12-2022

Towards Reduced-order Model Accelerated Optimization for Aerodynamic Design

Andrew L. Kaminsky
University of Tennessee, Knoxville, akamins1@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Andrew L. Kaminsky entitled "Towards Reduced-order Model Accelerated Optimization for Aerodynamic Design." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

Kivanc Ekici, Major Professor

We have read this dissertation and recommend its acceptance:

Kivanc Ekici, Jay Frankel, Vasilios Alexiades, Zhili Zhang

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Towards Reduced-order Model Accelerated Optimization for Aerodynamic Design

A Dissertation Proposal Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Andrew Lawrence Kaminsky

December 2022

to my girls: Courtney, Eleanor, and Emily

Acknowledgments

I would like to thank my parents. Mom, thank you for fostering my creativity and for modeling how to research. Dad, thank you for demonstrating the value of hard work and for pushing me to strive for goals that feel unattainable. These traits were critical to completing my graduate studies.

I would also like to thank my mentors at Oak Ridge National Laboratory and at CFD Research for their investment in my growth. Ashraf, Mark, Bernie, and Phil thank you for introducing me to computational fluid dynamics, experimental flow testing, and many many other engineering disciplines. Your mentoring led me to pursue graduate studies. Yi and Kapil thank you for opening the door to continue modeling, simulation, and optimization of real world problems and for your collective guidance and mentoring. My time in industry has provided a much deeper appreciation for the knowledge I accumulated in my graduate studies.

To my past and present labmates, Reza, Hang, Seth, Brandon, Jason, and John, I thank you for your collaborative spirits and for our time shared in technical brainstorming sessions, code debugging, and celebration of milestones. I would also like to thank friends old and new for their friendship and support during my graduate studies. I greatly cherish our time shared during bridge nights, intramural frisbee games, and Sunday school.

I would like to thank my committee members Dr. Jay Frankel, Dr. Zhili Zhang, and Dr. Vasilios Alexiades. In addition to your participation on my committee your heat transfer, aerodynamics, and applied mathematics courses were among those I enjoyed most. I would also like to thank Dr. Charles Mader at the University of Michigan for his help with adjoint sensitivities and dynamic stability calculations; and Dr. Jens-Dominik Mueller at Queen Mary University of London for fielding questions on the primal timestepping adjoint.

I am deeply indebted to my advisor, Dr. Kivanc Ekici. Dr Ekici, you provided opportunities innumerable and gave me a firm push when I most needed it. Thank you for your help with securing support for my graduate studies while I was fighting health battles. Thank you for fostering a lab environment that allowed me to explore my technical interests. Thank you for your technical guidance and the professional advice you bestowed upon me. I vividly remember an early conversation with you in which you advised me that selecting a Ph.D. advisor and a spouse may be the two most important decisions of student's life. I am happy to say that I feel blessed with the decision I made in both regards.

Finally, I would like to express my deepest appreciation to my wife, Courtney. Your patience, selflessness, support, and love made completing my graduate studies possible. Thank you for everything. I love you.

Completing my graduate studies would not have been possible without the support I received during my Ph.D. studies from the National Science Foundation (under grant No.: CBET-1150332); the Bredesen Center for Interdisciplinary Research and Graduate Education Fellowship and the Mechanical, Aerospace, and Biomedical Engineering (MABE) department Chancellor's Fellowship; the Engineering Fundamentals Program at the University of Tennessee; and CFD Research Corporation. Thank you for your support.

Abstract

The adoption of mathematically formal simulation-based optimization approaches within aerodynamic design depends upon a delicate balance of affordability and accessibility. Techniques are needed to accelerate the simulation-based optimization process, but they must remain approachable enough for the implementation time to not eliminate the cost savings or act as a barrier to adoption.

This dissertation introduces a reduced-order model technique for accelerating fixed-point iterative solvers, such as those employed to solve primal equations, sensitivity equations, design equations, and their combination. The reduced-order model-based acceleration technique collects snapshots of early iteration (pre-convergent) solutions and residuals and then uses them to project to significantly more accurate solutions, i.e., those associated with smaller residuals. The acceleration technique can be combined with other convergence schemes like multigrid and adaptive timestepping. The technique is also generalizable and in this work is demonstrated to accelerate steady and unsteady flow solutions; continuous and discrete adjoint sensitivity solutions; and one-shot design optimization solutions. This final application, reduced-order model accelerated one-shot optimization approach, in particular represents a step towards more efficient aerodynamic design optimization.

Through this series of applications, different basis vectors were considered and best practices for snapshot collection procedures were outlined for the reduced-order model acceleration technique. The major outcome of this dissertation is the development and demonstration of this reduced-order model acceleration technique. This work includes the first application of the reduced-order model-based acceleration method to an explicit one-shot iterative optimization process.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Background and Related Work . . . 2
  - 1.2.1 Optimization . . . 2
  - 1.2.2 Flow Simulation via Computational Fluid Dynamics . . . 3
  - 1.2.3 Sensitivity Analysis . . . 5
  - 1.2.4 Reduced-order modeling . . . 9
- 1.3 Contributions to the State of Art . . . 11
- 1.4 Outline . . . 11
- 1.5 Related Published Works . . . 12

2 Governing Flow Equations **14**
- 2.1 The Navier-Stokes Equations . . . 14
  - 2.1.1 The Perfect Gas Model . . . 16
  - 2.1.2 The Spalart-Allmaras Turbulence Model . . . 17
- 2.2 The Euler Equations . . . 18
  - 2.2.1 Quasi-1D Euler Equations . . . 19
- 2.3 The Harmonic Balance Equations . . . 19

3 Numerical Approach **22**
- 3.1 Non-Dimensionalization . . . 22
- 3.2 Spatial Discretization . . . 22
  - 3.2.1 Cell-Centered Discretization . . . 23
- 3.3 Temporal Discretization . . . 25
- 3.4 Discrete Boundary Conditions . . . 27
  - 3.4.1 Solid Wall Boundaries . . . 27
  - 3.4.2 Far Field Boundaries . . . 28
- 3.5 Convergence Acceleration Techniques . . . 30
  - 3.5.1 Local Time-Stepping . . . 30
  - 3.5.2 Residual Smoothing . . . 31
  - 3.5.3 Multigrid . . . 31

4 Sensitivity Equations and Analysis **34**
4.1 Derivation of the Direct Approach . . . 34
4.2 Derivation of the Adjoint Approach . . . 36
4.2.1 Continuous adjoint method . . . 37
4.3 Automatic Differentiation . . . 42
4.3.1 Forward mode automatic differentiation . . . 43
4.3.2 Reverse mode automatic differentiation . . . 44

5 Validation and Verification **49**
5.1 Quasi-1D Flow through a Nozzle . . . 49
5.1.1 Nozzle with subsonic flow field . . . 50
5.1.2 Nozzle with subsonic inlet supersonic outlet . . . 50
5.1.3 Nozzle with shocked flow . . . 51
5.2 Viscous 2D flow over an RAE2822 Airfoil . . . 51
5.2.1 AGARD Test Case 1 . . . 52
5.2.2 AGARD Test Case 6 . . . 52
5.3 Inviscid 2D Flow over an Oscillating NACA0012 Airfoil . . . 54
5.4 Inviscid 2D Flow over a Plunging NACA0012 Airfoil . . . 54
5.4.1 Time-spectral Stability Derivative Method . . . 54
5.4.2 Unsteady Adjoint Sensitivity Verification for Dynamic Stability Derivatives . . . 56

6 Optimization methods **58**
6.1 Traditional Gradient-Based Optimization . . . 58
6.1.1 Steepest Descent Method . . . 59
6.1.2 Newton's Method . . . 59
6.1.3 Quasi-Newton Broyden, Fletcher, Goldfarb, and Shanno . . . 60
6.2 One-Shot Gradient-Based Optimization . . . 61
6.2.1 Single-step One-shot Optimization . . . 62
6.2.2 Pseudo-time Stepping One-shot Optimization . . . 62

7 Reduced-Order Modeling **64**
7.1 Reduced-Order Model-based Convergence Acceleration of Fixed-Point Iterators 65
7.1.1 Reduced-order Model Acceleration with Snapshot Basis Vectors . . . 66
7.1.2 Reduced-order Model Acceleration with Covariance Basis Vectors . . . 68
7.1.3 Reduced-order Model Acceleration with Orthogonal Basis Vectors . . . 70

8 Demonstration of Reduced-order Model Acceleration **72**
8.1 Acceleration of an Unsteady Harmonic Balance Solution . . . 72
8.1.1 Flow over an Oscillating RAE 2822 Airfoil . . . 73
8.2 Acceleration of a Continuous Adjoint Solver . . . 73
8.2.1 Case 1: fully subsonic nozzle . . . 74
8.2.2 Nozzle with subsonic inlet supersonic outlet . . . 78

8.2.3 Nozzle with shocked flow . . . 79
8.3 Accelerated Nested Optimization Study through Projected Discrete Adjoint Sensitivities . . . 80
8.3.1 Sensitivity Projection for Inverse Design of a 2D Cascade . . . 81
8.4 Acceleration of a Nested Optimization Scheme . . . 83
8.4.1 Inverse Design of NREL S809 Airfoil in Inviscid Flow Field . . . 83
8.4.2 Inverse Design of NREL S809 in a Viscous Flow Field . . . 87
8.5 Acceleration of a One-shot Optimization Scheme . . . 89
8.5.1 Inverse Design of Converging-Diverging Nozzle . . . 89

9 Conclusions and Recommendations **94**
9.1 Summary . . . 94
9.2 Future Work . . . 95

Bibliography **97**

Appendix **115**

Vita **179**

List of Tables

1 Coefficients for hybrid multistage Runge–Kutta scheme 116
2 Flow condition at far field boundary for one-dimensional flow. 116
3 RAE 2822 AGARD case parameters and computational corrections 128
4 RAE 2822 AGARD case 1 experimental and CFD results 128
5 Computational cost for convergence acceleration techniques 128
6 RAE 2822 AGARD case 6 parameters and computational corrections 128
7 A comparison of sensitivity values for AGARD case 6 133
8 A comparison of sensitivity values at different iterations 133
9 RAE 2822 AGARD case 9 parameters and computational corrections 133
10 Snapshot collection parameters for the varied initial snapshot location cases . 141
11 Iteration reduction for increasing snapshot count 141
12 Acceleration performance for increasing snapshot counts 147
13 Iteration reduction for increasing snapshot count with a single application . . 147
14 Iteration reduction for increasing snapshot count with a single application applied to the 1 level multigrid scheme 153
15 Iteration reduction for increasing snapshot count with multiple applications applied to the 1 level multigrid scheme 153
16 Snapshot collection parameters for the varied snapshot cases. 160
17 Snapshot collection parameters for the varied initial snapshot location cases . 160
18 Computation cost comparison of each sensitivity approach to reach $\|\bar{\boldsymbol{R}}(\bar{\boldsymbol{U}})\|_2 = 1 \times 10^{-15}$ 160
19 Comparison of loading coefficients for the NREL S809 airfoil. 160
20 Snapshot collection parameters for the viscous flow case. 162
21 Computational cost comparison of design optimization 162
22 Snapshot collection parameters for the nozzle one-shot optimization case with a single acceleration. 162
23 Snapshot collection parameters for the nozzle one-shot optimization case. . . 162

List of Figures

1 Nested gradient-based design-optimization flow chart. 117
2 Sub-time level Mach number contours for a pitching NACA 0012. [109] . . . 118
3 Spatial discretization using (a) cell-vertex and (b) cell-centered schemes [16]. 118
4 The auxiliary control volume for derivative evaluation in a cell-centered scheme [16]. 119
5 Ghost cell indices across solid boundary. 119
6 Characteristics at the flow inlet for (a) subsonic and (b) supersonic flow. . . 120
7 Typical convergence history of an explicit steady state solver. 120
8 The multigrid cycle from fine to coarse and back. Here • denotes restriction and ∘ corresponds to prolongation. 121
9 Cell-centered multigrid (a) Restriction and (b) Prolongation. Here are the centers of the fine grid and ■ is the center of the coarse grid. 121
10 Forward mode operator overloading for multiplication. 122
11 Code utilizing operator overloading. 123
12 Forward Mode Automatic Differentiation of Convective Flux Subroutine. . . 124
13 Simplified flow solver code. 125
14 Automatically differentiated flow solver code. 125
15 Reverse mode AD sensitivity calculation flow chart. 125
16 Simplified flow solver code. 126
17 Unsteady adjoint code from brute force automatic differentiation. 126
18 (a) Pseudo-code for the forward pass and (b) Pseudo-code for the reverse pass using brute force AD. Adapted from Christakopoulos et al. [27]. 127
19 Pseudo-code for the reverse pass using the primal time-stepping adjoint AD approach. Adapted from Christakopoulos et al. [27] 127
20 The cross-sectional area of the nozzle is defined by a sinusoidal decrease near the throat. 129
21 Fully subsonic nozzle solution (a) convergence history and (b) Mach number distribution. 129
22 Adjoint solver (a) convergence history and (b) comparison of the adjoint solution vector with that of Lozano and Ponsin [128]. 130
23 Transonic nozzle (a) Mach number distribution and (b) adjoint vector solution both match the values reported by Giles and Pierce [62]. 130

24 Shocked nozzle (a) Mach number distribution and (b) adjoint vector solution match those of Giles and Pierce [62]. . . . 131
25 RAE 2822 viscous grid . . . 131
26 (a) Pressure coefficient distribution and (b) convergence history for the RAE 2822 airfoil at AGARD case 1. . . . 132
27 Lift coefficient for varying angles of attack over RAE 2822 at $M = 0.725$ and $\mathrm{Re} = 6.5 \times 10^6$ . . . 132
28 AGARD Case 6 lift coefficients and sensitivity to angle of attack. . . . 134
29 (a) Pressure coefficient and (b) Mach contours for the RAE airfoil at $M = 0.734$, $\alpha = 2.79°$, and 6.5×10^6. . . . 134
30 NACA 0012 inviscid grid . . . 135
31 Harmonic balance unsteady Mach number contours for unsteady NACA 0012 case at each sub-time. . . . 136
32 NACA 0012 (a) normal force and(b) pitching moment coefficients vs. α. Note: Ronch data is digitized from Ref.[33] . . . 136
33 NACA 0012 (a) zeroth and (b) first harmonic imaginary unsteady surface pressure coefficient distribution . . . 137
34 Typical harmonic balance solutions: C_L vs. α for an oscillating flat plate . . 137
35 Typical harmonic balance solutions: R_{C_L} vs. $\dot{\alpha}$ for an oscillating flat plate . 138
36 NACA 0012 inviscid grid . . . 138
37 NACA 0012 time-spectral stability derivative verification . . . 139
38 Comparison of adjoint $\mathrm{d}C_{L_\alpha}/\mathrm{d}k$ derivatives with time-spectral dynamic stability profile . . . 140
39 Comparison of $\mathrm{d}C_{L_\alpha}/\mathrm{d}k$ calculated via an adjoint approach and finite difference 140
40 RAE 2822 viscous grid. . . . 142
41 Surface pressure distributions for the five harmonic balance sub-time solutions. 143
42 Unaccelerated harmonic balance primal solution convergence history. . . . 143
43 Comparison of the convergence history of the primal harmonic balance solver accelerated with the reduced-order model-base acceleration technique. . . . 144
44 Adjoint solver (a) convergence history and (b) comparison of the adjoint solution vector with that of Lozano and Ponsin [128]. . . . 144
45 Convergence history achieved by varying snapshot count: increased snapshot quantities improve convergence acceleration . . . 145
46 Projected adjoint vector profiles offer considerable improvement over the adjoint vector profile of the most converged snapshot. . . . 145
47 The differences between the projected adjoint vector profiles and converged values show that the projections significantly improve the present solution. . 146
48 Multiple applications of the convergence acceleration technique (a) using double precision reaches an artificial convergence limit but (b) using quadruple precision eliminates this limit. . . . 146
49 Convergence acceleration using orthogonal basis vectors with snapshot collection beginning at (a) iteration 500 and (b) iteration 2. . . . 148

50 Delaying the first iteration location for the 41 snapshot case improves projections until a gradual plateau is reached. 148
51 Convergence acceleration using multiple applications of the POD-based correlation technique with snapshot collection beginning at (a) iteration 2 and (b) iteration 500. 149
52 Convergence acceleration using multiple applications of the POD-based correlation technique stabilized by increasing the delay between acceleration cycles for 21 snapshots 149
53 Convergence acceleration using multiple applications of the POD-based acceleration technique delayed 13 spans. 150
54 Convergence acceleration using multiple applications of the POD-based correlation technique with 21 snapshots varying the iteration interval between snapshots. 150
55 Convergence acceleration using orthogonal basis vectors with snapshot spans of (a) 80 iterations, (b) 160 iterations, (c) 320 iterations, and (d) 640 iterations 151
56 Convergence acceleration using orthogonal basis vectors applied once. 152
57 Convergence acceleration using orthogonal basis vectors applied once to the multigrid scheme 152
58 Convergence acceleration using orthogonal basis vectors applied multiple times to the multigrid scheme 154
59 Perturbed 10th standard inviscid HOH grid 155
60 Convergence of global sensitivity residual of first design cycle. 156
61 The initial, target and optimized (a) surface pressure distribution and (b) blade shape 156
62 Comparison of cost function reduction versus CPU time for the traditional and projected adjoint sensitivity methods. 157
63 Percent difference from converged sensitivity values for each design variable at initial design point 158
64 Percent difference from converged sensitivity values for each design cycle of design variable 12 158
65 Average percent difference from converged sensitivity values for each design cycle over all design cycles 159
66 Inviscid O-type computational grids for: (a) the NREL S809 airfoil (target), and (b) the RAE 2822 airfoil (initial). Both grids are made up of 201×70 nodes. 161
67 Comparison of surface pressure distributions for the NREL S809 and RAE 2822 airfoils. 163
68 Comparison of the sensitivity values calculated via the brute force and primal time-stepping adjoint approaches. 164
69 Convergence history of the brute force and primal time-stepping adjoint. . . 165

70 Comparison of the convergence history of the primal time-stepping adjoint accelerated with reduced-order models built using: (a) different snapshot quantities, and (b) varied initial snapshot collection iterations. 165
71 Convergence history of the traditional and accelerated primal time-stepping adjoint. 166
72 Eigenvalue distribution of POD basis for the first three acceleration projections. 167
73 Cost function history of the L-BFGS optimizer with sensitivities calculated via the PTS adjoint approach with and without the ROM acceleration over (a) CPU time and (b) design cycle. 167
74 Comparison of the initial, target, and optimized (a) surface pressure profiles. and (b) airfoil surfaces. 168
75 Viscous grids for the (a) NREL S809 (target design) and (b) RAE 2822 (initial design) airfoils. Both grids have 257×127 nodes in the streamwise and normal directions, respectively. 168
76 A comparison of the surface pressure distribution of the NREL S809 airfoil. . 169
77 The adjoint sensitivity values of the b-spline control points at the initial design. Design variables 1-15 modify the top surface and 16-30 control the bottom surface. 169
78 Convergence history of the primal time-stepping adjoint sensitivity approach for the viscous flow case with and without the ROM acceleration. 170
79 History of the cost function for L-BFGS-B optimizer using PTS adjoints calculated with and without acceleration over (a) CPU time and (b) design cycle. 170
80 Comparison of the initial, target, and optimized (a) airfoil surfaces and (b) surface pressure profiles. 171
81 Inverse design problem (a) nozzle cross-section area and (b) flow solution of initial design. 171
82 Comparison of the inverse design problem initial and target (a)nozzle cross-section area and (b) pressure profile. 172
83 Convergence history of the primal, brute-force adjoint, and time-stepping adjoint sensitivity approach for the initial nozzle design. 172
84 Cost function history over (a) each design cycle and (b) time. 173
85 Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the L-BFGS-B optimizer. 173
86 Convergence history of the primal, adjoint, and design equations solved via a one-shot optimization approach. 174
87 Cost function history over (a) each design cycle and (b) time. 174
88 Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the one-shot optimizer. 175
89 Convergence history of the primal, adjoint, and design equations solved via a one-shot optimizer with a single application of the reduced-order model acceleration technique. 175

90 Design variable history for the one-shot optimizer with a single application of the reduced-order model acceleration technique. 176
91 Convergence history of the primal, adjoint, and design equations solved via the ROM-accelerated one-shot optimization approach. 176
92 Cost function history over (a) each design cycle and (b) time. 177
93 Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the ROM-accelerated one-shot optimizer. 177
94 Comparison of the optimized design variable values from each optimization approach. 178

Nomenclature

A	cross-sectional area
$\boldsymbol{A}$	Jacobian matrix
$\boldsymbol{A}_c$	convective flux Jacobian
A_n, B_n	Fourier coefficients of time level n
$\boldsymbol{b}$	right hand side vector
c	speed of sound
C_D	drag coefficient
C_L	lift coefficient
C_m	moment coefficient
$C_{L_\alpha}, C_{L_{\dot{\alpha}}}, ...$	stability derivatives
$\boldsymbol{D}$	artificial dissipation
$\boldsymbol{\mathcal{D}}$	pseudo-spectral operator
E	total energy
$\boldsymbol{E}$	discrete Fourier transform
$\boldsymbol{E}^{-1}$	inverse discrete Fourier transform
$\dot{f}, \dot{g}, \dot{h}$	grid velocity
$\boldsymbol{f}_e$	external force
f_i	function
$\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$	conservation flux vectors
$\boldsymbol{F}_C$	convective flux
$\boldsymbol{F}_D$	diffusive flux
g	sensitivity source term
h	enthalpy
J	cost function
k	coefficient of thermal conductivity or reduced frequency
K_r	Krylov subspace
$\mathcal{K}$	correlation operator
M	number of basis vectors or Mach number
N_F	number of faces
N_H	number of harmonics
p	pressure
p_t	total pressure
Pr_l	laminar Prandtl number

Pr_t	turbulent Prandtl number
$\boldsymbol{Q}$	source vector
$\boldsymbol{Q}_F$	multigrid forcing function
$\boldsymbol{R}$	residual vector
R^-, R^+	Riemann invariants
R_g	gas constant per unit mass
Re	Reynolds number
$\boldsymbol{s}$	state vector
$\boldsymbol{S}$	control volume surface
ΔS	face area
S_t	Spalart-Allmaras source term
t	time
t_1, t_2	Hicks-Henne bump function control parameters
$\mathcal{T}$	pseudo-time
T	temperature or period
$\boldsymbol{U}$	vector of conservation variables
$\boldsymbol{U}^*$	conservation variables at all sub-time levels
$\tilde{\boldsymbol{U}}$	Fourier coefficients of conservation variables
u, v, w	velocity components
$\boldsymbol{v}$	velocity vector
V	contravariant velocity
$\mathcal{V}$	volume
$\mathcal{V}'$	auxiliary volume
x, y, z	Cartesian coordinates
α	angle of attack
α_0	mean angle of attack
α_1	unsteady oscillation amplitude
α_{cor}	corrected angle of attack
α_m	stage coefficients
β_m	blending coefficients
$\boldsymbol{\beta}$	design variables
δ	Kronecker delta or perturbation
ϵ	smoothing coefficients
$\epsilon^{(2)}, \epsilon^{(4)}$	artificial dissipation coefficients
γ	specific heat ratio
λ	spectral radii or eigenvalues
λ_c	spectral radius of convective flux
λ_v	spectral radius of viscous flux
μ	dynamic viscosity
μ_l	laminar viscosity
μ_t	turbulent viscosity
ν	pressure switch

$\tilde{\nu}$	Spalart-Allmaras turbulent working variable
ω	fundamental frequency
Ω	flow domain
$\mathbf{\Phi}$	basis vector matrix
$\boldsymbol{\psi}$	adjoint vector or Lagrange multiplier
ρ	density
$\bar{\sigma}$	internal stress
$\bar{\tau}$	shear stress
$\tau_{xx}, \tau_{xy}, ...$	viscous stress
ξ	weighting coefficients
ζ	second coefficient of viscosity

Chapter 1

Introduction

1.1 Motivation

Continually increasing hardware computing resources and improvement in our ability to simulate aerodynamic flows using computational fluid dynamics (CFD) has fundamentally changed the aerodynamic design process. Industrial aerodynamic design processes now routinely use CFD alongside traditional wind-tunnel and flight testing [141], where it is commonly employed to analyze candidate designs and understand the resulting flow phenomena. In this capacity, CFD simulation is regularly employed to guide experimental testing and reduce experimental testing budgets. The use of CFD has grown to be such an integral part of the aerodynamic design process that it is hard to imagine it without CFD [141]. However, the true potential of CFD simulation within the industrial aerodynamic design process has yet to be fully realized. Extending the industrial aerodynamic design process to leverage CFD simulation within formal numerical optimization procedures would further enhance the aerodynamic design process [174].

Numerical optimization of CFD simulation-based aerodynamic design processes is complicated by high-dimensional, multi-disciplinary design spaces, expensive objective metric simulation processes, and even more expensive sensitivity calculation processes. Over the past few decades significant research and development has been performed to form coupled CFD and numerical optimization processes that can be employed to realize CFD-based design. Perhaps the most important development was Jameson's seminal work demonstrating use of the adjoint sensitivity method, borrowed from optimal control, to affordably calculate gradients for a large number of design parameters [98]. Reuther [164, 165], Giles [61, 59], Martins [142, 140, 143], and many others [160, 27, 70, 71, 69] have refined the ajdoint sensitivity calculation methodology.

The ability to simulate flow solutions and calculate their sensitivities to design parameters has led to development of several coupled CFD-simulation and numerical optimization in frameworks utilizing nested optimization strategies such as OpenMDAO [65], UNPAC [36], and SU2 [2]. To reduce the cost of simulation based optimization Hafka [73], Ta'asan [182,

184], Hazra [79, 77], Ozkaya [158] and others [53, 72, 69, 18, 115] have demonstrated paths towards more efficient one-shot optimization procedures.

Greater adoption of mathematically formal simulation-based optimization approaches within aerodynamic design depends upon a delicate balance of affordability and accessibility. Thus, techniques are needed to accelerate the simulation-based optimization process, but these techniques must remain approachable enough for the implementation time to not eliminate the cost savings. The effort of the work communicated within this dissertation has been motivated by a search for techniques to accelerate simulation-based optimization and its underlying components in an approachable manner.

1.2 Background and Related Work

A typical gradient-based design-optimization procedure can be broken down into four steps: (1) problem setup; (2) flow solution; (3) sensitivity calculation; and (4) design update, as depicted in Fig. 1. During the problem setup, the initial design, objective functions, and design variables are specified. Following the problem definition, the flow equations must be solved to evaluate the present design. If the design is deemed adequate the optimization procedure is completed; otherwise, the process must be continued. The optimization process begins with calculation of the sensitivities. Specifically, sensitivities of the objective function to the design variables are calculated to determine an appropriate optimization search direction for a gradient-based optimization procedure. The primary goal of the present work is to accelerate this process.

1.2.1 Optimization

Optimization is the process of identifying a global or local extremum of a given function. Design optimization is performed by defining an objective function that is maximized (or minimized) when an ideal design is reached. In simulation-based aerodynamic design, the objective function is based upon the converged flow solution. Typical cost functions include: lift to drag ratios [84], stage efficiency [198], the difference from a target surface pressure profile [94, 107], or even fuel efficiency [122]. Optimization methods for identifying the objective function extremum can be classified as either gradient-based or gradient-free.

In the gradient-free approach, optimization is performed by evaluating the cost function at different design points to find the optimum point. Gradient-free techniques include genetic algorithms, random search, and space filling methods. Gradient-free optimization can provide the global optimum of any function, but as the dimension of the design space increases the number of function evaluations required can cause them to be intractable [132, 161].

Gradient-based optimization provides an interesting alternative when a local extremum is sufficient. In aerodynamic design optimization, engineers typically have an initial guess to the desired design, and an optimum near this original design is often sufficient. To find a local extremum, gradient-based procedures use the gradient of the cost function to decision

parameters to determine the search direction and step size used to update the design within an iterative design process. The steepest descent method, a simple first order gradient-based technique, defines the search direction opposite that of the gradient. Convergence to a local minimum requires a large number of design cycles, on the order of N^2, where N is the dimension of the design space. Newton's method is a second-order gradient-based technique that converges quadratically, but it requires the calculation of the Hessians, the second-order derivative matrices, for each design cycle. The Hessian calculation is computationally very expensive and can be cost prohibitive. Quasi-Newton methods provide a compromise by approximating the Hessian using the changes in the gradients between design cycles. The Davidon-Fletcher-Powell (DFP) updating algorithm [34, 204] was the first quasi-Newton algorithm, but its popularity has since been superseded by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [204].

Recently one-shot optimization procedures which solve the design system (flow, gradient and design solutions) simultaneously have been considered for aerodynamic design optimization. Kuruvila et al. [113] originally used a multi-level Newton method to advance all parts. Griewank and Faure [69] developed a 'piggy-back' method to simultaneously calculate the flow, sensitivity, and design solutions. More recently, pseudo-time-stepping approaches [27, 77, 104] have been used to march the flow, sensitivity, and design solutions simultaneously. By coupling the convergence of the design system, increasingly accurate flow and gradient information are used to update the design. This is beneficial because the procedure is able to update the early design with relatively inaccurate gradients when only a search direction is needed. Then as the design begins to converge, highly accurate gradients are used to update the design. One-shot approaches have been shown to result in considerable cost savings.

These gradient-based optimization procedures return a local optimum which, may or may not be the global optimum. Optimization from multiple start locations can be considered to surmount this, but in numerous aerodynamic considerations the design is initiated from a baseline shape so that a local-optimum is sufficient. A gradient-based optimization process can also be complicated by noisy or non-smooth results, but multi-point optimization can be used to overcome this challenge [26]. The biggest cost associated with gradient-based optimization is the sensitivity calculation, which is therefore a point of considerable interest in our pursuit to accelerate the design cycle.

1.2.2 Flow Simulation via Computational Fluid Dynamics

In design optimization, it is often important to consider unsteady flow behavior in addition to time-averaged steady states. Problems such as flutter [41, 121], wake-shock interactions [64], vortex shedding [63], and row interactions [194], can have significant impacts on the loading, efficiency, and stability, and should be considered.

Work on unsteady CFD time-stepping approaches began as early as the 1950s [75], but it was Jameson's dual time-stepping approach [99] introduced in 1991 that made time-dependent solutions practical. In the dual time-stepping method, the temporal derivative

term is approximated by a second-order backward difference operator. To solve the discretized equations, a pseudo-time derivative is added to the equations. The solution is marched in pseudo-time until convergence is reached (usually when the residual has been dropped two or three orders of magnitude). When convergence is reached, the pseudo-time derivative term vanishes, and the original equations are recovered. The solution is then advanced to the next physical time. The dual time-stepping method has been used to model unsteady shock motion for an airfoil [170], fluid-structure interaction for wing flutter [124], and turbomachinery [28, 81]. The dual time-stepping method accurately models the flow in a straightforward way, but the computational cost can become unmanageable for large time-scale solutions. This cost is further exacerbated when sensitivity calculations are needed due to reasons that will be explained in detail throughout this work.

Flows over wings, through turbomachinery, and around wind turbines often behave periodically, which makes aerodynamic analysis of these systems well-suited to frequency domain techniques like the harmonic balance method [74] proposed by Hall et al. [74]. The harmonic balance method models the unsteady flow variables using a Fourier series in time with spatially varying coefficients. The Fourier series coefficients are computed by evaluating the flow solution at equally spaced sub-time levels over a single period, as shown in Fig. 2. These sub-time solutions are coupled using a pseudo-spectral operator to approximate the physical time derivative term. Since the coupled sub-time level solutions are mathematically steady, common steady equation acceleration techniques including multigrid, residual smoothing, and local time-stepping can be used to significantly reduce the computational cost relative to traditional unsteady methods [45, 48]. The harmonic balance method has been shown to be up to two orders of magnitude faster than traditional time-accurate solutions [46] or roughly 5-10 times the cost of a steady Navier-Stokes analysis [89]. Among many other problems, the harmonic balance method has been used to model, unsteady flows over unsteady turbomachinery [74], limit-cycle oscillations [45, 185, 186], and unsteady flows about helicopter rotors [46], to name a few.

Jameson et al. [100] and McMullen and Jameson [148] developed a variation called the non-linear frequency domain (NLFD) method, which uses Fourier coefficients as the dependent variables. This requires Fourier transformation and inverse Fourier transformation of conservation variables and residuals between the time and frequency domain at each iteration. Accordingly, the harmonic balance technique is easier to implement.

Several have developed improvements or extensions of the original harmonic balance method. Maple et al. [136] proposed an adaptive harmonic balance to consider flows with varying levels of unsteadiness. Fewer harmonics are retained in relatively steady regions, and more harmonics are retained in highly unsteady regions. The multi-frequency harmonic balance technique was developed for application to aperiodic flows [43, 44]. In some instances, the frequency of interest is unknown *a priori*. Several techniques have been developed to determine the appropriate frequency [49, 63, 178] during the solution.

The harmonic balance technique is a model-order reduction technique that can considerably reduce the computational cost for unsteady considerations while providing quite accurate solutions. However, like all model-order reduction techniques some accuracy is

traded for the computational speedup. For example, the harmonic balance approach is shown to introduce aliasing errors for some highly nonlinear problems and investigators have developed/sought ways to address these issues.

1.2.3 Sensitivity Analysis

Following calculation of the flow solution and the objective metric, the sensitivity can be calculated to inform the design update. The sensitivity of the objective function to design variables can be calculated in several ways including: finite differencing, the direct approach, and the adjoint approach.

Finite differencing is the most straightforward method, since only a flow solver is needed. This method approximates the sensitivity of the objective function to a design parameter by evaluating the objective function at a given design point, and then re-evaluating the objective function after perturbing the design parameter. The difference between the initial and perturbed solution is divided by the size of the perturbation to yield the gradient of the objective function with respect to the single perturbed design variable. Sobieszczanski-Sobieski [175] highlighted the need to determine sensitivity derivatives, and presented finite differencing as a potential technique. Beux and Dervieux [11] considered derivatives approximated using finite differences to optimize a nozzle while studying the impact of second order accuracy on sensitivity derivatives. Hicks et al. [85] used finite differencing for airfoil design.

The primary drawback of finite differencing is that the process must be repeated for each design variable of interest, so the computational cost associated with sensitivity calculation is directly proportional to the number of design variables considered. Aerodynamic design optimization procedures potentially require tens to hundreds of design variables [146], which makes this type of sensitivity calculation undesirable. Another drawback of this approach is the dependence of its accuracy to the perturbation size. The choice of perturbation size is faced with mutual influence of truncation and cancellation error [14]. The step size must be small enough to decrease the truncation error but still large enough to limit the cancellation of significant digits that cause round-off error. The sensitivity to step size can be removed though by considering the complex step method developed by Martins et al. [145]. Due to these limitations, finite differencing is not commonly used to calculate aerodynamic design sensitivities any more, but it is frequently used to verify other sensitivity approaches [3].

Direct sensitivity analysis

The direct sensitivity approach offers an another alternative that alleviates the dependence on perturbation step size. In the direct sensitivity approach, the flow solution is computed once, and then a differentiated state problem is computed for each design parameter. The computation cost of the differentiated state problem is typically slightly more than that of a flow solution. However, if multiple cost functions are of interest, the sensitivity of all cost functions to a single design variable can be calculated simultaneously, which can result in cost reduction.

Development of the differentiated state problem can be automated using automatic differentiation, which yields a fixed-point iterative solver that can be accelerated using traditional acceleration techniques. The direct sensitivity approach is easily parallelized, because the sensitivity solution of each variable can be run independently. Differentiation of parallelized solvers is quite challenging [163], so this inherent parallelization is quite valuable. If only a few design variables are considered, the direct sensitivity approach represents a valuable technique.

Within aerodynamic design optimization, the direct sensitivity approach has been applied by Baysal and Eleshaky [8] and Baysal et al. [9] to optimize a scramjet nozzle-afterbody section. Bischof et al. [15] used a forward mode automatic differentiation to determine the sensitivity of flow over a backward-facing step to empirical parameters of several turbulence models. Within a design optimization context the direct sensitivity method can also accelerate Hessian calculations [55], which can be used within Newton-based optimization schemes.

Adjoint sensitivity analysis

The final sensitivity calculation technique is the adjoint sensitivity method [123]. The adjoint method can be used to find the gradient information at a cost that is independent of the number of design variables considered. The adjoint sensitivity method begins with the solution of the flow equation followed by a solution of the adjoint equations. The flow equation is typically referred to as the forward problem, and the adjoint equation is often called the backward or reverse problem. The forward and backward problems can both be solved using the same numerical techniques. Theory suggests that the required number of floating point operations for adjoint computations is no more than three times that of the original code [61]. The key feature of the adjoint method is it provides a significant reduction in computational costs for problems, like aerodynamic design optimization, that utilize several design variables.

From a mathematical viewpoint, the analytic adjoint equations are obtained by linearizing the flow equations [128]. For numerical applications, a discretized version of the adjoint equations is required which can be developed using one of two approaches: the continuous [102] or discrete [61, 188] approach.

The continuous approach formulates the adjoint of the governing partial differential equations, and then discretizes these linear equations. This approach was the method originally developed by Jameson in his seminal work [98]. Borrowing from the field of optimal control Jameson applied calculus of variations to the governing flow equations to obtain the adjoint equations [141]. The continuous adjoint approach has the advantage that it conveys the physical significance of the adjoint variables and boundary conditions. In direct comparisons [149], the continuous approach has been demonstrated to form solvers that were less computationally expensive. However derivation of the adjoint boundary conditions for the continuous adjoint approach can be challenging and has received a lot of attention [60, 128]. It can also be difficult to derive the adjoint equations for

turbulence models [210]. In practice, the turbulence field is often frozen and considered to be independent of the design variables. Finally because discretization and linearization are generally noncommutative, the sensitivities found by a continuous solver are not consistent with the discretization and therefore won't exactly match those found through finite difference approximations.

Conversely, the discrete approach develops the discretized adjoint equations in the reverse manner by first discretizing the governing partial differential equations (i.e., the CFD solver) and then linearizing the flow solver. The discrete adjoint approach provides the exact gradient of the discrete objective function [61]. Implementation of the discrete adjoint method can be simplified by using automatic differentiation, which will be discussed in the following subsection. The more straightforward implementation has lead to greater adoption of the discrete adjoint method [141].

Within aerodynamic design optimization the adjoint sensitivity method is the most prevalent sensitivity technique. As mentioned earlier, the continuous adjoint approach was first considered by Jameson et al [98], within aerodynamic design. It has since been extended to multi-element high-lift configurations [111], 3D wing [102] and wing-body shape optimization [164, 165], 3D turbomachinery shape optimization [131], and coupled aerodynamic-structural optimization [140, 142, 143]. These design studies have improved lift to drag ratios, increased flight range, and improved off design performance. In earlier work, the discrete adjoint approach has been hand-coded for 3D Euler equation-based shape optimization [50], and the 3D Navier-stokes equation based shape optimization [155]. More commonly though, the discrete adjoint approach has also been implemented using automatic differentiation. For example, automatic differentiation has been used by Giles et al. [59] for sensitivity calculation of airfoils in inviscid and viscous flows, by Nielson et al. [154] for 3D wing optimization, and by Wang and He [197] and Wang et al. [198] for sensitivities of multi-stage turbomachinery.

The adjoint method has enjoyed more popularity within steady flow considerations. Application of the adjoint method to unsteady flows is complicated by the reversal nature of the adjoint calculation, which requires storage of the time-accurate flow solution history [151]. Storage of the unsteady solution history can be untenable. However, the harmonic balance method enables unsteady flows to be considered using mathematically steady equations, which alleviates the solution storage requirement. The adjoint technique has been applied to the harmonic balance method for unsteady design optimization of wings and airfoils [150, 151], unsteady turbomachinery [94, 40], and unsteady stability derivative calculations by the present author [109].

Automatic Differentiation

Development of the direct method and the discrete adjoint approach can be streamlined using automatic differentiation. Automatic differentiation is based upon the premise that numerical codes like CFD solvers are made up of a series operations that can be reduced to elementary arithmetic operations. Individually, each of these simple operations can easily

be differentiated. Automatic differentiation techniques apply the chain rule in an automated manner to generate the derivative of the complete sequence [152]. Two common approaches are followed for implementing automatic differentiation: operator overloading [70] and source code transformation [76].

The operator overloading approach 'overloads' real numbers and integers so that the defined type variables store the parameter of interest as well as that parameter's derivative. In addition, every intrinsic function and operation (e.g., addition, subtraction, multiplication, exponential, etc.) must be overloaded so that when those operations are performed, both the nominal variable values and the values of the derivatives of the variables are calculated. An advantage of this approach is that very few changes to the original source code are required. Generally, only changes to the data types are required. However, this approach can only be utilized in languages where overloading is supported. In languages where it is supported, operator overloading can apply a great deal of pressure on the compiler to remove all the extra dispatches, allocations, and memory references it introduces. Also it is often more costly compared to the source code transformation approach, because derivatives are calculated for every operation regardless of whether it is pertinent to the quantities of interest.

In source code transformation, the source code for a function is replaced by an automatically generated source code. This code includes statements that calculate derivatives of functions found within the original code. Source code transformation can be implemented for any programming language. However, the creation of a source code transformation type AD tool is difficult, but several AD packages including ADIFOR [12], TAF [54], and TAPENADE [76] are available. Existing automatic differentiation tools generally only require the user to specify which function's gradients are desired and for which variables the gradients should be found. TAPENADE is the source-code transformation tool used within this work. Automatic differentiation can be performed in forward or reverse modes, which respectively correspond to the direct and adjoint sensitivity methods. The resulting derivative code generated using source code transformation is typically easier to optimize because the derivative code is directly available.

Early efforts to apply automatic differentiation to CFD began in the 1990s with the calculation of non-geometric sensitivity derivatives [13], followed by sensitivity to turbulence modeling parameters[66], and sensitivity to geometric inputs [24]. It has since been utilized within frequency domain solvers [94, 188] and unsteady flows [200].

Though this approach is called automatic differentiation it is not entirely automatic since considerable work can be required to prepare the code before and after differentiation [106]. The simplest application of automatic differentiation is called the 'brute force' approach. In the 'brute force' approach, the entire source code is provided to the source code transformation tool. In the forward mode, 'brute force' automatic differentiation builds a fixed-point iterative method to determine the sensitivity values, but in the reverse mode the automatic differentiation does not yield a fixed-point method [27]. As a result, many investigators have developed alternative applications of the reverse mode. Mader and Martins [134] proposed the ADjoint approach which builds the adjoint problem term by

term. Christakopoulos et al. [27] introduced the 'primal time-stepping' adjoint which uses hand assembly to form an iterative fixed-point iterative approach.

Automatic differentiation has primarily been applied to steady problems, because the reversal required in the adjoint calculation requires storage of flow history, which can be quite large. A few techniques have been developed to reduce the associated cost. Wang [199] developed a checkpointing algorithm, which trades memory usage for run-time by only reversing parts of the program at a time [71]. Beran et al. [10] proposed a POD compression technique to compress the time-accurate solution history. Frequency domain techniques like the harmonic balance and non-linear frequency domain method are mathematically steady; therefore, storage of the solution history is not necessary. This makes them well-suited to automatic differentiation. Adjoint harmonic balance and non-linear frequency domain solvers have been developed for helicopter rotor design [25], a pitching and plunging wing [135], and turbomachinery stage optimization [94].

1.2.4 Reduced-order modeling

High-fidelity computational fluid dynamic models and sensitivity analysis are becoming increasingly valuable design tools. Their utility is only limited by their computational cost and the information they are able to convey for flows of interest [129]. Model order reduction can be used to decrease computational cost by distilling the system to the minimum information needed. Projection-based model order reduction methods can be used to project the high-order equations onto a subspace of the original space. This is equivalent to transforming the original high-order system of equations to a system of much lower order, by retaining only the most dominant portions of the solution dynamics [7]. The projection-based model effectively "compresses" the system's state information by projecting the state behavior onto a lower dimensional subspace thereby rewriting the governing equations in a compressed representation [137]. This low dimensional space consists of a basis that is developed to capture the features of the solution using the fewest states possible. The accuracy of any reduced-order model is dependent upon how well the solution features are captured.

In 1807 Fourier presented one of the first attempts to approximate a complicated function with a simpler formulation [171]. He used the orthogonal relationship of the sine and cosine functions to form a basis, in what is now known as the Fourier series. Model order reduction in its present form really began with Lanczos [171] who tried to reduce tridiagonal matrices and Arnoldi who realized that smaller matrices could provide a good approximation of a larger matrix [5]. Since then several methods to form reduced-order bases have been proposed including: Krylov-subspaces and proper orthogonal decomposition (POD) [126].

Krylov-based methods find the eigenvalues of large sparse matrices or solve large linear systems of equations without performing a direct linear solve, but instead use matrix-vector multiplication. Given a linear system of equations $\mathbf{Ax} = \mathbf{b}$, an order-r Krylov subspace is built by multiplying right hand side vector $\mathbf{b}$ in succession with the first r powers of $\mathbf{A}$, i.e., $K_r = \text{span}\left(\mathbf{b}, \mathbf{Ab}, \mathbf{A}^2\mathbf{b}, ..., \mathbf{A}^{r-1}\mathbf{b}\right)$. However, the vectors become linearly dependent

so Krylov-based methods commonly use an orthogonalization scheme [82], such as Lanczos iteration or Arnoldi iteration, to form the reduced-order basis. Krylov-based methods are well-suited for reduction of large systems, because they preserve stability and passivity which ensures the system does not produce energy. Krylov subspace methods have been used for simulation of circuits [6, 166, 190], micromachined devices [166, 207], and wireless systems [83]. Within computational fluid dynamics, among many others, Wilcox et al. [202] developed an Arnoldi approach to model mistuned turbomachinery rotors and Lassaux [117] used an Arnoldi derived reduced-order model for active shock position control.

Proper orthogonal decomposition is used to decompose a spatio-temporal signal into a linear combination of orthogonal spatial basis functions and temporal coefficients. The basis is chosen such that it is "optimal" in the sense that the error in the projection onto the subspace is minimized. The solution of the optimization problem reduces to an eigenvalue problem, and provides an orthogonal basis for the optimal subspace [168]. The method of snapshots introduced by Sirovich [173], can be used to determine the eigenfunctions from a set of samples or snapshots of the system.

Within fluid mechanics, proper orthogonal decomposition was first applied to the study of turbulence [130]. Proper orthogonal decomposition is arguably the most popular model order reduction technique for non-linear systems [23]. As such it has been extended to a wide range of applications. Bui-Thahn et al. [20] proposed a gappy POD method to reconstruct flow fields from incomplete data sets. LeGreseley and Alonso [119] considered POD-based reduced-order models for design optimization by collecting snapshots at different design variable values. Ekici and Hall [42] used POD to estimate time-linearized unsteady flows in turbomachinery for different interblade phase angles.

Convergence acceleration

In addition to the previously discussed applications, reduced-order modeling techniques can be used as convergence acceleration techniques by transforming, or projecting, a slowly converging sequence to a new sequence that converges more rapidly to the same limit. These methods, called extrapolation methods, were first pioneered by Wynn's scalar and vector epsilon algorithms [206]. More recently, the minimal polynomial extrapolation has been used to approximate the eigenvalues of the flow solver [22]. The eigenvalues are used in the characteristic polynomial to extrapolate a solution to minimize the convergence error. Alternatively, Jespersen and Bunning accelerated an iterative process for the Euler equations by annihilating the unstable eigenvalues [105]. Ekici et al. [47] extended this concept to the Navier–Stokes equations by employing proper orthogonal decomposition (POD) to approximate the unstable eigenvalues. Mader et al. [134] used GMRES, a Krylov subspace method, to calculate adjoint sensitivities. Model order reduction has also been performed to find an initial guess for iterative solvers. Clemens et al. [29] used Gram-Schmidt orthogonalization to form a basis to extrapolate an initial guess in their subspace projection extrapolation scheme. Liu et al. [125] used dynamic mode decomposition, a Krylov subspace

derivative, in the mode multigrid method to accelerate steady flow solutions. Markovinović and Jansen used POD to project initial guesses from previous time-steps [137].

The present author has also co-developed a reduced-order model-based acceleration scheme to approximate converged flow solutions [38] with considerable success. The same technique has been applied to accelerate continuous adjoint solutions [108] and discrete adjoint projection [107, 110]. This acceleration scheme avoids the need to generate several high-fidelity solutions, which is a weakness of previous POD techniques. The technique collects snapshots during a single solution at a set of preconvergent iterations. The technique can be used in conjunction with traditional acceleration schemes including local time-stepping and multigrid for even greater acceleration. It should be noted that despite the relatively few attempts to accelerate the solution of the sensitivity equations, the adjoint and direct method are well-suited to reduced-order model-based acceleration because they are linear. Furthermore, acceleration of the sensitivity procedure is especially valuable because of its higher cost relative to the flow solution. This technique is a key component of the present dissertation.

1.3 Contributions to the State of Art

This dissertation proposal puts forward a novel reduced-order model accelerated one-shot optimizer for aerodynamic design. The reduced-order modeling technique dramatically reduces the cost associated with one-shot optimization. The traditional one-shot approach limits the cost of early design cycles by performing design updates before convergence is reached. The flow solution and the adjoint solution are converged simultaneously to an accuracy that is increased as the design updates progress. The reduced-order model acceleration technique takes snapshots of preconvergent solutions and uses them to project the solution forward towards to a significantly more accurate solution, i.e. smaller residual. In this work the reduced-order model based acceleration technique is demonstrated to independently accelerate primal and adjoint solutions and combined primal, adjoint and design solutions within the one-shot approach. Development of the proposed reduced-order model accelerated one-shot optimizer for aerodynamic design requires:

- Reduced-order model-based acceleration of the primal flow solutions.
- Reduced-order model-based acceleration of the sensitivity analysis.
- Reduced-order model-based acceleration of the one-shot design step.

1.4 Outline

The gradient-based aerodynamic design cycle requires a flow solution, sensitivity calculation, and design update. Before the accelerated unsteady one-shot gradient-based optimizer can

be built, the theory, implementation, and verification of the flow solver, sensitivity solver, and reduced-order model acceleration technique must be demonstrated.

In Chapter 2, the governing flow equations are presented. The high-fidelity Navier–Stokes equations are outlined and the simplified inviscid Euler equations are derived. The harmonic balance method is presented for efficient analysis of unsteady periodic flows. Following the introduction of the governing equations, the discretization method is discussed in Chapter 3. The numerical approaches of the in-house solvers, *external* and *cascade*, are outlined.

Next the sensitivity equations are introduced in Chapter 4. Initially the direct (or forward) sensitivity is derived. Afterwards the adjoint sensitivity approach commonly used in aerodynamic design is detailed. Both the continuous and discrete variants of the adjoint approach are considered. Finally, automatic differentiation is introduced as a method for efficiently developing derivative code.

In Chapter 5, the flow and sensitivity solvers are validated and verified. The flow solvers are validated against experimental and externally generated computational data for steady and unsteady flows. Traditional convergence acceleration techniques are considered to establish baseline performance. The sensitivity approaches are verified through comparison with finite differencing and validated with sensitivity results from the literature.

Chapter 6 introduces optimization techniques employing the primal flow solvers and adjoint sensitivity solvers. First, traditional nested optimization approaches are detailed. Moving from techniques of low complexity and slow convergence rates to greater complexity and faster convergence rates. The chapter culminates with the introduction of one-shot optimization techniques, which drop feasibility requirements to simultaneously solve the primal, adjoint, and design solutions to accelerate the process.

Chapter 7, introduces reduced-order modeling techniques and details the novel convergence acceleration technique that is the focus of this dissertation. The convergence acceleration technique is developed to accelerate convergence of fixed-point iterative solvers. The motivation for the reduced-order modeling approach is discussed, and its implementation is described for three different basis vector types.

Chapter 8 demonstrates the reduced-order model acceleration technique in a number of applications. First acceleration of a harmonic balance flow solution is presented. Next, acceleration of a continuous adjoint solver and discrete adjoint solver is considered. Finally, the acceleration technique is demonstrated to accelerate a nested design loop and a one-shot gradient-based optimizer for aerodynamic design. Chapter 9 summarizes the findings and presents interesting paths forward.

1.5 Related Published Works

This dissertation is based upon work published within the following peer reviewed journals and conference proceedings:

1. **Kaminsky, Andrew L.**, and Kivanc Ekici. "Sensitivity and stability derivative analysis using an efficient adjoint harmonic balance technique." In 54th AIAA aerospace sciences meeting, p. 0808. 2016.

2. **Kaminsky, Andrew L.**, Reza Djeddi, and Kivanc Ekici. "An Efficient Reduced-Order-Model for Accurate Projection of Adjoint Sensitivities." In 55th AIAA Aerospace Sciences Meeting, p. 0037. 2017.

3. Djeddi, Reza, **Andrew Kaminsky**, and Kivanc Ekici. "Convergence acceleration of fluid dynamics solvers using a reduced-order model." AIAA Journal 55.9 (2017): 3059-3071

4. **Kaminsky, Andrew L.**, Reza Djeddi, and Kivanc Ekici. "Convergence acceleration of continuous adjoint solvers using a reduced-order model." International Journal for Numerical Methods in Fluids 86, no. 9 (2018): 582-606.

5. **Kaminsky, Andrew L.**, and Kivanc Ekici. "Efficient prediction of forward aerodynamic sensitivities using a reduced-order model." In 2018 Multidisciplinary Analysis and Optimization Conference, p. 3744. 2018.

6. **Kaminsky, Andrew L.**, and Kivanc Ekici. "Reduced-order model-based convergence acceleration of reverse mode discrete adjoint solvers." Aerospace Science and Technology 93 (2019): 105334.

7. Thress, John F., **Andrew L. Kaminsky**, Reza Djeddi, and Kivanc Ekici. "One-shot Design Optimization Based on the Adjoint Harmonic Balance Technique." In AIAA AVIATION 2020 FORUM, p. 3128. 2020.

8. Thress, John, **Andrew L. Kaminsky**, Reza Djeddi, and Kivanc Ekici. "Monolithic One-Shot Optimization for Time-Periodic Flows Using Harmonic Balance." AIAA Journal 60, no. 6 (2022): 3539-3554.

At various points within this dissertation, material from these papers will be presented. Footnotes will be used to highlight this use.

Chapter 2

Governing Flow Equations

This chapter presents the governing flow equations. The flow equations are solved to characterize the performance of a given aerodynamic design through an objective metric, which is defined such that either maximization or minimization corresponds to an optimal design. Thus, in the context of aerodynamic design optimization the flow solution can be thought of as the primal problem.

Mathematical models with varying levels of approximation are considered to resolve the flow field. First, the Reynolds-averaged Navier–Stokes equations are introduced in Section 2.1. In Section 2.2, physical simplifications are made to derive the Euler equations which model inviscid flows. Finally, Section 2.3 details the harmonic balance approach [74] which extends these equations to efficiently solve unsteady flow solutions for periodic flows.

2.1 The Navier-Stokes Equations

In this work, the two-dimensional unsteady Navier–Stokes equations will be used to describe fluid motion through a time-dependent continuity equation for conservation of mass, two time-dependent conservation of momentum equations, and a time-dependent conservation of energy equation. In integral form, the Navier–Stokes equations are written as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho \, d\mathcal{V} + \oint_S \rho \boldsymbol{v} \cdot d\boldsymbol{S} = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho \boldsymbol{v} \, d\mathcal{V} + \oint_S \rho \boldsymbol{v} \left(\boldsymbol{v} \cdot d\boldsymbol{S}\right) = \int_{\mathcal{V}} \rho \boldsymbol{f_e} \, d\mathcal{V} - \oint_S p \cdot d\boldsymbol{S} + \oint_S \bar{\bar{\tau}} \cdot d\boldsymbol{S} \tag{2.2}$$

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho E \, d\mathcal{V} + \oint_S \rho E \boldsymbol{v} \cdot d\boldsymbol{S} = \oint_S k \boldsymbol{\nabla} T \cdot d\boldsymbol{S} + \int_{\mathcal{V}} \left(\rho \boldsymbol{f_e} \cdot \boldsymbol{v} + q_H\right) d\mathcal{V} + \oint_S \left(\bar{\bar{\sigma}} \cdot \boldsymbol{v}\right) \cdot d\boldsymbol{S}. \tag{2.3}$$

The Navier–Stokes equations describe the relationship between the velocity $\boldsymbol{v}$, pressure p, temperature T, and density ρ of a moving fluid. In two-dimensional flow these dependent

variables are functions of three independent variables, the two spatial coordinates (x and y) and time t.

The Navier–Stokes equations can also be written in strong conservation form

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \boldsymbol{Q} \,, \tag{2.4}$$

where the vector of conservation variables $\boldsymbol{U}$; the flux vectors $\boldsymbol{F}$, and $\boldsymbol{G}$; and the source vector $\boldsymbol{Q}$ are given by

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \,, \quad \boldsymbol{F} = \begin{bmatrix} \rho u - \rho \dot{f} \\ \rho u^2 + p - \tau_{xx} - \rho u \dot{f} \\ \rho uv - \tau_{xy} - \rho v \dot{f} \\ \rho uh - \tau_{xh} - \rho E \dot{f} \end{bmatrix} \,, \quad \boldsymbol{G} = \begin{bmatrix} \rho v - \rho \dot{g} \\ \rho uv - \tau_{yx} - \rho u \dot{g} \\ \rho v^2 + p - \tau_{yy} - \rho v \dot{g} \\ \rho vh - \tau_{yh} - \rho E \dot{g} \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} .$$

The conservation variables $\boldsymbol{U}$ represent the flow solution state. In the absence of Coriolis and body forces the source vector $\boldsymbol{Q}$ vanishes. In the equations above, $\dot{f}$ and $\dot{g}$ are the x and y components of the grid velocity respectively. From Eq. (2.4), it can be seen that beyond grid motion the flux results from two sources: convection and diffusion. The convective flux represents the transport of quantities due to the motion of the bulk flow. In the x-direction these terms are

$$\boldsymbol{F}_C = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uh \end{bmatrix} . \tag{2.5}$$

Each component of the convective flux contains the directional velocity to model the bulk motion in that direction. These terms grow as the velocity of the fluid increases, and they are zero when the fluid is at rest.

The vector of diffusive or viscous fluxes represents the macroscopic transport due to molecular agitation. The diffusive flux drives the flow toward equilibrium and uniformity. In the x-direction, the diffusive flux is

$$\boldsymbol{F}_D = \begin{bmatrix} 0 \\ -\tau_{xx} \\ -\tau_{xy} \\ -\tau_{xh} \end{bmatrix} . \tag{2.6}$$

Not all physical properties are diffusive. For instance, the mass of a fluid does not diffuse. Any displacement of mass would imply displacement of fluid particles which would by definition be convective motion. This property can be observed by the zero in the mass term of the diffusive fluxes. Fick [159] observed that the diffusive flux is proportional to the gradient of the concentration but moves in the opposite direction. For a Newtonian fluid, the

viscous shear stress is proportional to the velocity gradient, and the diffusive vector contains a viscous shear stress term τ for the momentum and energy equations. The viscous stress components in the x-direction are defined by

$$\tau_{xx} = \zeta \left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right] + 2\mu \frac{\partial u}{\partial x} \tag{2.7}$$

$$\tau_{xy} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right). \tag{2.8}$$

Here μ represents the dynamic viscosity and ζ is the second viscosity coefficient. Stokes [179] introduced the hypothesis that the two are related according to

$$\zeta + \frac{2}{3}\mu = 0. \tag{2.9}$$

This relation is called the bulk viscosity, and it represents the property responsible for energy dissipation in a fluid at uniform temperature during a constant rate change in volume [16]. With this relation, the stress tensor terms can be simplified so that they only depend on the dynamic viscosity term μ, which can be broken up into two components: μ_l the laminar viscosity and μ_t the eddy viscosity. The laminar viscosity is a fluid property that can be determined using the Sutherland formula (in Kelvin) [180]

$$\mu_l = \frac{1.45\, T^{3/2}}{T + 110} \cdot 10^{-6} \tag{2.10}$$

The eddy viscosity can be computed using turbulence models like the $K-\epsilon$ [118], $K-\omega$ [201], or Spalart-Allmaras [177] turbulence model. As discussed in Section 2.1.2, the Spalart-Allmaras turbulence model is employed in the present effort.

2.1.1 The Perfect Gas Model

The two-dimensional Navier–Stokes equations consist of four equations for the four conservative variables ρ, ρu, ρv, and ρE, which contain six unknown flow field variables ρ, u, v, E, p, and T. Since there are more two-more unknowns than equations, two equations of state are needed to close the mathematical model. The equations of state are developed by assuming the compressible fluid behaves as a perfect gas with constant specific heat ratios. This approximation can be used to define a relationship between the pressure and the conservation variables

$$p = (\gamma - 1)\rho \left[E - \frac{1}{2}(u^2 + v^2) \right], \tag{2.11}$$

and the temperature can be calculated through the ideal gas law [87],

$$p = \rho R_g T \tag{2.12}$$

where R_g is the gas constant per unit mass.

2.1.2 The Spalart-Allmaras Turbulence Model

Modeling turbulent flow presents another obstacle. A direct numerical simulation (DNS) of the time-dependent Navier–Stokes equations, requires the grid resolution scale with the Reynolds number according to $Re^{9/4}$[16] and CPU-times that scale according to Re^3. In engineering applications this cost is generally impractical, and the effects of turbulence typically need to be approximated using models of reduced complexity. The Reynolds-averaged Navier–Stokes (RANS) equations are the most widely applied approximation for industrial CFD applications [87].

The RANS equations decompose the flow variables into mean and fluctuating parts, which are then time averaged [167]. Inserting the decomposed variables into the Navier–Stokes equations Eq. (2.4) and averaging, results in the same equations for the mean variables with two additional terms. The first is the Reynolds-stress tensor

$$\bar{\bar{\tau}}_{ij} = -\bar{\rho}\widetilde{v_i'' v_j''}\,, \tag{2.13}$$

and the second is the turbulent heat-flux vector

$$F_{HF} = -\bar{\rho}\widetilde{h'' v''}. \tag{2.14}$$

Here v_i'' and v_j'' denote the density-weighted fluctuating parts of the velocity components. The over-bar and over-tilde represent ensemble and design weighted averages, respectively. To close the RANS equations, a large variety of turbulence models have been devised. In this work, a first-order closure approach is considered through the Spalart-Allmaras turbulence model.

The Spalart-Allmaras turbulence model is based on the Boussinesq assumption

$$\tau_{ij} = 2\mu_t \left(S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\rho k \delta_{ij}, \tag{2.15}$$

which states that the turbulent shear stress is linearly proportional to the mean strain rate [16]. The eddy viscosity is the proportionality constant and can be combined with the laminar viscosity to calculate the dynamic viscosity term.

The Spalart-Allmaras turbulence model is used solve for the eddy viscosity term. It can be implemented by concatenating an equation to the Navier–Stokes equations Eq. (2.4),

$$\boldsymbol{U} = \begin{bmatrix} \rho\tilde{\nu} \end{bmatrix}\,, \quad \boldsymbol{F} = \begin{bmatrix} \rho u\tilde{\nu} - \tau_{x\nu} - \rho\tilde{\nu}\dot{f} \end{bmatrix}\,, \quad \boldsymbol{G} = \begin{bmatrix} \rho v\tilde{\nu} - \tau_{y\nu} - \rho\tilde{\nu}\dot{g} \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} S_t \end{bmatrix}.$$

The eddy viscosity can then be found by [177]

$$\mu_t = \rho\tilde{\nu} f_{v1} \tag{2.16}$$

where

$$f_{vs} = 1 - \frac{\chi}{1 + \chi f_{v1}}; \quad \chi = \frac{\rho\tilde{\nu}}{\mu_l}. \tag{2.17}$$

The shear stress term in the x-direction is given by

$$\tau_{x\nu} = \frac{1}{\sigma}\left(\mu_l + \rho\tilde{\nu}\right)\frac{\partial \nu}{\partial x}. \tag{2.18}$$

The source term contains an eddy-viscosity production, turbulence destruction, and non-conservative diffusion term

$$S_t = C_{b1}\left(1 - f_{t2}\right)\tilde{S}\rho\tilde{\nu} - \rho\left[C_{w1} f_w - \frac{C_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{\rho C_{b2}}{\sigma}\left[\left(\frac{\partial\tilde{\nu}}{\partial x}\right)^2 + \left(\frac{\partial\tilde{\nu}}{\partial y}\right)^2\right]. \tag{2.19}$$

The production term is evaluated from

$$\tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}; \quad f_{vs} = 1 - \frac{\chi}{1 + \chi f_{v1}}; \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \quad \chi = \frac{\rho\tilde{n}\tilde{u}}{\mu_l},$$

where Ω is the vorticity magnitude. The terms controlling the destruction are found using

$$f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right); \quad g = r + c_{cw2}\left(r^6 - r\right); \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}\right).$$

2.2 The Euler Equations

Though the RANS equations are more accurate, it is often advantageous and appropriate to consider models of reduced complexity and computational cost. The Euler equations are an example of a physical simplification that can be made to model certain types of flow at considerably reduced computational cost and little impact on the solution accuracy. The Euler equations are obtained by neglecting the diffusive terms from the governing equations Eq. (2.4) yielding:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \boldsymbol{Q}\,. \tag{2.20}$$

The vector of conservation variables $\boldsymbol{U}$, the flux vectors $\boldsymbol{F}$ and $\boldsymbol{G}$, and the source vector $\boldsymbol{Q}$ are given by

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \rho u - \rho \dot{f} \\ \rho u^2 + p - \rho u \dot{f} \\ \rho uv - \rho v \dot{f} \\ \rho uh - \rho E \dot{f} \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} \rho v - \rho \dot{g} \\ \rho uv - \rho u \dot{g} \\ \rho v^2 + p - -\rho v \dot{g} \\ \rho vh - \rho E \dot{g} \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

By ignoring the viscous effects, the associated computational cost can be decreased considerably. The reduced complexity also facilitates sensitivity derivation, because sensitivity methods are often complicated by viscous effects and turbulence models.

2.2.1 Quasi-1D Euler Equations

The 2D Euler equations can be further simplified when applied to modeling asymmetric nozzle flow. Specifically, the 2D Euler Equations can be reformulated in a steady-state quasi-one-dimensional form

$$\mathbf{R}\left(\mathbf{U}, A\right) = \frac{\mathrm{d}}{\mathrm{d}x}\left(A\mathbf{F}\right) - \mathbf{Q}\frac{\mathrm{d}A}{\mathrm{d}x} = 0 \tag{2.21}$$

Here A denotes the cross-sectional area, and the vector of conservation variables $\mathbf{U}$, the convective flux vector $\mathbf{F}$, and the source term $\mathbf{Q}$ are given by

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho h u \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix}$$

Again, the pressure and total enthalpy are related to the conservation variables through

$$p = (\gamma - 1)\rho\left[E - \frac{1}{2}u^2\right]$$
$$h = \frac{\rho E + p}{\rho}$$

The quasi-one-dimensional Euler equations were considered in this work, because they are simple enough to have exact analytic solutions [62], while retaining complex flow features, like shocks. This makes the quasi-one-dimensional Euler equations well-suited to validation and verification studies, particularly continuous adjoint solvers and their boundary conditions.

2.3 The Harmonic Balance Equations

The harmonic balance technique is another simplification technique that can be used to efficiently specific types of unsteady flow problems. The harmonic balance technique was inspired by recognizing many flows of interest are temporally periodic due to vibration,

pitching, or cyclical operation. These cyclical flow behaviors in time can then be accurately represented by a Fourier series with spatially varying coefficients e.g.,

$$\boldsymbol{U}(x,y,t_i) = \boldsymbol{A}_0(x,y) + \sum_{n=0}^{N_H} \left[\boldsymbol{A}_n(x,y)\cos(\omega n t_i) + \boldsymbol{B}_n(x,y)\sin(\omega n t_i)\right]. \tag{2.22}$$

Here ω is the fundamental frequency and $\boldsymbol{A}_0,\ \boldsymbol{A}_n,$ and $\boldsymbol{B}_n$ are the Fourier coefficients of the conservation variables. For affordable simulation, the harmonic balance technique truncates this Fourier series to N_H harmonics. The Fourier coefficients $\tilde{\boldsymbol{U}}$ can be found from conserved flow variables at $2N_H+1$ sub-time levels $\boldsymbol{U}^*$ using the inverse discrete Fourier transform

$$\tilde{\boldsymbol{U}} = \boldsymbol{E}^{-1}\boldsymbol{U}^*. \tag{2.23}$$

The inverse discrete Fourier transform is given by

$$\boldsymbol{E}^{-1} = \frac{2}{2N_H+1}\begin{bmatrix} 1/2 & 1/2 & 1/2 & \dots & 1/2 \\ \cos\omega t_1 & \cos\omega t_2 & \cos\omega t_3 & \dots & \cos\omega t_{2N_H+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \cos N_H\omega t_1 & \cos N_H\omega t_2 & \cos N_H\omega t_3 & \dots & \cos N_H\omega t_{2N_H+1} \\ \sin\omega t_1 & \sin\omega t_2 & \sin\omega t_3 & \dots & \sin\omega t_{2N_H+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \sin N_H\omega t_1 & \sin N_H\omega t_2 & \sin N_H\omega t_3 & \dots & \sin N_H\omega t_{2N_H+1} \end{bmatrix}. \tag{2.24}$$

Alternatively, the sub-time level solutions $\boldsymbol{U}^*$ can be recovered from the Fourier coefficients $\tilde{\boldsymbol{U}}$ using the discrete Fourier transform

$$\boldsymbol{E} = \begin{bmatrix} 1 & \cos\omega t_1 & \dots & \cos N_H\omega t_1 & \sin\omega t_1 & \dots & \sin N_H\omega t_1 \\ 1 & \cos\omega t_2 & \dots & \cos N_H\omega t_2 & \sin\omega t_2 & \dots & \sin N_H\omega t_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \cos\omega t_{2N_H+1} & \dots & \cos N_H\omega t_{2N_H+1} & \sin\omega t_{2N_H+1} & \dots & \sin N_H\omega t_{2N_H+1} \end{bmatrix} \tag{2.25}$$

through

$$\boldsymbol{U}^* = \boldsymbol{E}\tilde{\boldsymbol{U}}. \tag{2.26}$$

The unsteady flow equations, Eq. (2.4), can be rewritten so that multiple sub-time levels are considered simultaneously, i.e.,

$$\frac{\partial \boldsymbol{U}^*}{\partial t} + \frac{\partial \boldsymbol{F}^*}{\partial x} + \frac{\partial \boldsymbol{G}^*}{\partial y} = \boldsymbol{Q}^*\,. \tag{2.27}$$

$\boldsymbol{F}^*$ and $\boldsymbol{G}^*$ are the flux vectors evaluated at $\boldsymbol{U}^*$. The $2N_H+1$ sub-time levels are coupled through the time derivative term

$$\frac{\partial}{\partial t}\boldsymbol{U}^* = \sum_{n=1}^{N_H} \left[-\omega n \cdot \boldsymbol{A}_n(x,y)\sin(\omega n t) + \omega n \cdot \boldsymbol{B}_n(x,y)\cos(\omega n t)\right]. \tag{2.28}$$

Recalling the inverse discrete Fourier transform $\boldsymbol{E}^{-1}$, the time derivative term may be written in matrix form as

$$\frac{\partial \boldsymbol{U}^*}{\partial t} = \frac{\partial \boldsymbol{E}}{\partial t}\tilde{\boldsymbol{U}}. \tag{2.29}$$

The inverse discrete Fourier transform, Eq. (2.24), can then be used so that the time derivative term is dependent on the sub-time level solutions

$$\frac{\partial \boldsymbol{U}^*}{\partial t} = \frac{\partial \boldsymbol{E}}{\partial t}\boldsymbol{E}^{-1}\boldsymbol{U}^* = \boldsymbol{\mathcal{D}}\boldsymbol{U}^*. \tag{2.30}$$

where $\boldsymbol{\mathcal{D}}$ is known as the pseudo-spectral operator. The harmonic balance equations can then be written as

$$\boldsymbol{\mathcal{D}}\boldsymbol{U}^* + \frac{\partial \boldsymbol{F}^*}{\partial x} + \frac{\partial \boldsymbol{G}^*}{\partial y} + \frac{\partial \boldsymbol{H}^*}{\partial z} = \boldsymbol{Q}^* \,. \tag{2.31}$$

The pseudo-spectral operator removes the time dependence of the Navier-Stokes equations so that the harmonic balance equation is mathematically steady. A pseudo-time term can then be added and used to march Eq. (2.31) to steady state

$$\frac{\partial \boldsymbol{U}^*}{\partial \mathcal{T}} + \boldsymbol{\mathcal{D}}\boldsymbol{U}^* + \frac{\partial \boldsymbol{F}^*}{\partial x} + \frac{\partial \boldsymbol{G}^*}{\partial y} + \frac{\partial \boldsymbol{H}^*}{\partial z} = \boldsymbol{Q}^* \,. \tag{2.32}$$

Chapter 3

Numerical Approach

3.1 Non-Dimensionalization

It is convenient to work in a dimensionless form of the governing flow equations [90]. The non-dimensional form provides unitless results and isolates parameters which makes comparisons with external sources easier. Within this work, the non-dimensionalization is based on a reference pressure, temperature, and length term which can be used to calculate additional reference quantities [90], given as

$$\rho_{ref} = \frac{p_{ref}}{R_{gas}T_{ref}}; \quad V_{ref} = \sqrt{R_{gas}T_{ref}}; \quad a_{ref} = \sqrt{\gamma R_{gas}T_{ref}}; \quad \omega_{ref} = \frac{V_{ref}}{L_{ref}}. \tag{3.1}$$

With these reference values defined the non-dimensionalized variables are:

$$\hat{x} = \frac{x}{L_{ref}}; \quad \hat{y} = \frac{y}{L_{ref}}; \quad \hat{z} = \frac{z}{L_{ref}} \quad \hat{t} = \frac{t}{L_{ref}/V_{ref}} \tag{3.2}$$

$$\hat{u} = \frac{u}{V_{ref}}; \quad \hat{v} = \frac{v}{V_{ref}}; \quad \hat{w} = \frac{w}{V_{ref}}; \quad \hat{\omega} = \frac{\omega}{\omega_{ref}}; \tag{3.3}$$

$$\hat{\mu} = \frac{\mu}{\rho_{ref}V_{ref}L_{ref}}; \quad \hat{\rho} = \frac{\rho}{\rho_{ref}}; \quad \hat{p} = \frac{p}{p_{ref}}; \quad \hat{T} = \frac{T}{T_{ref}} \tag{3.4}$$

where hats denote dimensionless quantities.

3.2 Spatial Discretization

Within this work two in-house flow solvers *cascade* [92, 93] and *external* [37, 38, 88] are used. The *cascade* solver employs a vertex-centered spatial discretization method [97]. In this approach, the control volume is formed by taking the union of cells that meet at that vertex. In two-dimensional cases the control volume is made of four cells as shown in Fig 3a. The *external* solver employs a cell-centered spatial discretization approach to define control volumes that coincide with the cells made by the grid, as shown in Fig. 3b. In this approach,

the flow variables are stored at the centroids of the control volumes and the fluxes are calculated on the control volume faces. Solvers with differing spatial discretization schemes are used to evaluate the acceleration technique introduced in Chapter 7 and demonstrate its versatility. However, in the interest of conciseness only the cell-centered approach will be outlined here.

3.2.1 Cell-Centered Discretization

Convective Fluxes

A central scheme with artificial dissipation known as the Jameson-Schmidt-Turkel (JST) scheme [103] is used to compute the convective fluxes. In the JST scheme, the convective flux is computed at each control volume face using an average of the flow variables to either side of the face. Denoting face surfaces using 1/2-indexing, the convective flux can be written as

$$(\boldsymbol{F}_c \Delta S)_{i+1/2,j,k} \approx \boldsymbol{F}_c \left(\boldsymbol{U}_{i+1/2,j,k} \right) \Delta S_{i+1/2,j,k} \tag{3.5}$$

where the conservative variables at the face are defined as the arithmetic average of the values of the two adjacent cells,

$$\boldsymbol{U}_{i+1/2,j,k} = \frac{1}{2} \left(\boldsymbol{U}_{i,j,k} + \boldsymbol{U}_{i+1,j,k} \right). \tag{3.6}$$

The central scheme suffers from odd-even decoupling which manifests itself as oscillations in the solution due to the generation of two independent solutions of the discretized equations. Artificial dissipation is typically added to stabilize the solution. Adding artificial dissipation, the convective flux becomes

$$(\boldsymbol{F}_c \Delta S)_{i+1/2,j,k} \approx \boldsymbol{F}_c \left(\boldsymbol{U}_{i+1/2,j,k} \right) \Delta S_{i+1/2,j,k} - \boldsymbol{D}_{i+1/2,j,k}\,, \tag{3.7}$$

where $\boldsymbol{D}_{i+1/2,j,k}$ is the artificial dissipation flux at the cell face. In the traditional JST scheme, the artificial dissipation flux consists of a blend of the second- and fourth-order difference operators

$$\begin{aligned} \boldsymbol{D}_{i+1/2,j,k} = & \lambda_{i+1/2,j,k} \Big[\epsilon^{(2)}_{i+1/2,j,k} \left(\boldsymbol{U}_{i+1,j,k} - \boldsymbol{U}_{i,j,k} \right) \\ & - \epsilon^{(4)}_{i+1/2,j,k} \left(\boldsymbol{U}_{i+2,j,k} - 3\boldsymbol{U}_{i+1,j,k} + 3\boldsymbol{U}_{i,j,k} - \boldsymbol{U}_{i-1,j,k} \right) \Big]. \end{aligned} \tag{3.8}$$

The dissipation is scaled by the sum of the spectral radii of the convective flux Jacobian in each coordinate direction

$$\lambda_{i+1/2,j,k} = \lambda^{i}_{i+1/2,j,k} + \lambda^{j}_{i+1/2,j,k} + \lambda^{k}_{i+1/2,j,k}\,. \tag{3.9}$$

The spectral radius at the face is taken as the average of the spectral radii of the two adjacent cells, where the spectral radius is evaluated using the formula

$$\lambda^i_{i,j,k} = (|V| + c)\,\Delta S \tag{3.10}$$

where V is the contravariant velocity, and c the speed of sound. The coefficients $\epsilon^{(2)}$ and $\epsilon^{(4)}$ are used to tune the artificial dissipation depending on the flow conditions, according to

$$\epsilon^{(2)}_{i,j,k} = k^{(2)} \max\left(\nu_{i,j,k}, \nu_{i+1,j,k}\right) \tag{3.11}$$

$$\epsilon^{(4)}_{i,j,k} = \max\left[0, \left(k^{(0)} - \epsilon^{(2)}_{i+1/2,j,k}\right)\right] \tag{3.12}$$

where $\nu_{i,j,k}$ is a pressure switch given by

$$\nu_{i,j,k} = \frac{|p_{i-1,j,k} - 2p_{i,j,k} + p_{i+1,j,k}|}{p_{i-1,j,k} + 2p_{i,j,k} + p_{i+1,j,k}}\,. \tag{3.13}$$

The switch is used to adjust the impact of the second- and fourth-order artificial dissipation terms. In smooth flow regions, the second-order term is turned off while the fourth-order term is active to damp oscillations caused by odd-even decoupling. In regions with discontinuities, the second-order term is switched on and the fourth-order term is switched off to prevent strong oscillations. Values for these constants are typically in the ranges $0.25 \leq k^{(2)} \leq 0.5$ and $0.008 \leq k^{(4)} \leq 0.032$.

The solution accuracy can be improved by reducing the numerical dissipation. The JST scheme can be modified to act more like an upwind scheme which allows the artificial dissipation to be limited. The matrix dissipation scheme [181] uses a matrix consisting of the convective flux Jacobian instead of the spectral radius to limit the dissipation terms. As a result, each equation is scaled by its corresponding eigenvalue and Eq. (3.8) becomes

$$\begin{aligned}\boldsymbol{D}_{i+1/2,j,k} =& |\boldsymbol{A}_c|_{i+1/2,j,k}\Big[\epsilon^{(2)}_{i+1/2,j,k}\left(\boldsymbol{U}_{i+1,j,k} - \boldsymbol{U}_{i,j,k}\right) \\ &- \epsilon^{(4)}_{i+1/2,j,k}\left(\boldsymbol{U}_{i+2,j,k} - 3\boldsymbol{U}_{i+1,j,k} + 3\boldsymbol{U}_{i,j,k} - \boldsymbol{U}_{i-1,j,k}\right)\Big]\,.\end{aligned} \tag{3.14}$$

where

$$|\boldsymbol{A}_c| = T|\lambda_c \Delta S|T^{-1}. \tag{3.15}$$

$\boldsymbol{A}_c = \partial \boldsymbol{F}_c / \partial \boldsymbol{U}$ is the convective flux Jacobian, T and T^{-1} are the right and left eigenvectors and λ_c is the diagonal matrix of eigenvalues. Blazek [16] presents a thorough derivation of the eigenvalues and eigenvectors for 2D and 3D cases. The JST scheme with artificial dissipation Eq 3.14 provides a simple to implement and computationally efficient approach for convective flux calculation.

Viscous Fluxes

To consider viscous flow fields, a numerical scheme must be developed for the viscous fluxes as well. Like the convective flux scheme, the viscous flux scheme also follows a central difference-based stencil [139]. Therefore, the values at the faces again result from the average of the values of the adjacent cell centers. However, the viscous flux terms contain velocity derivative terms that need to be evaluated at the cell faces. These derivative terms can be evaluated using either finite differences or Green's theorem.

Green's theorem relates the volume integral of the first derivative of a variable to the surface integral of the same variable. The derivative of u in the x-direction is approximated as

$$\frac{\partial u}{\partial x} = \frac{1}{\mathcal{V}'} \iint_{\mathcal{V}'} \frac{\partial u}{\partial x} dxdy = \frac{1}{\mathcal{V}'} \int_{\mathcal{V}'} u dy \approx \frac{1}{\mathcal{V}'} \sum_{m=1}^{N_F} u_m S'_{x,m} \tag{3.16}$$

where $\mathcal{V}'$ is the volume of an auxiliary cell shown in Fig. 4 and N_F is the number of faces of the auxiliary cell. It can be seen from the figure that the values at the left and right faces are already known, but values on the upper and lower faces need to be determined. The value of u at location $i + 1/2, j + 1/2$ and $i + 1/2, j - 1/2$ are taken as the average of the four surrounding cells, for example

$$u_{i+1/2,j+1/2} = \frac{1}{4} \left(u_{i,j} + u_{i+1,j} + u_{i,j+1} + u_{i+1,j+1} \right) . \tag{3.17}$$

3.3 Temporal Discretization

The temporal domain must also be discretized. Within this work, Jameson's multi-stage Runge–Kutta scheme [103] is used to time-march the discretized equations. The multi-stage technique divides the update into a number of steps. Both the Euler and Navier-Stokes equations can be summarized as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{R}^n_{i,j,k} = 0 \tag{3.18}$$

where $\boldsymbol{R}^n_{i,j,k}$ is the residual at the current time level. A 5-stage Runge–Kutta scheme can be used to update the solution in successive steps as described by

$$\boldsymbol{U}^{(0)}_{i,j,k} = \boldsymbol{U}^n_{i,j,k} \tag{3.19}$$

$$\boldsymbol{U}^{(1)}_{i,j,k} = \boldsymbol{U}^{(0)}_{i,j,k} - \alpha_1 \frac{\Delta t_{i,j,k}}{\mathcal{V}_{i,j,k}} \left[\boldsymbol{R}^{(0)}_c - \boldsymbol{R}^{(0)}_d\right]_{i,j,k} \tag{3.20}$$

$$\boldsymbol{U}^{(2)}_{i,j,k} = \boldsymbol{U}^{(0)}_{i,j,k} - \alpha_2 \frac{\Delta t_{i,j,k}}{\mathcal{V}_{i,j,k}} \left[\boldsymbol{R}^{(1)}_c - \boldsymbol{R}^{(0)}_d\right]_{i,j,k} \tag{3.21}$$

$$\boldsymbol{U}^{(3)}_{i,j,k} = \boldsymbol{U}^{(0)}_{i,j,k} - \alpha_3 \frac{\Delta t_{i,j,k}}{\mathcal{V}_{i,j,k}} \left[\boldsymbol{R}^{(2)}_c - \boldsymbol{R}^{(2,0)}_d\right]_{i,j,k} \tag{3.22}$$

$$\boldsymbol{U}^{(4)}_{i,j,k} = \boldsymbol{U}^{(0)}_{i,j,k} - \alpha_4 \frac{\Delta t_{i,j,k}}{\mathcal{V}_{i,j,k}} \left[\boldsymbol{R}^{(3)}_c - \boldsymbol{R}^{(2,0)}_d\right]_{i,j,k} \tag{3.23}$$

$$\boldsymbol{U}^{(5)}_{i,j,k} = \boldsymbol{U}^{(0)}_{i,j,k} - \alpha_5 \frac{\Delta t_{i,j,k}}{\mathcal{V}_{i,j,k}} \left[\boldsymbol{R}^{(4)}_c - \boldsymbol{R}^{(4,2)}_d\right]_{i,j,k} \tag{3.24}$$

where

$$\boldsymbol{R}^{(2,0)}_d = \beta_3 \boldsymbol{R}^{(2)}_d + (1-\beta_3)\boldsymbol{R}^{(0)}_d \tag{3.25}$$

$$\boldsymbol{R}^{(4,2)}_d = \beta_5 \boldsymbol{R}^{(4)}_d + (1-\beta_5)\boldsymbol{R}^{(2,0)}_d \tag{3.26}$$

This is a hybrid Runge–Kutta scheme, because the viscous terms on are only re-evaluated on the odd stages. Mavriplis and Jameson [147] outlined this particular hybrid scheme called the (5,3)-scheme since it is a 5-stage Runge-Kutta integration with dissipation calculated during three of the stages. Using optimized stage coefficients, the hybrid schemes are as robust as the basic multistage schemes but more computationally efficient. The stage α_m and blending β_m coefficients for the (5,3)-scheme are given in Table 1. Hybrid Runge–Kutta schemes are also memory efficient because only the zeroth solution and final residual must be stored. This is in contrast to classical Runge–Kutta schemes which require storage of all the intermediate solutions. The five-stage Runge–Kutta update for the two-dimensional RANS equations can then be conveniently written as:

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \boldsymbol{R}\left(\boldsymbol{U}^n\right), \tag{3.27}$$

Explicit methods are limited by a maximum timestep. This timestep $\Delta t_{i,j,k}$ must satisfy the Courant-Friedricks-Lewy (CFL) condition [31] for each control volume to remain stable. For explicit time-stepping schemes the timestep must be smaller than the time required to transport information across the spatial discretization stencil. Therefore, the maximum timestep can be found by

$$\Delta t_i = CFL \frac{\mathcal{V}}{\lambda_c + 4\lambda_v} \tag{3.28}$$

where λ_c is the spectral radius of the convective flux and λ_v is the viscous spectral radius:

$$\lambda_v = \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right)\left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right)\frac{\Delta S_i}{\mathcal{V}_{i,j,k}}\,. \tag{3.29}$$

Here Pr_L and Pr_T are the laminar and turbulent Prandtl numbers, respectively. For cases considering high reduced frequencies or numerous harmonics, Van der Weide et al. [193] added the excitation frequency and number of harmonics N_H into the maximum pseudo-time-step calculation

$$\Delta\tau = CFL\frac{\mathcal{V}}{\lambda_c + \omega N_H \mathcal{V}} \tag{3.30}$$

This addition has been found to stabilize performance. The maximum stable CFL number for the (5,3) Hybrid Runge–Kutta scheme with a central spatial discretization is 3.5 [183].

3.4 Discrete Boundary Conditions

3.4.1 Solid Wall Boundaries

Inviscid flow

The fluid in inviscid flow simulations is considered to slip over the surface at solid wall boundaries. In this slip condition the velocity is tangent to the solid boundary surface. The velocity component normal to the wall is set to zero so that the fluid does not penetrate the wall. As a result, the convective flux vector is reduced to only the pressure terms at the wall. As illustrated in Fig. 5, Ghost cells can be used to satisfy the boundary conditions without making special accommodations for boundary cells.

Using the ghost cell indices from the figure, the density values of the ghost cells are mirrored across the boundary according to

$$\rho_0 = \rho_1 \tag{3.31}$$

$$\rho_{-1} = \rho_2 \tag{3.32}$$

In the cell-centered scheme, the conserved values are evaluated at the cell centroids. The solid wall appears at the cell face $\boldsymbol{S}$ so the conserved values at the wall must be obtained through extrapolation from the interior of the domain,

$$\boldsymbol{U}_w = \frac{3}{2}\boldsymbol{U}_1 - \frac{1}{2}\boldsymbol{U}_2. \tag{3.33}$$

Now the velocity and more importantly the normal velocity component can be determined at the wall. For a moving wall, the normal velocity can be written as

$$V_{normal} = \left(u_w - \dot{f}\right)\eta_x + \left(v_w - \dot{g}\right)\eta_y. \tag{3.34}$$

The velocity of the ghost cells can be found from an average of the wall and interior domain. However, the normal component of the velocity at the wall needs to be removed so that the flow does not penetrate the solid wall boundary. Therefore, the velocity components of the first row of ghost cells are

$$u_0 = 2\left(u_w - \eta_x V_{normal}\right) - u_1 \tag{3.35}$$
$$v_0 = 2\left(v_w - \eta_y V_{normal}\right) - v_1. \tag{3.36}$$

The second row of ghost cells can be found similarly

$$u_{-1} = 2\left(u_w - \eta_x V_{normal}\right) - u_2 \tag{3.37}$$
$$v_{-1} = 2\left(v_w - \eta_y V_{normal}\right) - v_2. \tag{3.38}$$

Viscous flow

Viscous flow at a solid wall boundary must satisfy the no-slip condition, which requires the relative velocity between the surface and the fluid to equal zero. The velocity components of the first and second rows of the ghost cells are

$$u_0 = 2\dot{f} - u_1 \tag{3.39}$$
$$v_0 = 2\dot{g} - v_1 \tag{3.40}$$
$$u_{-1} = 2\dot{f} - u_2 \tag{3.41}$$
$$v_{-1} = 2\dot{g} - v_2 \tag{3.42}$$
$$\tag{3.43}$$

For an adiabatic wall, the density is found in the same manner as the inviscid case, and the total energy in the first ghost cell can be calculated as

$$(\rho E) = \frac{p_0}{\gamma - 1} + \frac{1}{2\rho_0}\left[(\rho u)_0^2 + (\rho v)_0^2\right]. \tag{3.44}$$

The Spalart-Allmaras working variable is also set to zero at a solid wall, so the ghost cell value is set according to

$$\tilde{\nu}_0 = -\tilde{\nu}_1 \tag{3.45}$$

3.4.2 Far Field Boundaries

In numerical simulations, it is necessary to bound the domain using an artificial far field boundary condition. This boundary condition must have no noticeable effect on the flow solution when compared with an infinite domain. Subsonic and transonic flow problems are particularly sensitive to the far field boundary conditions. To define boundary conditions, we must determine the number of conditions of physical origin that must be imposed and how to define the remaining variables.

The far field values are often treated using characteristic variables, since the governing equations are dominated by hyperbolic propagation [86]. The characteristic variables or the Riemann invariants express the propagation properties of the flow in a straightforward way. Since the boundary behavior of one-dimensional flow is readily extended to higher dimensions, consider the development of Riemann invariant boundary conditions using the eigenvalues of the one-dimensional convective flux Jacobian:

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} u - c \\ u \\ u + c \end{bmatrix} \tag{3.46}$$

The sign of the eigenvalue determines whether information is transported into or out of the computational domain. For a boundary node subject to subsonic and supersonic flow the situation is depicted in Fig. 6a and 6b, respectively. Table 2 lists the number of physical and extrapolated boundary conditions for each boundary type.

The corresponding left eigenvectors

$$\mathbf{L}^{-1} = \begin{bmatrix} 0 & 1 & -1/\rho c \\ 1 & 0 & -1/c^2 \\ 0 & 1 & 1/\rho c \end{bmatrix} \tag{3.47}$$

can be used to define a set of characteristic equations

$$du - \frac{dp}{\rho c} = 0 \tag{3.48}$$

$$d\rho - \frac{dp}{c^2} = 0 \tag{3.49}$$

$$du + \frac{dp}{\rho c} = 0 \tag{3.50}$$

Riemann invariants or characteristic variables are obtained by integrating Eqs. (3.48) and (3.50) [90]

$$R^- = u - \int \frac{dp}{\rho c} \tag{3.51}$$

$$R^+ = u + \int \frac{dp}{\rho c} \,. \tag{3.52}$$

Isentropic relations

$$p = k\rho^\gamma \quad \text{and} \quad c^2 = k\gamma\rho^{\gamma-1} \tag{3.53}$$

can be used to simplify the Riemann invariants to

$$R^- = u - \frac{2c}{\gamma - 1} \tag{3.54}$$

$$R^+ = u + \frac{2c}{\gamma - 1} \, . \tag{3.55}$$

The two equations can then be combined to solve for the velocity and speed of sound at the boundary by

$$u = \frac{1}{2}\left(R^+ + R^-\right) \tag{3.56}$$

$$c = \frac{\gamma - 1}{4}\left(R^+ - R^-\right) \tag{3.57}$$

The remaining characteristic equation, Eq. (3.49) is integrated directly

$$\frac{p}{\rho^\gamma} = \text{Constant} \tag{3.58}$$

Eqs. (3.56) - (3.58) are used to define the far field flow conditions for one-dimensional flow. For two- and three-dimensional flow, each dimension adds a tangential velocity component. The tangential velocity is constant across the Riemann invariant waves. This can be used to define the additional boundary condition.

3.5 Convergence Acceleration Techniques

The computational fluid dynamic model outlined here provides accurate flow solutions that convey insight into flow fields of interest. However, CFD's utility is limited by its computational cost. Within the CFD literature, many techniques have been proposed to accelerate the solution. Below a few of the more common techniques and their implementation within the external flow solver are discussed.

3.5.1 Local Time-Stepping

The local time-stepping approach achieves acceleration by using the maximum timestep for each individual control volume, rather than a global timestep permitted by all control volumes. This acceleration technique is only viable for steady-state solutions, because the approach destroys the time-accuracy since the control volumes are advanced by different timesteps.

3.5.2 Residual Smoothing

The residual smoothing technique aims to increase the maximum possible time-step by introducing a certain amount of implicitness into the explicit scheme. Introduced by Jameson and Baker [101], the central implicit residual smoothing scheme replaces the residual at each point with a weighted average of the residuals from neighboring points, according to

$$-\epsilon^i \boldsymbol{R}^*_{i-1,j,k} + \left(1 + 2\epsilon^i\right) \boldsymbol{R}^*_{i,j,k} - \epsilon^i \boldsymbol{R}^*_{i+1,j,k} = \boldsymbol{R}_{i,j,k} \tag{3.59}$$

$$-\epsilon^j \boldsymbol{R}^{**}_{i,j-1,k} + \left(1 + 2\epsilon^j\right) \boldsymbol{R}^{**}_{i,j,k} - \epsilon^j \boldsymbol{R}^{**}_{i,j+1,k} = \boldsymbol{R}^*_{i,j,k} \tag{3.60}$$

$$-\epsilon^k \boldsymbol{R}^{***}_{i,j,k-1} + \left(1 + 2\epsilon^k\right) \boldsymbol{R}^{***}_{i,j,k} - \epsilon^k \boldsymbol{R}^{***}_{i,j,k+1} = \boldsymbol{R}^{**}_{i,j,k} \tag{3.61}$$

where $\boldsymbol{R}^*_{i,j,k}$, $\boldsymbol{R}^{**}_{i,j,k}$, and $\boldsymbol{R}^{***}_{i,j,k}$ are the smoothed residuals in the i-, j-, and k-directions. The residual smoothing scheme results in a tridiagonal matrix that must be numerically inverted. Relative to other implicit schemes it has the advantage that it requires less computational effort. It has an associated cost of about 20% more effort, but the original time-step can be increased by a factor of 2-3. The smoothing coefficients are defined as functions of the spectral radii, and several formulations are provided by Turkel et al [192]. The residual smoothing technique damps high-frequency error components of the residual, which is advantageous to other acceleration techniques like the multigrid method and the reduced-order model-based acceleration technique.

3.5.3 Multigrid

The multigrid technique accelerates convergence to steady state by using solutions on coarsened grids to update the solution on the grid of interest. The multigrid technique arises from the observation that most iterative time-stepping schemes efficiently reduce high-frequency components of the solution error but not low-frequency components. This results in solvers with rapid convergence during an initial phase where the high-frequency components are damped, followed by slow convergence, as shown in Fig. 7. Multigrid helps address this issue because low-frequency components on the fine grid become high-frequency components on the coarse grid that can be reduced efficiently. Through this process the error is quickly reduced over the entire spectrum, providing significant acceleration.

The basic multigrid scheme begins by creating the coarsened grids. This is typically done by coarsening the grid evenly in all coordinate directions by removing every other point, as shown in Fig. 8. Following grid creation, the multigrid scheme starts from a known solution $\boldsymbol{U}^n$ on the fine grid, a new solution $\boldsymbol{U}^{n+1}$ is obtained after one timestep of the technique outlined in Section 3.3. A new residual is evaluated with this solution. The multigrid scheme then seeks to improve the new solution $\boldsymbol{U}^{n+1}$ using a coarse grid through the following steps. For cell-centered schemes, the solution and residual are transferred from the fine grid (denoted by the subscript h) to the coarse grid (denoted by the subscript $2h$) using an interpolation operator,

$$\boldsymbol{U}^{(0)}_{2h} = I^{2h}_h \boldsymbol{U}^{n+1}_h. \tag{3.62}$$

The interpolation operator transfers the solution on the fine grid to the coarse grid through a volume weighted interpolation. In 2D the interpolation is

$$\left(\boldsymbol{U}_{2h}^{(0)}\right)_{i,j} = \frac{\left(\boldsymbol{U}_h^{n+1}\right)_{i,j}\mathcal{V}_{i,j} + \left(\boldsymbol{U}_h^{n+1}\right)_{i+1,j}\mathcal{V}_{i+1,j} + \left(\boldsymbol{U}_h^{n+1}\right)_{i,j+1}\mathcal{V}_{i,j+1} + \left(\boldsymbol{U}_h^{n+1}\right)_{i+1,j+1}\mathcal{V}_{i+1,j+1}}{\mathcal{V}_{i,j} + \mathcal{V}_{i+1,j} + \mathcal{V}_{i,j+1} + \mathcal{V}_{i+1,j+1}} \tag{3.63}$$

The residuals are restricted by summing the residuals from the fine grid cells that make up the new coarse grid cell. Following transfer to the coarse grid, new volumes and timesteps must be calculated for the coarse grid. The coarse grids allow larger timesteps because the cell volumes are larger. For a cell-centered scheme the volumes are restricted as shown in Fig. 9a.

To retain the accuracy of the fine grid on the coarse grid, a forcing function $\boldsymbol{Q}_F$ is calculated as the difference between the restricted residuals and the residual from the restricted working variables on the coarse grid,

$$(\boldsymbol{Q}_F)_{2h} = I_h^{2h}\boldsymbol{R}_h^{n+1} - \boldsymbol{R}_{2h}^{(0)}. \tag{3.64}$$

The solution is then evaluated on the coarse grid in the same manner as on the fine grid, with the addition of the forcing function added to the coarse residual,

$$(\boldsymbol{R_F})_{2h} = \boldsymbol{R}_{2h} + (\boldsymbol{Q}_F)_{2h}. \tag{3.65}$$

Therefore, using the same multistage time-stepping scheme, from Eq (3.19), the coarse grid solution can be obtained from

$$\boldsymbol{U}_{2h}^{(k)} = \boldsymbol{U}_{2h}^{(0)} + \alpha_k \frac{\Delta t_{2h}}{\mathcal{V}_{2h}}\left[\boldsymbol{R}_{2h}^{(k-1)} + (\boldsymbol{Q}_F)_{2h}\right], \quad k = 1, ..., m \tag{3.66}$$

The solution on the coarse grid features less computational effort due to the reduced node count and utilization of lower-order solution schemes.

After the time-step is carried out, the correction on the coarse grid is calculated as the difference between the coarse grid solution and the initial approximation,

$$\delta\boldsymbol{U}_{2h} = \boldsymbol{U}_{2h}^{n+1} - \boldsymbol{U}_h^{(0)} \tag{3.67}$$

The coarse grid correction is then prolongated to the finer grid via

$$\boldsymbol{U}_h^{+} = \boldsymbol{U}_h^{n+1} + I_{2h}^h \delta\boldsymbol{U}_{2h} \tag{3.68}$$

where I_{2h}^h is the prolongation operator. For a cell-centered scheme, a zeroth-order prolongation is accomplished by adding the coarse grid residual to the constituent fine grid cells

$$\left(\boldsymbol{U}_h^{+}\right)_{i+1,j} = \left(\boldsymbol{U}_h^{n+1}\right)_{i+1,j} + (\delta\boldsymbol{U}_{2h})_{i,j} \tag{3.69}$$

Fig. 9b shows this simple prolongation. Multigrid is one of the most popular acceleration techniques for CFD solvers. It was originally developed by Brandt [19] for elliptic partial differential equations and Jameson later applied the method to the Euler equations [96]. Application of multigrid can offer considerable acceleration as will be shown in Section 5.2.1.

Chapter 4

Sensitivity Equations and Analysis

This chapter covers the second phase of the design cycle, the sensitivity calculation. The chapter begins by outlining the general form of sensitivity equation and introduces the direct or forward method in Section 4.1. Next, the adjoint sensitivity approach is introduced through its analytic derivation 4.2. Following the analytic derivations, Section 4.3 introduces automatic differentiation which can be employed as a practical approach for calculating sensitivities of CFD solvers. Validation and verification of the sensitivity calculation methods implemented in this work can be found in Chapter 5.

4.1 Derivation of the Direct Approach

The objective of sensitivity analysis is to calculate the sensitivity, or dependence, of a function or parameter of interest, called the objective function, with respect to a number of other parameters, referred to as the design variables. The objective function $\boldsymbol{J}$ (also sometimes referred to as the cost function or loss metric) is typically defined such that either the maximum or minimum corresponds to an optimal design. In the practice of aerodynamic design, the cost function frequently depends not only on the design variables $\boldsymbol{\beta}$ but also on the state variables $\boldsymbol{U}$, as well. These state variables can also be dependent on the design variables, causing the objective function to be both directly and indirectly dependent upon the design variables. Consequently, the objective function can be expressed as

$$\boldsymbol{J} = \boldsymbol{J}\left(\boldsymbol{\beta}, \boldsymbol{U}\left(\boldsymbol{\beta}\right)\right) . \tag{4.1}$$

The chain rule can be used to derive the total derivative of the objective function with respect to the design variables of interest as

$$\frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\beta}} + \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{U}} \frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}\boldsymbol{\beta}} . \tag{4.2}$$

Here the partial derivative terms on the right-hand side are easily determined through finite differencing, i.e., by perturbing each independent variable and re-evaluating the dependent

variable. However, the total derivative of the state vector with respect to the design variable vector requires a full solution of the governing equations for each component of $\boldsymbol{\beta}$ [144].

An alternative form for the sensitivity equation can developed by considering the governing equations in residual form,

$$\boldsymbol{R}(\boldsymbol{\beta}, \boldsymbol{U}(\boldsymbol{\beta})) = 0 \tag{4.3}$$

which states that regardless of the flow conditions the residual will equal zero at the converged solution. As a result, the total derivative of the residual with respect to the design variable must also be zero,

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} \frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}\boldsymbol{\beta}} = 0\,. \tag{4.4}$$

Eq. (4.4) can be rearranged to isolate the derivative of the state vector with respect to the design variable, i.e.,

$$\frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}\boldsymbol{\beta}} = -\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^{-1} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}}\,. \tag{4.5}$$

Substituting Eq. (4.5) into Eq. (4.2) yields

$$\frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\beta}} \underbrace{- \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{U}} \overbrace{\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^{-1}}^{\mathrm{d}\boldsymbol{U}/\mathrm{d}\boldsymbol{\beta}}}_{\boldsymbol{\psi}^T} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}}\,. \tag{4.6}$$

The direct approach can be employed to solve Eq. (4.6) by calculating $\mathrm{d}\boldsymbol{U}/\mathrm{d}\boldsymbol{\beta}$ using Eq. (4.5). Solving for $\mathrm{d}\boldsymbol{U}/\mathrm{d}\boldsymbol{\beta}$ requires the solution of the matrix equation for each design variable β_i. It should be noted that a change in the design variable doesn't affect the Jacobian matrix $\partial \boldsymbol{R}/\partial \boldsymbol{U}$, so if the Jacobian can be calculated explicitly, solving for multiple design variables by back substitution would be relatively inexpensive. However, for the large iterative problems found in computational fluid dynamics problems the Jacobian generally is not factored explicitly. Instead, the system of equations requires an iterative solution with a computational cost that is typically slightly more than that of flow equations. Multiplying this cost by the number of design variables causes the direct approach to become unsuitable as the number of design variables increases. The forward approach is applicable when only a few design variables are of interest or if multiple cost functions are of interest.

4.2 Derivation of the Adjoint Approach

The sensitivity equation Eq. (4.6)

$$\frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\beta}} \underbrace{- \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{U}} \overbrace{\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^{-1}}^{\mathrm{d}\boldsymbol{U}/\mathrm{d}\boldsymbol{\beta}}}_{\boldsymbol{\psi}^T} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}}, \tag{4.6}$$

can also be solved using the adjoint approach by calculating the adjoint vector $\boldsymbol{\psi}$ through the adjoint equations,

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^T \boldsymbol{\psi} = -\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{U}}^T. \tag{4.7}$$

The adjoint vector, $\boldsymbol{\psi}$, can be substituted into Eq. (4.6) to find the total sensitivity.

$$\frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\beta}} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}}, \tag{4.6}$$

In contrast to the direct method, the adjoint vector is independent of the number of design variables, $\boldsymbol{\beta}$. Instead, it is dependent on the number of cost functions of interest. The solution methodology has a considerable impact on the cost of the sensitivity analysis. If the number of design variables exceeds the number of cost functions, then the adjoint method should be favored. Otherwise, the direct approach should be considered.

From a mathematical viewpoint, the analytic adjoint equations are obtained by linearizing the flow equations [128]. For numerical applications, a discretized version of the adjoint equations is required. Derivation of the discretized adjoint equations can be divided into the continuous and discrete approaches [153]. In the continuous adjoint approach, discussed in Section 4.2.1 the discretized adjoint equations are obtained by linearizing the flow equations to obtain the analytic adjoint equations, which are then discretized. The discrete adjoint approach, discussed in Section 4.3.2, reverses this procedure. The flow equations are initially discretized and then linearized.

Sensitivity values found using the two approaches may vary, since discretization and linearization are generally noncommutative. The discrete approach delivers the exact gradients of the discrete objective function and is consistent with gradients found through finite difference approximations.

4.2.1 Continuous adjoint method[1]

In the continuous adjoint approach, the PDEs are differentiated prior to discretization. The adjoint equations are obtained by defining a cost function which is augmented with the flow equations enforced as constraints through Lagrange multipliers [4].

This approach was the method originally developed by Jameson in his seminal work [98]. Borrowing from the field of optimal control Jameson applied calculus of variations to the governing flow equations to obtain the adjoint equations [141]. Derivation of the adjoint equations through the continuous adjoint approach has the advantage that it conveys the physical significance of the adjoint variables and boundary conditions in a meaningful manner. In comparison to discrete approach, solvers developed using the continuous adjoint approach typically feature reduced memory requirements and faster run times [149]. However, difficulty of implementation, which will be illustrated in the following subsection, has led to many favoring the discrete adjoint approach over the continuous adjoint approach [141].

Flow equations: quasi-one-dimensional Euler equations

To demonstrate the complexity of the continuous adjoint approach, the continuous adjoint approach is derived for the quasi-1D Euler equations introduced in Section 2.2.1. First, the quasi-1D Euler flow equations are reintroduced. Then, the continuous adjoint equations and boundary conditions are derived. The quasi-1D Euler equations were utilized because they are simple enough to have exact analytic solutions, while retaining complex flow features, such as shocks [62]. Although the method is presented for a quasi-1D problem, it is extendable to two-dimension and three-dimension CFD-based adjoint flow solvers.

In the present derivation, the quasi-1D Euler equations were employed to model flow through a nozzle and can be written as

$$\boldsymbol{R}\left(\boldsymbol{U}, A\right) = \frac{\mathrm{d}}{\mathrm{d}x}\left(A\boldsymbol{F}\right) - \boldsymbol{Q}\frac{\mathrm{d}A}{\mathrm{d}x} = \boldsymbol{0}\,. \tag{2.21}$$

Here A denotes the cross-sectional area, and the vector of conservation variables $\boldsymbol{U}$, the convective flux vector $\boldsymbol{F}$, and the source term $\boldsymbol{Q}$ are given by

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix}\,, \quad \boldsymbol{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho h u \end{bmatrix}\,, \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix}\,.$$

[1]This section, in part, is a reprint of the material as it appears in the International Journal of Numerical Methods in Fluids 86 (9), 582-606 titled "*Convergence acceleration of continuous adjoint solvers using a reduced-order model*" (2018). Authors: **Andrew Kaminsky**, Reza Djeddi, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Wiley's copyright policies permit the use of the article in full or in part within the author's thesis or dissertation for non-commercial purposes.

Adjoint equations for the quasi-one-dimensional Euler equations

The derivation of the continuous adjoint equations presented here closely follows the approaches of Lozano and Ponsin [128] and Giles and Pierce [62]. The derivation begins by defining a cost function, J. For example, consider a cost function defined as the integral of pressure along the nozzle

$$J = \int_\Omega p \, \mathrm{d}x \,, \tag{4.8}$$

where Ω represents the flow domain. This cost function is analogous to the lift integral, which is of considerable importance in aerodynamic design. From this definition, it should be observed that the cost function depends on the flow solution. Therefore, a perturbation in the flow solution, $\delta \boldsymbol{U}$, causes a change in the cost function, which is expressed by

$$\delta J = \int_\Omega \frac{\partial p}{\partial \boldsymbol{U}} \delta \boldsymbol{U} dx \,. \tag{4.9}$$

To calculate the change in the cost function, Eq. (2.21) must be solved for each perturbation, $\delta \boldsymbol{U}$, which is computationally demanding.

The method of Lagrange multipliers can be used to alleviate this requirement through the augmented cost function, written as

$$J = \int_\Omega p \, \mathrm{d}x - \int_\Omega \boldsymbol{\psi}^T \boldsymbol{R}\left(\boldsymbol{U}, A\right) \, dx \,. \tag{4.10}$$

It should be noted that the augmented cost function Eq. (4.10) is equivalent to Eq. (4.8) because $\boldsymbol{R}\left(\boldsymbol{U}, A\right) = 0$ for a converged flow solution. Linearization of the new augmented cost function with respect to changes in the flow, results in

$$\delta J = \int_\Omega \frac{\partial p}{\partial \boldsymbol{U}} \delta \boldsymbol{U} dx - \int_\Omega \boldsymbol{\psi}^T \left[\frac{\mathrm{d}}{\mathrm{d}x} \left(A \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}} \delta \boldsymbol{U} \right) - \frac{\mathrm{d}h}{\mathrm{d}x} \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{U}} \delta \boldsymbol{U} - \frac{\mathrm{d}\delta A}{\mathrm{d}x} \boldsymbol{Q} + \frac{\mathrm{d}}{\mathrm{d}x} \left(\delta A \boldsymbol{F} \right) \right] dx \,. \tag{4.11}$$

Integration by parts followed by isolation of terms dependent on changes in the flow solution, $\delta \boldsymbol{U}$, yields

$$\begin{aligned} \delta J = & \int_\Omega \boldsymbol{\psi}^T \left[\frac{\mathrm{d}\delta A}{\mathrm{d}x} \boldsymbol{Q} - \frac{\mathrm{d}}{\mathrm{d}x} \left(\delta A \boldsymbol{F} \right) \right] \mathrm{d}x \\ & - \int_\Omega \delta \boldsymbol{U}^T \left[-A \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}^T \frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}x} - \frac{\mathrm{d}A}{\mathrm{d}x} \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{U}}^T \boldsymbol{\psi} - \frac{\partial p}{\partial \boldsymbol{U}}^T \right] \mathrm{d}x - \left[A \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}} \delta \boldsymbol{U} \right]_{x_i}^{x_o} . \end{aligned} \tag{4.12}$$

Here x_i and x_o are the inlet and outlet locations, respectively. This equation is valid for any value of $\boldsymbol{\psi}$. As a result, Eq. (4.12) can be simplified so that the variation of the cost function, δJ, no longer depends on the variation of the flow solution, $\delta \boldsymbol{U}$, by choosing $\boldsymbol{\psi}$ so that these terms are zero, i.e.,

$$-A\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}^T\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}x}-\frac{\mathrm{d}A}{\mathrm{d}x}\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{U}}^T\boldsymbol{\psi}-\frac{\partial p}{\partial \boldsymbol{U}}^T=\boldsymbol{0} \tag{4.13}$$

and

$$\left[A\boldsymbol{\psi}^T\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}\delta\boldsymbol{U}\right]_{x_i}^{x_o}=\boldsymbol{0}\,. \tag{4.14}$$

With these conditions satisfied, the variation of the cost function can be calculated, independently of $\delta\boldsymbol{U}$, through the remaining terms,

$$\delta J=\int_\Omega \boldsymbol{\psi}^T\left[\frac{\mathrm{d}\delta A}{\mathrm{d}x}\boldsymbol{Q}-\frac{\mathrm{d}}{\mathrm{d}x}\left(\delta A\boldsymbol{F}\right)\right]\mathrm{d}x\,. \tag{4.15}$$

Adjoint boundary conditions for the quasi-one-dimensional Euler equations

To solve the adjoint equations, Eq. (4.13), appropriate boundary conditions must be derived to satisfy Eq. (4.14). The mathematical formulation of appropriate boundary conditions for the adjoint equations is one of the main challenges associated with the continuous adjoint method. The inlet and outlet adjoint boundary conditions must be formulated so that Eq. (4.14) is satisfied. In doing so, it is helpful to keep in mind that the adjoint characteristics are inversely related to the flow characteristics. For the quasi-one-dimensional Euler equations, if the flow equations have n incoming characteristics, then the adjoint equations have $(3-n)$ incoming characteristics, and an equivalent number of boundary conditions must be specified [58].

Subsonic inlet boundary conditions

In the flow regime, a subsonic inlet has two incoming characteristics (corresponding to right-traveling pressure and entropy waves), with the eigenvalues $(u+c)$ and u, and one outgoing characteristic (corresponding to a left-traveling pressure wave), with the eigenvalue, $(u-c)$. As a result, the flow equations require two physical and one numerical boundary condition. One possibility is to specify the stagnation enthalpy, h, and the total pressure, p_t, while numerically extrapolating the Mach number, M, from the interior.

Recall that the adjoint boundary condition must satisfy Eq. (4.14). Expanding the adjoint boundary condition in terms of the inlet flow variables (h, p_t, M) yields

$$\boldsymbol{\psi}^T\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}\delta\boldsymbol{U}=\boldsymbol{\psi}^T\frac{\partial \boldsymbol{F}}{\partial h}\bigg|_{p_t,M}\delta h+\boldsymbol{\psi}^T\frac{\partial \boldsymbol{F}}{\partial p_t}\bigg|_{h,M}\delta p_t+\boldsymbol{\psi}^T\frac{\partial \boldsymbol{F}}{\partial M}\bigg|_{h,p_t}\delta M=\boldsymbol{0}\,. \tag{4.16}$$

Here the change in total enthalpy, δh, and total pressure, δp_t, are both zero if no energy is added or removed from the system. This leaves only the term dependent on the change in Mach number, δM, which can be eliminated by imposing the adjoint boundary condition:

$$\boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial M}\bigg|_{h,p_t} = \mathbf{0} \tag{4.17}$$

The inlet boundary condition in the continuous adjoint solver is implemented by imposing the adjoint boundary condition given by Eq. (4.17) and then extrapolating the two additional adjoint variables from the interior of the domain. This is accomplished by defining adjoint inlet variables, $\tilde{\boldsymbol{\psi}}_{in} = \begin{bmatrix}\tilde{\psi}_M & \tilde{\psi}_H & \tilde{\psi}_{p_t}\end{bmatrix}^T$, as follows

$$\begin{aligned}
\tilde{\psi}_M =& \frac{M\left(1+\frac{\gamma-1}{2}M^2\right)}{\rho u\left(1-M^2\right)}\left(\frac{\partial \boldsymbol{F}}{\partial M}\right)^T_{h,p_t}\boldsymbol{\psi} = \psi_1 + u\psi_2 + h\psi_3 \\
\tilde{\psi}_h =& \frac{2h}{\rho u}\left(\frac{\partial \boldsymbol{F}}{\partial h}\right)^T_{M,p_t}\boldsymbol{\psi} = -\psi_1 + h\psi_3 \\
\tilde{\psi}_{p_t} =& p_t\left(\frac{\partial \boldsymbol{F}}{\partial p_t}\right)^T_{h,M}\boldsymbol{\psi} = \rho u\psi_1 + \left(\rho u^2 + p\right)\psi_2 + \rho u h\psi_3\,,
\end{aligned} \tag{4.18}$$

where $\tilde{\psi}_M = 0$, while $\tilde{\psi}_h$ and $\tilde{\psi}_{p_t}$ are extrapolated from the interior. Inversely, the adjoint variables at the inlet, $\boldsymbol{\psi}_{in} = [\psi_1 \;\; \psi_2 \;\; \psi_3]^T_{in}$ are related to $\tilde{\boldsymbol{\psi}}_{in}$ through

$$\boldsymbol{\psi}_{in} = \begin{bmatrix}\psi_1\\ \psi_2\\ \psi_3\end{bmatrix}_{in} = \begin{bmatrix}\frac{\rho u^2+p}{2p} & -\frac{1}{2} & -\frac{u}{2p}\\ -\frac{\rho u}{p} & 0 & \frac{1}{p}\\ \frac{\rho u^2+p}{2ph} & \frac{1}{2h} & -\frac{u}{2ph}\end{bmatrix}\begin{bmatrix}0\\ \tilde{\psi}^{(e)}_h\\ \tilde{\psi}^{(e)}_{p_t}\end{bmatrix}, \tag{4.19}$$

where the superscript (e) indicates extrapolated values from the interior.

Subsonic outlet boundary conditions

A subsonic outlet in the flow domain has one incoming characteristic (left-traveling pressure), corresponding to the eigenvalue $(u-c)$, and two outgoing characteristics (right-traveling pressure and entropy), corresponding to the eigenvalues $(u+c)$ and u, respectively. As such, the flow equations require one physical and two numerical boundary conditions. One possibility is setting the exit pressure, p, and then extrapolating two other variables such as the density, ρ, and the velocity, u, numerically from the interior [128].

The outlet adjoint boundary conditions must be defined to enforce the exit condition of Eq. (4.14), which can be expanded in terms of the exit flow variables (ρ, u, p) yielding

$$\boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}\delta\boldsymbol{U} = \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \rho}\bigg|_{p,u}\delta\rho + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial u}\bigg|_{p,\rho}\delta u + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial p}\bigg|_{\rho,u}\delta p\,. \tag{4.20}$$

Note that δp is zero if the back pressure is held constant, which eliminates the dependence of the adjoint outlet boundary condition, Eq. (4.20), on the pressure. To remove the dependence

of the adjoint outlet boundary condition, Eq. (4.20), on $\delta\rho$ and δu the adjoint boundary conditions require

$$\begin{aligned} \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \rho}\bigg|_{p,u} &= \mathbf{0} \\ \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial u}\bigg|_{p,\rho} &= \mathbf{0}\,. \end{aligned} \tag{4.21}$$

The outlet boundary condition in the continuous adjoint solver is thus implemented by imposing the adjoint boundary condition given by Eq. (4.21) and then extrapolating the remaining independent adjoint variable from the interior of the domain. Thus, the adjoint outlet variables, $\tilde{\boldsymbol{\psi}}_{out} = \begin{bmatrix}\tilde{\psi}_\rho & \tilde{\psi}_u & \tilde{\psi}_p\end{bmatrix}^T$, are

$$\tilde{\boldsymbol{\psi}}_{out} = \begin{bmatrix} \tilde{\psi}_\rho \\ \tilde{\psi}_u \\ \tilde{\psi}_p \end{bmatrix} = \begin{bmatrix} \frac{1}{u}\left(\frac{\partial \boldsymbol{F}}{\partial \rho}\right)^T \boldsymbol{\psi} \\ \frac{1}{\rho}\left(\frac{\partial \boldsymbol{F}}{\partial u}\right)^T \boldsymbol{\psi} \\ \left(\frac{\partial \boldsymbol{F}}{\partial p}\right)^T \boldsymbol{\psi} \end{bmatrix} = \begin{bmatrix} 1 & u & \frac{u^2}{2} \\ 1 & 2u & h+u^2 \\ 0 & 1 & \frac{\gamma}{\gamma-1}u \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix}. \tag{4.22}$$

The adjoint exit boundary conditions impose that $\tilde{\psi}_\rho = 0$ and $\tilde{\psi}_u = 0$, while $\tilde{\psi}_p$ is extrapolated from the interior of the domain. The relation of these adjoint inlet variables can be inverted, which gives the final relation for the adjoint variables

$$\boldsymbol{\psi}_{out} = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix}_{out} = \frac{1}{\frac{\gamma+1}{\gamma-1}\frac{u^2}{2} - h} \begin{bmatrix} \frac{u^2(\gamma+1)-h(\gamma-1)}{\gamma-1} & -\frac{u^2}{2}\frac{\gamma+1}{\gamma-1} & uh \\ -\frac{\gamma u}{\gamma-1} & \frac{\gamma u}{\gamma-1} & -h-\frac{u^2}{2} \\ 1 & -1 & u \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \tilde{\psi}_p^{(e)} \end{bmatrix}. \tag{4.23}$$

Supersonic outlet boundary conditions

In the flow regime, a supersonic outlet has no incoming characteristics, and three outgoing characteristic, corresponding to the eigenvalues of u, $(u-c)$, and $(u+c)$. Therefore, the flow equations require three numerical boundary conditions, and the three exit flow variables, p, ρ, and u, are extrapolated numerically from the interior.

The outlet boundary conditions again must be defined to enforce the exit condition of Eq. (4.14), which can be expanded in terms of the exit flow variables (ρ, u, p) yielding

$$\boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}} \delta \boldsymbol{U} = \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \rho}\bigg|_{p,u} \delta\rho + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial u}\bigg|_{p,\rho} \delta u + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial p}\bigg|_{\rho,u} \delta p\,. \tag{4.24}$$

For this condition to be satisfied, the adjoint boundary conditions require

$$\begin{aligned}\boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial \rho}\bigg|_{p,u} &= \mathbf{0} \\ \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial u}\bigg|_{p,\rho} &= \mathbf{0}\,. \\ \boldsymbol{\psi}^T \frac{\partial \boldsymbol{F}}{\partial p}\bigg|_{u,\rho} &= \mathbf{0}\end{aligned} \tag{4.25}$$

As a result, the adjoint variables for a supersonic outlet will all equal zero. Examining the adjoint outlet boundary condition given by Eq. (4.22) and Eq. (4.23) reaffirms this according to

$$\boldsymbol{\psi}_{out} = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix}_{out} = \frac{1}{\frac{\gamma+1}{\gamma-1}\frac{u^2}{2} - h} \begin{bmatrix} \frac{u^2(\gamma+1)-h(\gamma-1)}{\gamma-1} & -\frac{u^2}{2}\frac{\gamma+1}{\gamma-1} & uh \\ -\frac{\gamma u}{\gamma-1} & \frac{\gamma u}{\gamma-1} & -h-\frac{u^2}{2} \\ 1 & -1 & u \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.26}$$

since $\tilde{\psi}_\rho$, $\tilde{\psi}_u$, and $\tilde{\psi}_p$ must all equal zero.

4.3 Automatic Differentiation

The derivation of the continuous adjoint approach for the quasi-one-dimensional Euler equations highlights the challenges associated with the continuous adjoint technique. Adding the viscous terms, only further complicates development [210]. The discrete adjoint approach represents an appealing alternative, because development can be simplified through the use of automatic differentiation.

As the name suggests, automatic differentiation is a set of techniques developed to evaluate the derivative of a function specified by a computer program in an automated manner. Automatic differentiation is based upon the premise that a CFD solver is composed of lines of code that can be broken down to a sequence of elementary arithmetic operations and functions. Thus, the flow solver can be rewritten as a sequence of functions f_i,

$$f = f_1 \circ f_2 \circ f_3 \circ ... \circ f_n. \tag{4.27}$$

Here f_1 is the first action executed by the code, f_2 is the second, and so on. Each operation has a simple derivative, and the chain rule can be used to find the derivative of the complete sequence. Automatic differentiation systematically applies the chain rule to each operation of the code so that the derivative of each step can be accumulated to give the directional derivative of interest.

$$\frac{\partial f}{\partial \beta} = \overset{\overset{\leftarrow adjoint\ accumulation}{}}{\underset{forward\ accumulation \rightarrow}{\frac{\partial \beta}{\partial \beta}\frac{\partial f_1}{\partial \beta}\frac{\partial f_2}{\partial f_1}\frac{\partial f_3}{\partial f_2}\cdots\frac{\partial f_n}{\partial f_{n-1}}}}. \tag{4.28}$$

The automatic differentiation tool calculates the derivative by tracing the flow of information in one of two directions and then assembling the derivative of each statement of the solver using the chain rule. In forward mode, the derivative is traced from the inputs to the output. In the reverse mode the process is reversed, and the derivatives are traced backwards from the output to the inputs.

4.3.1 Forward mode automatic differentiation [2]

A direct sensitivity solver can be developed using forward mode automatic differentiation. In forward mode automatic differentiation, the directional derivative is found by specifying the independent variable of interest, or the direction. The derivative of each intermediate term with respect to this independent variable is then determined by calculating the derivative of each sub-expression recursively. This process is repeated until the derivative of interest is reached.

Automatic differentiation can be implemented using one of two strategies: source code transformation (SCT) or operator overloading (OO). Operator overloading can only be utilized in languages where it is supported. Fortran is one such language. Following the method presented by Thomas et al. [187], Fig. 10 presents an example of operator overloading implemented in Fortran and its utilization for a simple function is shown in Fig. 11.

Operator overloading has the inherent advantage that it doesn't require changes to the form of the original source code, often the only changes required are changes to the data types. Additionally, forward mode operator overloading is easy to implement and alleviates the need for third-party source code transformation tools. However, operator overloading implementations apply a great deal of pressure on the compiler to remove all the extra dispatches, allocations, and memory references it introduces. Operator overloading is also frequently more costly than the source code transformation approach, because derivatives are calculated for every operation regardless of whether it is pertinent to the quantities of interest. However, Djeddi and Ekici recently proposed techniques to reduce the cost of OO-based automatic differentiation methods [35].

In the source code transformation approach, a transformation tool, like TAPENADE [76], is used to parse the source code and return output code that has statements for calculating the derivatives interwoven with the original lines of code as shown within Fig. 12. Within Fig. 12, the derivative variables are identified by the suffix (_d), and each derivative

[2] This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3744 titled "*Efficient Prediction of Forward Mode Aerodynamic Sensitivities using a Reduced-order Model*" (2018). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Andrew Kaminsky** and Kivanc Ekici.

calculation precedes the corresponding flow calculation so that the derivative is evaluated before the variable is updated.

For the present research, SCT has the advantage that the derivative code is human readable and can be edited separately from the base flow solver. Another feature is the SCT tool can be employed to parse and isolate the relationship between the cost function and the design variable so that derivatives are only computed for germane variables, which reduces the computational cost of the sensitivity code. The primary challenge of the SCT approach is the development of the source code transformation tool. However, several capable source code transformation tools available including FDOT [35], ADOL [70, 172], and TAPENADE [76]. The source code transformation tool TAPENADE [76] was used in this work.

To automatically differentiate the code, the source code is passed to the SCT tool and the input(s) and output(s) of interest are specified. The resulting differentiated code is structurally very similar to the original flow solver. This similarity is illustrated by considering the simplified flow solver presented in Fig. 13 and its forward mode differentiation code included in Fig. 14. Within the Figures, U represents the flow solution, R the residuals, y defines the grid, β the design variables, J the cost function, $\rightarrow$ represents an input and $\leftarrow$ is an output. The dot derivatives correspond to the derivative of the variable with respect to the specified design variable of interest i.e.,

$$\dot{y} = \frac{\partial y}{\partial \beta}, \quad \dot{U} = \frac{\partial U}{\partial \beta}, \quad \dot{R} = \frac{\partial R}{\partial \beta} \quad \dot{J} = \frac{\partial J}{\partial \beta}.$$

The forward sensitivity method calculates the sensitivity of every intermediate variable to the specified design variable, β. Thus, the forward sensitivity method requires a sensitivity solution for each design variable of interest. The intermediate functions can be grouped together so that the calculation of the forward sensitivity value of interest is calculated iteratively:

$$\dot{\boldsymbol{U}}_{i,j}^{n+1} = \dot{\boldsymbol{U}}_{i,j}^{n} + \dot{\boldsymbol{R}}_{i,j}\left(\boldsymbol{U}\right). \tag{4.29}$$

This can be seen by examining the automatic differentiated code shown in Fig. 14, where `update_d` updates both the flow solution U and the derivative $\dot{U}$.

4.3.2 Reverse mode automatic differentiation[3]

Automatic differentiation can also be used to dramatically simplify the development of a discrete adjoint solver through reverse mode automatic differentiation. This fact is the primary advantage of the discrete adjoint approach. Reverse mode automatic differentiation can generate code through an all-at-one 'brute force' approach (similar to the forward approach) or through a multiple-application 'assembled' approach. The multiple application

[3]This section, in part, is a reprint of the material as it appears in Aerospace Sciences 93 titled "*Reduced-order model-based convergence acceleration of reverse mode discrete adjoint solvers*" (2019). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Elsevier's copyright policies permit the material in the paper to be included, in full or in part, within the author's thesis or dissertation for non-commercial purposes.

assembled approach has a number of benefits over the brute force approach. Many of these advantages are critical to the ROM convergence acceleration. The following sections will introduce the two adjoint approaches and highlight the relative limitations and advantages of each approach.

Brute Force Automatic Differentiation

In the brute force approach, the entire CFD source code is passed to an automatic differentiation tool while specifying the dependent output and independent input variables. The sensitivity is then calculated by tracing the flow of information and assembling the derivative of each statement of the solver using the chain rule. In contrast to the forward approach which accumulates derivatives in a single forward pass (see Fig. 14) adjoint solvers built using reverse mode automatic differentiation calculate sensitivities in a two-step process as depicted in Fig. 15.

A forward sweep is performed first to compute the primal flow solution, and then a reverse mode sweep traces the derivatives from the cost function all the way back to the design variables. This reverse sweep traces the derivatives backwards from the objective function to the design parameters, or:

$$\frac{\partial f}{\partial \beta} = \overset{\leftarrow \textit{adjoint accumulation}}{\frac{\partial \beta}{\partial \beta}\frac{\partial f_1}{\partial \beta}\frac{\partial f_2}{\partial f_1}\frac{\partial f_3}{\partial f_2}\cdots\frac{\partial f_n}{\partial f_{n-1}}}. \tag{4.30}$$

The code generated from reverse mode automatic differentiation is an unsteady adjoint of the original solver [91]. This results in an inefficient and costly adjoint sensitivity code. The unsteady adjoint stores the solution history by saving the value of the intermediate variables at every iteration during the forward sweep. During the reverse sweep these stored values are reloaded in the corresponding reverse iterations. TAPENADE performs storage and loading through PUSH and POP statements, respectively.

To demonstrate this process we again consider the our simplified flow solver code in Fig. 16 and the resulting reverse mode automatic differentiated code in Fig. 17. Within the figures the bar derivatives correspond to the derivative of the cost function with respect to the named variable i.e.,

$$\bar{y} = \frac{\partial J}{\partial x}, \quad \bar{U} = \frac{\partial J}{\partial U}, \quad \bar{R} = \frac{\partial J}{\partial R} \quad \bar{J} = \frac{\partial J}{\partial J}.$$

The adjoint code contains a forward sweep that calculates the flow solution and stores the solution history using PUSH statements, then during the reverse sweep the sensitivities are calculated and the stored solution history is reloaded using the POP statements. It is incredibly expensive to store the solution history, and typically the storage cannot be handled using RAM alone, which further slows down the calculation. Fortunately the costly solution storage can be neglected for steady flow considerations. In steady state considerations, the time-stepping is performed in pseudo-time and only the final solution state is of interest. Once convergence is reached, the flow solution ceases to change from one

iteration to the next, which means the intermediate variables remain constant (to machine accuracy). Since the intermediate variables following convergence remain constant, storage of the solution history can be avoided, because the sensitivity solution can keep using the same flow solution values. This means the top level PUSH and POP statements, specifically those in the iterative loop that store the intermediate values for each iteration can be removed. Unfortunately, the removal of the PUSH and POP statements is only applicable to steady state considerations. To consider unsteady time-accurate flow solutions, checkpointing [199] and POD basis vectors [10] have been used to limit the storage costs. However, both add another level of complexity to the sensitivity analysis. The harmonic balance equations developed in Section 2.3, provide an interesting alternative for sensitivity analysis of unsteady flows. Since the harmonic balance equations consider unsteady periodic flow fields using a mathematically steady model, the solution storage can be neglected which makes the harmonic balance equations well-suited to sensitivity analysis of unsteady problems.

It is useful to break the forward and reverse passes into pre-iterative, iterative, and post-iterative sections following the approach presented by Christakopoulos et al. [27]. Fig. 18 presents an example of code built using brute force reverse mode AD applied to flow solver broken into these pre-iterative, iterative, and post-iterative sections. In the figure, the terms with an overbar represent the adjoint quantities. The derivative code propagates the sensitivity from the cost function, through the iterative loop, to the grid, and finally the defining design variables. This reversed nature can be seen by examining the reverse sweep subroutines and noticing that when the primal variable was an input, the derivative variable (for example $\bar{\boldsymbol{U}}$) is now an output and vice versa. Propagating the sensitivities through the reverse operations, the boxes associated with the post-iterative, iterative, and pre-iterative parts of the code calculate the following derivatives:

- `cost_function_b`: Initializes the flow solution adjoint $\bar{\boldsymbol{U}} = \frac{\partial J}{\partial \boldsymbol{U}}$.
- `The iterative loop`: Accumulates the sensitivity to the grid $\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{y}} \frac{\partial J}{\partial \boldsymbol{U}}$.
- `generate_grid_b`: Completes the sensitivity to the design variables $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\beta}} \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{y}} \frac{\partial J}{\partial \boldsymbol{U}}$.

The intermediate functions within the iterative loop can be grouped together so that the sensitivities can be calculated iteratively, i.e.,

$$\bar{\boldsymbol{U}}^{n+1} = \bar{\boldsymbol{U}}^{n} + \bar{\boldsymbol{R}}^{n}. \tag{4.31}$$

This can be seen by examining the subroutine `residual_b` in the automatic differentiated code shown in Fig. 18b. Each pass through the subroutine `residual_b` calculates

$$\bar{\boldsymbol{U}}^{n} = \frac{\partial J}{\partial \boldsymbol{U}^{n+1}} \frac{\partial \boldsymbol{U}^{n+1}}{\partial \boldsymbol{R}^{n}} \frac{\partial \boldsymbol{R}^{n}}{\partial \boldsymbol{U}^{n}} \tag{4.32}$$

as shown by the $\leftarrow$ `U` output. It also builds the sensitivity to the grid

$$\bar{\boldsymbol{y}} = \frac{\partial J}{\partial \boldsymbol{U}^{n+1}} \frac{\partial \boldsymbol{U}^{n+1}}{\partial \boldsymbol{R}^{n}} \frac{\partial \boldsymbol{R}^{n}}{\partial \boldsymbol{U}^{n}} \frac{\partial \boldsymbol{U}^{n}}{\partial \boldsymbol{y}} \tag{4.33}$$

as indicated by the $\leftarrow \bar{\mathrm{y}}$ output. This process is repeated in a reverse manner for $n = N_{max}, N_{max}-1, ..., 2, 1$. Within the loop, the sensitivity to the grid is accumulated at each backward iteration, but it is not used as an input. As a result, the working variable of the iterative loop is the sensitivity of the cost function to the flow variable during the iterative loop, and at each iteration the sensitivity to the grid is accumulated for use following the iterative loop.

It is very important to note that brute force automatic differentiation of a fixed-point solver does not yield a fixed-point iterative adjoint solver [1]. That is, $\bar{\boldsymbol{R}}^n$ is not a function of $\bar{\boldsymbol{U}}^n$. Instead, the code is merely a series of derivatives accumulated via the chain rule. A change to the intermediate sensitivity variables would befoul the derivative aggregation and lead to incorrect sensitivity values. Accordingly, code developed using brute force automatic differentiation must initialize the iterative loop to the value of the gradient of the cost function. Therefore, "hot-starting" from another value, for example the previous solution of the last design cycle, typically results in the accumulation of an incorrect derivative [27]. This acts as a barrier to several acceleration techniques, including the reduced-order model acceleration technique presented in Chapter 7 that is the focus of this dissertation [109]. As a result, an alternative automatic differentiation approach is needed, if computational cost reduction is sought.

Primal time-stepping

To maximize the efficacy of the proposed reduced-order model adjoint sensitivity acceleration technique, a fixed-point iterative method is needed for the discrete adjoint approach. The "primal time-stepping" adjoint approach proposed by Christakopoulos [27] is a straightforward method for employing reverse mode automatic differentiation to develop a fixed-point adjoint solver. The primal time-stepping approach hand assembles automatic differentiated code to develop a fixed-point iterative method for the adjoint sensitivities through a process that reuses the time-stepping of the flow solution within the adjoint solver. Thus, the adjoint variables are updated in a fixed-point manner according to

$$\bar{\boldsymbol{U}}^{n+1} = \bar{\boldsymbol{U}}^n + \bar{\boldsymbol{R}}\left(\bar{\boldsymbol{U}}^n\right), \tag{4.34}$$

where $\bar{\boldsymbol{U}}$ is the adjoint solution, $\bar{\boldsymbol{R}} = -\Delta t\left(\boldsymbol{A}^T\bar{\boldsymbol{U}} - \boldsymbol{g}\right)$ is the adjoint residual where Δt is the primal time-step.

Following [27] The primal time-stepping approach was motivated by noticing that in the brute force adjoint approach the sensitivity of the flow solver to the grid, $\bar{\boldsymbol{y}}$, is calculated every iteration. However, these grid sensitivity values are only used following the loop, so they actually only need to be calculated once, after the adjoint solution is converged. The primal time-stepping adjoint technique removes this calculation from the iterative loop by differentiating the residual subroutine twice, once with respect to the flow solution $\boldsymbol{U}$ and once with respect to the grid parameters $\boldsymbol{y}$ [27]. The two applications of AD result in subroutines labeled `residual_u` and `residual_y` to demarcate which variable the subroutine

was differentiated with respect to. A fixed-point adjoint solver can then be hand assembled in the form of the primal flow solution, as shown in Fig. 19.

The iterative code block now follows the structure found in the flow solver, i.e., a residual calculation followed by a solution update. In fact, the adjoint PTS loop actually uses the `update` subroutine from the original flow solver to iteratively update the sensitivity values. To keep the logic of the adjoint variables and residuals intact, the positions of $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{R}}$ must be switched manually (compared to the brute force approach) in the invocation of the subroutine `residual_u` [27]. Following the operations of the PTS adjoint approach presented in Fig. 19, the derivatives are calculated in the following manner:

- `cost_function_b`: Calculates the source term $\boldsymbol{g} = \frac{\partial J}{\partial \boldsymbol{U}}$.
- `The iterative loop`: Computes the adjoint variables $\boldsymbol{\psi} = \boldsymbol{A}^{-T}\boldsymbol{g} = \frac{\partial J}{\partial \boldsymbol{R}}$.
- `residual_y`: Calculates the sensitivity to the grid $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{y}} \frac{\partial J}{\partial \boldsymbol{R}}$.
- `read_grid_b`: Completes the sensitivity calculation $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\beta}} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{y}} \frac{\partial J}{\partial \boldsymbol{R}}$.

As before, the subroutine `cost_function_b` calculates the source term. Manually switching the invocation of the subroutine `residual_u` allows us to compute product of $\boldsymbol{A}^T$ and the input seed vector $\bar{\boldsymbol{U}}$, which gives the current iterate of the adjoint variables, $\boldsymbol{\psi}$, from the adjoint equation

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^T \boldsymbol{\psi} = -\frac{\partial J}{\partial \boldsymbol{U}}^T, \tag{4.35}$$

to calculate $\boldsymbol{A}^T\bar{\boldsymbol{U}}$. Subtracting the source term from the output of `residual_u` provides the adjoint residual used by `update`. After the iterative loop has reached a converged adjoint solution, the subroutine `residual_y` calculates the sensitivity to the grid only once for the converged adjoint solution, which is then extended to the design variable sensitivities in `read_grid_b`.

The PTS adjoint generalizes the adjoint initialization parameter due to the inclusion of the source term so that it is the current iterate of the adjoint vector. As a result the PTS adjoint code is a fixed-point method that can recover from a solution perturbation or a bad initial guess. Hence, the solution can be "hot-started," and is no longer required to start from the original source vector. These features allow the novel reduced-order model acceleration technique to supply an updated adjoint vector and resume traditional iteration towards convergence. This technique is discussed in Chapter 7 and demonstrated in Chapter 8

Chapter 5

Validation and Verification

This chapter presents case studies performed to validate and verify the numerical results of the different flow and sensitivity solvers. Initially, the quasi-one-dimensional Euler solver is applied to three nozzle case studies from the literature. The numerical results of the flow and continuous adjoint solvers are compared with external sources. Next, *external* solver is applied to resolve two-dimensional flow over the RAE2822 airfoil. Three AGARD case studies are examined to verify the steady-state flow solution and the adjoint sensitivities of a brute-force discrete adjoint solver. The next case study considers two-dimensional flow over an oscillating NACA 0012 to verify the harmonic balance technique and discrete adjoint differentiation of the method. In addition, to validation, salient features of the solvers will be examined, particularly the performance of the traditional convergence acceleration techniques. Although not included here, a grid convergence was performed for all cases. In combination, these case studies instill sufficient confidence in the numerical solvers employed in this work. The flow solvers and sensitivity solvers validated here will be accelerated and utilized within optimization procedures in Chapter 8 .

5.1 Quasi-1D Flow through a Nozzle [1]

This case study considers inviscid flow through a converging diverging nozzle. This case will be used to verify the solver developed for the quasi-1D Euler equations introduced in Section 2.2.1 and the solver developed for the continuous adjoint equations detailed in Section 4.2.1. Unlike the discrete adjoint approach, the sensitivities calculated using the continuous adjoint formulation will not exactly match the finite difference or discrete sensitivity values, since linearization and discretization are noncommutative. As a result,

[1] This section, in part, is a reprint of the material as it appears in the International Journal of Numerical Methods in Fluids 86 (9), 582-606 titled "*Convergence acceleration of continuous adjoint solvers using a reduced-order model*" (2018). Authors: **Andrew Kaminsky**, Reza Djeddi, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Wiley's copyright policies permit the use of the article in full or in part within the author's thesis or dissertation for non-commercial purposes.

validation of the quasi-1D Euler solver and the continuous adjoint formulation was performed through comparison with external results [128, 62, 127].

Flow was considered for a nozzle given by

$$h(x) = \begin{cases} 2 & 0 \leq x \leq 0.5 \\ 1 + \sin^2(\pi(x - 1.0)) & 0.5 < x < 1.5 \\ 2 & 1.5 \leq x \leq 2\,, \end{cases} \tag{5.1}$$

and included in Fig. 20. Three cases with increasingly complex flow fields were considered. These cases were chosen because each had been previously examined, thus enabling both internal and external verification of the adjoint vector [128, 62, 127].

5.1.1 Nozzle with subsonic flow field

The first case considered a nozzle with a fully subsonic flow field. The case was defined by a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.9899. This corresponds to the cases of Lozano [127] and Giles and Pierce [62] defined by an inlet Mach number of $M_{in} = 0.12$.

Flow solution

The sensitivity calculation procedure began with determination of the nozzle flow field solution. The numeric flow solution was obtained using a dual control-volume discretization and a central scheme with scalar artificial dissipation on a 299-node uniform grid. The Mach number distribution is shown to agree with the analytic flow solution [128] in Fig. 21b. The resulting flow field is entirely subsonic and in the incompressible regime. As a result, the density-based compressible flow solver converges rather slowly as shown in Fig.21a.

Adjoint solution

Once the flow solution was obtained, the results were used to solve for the adjoint vector using a continuous adjoint solver. A traditional (non-accelerated) continuous adjoint solver was used to solve for the adjoint variables. The convergence rate of the adjoint solution was similar to that found for the flow solution and is included in Fig. 22a. The adjoint vector is verified in Fig. 22b using the results presented by Lozano and Ponsin [128]. The adjoint solution obtained using the continuous adjoint solver agrees well with Lozano and Ponsin [128].

5.1.2 Nozzle with subsonic inlet supersonic outlet

The second case considered a nozzle with a subsonic inlet flow and supersonic outlet. The case was specified by a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.52. This case corresponds to the case of Giles and Pierce [62] designated

by $H_{in} = 4$ and $p_{t_{in}} = 2$. The Mach number profile and adjoint solution using the traditional approach are compared with those of Giles and Pierce [62] in Figs. 23a and 23b, respectively. In this case a finer 899-node grid was considered. Inspecting the adjoint vector solution, it is interesting to note the asymptotic behavior at the nozzle throat. This is expected since a small change at the throat within a transonic flow field can have large ramifications. It can also be seen that the adjoint vector is zero at the outlet boundary, which is forced by the boundary conditions.

5.1.3 Nozzle with shocked flow

The third case considered a nozzle with a transonic shocked flow field with a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.84317. This case is consistent with the $H_{in} = 4$, $p_{t_{in}} = 2$, and $p_{ex} = 1.6$ case presented by Giles and Pierce [62]. Comparisons of the Mach number profile, adjoint solution, and those reported by Giles and Pierce [62] are presented in Figs. 24a and 24b.

The adjoint and flow solutions were obtained using a 1025-node grid. The Mach number profile, shown in Fig. 24a, clearly shows a shock in the flow field. Despite the shock in the flow field, the corresponding adjoint vector, Fig. 24b, is actually quite similar to the prior subsonic-supersonic case. The adjoint vector field again behaves asymptotically near the throat but remains smooth and continuous at the shock location. This behavior is expected, and an in-depth explanation is presented by Giles and Pierce [60]. Additionally, the adjoint vector is no longer zero at the exit.

These three cases verify the implementation of the quasi-1D Euler solver and the accompanying continuous adjoint solver. The continuous adjoint solver was one of the first solvers used to demonstrate the reduced-order model-based convergence acceleration technique, presented in Section 7, and these cases will be revisited in Section 8.2 to demonstrate the reduced-order model-based convergence acceleration technique.

5.2 Viscous 2D flow over an RAE2822 Airfoil

This case study considers two-dimensional viscous flow over an RAE 2822 airfoil. This case will be used to verify *external* flow solver detailed in Section 3.2 and a discrete adjoint solver developed through reverse mode automatic differentiation of the *external* flow solver following the brute force approach discussed in Section 4.3.2. Three flight conditions defined by the Advisory Group for Aerospace Research and Development (AGARD) test cases 1, 6, and 9 were considered to validate the *external* flow solver. These cases examine steady viscous flow over an RAE2822 airfoil. The grid employed for the RAE2822 is shown in Fig. 25.

The experimental case definition parameters are included in Table 3. Angle of attack corrections are extremely important for this airfoil [138] so a corrected angle of attack is included for each case. Traditionally, the angle of attack corrections are found by changing the angle of attack until the lift coefficient matches the experimental value, but since the

main purpose here is validation the corrected angle of attacks were set to values presented by other authors [138, 181, 196].

5.2.1 AGARD Test Case 1

The AGARD test case 1 for the RAE 2822 airfoil was considered using a corrected angle of attack $\alpha = 1.83°$ [138], Mach number $M_\infty = 0.676$, and Reynolds number $\text{Re} = 5.7 \times 10^6$. Table 4 presents a comparison of the lift, drag, and moment coefficients with the experimental results [30]. The lift coefficient values match to within 1%, but the CFD solution over predicts the drag. This case was further examined by comparing the surface pressure coefficient with experimental values, as shown in Fig. 26a. The general profile matches well, and the largest discrepancy occurs at the leading edge.

This case was also used to examine traditional convergence acceleration techniques. The convergence histories of the external solver accelerated using residual smoothing, 1- and 2-level multigrid, and simultaneous application of multigrid and residual smoothing are presented in Fig. 26b. Both multigrid and residual smoothing significantly reduce the number of iterations and time required to reach convergence. A comparison of the cost for each case is included in Table 5. The combined multigrid and residual smoothing case decreases the computational time by 52.6% and the iteration count by 82.3%. The computational time is not reduced as dramatically as the iteration count, because the acceleration techniques increase the computational cost associated with each iteration.

5.2.2 AGARD Test Case 6

AGARD test case 6 considers flow over the RAE 2822 airfoil, at a corrected angle of attack $\alpha = 2.44°$ suggested by Veldman [196], Mach number $M_\infty = 0.725$, and Reynolds number $\text{Re} = 6.5 \times 10^6$.

Flow Solution Verification

The lift coefficient calculated by the *external* solver again matches to within 1%, but the drag and moment coefficients are under- and over-predicted respectively as shown in Table 6. This case has been considered by Giles et al. [57] for varying angles of attack. The lift coefficients calculated using the *external* solver are compared with those provided by Giles at varying angles of attack, in Fig. 27. The two are also in good agreement, though our solver predicts a slightly steeper slope.

Discrete Adjoint Sensitivity Verification

The literature often validates sensitivity derivative solvers by examining the sensitivity to design variables that change the grid or design shape in some manner. The sensitivity values are then verified by comparing the results of the adjoint or forward mode with either finite differencing or the complex-step method. This approach is sufficient for solver verification,

but comparison with other solvers is difficult because grid sensitivities or shape sensitivity can change based on discretization type and grid.

To validate the sensitivity solver, the sensitivities to flow parameters were considered. In contrast to more grid dependent design parameters the sensitivity to flow parameters should remain relatively constant from one solver to another. Thus, sensitivities of the loading coefficients to varying angles of attack were considered for the flow conditions of the RAE 2822 AGARD case 6 outlined in Section 5.2.2. Sensitivity analysis of this case has been previously considered by Giles et al. [57]. The sensitivities of the loading parameters to the angle of attack were first calculated using finite differences. Then sensitivities for the direct and adjoint approaches were found through forward and reverse mode automatic differentiation of the *external* solver. A comparison of the converged values is included in Table 7. For the external solver all three sensitivity methods match well. A comparison with those reported by Giles et al.[57] shows that the sensitivity values calculated with *external* are slightly higher. This is expected based on the two profiles of the lift coefficient values for different angles of attack shown in Fig. 28, where the present solver's distribution has a slightly steeper slope.

Verification of the direct and adjoint approach can be further proven using the adjoint-tangent linearization equivalence:

$$g^T u = (\boldsymbol{A}^T \boldsymbol{v})^T \boldsymbol{u} = \boldsymbol{v}^T \boldsymbol{A}\boldsymbol{u} = \boldsymbol{v}^T \boldsymbol{f} \tag{5.2}$$

This implies that the direct and adjoint gradients should match to machine precision. The two automatically generated solvers compute the same derivative, but in reverse directions. Table 8 presents a comparison of the direct and adjoint sensitivities at different iterations within the convergence process for an angle of attack of 0.0°. As expected the sensitivity solvers match to machine precision at the final iteration. What is even more interesting though is that the sensitivity values match to machine accuracy at every iteration. This provides a method to validate our solvers without full sensitivity solutions.

AGARD Test Case 9

The final AGARD case, test case 9, considered flow over the RAE 2822 airfoil, at a corrected angle of attack $\alpha = 2.79°$[181], Mach number $M_\infty = 0.734$, and Reynolds number $\text{Re} = 6.5 \times 10^6$. The lift, drag, and moment coefficients match the experimental results most closely in this case, as shown in Table 9. The lift and drag coefficients both match to within 1% and 2% respectively. This case was further validated by comparing the surface pressure coefficient with experimental data and the numerical results of Swanson and Turkel [181], shown in Fig. 29a. The Mach contours are also included in Fig. 29b.

In combination, the three AGARD case studies validated and verified the *external* flow solver and the discrete adjoint implementation. The flow solver and the discrete adjoint technique will be reconsidered in Section 8.1 and Section 8.4 to demonstrate acceleration that can be achieved by applying the reduced-order model acceleration to the adjoint sensitivity calculation within a nested design process.

5.3 Inviscid 2D Flow over an Oscillating NACA0012 Airfoil [2]

This case study considers periodically unsteady inviscid transonic flow over an oscillating NACA0012 airfoil to validate the implementation of the harmonic balance method within the *external* solver. The case study considered is the AGARD CT5 case [116] which exhibits an unsteady transonic dynamically nonlinear flow condition. The nominal flow conditions considered were $M_\infty = 0.755$, $\alpha_0 = 0.016°$, $\alpha_1 = 2.51°$, and $k = 0.0814$. The inviscid mesh used for this case was generated using a conformal transformation [195], and it is included in Fig. 36. The sub-time level solutions for the three harmonic case are included in Fig. 31. As expected, the resulting flow field contains strong non-linearities, which can be attributed to shock wave motion occurring during the oscillation period.

This case was previously studied by Da Ronch et al. [33]. Fig. 32 presents a comparison of the harmonic balance aerodynamic loading coefficients of a case considering seven harmonics with Da Ronch's time accurate solution. Further assessments of the frequency domain method can be made by considering the zeroth and first harmonic unsteady surface pressure coefficient distributions as shown in Fig. 33. It can be seen that the results of the solver presented here agree well with those of Da Ronch.

5.4 Inviscid 2D Flow over a Plunging NACA0012 Airfoil [3]

This case study considers two-dimensional flow over a plunging NACA0012 airfoil in which the dynamic stability derivatives were found using the harmonic balance and a discrete adjoint solver. This case will verify the ability to perform sensitivity analysis of an unsteady flow.

5.4.1 Time-spectral Stability Derivative Method

First some background, Mader and Martin [133] developed the time-spectral stability method based on linear air reaction theory. Their method leverages a time-spectral CFD solution with a linear regression technique to generate estimates for the dynamic stability derivative of the force, lift, or moment coefficient with respect to an oscillating parameter and the

[2]This section, in part, is a reprint of the material as it appears in AIAA Paper 2016-0808 titled *"Sensitivity and Stability Derivative Analysis using an Efficient Adjoint Harmonic Balance Technique"* (2016). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Andrew Kaminsky** and Kivanc Ekici.

[3]This section, in part, is a reprint of the material as it appears in AIAA Paper 2016-0808 titled *"Sensitivity and Stability Derivative Analysis using an Efficient Adjoint Harmonic Balance Technique"* (2016). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Andrew Kaminsky** and Kivanc Ekici.

derivative of this oscillating parameter. The simple algebraic nature allows it to be used in conjunction with the adjoint method to compute the gradients for aerodynamic optimization.

Linear air reaction theory states that for a simple motion consisting of a single dynamic state, α, the lift coefficient can be approximated as

$$C_L = C_{L_0} + C_{L_\alpha}\Delta\alpha + C_{L_{\dot\alpha}}\Delta\dot\alpha + C_{L_{\ddot\alpha}}\Delta\ddot\alpha + \dots \,. \tag{5.3}$$

This approximation can be truncated by assuming that the higher-order derivatives are small and generalized so that

$$C_i = C_{i_0} + C_{i_j}\Delta j + C_{i_{\dot j}}\Delta\dot j \,, \tag{5.4}$$

where $i = L, D, M_x, M_z, M_y$ and $j = \alpha, \beta, q, r$. Equation (5.4) leaves us with three unknowns, $C_{i_0}, C_{i_j}, C_{i_{\dot j}}$. It is key to define a motion that excites a single dynamic state to identify the values of individual stability derivatives. Etkin[51] introduced a two-dimensional path to isolate α by plunging an airfoil so that the grid velocity adds a transient y component to the flow velocity changing the angle of attack according to

$$\alpha = A\sin\omega t \tag{5.5}$$

A time-spectral solution of a case considering a single dynamic state consists of state variables at $2N+1$ time instances distributed across a single period. This periodic time history of the force and moment coefficients can be used to compute the dynamic stability derivatives with respect to the oscillating parameter by relating the coefficients to the oscillating parameter through the time variable. Fig. 34 shows the lift coefficient results of a typical harmonic balance solution. It can be seen that a linear trend exists between the coefficient and motion parameter with a notable hysteresis effect resulting from the difference in the coefficient value during up and down stroke.

The time-spectral stability derivative method computes the stability derivative from the harmonic balance solution by performing a least squares fit of the coefficient with respect to the motion variable, which gives an approximation of the form

$$y = C_1 x + C_2 \,. \tag{5.6}$$

The resulting slope is the dynamic stability derivative, C_{L_α}, and the lift coefficient at the zero value of motion, C_{L_0}, is the y intercept. This can be generalized to

$$y = C_{i_j} j + C_{i_0} \,. \tag{5.7}$$

The solution hysteresis can be used to calculate the transient or "dot" derivatives can be calculated by subtracting the linear regression from the harmonic balance solution:

$$R^n_{C_i} = C^n_i + y(x^n) \,. \tag{5.8}$$

This removes the dependence of the solution on the main motion variable, leaving just $R^n_{C_i}$, which is the variation dependent on the latent hysteresis. A strong linear relationship between exists between $R^n_{C_i}$ and the time derivative of the motion variable, $\dot{\alpha}$, as shown in Fig. 35. The slope of this line is equivalent to $C_{L_{\dot{\alpha}}}$, the transient derivative term. Furthermore, the plot shows the linear approximation is a near exact match of the data indicating that dependence on higher-order time derivatives is negligible. Thus, the assumption to truncate Eq. (5.4) is warranted.

The time-spectral derivative method calculates stability derivatives through a simple algebraic technique and offers a rare opportunity to verify adjoint sensitivity calculations for an unsteady flow solver, since relatively little sensitivity analysis is available for unsteady flow fields.

5.4.2 Unsteady Adjoint Sensitivity Verification for Dynamic Stability Derivatives

With the time-spectral derivative method outlined, consider unsteady flow over a NACA0012 airfoil defined by a Mach number of $M_\infty = 0.1$ and an angle of attack of $\alpha_0 = 0.0$, to perform unsteady sensitivity analysis. Small amplitude plunging oscillations were considered so that the angle attack oscillation amplitude was $\alpha_1 = 0.5$. This is a highly linear case and only a single harmonic $N_H = 1$ is needed to resolve the flow solution. The NACA0012 airfoil is symmetric and has a 12 percent thickness with respect to its chord. The inviscid NACA0012 grid was generated using a conformal transformation[195], and it can be seen in Fig. 36. The flow field was simulated using the Euler equations.

NACA 0012: Spectral Stability Derivatives

The stability derivatives were calculated using Mader's time-spectral stability derivative method for different oscillation frequencies. A comparison of the NACA 0012 stability derivative profile with those of a flat plate and theoretical Theodorsen values is included in Fig. 37. Although the NACA 0012 is not thin, the aerodynamic response of the NACA 0012 demonstrates a similar trend to that of the flat plate due to its symmetry. The NACA 0012 dynamic stability derivatives, C_{L_α} and C_{m_α}, tend to overshoot those of the thin-airfoil theory at low frequencies while slightly undershooting them at higher frequencies. The transient stability derivatives, $C_{L_{\dot{\alpha}}}$ and $C_{m_{\dot{\alpha}}}$, both have profiles similar to those of the thin-airfoil theory with a small offset. This case has previously been considered by Mader and Martins[133], and the results presented here are in good agreement with their findings.

NACA 0012: Adjoint Stability Derivatives

These stability derivative calculations present two opportunities to evaluate the adjoint sensitivity method. The static stability derivatives, C_{L_α} and C_{m_α} at $k = 0.0$, can be calculated from a steady case by applying the adjoint method to the harmonic balance

solver considering zero harmonics, and taking the cost function as the lift coefficient

$$J = C_L \,, \tag{5.9}$$

and the design variable as the angle of attack

$$\beta = \alpha \,. \tag{5.10}$$

As expected, the time-spectral stability derivative values at low reduced frequencies converge to the value found through the adjoint technique, as shown in Fig. 38. The lowest reduced frequency considered was $k = 0.001$ and the subsequent C_{L_α} was within 0.05 percent of the value calculated through the adjoint sensitivity approach. One drawback of this steady method is the lack of time-dependent information in the solution, which makes the transient derivatives $C_{L_{\dot{\alpha}}}$ and $C_{m_{\dot{\alpha}}}$ unrecoverable. Thus, only the static stability derivatives can be found in this manner.

The dynamic stability derivative profiles, also offer an opportunity to perform sensitivity verification of unsteady flow solutions. An adjoint solver was developed to calculate the sensitivity of the lift dynamic stability derivative, C_{L_α}, with respect to the reduced frequency, k, or $\mathrm{d}C_{L_\alpha}/\mathrm{d}k$. Fig. 38 shows the stability derivative profile generated through the time-spectral stability derivative method overlaid with tangent lines of the derivative calculated through the adjoint sensitivity approach. It can be seen that the adjoint sensitivities, shown as the black tangent lines, clearly agree with those suggested by the time-spectral stability derivative profile.

In addition to this qualitative verification, the finite difference method can be used to quantitatively assess the derivative values using a perturbation size of 1×10^{-7}. Overall, the adjoint sensitivities match well with the values obtained using the finite difference approach, and a comparison can be seen in Fig. 39. The adjoint sensitivities and finite difference values all fall within a few percent of each other.

Chapter 6

Optimization methods

Aerodynamic design optimization problems have historically been formulated as a constrained minimization problem:

$$\min_{\boldsymbol{\beta}\in\mathcal{D}} J\left(\boldsymbol{\beta}\right) \tag{6.1}$$

where J is the objective function and $\boldsymbol{\beta}$ is the vector of design variables which belong to design space

$$\mathcal{D} := \left\{\boldsymbol{\beta} \in \mathbb{R}^n | h(\boldsymbol{\beta}) = \mathbf{0}\right\}. \tag{6.2}$$

The function h: $\mathbb{R}^n \rightarrow \mathbb{R}^r$ represents the equality constraints on the design space. In the case of aerodynamic optimization, the design variables are limited to the set of designs that provide a converged flow solution. The goal of the aerodynamic design problem is to find the design that minimizes the objective function.

6.1 Traditional Gradient-Based Optimization

Gradient-based optimization has emerged as the dominant optimization strategy for aerodynamic design. Gradient-based optimization methods iteratively update the design of design cycle n through a step in a search direction:

$$\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} + \delta \boldsymbol{d}^{(n)}, \;\; n = 1, 2, 3, ... \tag{6.3}$$

Here $\boldsymbol{d}$ defines the search direction, which is generally informed by the sensitivity of the objective metric to the design variables and the step size δ defines the magnitude of the perturbation.

6.1.1 Steepest Descent Method

The method of steepest descent defines the search direction as the negated gradient of the objective function:

$$\boldsymbol{d}^{(n)} = -\nabla f(\boldsymbol{\beta}^{(n)}) \tag{6.4}$$

This search direction intuitively falls out from a first-order Taylor expansion:

$$f(\boldsymbol{\beta} + \boldsymbol{t}) \approx f(\boldsymbol{\beta}) + \nabla f(\boldsymbol{\beta}^{(n)})^T \boldsymbol{t} + \dots . \tag{6.5}$$

The steepest descent design update is

$$\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} - \delta \nabla f(\boldsymbol{\beta}^{(n)}). \tag{6.6}$$

The steepest descent method is popular due its ease of implementation, but its convergence rate to the optimized design is slow in comparison to other more involved techniques.

6.1.2 Newton's Method

Newton's method can be derived in a similar manner using a second-order Taylor expansion in place of the first-order expansion to provide a better approximation of the search direction, e.g.,

$$f(\boldsymbol{\beta}^{(n)} + \boldsymbol{t}) \approx f(\boldsymbol{\beta}^{(n)}) + \nabla f(\boldsymbol{\beta}^{(n)})^T \boldsymbol{t} + \frac{1}{2}\nabla^2 f(\boldsymbol{\beta}^{(n)})^T \boldsymbol{t}^2 \dots . \tag{6.7}$$

By defining the next iterate as $\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} + t^{(n)}$ and the steepest descent step is found by minimizing the quadratic approximation in $\boldsymbol{t}$ based on the assumption that approximation is a convex function and its minimum can be found by setting the derivative to zero

$$0 = \frac{d}{d\boldsymbol{t}} \left(f(\boldsymbol{x}^{(n)}) + \nabla f(\boldsymbol{x}^{(n)})^T \boldsymbol{t} + \frac{1}{2}\nabla^2 f(\boldsymbol{x}^{(n)})^T \boldsymbol{t}^2 \right) = \nabla f(\boldsymbol{x}^{(n)}) + \nabla^2 f(\boldsymbol{x}^{(n)})^T \boldsymbol{t}. \tag{6.8}$$

The minimum of which can be found to be

$$\boldsymbol{t} = \frac{\nabla f(\boldsymbol{x}^{(n)})}{\nabla^2 f(\boldsymbol{x}^{(n)})} . \tag{6.9}$$

Using this result, the Newton iteration step direction is taken to be

$$\boldsymbol{d} = -\nabla^2 f(x^{(n)})^{-1} \nabla f(x^{(n)}) . \tag{6.10}$$

Generally this step direction is augmented with a small step size $0 \leq \delta \leq 1$ to meet the Wolfe or Armijo conditions leading to the Newton update performed as

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \delta \left[\nabla^2 f(x^{(n)}) \right]^{-1} \nabla f(x^{(n)}) . \tag{6.11}$$

Newton's method features faster convergence but is accompanied by the requirement of calculating the second derivative, or Hessian, of the function.

6.1.3 Quasi-Newton Broyden, Fletcher, Goldfarb, and Shanno

Calculating the Hessian of iterative CFD solvers can be challenging and computationally expensive. Therefore, it is desirable to avoid its calculation. Quasi-Newton methods remove the requirement of calculating the true Hessian $\nabla^2 f(x^{(n)})$ by forming an approximation of the Hessian $H^{(n)}$ at each design step. The Broyden, Fletcher, Goldfarb, and Shanno (BFGS) algorithm and its variants are perhaps the most popular of the quasi-Newton methods. The BFGS update replaces the true Hessian with the approximation and is expressed as

$$\boldsymbol{d} = \left[H^{(n)})\right]^{-1} \nabla f(x^{(n)})\,. \tag{6.12}$$

The BFGS method approximates the Hessian as

$$H^{(n+1)} = H^{(n)} + \frac{y^{(n)} y^{(n)^T}}{y^{(n)^T} s^{(n)}} - \frac{H^{(n)} s^{(n)} s^{(n)^T} H^{(n)}}{s^{(n)^T} H^{(n)} s^{(n)}} \tag{6.13}$$

where

$$s^{(n)} = x^{(n+1)} - x^{(n)}, \text{ and } y^{(n)} = \nabla f(x^{(n+1)}) - \nabla f(x^{(n)}). \tag{6.14}$$

The BFGS update generates a positive definite approximation of $H^{(n+1)}$ if the initial approximation $H^{(n)}$ is positive definite. It also mimics the properties of the true Hessian by satisfying the secant equation and maintaining symmetry [156]. Note the inverse is Hessian is used as the search direction, thus it is appealing to directly update the inverse

$$\begin{aligned}[H^{(n+1)}]^{-1} = [H^{(n)}]^{-1} &+ \frac{y^{(n)} y^{(n)^T}}{y^{(n)^T} s^{(n)}} - \frac{s^{(n)} y^{(n)^T} [H^{(n)}]^{-1} + [H^{(n)}]^{-1} y^{(n)} s^{(n)^T}}{y^{(n)^T} s^{(n)}} \\ &+ \frac{\left(y^{(n)^T} [H^{(n)}]^{-1} y^{(n)}\right)\left(y^{(n)} s^{(n)^T}\right)}{\left(y^{(n)^T} s^{(n)}\right)^2}\end{aligned} \tag{6.15}$$

The BFGS algorithm is quite attractive in that attains superlinear convergence rates but does not require computation of the Hessian [156]. An efficient limited-memory version of variant of the BFGS algorithm, known has L-BFGS has been developed to solve larger problems. The BFGS algorithm stores a dense $N \times N$ matrix to approximate the inverse Hessian matrix, which can lead to a significant memory footprint [156]. L-BFGS reduces the memory requirements by discarding less relevant early curvature information. The Hessian is then approximated using only M vectors where ($M << N$).

6.2 One-Shot Gradient-Based Optimization

In aerodynamic design the objective metric typically includes solving for computationally expensive numerical solutions of a system of partial differential equations. Classical nested optimization approaches fully solve the state equations and their accompanying adjoint and then update the designs on fully converged numerical solutions. For many aerodynamic design problems, this approach is computationally expensive and severely limit the number of design updates that can be performed. More recently, one-shot optimization methods have been developed in an attempt to reduce the cost of optimization by reframe the approach to iterate design while solving the state and adjoint equations. That is one-shot approaches seek to converge the state, adjoint, and design questions simultaneously through coupled iterative approaches.

Recalling the aerodynamic design problem can be represented as

$$\min_{\beta \in \mathcal{D}} J(\boldsymbol{\beta}, \boldsymbol{U}) \quad \text{subject to} \quad \boldsymbol{R}(\boldsymbol{\beta}, \boldsymbol{U}) = 0, \tag{6.16}$$

where $J(\boldsymbol{\beta}, \boldsymbol{U})$ is the objective metric, $\boldsymbol{R}(\boldsymbol{\beta}, \boldsymbol{U})$ is the residual of the primal (flow) solver, $\boldsymbol{\beta}$ is the design vector and $\boldsymbol{U}$ is the state vector. This constrained optimization problem can be reformulated as in unconstrained Lagrange multiplier problem form

$$\min \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{U}, \boldsymbol{\psi}) = J(\boldsymbol{\beta}, \boldsymbol{U}(\boldsymbol{\beta})) + \boldsymbol{\psi}^T \boldsymbol{R}(\boldsymbol{\beta}, \boldsymbol{U}(\boldsymbol{\beta})) \tag{6.17}$$

where the Lagrange multiplier $\boldsymbol{\psi}$ is the adjoint vector. Taking the derivative of Eq. (6.17) with respect to $\boldsymbol{U}$, $\boldsymbol{\beta}$, $\boldsymbol{\psi}$ and finding the minimum (e.g., when the derivative equals zero) leads us to the first-order necessary conditions for optimality known as the Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\psi}} = \boldsymbol{R}(\boldsymbol{\beta}, \boldsymbol{U}(\boldsymbol{\beta})) \quad = 0 \qquad \text{(State equation)} \tag{6.18a}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}} = \frac{\partial J}{\partial \boldsymbol{U}} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} \quad = \nabla_{\boldsymbol{U}} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{U}, \boldsymbol{\psi}) = 0 \qquad \text{(Adjoint equation)} \tag{6.18b}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \frac{\partial J}{\partial \boldsymbol{\beta}} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}} \quad = \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{U}, \boldsymbol{\psi}) = 0 \qquad \text{(Design equation)} \tag{6.18c}$$

The traditional nested optimization method identifies the KKT optimality point by solving Eq. (6.18a), then Eq. (6.18b), and finally performing a design update. The classical approach thus establishes feasibility of the state equation and the adjoint equation in each design update, which leads to the question of whether it is necessary to recreate feasibility if the design solution is not yet optimal [114].

One-shot optimization techniques seek to accelerate optimization by relaxing the feasibility requirement while the solution is far from the optimal design point. Instead of establishing feasibility for every design, one-shot approaches [182, 17, 158, 78] (also known as simultaneous analysis and design (SAND) [73], the all-at-once approach [32], and the

simultaneous optimization approach [191]) converge the state feasibility, adjoint feasibility, and design optimization simultaneously. Kusch [114] provides an excellent summary of one-shot methods, which she divides into two main strategies.

6.2.1 Single-step One-shot Optimization

The first strategy is the so-called single-step one-shot approach initially developed by Griewank [67]. The single-step one-shot strategy is derived using the discretized PDEs, and its formulation is based on fixed-point iteration. The single-step one-shot approach extends the piggy-back iteration [68, 69], which simultaneously solves the state equation together with the adjoint equation for a fixed design and iteration index, to include the design variable. The single-step one-shot approach can be interpreted as a Jacobi-type iteration procedure [114]. The base formulation approach has been extended within the literature to support additional constraints [115]; unsteady PDE constraints [72]; and multi-step Seidel-type iteration [18].

6.2.2 Pseudo-time Stepping One-shot Optimization

The second strategy is pseudo-time stepping techniques. Ta'asan [184], Iollo [95], and Hazra [77] proposed pseudo time-stepping approaches that could be employed to simultaneously solve the primal, adjoint, and design equations. More recently this approach has been formulated as an inexact reduced sequential quadratic programming (rSQP) approach [79, 77, 80], which is applied to the continuous optimization problem and then discretized afterwards. In this formulation, the step is

$$\begin{bmatrix} 0 & 0 & \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}^T \\ 0 & \frac{\partial^2 J}{\partial \boldsymbol{\beta}^2} & \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}}^T \\ \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} & \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\beta}} & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{U} \\ \Delta \boldsymbol{\beta} \\ \Delta \boldsymbol{\psi} \end{bmatrix} = \begin{bmatrix} -\nabla_{\boldsymbol{U}} \mathcal{L}\left(\boldsymbol{\beta}, \boldsymbol{U}, \boldsymbol{\psi}\right) \\ -\nabla_{\boldsymbol{\beta}} \mathcal{L}\left(\boldsymbol{\beta}, \boldsymbol{U}, \boldsymbol{\psi}\right) \\ -\boldsymbol{R}\left(\boldsymbol{\beta}, \boldsymbol{U}\left(\boldsymbol{\beta}\right)\right) \end{bmatrix} \tag{6.19}$$

where $\boldsymbol{U}$ is the primal state, $\boldsymbol{\psi}$ is the adjoint state, $\boldsymbol{\beta}$ is the design state with $\boldsymbol{U}^{(n+1)} = \boldsymbol{U}^{(n)} + \Delta \boldsymbol{U}^{(n)}$, $\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} + \Delta \boldsymbol{\beta}^{(n)}$, and $\boldsymbol{\psi}^{(n+1)} = \boldsymbol{\psi}^{(n)} + \Delta \boldsymbol{\psi}^{(n)}$. The formulation above includes the Hessian $\frac{\partial^2 J}{\partial \boldsymbol{\beta}^2}$ which can be cumbersome to calculate. However, the Hessian has been shown to primarily affect the rate of convergence [189] and Hazra [79] and Ozkaya [157] showed that Hessian can be replaced with an BFGS approximation or even steepest descent approximation. This pseudo-time stepping method for the KKT system effectively employs a Gauss-Seidel iteration strategy in which the state and adjoint variable are used in the design update as soon as they are available. Kusch highlights that the reduced SQP approach matches the explicit Euler method for a preconditioned system [114].

Both one-shot optimization strategies have been applied to aerodynamic optimization. In these applications, the one-shot techniques have exhibited improved efficiency relative to nested optimization strategies, so long as the one-shot optimizers have been carefully constructed with a suitable design space preconditioner. Underrelaxation of the design through a preconditioner is critical for attaining these cost reductions. Various approaches

have been proposed based the runtime, number of fixed-point iterations, or the contraction rate of the coupled fixed-point iterations [17]. If chosen correctly costs for the one-shot optimizer are typically only a small multiple of the cost for solving the underlying PDE [53, 158, 114].

Chapter 7

Reduced-Order Modeling

This chapter introduces a novel acceleration technique developed to accelerate the convergence of fixed-point iterative solvers. In the preceding chapters pseudo-time stepping formulations have been developed and detailed for computational fluid dynamic primal equations, adjoint sensitivity equations, and the design equation. Each of these solutions effectively time marches a solution of the form:

$$\frac{d\boldsymbol{s}}{dt} + \mathcal{N}(\boldsymbol{s}) = 0 \tag{7.1}$$

where s is an iterated state which could consist of the primal, adjoint, or design states (or some combination), and $\mathcal{N}$ is the nonlinear discretization operator [39] for the corresponding state equations. Each of these governing equations is high dimension with a large number of degrees of freedom. For instance, the governing flow equations, outlined in the previous sections, must solve for between 3-7 variables per grid point in the computational domain which can easily be on the order of millions of degrees of freedom. Moreover, design studies add an equivalent number of degrees of freedom for the sensitivity equation and a set of design variables further enlarging the size of the system.

The size of these systems makes the numerical solution of these equations computationally expensive. As discussed above, high-fidelity computational fluid dynamic models and sensitivity analysis are becoming increasingly valuable design tools, but their adoption is limited by their computational cost [129]. Techniques that accelerate the primal, adjoint, and design equation solution process is critical to further adoption of numerical optimization procedures in aerodynamic design.

Projection-based reduced-order modeling is an avenue for accelerating the iterative solution. Projection-based reduced-order modeling transform the high-order model of the governing equations to a simplified reduced-order model which provides an accurate approximation of the system but with considerably fewer states [7]. This is achieved by projecting the state and residual behavior onto lower dimensional subspaces thereby rewriting the governing equations in a compressed representation [137]. These subspaces are defined by a set of basis vectors $\boldsymbol{\Phi}$ which are chosen so that the relevant system dynamics captured

by the snapshots are well represented with a reduced number of states e.g.

$$\boldsymbol{s} \cong \boldsymbol{\Phi}\boldsymbol{\xi} \tag{7.2}$$

The basis and its Hermitian transpose can then be applied to Eq. (7.3) to form a reduced-order model [38]

$$\boldsymbol{\Phi}^T \frac{d}{dt}(\boldsymbol{\Phi}\boldsymbol{\xi}) + \boldsymbol{\Phi}^T \mathcal{N}(\boldsymbol{\Phi}\boldsymbol{\xi}) = 0 \tag{7.3}$$

where the basis vector coefficients $\boldsymbol{\xi}$ must be found.

7.1 Reduced-Order Model-based Convergence Acceleration of Fixed-Point Iterators [1]

Within this section, reduced-order modeling techniques are developed to accelerate fixed-point iterative methods, by considering the pseudo-time as the varied parameter. By using the pseudo-time as the varied parameter, the snapshots can be collected within a single solution procedure. Consider a fixed-point scheme for a general state vector $\boldsymbol{s}$, where the solution is updated in an iterative process

$$\boldsymbol{s}^{n+1} = s^n + \boldsymbol{R}(\boldsymbol{s}^n)\ . \tag{7.4}$$

Here n is the iteration number, $\boldsymbol{R}$ is the residual at each stage of the iteration process, and $\boldsymbol{s}$ is the state vector, which can represent the flow, sensitivity, or design states. This iterative process drives the solution to convergence which corresponds to a zero residual.

The idea for a path towards convergence acceleration is that there are conditions that allow the residual vector to be approximated as a linear function

$$\boldsymbol{R} \approx \boldsymbol{A}\,\boldsymbol{s} - \boldsymbol{b}\,. \tag{7.5}$$

This assumption is valid for the adjoint and forward sensitivity equations which are developed through linearization of governing flow equations. It has also been found to be valid for non-linear considerations like flow solutions that behave linearly near convergence [38, 39]. A key contribution of this dissertation is demonstration that the assumption can also be extended to one-shot design solutions as shown in Section 8.5.

The system in Eq. (7.10) is still the full-order system with N degrees of freedom, equal to the number of grid points times the number of governing equations. Projection-based reduced-order modeling can be applied to reform the equation on a reduced basis by substituting Eq. (7.2) into Eq. (7.10) yielding:

[1] This section, in part, is a reprint of the material as it appears in Aerospace Sciences 93 titled *"Reduced-order model-based convergence acceleration of reverse mode discrete adjoint"* (2019). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Elsevier's copyright policies permit the material in the paper to be included, in full or in part, within the author's thesis or dissertation for non-commercial purposes.

$$\boldsymbol{R} \approx \boldsymbol{A}\,\boldsymbol{\Phi}\boldsymbol{\xi} - \boldsymbol{b}\,. \tag{7.6}$$

This reduced-order model can then be solved to approximate the converged fixed-point solution by determining the solution that corresponds to when the residual is equal to zero

$$\boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{\xi}) \approx \boldsymbol{R}(\boldsymbol{s}) = 0. \tag{7.7}$$

Identifying the solution on the reduced-order model, requires the span of the basis vectors contains the fully converged (or at least a better) solution, i.e.,

$$\boldsymbol{s}\,|_{\boldsymbol{R}(\boldsymbol{s})=0} \in \ \text{span}\,(\boldsymbol{\Phi}) = \left\{ \sum_{i=1}^{M} \boldsymbol{\phi}_i \xi_i \quad | \quad \xi_i \in R \text{ and } \phi_i \in \boldsymbol{\Phi} \right\}. \tag{7.8}$$

Therefore, proper selection of the basis is critical. Over the course of several research papers and convergence proceedings [39, 107, 38, 108, 110] the present author and colleagues considered three basis vector types for convergence acceleration: snapshot, covariance, and orthogonal. Each of these methods forms a basis in an online manner by collecting snapshots of the solution state and residual behavior during preconvergent iterations. The fixed-point iteration process drives the state solution to convergence; therefore, the snapshots tend to cover the relevant solution space as the solution converges. Intuitively, increasing the number of basis vectors also expands the solution set span and will typically lead to better performance.

The procedure for reduced-order model-based acceleration using each of the basis vector types is detailed in the following subsections. For each method the basis formulation is introduced and the procedure for calculating the basis vector coefficients for approximation of the converged full state solution is discussed. Additionally, the applicability, strengths, and limitations of each technique for convergence acceleration of fixed-point iterators are highlighted.

7.1.1 Reduced-order Model Acceleration with Snapshot Basis Vectors

The initial form of the reduced-order model acceleration [39, 107] used the snapshot solution vectors as the basis vector set

$$\boldsymbol{\Phi} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{s}_1 & \boldsymbol{s}_2 & \boldsymbol{s}_3 & \dots & \boldsymbol{s}_M \\ | & | & | & & | \end{bmatrix}_{N \times M}, \tag{7.9}$$

where $\boldsymbol{s}_i$ is the state of the fixed-point solution of interest. The use of snapshots was motivated ease of implementation and by Ekici and Hall [42] who demonstrated a non-orthogonal basis formed from snapshot solutions could be used as the foundation of a reduced-order model.

Returning to the linearized residual function Eq. (7.10)

$$\boldsymbol{R} \approx \boldsymbol{A}\,\boldsymbol{s} - \boldsymbol{b}\,. \tag{7.10}$$

The reduced-order formulation is begun by substituting the linear combination of snapshot vectors $\boldsymbol{\Phi\xi}$ and their coefficients for the state $\boldsymbol{s}$ arriving at

$$\boldsymbol{R} \approx \boldsymbol{A}\,\boldsymbol{\Phi\xi} - \boldsymbol{b}., \tag{7.6}$$

This equation can then be used to solve for the appropriate basis vector coefficients, by recalling that a zero residual corresponds to the desired converged sensitivity solution. Therefore, the coefficients $\boldsymbol{\xi}$, are chosen such that the residual, $\dot{\boldsymbol{R}}$, projected onto the space spanned by the basis vectors is zero

$$\boldsymbol{R} = \boldsymbol{A\Phi\xi} - \boldsymbol{b} = \boldsymbol{0}\,. \tag{7.11}$$

The size of the system in Eq. (7.33) is N e.g., the number of state equations times the number of grid points, which can be computationally demanding to solve as the problem size grows. However, the size can be reduced by pre-multiplying by the transpose of $\boldsymbol{\Phi}$, which reduces the system size to the number of snapshots M

$$\boldsymbol{\Phi}^T\boldsymbol{A\Phi\xi} = \boldsymbol{\Phi}^T\boldsymbol{b}\,. \tag{7.12}$$

This equation can be solved more easily due to the reduced dimension. To solve Eq. (7.35) the right-hand side vector, $\boldsymbol{b}$ and Jacobian, $\boldsymbol{A}$, need to be calculated. First, the right-hand side vector can be calculated by setting the state in Eq. (7.10) to zero and performing a single iteration. This residual can then be used to find $\boldsymbol{b}$ according to

$$\boldsymbol{R}\,(\boldsymbol{s} = \boldsymbol{0}) = -\boldsymbol{b}\,. \tag{7.13}$$

In practice, it is not always possible to set the state vector to zero, for example in density-based flow solvers require a positive non-zero density state. In instances where a zero state is not allowed, a matrix-fee Jacobian approximation method [112] can be used to approximate the Jacobian vector product through second-order central-difference[38]

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{s}}\boldsymbol{s} = \boldsymbol{A}\boldsymbol{s} \approx \frac{\boldsymbol{R}\,(\boldsymbol{s} + \epsilon\boldsymbol{s}) - \boldsymbol{R}\,(\boldsymbol{s} - \epsilon\boldsymbol{s})}{2\epsilon} \tag{7.14}$$

Here ϵ is a small perturbation. Using this approximation, the right-hand side vector $\boldsymbol{b}$ can then be calculated using Eq. (7.10)

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} - \boldsymbol{R}(\boldsymbol{s}) \tag{7.15}$$

Following calculation of the right-hand side vector the Jacobian needs to be found. Rather than explicitly calculating the Jacobian, $\boldsymbol{A}$, in Eq. (7.35) directly, the matrix-free approach outlined by Ekici and Hall [42] is adopted to compute $\boldsymbol{A\Phi}$ column by column. The column

of each matrix-vector product, $\boldsymbol{A\Phi_i}$, is calculated by initializing the state solver with the first basis vector and calculating the resulting residual of the adjoint solver by running it for a single iteration. The first column of the matrix-vector product is then obtained from

$$\boldsymbol{A\phi}_1 = \boldsymbol{R}\left(\boldsymbol{\phi}_1\right) + \boldsymbol{b}\,. \tag{7.16}$$

The remaining matrix-vector products $\boldsymbol{A\phi}_2, ..., \boldsymbol{A\phi}_M$, are found in the same manner. It should be noted that the snapshot solutions and their residual are naturally calculated during the iterative solution. They can be stored during the nominal solution to alleviate the need for recalculation.

With $\boldsymbol{b}$, $\boldsymbol{\Phi}$, $\boldsymbol{A\Phi}$ known, the weighting coefficients, $\boldsymbol{\xi}$, can be found by solving the reduced-order problem Eq. (7.35). The full-order state vector can then be approximated through the linear combination given by Eq. (7.2). As a result, the converged state solution can be approximated using a number of early iteration unconverged state and residual vectors. The accuracy of converged state approximation can be checked by loading the state into the solver and resuming iteration. The solution of the reduced-order systems is typical on the order of M iterations. Therefore, the iterative solution can be accelerated if the residual of the projected state is smaller than the residual that could be achieved by an additional M iterations. As will be shown in Chapter 8 this approach proves to be a very efficient way to accelerate fixed-point iterative solvers.

7.1.2 Reduced-order Model Acceleration with Covariance Basis Vectors

For improved resolution an alternate basis can be created by subtracting the snapshot mean from the snapshots

$$\boldsymbol{\Phi} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{s}_1 - \bar{\boldsymbol{s}} & \boldsymbol{s}_2 - \bar{\boldsymbol{s}} & \boldsymbol{s}_3 - \bar{\boldsymbol{s}} & \dots & \boldsymbol{s}_M - \bar{\boldsymbol{s}} \\ | & | & | & & | \end{bmatrix}, \tag{7.17}$$

where

$$\bar{\boldsymbol{s}} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{s} \tag{7.18}$$

is the mean of the snapshots. For these basis vectors the solution approximation is now defined as a linear combination of the basis vectors plus the snapshot mean

$$\boldsymbol{s} = \boldsymbol{\Phi\xi} + \bar{\boldsymbol{s}}. \tag{7.19}$$

Again, since a zero residual corresponds to a converged solution, the coefficients $\boldsymbol{\xi}$ are chosen such that the residual of the approximation is assumed to be small (or machine accuracy zero)

$$\boldsymbol{R}\left(\boldsymbol{\Phi\xi}+\bar{\boldsymbol{s}}\right)\approx 0. \tag{7.20}$$

Substituting the approximation Eq. (7.19) into the governing equations Eq. (7.10) yields

$$\boldsymbol{R}\left(\boldsymbol{\Phi\xi}+\bar{\boldsymbol{s}}\right)=\boldsymbol{A}\left(\boldsymbol{\Phi\xi}+\bar{\boldsymbol{s}}\right)-\boldsymbol{b}=\boldsymbol{A}\bar{\boldsymbol{s}}+\boldsymbol{A\Phi\xi}-\boldsymbol{b}=0 \tag{7.21}$$

By recalling

$$\boldsymbol{R}\left(\bar{\boldsymbol{s}}\right)=\boldsymbol{A}\bar{\boldsymbol{s}}-\boldsymbol{b}, \tag{7.22}$$

Eq. (7.21) can be simplified due to linearity to

$$\boldsymbol{A\Phi\xi}+\boldsymbol{R}\left(\bar{\boldsymbol{s}}\right)=0, \tag{7.23}$$

The size of the system is still of the same order as the original governing state equations. However, by pre-multiplying the equation by the transpose of the basis vector $\boldsymbol{\Phi}^T$ the system size can be reduced to M, the number of snapshots

$$\boldsymbol{\Phi}^T\mathbf{A}\boldsymbol{\Phi\xi}=-\boldsymbol{\Phi}^T\boldsymbol{R}\left(\bar{\boldsymbol{s}}\right)\,. \tag{7.24}$$

Once again the Jacobian matrix is not computed directly. Instead $\boldsymbol{A\Phi}$ is computed column by column as a matrix vector product

$$\boldsymbol{A}\boldsymbol{\phi}_1=\boldsymbol{A}\left(\boldsymbol{s}_1-\bar{\boldsymbol{s}}\right)\,. \tag{7.25}$$

This can be simplified by the fact that

$$\boldsymbol{A}\boldsymbol{\phi}_1=\boldsymbol{A}\boldsymbol{s}_1-\boldsymbol{A}\bar{\boldsymbol{s}}\,, \tag{7.26}$$

where $\boldsymbol{A}\boldsymbol{s}_1$ and $\boldsymbol{A}\bar{\boldsymbol{s}}$ can be directly related to the solution residuals through

$$\boldsymbol{A}\boldsymbol{\phi}_1=\underbrace{\left(\boldsymbol{R}\left(\boldsymbol{s}_1\right)+\boldsymbol{b}\right)}_{\boldsymbol{A}\boldsymbol{s}_1}-\underbrace{\left(\bar{\boldsymbol{R}}\left(\bar{\boldsymbol{s}}\right)+\boldsymbol{b}\right)}_{\boldsymbol{A}\bar{\boldsymbol{s}}} \tag{7.27}$$

Observe, the right-hand side vectors cancel out, which leaves

$$\boldsymbol{A}\boldsymbol{\phi}_1=\boldsymbol{R}\left(\boldsymbol{s}_1\right)-\boldsymbol{R}\left(\bar{\boldsymbol{s}}\right)\,. \tag{7.28}$$

This procedure is repeated for the remaining vector products. With all of the other terms known, the weighting coefficients, $\boldsymbol{\xi}$, can be simply computed by solving Eq. (7.24). The weighting coefficients can then be used to construct the approximate "converged" state solution through Eq. (7.19). The accuracy of the approximation depends on how well the solution behavior is captured by the snapshots, and the quality of the approximation can be evaluated by initializing the fixed-point solver with the approximated solution and performing a single iteration after the projection. The resulting residual can be used to gauge the solution accuracy. After restarting the solution, the ROM acceleration procedure can be repeated until the desired solution accuracy is reached. A key benefit of this approach is

removal of the need to calculate the right-hand side vector $\boldsymbol{b}$. Which is useful when a solver cannot be initialized from a zero solution. This allows general consideration of fixed-point iterators for primal, adjoint, and design solutions. Finally, this approach has been shown to provide better projections that the technique using the snapshot basis vectors.

7.1.3 Reduced-order Model Acceleration with Orthogonal Basis Vectors

The final basis is an orthogonal basis developed through proper-orthogonal decomposition. This basis is generated by collecting the M snapshots collected in a matrix column by column

$$\boldsymbol{S} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{s}_1 & \boldsymbol{s}_2 & \boldsymbol{s}_3 & \dots & \boldsymbol{s}_M \\ | & | & | & & | \end{bmatrix}_{N \times M}, \tag{7.29}$$

Proper orthogonal decomposition is performed on this snapshot matrix to form an orthogonal basis by solving the eigenvector problem defined by

$$\boldsymbol{S}^T \boldsymbol{S} \boldsymbol{\Psi} = \boldsymbol{\lambda} \boldsymbol{\Psi}, \tag{7.30}$$

where λ and $\boldsymbol{\Psi}$ are the eigenvalues and right eigenvectors of the symmetric matrix $\boldsymbol{S}^T \boldsymbol{S}$. Since the matrix is symmetric, each eigenvector corresponding to a unique eigenvalue will be orthogonal. These eigenvectors are then multiplied by the snapshot matrix to form an orthogonal basis

$$\boldsymbol{\Phi} = \boldsymbol{S} \boldsymbol{\Psi} \tag{7.31}$$

where:

$$\boldsymbol{\Phi} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3 & \dots & \boldsymbol{\phi}_M \\ | & | & | & & | \end{bmatrix}_{N \times M}. \tag{7.32}$$

The orthogonal basis has the benefit that it maximizes the span of the basis vectors of a given order, which is beneficial since the model is limited to the solutions within its span Eq. (7.8). The projected solution is found by substituting this reduced-order model approximation from Eq. (7.2) into the full-order governing equations, Eq. (7.10), yielding

$$\boldsymbol{R} = \boldsymbol{A} \boldsymbol{\Phi} \boldsymbol{\xi} - \boldsymbol{b}, \tag{7.33}$$

which can be used to determine the appropriate weighting coefficients that drive the residual of the adjoint solver to machine accuracy, i.e.,

$$\boldsymbol{R} = \boldsymbol{A} \boldsymbol{\Phi} \boldsymbol{\xi} - \boldsymbol{b} = \boldsymbol{0}. \tag{7.34}$$

Again, the order of the system is reduced by pre-multiplying by $\boldsymbol{\Phi}^T$

$$\boldsymbol{\Phi}^T \boldsymbol{A} \boldsymbol{\Phi} \boldsymbol{\xi} = \boldsymbol{\Phi}^T \boldsymbol{b} \,. \tag{7.35}$$

The determination of the basis vector calculation begins by calculating the right-hand side vector, $\boldsymbol{b}$, first. This is done by initializing the adjoint solution vector to zero and performing a single iteration. The residual of the adjoint solver can then be used to find $\boldsymbol{b}$ according to

$$\boldsymbol{R}\left(\boldsymbol{0}\right) = -\boldsymbol{b} \,. \tag{7.36}$$

Again, the matrix-free approach outlined by Ekici and Hall [42] is followed to compute $\boldsymbol{A\Phi}$ column by column. This is accomplished by initializing the state solver with the first basis vector and calculating the resulting residual by running it for a single iteration. The matrix vector product $\boldsymbol{A}\boldsymbol{\phi}_1$ is then obtained from

$$\boldsymbol{A}\boldsymbol{\phi}_1 = \boldsymbol{R}\left(\boldsymbol{\phi}_1\right) + \boldsymbol{b} \,. \tag{7.37}$$

The remaining matrix-vector products, $\boldsymbol{A}\boldsymbol{\phi}_2, ..., \boldsymbol{A}\boldsymbol{\phi}_M$, are computed in a similar manner. In the snapshot approach, the residual can be stored for each snapshot during collection, but for orthogonal basis vectors the residual must be calculated for each basis vector. Thus, the orthogonal basis vector approach has an increased cost associated with the basis formation eigenproblem and residual calculation. Furthermore, the orthogonal basis vectors may contain negative or zero values, which can be incompatible with some fixed-point iterators like a density-based CFD solver. However, in several applications like fixed-point sensitivity solvers zero or negative values are of no concern and the orthogonal basis vector approach can provide significantly more accurate approximations, as shown in during the verification of the reduced-order model acceleration techniques in Chapter 8.

Chapter 8

Demonstration of Reduced-order Model Acceleration

This chapter presents case studies performed to demonstrate the reduced-order model-based acceleration techniques presented in Chapter 7. The computational cost reduction that can be achieved is demonstrated for a range of fixed-point iterators. Initially, the reduced-order model acceleration technique is applied to accelerate the flow solution of a harmonic balance solver considering flow over an oscillating RAE 2822 airfoil. Next, the reduced-order model convergence acceleration technique is applied to accelerate a continuous adjoint solver derived for the quasi-one-dimensional Euler equations. Three case studies are considered to evaluate snapshot collection approaches for varying basis vectors. The reduced-order model acceleration technique is then applied to discrete adjoint solvers. The first discrete adjoint solver case considered the reduced-order model acceleration technique applied to a brute force implementation, which prevented continuation of the iterative solution following projection. To overcome this limitation, the reduced-order model acceleration technique was next applied to a fixed-point iterative primal time-stepping discrete adjoint solver. Finally, the reduced-order model acceleration technique was applied to accelerate a one-shot optimizer. The range of applications considered here demonstrates the robustness of the approach and its ability to generalize to a broad set of fixed-point iterators.

8.1 Acceleration of an Unsteady Harmonic Balance Solution

The first case presented here, considers application of the reduced-order model acceleration technique to accelerate a harmonic balance primal flow solution and its adjoint. The initial application of the reduced-order model acceleration technique to computational fluid dynamic flow solutions was performed by Djeddi, Kaminsky and Ekici in [37]. The present case study presents extension of the technique to unsteady flows through acceleration of a mathematically steady harmonic balance solver.

8.1.1 Flow over an Oscillating RAE 2822 Airfoil

The case study considered unsteady two-dimensional viscous flow over an RAE 2822 airfoil. The flow was defined by a Mach number of $M_\infty = 0.78$ with a nominal angle of attack $\alpha_0 = 0.0°$. The airfoil was forced to oscillate with small amplitude, $\alpha_1 = 1.0°$, pitching oscillations at a reduced frequency of $k = 0.2$. The grid employed for the RAE2822 is shown in Fig. 40.

The unsteady flow behavior was resolved using a harmonic balance solver considering two harmonics. Thus, five coupled sub-time solutions were simulated. The surface pressure distributions for each timestep are presented in Fig. 41. The harmonic balance solver employed the Fifth-order Runge–Kutta scheme detailed in Section 3.3 and the convergence history is included in Fig. 42.

With this baseline convergence rate established for the primal solver, the investigation turns to the evaluation of the reduced-order model convergence acceleration technique. Multiple applications of the reduced-order model-based acceleration technique were performed using the covariance basis vectors. The different cases, detailed in Table 10, considered the reduced-order model acceleration technique with snapshot collection procedures varying initial snapshots collection points. The convergence histories of the reduced-order model accelerated cases are compared with the baseline convergence rate in Fig. 43. From the table and figure it can be seen that the reduced-order model-based acceleration technique dramatically reduces the number of iterations required to reach convergence. The average reduction in iterations to reach machine accuracy $R(U) < 1\times10^{-18}$ was 34.8 percent. Varying the initial snapshot location led to only minor deviations, illustrating the reduced-order model acceleration was robust to snapshot location. This case successfully demonstrates the reduced-order model acceleration can be deployed to accelerate unsteady flow solutions considered by primal harmonic balance solvers.

8.2 Acceleration of a Continuous Adjoint Solver[1]

The proposed ROM-based acceleration technique was also applied to the continuous adjoint solver cases outlined in Section 5.2. The same three cases were considered, but this time the adjoint vector, $\boldsymbol{\psi}^T$, solution was accelerated using the reduced-order model convergence acceleration technique. This case was used to evaluate snapshot collection procedures and the effects of delayed starts, snapshot intervals, snapshot quantities, and repeated application. Unlike the density-based flow solver considered in the previous case, the continuous adjoint sensitivity fixed-point iterative solver does not have any limitations on the state values. As a result, this solver also supports comparison of the different basis vector types.

[1]This section, in part, is a reprint of the material as it appears in the International Journal of Numerical Methods in Fluids 86 (9), 582-606 titled "*Convergence acceleration of continuous adjoint solvers using a reduced-order model*" (2018). Authors: **Andrew Kaminsky**, Reza Djeddi, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Wiley's copyright policies permit the use of the article in full or in part within the author's thesis or dissertation for non-commercial purposes.

8.2.1 Case 1: fully subsonic nozzle

The first case considers a nozzle with a fully subsonic flow field. The case is defined by a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.9899. The traditional (non-accelerated) continuous adjoint solver is used as a baseline to evaluate the performance of the reduced-order model acceleration technique. The flow solution was obtained, and then the adjoint vector was solved for using a continuous adjoint solver.

To establish a baseline for the reduced-order model acceleration, a traditional (non-accelerated) continuous adjoint solver was used to solve for the adjoint variables. The convergence rate of the adjoint solution was similar to that found for the flow solution and is included in Fig. 44a. The adjoint vector is verified in Fig. 44b using the results presented by Lozano and Ponsin [128]. Note, verification of the traditional flow solution and continuous adjoint solution for case was considered previously in detail in Section 5.1.1.

Convergence acceleration with snapshot basis vectors

The reduced-order model acceleration of the continuous adjoint solver was first considered using snapshots as basis vectors for this case. The snapshot-based acceleration, the simplest of the proposed techniques, utilizes the snapshots as the basis vectors to project the converged solution. Several parameters including snapshot quantity, first snapshot placement, iteration interval between snapshots, and snapshot span, are used to control the convergence acceleration technique.

The acceleration technique's dependence on the snapshot quantity was evaluated first. To isolate the snapshot count as the sole variable of interest, the snapshots were collected over the same iteration span by varying the number of iterations between each snapshot for each case. This ensured that all the snapshots started and ended on the same iteration. Each time the snapshot count was halved, the acceleration technique effectively samples every other snapshot. The global residual convergence rates for acceleration utilizing increasing snapshot quantities are presented in Fig. 45.

Table 11 includes the snapshot collection parameters and the number of iterations to reach convergence, $R(\boldsymbol{\psi}) = 1 \times 10^{-16}$, for each case. Table 11 also presents the iteration reduction for each case. An idea of the cost associated with the ROM-based acceleration at different snapshot quantities can be observed by examining the correlation between the reduction in iterations and the time. As the snapshot count increases, the computational time reductions are diminished relative to the reduction in iteration count due to the time associated with the ROM-based convergence acceleration.

For this case, it is clear that collecting additional snapshots causes increasingly large global residual drops. This is likely due to the increased span of the solution space resulting from the additional degrees of freedom in Eq. (7.8). It should be noted that all the acceleration cases reach machine accuracy before the traditional method. This occurs despite the fact that the 11 snapshot case residual actually jumps to a value higher than that found by the traditional method. The convergence rate increases after the jump so that even the 11 snapshot case is able to catch up to and eventually surpass the convergence of the

traditional method. However, it is apparent that for the case considered here 11 snapshots are insufficient to provide enough information to achieve a drop in the adjoint residual after the projection.

A comparison of the projected adjoint vectors with the fully converged solution and the adjoint vector at the last snapshot is presented in Fig. 46. It is clear that all of the projected adjoint solutions are closer to the fully converged solution; the 41, 81, and 161 snapshot cases are visually indistinguishable from the fully converged profile. Examining the 11 snapshot profile, it can be seen that the residual jump in the 11 snapshot case is produced by the steep gradient oscillation in the projected solution near $x = 0.25$. After the oscillations are attenuated, the profile matches the converged solution more closely which leads to an increased convergence rate. The difference of each projection from the converged value is also included in Fig. 47 on a log scale, which demonstrates how accurately the projections match the converged solution.

Thus far, only a single application of the convergence acceleration technique has been considered. However, the proposed method can be applied multiple times during a solution procedure to further improve the convergence rate. The performance of repeated application of the acceleration technique is presented for each snapshot quantity using double precision in Fig. 48a and quadruple precision in Fig. 48b. Examining the convergence rates in Fig. 48a for multiple applications of the acceleration it can be seen that all but the 11 snapshot case reach a convergence limit at a global residual value between 1×10^{-8} and 1×10^{-10} using double precision floating point numbers. The correlation-based acceleration technique is unable to project to a solution with improved accuracy at this point. This results from the fact that as the solution converges, the snapshot values will be increasingly similar and thus become less linearly independent making the reduced-order model ill-conditioned. As a result, the potential solution space given by Eq. (7.8) diminishes to a point where solution improvement is no longer attainable. However, if application of the acceleration technique is ceased once the residual convergence limit (1×10^{-10}) is reached the solution will continue to converge until machine accuracy.

To test the assertion that the method fails due to nearly linearly dependent snapshots, the solver was run using quadruple precision. Fig.48b shows that using quadruple precision eliminates the artificial convergence limit. However, it also increases the computational cost as well as the memory requirement by a factor of two. The additional precision helps to distinguish between the snapshots as they become increasingly linearly dependent. However, even quadruple precision fails to add additional convergence acceleration for repeated application of the 11 and 21 snapshot cases with these parameters, so stabilization of the method is desirable. While quadruple precision succeeds in reducing the total number of iterations for the 41, 81 and 161 snapshot cases, the up to four-fold cost increase of each iteration is untenable. Thus, another method is desired to make the acceleration tractable for convergence to machine accuracy.

Convergence acceleration with orthogonal basis vectors

To circumvent problems with ill-conditioned ROMs, proper orthogonal decomposition can be used to maximize the span of the solution space, Eq. (7.8), by forming orthogonal basis vectors. The convergence acceleration for application of the correlation-based acceleration technique with orthogonal basis vectors is presented for cases using increasing quantities of snapshots in Fig. 49a. The orthogonal basis vectors used here were formed from the same snapshots used for the convergence acceleration shown in Fig. 45. It can be seen that a single application the orthogonal basis vectors only results in a slight improvement in the convergence drop. The most notable drop occurs for the 161 snapshot case which now outperforms the 81 snapshot case. The 161 snapshot case projection also converges past the artificial convergence limit found in the prior case using double precision arithmetic.

Thus far, sampling of the first snapshot has been delayed until iteration 500. Fig. 49b presents convergence acceleration for the same snapshot selection procedure but with snapshot sampling beginning at iteration 2 instead of iteration 500. Examining Figs. 49a and 49b it is evident that delayed snapshot collection improves the performance of the acceleration technique. Table 12 presents the snapshot collection parameters and the relative acceleration improvement offered by delayed snapshot selection.

Delaying snapshot collection offers considerable improvement, particularly for the cases with higher snapshot counts. Even this small delay achieves an additional cost reduction of up to 68%. By delaying snapshot collection, sampling of the initial transient behavior of the solver is avoided. This leads to less oscillatory basis vectors which leads to improved convergence acceleration. Delaying the first snapshot location even further conveys improved convergence acceleration until an eventual plateau is reached, as shown in Fig. 50, for the 41 snapshot case. While it is clear an optimal delay for the first snapshot location exists, all of the locations provide convergence acceleration. Additionally, for the best performance out of a single application of the convergence acceleration technique it may be desirable to delay the application. However, it could also be argued that earlier improvement in the global residual is actually more significant.

The concern over when to begin snapshot sampling is largely alleviated by applying the convergence acceleration technique multiple times. Convergence acceleration achieved through multiple applications of the POD-based technique with snapshot sampling starting at iterations 2 and 500 is shown in Figs. 51a and 51b respectively. Orthogonal basis vectors convey considerable improvement in convergence rates for cases that make use of repeated application of the acceleration technique. Again, the convergence limit found when using snapshot basis vectors is no longer present. Delaying sampling of the first snapshot still yields improved performance but to a much smaller degree, and both repeated application cases significantly outperform even the very best single application case. The 81 snapshot delayed application case achieved the greatest speed up, reducing the total computational cost by 90%.

Only the 41, 81, and 161 snapshot cases were presented in Fig. 51, because despite the orthogonal basis, the 11 and 21 snapshot cases defined using the same snapshot selection parameters were unsuccessful in delivering additional convergence acceleration. Multiple

applications of the convergence acceleration technique using fewer snapshots results in unstable projections that can sometimes result in global residual drops but more frequently cause undesirable jumps. Two different techniques have been found to stabilize the performance of repeated application cases that utilize fewer snapshots. Both techniques were developed by recalling that the efficacy of a reduced-order model is dependent upon how much of the solution behavior is captured [137, 21].

The first approach is performed by increasing the space between consecutive acceleration spans. Previously, a one span delay between the last projection and the first snapshot of the following projection was used. An increased delay allows the solution to converge even further beyond the solution that was last projected. By sampling snapshots later, any transient behavior from oscillations in the previous projection has been dampened out, and new solution behavior is captured from the next cycle of snapshots. The stabilization provided by an increased delay is demonstrated for the 21 snapshot case in Fig. 52. Looking closely, it appears that there is a trade-off between the frequency with which the acceleration technique is applied and the stabilization provided by increasing the delay between acceleration cycles. Performance improves from the increased stability through the 9 span delay. For larger delays, the increased delay leads to fewer applications of the acceleration technique which leads to less acceleration comparatively.

Occasionally, using this approach the convergence rate jumps following the projection. This is likely due to small oscillations within the projected solution, like those shown in Fig. 46. These oscillations occur because the limited basis vectors provide fewer degrees of freedom to completely smooth out the profile. Using more basis vectors eliminates these jumps as shown in Fig. 53, where the 41 snapshot case is presented for comparison. However, it should be noted that despite the jumps, repeated application after the stabilization can outperform the single application case. The repeated application offers an improved slope following each projection. Examining the convergence history of the single application case, in Fig. 45, the initially steep slope following the projection deteriorates as the solution progresses leading to a concave shape. The repeated application ensures the steep convergence history slope is maintained. Finally, it is easy to evaluate a jump in the global residual, and if it is deemed too large it is easy to reset the solution to the pre-projection value.

The second approach is the increased interval approach. Before, the iteration interval between snapshots was selected so that the span of case was kept constant. This was done merely for evaluative purposes. By increasing the span, the reduced-order model can capture more of the solution behavior since the snapshots are collected over a longer pseudo-time. The improved stability conferred by increased iteration intervals for a 21 snapshot case is presented in Fig. 54. Again, there is a trade-off between the convergence acceleration application frequency and the stabilization provided by increasing the interval between snapshots, and the ideal snapshot interval for this case occurs at 32 iterations and achieves a 50% cost reduction. It is also notable that the increased interval approach slightly outperforms the increased delay approach. However, either approach can be used to stabilize multiple applications of convergence acceleration with fewer snapshots.

Lower snapshot counts are appealing because of the reduced memory requirements. For problems with more degrees of freedom, storing 161, 81, or even 41 snapshots within the Random Access Memory (RAM) could become unfeasible. However, the acceleration technique can be performed using either an input-output (I/O) efficient or a memory efficient approach. In the I/O efficient approach the solution and their residuals are written into external files at each snapshot. After all of the snapshots have been collected, the snapshot solutions and residuals are all read into the RAM. This requires enough RAM to store $N \times (M+1)$ values, where N is the number of degrees of freedom and M is the number of snapshots. This limits the number of I/O operations to only $2M$, since each snapshot and residual is written and then read only once. In the memory efficient approach, the left- and right-hand sides of Eq. (7.35) are built term by term by loading only 2 snapshots at a time, instead of loading all of the snapshots at once. This approach limits the memory requirements to $2N$ but requires $M^2 + 2M$ I/O operations. The additional I/O operations slow down the projection procedure but provide extension to problems with considerably more degrees of freedom [38].

8.2.2 Nozzle with subsonic inlet supersonic outlet

The continuous adjoint second case considered is specified by a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.52. This case corresponds to the case of Giles and Pierce [62] designated by $H_{in} = 4$ and $p_{t_{in}} = 2$. As shown in Section 5.1.2, the adjoint solution using the traditional approach compared well with that of Giles and Pierce [62] in Figs. 23a and 23b respectively. In this case a finer 899-node grid was used to show convergence acceleration for a case with a reduced convergence rate due to grid resolution.

Convergence acceleration with orthogonal basis vectors

The convergence acceleration technique utilizing orthogonal basis vectors was applied to this transonic flow problem. This case was used to further examine the importance of the interval between snapshots. Application of the convergence acceleration utilizing increasing spans of 80, 160, 320, and 640 iterations is shown in Fig. 55. For these comparisons the initial snapshot was taken at iteration 1000 to encourage stability among even the smallest span. As the span of the snapshots is increased, performance of the acceleration technique is improved. The convergence behavior for this case is smoother than the prior case which leads to large drops even for very small spans. The typical global residual drop is increased from 3 orders of magnitude to almost 10 by quadrupling the span. Overall, a single application of the acceleration technique can reduce the computation cost by over 50% for this case.

8.2.3 Nozzle with shocked flow

Flow solution and adjoint vector verification

The third continuous adjoint acceleration case reconsiders the nozzle with a transonic shocked flow field with a total inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.84317. This case is consistent with the $H_{in} = 4$, $p_{t_{in}} = 2$, and $p_{ex} = 1.6$ case presented by Giles and Pierce [62]. The adjoint solution will be compared with that of Giles and Pierce [62], as shown in Section 5.1.3.

Convergence acceleration with orthogonal basis vectors

As previously mentioned, the ROM-based acceleration scheme is easily implemented with existing convergence acceleration techniques. Here performance of the correlation-based acceleration technique with orthogonal basis vectors applied to a solver accelerated with a multigrid scheme is evaluated using the shocked flow condition. To establish a baseline, a single application of the ROM acceleration technique is performed without multigrid, and the convergence histories are compared with those of the varying levels of multigrid schemes in Fig. 56.

The cost reduction associated with each level of multigrid and the ROM convergence acceleration technique applied separately is included in Table 13. As found in the previous two cases, increasing the snapshot quantity again leads to larger global residual drops. It is interesting to note that there is no additional acceleration added by increasing the snapshot count from 21 to 41 snapshots. This is a known drawback of POD-based reduced-order models. Increased modes do not guarantee improved performance, and they can sometimes perform worse [169]. However, the general trend is still established by the improved performance offered by the 81 and 161 snapshot case. The 81 snapshot case performs the best in terms of computational time, and it can be observed that the computational time increases significantly for the 161 snapshot cased even though the iteration counts are similar. The time required for storage of the snapshots and the solution of the basis vector coefficients increases with the number of snapshots and as a result lower snapshot counts perform more efficiently.

Fig. 56 also demonstrates that both multigrid levels significantly reduce the number of iterations required to reach convergence, but the 1 level multigrid slightly outperforms the 2 level multigrid scheme when considering computational time as shown in Table 13. It should be noted that for this case the multigrid scheme outperforms all the ROM-based acceleration applications except the 81 snapshot case, with respect to computational time.

An advantage of the proposed scheme is that it can easily be compounded with other acceleration schemes, and application of the ROM-based acceleration scheme to a solver utilizing one level of multigrid is presented in Fig 57. A single application of the ROM-based convergence acceleration technique leads to a further reduction in required iterations and more importantly computational time for each snapshot quantity as shown in Table 14. The ROM-based acceleration applied to the multigrid scheme provides additional computational

cost reduction relative to the multigrid scheme alone, and the 81 snapshot case almost halves the computational time.

Note that for the multigrid case the snapshot collection was started earlier and over a smaller span. This was done for two reasons. First if application of the convergence technique was delayed until iteration 1500 on the multigrid scheme the solution would have been nearly converged, and thus not a fair assessment of the ROM-based acceleration technique. Secondly and more interestingly, the multigrid scheme actually appears to stabilize the performance of the ROM-based acceleration technique. Multigrid schemes remove low-frequency oscillation components from the solution behavior by solving the solution on a coarser grid, where they act as higher-frequency oscillations. When prolongation is performed to extend the coarse residuals to the fine grid, interpolation is required where data is not present. This interpolation leads to an inherent residual smoothing. The removal of the low-frequency oscillations coupled with the inherent residual smoothing allows the ROM-based acceleration technique to be applied earlier with smaller snapshot spans, because the residuals and basis vectors are less oscillatory.

Finally, the ROM-based acceleration technique was applied multiple times to the multigrid scheme, and the convergence history is included in Fig. 58 and Table 15. Multiple applications using 81 and 161 snapshots reduces the number of iterations required, but actually increases the computational time. From Fig. 58 it is clear that after the first application, there is little gained from additional applications of the ROM-based acceleration technique. Additionally, the cost associated with the ROM-based acceleration technique increases with the number of snapshots used. As a result, it is recommended that application of the ROM-based scheme be halted after a global residual level around 1×10^{-12} when utilizing large snapshot quantities in double precision, due to the diminishing returns.

While performance for the higher snapshot counts deteriorated when considering multiple applications, the performance of cases considering fewer snapshots improved. Smaller snapshot quantities require less RAM or storage space and are thus preferred to larger snapshot quantities. Multiple applications of the small quantity snapshot cases to the multigrid scheme, in particular the 21 snapshot case, perform nearly as well as the best case and thus is a valuable alternative when memory is a concern.

8.3 Accelerated Nested Optimization Study through Projected Discrete Adjoint Sensitivities [2]

The third case study considers application of the reduced-order model acceleration technique to accelerate a nested design optimization loop with a brute-force discrete adjoint

[2]This section, in part, is a reprint of the material as it appears in AIAA Paper 2017-0037 titled *"An Efficient Reduced-Order Model for Accurate Projection of Adjoint Sensitivities"* (2017). Authors: **Andrew Kaminsky**, Reza Djeddi, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Andrew Kaminsky**, Reza Djeddi, and Kivanc Ekici.

solver. The case originally presented in [107] was the first application of the technique to adjoint solvers. It is important to highlight that this case considered application to the brute-force automatic differentiation code. As discussed in Section 4.3.2, the brute-force solver is not a fixed-point iterator. Modifying the working variable in this solver will corrupt the chain rule and lead to an incorrect sensitivity. Therefore, in this case the reduced-order model acceleration technique will be used to approximate the converged solution, but the iterative solution will not be resumed.

8.3.1 Sensitivity Projection for Inverse Design of a 2D Cascade

This case considered an inverse design problem for a compressor stage. The intent of this inverse design problem is to recover the Tenth Standard Configuration [52] by minimizing the difference between the surface pressure distribution of the current shape and that of the Tenth Standard Configuration. This case has also been considered by Wu et al. [205] and Huang and Ekici [94].

The Tenth Standard Configuration is a steady two-dimensional inviscid compressor cascade consisting of a modified NACA 0006 airfoil with a stagger angle of 45 degrees and a gap to chord ratio of 1.0. For this case the transonic inlet condition was considered with an inlet Mach number of 0.8 and an inlet flow angle of 58 degrees. The initial shape was obtained by deforming the Tenth Standard geometry by adding a sine wave with an amplitude of 1% chord length to both the pressure and suction sides. The initial grid has been included in Fig. 59.

The inverse design problem was solved by defining the objective function as

$$J = \int_s \frac{1}{2} \left(p - p_{target}\right)^2 ds \tag{8.1}$$

Here p denotes the surface pressure values for the present design, and p_{target} denotes the target surface pressure values, those of the Tenth Standard geometry. Hicks-Henne bump functions were used to parameterize the blade shape. A set of 24 Hicks-Henne bump functions are equally spaced along the axial chord direction for both the pressure and suction sides, and are defined as

$$b_i(x) = a_i \left[\sin\left(\pi x^{\frac{\log 0.5}{\log t_{1_i}}}\right)\right]^4 \quad \text{for } i = 1, 2, ..., 24 \tag{8.2}$$

where

$$t_{1_i} = \frac{i}{25} \quad \text{and} \quad t_2 = 4.0$$

The design variables were defined as the amplitudes of the Hicks-Henne bump functions, a_i.

The flow solution was developed using the in house *cascade* solver presented in [91] and brute force automatic differentiation was performed to provide the sensitivities. Using the flow solver and its reverse mode automatic differentiation, the sensitivities of

the objective function with respect to each design variable were calculated using the traditional and the projected adjoint sensitivity methods for the initial geometry. The projected sensitivity method projects the adjoint sensitivities using the reduced-order model convergence acceleration technique and then halts the sensitivity iteration. The traditional adjoint method converged to machine accuracy after 1,400 reverse iterations as shown in Fig. 60. The convergence rate of the discrete adjoint solver was again found to match that of the forward flow solver. Note the cascade flow solution is converging more rapidly due to the implementation of a two level multigrid scheme. After the fully converged sensitivity values were calculated the projection method was employed for the first design cycle. The projection method was observed to only require 146 reverse iterations to project sensitivities within 1% of the converged value. The projection was performed using ten equally spaced basis vectors stored between reverse iterations 2 and 146 as shown in Fig. 60. This resulted in a reduction of over 1,000 reverse iterations per design cycle.

The projected sensitivities were employed in an L-BFGS optimization procedure to perform iterative design optimization until the desired surface pressured distribution was recovered. The pressure distributions for the optimized, initial, and target shapes are presented in Fig. 61a. The corresponding blade profiles are included in Fig. 61b. The projected sensitivity method succeeds in recovering the target pressure profile and shape. A comparison between the design convergence using the traditional adjoint and projected adjoint method is included in Fig. 62. The traditional adjoint sensitivity method reaches an optimization limit around 60,000 seconds, while the adjoint projection method reaches the same optimization level after only 11,000 seconds, thereby reducing the computational time by 80%. The optimization limit likely results from the Hicks-Henne bump functions forming an incomplete design space. However, the target design was still recovered.

The projected adjoint sensitivity values were again evaluated by comparing the projected sensitivities and the sensitivities of the most converged basis vector with the fully converged sensitivity values. The percent difference of the projected and final basis vector sensitivity values from the fully converged sensitivity values for each design variable at the initial design point is shown in Fig. 63. From Fig. 63 it can be seen that the profile for the final basis vectors and the projected values have similar profiles. This isn't surprising since as the accuracy of the basis vectors improve the accuracy of the projected values is expected to improve as a result. It can also be seen that the projected adjoint sensitivities are considerably closer to the converged values. In fact, the percent difference of the projected sensitivity values from the converged sensitivity value are on average more than an order of magnitude closer.

Additionally, a comparison of the percent difference of the projected and the final basis vector sensitivity values from the fully converged sensitivity values is shown for the twelfth design variable across all the design cycles in Fig. 64. The twelfth design variable was selected because it appears closest to the point of the maximum difference between the initial and target shapes. For this design variable it should be noted that the projected sensitivity values all fall within the specified 1% tolerance, even though some of final basis vector values are significantly less accurate. In particular, examine design cycle 20 for the twelfth design variable. The final basis vector sensitivity value for is off by 50% percent from

the fully converged value. Applying the projection method yields a value that is within 1% of the converged value. This improvement highlights the considerable value of the adjoint sensitivity projection method. Through a computational cost equivalent to roughly one reverse iteration the adjoint sensitivity values were remarkably improved.

Finally, a comparison of the average percent difference of the projected and final basis vector sensitivities from the fully converged value over the course of all 25 design cycles is presented for all the design variables in Fig. 65. The average percent difference of all the final basis vector sensitivity values from the fully converged values was found to be 0.85%. In comparison the average percent difference over all adjoint projection sensitivity values from the fully converged values was 0.13%. On average this is nearly equivalent to an additional digit of accuracy for each sensitivity value. This added accuracy coupled with the suppression of highly inaccurate sensitivity values enables design optimization with highly accurate sensitivities for a fraction of the cost of an optimization technique utilizing fully converged values.

8.4 Acceleration of a Nested Optimization Scheme [3]

The fourth case study again considers application of the reduced-order model acceleration technique to accelerate a nested design optimization loop. However, this time a fixed-point iteration approach is used to solve for the adjoint sensitivities. The case originally presented in [110] demonstrates a methodology that can be followed to accelerate the adjoint sensitivity solution through repeated application of the reduced-order model-based acceleration technique.

8.4.1 Inverse Design of NREL S809 Airfoil in Inviscid Flow Field

To evaluate the reduced-order model convergence acceleration techniques, flow and sensitivity solutions for a well-known horizontal axis wind turbine blade profile were considered. The analysis of the flow and sensitivities was performed for both inviscid and viscous flow fields. Furthermore, aerodynamic design optimization was also considered in the form of a simple inverse design problem where the surface pressure distribution of a known airfoil is prescribed as the target pressure profile. The design is then started from another geometry and optimization is performed to recover the target surface pressure profile and its corresponding airfoil geometry. In this work, the inverse design problem seeks to recover the NREL S809 airfoil from an RAE 2822 airfoil. As mentioned before, the NREL S809 is a well-studied airfoil for horizontal-axis wind turbines [162, 176]. It is a 21% thick laminar flow airfoil that has been used within NREL Phase II, Phase III, and Phase IV wind turbines [56].

[3] This section, in part, is a reprint of the material as it appears in Aerospace Sciences 93 titled *"Reduced-order model-based convergence acceleration of reverse mode discrete adjoint solvers"* (2019). Authors: **Andrew Kaminsky** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Elsevier's copyright policies permit the material in the paper to be included, in full or in part, within the author's thesis or dissertation for non-commercial purposes.

Flow Solution

The NREL S809 airfoil was designed for a maximum, but restrained lift coefficient that is insensitive to roughness with a low profile drag. The airfoil is typically subjected to low Mach number flows, and presently two-dimensional inviscid flow at a Mach number of 0.3 and 0.0 degree angle of attack is considered. This case corresponds to the flow near the end of a horizontal-axis wind turbine (HAWT) blade where the relative air speed is high, and the blade twist leads to a lower angle of attack. The computational grids with an O-type topology for both the NREL S809 airfoil (target design) and the RAE2822 airfoil (initial design) are presented in Figs. 66a and 66b, respectively. The O-grids were generated using conformal mapping [195], and the far-field boundary is placed 100 chord lengths away.

To start the analyses, the nominal flow solver is verified first by comparing the surface pressure profile for the NREL S809 with the surface pressure distributions of Somers's panel method and experimental wind-tunnel tests [176], in Fig. 67. The NREL S809 surface pressure distribution of the present solver matches that of Somers's panel method relatively well, and the slight differences are likely due to the fact that the present solver is of higher-fidelity. This is substantiated by the fact that the surface pressure distribution of the present solver lies closer to the experimental data where there is disparity. In Fig. 67, the surface pressure distribution of the RAE 2822 is also included to highlight the difference between the initial and target pressure distributions. It is clear that the pressure profile of the RAE 2822 airfoil is dramatically different from that of the NREL S809 and optimization will require significant changes to the airfoil shape to recover the target surface pressure.

Sensitivity Solution

To evaluate the performance and robustness of the sensitivity acceleration technique an inverse design problem is set up by defining the surface pressure distribution of the NREL S809 airfoil as an optimal, target pressure profile. The objective function, which will be used to drive the initial airfoil shape to that of the target shape, is based on the difference between the current pressure distribution and the target pressure distribution over the entire airfoil, i.e.,

$$J = \frac{1}{2}\int_S (p - p_{\text{target}})^2 \, ds \,, \tag{8.3}$$

to quantify the quality of the current design. By minimizing the cost function, the desired NREL S809 airfoil should be recovered. The original airfoil shape is parameterized using fourth order b-splines

$$C(t) = \sum_{i=0}^{n} \boldsymbol{\beta} \boldsymbol{B}_{i,p}(t) \,, \quad n = 15 \,, \tag{8.4}$$

where $\boldsymbol{\beta}$ are the control points and $\boldsymbol{B}$ are the basis functions. For the cases considered in this work, a total of 30 surface control points (15 points on each of the suction and pressure surfaces) are used as design variables. The b-spline basis functions can be derived by means

of the Cox-de Boor recursion formula [120] given as:

$$\boldsymbol{B}_{i,0} = \begin{cases} 1 & \text{if} \quad t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{8.5}$$

$$\boldsymbol{B}_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i}\boldsymbol{B}_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}}\boldsymbol{B}_{i+1,k-1}(x) \quad k = 1, ..., 3\,. \tag{8.6}$$

The adjoint solver is used to calculate the sensitivity of the cost function to the design variables $\frac{\partial J}{\partial \boldsymbol{\beta}}$, which can later be utilized in a gradient-based optimization procedure to obtain the target design. The sensitivity solutions are calculated using the fixed-point iterative method developed via the primal time-stepping adjoint technique. The fixed-point method can be verified by comparing the sensitivity values of the primal time-stepping approach with the brute-force adjoint approach, as shown in Fig. 68. It is clearly seen that the sensitivities obtained through the brute-force adjoint approach and the newly developed primal time-stepping approach match to machine accuracy.

The sensitivities convey the relationship between the cost function and each design variable, and the magnitude and sign represent the step size and the direction that the b-spline control points should be moved. The first 15 design variables control the top surface of the airfoil, and the next 15 control the bottom surface. The control points are ordered from leading to trailing edge, which explains the large jump discontinuity between design variable 15 and 16. From the sensitivity values, it can be seen that top surface should be expanded upwards near the mid-chord where the sensitivities are negative and narrowed near the leading and trailing edges where the sensitivities are positive. Conversely, the bottom surface should be expanded downwards where sensitivities are positive and narrowed where the sensitivities are negative. In combination, this leads to an updated airfoil shape which expands the RAE 2822 towards the NREL S809 by increasing the curvature of the top surface and forming the belly of the airfoil on the bottom surface.

A comparison of the convergence histories of the primal time-stepping and brute force adjoint approaches is also provided in Fig. 69 to offer further insight into the relationship between the brute force and primal time-stepping adjoint approaches. As can be seen, the convergence histories of the two approaches are identical. The convergence history of the sensitivity solvers is smooth, and the convergence rate of the sensitivity solvers match that of the flow solver, as expected. Although not shown here, the sensitivity values actually match at every iteration [27]. It must be emphasized that even though the convergence histories for two approached are identical, the sensitivity calculation process is not. Additionally, the adjoint sensitivities were converged until the residual is less than 1×10^{-15}. Converging the adjoint sensitivities to this residual criterion may not be necessary during the entire process, but as the solution converges increasingly accurate derivative information is required by the L-BFGS optimizer [209]. Convergence has been enforced to a smaller residual during the whole design optimization process, to demonstrate the acceleration technique is robust and can be applied consistently to obtain a numerically identical solution.

Reduced-order Model Convergence Acceleration for Sensitivity Analysis

The goal now is to reduce the cost associated with calculating adjoint sensitivities. Initially, the sensitivities of the first design stage are used to evaluate the performance of the ROM convergence acceleration techniques. Comparisons of the convergence histories of the baseline primal time-stepping approach and the ROM accelerated primal time-stepping approach are presented in Figs. 70a and 70b. The effects of the number of snapshots as well as the iteration at which the first snapshot is collected are investigated in the figure. Furthermore, the snapshot collection parameters for the ROMs are included in Table 16 for varied snapshot counts, and in Table 17 for varied initial snapshot locations. The convergence history of the PTS adjoint with covariance acceleration features regions of smooth convergence where snapshots are collected while the fixed-point iterative method is performed in the nominal manner. Once the snapshots are collected, the reduced-order model is formed and used to project the converged solution. The solution projections occur at the sharp drops found in the convergence histories. These drops result from restarting the fixed-point solver from the projected solutions which have smaller associated residual values due to their improved accuracy. By repeating this projection process, considerable cost reductions can be obtained.

Investigating Figs. 70a and 70b, some insight into proper snapshot selection and some general rules of thumb are: (1) delaying the initial snapshot leads to an improved initial projection until a plateau is reached, but repeated application leads to comparable convergence histories; (2) increasing the span between snapshots typically improves performance; and (3) additional snapshots lead to better projections but more costly models. The improved performance resulting from additional basis vectors is caused by the expansion of the solution space. Considerable attention was given to proper snapshot selection in the authors' previous work [108], and further information is available to the interested reader.

Gradient-based Optimization with ROM Accelerated Sensitivities

To emulate snapshot selection in a typical case, a set of snapshot selection parameters were selected in a uniform manner. A reduced-order model is built using 101 snapshots that were initially collected at iteration 500 and then sampled every 5 iterations. Following a projection, the snapshot collection was delayed 500 iterations and then samples are collected every 5 iterations until 101 snapshots are obtained. These parameters can be optimized, but in general the method is robust so long as a sufficient number of snapshots are collected. A comparison of the convergence histories of the primal time-stepping approach with and without the ROM convergence acceleration is presented in Fig. 71 for the sensitivities at the initial design. The convergence history of the PTS adjoint with the acceleration techniques feature sharp drops where the reduced-order model-based projections are made to predict the converged flow solution. In contrast, the POD acceleration technique features jumps followed by sudden drops in the convergence history. The jumps correspond to the residuals calculated for the orthogonal basis vectors, and as such, they are considered "synthetic," and are of no concern here. It also should be noted that both methods can project solutions to machine accuracy.

The POD projection method utilized the full POD basis set, and a comparison of the eigenvalues for the first three projections is included in Fig. 72 for this case study. From the figure it can be seen that as the solution converges, the early POD modes have increasingly larger eigenvalues.

The computational cost associated with each of the sensitivity calculation techniques is included in Table 18. The primal time-stepping adjoint is faster than the brute force approach. This is because the brute force adjoint approach propagates the sensitivity of the flow solver to the grid every iteration, while the primal time-stepping adjoint method determines the sensitivity of the objective function to the flow solution, and then propagates the sensitivity to the grid only once the flow sensitivities are converged. In the context of design optimization, the cost of the primal time-stepping adjoint can be further reduced, by "hot-starting" discrete adjoint computations from the converged solution of the prior design cycle. Since the primal time-stepping approach is based on a fixed-point iterative method, it should be possible to start the computations from any value and recover the correct solution. Conversely, the brute force adjoint approach is merely a systematic execution of the chain rule. Any perturbation or deviation from the exact values in the intermediate accumulation steps will ultimately lead to incorrect sensitivity values [107, 27].

From this study, it is clear that both acceleration techniques offer considerable cost reductions. The POD acceleration technique offers an 80% cost reduction whereas the covariance acceleration technique offers an 83% cost reduction for this particular test case.

Following application to the first design step, the sensitivity convergence acceleration technique was next employed within a nested design optimization loop to demonstrate the robustness of the technique. Unconstrained optimization was performed using the L-BFGS gradient-based optimizer [204]. A comparison the cost function over 25 design cycles is shown in Fig. 73. From the figure it can be seen that the objective function values for each incremental design are nearly identical for both methods. In fact, the final cost function values only differ by 5.20×10^{-10}. This instills confidence in the method since the sensitivity values are identical with and without the use of the acceleration technique, and the acceleration results entirely from the reduced cost of the sensitivity calculation.

The optimization with the sensitivity convergence acceleration was able to recover the target surface as shown in Fig 74a. Correspondingly the optimized airfoil profile matches that of the NREL S809 as shown in Fig. 74b. This case demonstrates that the sensitivity acceleration techniques provide sensitivities of comparable accuracy at a fraction of the computational cost. The optimization with the covariance convergence acceleration technique reduced the computational time by 80% relative to the optimization based on the nominal primal time-stepping adjoint sensitivities without acceleration.

8.4.2 Inverse Design of NREL S809 in a Viscous Flow Field

Flow Solution

The same inverse design problem was also considered with a viscous flow field. A Reynolds number of 2×10^6 is considered to match Somers's experimental flow condition [176]. For this

case, O-type grids [as shown in Figs. 75a and 75b] are generated with 257×129 nodes in the circumferential and radial directions respectively, and the far-field boundary is placed 100 chords away from the airfoils. The viscous flow solution accuracy is evaluated by comparing the surface pressure profile of the NREL S809 with the CFD-ACE solver [203] as well as the available experimental wind-tunnel measurements [176], as shown in Fig. 76. Both solvers capture the surface pressure distribution well, though the present solver matches the experimental data slightly better particularly near the mid-chord. A comparison of the loading coefficients is also included in Table 19. The lift, drag, and moment coefficients are quite similar to those reported by Wolfe and Ochs [203]. Both solvers underestimate the lift coefficient and overestimate the drag and moment coefficients.

Sensitivity solution

The sensitivity solution is again considered for the initial design cycle. The sensitivity of the cost function to the design variables was calculated using the brute force, PTS adjoint, and ROM accelerated PTS adjoint approaches. A comparison of the sensitivity solutions is included in Fig. 77, and all three sensitivity solutions match quite well. The sensitivities of the viscous case are qualitatively similar to the inviscid case, but the local extrema appear to be more pronounced relative to the inviscid case.

The reduced-order model was built using the snapshot collection parameters in the Table 20. A higher snapshot quantity was used due to the added complexity of the flow behavior resulting from including the viscous effects and the additional degrees of freedom associated with the turbulence model. The collection of the snapshots was also delayed due to the flow field taking longer to develop past the initial transient start-up behavior. A comparison of the convergence rate using the primal time-stepping approach with and without the ROM based acceleration is included in Fig. 78. It is apparent that the proposed approach is able to accelerate the convergence rate of the discrete adjoint solver significantly. For this case, it is interesting to note that the first projection actually slightly increases the global residual rather than leading to a global residual drop. This indicates that snapshots were likely gathered during the initial development of the sensitivity field and performance could be improved by lagging snapshot collection further. However, despite the poor initial projection the subsequent applications of the reduced-order model convergence acceleration technique led to a marked acceleration, thus demonstrating the robustness of the technique to recover following a poor projection.

Gradient-based optimization with ROM accelerated sensitivities

Again, the design optimization was started from an RAE 2822 airfoil. From Fig. 76 it is clear that the surface pressure distributions are considerably different, and the design optimization process must be able to handle large design variations. To assist with robustness a bounded optimization was performed using the L-BFGS-B solver [208]. The design variables were bounded so that the suction and pressure sides of the airfoil were never allowed to intersect.

The design optimization was performed for 25 major L-BFGS-B design cycles, using sensitivities found from the baseline PTS adjoint and the ROM accelerated PTS adjoint approach. A comparison of the cost function over time for the two approaches is included in Fig. 79 and Table 21 indicating computational savings of around 57%. Again, the cost function values at each design cycle are nearly identical, and the optimizer is quite successful in consistently decreasing the cost function with the sensitivities provided. The cost function can be decreased even further, but the surface pressure distribution and airfoil profile are adequately recovered as shown in Fig. 80a and Fig. 80b, respectively.

8.5 Acceleration of a One-shot Optimization Scheme

In the previous case study, the optimization process was accelerated by using the Reduced-order model-based acceleration technique to accelerate the adjoint solver. In the next study, the ability of the Reduced-order model-based acceleration to generalize to different types of fixed-point iterators will be demonstrated by applying it to accelerate a one-shot optimizer simultaneously solving the primal, adjoint, and design equations.

8.5.1 Inverse Design of Converging-Diverging Nozzle

The ROM-based acceleration of a one-shot was demonstrated for a nozzle inverse design problem. An initial nozzle shape was defined as

$$h(x) = \begin{cases} 0.5\,(a_1 - 1.0)\,(1 + \cos\left(\frac{\pi x}{0.35}\right) & 0 \leq x \leq 0.35 \\ 0.5\,(a_2 - 1.0)\,(1 + \cos\left(\frac{\pi(x-0.35)}{0.65}\right) & 0.35Vx \leq 1\,, \end{cases} \tag{8.7}$$

where $a_1 = 1.5$ and $a_2 = 2.5$. The nozzle cross-section is included in Fig.81a. The flow through the nozzle will be modeled using the quasi-1D Euler equations detailed in Section 2.2.1. The nozzle boundary conditions were defined as an inlet pressure of 1.0, a total inlet temperature of 1.0, and an exit pressure of 0.7. These boundary conditions lead to a nozzle flowfield featuring a normal shock. The resulting conservation variable and pressure distribution is included in Fig.81b.

The optimization problem was defined as an inverse design problem, in which the objective was to recover a target pressure profile. The objective metric was defined as the root mean square of the difference between the current pressure and the target pressure

$$I = \frac{1}{2}\int_0^1 (p - p_{\text{target}})^2\, dx\,. \tag{8.8}$$

The optimizer was applied to modify the nozzle shape to recover the target profile. The target pressure profile corresponds to a perturbed nozzle cross-section generated by adding fourth-order b-splines to the baseline nozzle cross-section

$$h_{target}(x) = h(x) + \sum_{i=0}^{n} \boldsymbol{\beta} \boldsymbol{B}_{i,p}(x)\,, \quad n = 9\,, \tag{8.9}$$

where $\boldsymbol{\beta}$ are the 9 control points and $\boldsymbol{B}$ their basis functions. The b-spline basis functions can be derived by means of the Cox-de Boor recursion formula [120] given as:

$$\boldsymbol{B}_{i,0} = \begin{cases} 1 & \text{if} \quad t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{8.10}$$

$$\boldsymbol{B}_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i} \boldsymbol{B}_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} \boldsymbol{B}_{i+1,k-1}(x) \quad k = 1, ..., 3\,. \tag{8.11}$$

The perturbed nozzle shape is compared to the baseline nozzle in Fig. 82a, and the pressure profiles are compared in Fig. 82b.

A set of nested and one-shot optimization approaches were applied to the nozzle inverse design problem. Optimization with the nested and traditional one-shot optimizer was performed to establish a baseline that could be used to assess the value provided by the ROM-based acceleration technique.

L-BFGS-B Optimization Baseline

To demonstrate the accumulated computational cost savings, the initial optimization study utilized a traditional nested optimization approach. The L-BFGS-B optimizer was used to perform design updates based on fully-converged primal and adjoint solutions. The metric for reaching full convergence of the primal and adjoint was defined as completion of 20,000 fixed-point iterations for each solution. This metric was selected by observing the convergence histories of the initial design state, included in Fig. 83. From the figure it can be seen that the primal solution and the fixed-point, primal time-stepping, adjoint solution are fully converged by iteration 20,000, and some leeway is provided for designs that might converge more slowly.

Following primal and adjoint solution convergence the design was updated using the L-BFGS-B algorithm from [156]. The L-BFGS-B algorithm provided the step direction and the step size δ was defined as 0.01 to maintain design solution stability through under-relaxation. The resulting design update was

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \delta \left[H^{(n)}) \right]^{-1} \nabla f(\boldsymbol{x}^{(n)})\,. \tag{8.12}$$

In total two hundred design cycles were completed, and the corresponding objective function history is presented in Fig. 84. From the figure it can be seen that the optimizer reduces the objective metric from $\mathcal{O}(10^0)$ to $\mathcal{O}(10^{-3})$. The gradient-based optimizer makes fairly steady progress but appears to overstep the minimum three times during the optimization. The recovered pressure profile and nozzle shape are compared with the target and initial values in Fig. 85. The recovered nozzle does not perfectly match that of the target

profile. However as expected from the objective metric value, the recovered pressure profile matches quite well. Thus, the optimization was successful. The mismatch in the nozzle shape likely indicates that the inverse design problem was ill-posed and there exists multiple nozzle profiles that recover the desired pressure profile. This is not particularly surprising, because inverse design problems are notoriously challenging to define in a manner that avoids multiple solutions.

For the purposes of evaluating the optimizer and various acceleration techniques recovery of the target pressure profile is sufficient. The results of this nested optimization study provide a useful benchmark, which will be used to assess the one-shot approaches in the following.

One-shot Optimization Baseline

Next the nozzle inverse design problem was considered with a one-shot optimizer. A single iteration of the one-shot optimizer employed here sequentially executed updates of the primal solution, followed by the adjoint solution, and finally the design solution in a Gauss-Seidel manner. In this manner, the solutions to each of the equations is converged simultaneously as shown in Fig. 86.

The primal solution was advanced with a fifth-order Runge–Kutta scheme. The primal time-stepping adjoint developed by automatically differentiating the primal solver was used to perform fixed-point iterative updates of the adjoint solution. Finally, the steepest descent algorithm was used to update the design. To maintain solution stability a step size of $\delta = 0.000001$ was selected so that the design update was:

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \delta \nabla f(\boldsymbol{x}^{(n)}) \ . \tag{8.13}$$

In total five million one-shot iteration cycles were completed, and the corresponding objective function history is presented in Fig. 87. From the figure it can be seen that the one-shot optimizer monotonically decreases the cost function. Much of the progress is made early (e.g., between iteration 1 and 500,000). Beyond this point the cost function is decreased at a much slower rate. The slow convergence is likely attributable to vanishing gradients and the small step size. As the design nears the local minimum the gradients approach zero and the perturbation to the design becomes quite small stalling convergence. An adaptive step size could be implemented to encourage continued reduction of the cost function. However, the recovered pressure profile matches the target sufficiently well enough at the elbow that the solution could have been halted prior to the convergence stagnation.

Examining the cost function versus time in Fig. 87b, it can be seen that early the one-shot optimizer outperforms the traditional nested optimization approach. However, the solution stagnates, and the nested optimization approach is able to overtake it after 2500 seconds. The pressure profile and nozzle shape recovered by the one-shot optimizer are quite similar to those identified by the nested optimizer, and a comparison with the target and initial values is included in Fig. 88. Again the recovered nozzle profile is not a perfect match for the original perturbed nozzle, but the pressure profile is recovered very well.

The optimizer successfully minimizes the objective metric and recovers the desired pressure profile. These results, illustrate the ability of the one-shot optimizer to identify good (if not truly optimal) designs significantly faster than the nested optimization approach. The results of this one-shot optimization study provide a useful benchmark, which will be used to assess the ROM-accelerated one-shot approach detailed next.

ROM-Accelerated One-shot Optimization

Finally, the nozzle inverse design problem was considered with a one-shot optimizer accelerated with the ROM-based acceleration technique employed to project the primal, adjoint, and design solution to more converged solutions. The one-shot optimizer is identical to that of the previous case, except for the addition of the ROM-acceleration technique. To demonstrate proof of concept, initially only a single application was considered with the snapshot collection parameters in Table 22. Each snapshot contained the full primal, adjoint, and design state vectors concatenated into a single column vector. During the solution the snapshots and their residuals were stored for use in the ROM-based acceleration procedure. The snapshot basis vector technique detailed in Section 7.1.1 was considered. To demonstrate proof-of-concept, the ROM-based acceleration was initially only applied once. A comparison of the convergence rate using the one-shot optimizer with and without the ROM-based acceleration is included in Fig. 89

From the figure it is apparent that the proposed ROM-based acceleration approach is able to accelerate the primal, adjoint and design solution. Observing the region immediately following the projection (at iteration 90,000) it can be seen that the residual of the primal and adjoint solution initially jump up and the design solution residual drops an order of magnitude. This signifies the projected flow and adjoint solution states are not perfectly tuned to the new design. However, following iterations quickly damp out the initial high-frequency behavior in the solution and reach residual values much lower than the unaccelerated solution. Additionally, it can be seen that due to the solution stagnation the unaccelerated solution is able to catch back up to the reduced-order model accelerated solution if further projections are not performed.

The impact of the ROM-based acceleration on the design variables can be seen in the Fig. 90 which presents the variable evolution over the iterative solution. The design variable history shows reduced-order model-based acceleration technique projection causes discrete jumps in the design variable value. Interestingly, it can also be observed that the values jumped to compare well to the values reached asymptotically by the unaccelerated one-shot solver.

Following this single application case, repeated application of the convergence acceleration technique was performed using the snapshot collection parameters included in Table 23. The convergence history of one-shot optimizer with repeated application of the ROM acceleration technique is presented in Fig. 91. It is clear the primal, adjoint, and design solution all converge more rapidly by employing the ROM-acceleration technique. However

perhaps even more importantly Fig. 92 the value of the cost function is also reduced more rapidly using the ROM-acceleration technique.

Examining the figure it can be seen that cost function of the ROM-accelerated one-shot optimizer no longer stagnates. Following the initial project, the ROM-accelerated one-shot optimizer consistently outperforms the baseline one-shot and nested optimizers. A design with a cost function value less than that of the best design identified by the nested optimizer is found by the ROM-accelerated one-shot optimizer in 45% less time. This result illustrates the value of the proposed ROM-acceleration technique applied to one-shot optimizers.

The pressure profile and nozzle shape recovered by the one-shot optimizer are quite similar to those identified by the nested optimizer, and a comparison with the target and initial values is included in Fig. 93. The recovered nozzle profile and the pressure profile are quite similar to those found by the other optimizers. The lone caveat is the pressure profile recovered by the ROM-accelerated one-shot is an order-of magnitude closer to the target pressure. This is a result of the optimized design variables more closely matching those of the target design variables as shown in Fig. 94.

Chapter 9

Conclusions and Recommendations

This dissertation was motivated by the desire to accelerate numerical optimization of aerodynamic designs. Greater adoption of numerical optimization with high-fidelity analysis tools like CFD, will markedly improve the aerodynamic design process. However, greater adoption of mathematically formal simulation-based optimization approaches within aerodynamic design depends upon a delicate balance of reducing computational costs while maintaining accessibility. Techniques are needed to accelerate the simulation-based optimization process, while remaining approachable enough to encourage adoption. The contribution of this dissertation is associated with development and demonstration of a reduced-order model-based method for accelerating the primal (flow), adjoint (sensitivity), and design solutions independently and in combination.

9.1 Summary

A novel reduced-order model-based acceleration approach was developed. The approach builds the reduced-order model online (during traditional iteration) by collecting snapshots of the solution behavior. A linearity assumption is made, and the full-order system is approximated by a linear combination of basis vectors developed using the solution snapshots. The reduced-order model is then solved and used to project to an improved solution state.

Generalization of the technique was demonstrated by applying the reduced-order model to accelerate:

1. Primal harmonic balance solvers
2. Continuous adjoint solvers
3. Discrete adjoint solvers
4. One-shot design solvers

The variations of the ROM acceleration technique available for each application were discussed. Snapshot and covariance-based basis vectors were shown to work best for

primal solvers. For the adjoint solvers, proper orthogonal decomposition basis vectors were shown to work best. Additionally, an approach for implementing the accelerated adjoint strategy through reverse-mode automatic differentiated was detailed to facilitate the ease of implementation and foster greater adoption. Finally, the ROM acceleration technique was applied to accelerate a one-shot optimization approach using a snapshot basis.

Best practices for snapshot collection procedures were outlined. Delaying initial snapshot collection, increasing iterations between snapshots, and adding snapshots were found to form models with improved projections. While optimization of these parameters leads to improved performance, the technique has been shown to be robust so long as a sufficient number of snapshots is utilized. Additionally repeated application of the technique was consistently proven to significantly reduce the computational cost.

The convergence acceleration technique was evaluated using the initial design cycles of several inverse design problems. In these applications it was shown to consistently reduce the computational cost of the primal and adjoint solution. Robustness was established through utilization in a full design optimization, in which no fine tuning of the snapshot collection parameters was performed (e.g., the same snapshot parameters were used during the sensitivity calculation of each design cycle). The ROM acceleration technique was demonstrated to accelerate the nested optimization processes by reducing the primal and adjoint solutions. The ROM acceleration technique was also demonstrated to accelerate simultaneously converging primal, adjoint, and design solutions in a one-shot optimization approach.

The technique can be easily implemented into fixed-point iterative solvers, typically the only change needed to existing solvers is the storage of the pre-convergent fixed-point snapshot solutions and residuals. The ease of implementation and considerable computational cost reductions of the proposed ROM technique highlight its value.

9.2 Future Work

The present work demonstrated the ability of the reduced-order model-based acceleration technique accelerate steady and periodically unsteady flow solvers, continuous and discrete adjoint sensitivity solvers, and a one-shot design optimization solver. Several numerical case studies were performed to demonstrate proof-of-concept. However, future work is needed to increase the impact on numerical optimization and aerodynamic design and foster greater adoption. Recommendations for future work towards these objectives include:

1. Extension of the technique to more complex geometries and configurations specifically concentrating on moving towards 3D geometries of aircraft.

2. Reducing the memory requirements through better algorithms for formulating the basis or reduced representation.

3. Consideration of techniques for use of the method within parallelized solvers featuring domain decomposition.

4. Continued investigation of snapshot collection procedures and formalization of best practices.

5. Application to one-shot optimizers with a quasi-Newton update.

Bibliography

[1] Akbarzadeh, S., Wang, Y., and Mueller, J. D. (2015). Fixed point discrete adjoint of simple-like solvers. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2750. 47

[2] Albring, T. A., Sagebaum, M., and Gauger, N. R. (2016). Efficient aerodynamic design using the discrete adjoint method in su2. In *17th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 3518. 1

[3] Anderson, W. K., Newman, J. C., Whitfield, D. L., and Nielsen, E. J. (2001). Sensitivity analysis for navier-stokes equations on unstructured meshes using complex variables. *AIAA journal*, 39(1):56–63. 5

[4] Anderson, W. K. and Venkatakrishnan, V. (1999). Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4-5):443–480. 37

[5] Antoulas, A. C. (2005). *Approximation of large-scale dynamical systems*, volume 6. Siam. 9

[6] Bai, Z. (2002). Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Applied numerical mathematics*, 43(1-2):9–44. 10

[7] Baur, U., Benner, P., and Feng, L. (2014). Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Archives of Computational Methods in Engineering*, 21(4):331–358. 9, 64

[8] Baysal, O. and Eleshaky, M. E. (1992). Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA journal*, 30(3):718–725. 6

[9] Baysal, O., Eleshaky, M. E., and Burgreen, G. W. (1993). Aerodynamic shape optimization using sensitivity analysis on third-order euler equations. *Journal of Aircraft*, 30(6):953–961. 6

[10] Beran, P., Stanford, B., and Kurdi, M. (2010). Sensitivity analysis for optimization of dynamic systems with reduced order modeling. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1503. 9, 46

[11] Beux, F. and Dervieux, A. (1994). A hierarchical approach for shape optimization. *Engineering Computations*, 11(1):25–48. 5

[12] Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P. (1992). Adifor–generating derivative codes from fortran programs. *Scientific Programming*, 1(1):11–29. 8

[13] Bischof, C., Corliss, C., Green, L., Griewank, A., Haigler, K., and Newman, P. (2003). Automatic differentiation of advanced cfd codes for multidisciplinary design. 8

[14] Bischof, C. H. and Bücker, H. M. (2000). Computing derivatives of computer programs. Technical report, Argonne National Lab., IL (US). 5

[15] Bischof, C. H., Bücker, H. M., and Rasch, A. (2004). Sensitivity analysis of turbulence models using automatic differentiation. *SIAM Journal on Scientific Computing*, 26(2):510–522. 6

[16] Blazek, J. (2015). *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann. xi, 16, 17, 24, 118, 119

[17] Bosse, T., Gauger, N. R., Griewank, A., Günther, S., and Schulz, V. (2014a). One-shot approaches to design optimzation. *Trends in PDE Constrained Optimization*, pages 43–66. 61, 63

[18] Bosse, T., Lehmann, L., and Griewank, A. (2014b). Adaptive sequencing of primal, dual, and design steps in simulation based optimization. *Computational Optimization and Applications*, 57(3):731–760. 2, 62

[19] Brandt, A. (1982). Guide to multigrid development. In *Multigrid methods*, pages 220–312. Springer. 33

[20] Bui-Thanh, T., Damodaran, M., and Willcox, K. E. (2004). Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal*, 42(8):1505–1516. 10

[21] Bui-Thanh, T., Willcox, K., and Ghattas, O. (2008). Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288. 77

[22] Cabay, S. and Jackson, L. (1976). A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13(5):734–752. 10

[23] Cardoso, M. A. (2009). *Development and application of reduced-order modeling procedures for reservoir simulation*. Stanford University. 10

[24] Carle, A., CH, B., and PA, N. (1994). Applications of automatic differentiation in cfd. 8

[25] Choi, S., Lee, K., Potsdam, M. M., and Alonso, J. J. (2014). Helicopter rotor design using a time-spectral and adjoint-based method. *Journal of Aircraft*, 51(2):412–423. 9

[26] Christakopoulos, F. (2012). *Sensitivity computation and shape optimisation in aerodynamics using the adjoint methodology and Automatic Differentiation*. PhD thesis, Queen Mary University of London. 3

[27] Christakopoulos, F., Jones, D., and Müller, J.-D. (2011). Pseudo-timestepping and verification for automatic differentiation derived cfd codes. *Computers & Fluids*, 46(1):174–179. xi, 1, 3, 8, 9, 46, 47, 48, 85, 87, 127

[28] Cinnella, P., Cappiello, E., De Palma, P., Napolitano, M., and Pascazio, G. (2004). A numerical method for 3d turbomachinery aeroelasticity. In *ASME Turbo Expo 2004: Power for Land, Sea, and Air*, pages 539–550. American Society of Mechanical Engineers. 4

[29] Clemens, M., Wilke, M., Schuhmann, R., and Weiland, T. (2004). Subspace projection extrapolation scheme for transient field simulations. *IEEE transactions on magnetics*, 40(2):934–937. 10

[30] Cook, P., Firmin, M., and McDonald, M. (1977). *Aerofoil RAE 2822: pressure distributions, and boundary layer and wake measurements*. RAE. 52

[31] Courant, R., Frederick, K., and Lewy, H. (1928). On the partial difference equations of mathematical physics. *Mathematische Annalen*, 100:32–74. 26

[32] Cramer, E., Frank, P., Shubin, G., and Dennis, Jr., J. (1992). On alternative problem formulations for multidisciplinary design optimization. In *4th Symposium on Multidisciplinary Analysis and Optimization*, page 4752. 61

[33] Da Ronch, A., McCracken, A. J., Badcock, K. J., Widhalm, M., and Campobasso, M. (2013). Linear frequency domain and harmonic balance predictions of dynamic derivatives. *Journal of Aircraft*, 50(3):694–707. xii, 54, 136

[34] Davidon, W. C. (1991). Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17. 3

[35] Djeddi, R. and Ekici, K. (2019). Fdot: A fast, memory-efficient and automated approach for discrete adjoint sensitivity analysis using the operator overloading technique. *Aerospace Science and Technology*, 91:159–174. 43, 44

[36] Djeddi, R. and Ekici, K. (2021). Novel expression-template-based automatic differentiation of fortran codes for aerodynamic optimization. *AIAA Journal*, 59(1):88–103. 1

[37] Djeddi, R., Howison, J., and Ekici, K. (2016). A fully coupled turbulent low-speed preconditioner for harmonic balance applications. *Aerospace Science and Technology*, 53:22–37. 22, 72

[38] Djeddi, R., Kaminsky, A., and Ekici, K. (2017a). Convergence acceleration of fluid dynamics solvers using a reduced-order model. *AIAA Journal*, 55(9):3059–3071. 11, 22, 65, 66, 67, 78

[39] Djeddi, R., Kaminsky, A. L., and Ekici, K. (2017b). Convergence acceleration of fluid dynamics solvers using a reduced-order-model. AIAA Paper 2017-0726. 64, 65, 66

[40] Duta, M., Giles, M., and Campobasso, M. (2002). The harmonic adjoint approach to unsteady turbomachinery design. *International Journal for Numerical Methods in Fluids*, 40(3-4):323–332. 7

[41] Edwards, J. W. and Thomas, J. L. (1989). Computational methods for unsteady transonic flows. *Unsteady Transonic Aerodynamics*, 120:211–261. 3

[42] Ekici, K. and Hall, K. C. (2006). Fast estimation of unsteady flows in turbomachinery at multiple interblade phase angles. *AIAA journal*, 44(9):2136–2142. 10, 66, 67, 71

[43] Ekici, K. and Hall, K. C. (2007). Nonlinear analysis of unsteady flows in multistage turbomachines using harmonic balance. *AIAA journal*, 45(5):1047–1057. 4

[44] Ekici, K. and Hall, K. C. (2008). Nonlinear frequency-domain analysis of unsteady flows in turbomachinery with multiple excitation frequencies. *AIAA journal*, 46(8):1912–1920. 4

[45] Ekici, K. and Hall, K. C. (2011). Harmonic balance analysis of limit cycle oscillations in turbomachinery. *AIAA journal*, 49(7):1478–1487. 4

[46] Ekici, K., Hall, K. C., and Dowell, E. H. (2008). Computationally fast harmonic balance methods for unsteady aerodynamic predictions of helicopter rotors. *Journal of Computational Physics*, 227(12):6206–6225. 4

[47] Ekici, K., Hall, K. C., Huang, H., and Thomas, J. P. (2013). Stabilization of explicit flow solvers using a proper-orthogonal-decomposition technique. *AIAA Journal*, 51(5):1095–1104. 10

[48] Ekici, K., Hall, K. C., and Kielb, R. E. (2010). Harmonic balance analysis of blade row interactions in a transonic compressor. *Journal of Propulsion and Power*, 26(2):335–343. 4

[49] Ekici, K. and Huang, H. (2012). An assessment of frequency-domain and time-domain techniques for turbomachinery aeromechanics. In *30th AIAA Applied Aerodynamics Conference*, page 3126. 4

[50] Elliott, J. and Peraire, J. (1997). Practical three-dimensional aerodynamic design and optimization using unstructured meshes. *AIAA journal*, 35(9):1479–1485. 7

[51] Etkin, B. (2012). *Dynamics of atmospheric flight*. Courier Corporation. 55

[52] Fransson, T. and Verdon, J. (1993). Panel discussion on standard configurations for unsteady flow through vibrating axial-flow turbomachine-cascades. In *Unsteady aerodynamics, aeroacoustics, and aeroelasticity of turbomachines and propellers*, pages 859–889. Springer. 81

[53] Gauger, N., Griewank, A., Hamdi, A., Kratzenstein, C., Özkaya, E., and Slawig, T. (2012). Automated extension of fixed point pde solvers for optimal design with bounded retardation. In *Constrained Optimization and Optimal Control for Partial Differential Equations*, pages 99–122. Springer. 2, 63

[54] Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474. 8

[55] Giering, R., Kaminski, T., and Slawig, T. (2005). Generating efficient derivative code with taf: adjoint and tangent linear euler flow around an airfoil. *Future generation computer systems*, 21(8):1345–1355. 6

[56] Giguere, P. and Selig, M. S. (1999). Design of a tapered and twisted blade for the nrel combined experiment rotor. Technical report, National Renewable Energy Lab., Golden, CO (US). 83

[57] Giles, M., Duta, M., and Mueller, J.-D. (2001). Adjoint code developments using the exact discrete approach. In *15th AIAA Computational Fluid Dynamics Conference*, page 2596. 52, 53, 133

[58] Giles, M., Pierce, N., Giles, M., and Pierce, N. (1997). Adjoint equations in cfd-duality, boundary conditions and solution behaviour. In *13th Computational Fluid Dynamics Conference*, page 1850. 39

[59] Giles, M. B., Duta, M. C., M-uacute, J.-D., ller, and Pierce, N. A. (2003). Algorithm developments for discrete adjoint methods. *AIAA journal*, 41(2):198–205. 1, 7

[60] Giles, M. B. and Pierce, N. A. (1998). On the properties of solutions of the adjoint Euler equations. *Numerical methods for fluid dynamics VI. ICFD*, pages 1–16. 6, 51

[61] Giles, M. B. and Pierce, N. A. (2000). An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393–415. 1, 6, 7

[62] Giles, M. B. and Pierce, N. A. (2001). Analytic adjoint solutions for the quasi-one-dimensional euler equations. *Journal of Fluid Mechanics*, 426:327–345. xi, xii, 19, 37, 38, 50, 51, 78, 79, 130, 131

[63] Gopinath, A. and Jameson, A. (2006). Application of the time spectral method to periodic unsteady vortex shedding. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 449. 3, 4

[64] Gorrell, S. E., Car, D., Puterbaugh, S. L., Estevadeordal, J., and Okiishi, T. H. (2006). An investigation of wake-shock interactions in a transonic compressor with digital particle image velocimetry and time-accurate computational fluid dynamics. *Journal of turbomachinery*, 128(4):616–626. 3

[65] Gray, J. S., Hwang, J. T., Martins, J. R., Moore, K. T., and Naylor, B. A. (2019). Openmdao: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104. 1

[66] Green, L. L., Newman, P. A., and Haigler, K. J. (1996). Sensitivity derivatives for advanced cfd algorithm and viscous modeling parameters via automatic differentiation. *Journal of Computational Physics*, 125(2):313–324. 8

[67] Griewank, A. (2006). Projected hessians for preconditioning in one-step one-shot design optimization. In *Large-Scale Nonlinear Optimization*, pages 151–171. Springer. 62

[68] Griewank, A. and Faure, C. (2002). Reduced functions, gradients and hessians from fixed-point iterations for state equations. *Numerical Algorithms*, 30(2):113–139. 62

[69] Griewank, A. and Faure, C. (2003). Piggyback differentiation and optimization. In *Large-scale PDE-constrained optimization*, pages 148–164. Springer. 1, 2, 3, 62

[70] Griewank, A., Juedes, D., and Utke, J. (1996). Algorithm 755: Adol-c: a package for the automatic differentiation of algorithms written in c/c++. *ACM Transactions on Mathematical Software (TOMS)*, 22(2):131–167. 1, 8, 44

[71] Griewank, A. and Walther, A. (2000). Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45. 1, 9

[72] Guenther, S., Gauger, N. R., and Wang, Q. (2016). Simultaneous single-step one-shot optimization with unsteady pdes. *Journal of Computational and Applied Mathematics*, 294:12–22. 2, 62

[73] Haftka, R. T. (1985). Simultaneous analysis and design. *AIAA journal*, 23(7):1099–1103. 1, 61

[74] Hall, K. C., Thomas, J. P., and Clark, W. S. (2002). Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA journal*, 40(5):879–886. 4, 14

[75] Harlow, F. H. (2004). Fluid dynamics in group t-3 los alamos national laboratory:(la-ur-03-3852). *Journal of Computational Physics*, 195(2):414–433. 3

[76] Hascoet, L. and Pascual, V. (2013). The Tapenade automatic differentiation tool: principles, model, and specification. *ACM Transactions on Mathematical Software (TOMS)*, 39(3):20. 8, 43, 44

[77] Hazra, S., Schulz, V., Brezillon, J., and Gauger, N. (2005). Aerodynamic shape optimization using simultaneous pseudo-timestepping. *Journal of Computational Physics*, 204(1):46–64. 2, 3, 62

[78] Hazra, S. B. (2008). Multigrid one-shot method for aerodynamic shape optimization. *SIAM Journal on Scientific Computing*, 30(3):1527–1547. 61

[79] Hazra, S. B. and Schulz, V. (2004). Simultaneous pseudo-timestepping for pde-model based optimization problems. *Bit Numerical Mathematics*, 44(3):457–472. 2, 62

[80] Hazra, S. B. and Schulz, V. (2006). Simultaneous pseudo-timestepping for aerodynamic shape optimization problems with state constraints. *SIAM Journal on Scientific Computing*, 28(3):1078–1099. 62

[81] He, L. and Denton, J. (1993). Three dimensional time-marching inviscid and viscous solutions for unsteady flows around vibrating blades. In *ASME 1993 International Gas Turbine and Aeroengine Congress and Exposition*, pages V001T03A033–V001T03A033. American Society of Mechanical Engineers. 4

[82] Heres, P. and Schilders, W. (2004). Orthogonalisation in krylov subspace methods for model order reduction. *Scientific Computing in Electrical Engineering*, 9. 10

[83] Heres, P. and Schilders, W. (2006). Krylov subspace methods in the electronic industry. In *Progress in Industrial Mathematics at ECMI 2004*, pages 139–143. Springer. 10

[84] Hicks, R. M. and Henne, P. A. (1978). Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412. 2

[85] Hicks, R. M., Murman, E. M., and Vanderplaats, G. N. (1974). An assessment of airfoil design by numerical optimization. 5

[86] Hirsch, C. (2002). *Numerical computation of internal and external flows: volume 2: computational methods for inviscid and viscous flows*. Chichester [etc.]: John Wiley & Sons. 29

[87] Hirsch, C. (2007). *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Butterworth-Heinemann. 16, 17

[88] Howison, J. and Ekici, K. (2013). Unsteady analysis of wind turbine flows using the harmonic balance method. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 1107. 22

[89] Howison, J., Thomas, J., and Ekici, K. (2015). Aeroelastic analysis of a wind turbine blade using the harmonic balance method. *Wind Energy*. 4

[90] Howison, J. C. (2015). *Aeroelastic Analysis of a Wind Turbine Blade Using the Harmonic Balance Method*. PhD thesis, University of Tennessee. 22, 29

[91] Huang, H. (2013). Shape optimization of turbomachinery blades using an adjoint harmonic balance method. 45, 81

[92] Huang, H. and Ekici, K. (2011). A harmonic balance method for the analysis of unsteady flows in cascades. In *47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, page 5752. 22

[93] Huang, H. and Ekici, K. (2013). An efficient harmonic balance method for unsteady flows in cascades. *Aerospace Science and Technology*, 29(1):144–154. 22

[94] Huang, H. and Ekici, K. (2014). A discrete adjoint harmonic balance method for turbomachinery shape optimization. *Aerospace Science and Technology*, 39:481–490. 2, 7, 8, 9, 81

[95] Iollo, A., Kuruvila, G., and Ta'asan, S. (1995). Pseudo-time method for optimal shape design using the euler equations. Technical report, INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING HAMPTON VA. 62

[96] Jameson, A. (1983). Solution of the euler equations for two dimensional transonic flow by a multigrid method. *Appl. Math. Comput*, 13(3-4):327–355. 33

[97] Jameson, A. (1986). A vertex based multigrid algorithm for three dimensional compressible flow calculations. In *ASME Symposium on Numerical Methods for Compressible Flow, Anaheim*. 22

[98] Jameson, A. (1988). Aerodynamic design via control theory. *Journal of scientific computing*, 3(3):233–260. 1, 6, 7, 37

[99] Jameson, A. (1991). Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In *10th Computational Fluid Dynamics Conference*, page 1596. 3

[100] Jameson, A., Alonso, J., and McMullen, M. (2002). Application of a non-linear frequency domain solver to the euler and navier-stokes equations. In *40th AIAA Aerospace Sciences Meeting & Exhibit*, page 120. 4

[101] Jameson, A. and Baker, T. (1983). Solution of the euler equations for complex configurations. In *6th Computational Fluid Dynamics Conference Danvers*, page 1929. 31

[102] Jameson, A., Martinelli, L., and Pierce, N. (1998). Optimum aerodynamic design using the navier–stokes equations. *Theoretical and computational fluid dynamics*, 10(1-4):213–237. 6, 7

[103] Jameson, A., Schmidt, W., and Turkel, E. (1981). Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, page 1259. 23, 25

[104] Jaworski, A., Cusdin, P., and Müller, J. (2005). Uniformly converging simultaneous time-stepping methods for optimal design. *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN*. 3

[105] Jespersen, D. C. and Buning, P. G. (1985). Accelerating an iterative process by explicit annihilation. *SIAM Journal on Scientific and Statistical Computing*, 6(3):639–651. 10

[106] Jones, D., Müller, J.-D., and Christakopoulos, F. (2011). Preparation and assembly of discrete adjoint cfd codes. *Computers & Fluids*, 46(1):282–286. 8

[107] Kaminsky, A. L., Djeddi, R., and Ekici, K. (2017). An efficient reduced-order-model for accurate projection of adjoint sensitivities. In *55th AIAA Aerospace Sciences Meeting*, page 0037. 2, 11, 66, 81, 87

[108] Kaminsky, A. L., Djeddi, R., and Ekici, K. (2018). Convergence acceleration of continuous adjoint solvers using a reduced-order model. *International Journal for Numerical Methods in Fluids*, 86(9):582–606. 11, 66, 86

[109] Kaminsky, A. L. and Ekici, K. (2016). Sensitivity and stability derivative analysis using an efficient adjoint harmonic balance technique. AIAA Paper 2016-0808. xi, 7, 47, 118

[110] Kaminsky, A. L. and Ekici, K. (2019). Reduced-order model-based convergence acceleration of reverse mode discrete adjoint solvers. *Aerospace Science and Technology*, 93:105334. 11, 66, 83

[111] Kim, S., Alonso, J. J., and Jameson, A. (2004). Multi-element high-lift configuration design optimization using viscous continuous adjoint method. *Journal of Aircraft*, 41(5):1082–1097. 7

[112] Knoll, D. A. and Keyes, D. E. (2004). Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397. 67

[113] Kuruvila, G., Ta, S., et al. (1995). Airfoil design and optimization by the one-shot method. 3

[114] Kusch, L. (2020). *Robustness Measures and Optimization Strategies for Multi-Objective Robust Design*. Technische Universität Kaiserslautern. 61, 62, 63

[115] Kusch, L., Albring, T., Walther, A., and Gauger, N. R. (2018). A one-shot optimization framework with additional equality constraints applied to multi-objective aerodynamic shape optimization. *Optimization Methods and Software*, 33(4-6):694–707. 2, 62

[116] Landon, R. (2000). Naca 0012 oscillatory and transient pitching. Technical report, AIRCRAFT RESEARCH ASSOCIATION LTD BEDFORD (UNITED KINGDOM). 54

[117] Lassaux, G. (2002). *High-fidelity reduced-order aerodynamic models: Application to active control of engine inlets*. PhD thesis, Massachusetts Institute of Technology. 10

[118] Launder, B. E. and Spalding, D. B. (1983). The numerical computation of turbulent flows. In *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*, pages 96–116. Elsevier. 16

[119] LeGresley, P. and Alonso, J. (2001). Investigation of non-linear projection for pod based reduced order models for aerodynamics. In *39th Aerospace Sciences Meeting and Exhibit*, page 926. 10

[120] Leung, T. M. and Zingg, D. W. (2012). Aerodynamic shape optimization of wings using a parallel Newton-Krylov approach. *AIAA Journal*, 50(3):540–550. 85, 90

[121] Li, H. and Ekici, K. (2018). An improved one-shot approach for modeling viscous transonic limit cycle oscillations. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0460. 3

[122] Liem, R. P., Kenway, G. K., and Martins, J. R. (2015). Multimission aircraft fuel-burn minimization via multipoint aerostructural optimization. *AIAA Journal*, 53(1):104–122. 2

[123] Lions, J. L. (1971). Optimal control of systems governed by partial differential equations problèmes aux limites. 6

[124] Liu, F., Cai, J., Zhu, Y., Tsai, H., and F. Wong, A. (2001). Calculation of wing flutter by a coupled fluid-structure method. *Journal of Aircraft*, 38(2):334–342. 4

[125] Liu, Y., Zhang, W., and Kou, J. (2018). Mode multigrid-a novel convergence acceleration method. *arXiv preprint arXiv:1802.08962*. 10

[126] Loeve, M. (1978). Probability theory, vol. ii. *Graduate texts in mathematics*, 46:0–387. 9

[127] Lozano, C. (2016). A note on the dual consistency of the discrete adjoint quasi-one-dimensional euler equations with cell-centered and cell-vertex central discretizations. *Computers & Fluids*, 134:51–60. 50

[128] Lozano, C. and Ponsin, J. (2012). Remarks on the numerical solution of the adjoint quasi-one-dimensional euler equations. *International Journal for Numerical Methods in Fluids*, 69(5):966–982. xi, xii, 6, 36, 38, 40, 50, 74, 130, 144

[129] Lucia, D. J., Beran, P. S., and Silva, W. A. (2004). Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40(1-2):51–117. 9, 64

[130] Lumley, J. L. (1967). The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, 790:166–178. 10

[131] Luo, J., Xiong, J., Liu, F., and McBean, I. (2011). Three-dimensional aerodynamic design optimization of a turbine blade by using an adjoint method. *Journal of Turbomachinery*, 133(1):011026. 7

[132] Lyu, Z., Xu, Z., and Martins, J. (2014). Benchmarking optimization algorithms for wing aerodynamic design optimization. In *Proceedings of the 8th International Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China*, volume 11, page 585. 2

[133] Mader, C. A. and Martins, J. R. (2014). Computing stability derivatives and their gradients for aerodynamic shape optimization. *AIAA Journal*, 52(11):2533–2546. 54, 56

[134] Mader, C. A., RA Martins, J., Alonso, J. J., and Der Weide, E. V. (2008). Adjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA journal*, 46(4):863–873. 8, 10

[135] Mader, C. A. and RA Martins, J. R. (2012). Derivatives for time-spectral computational fluid dynamics using an automatic differentiation adjoint. *AIAA journal*, 50(12):2809–2819. 9

[136] Maple, R. C., King, P. I., Orkwis, P. D., and Wolff, J. M. (2004). Adaptive harmonic balance method for nonlinear time-periodic flows. *Journal of Computational Physics*, 193(2):620–641. 4

[137] Markovinović, R. and Jansen, J. (2006). Accelerating iterative solution methods using reduced-order models as solution predictors. *International Journal for Numerical Methods in Engineering*, 68(5):525–541. 9, 11, 64, 77

[138] Martinelli, L. and Jameson, A. (1988). Validation of a multigrid method for the reynolds averaged equations. In *26th Aerospace Sciences Meeting*, page 414. 51, 52

[139] Martinelli, L., Jameson, A., and Grasso, F. (1986). A multigrid method for the navier stokes equations. In *24th Aerospace Sciences Meeting*, page 208. 25

[140] Martins, J., Alonso, J., and Reuther, J. (2002). Complete configuration aero-structural optimization using a coupled sensitivity analysis method. In *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, page 5402. 1, 7

[141] Martins, J. R. (2022). Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391. 1, 6, 7, 37

[142] Martins, J. R., Alonso, J. J., and Reuther, J. (2001). Aero-structural wing design optimization using high-fidelity sensitivity analysis. Technical report, NATIONAL AERONAUTICS AND SPACE ADMINISTRATION MOFFETT FIELD CA AMES RESEARCH CENTER. 1, 7

[143] Martins, J. R., Alonso, J. J., and Reuther, J. J. (2005). A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62. 1, 7

[144] Martins, J. R. and Ning, A. (2021). *Engineering design optimization*. Cambridge University Press. 35

[145] Martins, J. R., Sturdza, P., and Alonso, J. J. (2003). The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262. 5

[146] Masters, D. A., Taylor, N. J., Rendall, T., Allen, C. B., and Poole, D. J. (2017). Geometric comparison of aerofoil shape parameterization methods. *AIAA Journal*. 5

[147] Mavriplis, D. J. and Jameson, A. (1990). Multigrid solution of the navier-stokes equations on triangular meshes. *AIAA journal*, 28(8):1415–1425. 26

[148] McMullen, M. S. and Jameson, A. (2006). The computational efficiency of non-linear frequency domain methods. *Journal of Computational Physics*, 212(2):637–661. 4

[149] Nadarajah, S. and Jameson, A. (2000). A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, page 667. 6, 37

[150] Nadarajah, S. and Jameson, A. (2007). Optimum shape design for unsteady three-dimensional viscous flows using a nonlinear frequency-domain method. *Journal of Aircraft*, 44(5):1513–1527. 7

[151] Nadarajah, S., McMullen, M., and Jameson, A. (2003). Optimum shape design for unsteady flow using time accurate and nonlinear frequency domain methods. *AIAA Pap*, (2003-3875). 7

[152] Neidinger, R. D. (2010). Introduction to automatic differentiation and matlab object-oriented programming. *SIAM review*, 52(3):545–563. 8

[153] Newman III, J. C., Taylor III, A. C., Barnwell, R. W., Newman, P. A., and Hou, G. J.-W. (1999). Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *Journal of Aircraft*, 36(1):87–96. 36

[154] Nielsen, E. J. (1998). *Aerodynamic design sensitivities on an unstructured mesh using the Navier-Stokes equations and a discrete adjoint formulation*. PhD thesis, Virginia Tech. 7

[155] Nielsen, E. J. and Anderson, W. K. (1999). Aerodynamic design optimization on unstructured meshes using the navier-stokes equations. *AIAA journal*, 37(11):1411–1419. 7

[156] Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer. 60, 90

[157] Özkaya, E. (2014). *One-shot methods for aerodynamic shape optimization*. PhD thesis, Aachen. Zsfassung in dt. und engl. Sprache; Aachen, Techn. Hochsch., Diss., 2014. 62

[158] Özkaya, E. and Gauger, N. R. (2009). Single-step one-shot aerodynamic shape optimization. In *Optimal control of coupled systems of partial differential equations*, pages 191–204. Springer. 2, 61, 63

[159] Philibert, J. (2006). One and a half century of diffusion: Fick, einstein, before and beyond. *Diffusion fundamentals*, 4(6):1–19. 15

[160] Pierce, N. A. and Giles, M. B. (2000). Adjoint recovery of superconvergent functionals from pde approximations. *SIAM review*, 42(2):247–264. 1

[161] Pulliam, T., Nemec, M., Holst, T., and Zingg, D. (2003). Comparison of evolutionary (genetic) algorithm and adjoint methods for multi-objective viscous airfoil optimizations. In *41st Aerospace Sciences Meeting and Exhibit*, page 298. 2

[162] Ramsay, R., Hoffman, M., and Gregorek, G. (1995). Effects of grit roughness and pitch oscillations on the s809 airfoil. Technical report, National Renewable Energy Lab., Golden, CO (United States). 83

[163] Rasch, A., Bucker, H. M., and Bischof, C. H. (2008). Automatic computation of sensitivities for a parallel aerodynamic simulation. *Parallel Computing: Architectures, Algorithms and Applications. Advances in Parallel Computing*, 15:303–310. 6

[164] Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D. (1996). Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In *34th Aerospace Sciences Meeting and Exhibit*, page 94. 1, 7

[165] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D. (1999). Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of aircraft*, 36(1):51–60. 1, 7

[166] Rewienski, M. and White, J. (2003). A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 22(2):155–170. 10

[167] Reynolds, O. (1895). On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. A*, 186:123–164. 17

[168] Rowley, C. W. (2002). *Modeling, simulation, and control of cavity flow oscillations*. PhD thesis, California Institute of Technology. 10

[169] Rowley, C. W. (2005). Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013. 79

[170] Rumsey, C. L., Sanetrik, M. D., Biedron, R. T., Melson, N. D., and Parlette, E. B. (1996). Efficiency and accuracy of time-accurate turbulent navier-stokes computations. *Computers & Fluids*, 25(2):217–236. 4

[171] Schilders, W. H., Van der Vorst, H. A., and Rommes, J. (2008). *Model order reduction: theory, research aspects and applications*, volume 13. Springer. 9

[172] Shiriaev, D. and Griewank, A. (1996). Adol-f: Automatic differentiation of fortran codes. *Computational Differentiation: Techniques, Applications, and Tools*, pages 375–384. 44

[173] Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571. 10

[174] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D. (2014). Cfd vision 2030 study: a path to revolutionary computational aerosciences. 1

[175] Sobieszczanski-Sobieski, J. (1987). The case for aerodynamic sensitivity analysis. 5

[176] Somers, D. M. (1997). Design and experimental results for the s809 airfoil. Technical report, National Renewable Energy Lab., Golden, CO (United States). 83, 84, 87, 88

[177] Spalart, P. and Allmaras, S. (1992). A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439. 16, 18

[178] Spiker, M., Thomas, J., Hall, K., Kielb, R., and Dowell, E. (2006). Modeling cylinder flow vortex shedding with enforced motion using a harmonic balance approach. In *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th*, page 1965. 4

[179] STOKES, G. (1845). On the theories of internal friction of fluids in motion. *Trans. Camb. Philos. Soc.*, 8:287–305. 16

[180] Sutherland, W. (1893). Lii. the viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531. 16

[181] Swanson, R. C. and Turkel, E. (1992). On central-difference and upwind schemes. *Journal of computational physics*, 101(2):292–306. 24, 52, 53

[182] Ta'asan, S., Kuruvila, G., and Salas, M. (1992). Aerodynamic design and optimization in one shot. In *30th aerospace sciences meeting and exhibit*, page 25. 1, 61

[183] Tai, C.-H., Sheu, J.-H., and Tzeng, P.-Y. (1996). Improvement of explicit multistage schemes for central spatial discretization. *AIAA journal*, 34(1):185–188. 27

[184] Taasan, S. (1995). Pseudo-time methods for constrained optimization problems governed by pde. Technical report. 2, 62

[185] Thomas, J. P., Dowell, E. H., and Hall, K. C. (2002). Nonlinear inviscid aerodynamic effects on transonic divergence, flutter, and limit-cycle oscillations. *AIAA journal*, 40(4):638–646. 4

[186] Thomas, J. P., Dowell, E. H., and Hall, K. C. (2004). Modeling viscous transonic limit cycle oscillation behavior using a harmonic balance approach. *Journal of aircraft*, 41(6):1266–1274. 4

[187] Thomas, J. P., Dowell, E. H., and Hall, K. C. (2010). Using automatic differentiation to create a nonlinear reduced-order-model aerodynamic solver. *AIAA journal*, 48(1):19–24. 43

[188] Thomas, J. P., Hall, K. C., and Dowell, E. H. (2005). Discrete adjoint approach for modeling unsteady aerodynamic design sensitivities. *AIAA journal*, 43(9):1931–1936. 6, 8

[189] Thress, J., Kaminsky, A., Djeddi, R., and Ekici, K. (2022). Monolithic one-shot optimization for time-periodic flows using harmonic balance. *AIAA Journal*, 60(6):3539–3554. 62

[190] Tiwary, S. K. and Rutenbar, R. A. (2006). Faster, parametric trajectory-based macromodels via localized linear reductions. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 876–883. ACM. 10

[191] Tjoa, I. B. and Biegler, L. T. (1991). Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, 30(2):376–385. 62

[192] Turkel, E., Swanson, R., Vatsa, V., and White, J. (1991). Multigrid for hypersonic viscous two-and three-dimensional flows. In *10th Computational Fluid Dynamics Conference*, page 1572. 31

[193] Van Der Weide, E., Gopinath, A., and Jameson, A. (2005). Turbomachinery applications with the time spectral method. In *35th AIAA Fluid Dynamics Conference and Exhibit*, page 4905. 27

[194] Van Zante, D., Chen, J., Hathaway, M., and Chriss, R. (2008). The influence of compressor blade row interaction modeling on performance estimates from time-accurate, multistage, navier–stokes simulations. *Journal of Turbomachinery*, 130(1):011009. 3

[195] Vassberg, J. C. and Jameson, A. (2010). In pursuit of grid convergence for two-dimensional euler solutions. *Journal of Aircraft*, 47(4):1152–1166. 54, 56, 84

[196] Veldman, A. (2005). Quasi-simultaneous viscous-inviscid interaction for transonic airfoil flow. In *4th AIAA Theoretical Fluid Mechanics Meeting*, page 4801. 52

[197] Wang, D. and He, L. (2010). Adjoint aerodynamic design optimization for blades in multistage turbomachines—part i: Methodology and verification. *Journal of Turbomachinery*, 132(2):021011. 7

[198] Wang, D., He, L., Li, Y., and Wells, R. (2010). Adjoint aerodynamic design optimization for blades in multistage turbomachines—part ii: Validation and application. *Journal of Turbomachinery*, 132(2):021012. 2, 7

[199] Wang, Q. (2009). *Uncertainty quantification for unsteady fluid flow using adjoint-based approaches*. Stanford University. 9, 46

[200] Wang, Q., Moin, P., and Iaccarino, G. (2009). Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567. 8

[201] Wilcox, D. C. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA journal*, 26(11):1299–1310. 16

[202] Willcox, K., Peraire, J., and White, J. (2002). An arnoldi approach for generation of reduced-order models for turbomachinery. *Computers & fluids*, 31(3):369–389. 10

[203] Wolfe, W. and Ochs, S. (1997). CFD calculations of S809 aerodynamic characteristics. AIAA Paper 97–0973. 88, 160

[204] Wright, S. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35(67-68):7. 3, 87

[205] Wu, H.-Y., Yang, S., Liu, F., and Tsai, H.-M. (2003). Comparison of three geometric representations of airfoils for aerodynamic optimization. In *AIAA Paper*. Citeseer. 81

[206] Wynn, P. (1962). Acceleration techniques for iterated vector and matrix problems. *Mathematics of Computation*, 16(79):301–322. 10

[207] Yang, Y.-J. and Shen, K.-Y. (2004). Nonlinear heat-transfer macromodeling for mems thermal devices. *Journal of micromechanics and microengineering*, 15(2):408. 10

[208] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560. 88

[209] Zou, X., Navon, I. M., Berger, M., Phua, K. H., Schlick, T., and Le Dimet, F.-X. (1993). Numerical experience with limited-memory quasi-Newton and truncated Newton methods. *SIAM Journal on Optimization*, 3(3):582–608. 85

[210] Zymaris, A., Papadimitriou, D., Giannakoglou, K., and Othmer, C. (2009). Continuous adjoint approach to the spalart–allmaras turbulence model for incompressible flows. *Computers & Fluids*, 38(8):1528–1538. 7, 42

Appendix

Table 1: Coefficients for hybrid multistage Runge–Kutta scheme

Stage	α	β
1	0.2500	1.0000
2	0.1667	0.0000
3	0.3750	0.5600
4	0.5000	0.0000
5	1.0000	0.4400

Table 2: Flow condition at far field boundary for one-dimensional flow.

Condition	Extrapolated	Specified
Subsonic inlet	1	2
Subsonic outlet	2	1
Supersonic inlet	3	0
Supersonic outlet	0	3

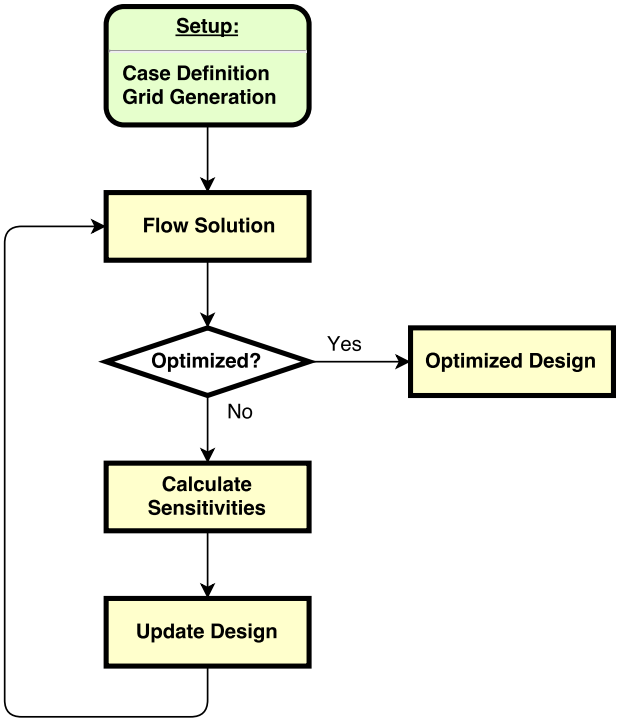
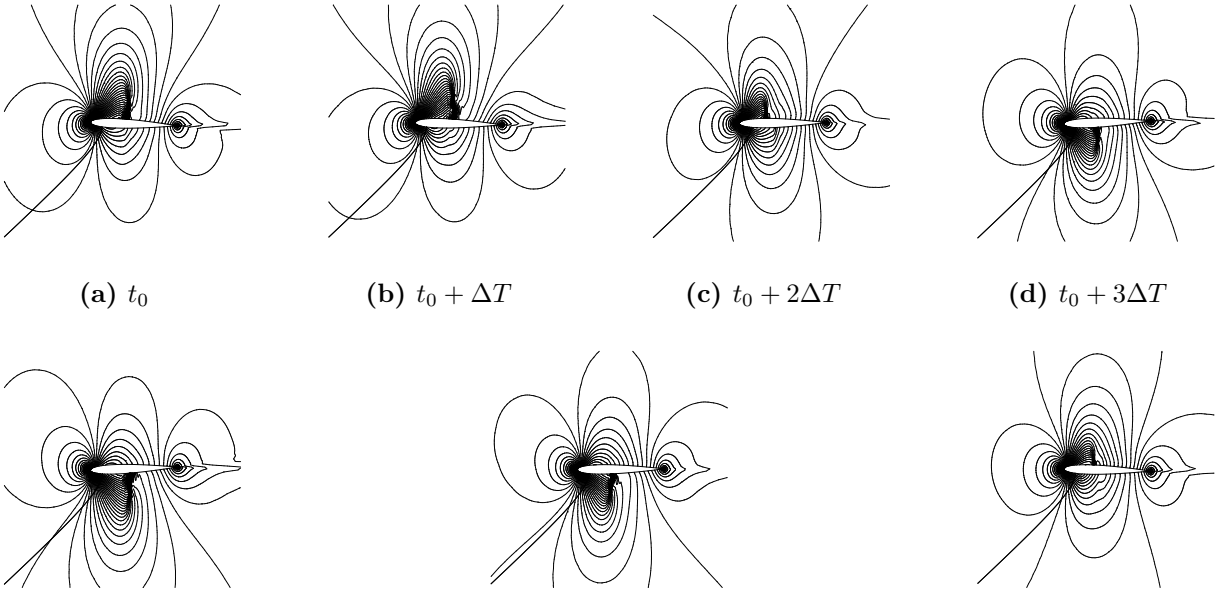


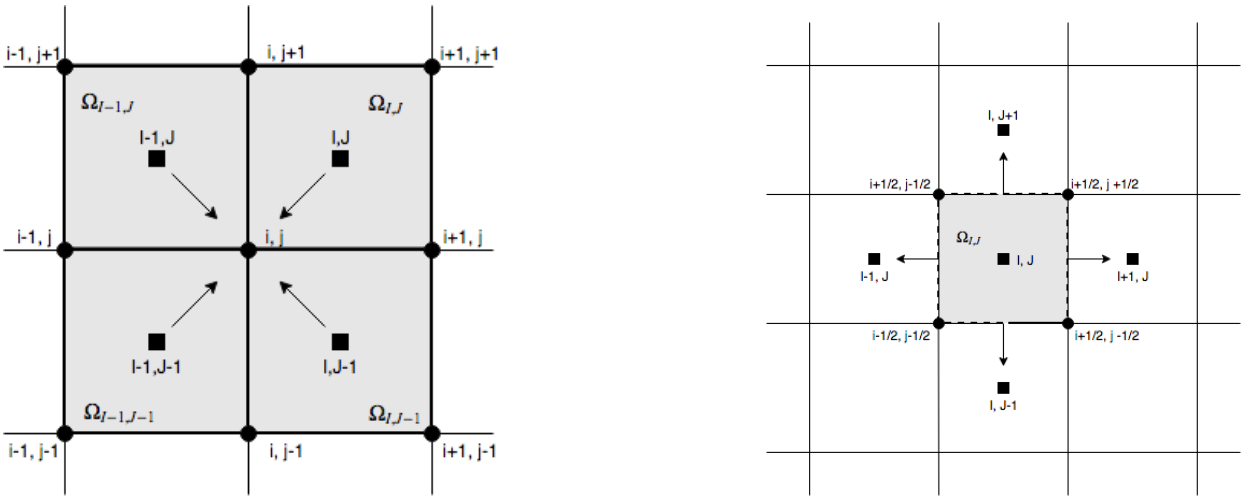
Figure 1: Nested gradient-based design-optimization flow chart.



(a) t_0 **(b)** $t_0 + \Delta T$ **(c)** $t_0 + 2\Delta T$ **(d)** $t_0 + 3\Delta T$

(e) $t_0 + 4\Delta T$ **(f)** $t_0 + 5\Delta T$ **(g)** $t_0 + 6\Delta T$

Figure 2: Sub-time level Mach number contours for a pitching NACA 0012. [109]



(a) Cell-vertex **(b)** Cell-centered

Figure 3: Spatial discretization using (a) cell-vertex and (b) cell-centered schemes [16].

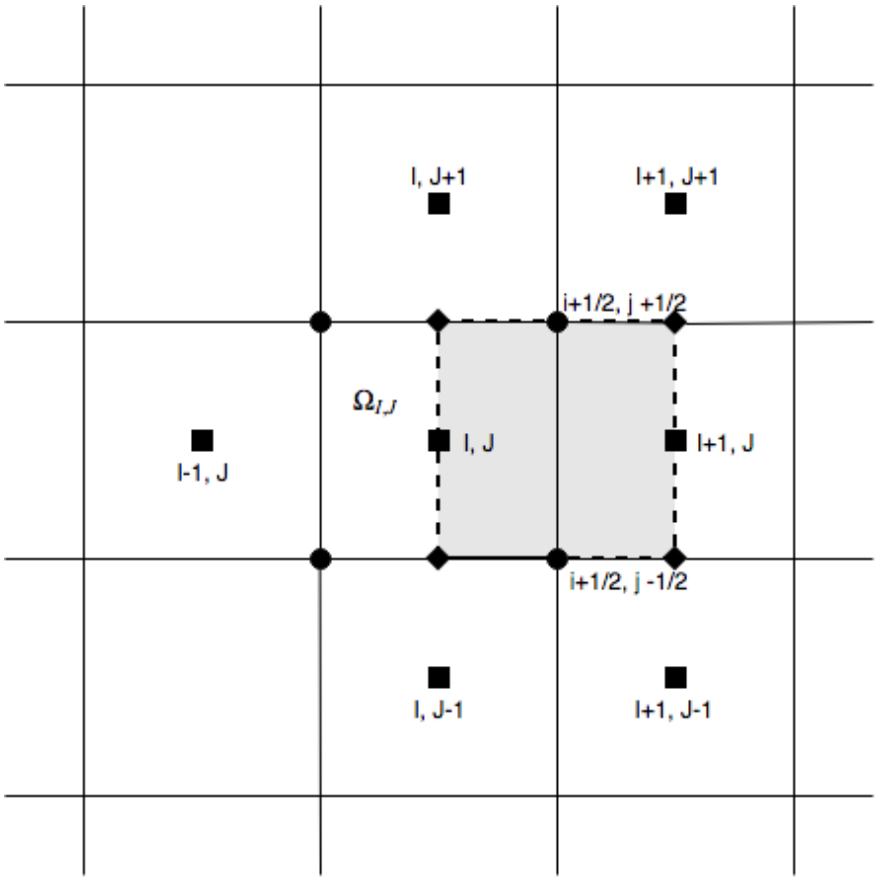


Figure 4: The auxiliary control volume for derivative evaluation in a cell-centered scheme [16].

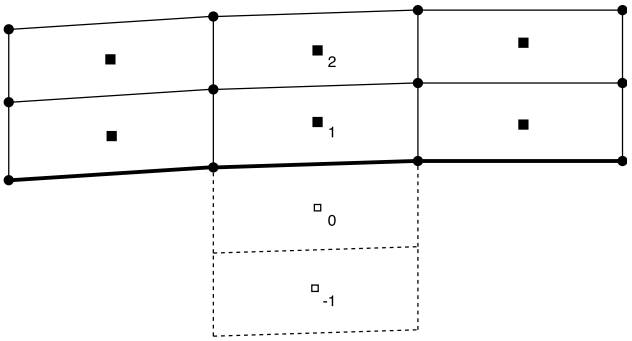
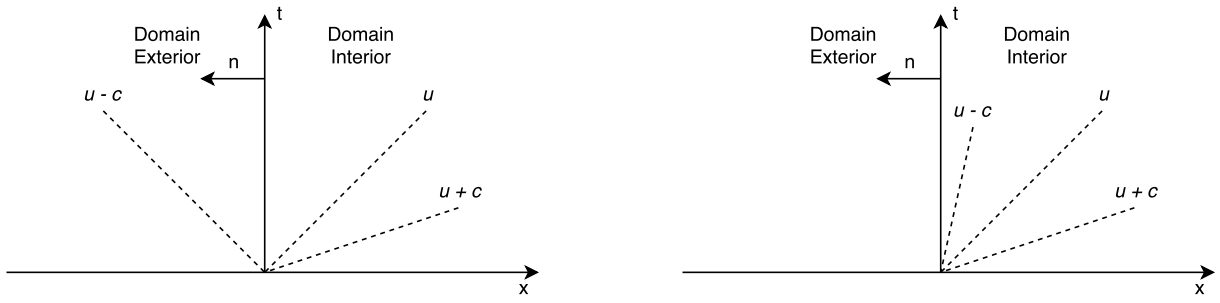


Figure 5: Ghost cell indices across solid boundary.



(a) Subsonic **(b)** Supersonic

Figure 6: Characteristics at the flow inlet for (a) subsonic and (b) supersonic flow.

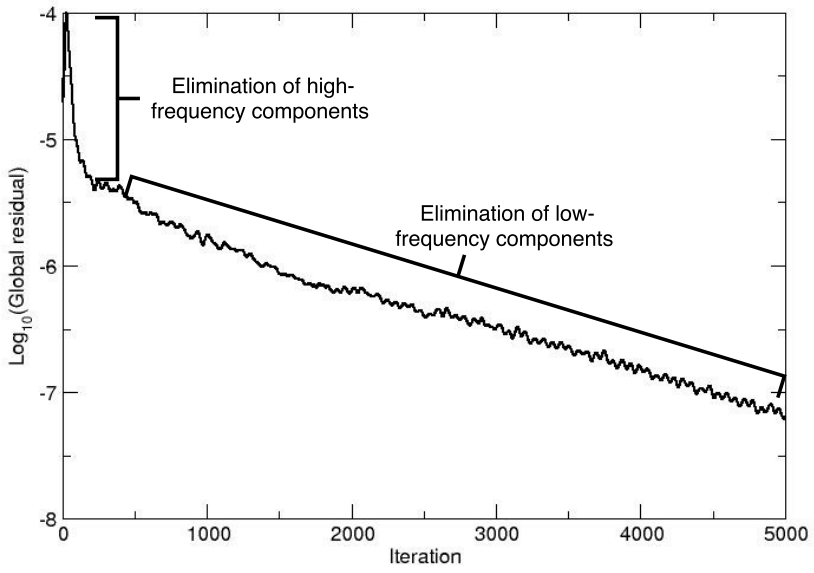
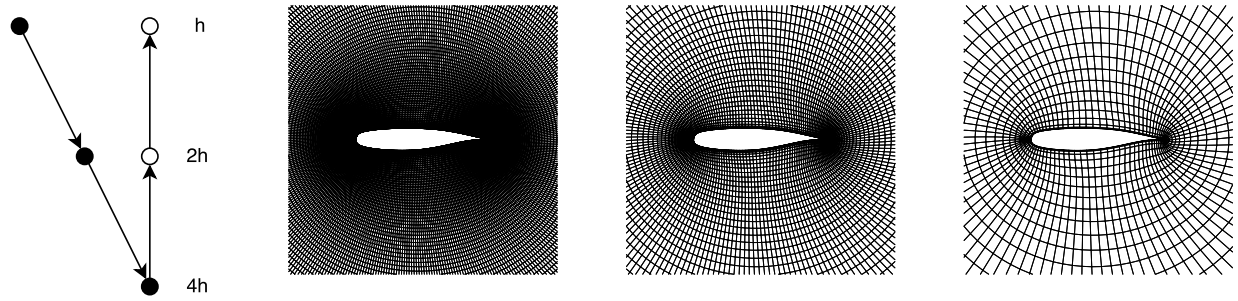
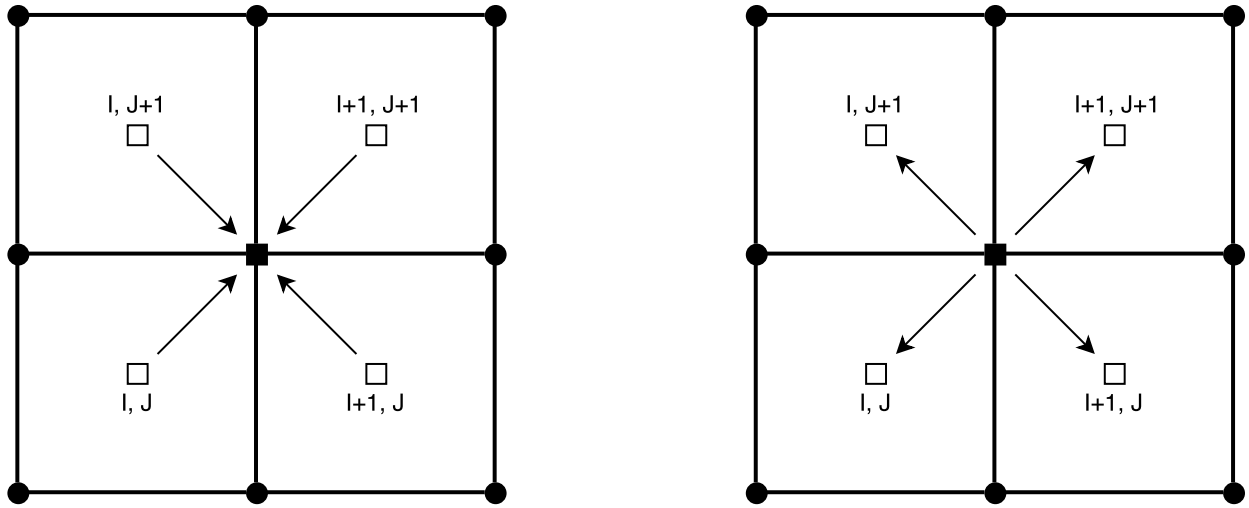


Figure 7: Typical convergence history of an explicit steady state solver.



(a) Multigrid cycle **(b)** p! **(c)** 2h **(d)** 4h

Figure 8: The multigrid cycle from fine to coarse and back. Here • denotes restriction and ◦ corresponds to prolongation.



(a) Restriction **(b)** Prolongation

Figure 9: Cell-centered multigrid (a) Restriction and (b) Prolongation. Here are the centers of the fine grid and ■ is the center of the coarse grid.

```
module forward_AD
implicit none

! Variables now contain the variable and its derivative
type real00
    real :: d0 ! variable
    real :: d1 ! derivative
end type real00

! interface definitions
interface operator (*)
    module procedure multiplication_d
end interface

contains

function multiplication_d (x,y) result (xy)
    type (real00), intent(in) :: x
    type (real00), intent(in) :: y
    type (real00)             :: xy
    xy.d0 = x.d0*y.d0
    xy.d1 = x.d1*y.d0 + x.d0*y.d1
end function multiplication_d
```

Figure 10: Forward mode operator overloading for multiplication.

```
program main
use forward_AD
implicit none

type(real00) :: f,x,y

x%d0 = 5.0
x%d1 = 1.0
y%d0 = 3.0
y%d1 = 0.0

f = x*y

write(*,*) 'derivatives of f = x*y for x = 5, y = 3:'
write(*,*) '----------------------------------------'
write(*,*) 'f(x=5) = ',f%d0
write(*,*) 'df/dx(x=5) = ',f%d1

end program main
```

Figure 11: Code utilizing operator overloading.

```
! - COMPUTE CONSERVATION VARIABLES AT NODES
rho_d    = qq_d(1,i,j,n)
rho      = qq(1,i,j,n)
rhou_d   = qq_d(2,i,j,n)
rhou     = qq(2,i,j,n)
rhov_d   = qq_d(3,i,j,n)
rhov     = qq(3,i,j,n)
rhoe_d   = qq_d(4,i,j,n)
rhoe     = qq(4,i,j,n)
rhoft_d  = qq_d(5,i,j,n)
rhoft    = qq(5,i,j,n)

! - COMPUTE PRIMITIVE VARIABLES AT NODES
rhoinv_d = -(rho_d/rho**2)
rhoinv   = 1.0/rho
u_d  = rhou_d*rhoinv + rhou*rhoinv_d
u    = rhou*rhoinv
v_d  = rhov_d*rhoinv + rhov*rhoinv_d
v    = rhov*rhoinv
p_d  = gam1*(rhoe_d-0.5*(rhou_d*u+rhou*u_d+rhov_d*v+ rhov*v_d))
p    = gam1*(rhoe-0.5*(rhou*u+rhov*v))
h_d  = (rhoe_d+p_d)*rhoinv + (rhoe+p)*rhoinv_d
h    = (rhoe+p)*rhoinv
ft_d = rhoft_d*rhoinv + rhoft*rhoinv_d
ft   = rhoft*rhoinv

! - X FLUX
f1_d(1,i,j,n) = rhou_d - fdot*rho_d
f1(1,i,j,n)   = rhou - rho*fdot
f1_d(2,i,j,n) = rhou_d*u + rhou*u_d + p_d - fdot*rhou_d
f1(2,i,j,n)   = rhou*u + p - rhou*fdot
f1_d(3,i,j,n) = rhou_d*v + rhou*v_d - fdot*rhov_d
f1(3,i,j,n)   = rhou*v - rhov*fdot
f1_d(4,i,j,n) = rhou_d*h + rhou*h_d - fdot*rhoe_d
f1(4,i,j,n)   = rhou*h - rhoe*fdot
f1_d(5,i,j,n) = rhou_d*ft + rhou*ft_d - fdot*rhoft_d
f1(5,i,j,n)   = rhou*ft - rhoft*fdot
```

Figure 12: Forward Mode Automatic Differentiation of Convective Flux Subroutine.

```
call generate_grid(β→,←y)
call initialization(→y,←U)
do iter = 1,iterMax
   call residual(→y,→U,←R)
   call update(⇋U,→R)
end do
call cost_function(→U,←J)
```

Figure 13: Simplified flow solver code.

```
call generate_grid_d(→β,←y,→β̇,←ẏ)
call initialization_d(→y,←U,→ẏ,←U̇)
do iter = 1,iterMax
   call residual_d(→y,→U,←R,→ẏ,→U̇,←Ṙ)
   call update_d(⇋U,→R,⇋U̇,→Ṙ)
end do
call cost_function_d(→U,←J,→U̇,←J̇)
```

Figure 14: Automatically differentiated flow solver code.

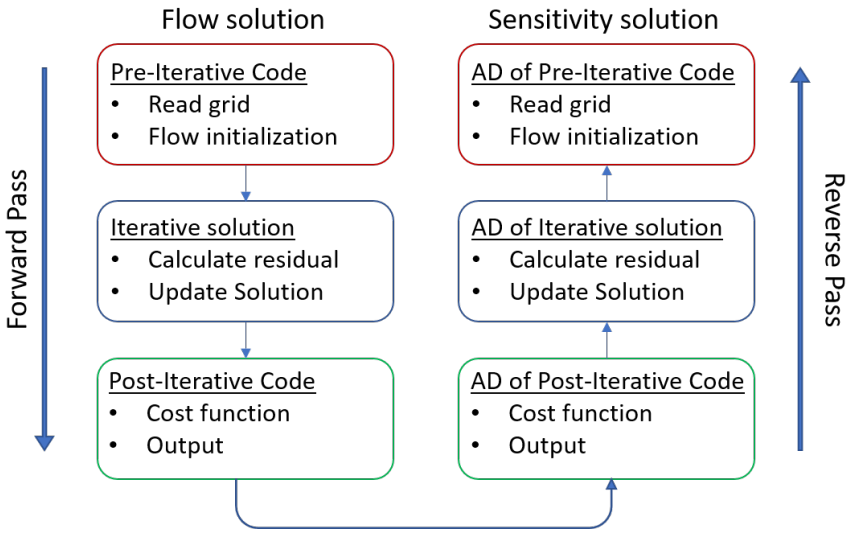


Figure 15: Reverse mode AD sensitivity calculation flow chart.

```
call generate_grid(→β,←y)
call initialization(y→,←U)
do iter = 1,iterMax
   call residual(→y,→U,←R)
   call update(⇋U,→R)
end do
call cost_function(→U,←I)
```

Figure 16: Simplified flow solver code.

```
! Forward sweep
PUSH(X,Y)
call generate_grid_cb(→β,←y)
call initialization_cb(→y,←U)
do iter = 1,iterMax
   PUSH(U,R)
   call residual_cb(→y,→U,←R)
   PUSH(U,R)
   call update_cb(⇋U,→R)
end do
PUSH(J)
call cost_function_cb(→U,←J)

! Reverse sweep
call cost_function_b(→U,←J,← Ū,→ J̄)
POP(J)
do iter = iterMax,1,-1
   call update_b(⇋U,→R,⇋ Ū,← R̄)
   POP(U,R)
   call residual_b(→y,→U,←R,← ȳ,← Ū,→ R̄)
   POP(U,R)
end do
call generate_grid_b(→β,←y,← β̄,→ ȳ)
POP(X,Y)
```

Figure 17: Unsteady adjoint code from brute force automatic differentiation.

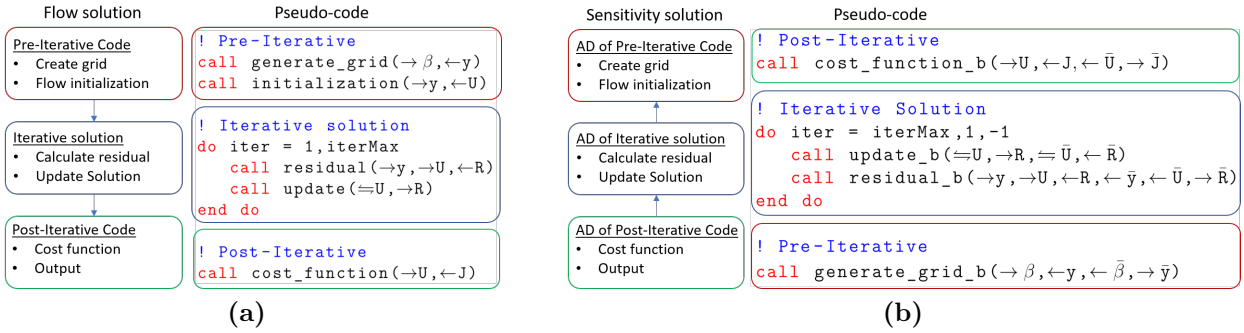


Figure 18: (a) Pseudo-code for the forward pass and (b) Pseudo-code for the reverse pass using brute force AD. Adapted from Christakopoulos et al. [27].

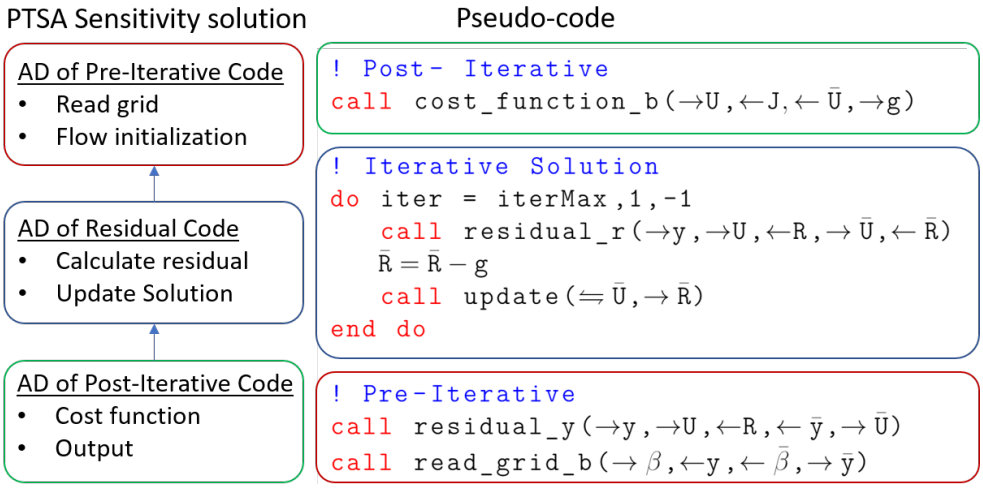


Figure 19: Pseudo-code for the reverse pass using the primal time-stepping adjoint AD approach. Adapted from Christakopoulos et al. [27]

Table 3: RAE 2822 AGARD case parameters and computational corrections

Case	Mach	α	α_{cor}	Re	C_L	C_D	C_M
AGARD Case 1	0.676	2.4°	1.83°	5.7×10^6	0.566	0.0085	-0.082
AGARD Case 6	0.725	2.92°	2.44°	6.5×10^6	0.743	0.017	-0.095
AGARD Case 9	0.73	3.19°	2.79°	6.5×10^6	0.803	0.0168	-0.099

Table 4: RAE 2822 AGARD case 1 experimental and CFD results

Case	Mach	α	Re	C_L	C_D	C_M
AGARD Case 1	0.676	2.4°	5.7×10^6	0.566	0.0085	-0.082
Case 1 CFD Correction	0.676	1.83°	5.7×10^6	0.562	0.0094	-0.083

Table 5: Computational cost for convergence acceleration techniques

Acceleration method	Iterations	Time (s)
Baseline	27155	1111.01
Residual smoothing	13440	791.91
1-level multigrid	8747	647.4
2-level multigrid	6989	602.35
2-level multigrid and residual smoothing	4798	526.82

Table 6: RAE 2822 AGARD case 6 parameters and computational corrections

Case	Mach	α	Re	C_L	C_D	C_M
AGARD Case 6	0.725	2.92°	6.5×10^6	0.743	0.017	-0.095
Case 6 CFD Correction	0.725	2.44°	6.5×10^6	0.744	0.012	-0.088

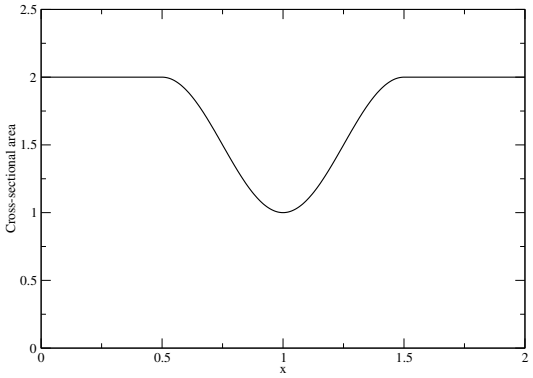
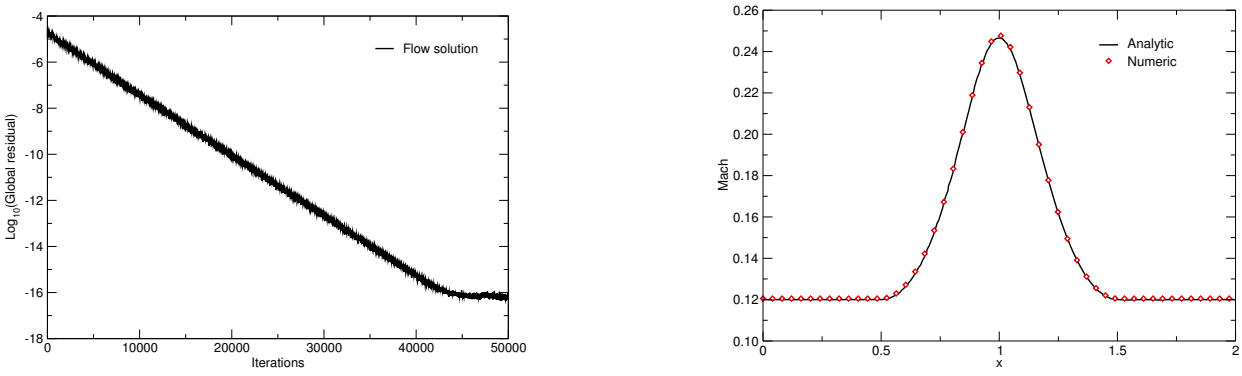
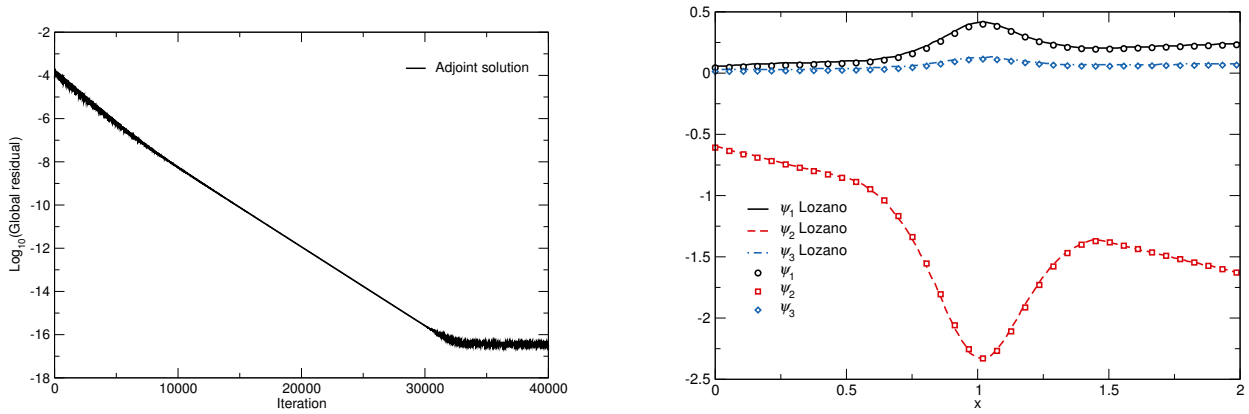


Figure 20: The cross-sectional area of the nozzle is defined by a sinusoidal decrease near the throat.



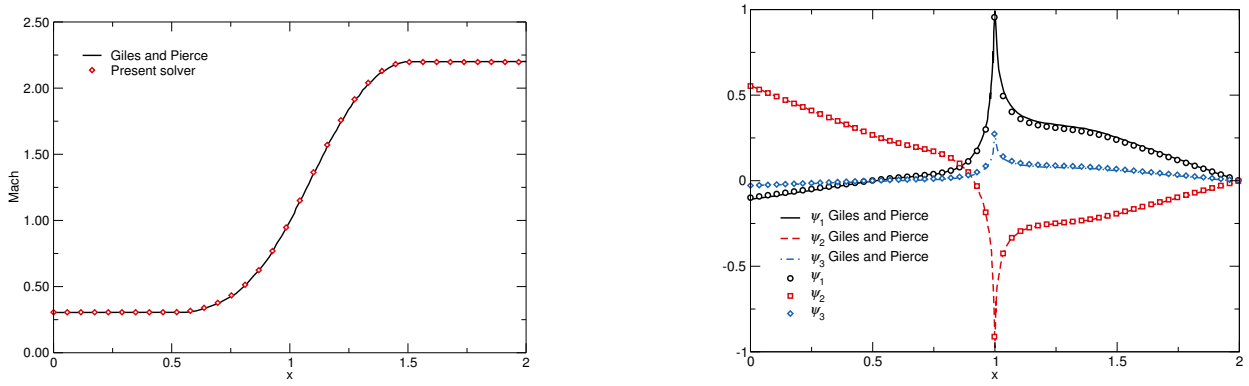
(a) **(b)**

Figure 21: Fully subsonic nozzle solution (a) convergence history and (b) Mach number distribution.



(a) **(b)**

Figure 22: Adjoint solver (a) convergence history and (b) comparison of the adjoint solution vector with that of Lozano and Ponsin [128].



(a) **(b)**

Figure 23: Transonic nozzle (a) Mach number distribution and (b) adjoint vector solution both match the values reported by Giles and Pierce [62].

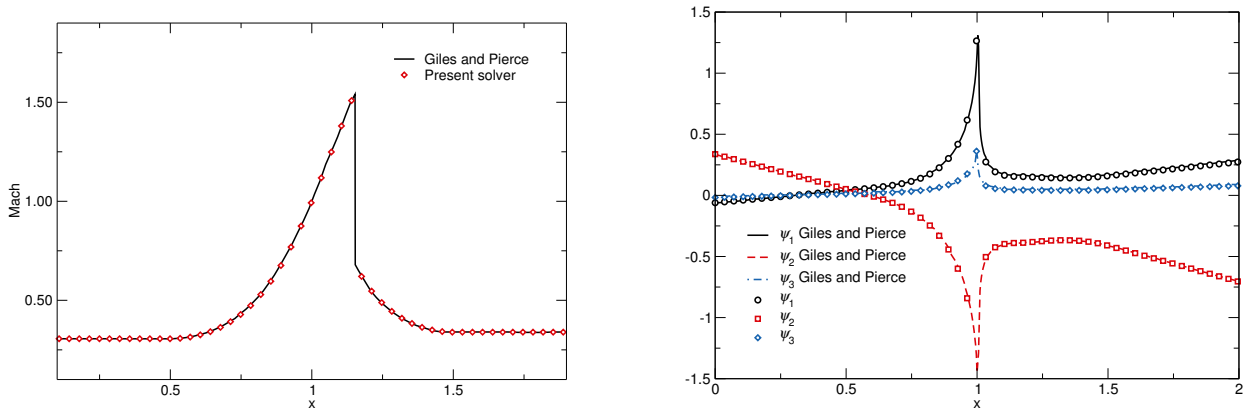


Figure 24: Shocked nozzle (a) Mach number distribution and (b) adjoint vector solution match those of Giles and Pierce [62].

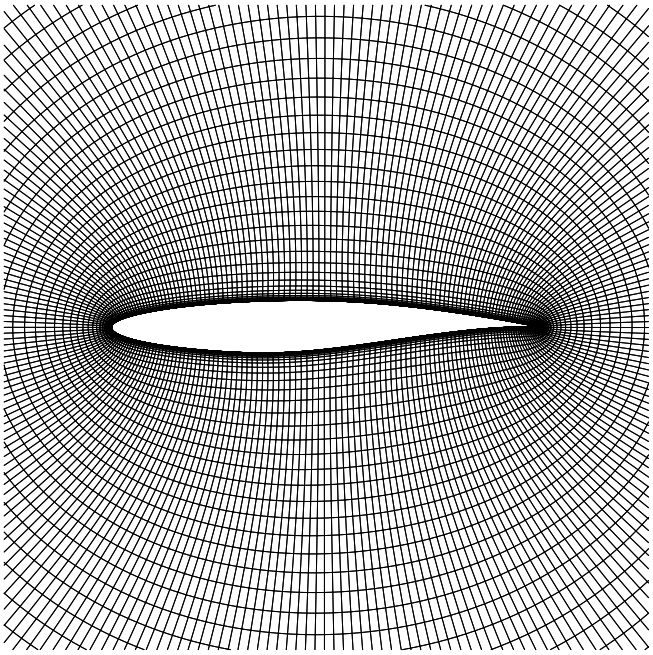
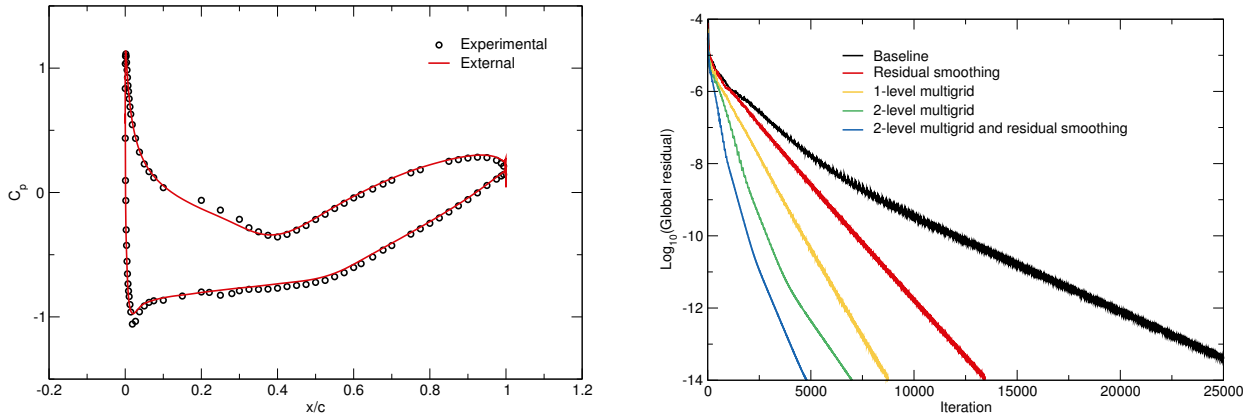


Figure 25: RAE 2822 viscous grid



(a) **(b)**

Figure 26: (a) Pressure coefficient distribution and (b) convergence history for the RAE 2822 airfoil at AGARD case 1.

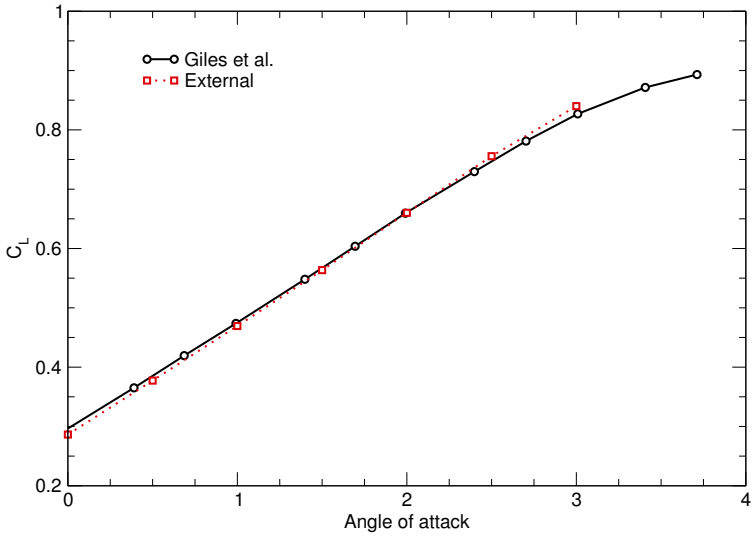


Figure 27: Lift coefficient for varying angles of attack over RAE 2822 at $M = 0.725$ and $\mathrm{Re} = 6.5 \times 10^6$

Table 7: A comparison of sensitivity values for AGARD case 6

Angle of Attack	Finite Difference $\partial C_L/\partial\alpha$	Forward $\partial C_L/\partial\alpha$	Adjoint $\partial C_L/\partial\alpha$	Giles et al.[57] $\partial C_L/\partial\alpha$
0.0°	**0.18107**899396602	**0.18140763446056**	**0.18140763446064**	–
1.0°	**0.18568**505799621	**0.18568442138793**	**0.18568442138789**	–
2.0°	**0.19174**383003584	**0.19174270151729**	**0.19174270151725**	0.1815048[1]
3.0°	**0.14459**725394822	**0.14459765476313**	**0.14459765476312**	0.1398736[2]

Table 8: A comparison of sensitivity values at different iterations

Iteration	Forward $\partial C_L/\partial\alpha$	Adjoint $\partial C_L/\partial\alpha$
1	**−0.000149042721458**	**−0.000149042721454**
100	**−0.000148838511933**	**−0.000148838511934**
1000	**0.175702978918**253	**0.175702978918**575
10000	**0.181269644794**812	**0.181269644794**956

Table 9: RAE 2822 AGARD case 9 parameters and computational corrections

Case	Mach	α	Re	C_L	C_D	C_M
AGARD Case 9	0.73	3.19°	6.5×10^6	0.803	0.0168	-0.099
Case 9 CFD Correction	0.734	2.79°	6.5×10^6	0.807	0.0164	-0.090

[1]Provided for angle of attack range $1.4 < \alpha < 2.4$.
[2]Provided for angle of attack range $2.4 < \alpha < 3.4$.

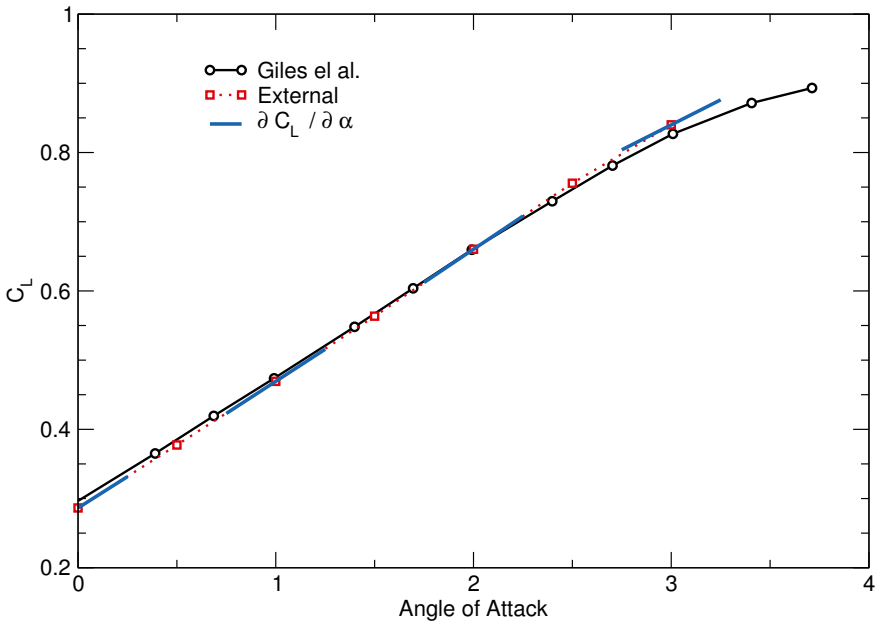
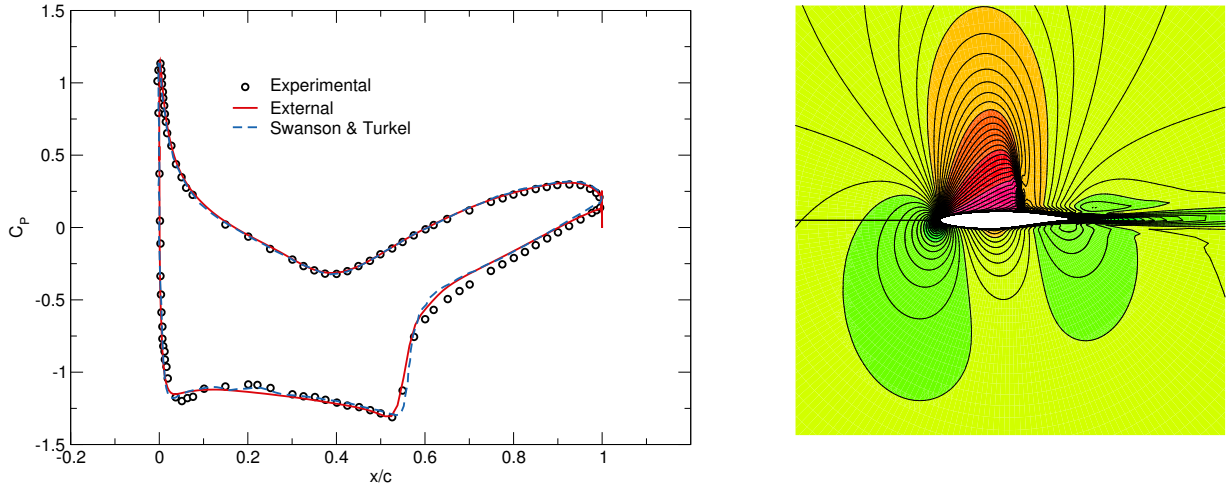


Figure 28: AGARD Case 6 lift coefficients and sensitivity to angle of attack.



(a) **(b)**

Figure 29: (a) Pressure coefficient and (b) Mach contours for the RAE airfoil at $M = 0.734$, $\alpha = 2.79°$, and 6.5×10^6.

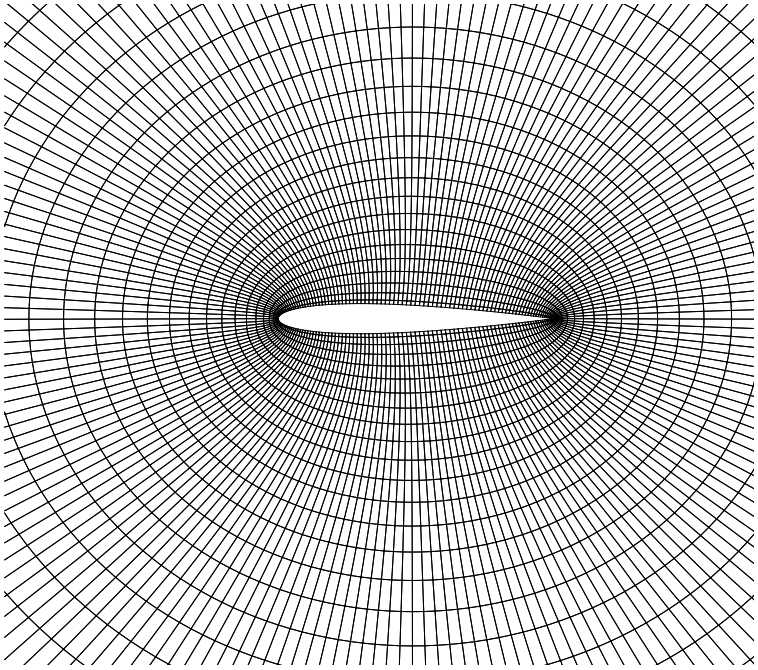
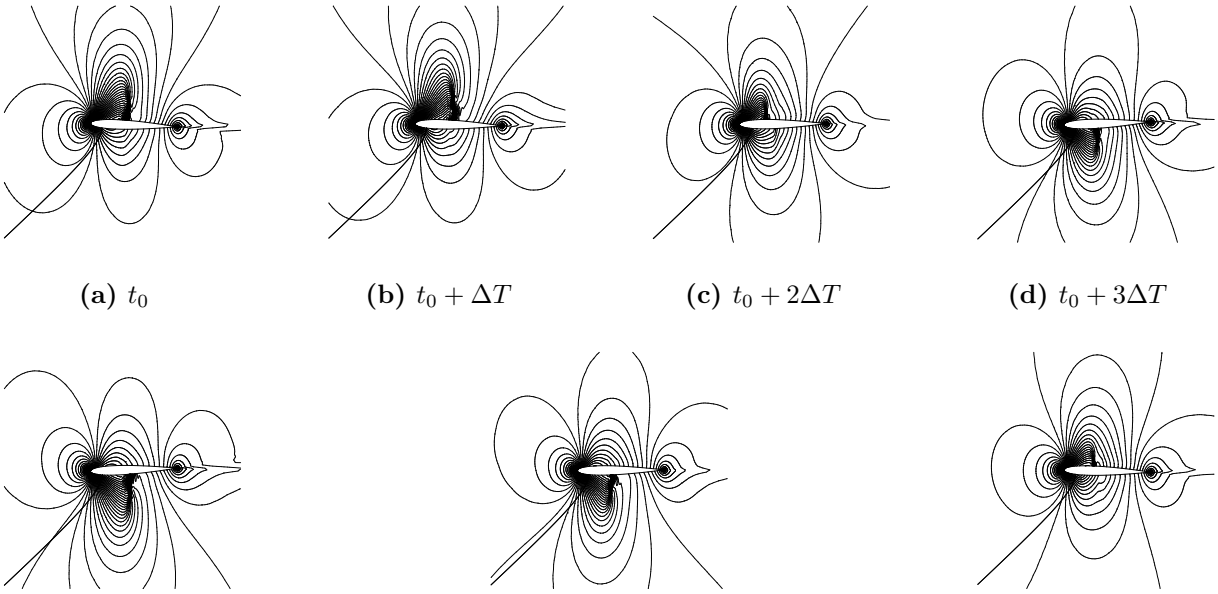


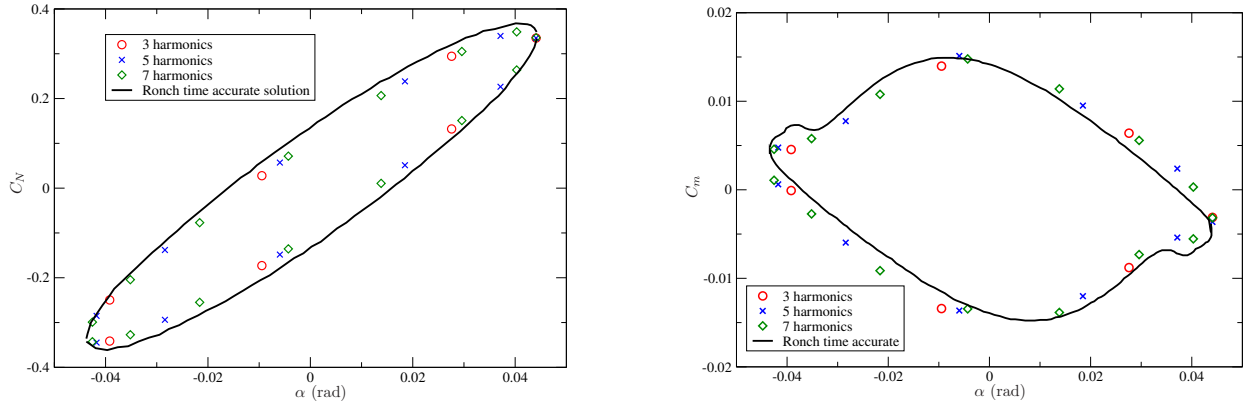
Figure 30: NACA 0012 inviscid grid



(a) t_0 **(b)** $t_0 + \Delta T$ **(c)** $t_0 + 2\Delta T$ **(d)** $t_0 + 3\Delta T$

(e) $t_0 + 4\Delta T$ **(f)** $t_0 + 5\Delta T$ **(g)** $t_0 + 6\Delta T$

Figure 31: Harmonic balance unsteady Mach number contours for unsteady NACA 0012 case at each sub-time.



(a) **(b)**

Figure 32: NACA 0012 (a) normal force and(b) pitching moment coefficients vs. α. Note: Ronch data is digitized from Ref.[33]

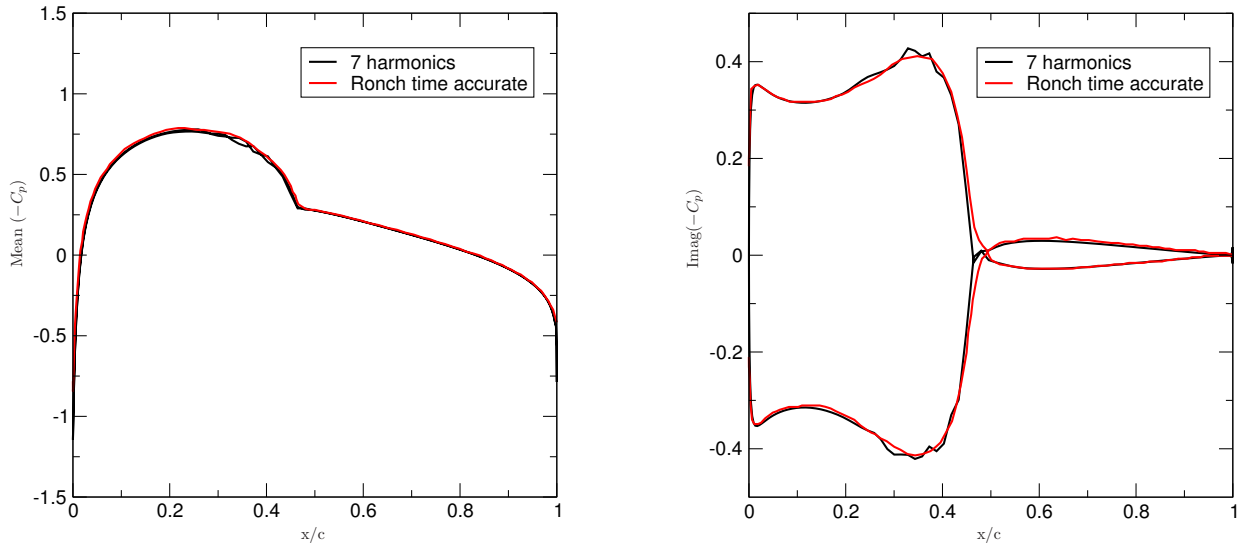


Figure 33: NACA 0012 (a) zeroth and (b) first harmonic imaginary unsteady surface pressure coefficient distribution

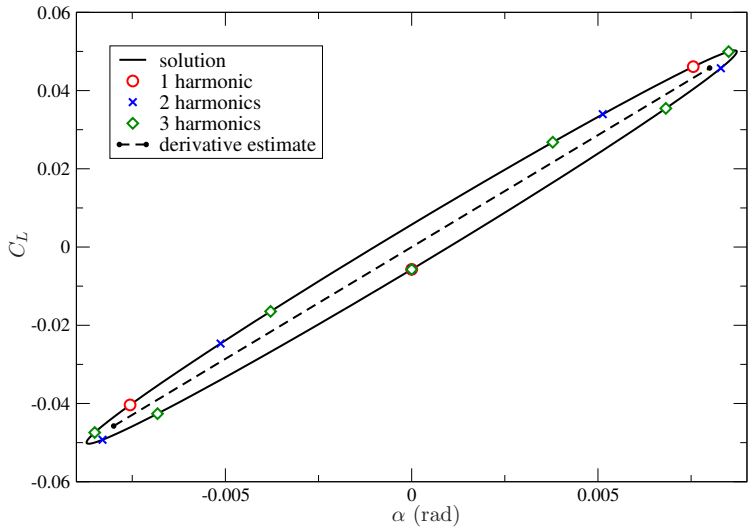


Figure 34: Typical harmonic balance solutions: C_L vs. α for an oscillating flat plate

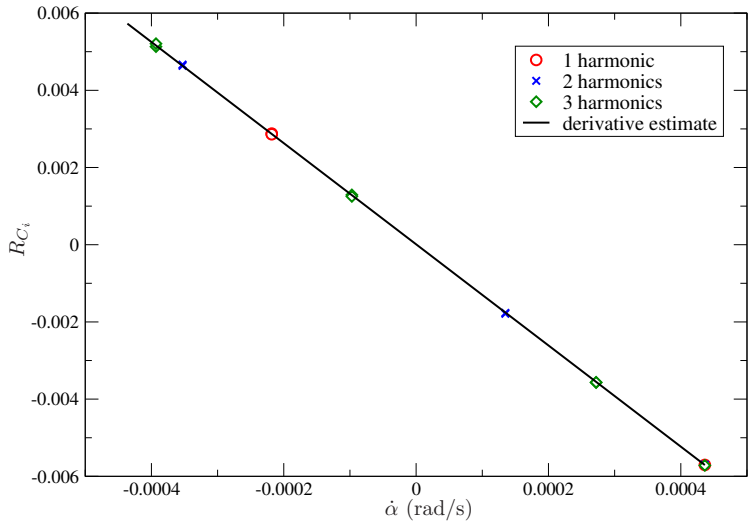


Figure 35: Typical harmonic balance solutions: R_{C_L} vs. $\dot{\alpha}$ for an oscillating flat plate

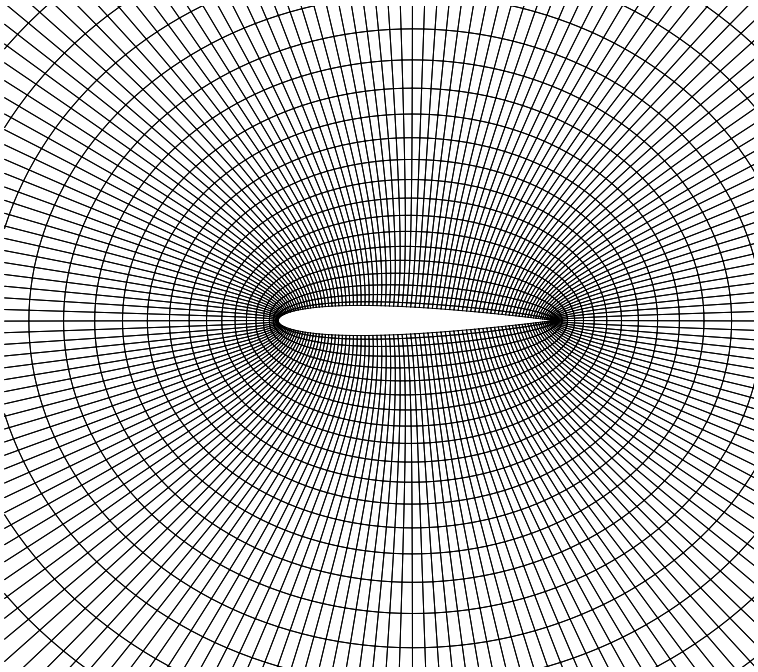
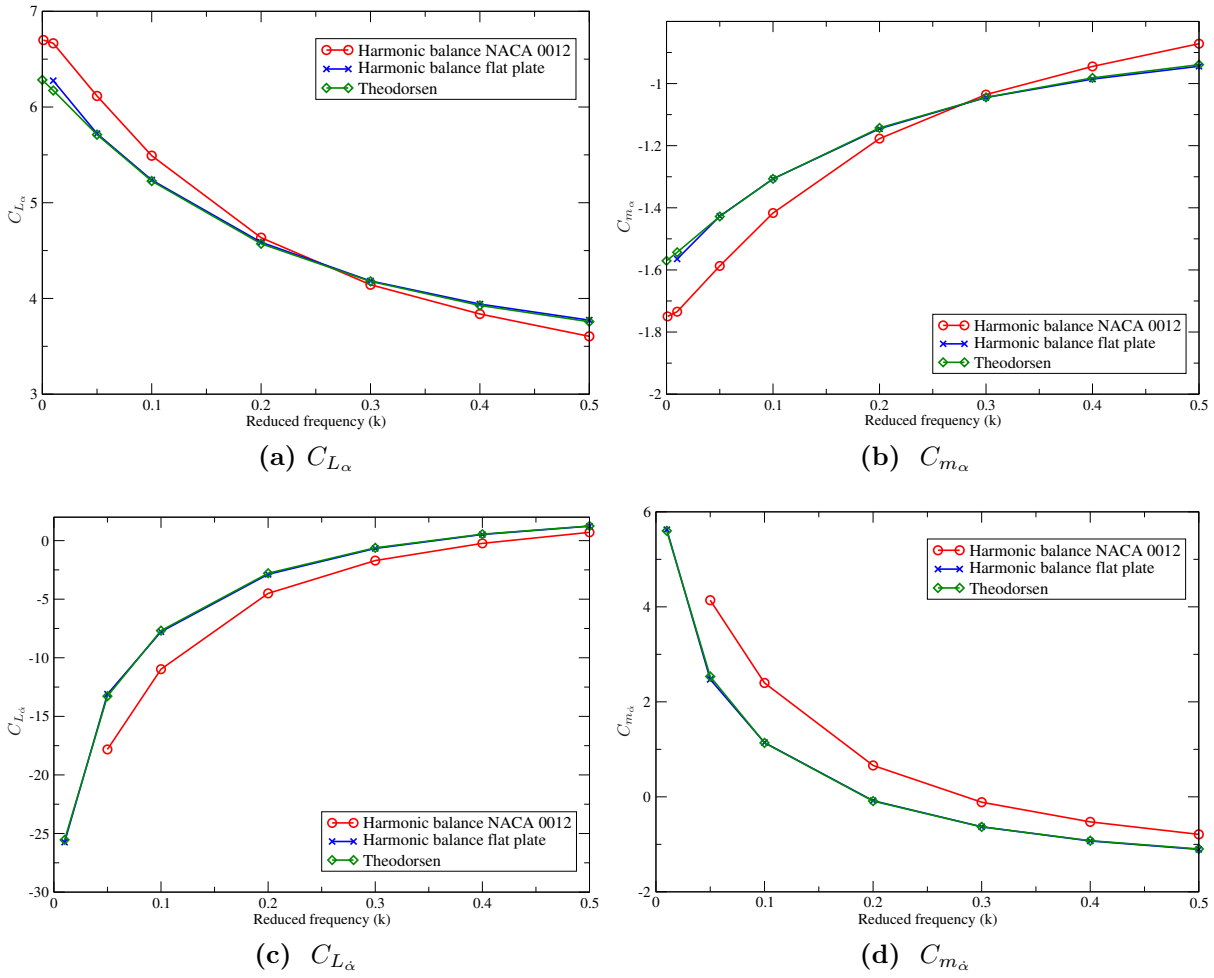


Figure 36: NACA 0012 inviscid grid



(a) C_{L_α} **(b)** C_{m_α}

(c) $C_{L_{\dot\alpha}}$ **(d)** $C_{m_{\dot\alpha}}$

Figure 37: NACA 0012 time-spectral stability derivative verification

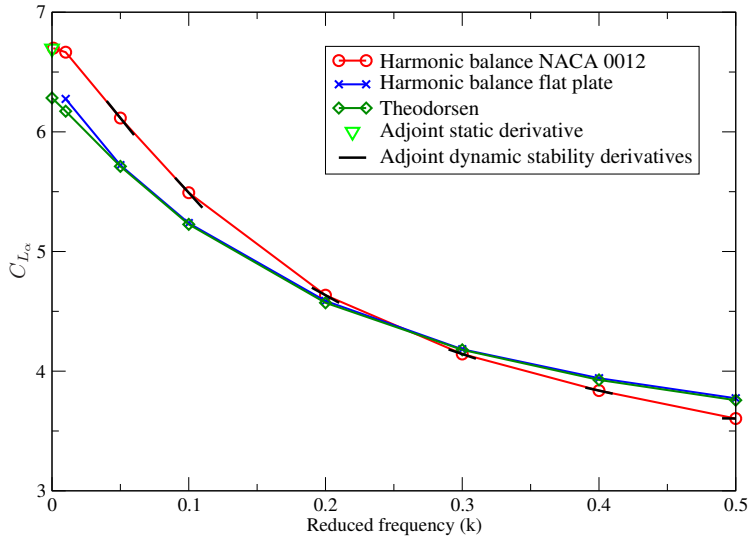


Figure 38: Comparison of adjoint $\mathrm{d}C_{L_\alpha}/\mathrm{d}k$ derivatives with time-spectral dynamic stability profile

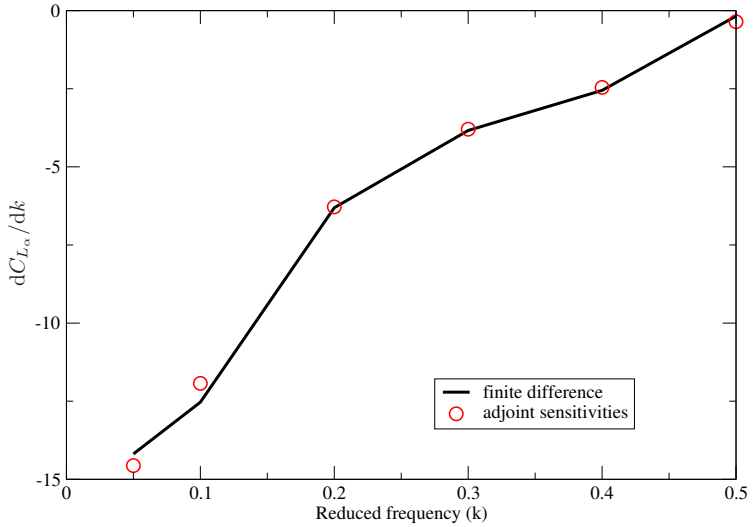


Figure 39: Comparison of $\mathrm{d}C_{L_\alpha}/\mathrm{d}k$ calculated via an adjoint approach and finite difference

Table 10: Snapshot collection parameters for the varied initial snapshot location cases

Case	Initial Snapshot	Snapshot Count	Snapshot Interval	Convergence Iteration	Percent Reduction (%)
Baseline	-	-	-	50,494	-
First snapshot at 10,000	10,000	401	10	33,762	33.1%
First snapshot at 11,000	11,000	401	10	31,924	36.8%
First snapshot at 12,000	12,000	401	10	32,031	36.6%
First snapshot at 13,000	13,000	401	10	32,881	34.9%
First snapshot at 14,000	14,000	401	10	34,016	32.6%

Table 11: Iteration reduction for increasing snapshot count

Case	First snapshot (Iteration)	Snapshot span (Iterations)	Converge (Iteration)	Reduction (%)	Time (Normalized)
Baseline	-	-	31175	-	1.0
11 snapshots	500	320	28790	7.65	0.935
21 snapshots	500	320	26361	15.44	0.859
41 snapshots	500	320	20230	35.11	0.663
81 snapshots	500	320	14656	52.99	0.499
161 snapshots	500	320	15595	49.98	0.550

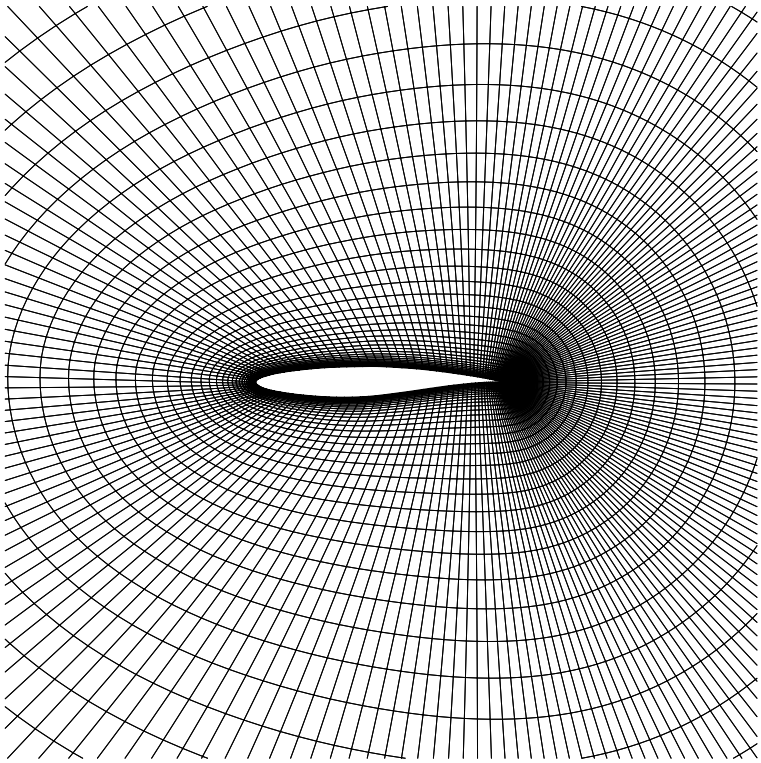


Figure 40: RAE 2822 viscous grid.

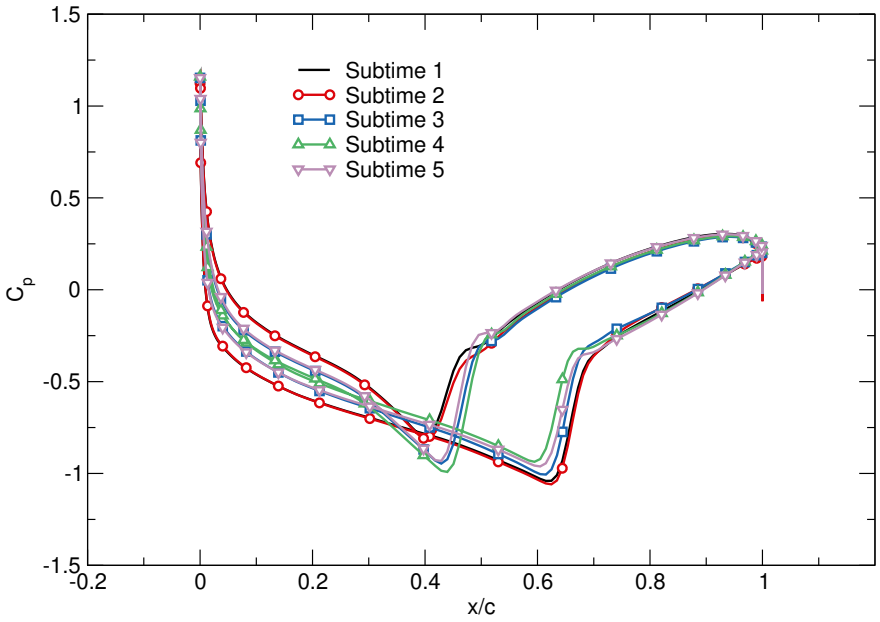


Figure 41: Surface pressure distributions for the five harmonic balance sub-time solutions.

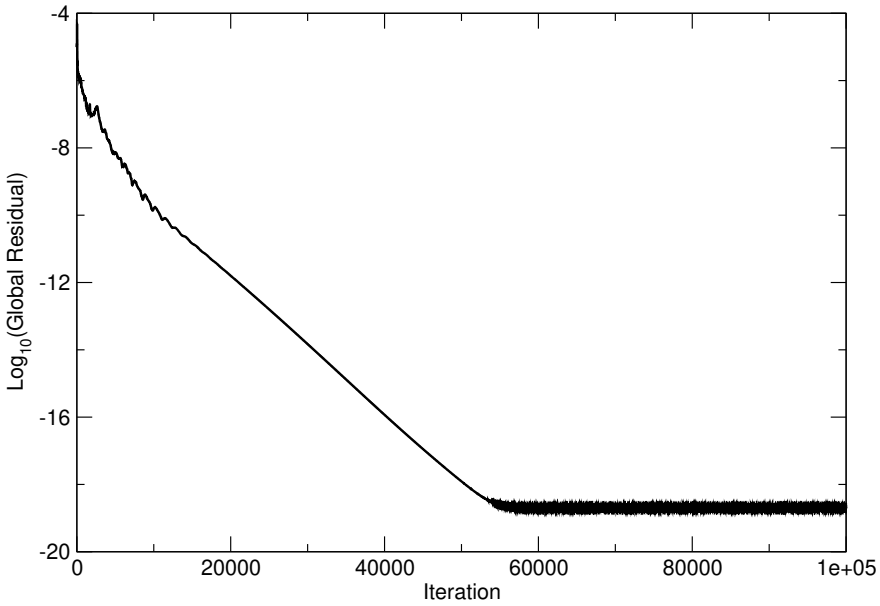


Figure 42: Unaccelerated harmonic balance primal solution convergence history.

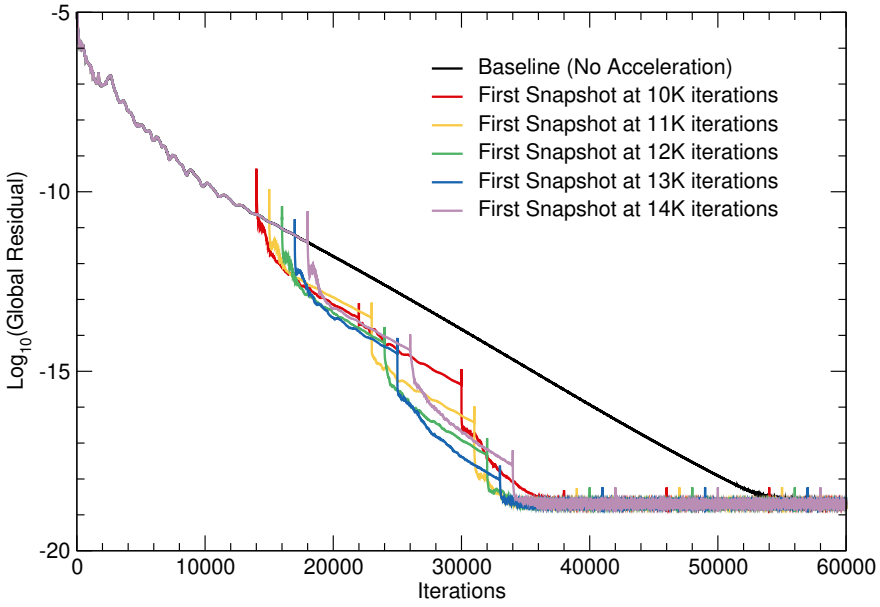
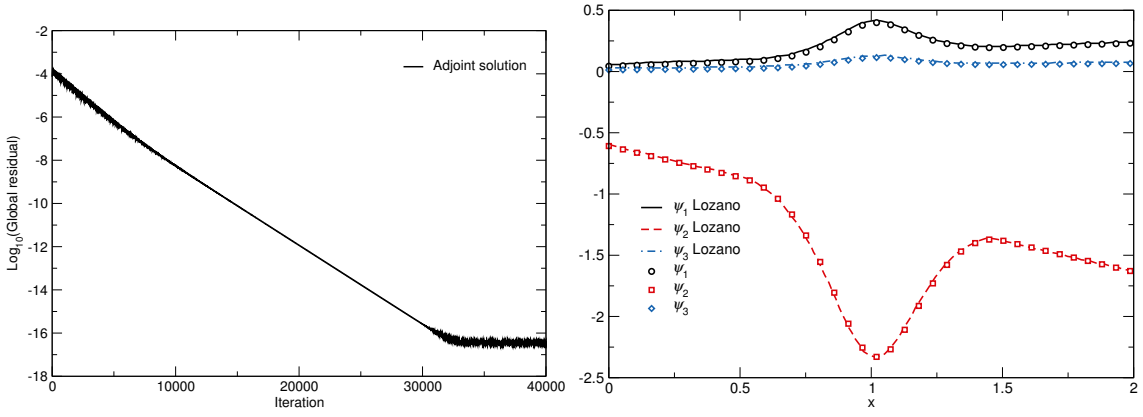


Figure 43: Comparison of the convergence history of the primal harmonic balance solver accelerated with the reduced-order model-base acceleration technique.



(a) **(b)**

Figure 44: Adjoint solver (a) convergence history and (b) comparison of the adjoint solution vector with that of Lozano and Ponsin [128].

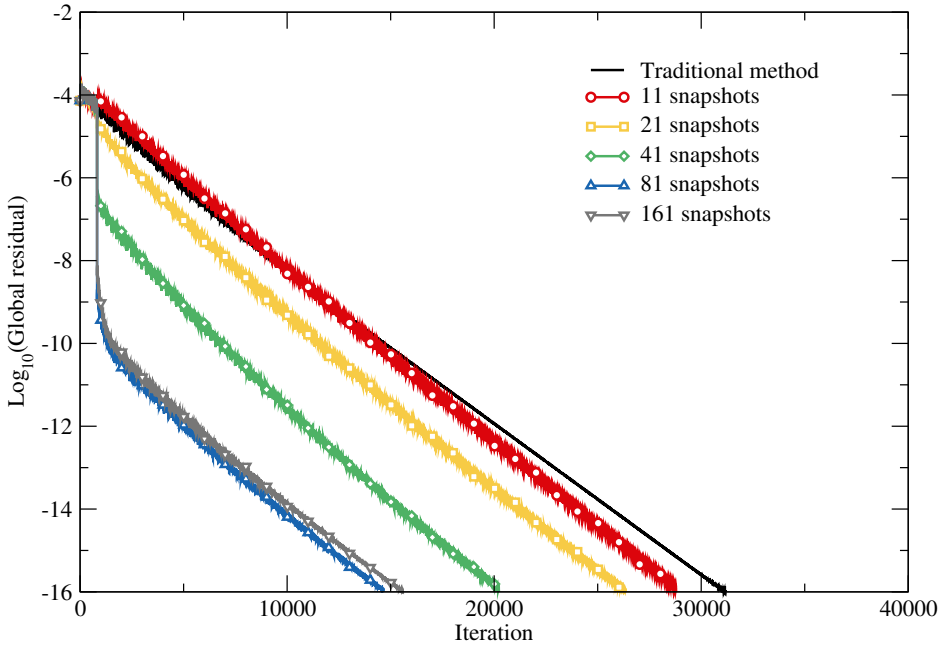
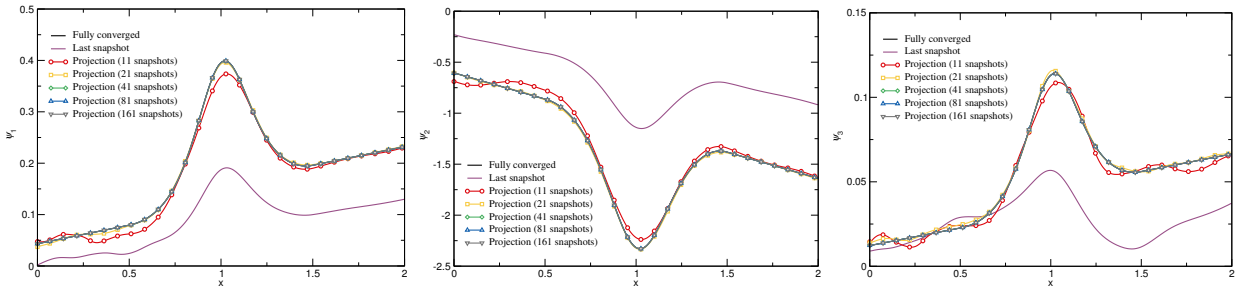
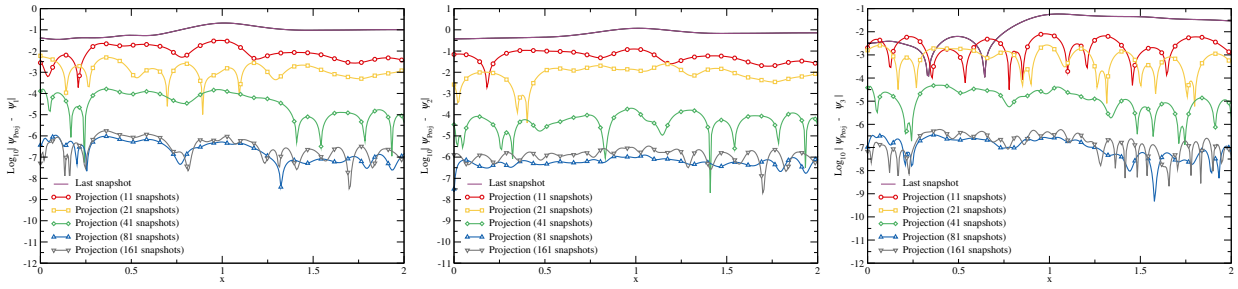


Figure 45: Convergence history achieved by varying snapshot count: increased snapshot quantities improve convergence acceleration



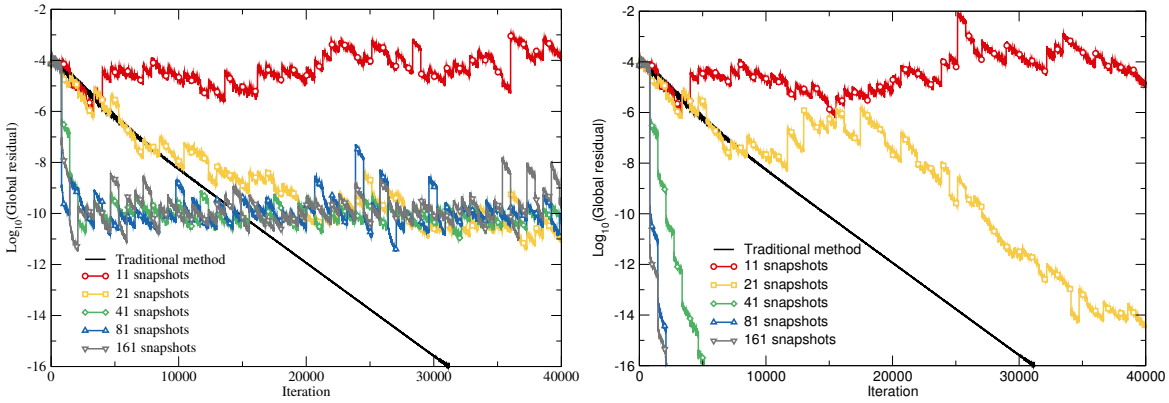
(a) **(b)** **(c)**

Figure 46: Projected adjoint vector profiles offer considerable improvement over the adjoint vector profile of the most converged snapshot.



(a) **(b)** **(c)**

Figure 47: The differences between the projected adjoint vector profiles and converged values show that the projections significantly improve the present solution.



(a) **(b)**

Figure 48: Multiple applications of the convergence acceleration technique (a) using double precision reaches an artificial convergence limit but (b) using quadruple precision eliminates this limit.

Table 12: Acceleration performance for increasing snapshot counts

Case	First snapshot (Iteration)	Interval (Iterations)	Convergence (Iteration)	Reduction (%)	Relative reduction (%)
Baseline	-	-	31175	-	-
11 snapshots	2	32	33144	-6.32	-
11 snapshots	500	32	28792	7.64	13.13
21 snapshots	2	16	27239	12.14	-
21 snapshots	500	16	26361	15.44	3.75
41 snapshots	2	8	26296	15.65	-
41 snapshots	500	8	20230	35.11	23.07
81 snapshots	2	4	27666	11.26	-
81 snapshots	500	4	10613	65.96	61.64
161 snapshots	2	2	25403	18.51	-
161 snapshots	500	2	7946	74.51	68.72

Table 13: Iteration reduction for increasing snapshot count with a single application

Case	First snapshot (Iteration)	Snapshot span (Iterations)	Converge (Iteration)	Reduction (%)	Time (Normalized)
Baseline	-	-	11267	-	1.0
Multigrid (1 level)	-	-	4755	57.80	0.674
Multigrid (2 levels)	-	-	3967	64.80	0.756
11 snapshots	1500	640	10971	2.63	0.991
21 snapshots	1500	640	9514	15.56	0.883
41 snapshots	1500	640	9368	16.85	0.893
81 snapshots	1500	640	5899	47.65	0.659
161 snapshots	1500	640	6236	44.65	0.788

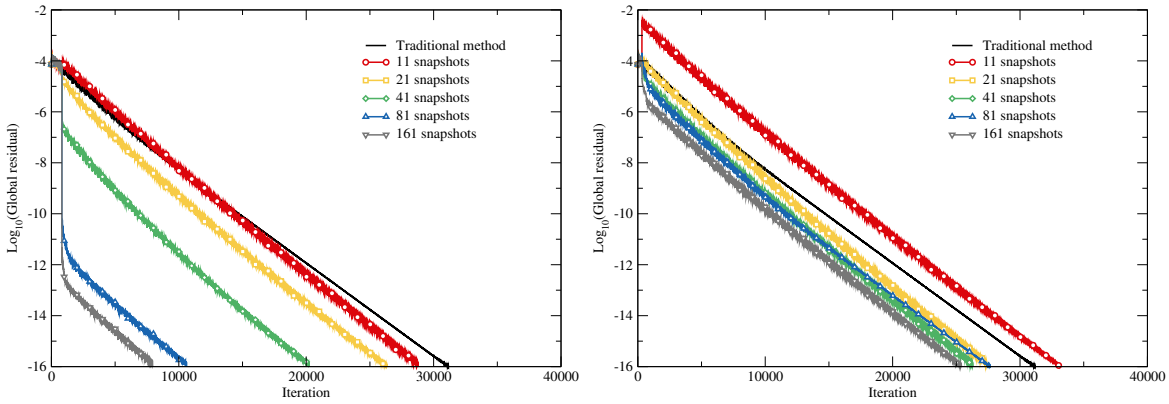


Figure 49: Convergence acceleration using orthogonal basis vectors with snapshot collection beginning at (a) iteration 500 and (b) iteration 2.

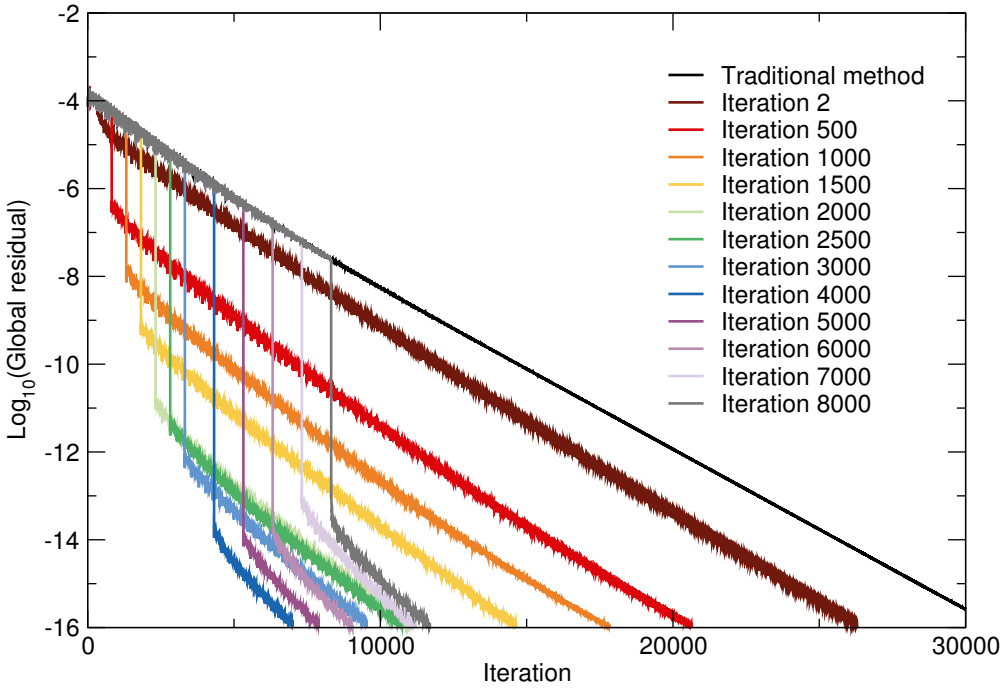


Figure 50: Delaying the first iteration location for the 41 snapshot case improves projections until a gradual plateau is reached.

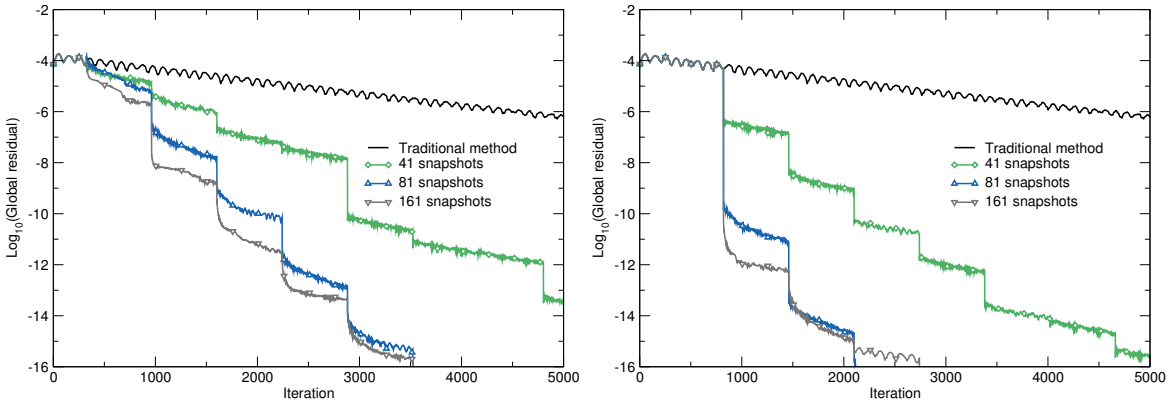


Figure 51: Convergence acceleration using multiple applications of the POD-based correlation technique with snapshot collection beginning at (a) iteration 2 and (b) iteration 500.

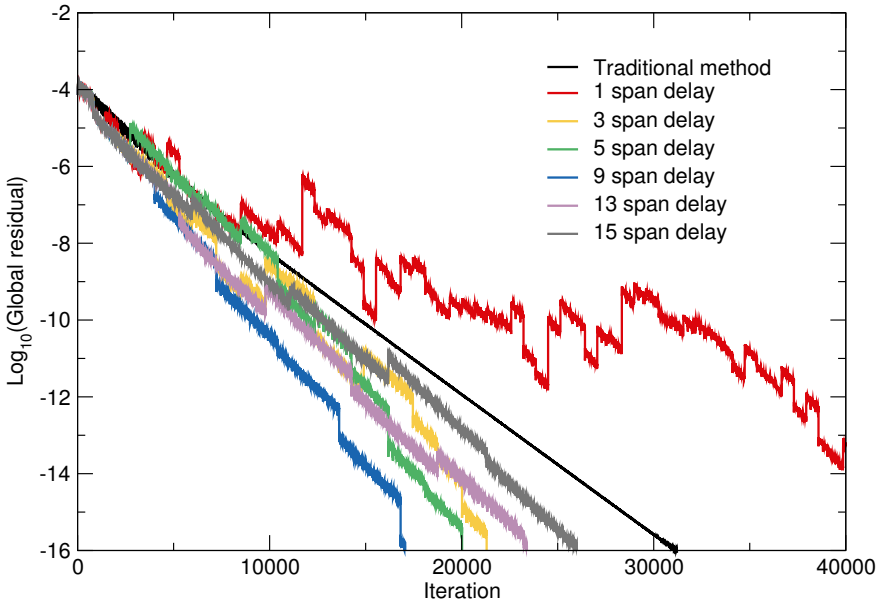


Figure 52: Convergence acceleration using multiple applications of the POD-based correlation technique stabilized by increasing the delay between acceleration cycles for 21 snapshots

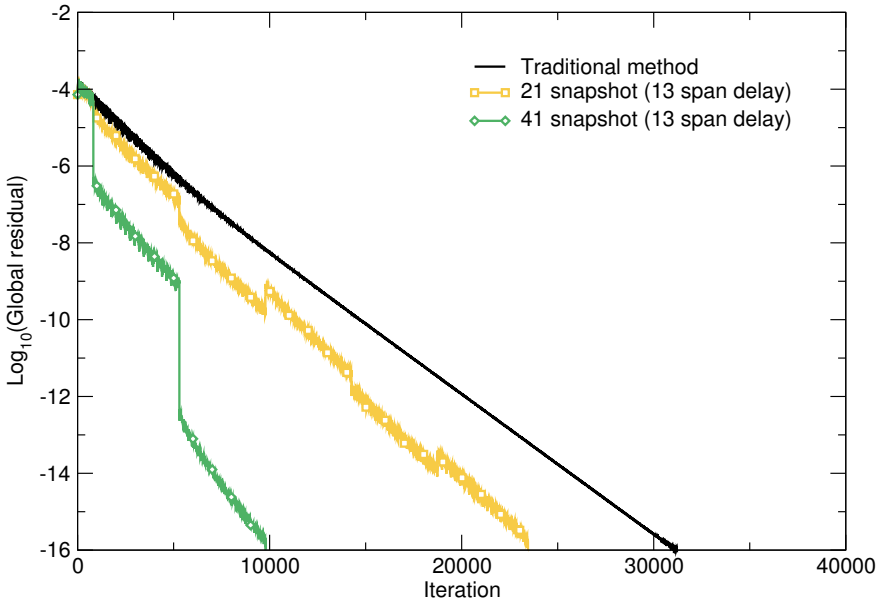


Figure 53: Convergence acceleration using multiple applications of the POD-based acceleration technique delayed 13 spans.

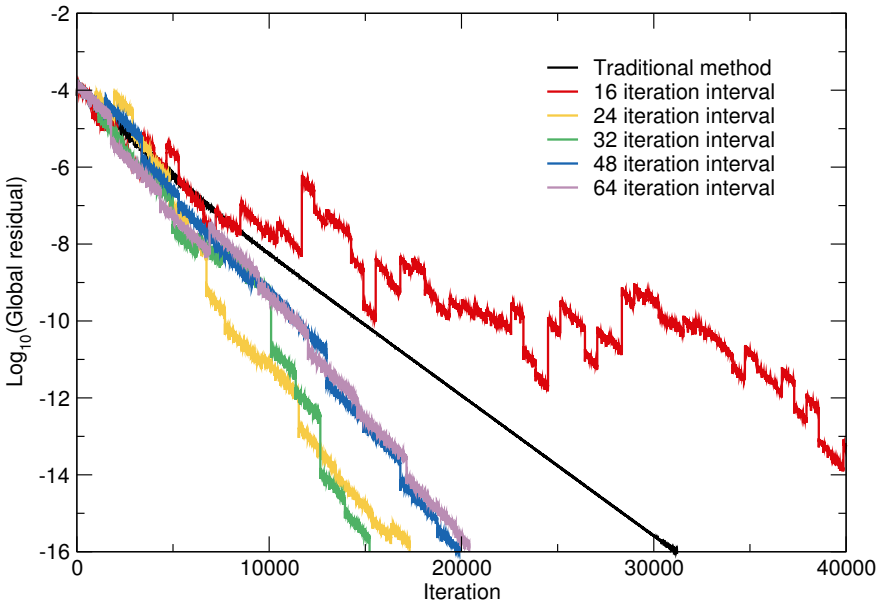


Figure 54: Convergence acceleration using multiple applications of the POD-based correlation technique with 21 snapshots varying the iteration interval between snapshots.

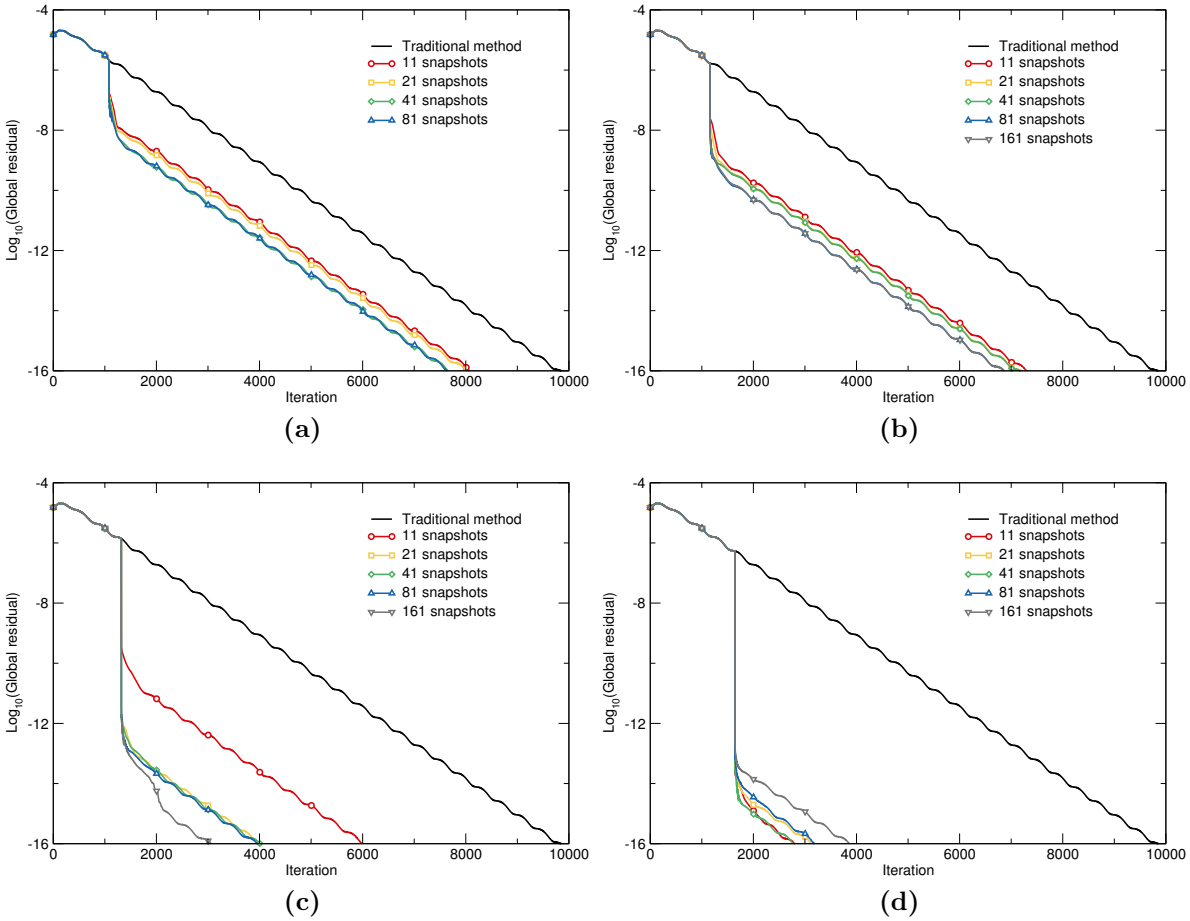


Figure 55: Convergence acceleration using orthogonal basis vectors with snapshot spans of (a) 80 iterations, (b) 160 iterations, (c) 320 iterations, and (d) 640 iterations

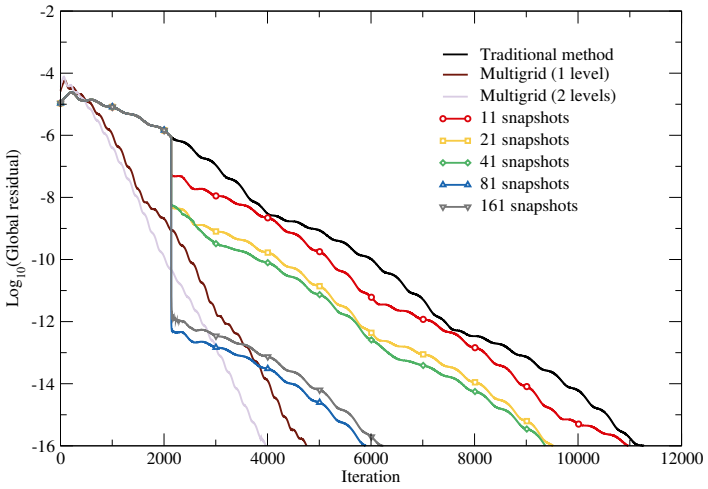


Figure 56: Convergence acceleration using orthogonal basis vectors applied once.

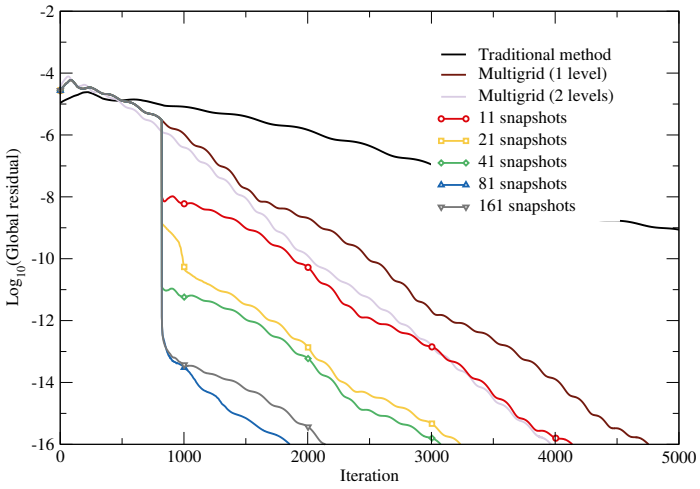


Figure 57: Convergence acceleration using orthogonal basis vectors applied once to the multigrid scheme

Table 14: Iteration reduction for increasing snapshot count with a single application applied to the 1 level multigrid scheme

Case	First snapshot (Iteration)	Snapshot span (Iterations)	Converge (Iteration)	Reduction (%)	Time (Normalized)
Baseline	-	-	11267	-	1.0
Multigrid (1 level)	-	-	4755	57.80	0.674
Multigrid (2 levels)	-	-	3967	64.80	0.756
11 snapshots	500	320	4139	63.26	0.588
21 snapshots	500	320	3235	71.29	0.476
41 snapshots	500	320	3075	72.71	0.484
81 snapshots	500	320	1856	83.53	0.366
161 snapshots	500	320	2143	80.98	0.502

Table 15: Iteration reduction for increasing snapshot count with multiple applications applied to the 1 level multigrid scheme

Case	First snapshot (Iteration)	Snapshot span (Iterations)	Converge (Iteration)	Reduction (%)	Time (Normalized)
Baseline	-	-	11267	-	1.0
Multigrid (1 level)	-	-	4755	57.80	0.674
Multigrid (2 levels)	-	-	3967	64.80	0.756
11 snapshots	500	320	2778	75.34	0.454
21 snapshots	500	320	2183	80.62	0.379
41 snapshots	500	320	2342	79.21	0.466
81 snapshots	500	320	1766	84.33	0.465
161 snapshots	500	320	2172	80.72	0.878

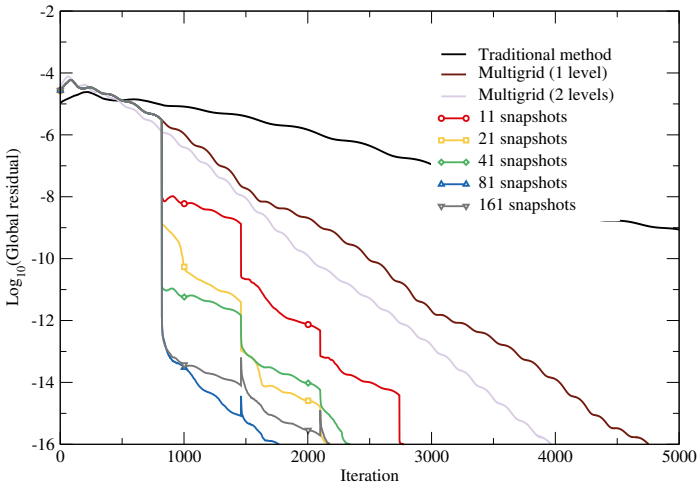


Figure 58: Convergence acceleration using orthogonal basis vectors applied multiple times to the multigrid scheme

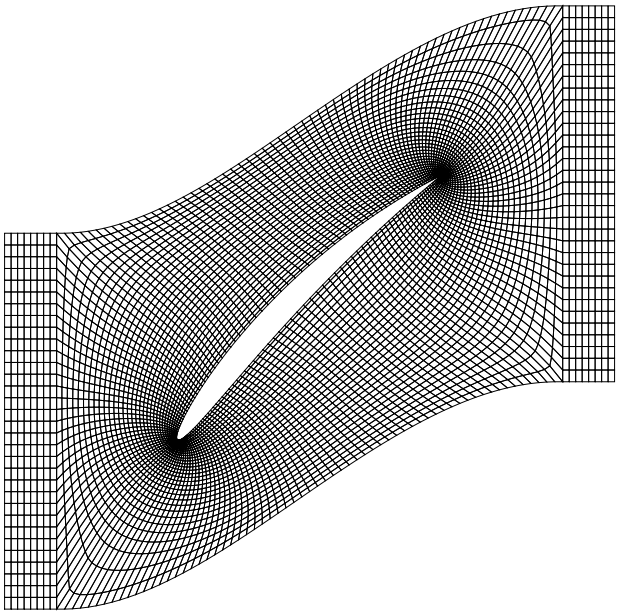


Figure 59: Perturbed 10th standard inviscid HOH grid

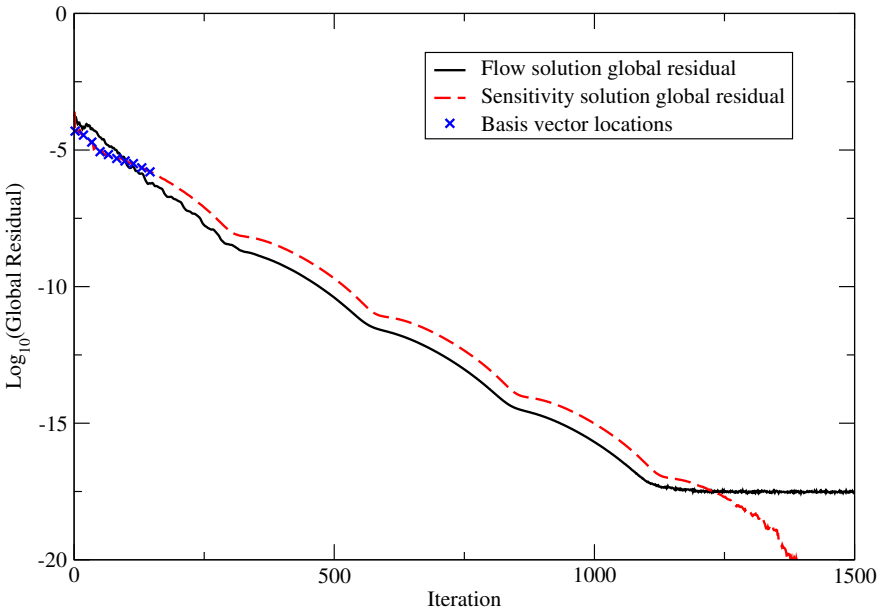


Figure 60: Convergence of global sensitivity residual of first design cycle.

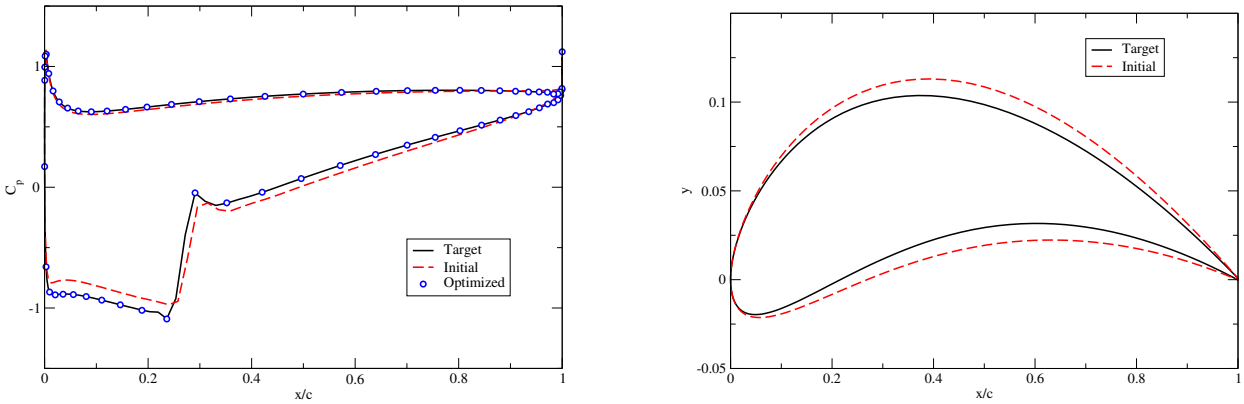


Figure 61: The initial, target and optimized (a) surface pressure distribution and (b) blade shape

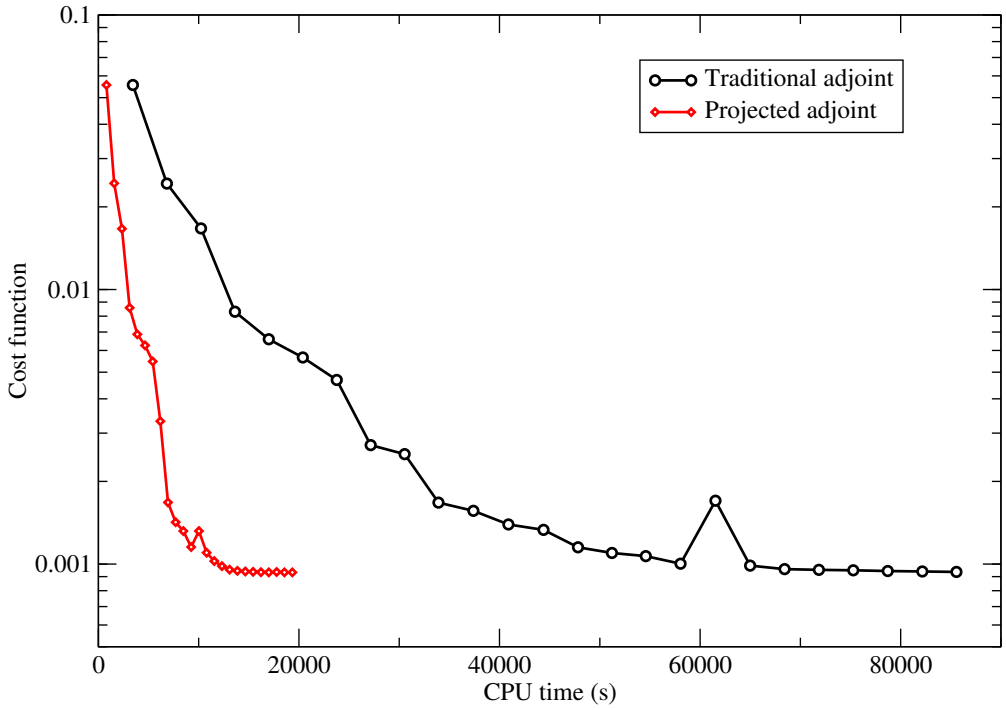


Figure 62: Comparison of cost function reduction versus CPU time for the traditional and projected adjoint sensitivity methods.

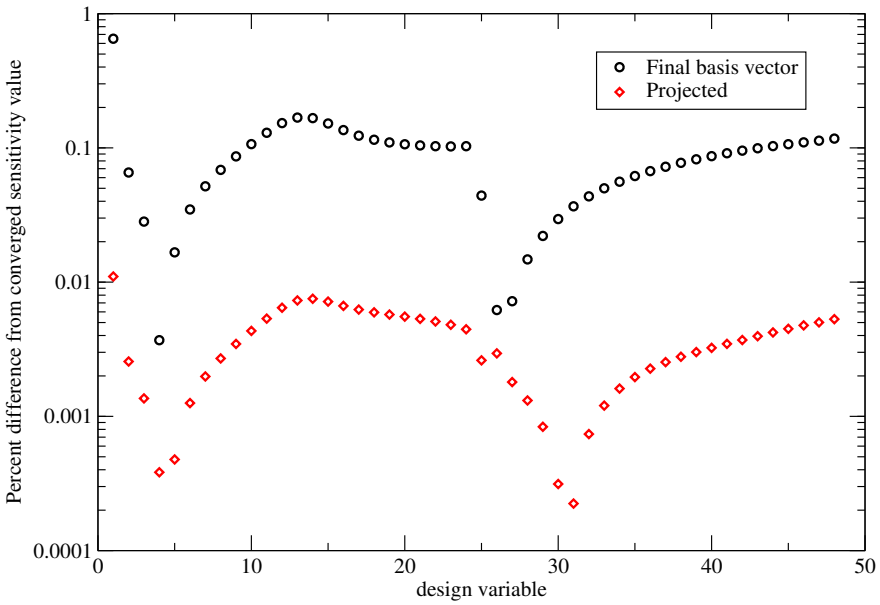


Figure 63: Percent difference from converged sensitivity values for each design variable at initial design point

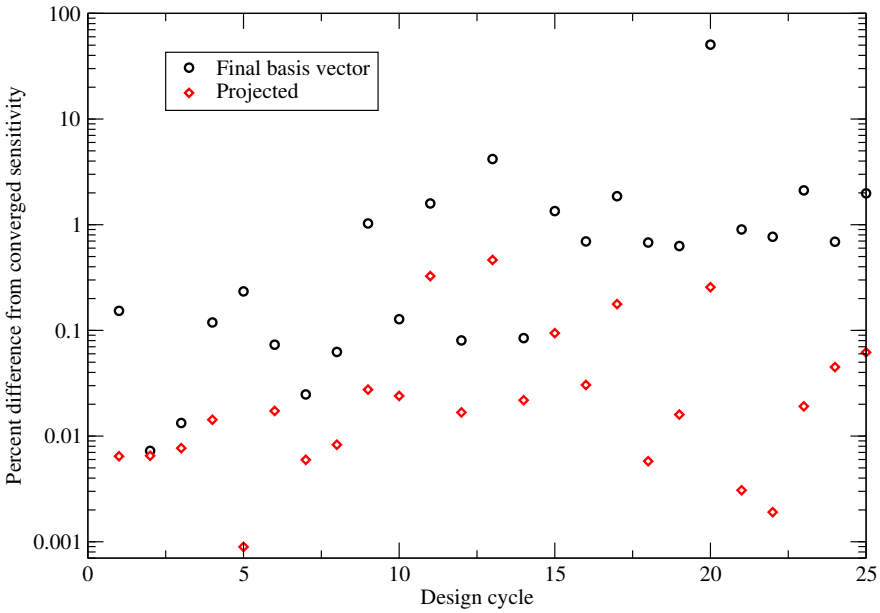


Figure 64: Percent difference from converged sensitivity values for each design cycle of design variable 12

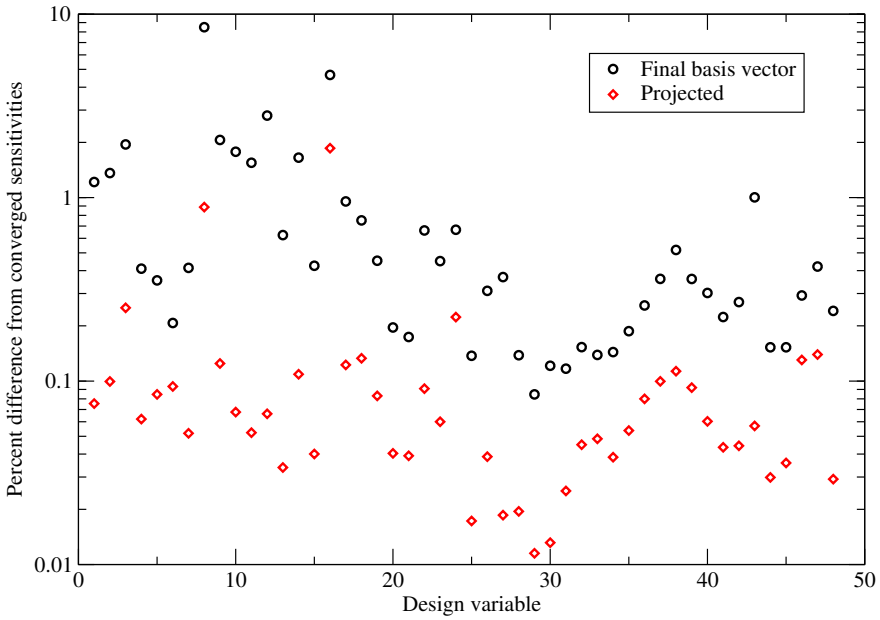


Figure 65: Average percent difference from converged sensitivity values for each design cycle over all design cycles

Table 16: Snapshot collection parameters for the varied snapshot cases.

Case	Snapshot count	Snapshot interval	First snapshot
201 Snapshot case	201	2	1000
101 Snapshot case	101	4	1000
51 Snapshot case	51	8	1000
26 Snapshot case	26	16	1000

Table 17: Snapshot collection parameters for the varied initial snapshot location cases

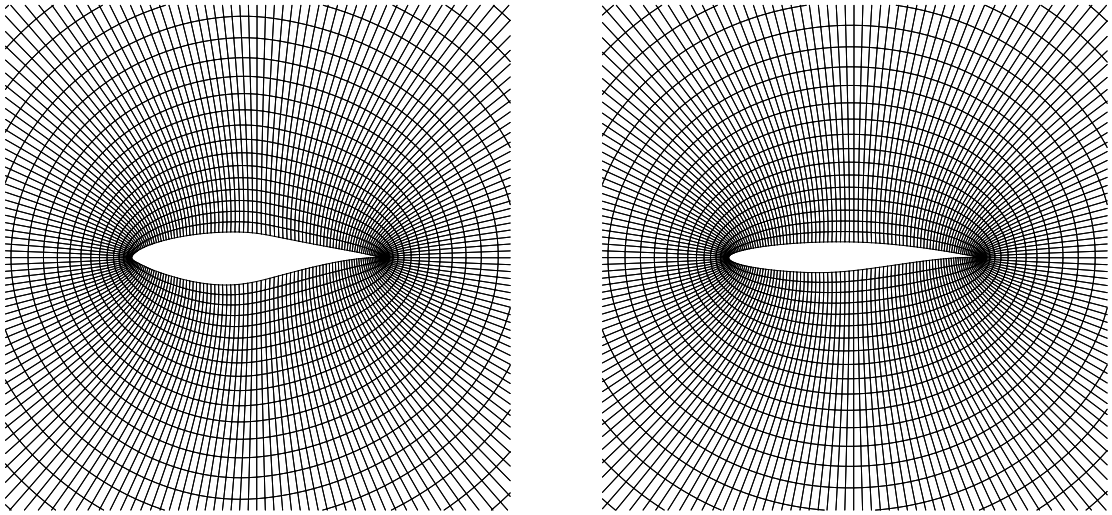
Case	Snapshot count	Snapshot interval	First snapshot
Started at 500	101	4	500
Started at 1000	101	4	1000
Started at 1500	101	4	1500
Started at 2000	101	4	2000

Table 18: Computation cost comparison of each sensitivity approach to reach $\|\bar{\boldsymbol{R}}(\bar{\boldsymbol{U}})\|_2 = 1 \times 10^{-15}$

Sensitivity Calculation Approach	Iterations	Normalized CPU Time
Brute force adjoint	46114	1.00
PTS adjoint	46114	0.81
PTS adjoint with POD acceleration	8810	0.20
PTS adjoint with Covariance acceleration	7007	0.17

Table 19: Comparison of loading coefficients for the NREL S809 airfoil.

Case	α	Re	C_L	C_D	C_M
Present Solver	0.0°	2.0×10^6	0.1335	0.0119	-0.0395
CFD-ACE [203]	0.0°	2.0×10^6	0.1324	0.0108	-0.0400
Experiment	0.0°	2.0×10^6	0.1469	0.0070	-0.0443



(a) **(b)**

Figure 66: Inviscid O-type computational grids for: (a) the NREL S809 airfoil (target), and (b) the RAE 2822 airfoil (initial). Both grids are made up of 201×70 nodes.

Table 20: Snapshot collection parameters for the viscous flow case.

Case	Snapshot count	Snapshot interval	First snapshot
Viscous ROM	201	10	4000

Table 21: Computational cost comparison of design optimization

Sensitivity Calculation Approach	CPU Time
PTS adjoint	1.00
PTS adjoint with Covariance acceleration	0.43

Table 22: Snapshot collection parameters for the nozzle one-shot optimization case with a single acceleration.

Case	Snapshot count	Snapshot interval	First snapshot
ROM-accelerated One-shot Solver	401	100	50000

Table 23: Snapshot collection parameters for the nozzle one-shot optimization case.

Case	Snapshot count	Snapshot interval	First snapshot
ROM-accelerated One-shot Solver	401	200	40000

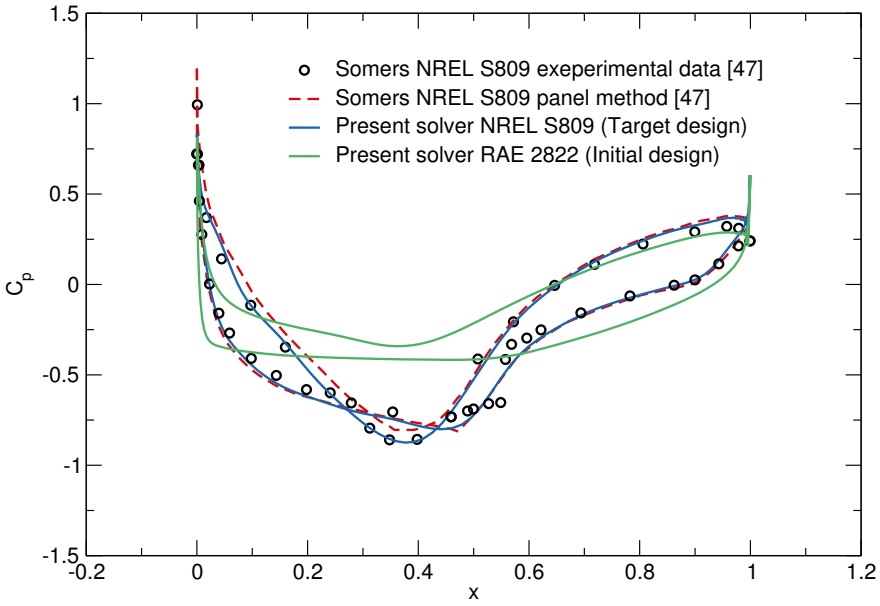


Figure 67: Comparison of surface pressure distributions for the NREL S809 and RAE 2822 airfoils.

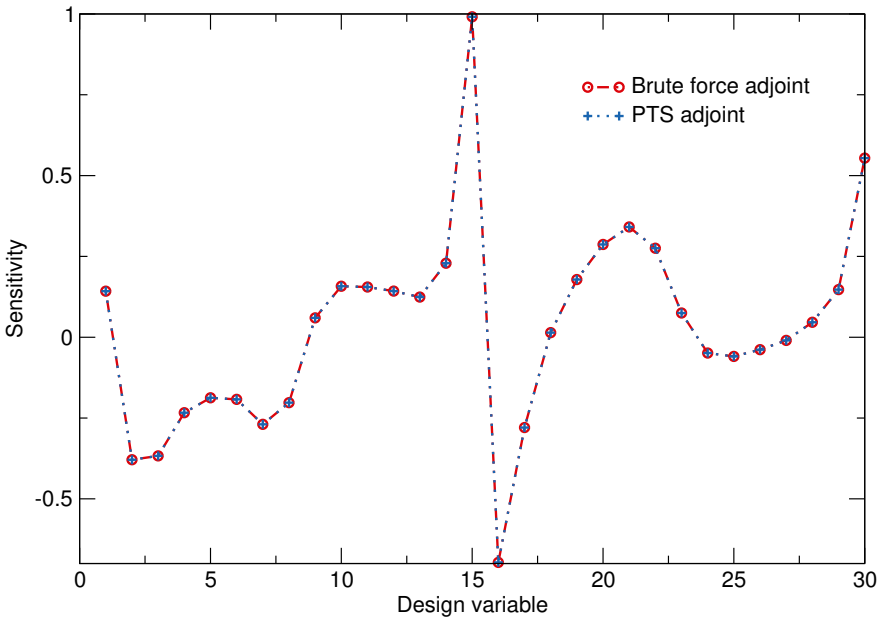


Figure 68: Comparison of the sensitivity values calculated via the brute force and primal time-stepping adjoint approaches.

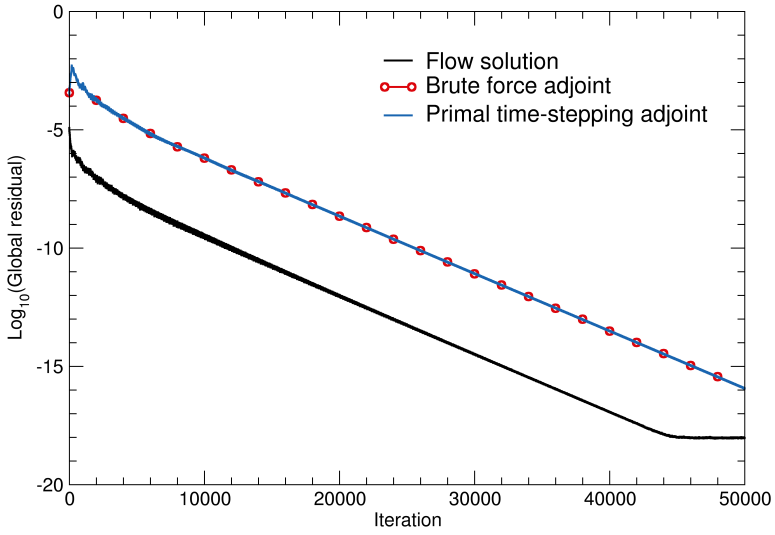
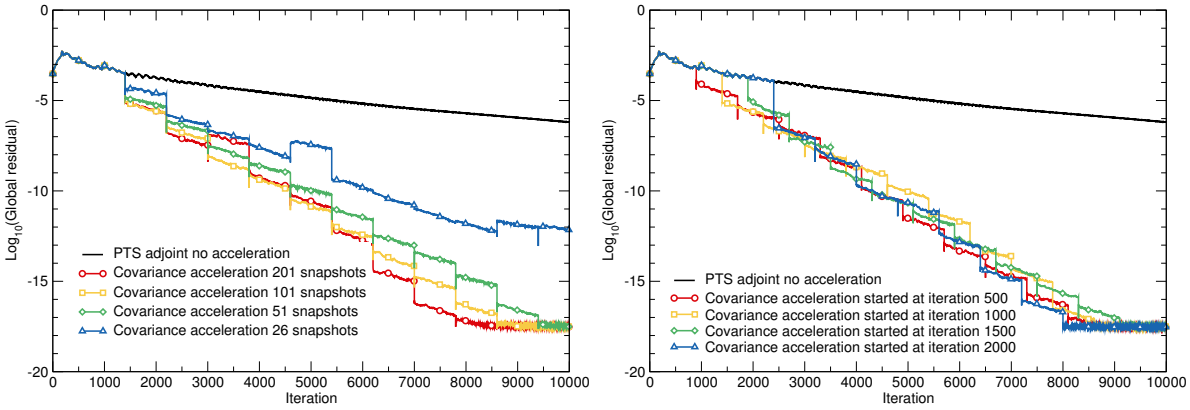


Figure 69: Convergence history of the brute force and primal time-stepping adjoint.



(a) **(b)**

Figure 70: Comparison of the convergence history of the primal time-stepping adjoint accelerated with reduced-order models built using: (a) different snapshot quantities, and (b) varied initial snapshot collection iterations.

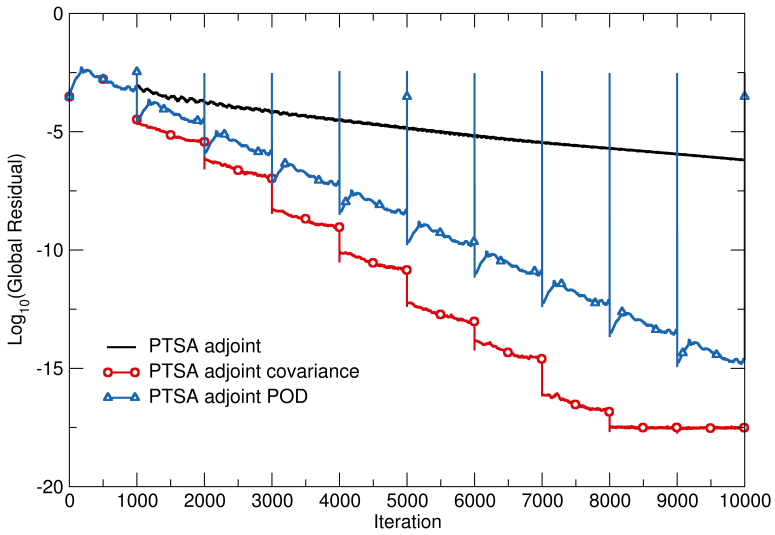


Figure 71: Convergence history of the traditional and accelerated primal time-stepping adjoint.

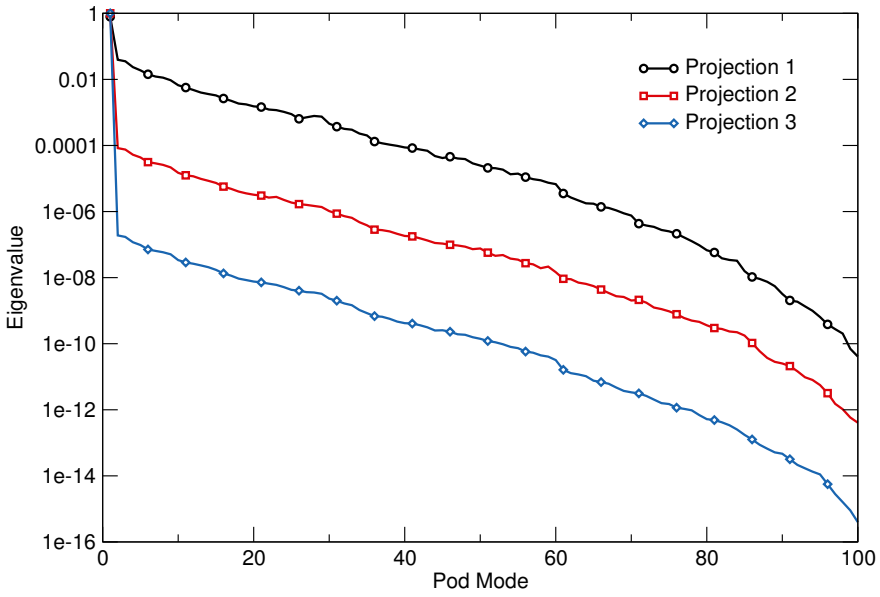
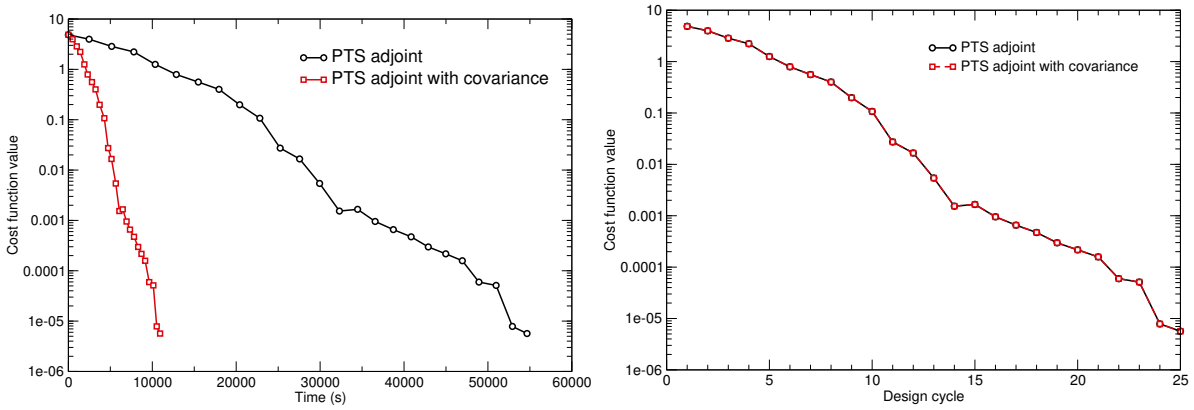


Figure 72: Eigenvalue distribution of POD basis for the first three acceleration projections.



(a) **(b)**

Figure 73: Cost function history of the L-BFGS optimizer with sensitivities calculated via the PTS adjoint approach with and without the ROM acceleration over (a) CPU time and (b) design cycle.

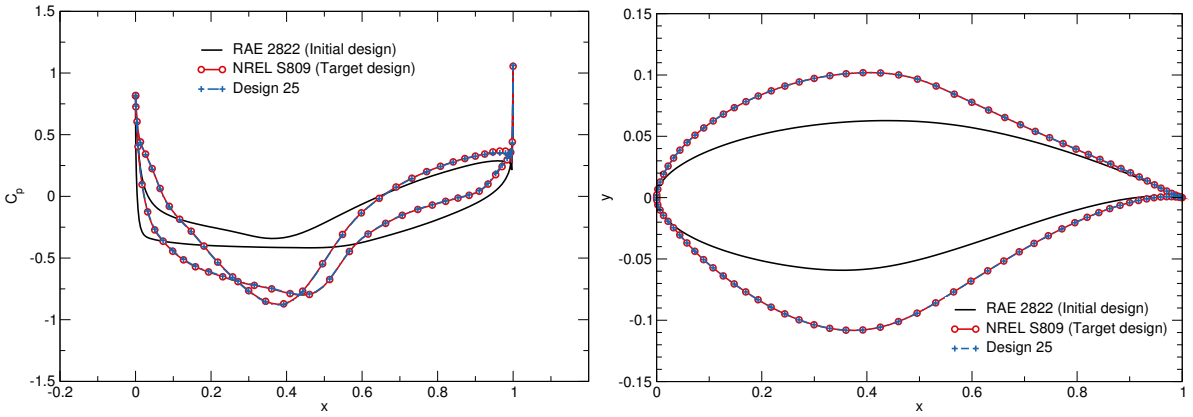


Figure 74: Comparison of the initial, target, and optimized (a) surface pressure profiles. and (b) airfoil surfaces.

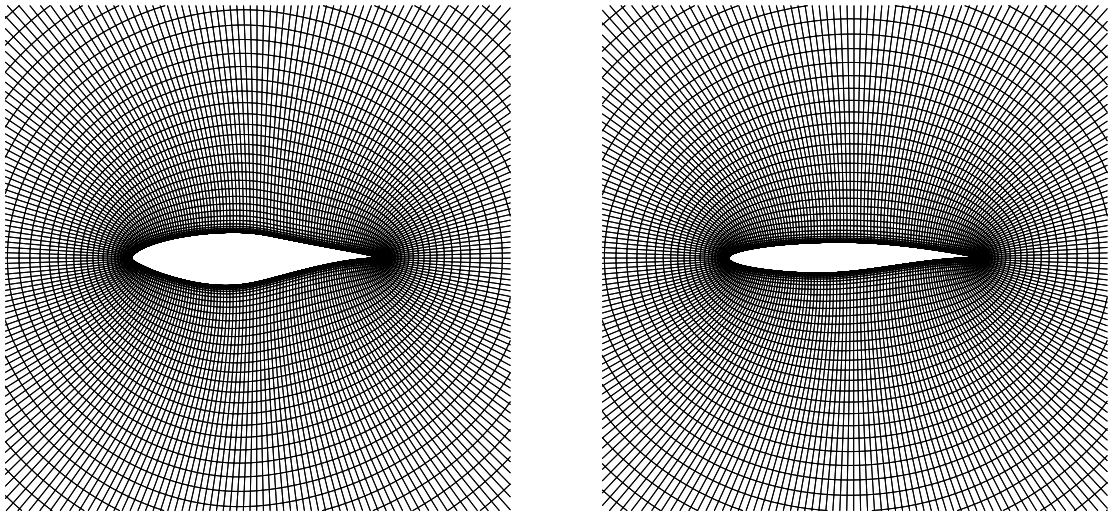


Figure 75: Viscous grids for the (a) NREL S809 (target design) and (b) RAE 2822 (initial design) airfoils. Both grids have 257×127 nodes in the streamwise and normal directions, respectively.

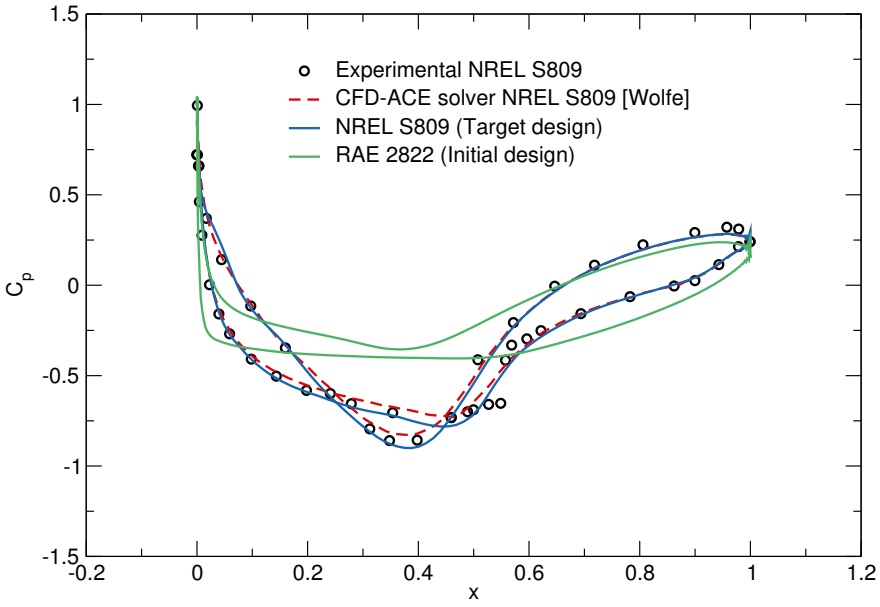


Figure 76: A comparison of the surface pressure distribution of the NREL S809 airfoil.

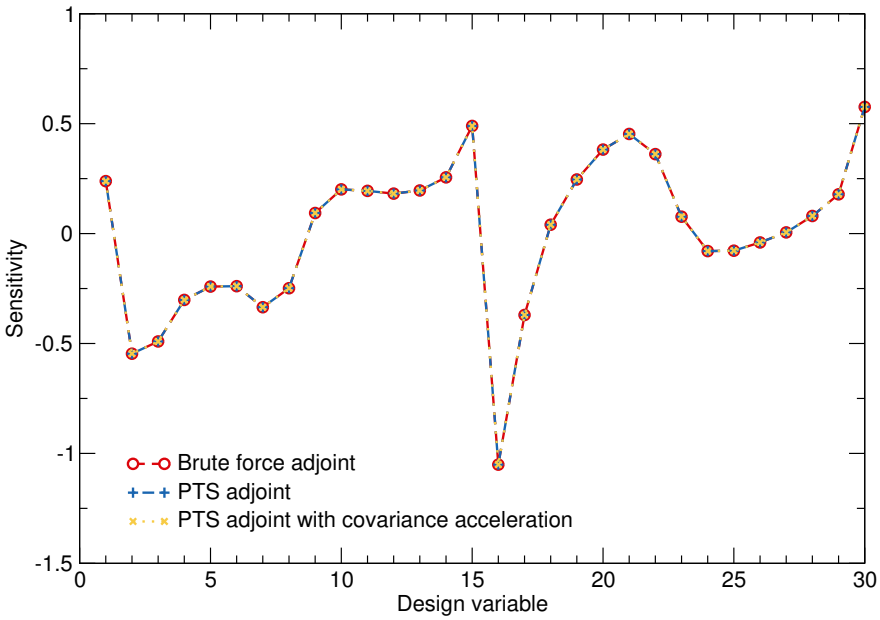


Figure 77: The adjoint sensitivity values of the b-spline control points at the initial design. Design variables 1-15 modify the top surface and 16-30 control the bottom surface.

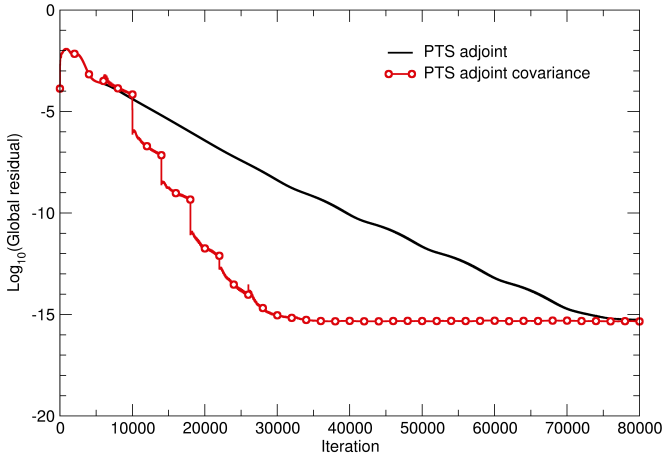


Figure 78: Convergence history of the primal time-stepping adjoint sensitivity approach for the viscous flow case with and without the ROM acceleration.

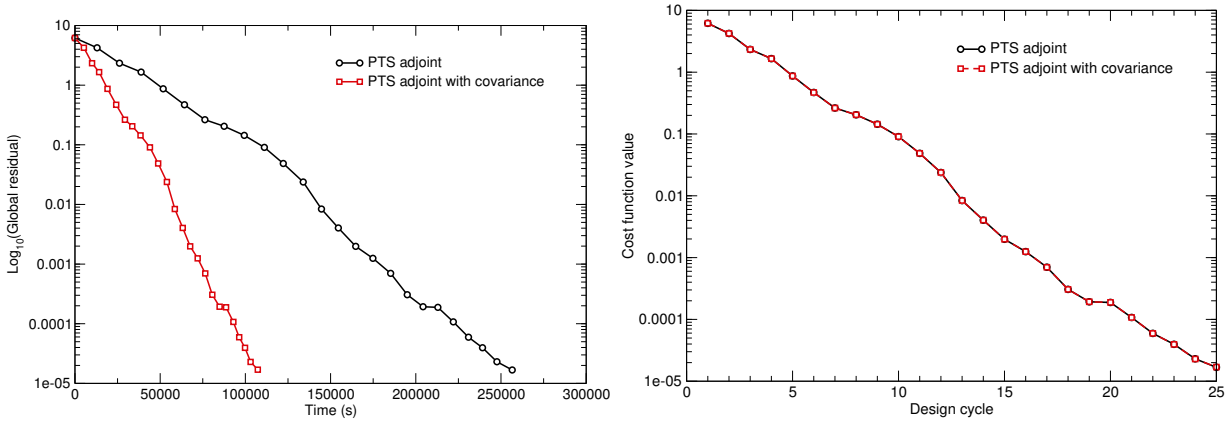


Figure 79: History of the cost function for L-BFGS-B optimizer using PTS adjoints calculated with and without acceleration over (a) CPU time and (b) design cycle.

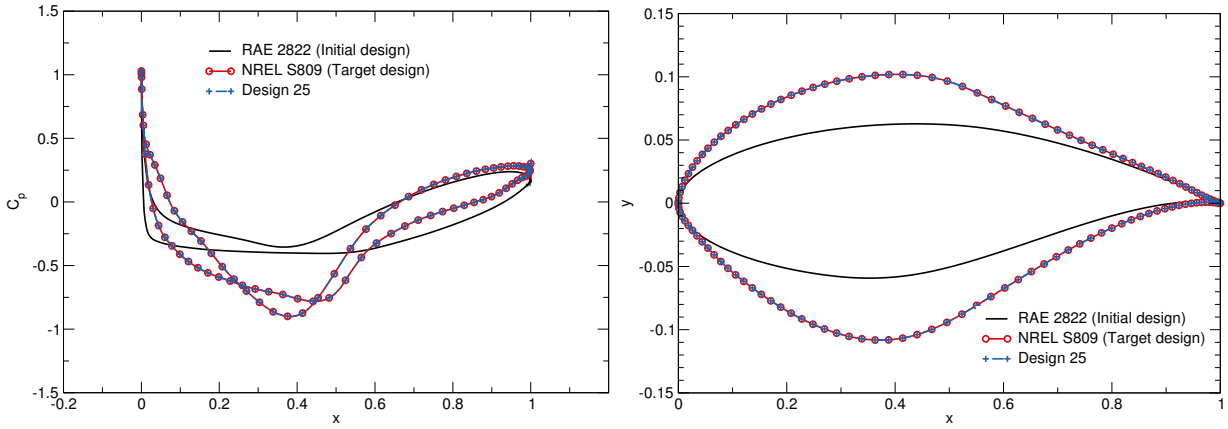


Figure 80: Comparison of the initial, target, and optimized (a) airfoil surfaces and (b) surface pressure profiles.

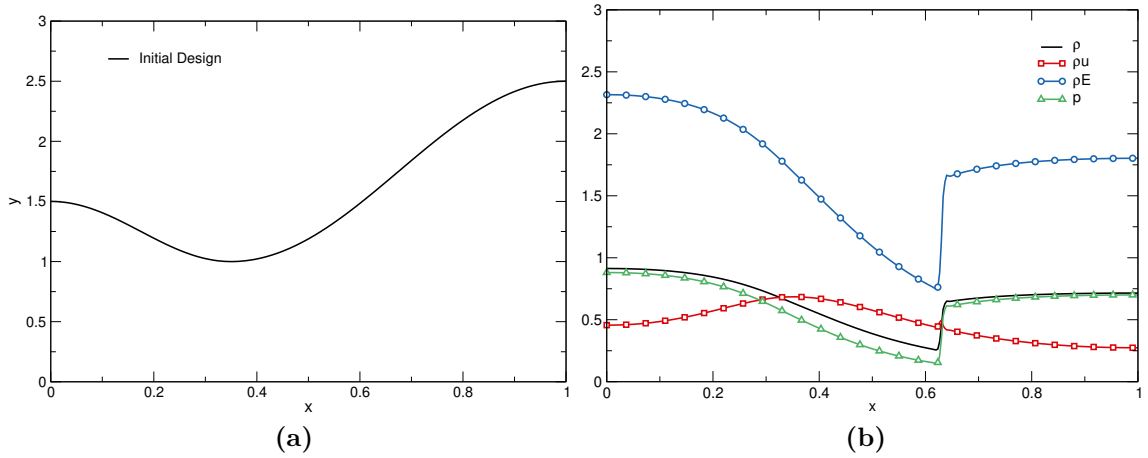


Figure 81: Inverse design problem (a) nozzle cross-section area and (b) flow solution of initial design.

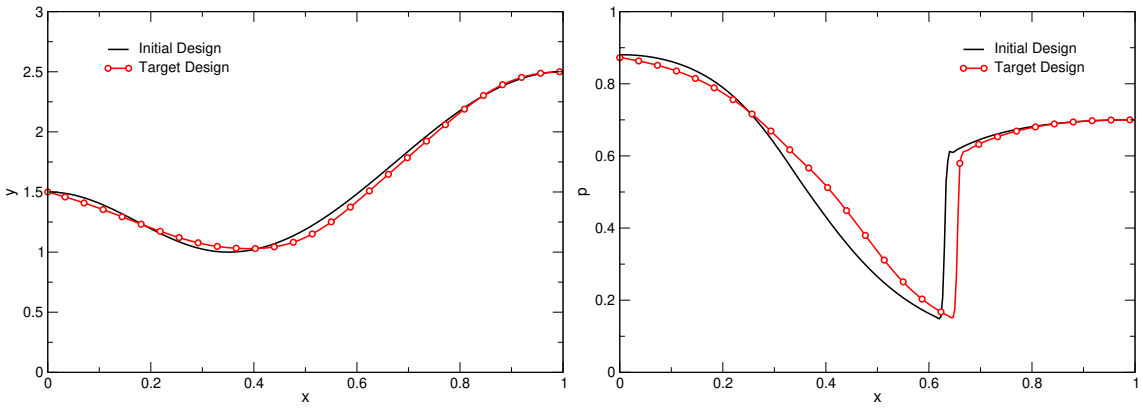


Figure 82: Comparison of the inverse design problem initial and target (a)nozzle cross-section area and (b) pressure profile.

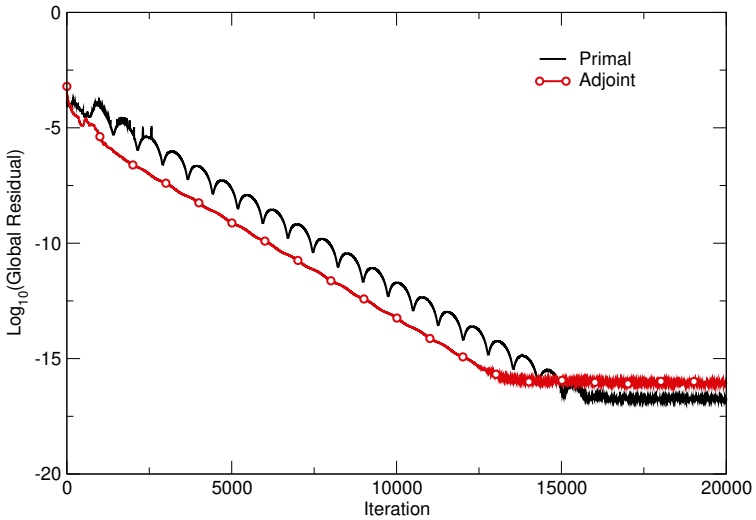


Figure 83: Convergence history of the primal, brute-force adjoint, and time-stepping adjoint sensitivity approach for the initial nozzle design.

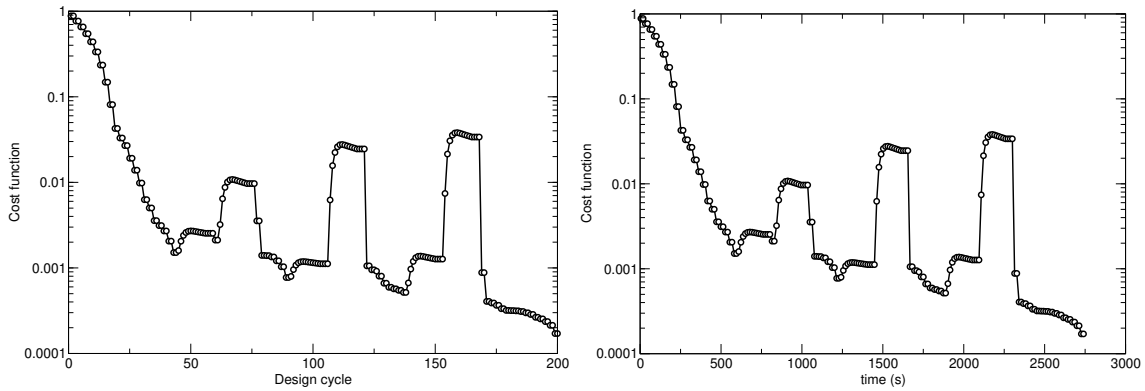


Figure 84: Cost function history over (a) each design cycle and (b) time.

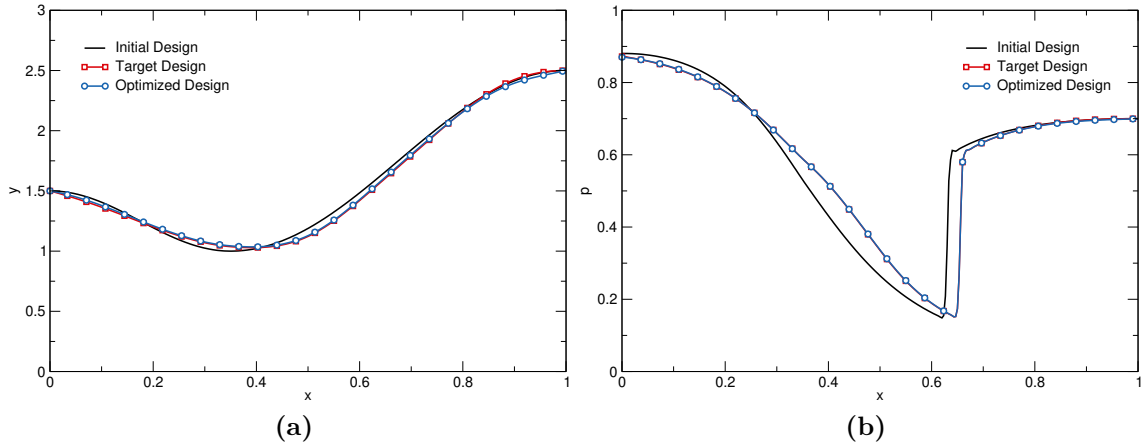


Figure 85: Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the L-BFGS-B optimizer.

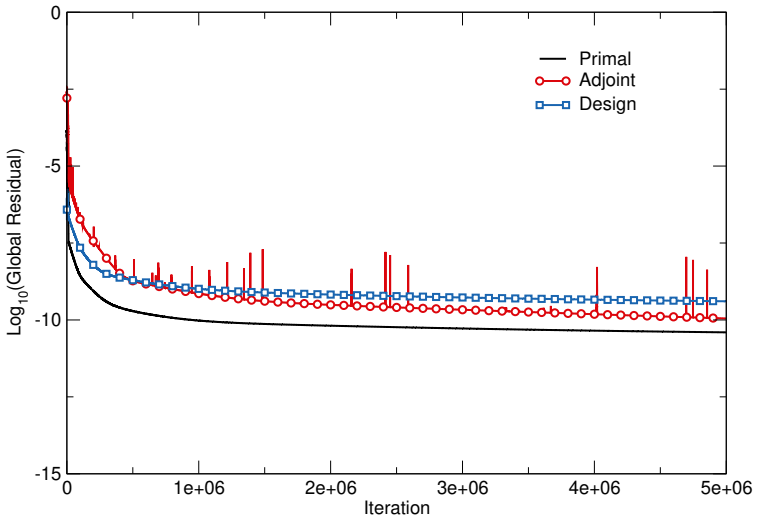
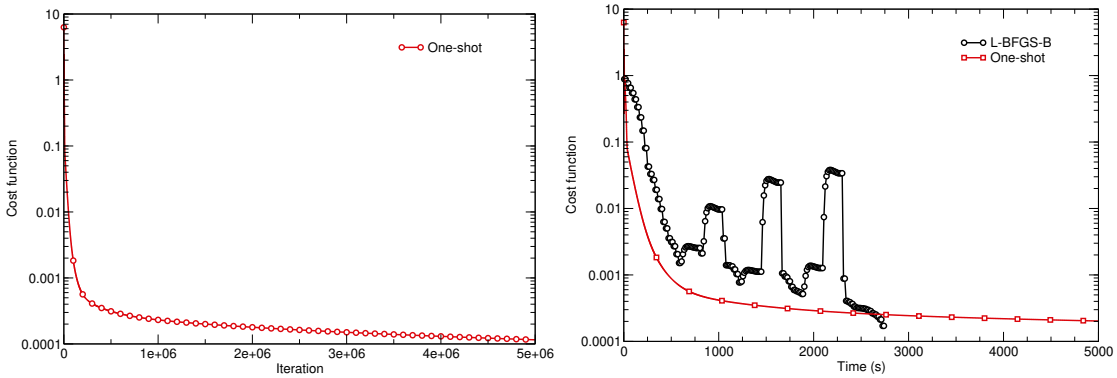


Figure 86: Convergence history of the primal, adjoint, and design equations solved via a one-shot optimization approach.



(a) **(b)**

Figure 87: Cost function history over (a) each design cycle and (b) time.

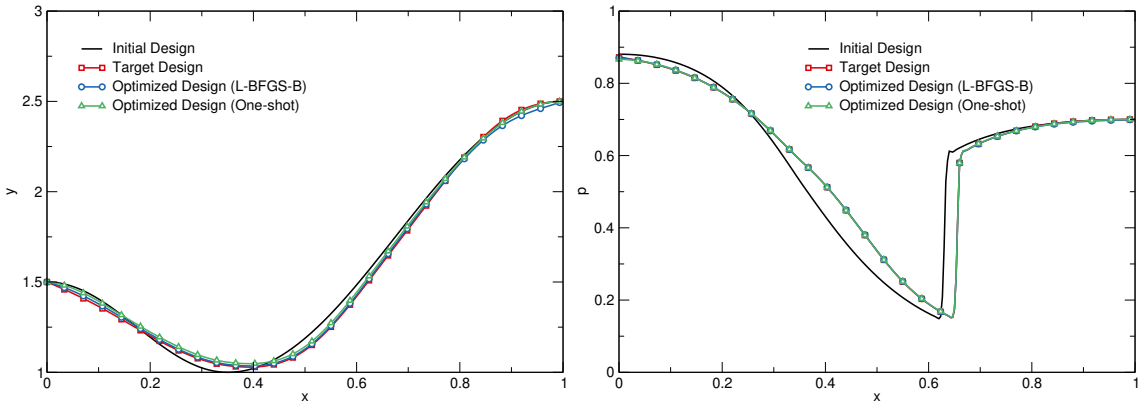


Figure 88: Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the one-shot optimizer.

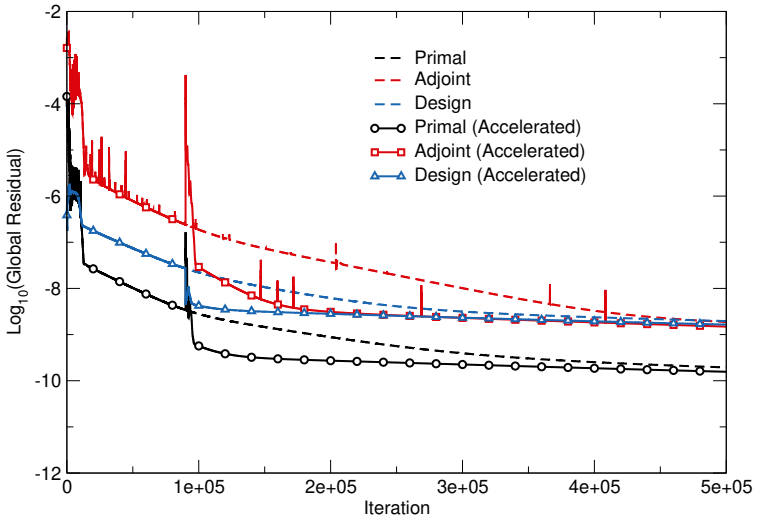


Figure 89: Convergence history of the primal, adjoint, and design equations solved via a one-shot optimizer with a single application of the reduced-order model acceleration technique.

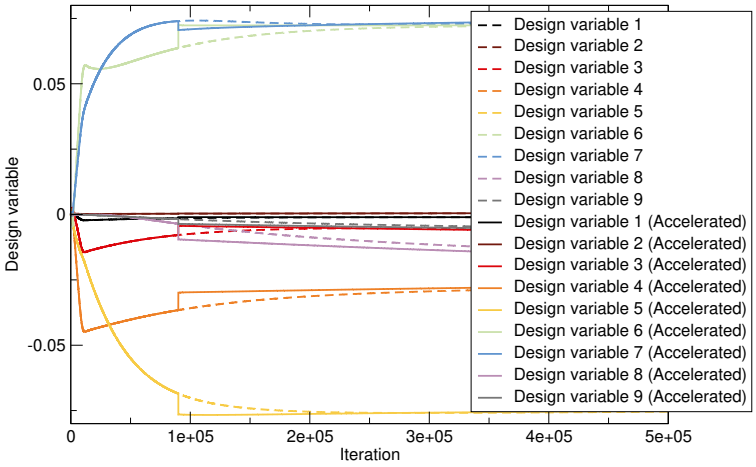


Figure 90: Design variable history for the one-shot optimizer with a single application of the reduced-order model acceleration technique.

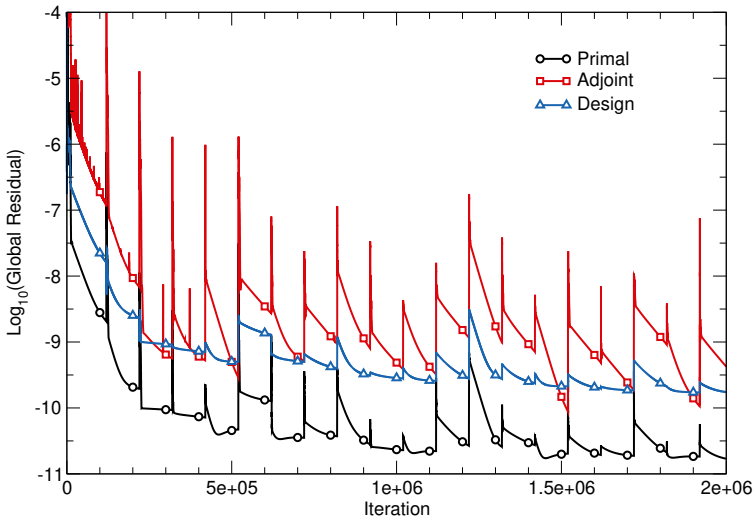


Figure 91: Convergence history of the primal, adjoint, and design equations solved via the ROM-accelerated one-shot optimization approach.

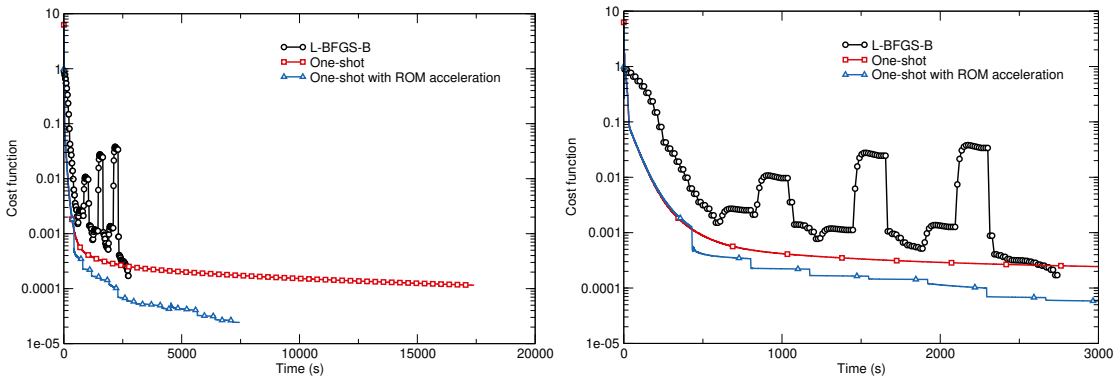


Figure 92: Cost function history over (a) each design cycle and (b) time.

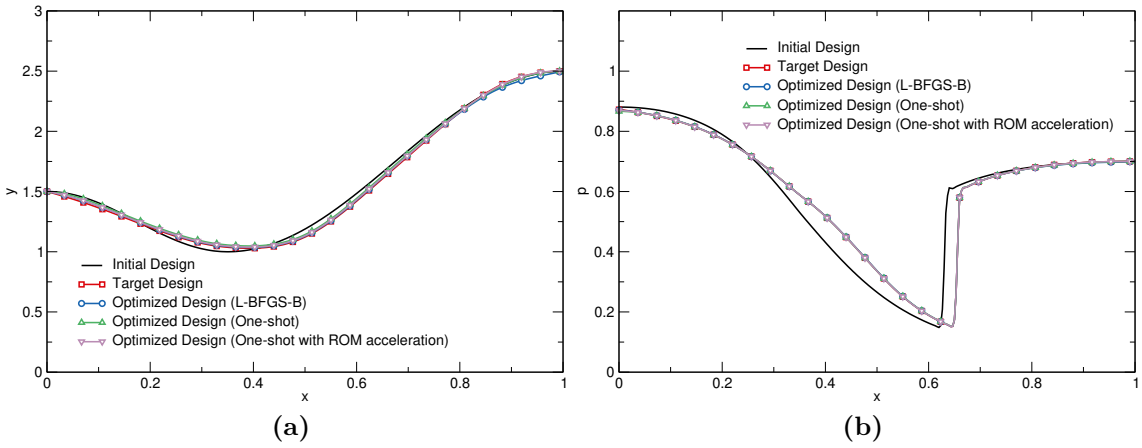


Figure 93: Comparison of the (a) nozzle cross-section and (b) pressure profile recovered by the ROM-accelerated one-shot optimizer.

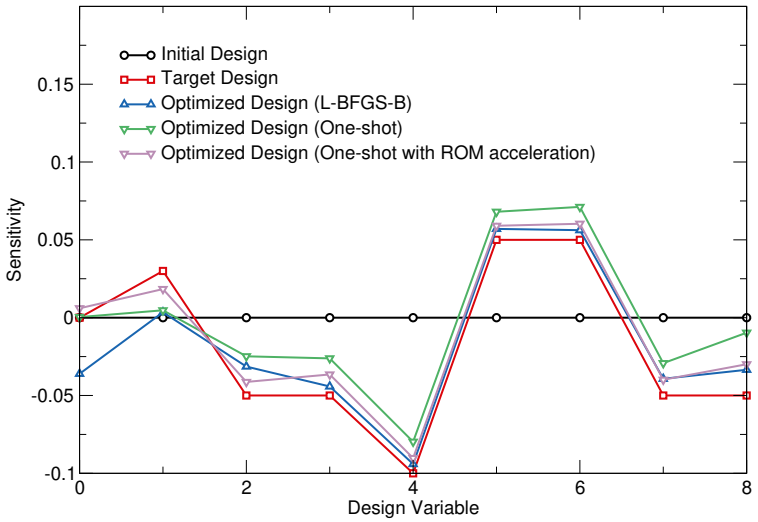


Figure 94: Comparison of the optimized design variable values from each optimization approach.

Vita

Andrew Kaminsky was born in Knoxville, Tennessee to Anne and Larry Kaminsky. He attended Farragut High School with his younger brother, Rob, and graduated in 2008. Andrew completed his undergraduate degree in Mechanical Engineering with a minor in Aerospace Engineering at the University of Tennessee, Knoxville and graduated in 2012. During his undergraduate studies, he interned at Oak Ridge National Laboratory in the Neutron Facilities Development Division through multiple Science Undergraduate Laboratory Internships (SULI), a Higher Education Research Experience (HERE) internship, and a Nuclear Engineering Science Laboratory Synthesis (NESLS) internship. Under these internships his mentors Ashraf Abdou, Mark Wendel, and Bernie Riemer first introduced him computational fluid dynamics, which he applied to model the liquid mercury flows within the Spallation Neutron Source target.

Following graduation, Andrew immediately began pursuit of his Ph.D. in Mechanical Engineering at the University of Tennessee, under the guidance of Dr. Kivanc Ekici as a joint Mechanical, Aerospace, and Biomedical engineering (MABE) chancellor's fellow and a Bredesen Center for Interdisciplinary Research and Graduate Education Energy Science and Engineering fellow. Through Dr. Ekici's mentorship, Andrew discovered his love for numerical optimization and aerodynamic design. During his graduate degree Andrew was elected President of the Graduate Association of MABE. As president he developed a STEM Outreach Program through the ASME's Diversity Action Grant. Andrew has been recognized by the University for his research with the Chancellor's Award for Extraordinary Professional Promise and his community outreach with the MABE Student Leadership and Outreach Award.

During his graduate studies, Andrew accepted a position with CFD Research Corporation, located in Huntsville, Alabama. His role at CFD Research has allowed him to continue pursuing his passion for efficient aerodynamic design. Andrew lives in Huntsville with the true highlights of his life, his wife Courtney and his daughters Eleanor and Emily.