}\right),\ \Delta\lambda^{(c)} = -\Delta\lambda\frac{\|\Delta\mathbf{q}\|^2}{\|\Delta\mathbf{q}\|^2 + \Delta\lambda^2}. \tag{E.9}$$

Defining the full corrector vector as $\mathbf{u}^{(c)} = \left[\mathbf{q}^{(c)}, \lambda^{(c)}\right]$, the L^2-norm is given by:

$$\left\|\Delta\mathbf{u}^{(c)}\right\| \equiv \sqrt{\left\|\Delta\mathbf{q}^{(c)}\right\|^2 + \Delta\lambda^2} = \frac{|\Delta\lambda|\,\|\Delta\mathbf{q}\|}{\sqrt{\|\Delta\mathbf{q}\|^2 + \Delta\lambda^2}}. \tag{E.10}$$

The vector $\Delta\mathbf{u}^{(c)}$ is normalized to calculate the unit vector $\Delta\hat{\mathbf{u}}^{(c)}$, though only the first N terms are needed (i.e. the part relating to the flow variables):

$$\Delta\hat{\mathbf{u}}^{(c)}_{[1:N]} \equiv \frac{1}{\left\|\Delta\mathbf{u}^{(c)}\right\|}\Delta\mathbf{q}^{(c)} = \Delta\mathbf{q}\left(\frac{\Delta\lambda}{\sqrt{\|\Delta\mathbf{q}\|^2 + \Delta\lambda^2}\,\|\Delta\mathbf{q}\|}\right). \tag{E.11}$$

Finally, the angle θ between $\Delta\mathbf{q}$ and $\Delta\mathbf{q}^{(c)}$ is given by

$$\cos\theta \equiv \Delta\hat{\mathbf{u}}\cdot\Delta\hat{\mathbf{u}}^{(c)} = \Delta\hat{\mathbf{q}}\cdot\Delta\hat{\mathbf{u}}^{(c)}_{[1:N]}, \tag{E.12}$$

which simplifies to

$$\Delta\lambda = \pm\,\|\Delta\mathbf{q}\|\cot\theta. \tag{E.13}$$

References

[1] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 1990.

[2] E. L. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, 2:1–64, 1993.

[3] 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(2):391–408, 1996.

[4] G. E. Andrews. *The Theory of Partitions*. Cambridge University Press, 1984.

[5] T. J. Baker, A. Jameson, and W. Schmidt. A family of fast and robust Euler codes. In K. C. Reddy and J. S. Steinhoff, editors, *Proceedings of Workshop on Computational Fluid Dynamics*. 1984. Princeton University, MAE 1652.

[6] B. S. Baldwin and H. Lomax. Thin layer approximation and algebraic model for separated turbulent flows. June 1978. AIAA 78-257.

[7] T. J. Barth and S. W. Linton. Unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. January 1995. AIAA 95-0221.

[8] F. Bassi and S. Rebay. A high-order discontinuous finite element method for the numerical solution of te compressible Navier-Stokes equations. *Journal of Computational Physics*, 131:267–279, 1997.

[9] R. M. Beam and R. F. Warming. An implicit finite-difference algorithm for hyperbolic systems in conservation-law form. *Journal of Computational Physics*, 22(1):87–110, 1976.

[10] S. Bernstein. Sur la généralisation du problème de Dirichlet. *Mathematische Annalen*, 69:82–136, 1910.

[11] R. N. Borkovsky, U. Doraszelski, and Y. Kryukov. A user's guide to solving dynamic stochastic games using the homotopy method. *Operations Research*, 58(4):1116–1132, 2010.

[12] A. Brandt. *Multigrid Methods*. Springer, 1982.

[13] F. H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Developement*, 16:504–522, 1972.

[14] D. A. Brown, H. P. Buckley, M. Osusky, and D. W. Zingg. Performance of a Newton-Krylov-Schur algorithm for the numerical solution of the steady Reynolds-Averaged Navier-Stokes equations. In *53rd AIAA Aerospace Sciences Meeting*, Kissimee, Florida, United States, January 2015. AIAA 2015-1744.

[15] D. A. Brown and D. W. Zingg. Performance of a Newton-Krylov-Schur algorithm for the numerical solution of the steady Reynolds-averaged Navier-Stokes equations. *AIAA Journal*. In press.

[16] D. A. Brown and D. W. Zingg. Advances in homotopy-based globalization strategies in computational fluid dynamics. June 2013. AIAA-2013-2944.

[17] D. A. Brown and D. W. Zingg. Development of homotopy continuation methods in CFD. In *21st*

Annual Conference of the CFD Society of Canada, Sherbrooke, Quebec, Canada, May 2013.

[18] D. A. Brown and D. W. Zingg. Development of a monolithic homotopy continuation algorithm for CFD applications. In *22nd Annual Conference of the CFD Society of Canada*, Toronto, Ontario, Canada, June 2014.

[19] D. A. Brown and D. W. Zingg. A new monolithic homotopy continuation algorithm with CFD applications. In *Eighth International Conference on Computational Fluid Dynamics*, Chengdu, China, July 2014.

[20] D. A. Brown and D. W. Zingg. Efficient high order differentiation of implicitly-defined curves with applications to homotopy continuation algorithms for CFD flow solvers. In *23rd Annual Conference of the CFD Society of Canada*, Waterloo, Ontario, Canada, June 2015.

[21] G. F. Carey and R. Krishnan. Continuation techniques for a penalty approximation of the Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 48:265–282, 1985.

[22] M. H. Carpenter, D. Gottlieb, and S. Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, 1994.

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

[24] T. T. Chisholm and D. W. Zingg. A Newton-Krylov algorithm for turbulent aerodynamic flows. January 2003. AIAA 2003-0071.

[25] T. T. Chisholm and D. W. Zingg. A Jacobian-free Newton-Krylov algorithm for compressible turbulent fluid flows. *Journal of Computational Physics*, 228:3490–3507, 2009.

[26] B. Cochelin. A path-following technique via an asymptotic-numerical method. *Computers & Structures*, 53(3):1181–1192, 1994.

[27] A. Dagan. A convergence accelerator for a linear system of equations based on the power method. *International Journal for Numerical Methods in Fluids*, 35:721–741, 2001.

[28] E. de Sturler. Truncation strategies for Krylov subspace methods. *SIAM Journal of Numerical Analysis*, 36:864–889, 1999.

[29] D. C. Del Rey Fernández, P. D. Boom, and D. W. Zingg. A generalized framework for nodal first derivative summation-by-parts operators. *Journal of Computational Physics*, 266:214–239, 2004.

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

[31] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[32] C. Den Heijer and W. C. Rheinboldt. On steplength algorithms for a class of continuation methods. *SIAM Journal on Numerical Analysis*, 18(5):925–948, 1981.

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

[34] S. Dias and D. W. Zingg. A high-order parallel Newton-Krylov flow solver for the Euler equations. June 2009. AIAA-2009-3657.

[35] DLR Germany. Taubench version 1.1, IPACS. `http://www.ipacs-benchmark.org`. Accessed: 2014-09-20.

[36] R. Doallo, B. B. Fraguela, J. Touriño, and E. L. Zapata. Parallel sparse modified Gram-Schmidt

QR decomposition. In *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, HPCN Europe 1996, pages 646–653, London, UK, 1996. Springer-Verlag.

[37] B. C. Eaves. Homotopies for computation of fixed points. *Mathematical Programming*, 3:1–22, 1972.

[38] B. C. Eaves. A short course in solving equations with PL homotopies. In R. W. Cottle and C. E. Lemke, editors, *Nonlinear Programming: SIAM-AMS Proceedings*, volume 9, pages 73–143. 1976.

[39] B. C. Eaves and H. Scarf. The solution of systems of piecewise linear equations. *Mathematics of Operations Research*, 1(1):1–27, 1976.

[40] S. C. Eisenstat and H. F. Walker. Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, 4:393–422, 1994.

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

[42] L. Eriksson and A. Rizzi. Analysis by computer of the convergence to steady state of discrete approximations to the Euler equations. 1983. AIAA 83-1951.

[43] S. Eyi. Convergence acceleration using convergence error estimation. 2015. AIAA 2015-2751.

[44] D. Funaro and D. Gottlieb. A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations. *Mathematics of Computation*, 51:599–613, 1988.

[45] J. Gatsis. *Preconditioning Techniques for a Newton-Krylov Algorithm for the Compressible Navier-Stokes Equations*. PhD thesis, University of Toronto, Toronto, Ontario, Canada, 2014.

[46] W. M. Gentleman. Least squares computations by Givens transformations without square roots. *Journal of the Institute of Mathematics and its Applications*, 12:329–336, 1973.

[47] K. Georg. A note on stepsize control for numerical curve following. In B. C. Eaves, F. J. Gould, H.-O. Peitgen, and M. J. Todd, editors, *Homotopy Methods and Global Convergence*, pages 145–154. Plenum Press, New York, 1983.

[48] N. H. Getz. *Dynamic Inversion of Nonlinear Maps with Applications to Nonlinear Control and Robotics*. PhD thesis, University of California at Berkeley, Berkeley, California, USA, 1995.

[49] N. H. Getz and J. E. Marsden. A dynamic inverse for nonlinear maps. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 4, pages 4218–4223, December 1995.

[50] N. H. Getz and J. E. Marsden. Tracking implicit trajectories. In *IFAC Symposium on Nonlinear Control Systems Design*, Tahoe City, June 1995.

[51] J. W. Givens. Computation of plane unitary rotations transforming a general matrix to a triangular form. *Journal of the Society for Industrial and Applied Mathematics*, 6:26–50, 1958.

[52] P. Godin, D. W. Zingg, and T. E. Nelson. High-lift aerodynamic computations with one- and two-equation turbulence models. *AIAA Journal*, 35(2):237–243, 1997.

[53] G. H. Golub and D. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948-1976. *SIAM Review*, 31(1):50–102, 1989.

[54] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[55] M. Hafez, E. Parlette, and M. Salas. Convergence acceleration of iterative solutions of Euler equations for transonic flow computations. *Computational Mechanics*, 1:165–176, 1986.

[56] W. Hao, J. D. Hauenstein, C.-W. Shu, A. J. Sommese, Z. Xu, and Y.-T. Zhang. A homotopy method based on WENO schemes for solving steady state problems of hyperbolic conservation

laws. *Journal of Computational Physics*, 250:332–346, 2013.

[57] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly high order essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, 1987.

[58] P. Hartman. *Ordinary Differential Equations*. Birkhauser, Berlin, second edition, 1982.

[59] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[60] J. E. Hicken. *Efficient Algorithms for Aircraft Design: Contributions to Aerodynamic Shape Optimization*. PhD thesis, University of Toronto, Toronto, Ontario, Canada, 2009.

[61] J. E. Hicken, H. Buckley, M. Osusky, and D. W. Zingg. Dissipation-based continuation: a globalization for inexact-Newton solvers. June 2011. AIAA 2011-3237.

[62] J. E. Hicken, M. Osusky, and D. W. Zingg. Comparison of parallel preconditioners for a Newton-Krylov flow solver. In *6th International Conference on Computational Fluid Dynamics*, St. Petersburg, Russia, July 2010.

[63] J. E. Hicken and D. W. Zingg. A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. *AIAA Journal*, 46(11):2773–2786, 2008.

[64] J. E. Hicken and D. W. Zingg. Globalization strategies for inexact-Newton solvers. June 2009. AIAA-2009-4139.

[65] J. E. Hicken and D. W. Zingg. Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement. *AIAA Journal*, 48(2):401–413, 2010.

[66] J. E. Hicken and D. W. Zingg. A simplified and flexible variant of GCROT for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 32(3):1672–1694, 2010.

[67] M. W. Hirsch. *Differential Topology*. Springer-Verlag, Berlin, Heidelberg, New York, 1976.

[68] N. A. L. Holt. High-fidelity aerodynamic shape optimization with high-order spatial discretization. Master's thesis, University of Toronto, Toronto, Ontario, Canada, 2014.

[69] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the Association for Computing Machinery*, 5:339–342, 1958.

[70] H. T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. June 2007. AIAA 2007-4079.

[71] Z. Jackiewicz. *General Linear Methods for Ordinary Differential Equations*. John Wiley & Sons, 2009.

[72] A. Jameson. A solution of Euler equations for two dimensional transonic flow by multigrid method. *Applied Mathematics and Computation*, 13:327–356, 1983.

[73] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite-volume methods using Runge-Kutta time-stepping schemes. June 1981. AIAA-1981-1259.

[74] A. Jameson, J. C. Vassberg, and K. Ou. Further studies of airfoils supporting non-unique solutions in transonic flow. *AIAA Journal*, 50(12):2865–2881, 2012.

[75] A. Jameson and S. Yoon. Multigrid solution of the Euler equations using implicit schemes. January 1985. AIAA-85-0293.

[76] D. C. Jespersen and P. G. Buning. Accelerating and iterative process by explicit annihilation. *SIAM Journal on Scientific and Statistical Computing*, 6(3):639–651, 1985.

[77] H. B. Keller. Global homotopies and Newton methods. In C. de Boor and G. H. Golub, editors, *Recent Advances in Numerical Analysis*, pages 73–94. Academic Press, New YorK, 1978.

[78] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM*

Journal on Numerical Analysis, 35(2):508–523, 1998.

[79] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. *AIAA Journal*, 52:935–951, 2014.

[80] F. Klein. Neue beiträge zur Riemannschen Funktionentheorie. *Mathematische Annalen*, 21, 1882-1883.

[81] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.

[82] D. A. Knoll and P. R. McHugh. Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow. *SIAM Journal on Scientific Computing*, 19(1):291–301, 1998.

[83] H.-O. Kreiss and G. Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations: proceedings of a symposium conducted by the Mathematics Research Center, the University of Wisconsin*, pages 195–212. Mathematics Research Centre, the University of Wisconsin, Academic Press, 1974.

[84] E. Kreyszig. *Differential Geometry*. University of Toronto Press, Toronto, Ontario, Canada, 1959.

[85] E. Lahaye. Une méthode de résolution d'une categorie d'équations transcendantes. *Comptes Rendus Hebdomataires Séances de l'Académie de Sciences*, 198:1840–1842, 1934.

[86] E. Lahaye. Sur la résolution des systèmes d'équations transcendantes. *Académie Royale de Belgique. Bulletin de la Classe de Sciences*, 5:805–822, 1948.

[87] H. Lahmam, J. M. Cadou, H. Zahrouni, N. Damil, and M. Potier-Ferry. High-order predictor-corrector algorithms. *International Journal for Numerical Methods in Engineering*, 55:685–704, 2002.

[88] K.-L. Lai and J. L. Crassidis. Extensions of the first and second complex-step derivative approximations. *Journal of Computational and Applied Mathematics*, 219:276–293, 2008.

[89] J. V. Lassaline and D. W. Zingg. Development of an agglomeration multigrid algorithm with directional coarsening. June 1999. AIAA 99-3338.

[90] P. Lax and B. Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13:217–237, 1960.

[91] C. Lee, D. Koo, K. Telidetzki, H. Buckley, H. Gagnon, and D. W. Zingg. Aerodynamic shape optimization of benchmark problems using jetstream. January 2015. AIAA 2015-0262.

[92] J. Lee and H.-D. Chiang. Constructive homotopy methods for finding all or multiple DC operating points of nonlinear circuits and systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(1):35–50, 2001.

[93] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103:16–42, 1992.

[94] C. E. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11(17):681–689, 1965.

[95] C. E. Lemke and J. T. Howson. Equilibrium points of bimatrix games. *SIAM Journal on Applied Mathematics*, 12(2):413–423, 1964.

[96] J. Leray and J. Schauder. Topologies et équations fonctionelles. *Annales Scientifiques de l'École Normale Supérieure*, 51:45–78, 1934.

[97] Y. Liu, M. Vinokur, and Z. J. Wang. Discontinuous spectral difference method for conservation

laws on unstructured grids. *Journal of Computational Physics*, 216:780–801, 2006.

[98] C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen, L. J. Dursi, J. Chong, S. Northrup, J. Pinto, N. Knecht, and R. van Zon. SciNet: Lessons learned from building a power-efficient top-20 system and data centre. *Journal of Physics: Conference Series*, 256, 2010.

[99] H. Lomax, T. H. Pulliam, and D. W. Zingg. *Fundamentals of Computational Fluid Dynamics*. Springer-Verlag, 2001.

[100] B. N. Lundberg and A. B. Poore. Variable order Adams-Bashforth predictors with an error stepsize control for continuation methods. *SIAM Journal on Scientific and Statistical Computing*, 12(3):695–723, 1991.

[101] J. N. Lyness. Numerical algorithms based on the theory of complex variable. In *Proceedings of the ACM National Meeting*, New York, United States, 1967.

[102] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM Journal on Numerical Analysis*, 4(2):202–210, 1967.

[103] R. W. MacCormack. The effect of viscosity in hypervelocity impact cratering. 1969. AIAA 69-354.

[104] W. Mackens. Numerical differentiation of implicitly defined space curves. *Computing*, 41:237–260, 1989.

[105] L. Martinelli, A. Jameson, and F. Grasso. A multigrid method for the Navier-Stokes equations. 1986. AIAA 86-0208.

[106] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523 – 530, 2004.

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

[108] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003.

[109] D. J. Mavriplis. Multigrid solution of the two-dimensional Euler equations on unstructured triangular meshes. *AIAA Journal*, 26(7):824–831, 1988.

[110] D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145(1):141–165, 1998.

[111] W. J. McCroskey. A critical assessment of wind tunnel results for the NACA 0012 airfoil. Technical report, Ames Research Center, Moffett Field, California, October 1987. NASA TM 100019.

[112] F. R. Menter. Zonal two-equation k-ω models for aerodynamic flows. July 1993. AIAA 93-2906.

[113] J. W. Milnor. *Topology from the Differentiable Viewpoint*. University Press of Virginia, Charlottesville, Virginia, United States, 1969.

[114] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[115] A. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[116] W. A. Mulder and B. van Leer. Experiments with explicit upwind methods for the Euler equations. *Journal of Computational Physics*, 59:232–246, 1985.

[117] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[118] T. E. Nelson and D. W. Zingg. Fifty years of aerodynamics: successes, challenges, and opportuni-

ties. *Canadian Aeronautics and Space Journal*, 50(1):61–84, 2004.

[119] M. Nemec, M. J. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. January 2008. AIAA-2008-725.

[120] M. Nemec and D. W. Zingg. Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations. *AIAA Journal*, 40(6):1146–1154, 2002.

[121] J. C. Newman III, D. L. Whitfield, and W. K. Anderson. Step-size independent approach for multidisciplinary sensitivity analysis. *Journal of Aircraft*, 40(3):566–573, 2003.

[122] R.-H. Ni. A multiple grid scheme for solving the Euler equations. 1981. AIAA 81-1025.

[123] J. C. Nichols and D. W. Zingg. A three-dimensional multi-block Newton-Krylov flow solver for the Euler equations. June 2005. AIAA-2005-5230.

[124] E. J. Nielsen, W. K. Anderson, R. W. Walters, and D. E. Keyes. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. June 1995. AIAA-95-1733.

[125] J. Nordström and M. H. Carpenter. Boundary and interface conditions for high-order finite-difference methods applied to the Euler and Navier-Stokes equations. *Journal of Computational Physics*, 148:621–645, 1999.

[126] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM, 1970.

[127] Tz. Ostromsky, P. C. Hansen, and Z. Zlatev. A parallel sparse QR-factorization algorithm. In J. Dongarra, K. Madsen, and J. Waśniewski, editors, *Applied Parallel Computing*, pages 462–472. Springer, Berlin, 1995.

[128] M. Osusky. *A Parallel Newton-Krylov-Schur Algorithm for the Reynold-Averaged Navier-Stokes Equations*. PhD thesis, University of Toronto, Toronto, Ontario, Canada, 2013.

[129] M. Osusky, P. D. Boom, and D. W. Zingg. Results from the Fifth AIAA Drag Prediction Workshop obtained with a parallel Newton-Krylov flow solver discretized using summation-by-parts operators. June 2013. AIAA-2013-2511.

[130] M. Osusky and D. W. Zingg. A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations discretized using summation-by-parts operators. *AIAA Journal*, 51(12):2833–2851, 2013.

[131] M. Osusky and D. W. Zingg. Steady three-dimensional turbulent flow computations with a parallel Newton-Krylov-Schur algorithm. January 2014. AIAA-2014-0242.

[132] K. Ou, A. Jameson, and J. C. Vassberg. Studies of wings supporting non-unique solutions in transonic flows. June 2014. AIAA-2014-2928.

[133] H. Poincaré. Sur les courbes definé par une équation differentielle. I-IV. In *Oevres*, volume 1. Gauthier-Villars, Paris, 1881-1886.

[134] G. Pönisch and H. Schwetlick. Computing turning points of curves implicitly defined by nonlinear equations depending on a parameter. *Computing*, 26:107–121, 1981.

[135] I. R. Porteous. *Geometric Differentiation: For the Intelligence of Curves and Surfaces*. Cambridge University Press, second edition, 2001.

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

[137] A. Pueyo and D. W. Zingg. Efficient Newton-Krylov solver for aerodynamic computations. *AIAA Journal*, 36(11):1991–1997, 1998.

[138] A. Pueyo and D. W. Zingg. Improvements to a Newton-Krylov solver for aerodynamic flows. January 1998. AIAA 98-0619.

[139] T. H. Pulliam. Artificial dissipation models for the Euler equations. *AIAA Journal*, 24(12):1931–1940, 1986.

[140] T. H. Pulliam. Efficient solution methods for the Navier-Stokes equations. Lecture Notes for the von Karman Institute for Fluid Dynamics Lecture Series, January 1986.

[141] T. H. Pulliam and D. S. Chaussee. A diagonal form of an approximate factorization algorithm. *Journal of Computational Physics*, 39(2), 1981.

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

[143] J. K. Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. In J. K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254. Academic Press, New York, 1971.

[144] W. C. Rheinboldt. Solution fields of nonlinear equations and continuation methods. *SIAM Journal on Numerical Analysis*, 17(2):221–237, 1980.

[145] D. S. Riley and K. H. Winters. A numerical bifurcation of natural convection in a tilted two-dimensional porous cavity. *Journal of Fluid Mechanics*, 215:309–329, 1990.

[146] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific and Statistical Computing*, 14(2):461–469, 1993.

[147] Y. Saad. Preconditioned Krylov subspace methods for CFD applications. In *Proceedings of the International Workshop on Solution Techniques for Large-Scale CFD Problems*, pages 179–195. Wiley, 1995.

[148] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, second edition, 2003.

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

[150] Y. Saad and M. Sosonkina. Distributed Schur complement techniques for general sparse linear systems. *SIAM Journal of Scientific Computing*, 21(4):1337–1357, 1999.

[151] J. Sanchez, F. Marques, and J. M. Lopez. A continuation and bifurcation technique for Navier-Stokes flows. *Journal of Computational Physics*, 180:78–98, 2002.

[152] V. Schmitt and F. Charpin. Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers. Technical report, Experimental Data Base for Computer Program Assessment, May 1979. Report of the Fluid Dynamics Panel Working Group 04, AGARD AR 138.

[153] H. Schwetlick and J. Cleve. Higher order predictors and adaptive steplength control in path following algorithms. *SIAM Journal on Numerical Analysis*, 24(6):1382–1393, 1987.

[154] X. Shen. Application of high-order summation-by-parts operators to the Reynolds-averaged Navier-Stokes equations. In *23rd Annual Conference of the Computational Fluid Dynamics Society of Canada*, 2015.

[155] S. Smale. A convergent process of price adjustment and global Newton methods. *Journal of Mathematical Economics*, 3(2):107–120, 1976.

[156] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. January 1992. AIAA 92-0439.

[157] W. Squire and G. Trapp. Using complex variables to estimate the derivatives of real functions. *SIAM Review*, 40:110–112, 1998.

[158] J. L. Steger. Implicit finite-difference simulation of flow about arbitrary two-dimensional geome-

tries. *AIAA Journal*, 16(7):679–686, 1977.

[159] B. Strand. Summation by parts for finite difference approximations for d/dx. *Journal of Computational Physics*, 110(1):47–67, 1994.

[160] M. Svärd and S. Nordström. On the order of accuracy for difference approximations of initial-boundary value problems. *Journal of Computational Physics*, 218(1):333–352, 2006.

[161] R. C. Swanson, R. Radespiel, and E. Turkel. On some numerical dissipation schemes. *Journal of Computational Physics*, 147:518–544, 1998.

[162] M. I. Syam and H. I. Siyyam. Numerical differentiation of implicitly defined curves. *Journal of Computational and Applied Mathematics*, 108:131–144, 1999.

[163] J. M. T. Thompson. The non-linear perturbation analysis of discrete structural systems. *International Journal of Solids and Structures*, 4:757–768, 1968.

[164] A. Ushida, Y. Yamagami, Y. Nishio, I. Kinouchi, and Y. Inoue. An efficient algorithm for finding multiple DC solutions based on the SPICE-oriented Newton homotopy method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(3):337–348, 2002.

[165] V. Venkatakrishnan and D. J. Mavriplis. Implicit solvers for unstructured meshes. *Journal of Computational Physics*, 105(1):83–91, 1993.

[166] C. Wales, A. L. Gaitonde, D. P. Jones, D. Avitabile, and A. R. Champneys. Numerical continuation of high Reynolds number external flows. *International Journal for Numerical Methods in Fluids*, 68(2):135–159, 2012.

[167] Z. J. Wang, K. Fidkowski, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. High-order CFD methods: Current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[168] G. S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 26(4):359–372, 1964.

[169] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.

[170] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Inc., third edition, 2010.

[171] K. H. Winters. A bifurcation study of laminar flow in a curved rectangular cross-section. *Journal of Fluid Mechanics*, 180:343–369, 1987.

[172] K. H. Winters and K. A. Cliffe. The prediction of critical points for thermal explosions in a finite volume. *Combustion and Flame*, 62(1):13–20, 1985.

[173] P. Wong and D. W. Zingg. Three-dimensional aerodynamic computations on unstructured grids using a Newton-Krylov approach. *Computers & Fluids*, 37(2):107–120, 2008.

[174] M. Yu and Z. J. Wang. Homotopy continuation for correction procedure via reconstruction - discrete Galerkin (CPR-DG) methods. January 2015. AIAA 2015-0570.

[175] T.-Y. Yu and B. K. Soni. Application of NURBS in numerical grid generation. *Computer-Aided Design*, 27:147–157, 1995.

[176] Z. J. Zhang, S. Khosravi, and D. W. Zingg. High-fidelity aerostructural optimization with integrated geometry parameterization and mesh movement. January 2015. AIAA 2015-1132.

[177] A. Zoghbi and I. Stojmenović. Fast algorithms for generating integer partitions. *International Journal of Computer Mathematics*, 70:319–332, 1998.