

Enabling Automated, Reliable, and Efficient Aerodynamic Shape Optimization With Output-Based Adapted Meshes

by

Guodong Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering and Scientific Computing)
in The University of Michigan
2020

Doctoral Committee:

Associate Professor Krzysztof J. Fidkowski, Chair

Associate Professor Karthik Duraisamy

Associate Professor Kevin J. Maki

Professor Joaquim R. R. A. Martins

Guodong Chen

cgderic@umich.edu

ORCID iD: [0000-0002-3294-4749](https://orcid.org/0000-0002-3294-4749)

© Guodong Chen 2020 All Rights Reserved

For my grandfather

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Professor Krzysztof J. Fidkowski, for his continuous support and insightful guidance throughout my graduate study. I am grateful to him for giving me the flexibility of trying out many different research ideas, while still keeping me on track towards the main dissertation topic. It is always a privilege to work with him either as a PhD student or as a teaching assistant. I have been and will be forever inspired by his enthusiasm for research and his commitment to teaching.

I would also like to thank my doctoral committee members, Professor Karthik Duraisamy, Professor Joaquim R. R. A. Martins, and Professor Kevin J. Maki, for their time and effort spent on this work. Particularly, Professor Duraisamy has taught my first graduate CFD class which refreshed my knowledge in this field and has always been accessible and helpful during my PhD; Professor Martins has benefited my work through his expertise in optimization over the years, from high-level motivations to detailed algorithm implementations; Professor Maki has provided valuable comments and feedback in a different perspective.

Life at Michigan would be much less fun without the cool friends I have met and befriended. I would like to thank my research labmates: Steve Kast, Johann Dham, Kyle Ding, Devina Sanjaya, Yuki Shimizu, Gustavo Halila, Kevin Doetsch, Gary Collins, Vivek Ojha, Matteo Franciolini, Qingzhao Wang, Miles McGruder, and Rakesh Halder. Special thanks go to Johann for his hands-on help with the group research code at my beginning stage of PhD. I feel lucky to sit in the same office with Devina, Kyle, and Yuki. I enjoy discussing research with you, I miss the time going out and having fun with you, and I am grateful for your advice on both professional and personal development. Office 2025 has most of my memories for PhD and you guys are forever my best friends.

I am also grateful to the friends outside my group who have stood with me. My current office mate Yifan Bai deserves many thanks for sharing the same space peacefully and for being a good friend. I cherish the friendship with Jiayang Xu, Shaowu Pan, Daning Huang, Sicheng He, Ping He, Jichao Li, Yayun Shi, and I have learned a lot from discussions of research with you. Many thanks go to Fabian Chacon, Samuel Chen, Doreen Fan, Chris Marley, Chris Wentland, Logan White, Aaditya Lakshmanan, Siddhartha

Srivastava, Avinkrishnan Vijayachandran, Shiyao Lin, and Minh Hoang Nguyen for being so welcoming of me. I enjoy hanging out with you guys and hope our paths cross again in the future. Hoang deserves recognition for making my life so much easier as a cool roommate such that I could focus on research and thesis writing at the very end of my PhD. To other friends over the years, Binbin Li, Jiabin Zhu, Tianqi Zhao, Yinong Tan, Huirong Ning, Lan Ma, Sunming Qin, Yucheng Liu, Xunwei Zhang, Zhiyi Wang, thank you for being you.

Finally, I would like to thank my parents Qinshun Chen and Zhongge Qu, for bring me to this world and shaping my life with their continuous encouragement and support, I could never have been this far without you. Last but certainly not least, I thank my beloved one, Xiaoya Ding, for her love and support. You are the light of my life and I look forward to the future of our lives together.

This work was supported by the Department of Energy under the grant DE-FG02-13ER26146/DE-SC0010341, the Boeing company under the contract PC-1658167, with technical monitor Dr. Mori Mani, the University of Michigan through Michigan Institute for Computational Discovery and Engineering (MICDE) Fellowship and the Rackham Graduate Student Research Grant.

TABLE OF CONTENTS

DEDICATION	ii
Acknowledgements	iii
List of Figures	ix
List of Tables	xiii
List of Abbreviations	xv
List of Symbols	xvii
Abstract	xx
CHAPTER	
1. Introduction	1
1.1 Motivation	1
1.2 Challenges in Aerodynamic Optimization	3
1.2.1 Design Automation	5
1.2.2 Design Assessment and Reliability	7
1.2.3 Computational Efficiency	11
1.3 Opportunities for Adaptive CFD in Aerodynamic Optimization	13
1.3.1 High-Order CFD Methods	13
1.3.2 Output Error Estimation	14
1.3.3 Mesh Adaptation	17
1.3.4 Aerodynamic Optimization With Adaptive CFD	18
1.4 Thesis Overviews	20
1.4.1 Major Contributions	20
1.4.2 Thesis Outline	21
2. Governing Equations and Discretization	22
2.1 Equations and Notation	22
2.2 Compressible Navier-Stokes Equations	23

2.3	Reynolds-Averaged Navier-Stokes Equations	26
2.4	Discontinuous Galerkin Discretization	29
2.4.1	Solution Approximation	29
2.4.2	Weak Form	31
2.4.3	Discrete Form and Solution Technique	35
3.	Adjoint-Based Sensitivity Analysis and Output Error Estimation	38
3.1	Adjoint and Duality	38
3.2	Discrete Adjoint	41
3.3	Adjoint-Based Sensitivity Analysis	45
3.4	Adjoint-Based Output Error Estimation	47
3.4.1	Error Localization	51
3.5	Adjoint Consistency	54
4.	Aerodynamic Optimization Problem Formulation, Error Estimation and Mesh Adaptation	56
4.1	Continuous and Discrete Optimization	57
4.2	Optimization Formulation via the Adjoint	59
4.2.1	Optimizer-Based Trimming	61
4.2.2	Solver-Based Trimming	62
4.3	Output Error Estimation for Optimization	65
4.3.1	Output Error Estimation for Standalone Simulations	66
4.3.2	Output Error Estimation for Optimization Problems	66
4.3.3	Implementation	72
4.4	Mesh Adaptation Incorporating Anisotropy	73
4.4.1	Error Localization	73
4.4.2	Continuous Mesh Framework	74
4.4.3	Hessian-Based Anisotropy Detection	76
4.4.4	Sampling-Based Anisotropy Detection	80
5.	Aerodynamic Optimization Framework with Adaptive CFD	88
5.1	Two-Dimensional Airfoil Optimization	88
5.1.1	Problem Statement	88
5.1.2	Geometry Parametrization	89
5.1.3	Angle of Attack Handling	90
5.2	Mesh Deformation	90
5.2.1	Inverse Distance Weighting Interpolation	91
5.2.2	Radial Basis Function Interpolation	92
5.3	Optimization Algorithm	93
5.3.1	Gradient-Based Optimizer	93
5.3.2	Incorporation with Output Error Estimation and Mesh Adaptation	95

5.3.3	Consistent Objective-Sensitivity Analysis	97
5.3.4	Multifidelity Optimization Algorithm Overviews	100
6.	Application to Aerodynamic Optimization	103
6.1	Solution Accuracy and Optimization Tolerance	103
6.2	On the Effects of Discretization Errors in Optimization	106
6.2.1	One-Dimensional Scalar Advection-Diffusion	106
6.2.2	Inviscid Transonic RAE 2822 Airfoil Optimization	108
6.3	Optimizations with Adaptive CFD	117
6.3.1	Revisit of the One-Dimensional Advection-Diffusion Problem	117
6.3.2	Revisit of the Inviscid Transonic Optimization on the RAE 2822 Airfoil	119
6.3.3	Tandem RAE 2822 Airfoils	128
6.3.4	ADODG Case 2: Turbulent Transonic Optimization on RAE 2822	134
6.4	Summary	138
7.	Extension to Multipoint Aerodynamic Optimization	140
7.1	Multipoint Optimization Problem	140
7.1.1	Weighted-Sum Approach	140
7.1.2	Adjoint and Design Equations	141
7.1.3	Output Error Estimation	143
7.2	Mesh Adaptation	145
7.2.1	Error/Cost Allocation for Multipoint Mesh Adaptation	146
7.2.2	Mesh Adaptation at Individual Design Points	148
7.3	Optimization Algorithms	148
7.4	Results	149
7.4.1	Two-Point Inviscid Transonic Airfoil Optimization	151
7.4.2	Three-Point Turbulent Transonic Airfoil Optimization	158
7.5	Summary	163
8.	Mesh Adaptation Acceleration Techniques Based on Machine Learning	166
8.1	Machine-Learning Anisotropy Detection	167
8.1.1	Primal and Adjoint Features	168
8.1.2	Error Indicator Features	169
8.1.3	Surrogate Model Using Neural Networks	170
8.2	Neural Network Training	173
8.3	Adaptive Simulation Results	176
8.3.1	NACA 0012 Airfoil	178
8.3.2	Diamond Airfoil	179

8.3.3	MDA 30P/30N Airfoil	183
8.3.4	Extrapolation: Tandem NACA 5410 Airfoils	184
8.3.5	Adaptive Iteration Comparison	188
8.3.6	$p = 3$ Results	191
8.4	Summary	196
9.	Adjoint-Free Error Estimation and Mesh Adaptation Using Convolutional Neural Networks	198
9.1	CNN-Based Model for Output Error Estimation	201
9.1.1	Parameterized PDEs and Output Error Estimation	201
9.1.2	Hanging-Node Mesh Adaptation	203
9.1.3	Surrogate Model as a Regression Problem	204
9.1.4	Convolutional Neural Networks	205
9.1.5	Proposed Architecture and Network Training	208
9.1.6	Fixed Network for Adaptive Simulation on General Domains	209
9.2	Two-Dimensional Advection-Diffusion Problem	211
9.2.1	Data Generation and Preprocessing	213
9.2.2	Network Implementation and Training	214
9.2.3	Network Testing and Model Deployment	218
9.3	Application in Aerodynamic Simulations Over Airfoils	226
9.3.1	Data Generation and Preprocessing	226
9.3.2	Network Implementation and Training	230
9.3.3	Network Testing and Model Deployment	232
9.4	Summary	242
10.	Conclusions and Future Work	245
10.1	Summary and Conclusions	245
10.2	Future Work	249
	Bibliography	250

LIST OF FIGURES

Figure

1.1	Examples of unconventional aircraft configurations.	3
1.2	An example of an aerodynamic optimization workflow. The essential modules are highlighted in gray boxes.	5
1.3	Aerodynamic design process including the problem setup and the design assessment.	6
1.4	Numerical errors in aerodynamic optimization.	8
1.5	Drag coefficients scatter from DPW-IV and DPW-V.	10
1.6	Examples of discretization error effects in optimization.	12
1.7	Aerodynamic optimization with error estimation and mesh adaptation.	19
1.8	Thesis outline.	21
2.1	An example unstructured mesh \mathcal{T}_h for flow over a smooth bump in the computational domain Ω	30
2.2	Solution approximation using the CG and DG methods.	31
3.1	Sample primal and adjoint solutions of a laminar flow simulation over the NACA 0012 airfoil.	44
3.2	Nonlinear system solve as a feedback control system.	47
3.3	A summary of the error estimation and error localization procedure applied to a laminar flow simulation over a NACA 0012 airfoil.	53
4.1	Flowchart of optimization using an optimizer-based trimming approach.	62
4.2	Trimming process using Newton-Raphson iteration.	63
4.3	Flowchart of optimization using a solver-based trimming approach.	65
4.4	An example of a two-dimensional metric field.	75
4.5	An illustration of metric-based global re-meshing in adaptation.	76
4.6	Four refinement options for a triangle. Each one is considered implicitly during error sampling, though the elements are never actually refined.	83
6.1	Optimization governed by a one-dimensional scalar advection-diffusion PDE on a uniform coarse mesh.	107
6.2	Manually generated fixed mesh on RAE 2822 airfoil	110
6.3	Comparison of optimizations on a fixed mesh with various approximation orders.	110
6.4	Mach contour and pressure distribution of the optimized designs from a fixed mesh with various approximation orders.	111
6.5	Re-analysis of the $p = 1$ designs on the $p = 4$ “true” space.	114

6.6	Re-analysis of the $p = 2$ designs on the $p = 4$ “true” space.	115
6.7	Re-analysis of the $p = 3$ designs on the $p = 4$ “true” space.	116
6.8	Optimization governed by the one-dimensional scalar advection-diffusion PDE on different meshes.	118
6.9	Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: meshes for multifidelity and fixed-fidelity optimizations. PDE on a uniform coarse mesh.	121
6.10	Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: objective convergence history and mesh size evolution for different methods.	122
6.11	Mach contours on the adapted meshes from different adaptation methods.	124
6.12	Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: final design comparison.	126
6.13	Tandem RAE 2822 airfoils: initial configuration and the corresponding primal and adjoint fields.	129
6.14	Tandem RAE 2822 airfoils: final adapted meshes on the optimized designs.	131
6.15	Tandem RAE 2822 airfoils: objective convergence and mesh size evolution.	131
6.16	Tandem RAE 2822 airfoils: comparison between the optimized designs from Hessian adaptation and MOESS. Mach number contours has the same scale, $[0.3, 1.35]$	133
6.17	ADODG case 2: the initial mesh and adapted meshes in the optimization.	135
6.18	ADODG case 2: objective convergence history and mesh size evolution for different methods.	136
6.19	ADODG case 2: local Mach number (0-1.3) and pressure distributions for the initial and the optimized designs.	137
7.1	Two-point inviscid transonic airfoil optimization: meshes for variable-fidelity and fixed-fidelity optimization.	154
7.2	Two-point inviscid transonic airfoil optimization: convergence history and mesh size evolution for different methods.	155
7.3	Meshes around the stagnation streamline, the left mesh is from cost-based Hessian adaptation, the right one is the MOESS adapted mesh, both at $M = 0.76$	156
7.4	Two-point inviscid transonic airfoil optimization: pressure distribution for the initial and optimized designs.	158
7.5	Two-point inviscid transonic airfoil optimization: Mach contours at $M = 0.70$. The color limit is $[0, 1.28]$ for all of the contours.	159
7.6	Two-point inviscid transonic airfoil optimization: Mach contours at $M = 0.76$. The color limit is $[0, 1.4]$ for all of the contours.	160
7.7	Three-point turbulent transonic airfoil optimization: initial mesh and final meshes during the optimization.	162
7.8	Three-point turbulent transonic airfoil optimization: objective convergence history and mesh size evolution for different methods.	162
7.9	Three-point turbulent transonic airfoil optimization: pressure distribution for the initial and optimized designs.	164
7.10	Three-point turbulent transonic airfoil optimization: Mach contours. The color limit is $[0, 1.4]$ for all of the contours.	165

8.1	Structures of artificial neural networks (ANNs).	171
8.2	Flowchart of the neural-network implementation.	173
8.3	Neural network A training loss history, using a 70% training, 30% validation split.	176
8.4	Visualization of the trained neural networks.	177
8.5	NACA 0012, $M_\infty = 0.8$, $\alpha = 1.25^\circ$, $Re = 5 \times 10^6$: output convergence history.	178
8.6	NACA 0012: final adapted meshes.	180
8.7	Diamond airfoil, $M_\infty = 1.5$, $\alpha = 2^\circ$, $Re = 1 \times 10^6$: output convergence history.	181
8.8	Diamond airfoil: final adapted meshes.	182
8.9	MDA 30P/30N, $M_\infty = 0.2$, $\alpha = 5^\circ$, $Re = 5 \times 10^6$: output convergence history.	184
8.10	MDA 30P/30N airfoil: final adapted meshes.	185
8.11	Tandem NACA 5410 airfoils: initial mesh and primal/adjoint solutions.	187
8.12	Tandem NACA 5410 airfoils: output convergence history.	188
8.13	Tandem NACA 5410 airfoils: final adapted meshes.	189
8.14	Tandem NACA 5410 airfoils: adaptive iteration performance.	190
8.15	Tandem NACA 5410 airfoils: meshes at adaptive iterations 1 (top) through 4 (bottom).	191
8.16	Visualization of the neural networks for $p = 3$. The format is the same as the $p = 2$ version in Figure 8.4.	192
8.17	Tandem NACA 5410 airfoils: $p = 3$ output convergence history.	193
8.18	Tandem NACA 5410 airfoils: $p = 3$ final adapted meshes.	194
8.19	Tandem NACA 5410 airfoils: $p = 3$ adaptive iteration performance.	195
8.20	Tandem NACA 5410 airfoils: $p = 3$ meshes at adaptive iterations 1 (top) through 4 (bottom).	195
9.1	Hanging-node adaptation for a quadrilateral mesh.	204
9.2	An example of convolution operations.	207
9.3	An example of encoder-decoder type convolutional neural networks.	207
9.4	Proposed encoder-decoder network architecture.	209
9.5	An example of building the proposed CNN model on rectangular domains.	210
9.6	An example of building the proposed CNN model on general domains.	212
9.7	Samples from the dataset of the scalar advection-diffusion problem.	215
9.8	Training history of the model for the scalar advection-diffusion problem.	216
9.9	Model performance of error indicator field predictions on the training and validation sets (scalar advection-diffusion).	218
9.10	Model performance of output error predictions on the training and validation sets (scalar advection-diffusion).	219
9.11	Model interpolation test on the testing dataset (scalar advection-diffusion).	220
9.12	Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion).	221
9.13	Model extrapolation test on the testing data (scalar advection-diffusion, unseen Pe).	222

9.14	Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion, unseen Pe).	223
9.15	Model extrapolation test on the testing data (scalar advection-diffusion, unseen α).	224
9.16	Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion, unseen α).	225
9.17	The starting coarse mesh for the adaptive simulations and the fixed reference fine mesh.	228
9.18	Samples from the dataset of the airfoil problem.	229
9.19	Training history of the model for airfoil simulations.	230
9.20	Model performance of error indicator field predictions on the training and validation sets (aerodynamic flow over airfoils).	232
9.21	Model performance of output error predictions on the training and validation sets (aerodynamic flow over airfoils).	233
9.22	Model interpolation test on the testing dataset (aerodynamic flow over airfoils).	234
9.23	Model deployment in adaptive simulations: NACA 2614 airfoil, $M = 0.54, \alpha = 1.2^\circ$	235
9.24	Model deployment in adaptive simulations: NACA 4410 airfoil, $M = 0.68, \alpha = 1.0^\circ$	236
9.25	Model deployment in adaptive simulations: NACA 0012 airfoil, $M = 0.50, \alpha = 2.0^\circ$	237
9.26	Model deployment in adaptive simulations: NACA 2412 airfoil, $M = 0.70, \alpha = 1.0^\circ$	238
9.27	Model deployment in adaptive simulations: NACA 0010 airfoil, $M = 0.60, \alpha = -1.0^\circ$	239
9.28	Model deployment in adaptive simulations: NACA 4412 airfoil, $M = 0.62, \alpha = 4.0^\circ$	240
9.29	Model deployment in adaptive simulations: NACA 3709 airfoil, $M = 0.66, \alpha = 0^\circ$	241
9.30	Model deployment in adaptive simulations: NACA 5715 airfoil, $M = 0.62, \alpha = 1.0^\circ$	242

LIST OF TABLES

Table

6.1	Optimization results (advection-diffusion PDE) on a uniform mesh. The discretization uses DG with $p = 2$ and $N_e = 8$ elements, the optimization tolerance is set to be $\epsilon = 10^{-10}$	108
6.2	Optimization results (inviscid transonic RAE 2822 airfoil optimization) on a fixed mesh with various approximation orders. The design index is the iteration number of the considered design, J_H denotes the objective evaluated on the current space, while J_h is the objective obtained on the “true” $p = 4$ space; the estimated δJ_h represents the error estimate using the adjoint-weighted residual, per Eqn. 4.37, while the “true” δJ measures the difference between the objectives on the current and the “true” space.	113
6.3	Optimization results (advection-diffusion PDE) on uniform and adapted meshes. The discretization uses DG with $p = 2$ and $N_e = 8$ elements, the optimization tolerance is set to be $\epsilon = 10^{-10}$	119
6.4	Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: optimization results summary on different meshes.	125
6.5	Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: computational cost comparison. In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity. The computational time results are obtained on the same HPC cluster (3.0 GHz Intel Xeon Gold 6154) using parallel runs with 16 cores and 16GB RAM.	127
6.6	Tandem RAE 2822 airfoils: computational cost comparison. In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity. Results are obtained on the same HPC cluster (3.0 GHz Intel Xeon Gold 6154) using parallel runs with 36 cores and 36GB RAM.	132
6.7	Tandem RAE 2822 airfoils: optimized objective on different meshes. The “true” objective are obtained using order increment from p to $p + 1$	132

6.8	Comparison of the drag coefficients (in drag counts) for RANS optimizations of the RAE 2822 airfoil. The mesh sizes are reported in the number of elements (with superscript \star) or the number of nodes (with superscript \dagger) in different works.	138
7.1	Multipoint aerodynamic shape optimization problem	149
7.2	Operating conditions for multipoint optimization problems	152
7.3	Two-point inviscid transonic airfoil optimization: computational cost comparison.	157
7.4	Two-point inviscid transonic airfoil optimization: results on different meshes.	161
8.1	Four neural network architectures for anisotropy detection.	172
8.2	Neural network training cases.	175
9.1	Network architecture for the scalar advection-diffusion problem.	217
9.2	Network architecture for aerodynamic simulations over airfoils	231

LIST OF ABBREVIATIONS

ADODG	Aerodynamic Design Optimization Discussion Group
AIAA	American Institute of Aeronautics and Astronautics
ALE	Arbitrary Lagrangian Eulerian
ANN	Artificial neural network
AR	Aspect ratio
AWR	Adjoint weighted residual
BAMG	Bi-dimensional Anisotropic Mesh Generator
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BR2	Second form of Bassi and Rebay
BWB	Blended wing body
CAGR	Compound annual growth rate
CFD	Computational fluid dynamics
CFDG	Computational fluid dynamics group
CFL	Courant-Friedrichs-Lewy
CG	Continuous Galerkin
CNN	Convolutional neural network
CO₂	Carbon dioxide
CORSIA	Carbon Offset and Reduction Scheme for International Aviation
CRM	Common Research Model
DG	Discontinuous Galerkin
DNS	Direct numerical simulation
DOF	Degrees of freedom
DPW	Drag Prediction Workshop
DWR	Dual weighted residual
EDG	Embedded discontinuous Galerkin

EPIC	Edge Primitive Insertion and Collapse
FCN	Fully connected network
GMRES	Generalized minimal residual method
GPU	Graphics processing unit
HDG	Hybridized discontinuous Galerkin
HPC	High performance computing
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IDW	Inverse distance weighting
KKT	Karush-Kuhn-Tucker
LES	Large eddy simulation
LSEI	Least squares with equality and inequality constraints
MDA	McDonnell Douglas Aerospace
MDO	Multidisciplinary design optimization
MLP	Multi-layer perceptron
MMT	Million metric ton
MOESS	Mesh optimization via error sampling and synthesis
NACA	National Advisory Committee for Aeronautics
NLP	Nonlinear programming
NS	Navier-Stokes
OBT	Optimizer-based trimming
ODE	Ordinary differential equation
PDE	Partial differential equation
QP	Quadratic programming
RAE	Royal Aircraft Establishment
RANS	Reynolds-averaged Navier-Stokes
RBF	Radial basis function
ReLU	Rectified linear unit
ROM	Reduced-order model
SA	Spalart-Allmaras
SBT	Solver-based trimming
SLSQP	Sequential Least Squares Programming
SPD	Symmetric positive definite
SQP	Sequential quadratic programming

LIST OF SYMBOLS

General

Re	Reynolds number
M	Mach number
Pe	Péclet number
μ	Dynamic viscosity
ν	Kinematic viscosity
α	Angle of attack
c_ℓ	Lift coefficient
c_d	Drag coefficient
Ω	Computational domain

Discretization

\mathcal{T}_h	Computational mesh
Ω_e	One mesh element
p	Solution approximation order; also used to denote pressure
\mathcal{V}, \mathbf{V}	Infinite dimensional solution space
$\mathcal{V}_h, \mathbf{V}_h$	Finite dimensional solution space
\mathbf{u}	Exact state vector in \mathbf{V}
\mathbf{u}_h	Discrete state vector in \mathbf{V}_h
\mathbf{w}, \mathbf{w}_h	Test function in \mathbf{V} and \mathbf{V}_h
$\vec{\mathbf{H}}$	Total flux
$\vec{\mathbf{F}}$	Inviscid flux
$\vec{\mathbf{G}}$	Viscous flux
\mathbf{S}	Source term
$\mathcal{R}, \mathcal{R}_h$	Semi-linear weak form of the equations in \mathbf{V} and \mathbf{V}_h

\mathbf{U}_h	Fully-discrete unknown vector of the state
\mathbf{R}_h	Fully-discrete residual vector

Optimization

\mathbf{x}	Design parameter
\mathbf{x}^*	Optimal design
J^{adapt}	Objective/adapt output
$\mathbf{J}^{\text{trim}}, \bar{\mathbf{J}}^{\text{trim}}$	Constraint/trim output vector and its target values
\mathbf{R}^{trim}	Trim equation residual vector
\mathcal{L}	Lagrangian function
λ, μ	Lagrange multipliers/coupled adjoints
\mathbf{x}_t	Trim variables
\mathbf{x}_s	Active design variables

Error Estimation and Mesh Adaptation

\mathcal{J}	Continuous output functional
J_h	Discrete output functional
δJ	Output error estimate
ψ	Exact adjoint solution in \mathcal{V}
ψ_h	Discrete adjoint solution in \mathcal{V}_h
Ψ_h	Fully-discrete unknown vector of the adjoint
H, h	Coarse and fine solution spaces (discretizations)
\mathbf{U}_h^H	Injected states from the coarse space to the finer one
\mathcal{E}	Error indicator
\mathcal{M}	Riemannian metric field
\mathbf{H}	Hessian matrix
\mathcal{S}	Step matrix
\mathcal{R}_e	Anisotropic output error convergence rate tensor

Machine Learning

μ	Equation parameters
\mathbf{W}	Network weight matrix
\mathbf{b}	Network bias vector

Θ	Network trainable parameters
σ	Activation function
\mathbf{h}	Network hidden layer
\mathbf{H}^u	Primal solution Hessian matrix
\mathbf{H}^ψ	Adjoint solution Hessian matrix
$\bar{\mathbf{H}}$	Normalized Hessian matrix
$\overline{\mathcal{M}}$	Normalized mesh metric

ABSTRACT

Simulation-based aerodynamic shape optimization has been greatly pushed forward during the past several decades, largely due to the developments of computational fluid dynamics (CFD), geometry parameterization methods, mesh deformation techniques, sensitivity computation, and numerical optimization algorithms. Effective integration of these components has made aerodynamic shape optimization a highly automated process, requiring less and less human interference. Mesh generation, on the other hand, has become the main overhead of setting up the optimization problem. Obtaining a good computational mesh is essential in CFD simulations for accurate output predictions, which as a result significantly affects the reliability of optimization results. However, this is in general a nontrivial task, heavily relying on the user’s experience, and it can be worse with the emerging high-fidelity requirements or in the design of novel configurations. On the other hand, mesh quality and the associated numerical errors are typically only studied before and after the optimization, leaving the design search path unveiled to numerical errors. This work tackles these issues by integrating an additional component, output-based mesh adaptation, within traditional aerodynamic shape optimizations.

First, we develop a more suitable error estimator for optimization problems by taking into account errors in both the objective and constraint outputs. The localized output errors are then used to drive mesh adaptation to achieve the desired accuracy on both the objective and constraint outputs. With the variable fidelity offered by the adaptive meshes, multi-fidelity optimization frameworks are developed to tightly couple mesh adaptation and shape optimization. The objective functional and its sensitivity are first evaluated on an initial coarse mesh, which is then subsequently adapted as the shape optimization proceeds. The effort to set up the optimization is minimal since the initial mesh can be fairly coarse and easy to generate. Meanwhile, the proposed framework saves computational costs by reducing the mesh size at the early stages of the optimization, when the design is far from optimal, and avoiding exhaustive search on low-fidelity meshes when the outputs are inaccurate. To further improve the computational efficiency, we also introduce new methods to accelerate the error estimation and mesh adaptation using machine learning techniques. Surrogate models are developed to predict the localized output error

and optimal mesh anisotropy to guide the adaptation. The proposed machine learning approaches demonstrate good performance in two-dimensional test problems, encouraging more study and developments to incorporate them within aerodynamic optimization techniques.

Although CFD has been extensively used in aircraft design and optimization, the design automation, reliability, and efficiency are largely limited by the mesh generation process and the fixed-mesh optimization paradigm. With the emerging high-fidelity requirements and the further developments of unconventional configurations, CFD-based optimization has to be made more accurate and more efficient to achieve higher design reliability and lower computational cost. Furthermore, future aerodynamic optimization needs to avoid unnecessary overhead in mesh generation and optimization setup to further automate the design process. The author expects the methods developed in this work to be the keys to enable more automated, reliable, and efficient aerodynamic shape optimization, making CFD-based optimization a more powerful tool in aircraft design.

CHAPTER 1

Introduction

1.1 Motivation

The last century has seen tremendous development and prosperity of civil aviation, since the first heavier-than-air flight of the Wright brothers. Progress in technology and innovations in industry has made air travel more accessible than ever, with lower fares that are affordable for more people. Worldwide air passenger numbers exceeded 4.3 billion in 2018 [1], and will continue to rise for the next 20 years (2019-2038) with a compound annual growth rates (CAGRs) of 4.6%, 4.3%, and 4.7%, forecast by Boeing, Airbus and Embraer respectively [2, 3, 4]¹. By connecting more and more major cities directly over the world, air transport has been one of the key forces pushing globalization, continuously supporting the economic growth through tourism and trade globally.

Despite the rapid development and growth of air transport, its environmental impact has become a major concern for both the industry and the public during the past several decades. The aviation sector accounts for approximately 2% of global anthropogenic carbon dioxide (CO₂) emissions [7, 8, 9], including international and domestic aviation. While the percentage of global CO₂ emissions from civil aviation has not significantly changed since 1992, the volume of emissions has increased along with the increase in global CO₂ emissions across other sectors. Civil aviation, as a whole, burned 77 million gallons of fuel, releasing 733 million metric tons (MMTs) of CO₂ in 2014 [10]. By the time of 2018, 905 MMTs of CO₂ have been emitted due to the consumption of 95 million

¹In fact, a significant reduction of the air travel has been seen in 2020 instead of the predicted increase, due to the global pandemic of COVID-19 (<https://www.who.int/emergencies/diseases/novel-coronavirus-2019>) and the associated travel restrictions. International Civil Aviation Organization (ICAO) has predicted an overall reduction of air passengers (both international and domestic) ranging from 35% to 65% in 2020 compared to 2019, while International Air Transport Association (IATA) projected a 48% decline of revenue passenger kilometers [5]. However, with more careful operational guidelines [6] and mitigation of the pandemic, we expect a steady recover of the aviation industry.

gallons of fuel [10], showing an increase of 23% over the past five years. On the other hand, aircraft noise is the most significant cause of community annoyance to the operation of local airports [7] and has long time been an obstacle preventing the embrace of supersonic business jets [11, 12].

Fortunately, the industry has foreseen the increasing effects on environment and has committed to reduce its environmental footprint. The International Air Transport Association (IATA) and the International Civil Aviation Organization (ICAO) have established several ambitious goals until 2050 to mitigate or reduce CO₂ emissions, largely stimulating the increase of fuel burn efficiency of new commercial aircraft. Meanwhile, airline profitability is another classical thrust for improving aircraft efficiency, since fuel consumption accounts for more than 20% of an airline’s operating cost. On the other hand, the reduction of noise emissions mainly relies on new standards adopted by agencies and the participation of countries. Many efforts, including engine improvements and drag reduction techniques have been devoted to meeting these concerns in aircraft design, making the modern commercial jets 80% more efficient [13] and 75% quieter [14] than their first-generation ancestors. However, new improvements are becoming marginal and more difficult to achieve with more and more refined aircraft designs over generations.

An alternative solution that has drawn much attention in recent years is that of unconventional aircraft configurations, including the blended wing body (BWB) [15], the jointed wing [16], the strut-braced wing [17] and the “double bubble” (D8) concept [18]. Figure 1.1 lists some examples of these unconventional configurations. Those configurations are promising as a considerable amount of efficiency improvement and emission reduction can be achieved just by the design shift from traditional single-tube-wing configurations. For example, the BWB configuration is predicted to obtain a 30% reduction in fuel burn per seat compared to a similar-sized conventional design (Airbus A380-700) using equivalent technology engines [15]; later on research also showed that the engine noise can be significantly reduced due to the shielding effects associated with the upper surface engine installation [19] for the BWB configuration.

Given the big potential to be more efficient and environmentally friendly, many design studies have been conducted on these configurations [20, 21, 22, 23, 24, 25, 26], most of which lie in extensive simulation-based aerodynamic shape optimization. However, unlike the well-established design philosophy and the good understanding for the traditional aircraft, there is no good way to evaluate the quality of the unconventional designs, even for experienced practitioners. Instead, the design largely relies on the numerical simulations rather than any reference database or empirical knowledge, such that it is critical to increase the confidence of the numerical solutions. Moreover, due to the lack of expe-

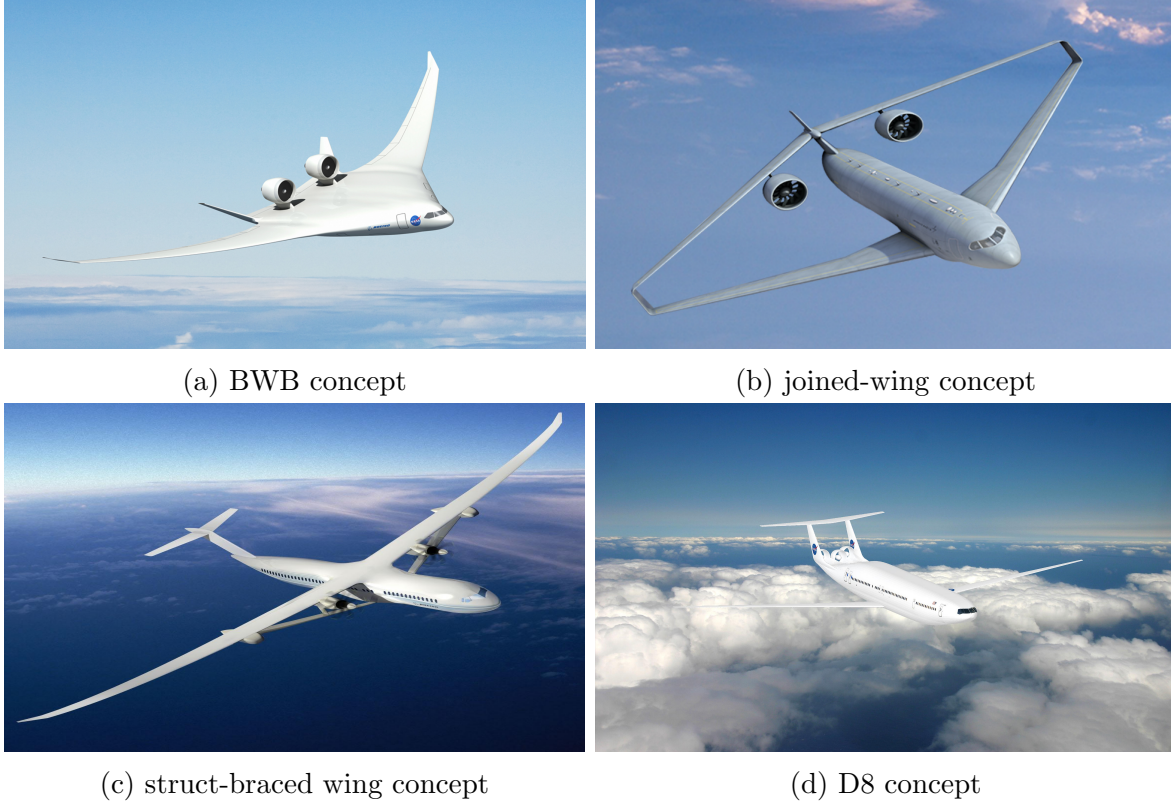


Figure 1.1: Examples of unconventional aircraft configurations. Retrieved from <https://www.nasa.gov/content/down-to-earth-future-aircraft-0>.

rience and limited intuitions, a fairly large design space needs to be explored to increase the chance of finding the optimal design. In order to enable fast unconventional design in practice, the design tools have to be highly automated to avoid unnecessary human interference and be efficient enough to reduce the turnaround time. Due to the exorbitant cost and business risk associated with the development of these novel configurations, the author believes that these issues/challenges have to be addressed before a credible effort in further development is undertaken.

1.2 Challenges in Aerodynamic Optimization

Aerodynamic shape optimization dates back long before the prevalence of CFD, when analytical flow solutions were used to determine the optimal shapes in design problems. Examples include the famous optimal elliptic lift distribution for fixed-span wings, and the Sears-Haack body that produces lowest wave drag in supersonic flows. However, these analytical solutions are restricted to relatively simple geometries under potential

flow assumptions, with or without compressibility corrections. Due to highly nonlinear nature of the flow governing equations, analytical solutions are largely unavailable and inadequate for design purposes.

Over the past 60 years, aerodynamic optimization has benefited greatly from the developments of computational fluid dynamics. The fast turnaround time, high degree of geometric flexibility, relatively low costs and almost arbitrary test conditions offered by CFD have made it an attractive tool for aerodynamic design since the 1970s. Hicks *et al.* [27] first studied the feasibility of using gradient-based numerical optimization in airfoil design problems, which was then extended to aircraft wing designs by Hicks and Henne [28]. These early attempts are based on potential flow simulations, with sensitivities obtained using finite difference methods. Since then, a variety of research efforts have been devoted to improving the accuracy and efficiency of both the flow simulations and the sensitivity calculations. With many fundamental contributions made from the 1970s to the 1990s [29], practical CFD is now capable of simulating the aerodynamic flows around complete aircraft configurations with turbulence modeling. In the context of sensitivity analysis, finite differences are certainly not the best choice as they require repeated flow solutions for each gradient component and often suffer from rounding errors. More favorable techniques have been developed during the past several decades, including the complex step method [30], algorithmic differentiation [31] and the adjoint method [32]. Among these, the adjoint method has been the most popular for derivative calculations since the pioneering work of Jameson [32, 33, 34]. Although the importance of CFD cannot be overstated in aerodynamic optimization, adjoint-based sensitivity analysis is the critical component that enables the use of CFD in large-scale practical aerodynamic optimizations [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]. Together with rapidly increasing computational power, CFD-based aerodynamic optimization has been pushed to further superiority over traditional methods, such that aerodynamic optimization today is mainly referred to as numerical optimization based on CFD simulations.

Besides the extensive use of CFD and adjoint-based sensitivity analysis, aerodynamic optimization also features several other components, including geometry parametrization methods, mesh deformation techniques, and numerical optimization algorithms. A complete aerodynamic optimization workflow requires a set of selections from each component, as sketched in Figure 1.2. Even though gradient-free optimization algorithms can also be applied to aerodynamic optimization problems, they in general require more function evaluations, *i.e.*, flow solutions, to converge [47]. The costs can be prohibitively expensive for high-dimensional problems [48], regardless of their improved robustness in non-smooth or non-convex problems. Therefore, gradient-based optimization has always

been the predominant approach in aerodynamic optimization, not only for legacy reasons but also for practical considerations. Many open-source or commercial gradient-based optimization libraries are available [49, 50, 51] and can be easily integrated with aerodynamic optimization as long as the sensitivity is accurately calculated. For geometric parametrization, an in-depth review can be found in [52], while [53] contains a detailed survey of the mesh deformation techniques.

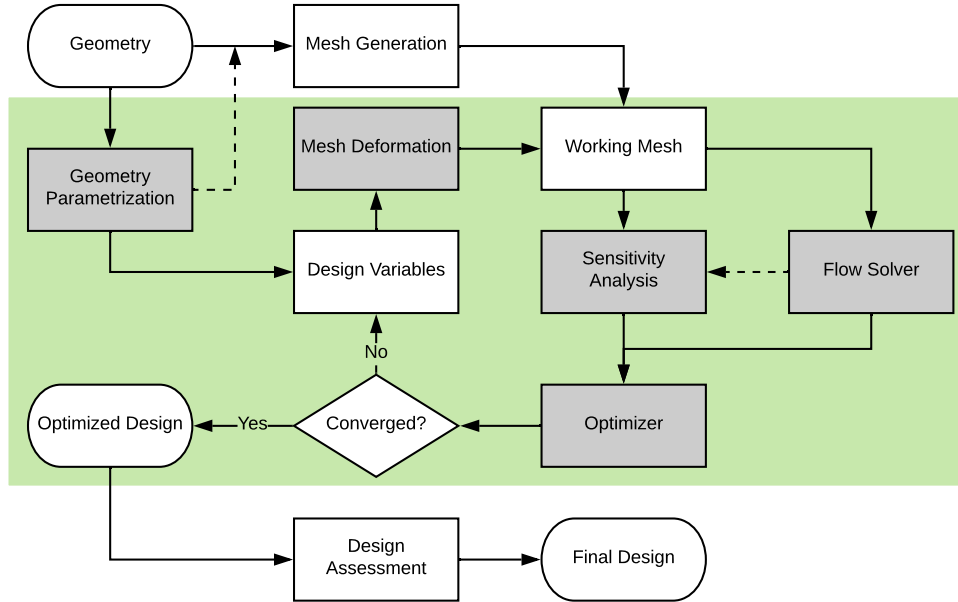


Figure 1.2: An example of an aerodynamic optimization workflow. The essential modules are highlighted in gray boxes.

With proper integration among different components, aerodynamic optimization itself can be a highly automated process, requiring minimal human interference. However, the total elapsed time for aerodynamic optimization should include the time to set up the optimization and the time spent on assessing the quality of the final designs. The efforts in setting up the optimization mainly reside in the mesh generation for the original design, while the design assessment is typically performed on the final converged design. Unlike numerical optimization, these two processes rely heavily on the user’s experience. The reliance on experienced practitioners imposes many challenges in the use of aerodynamic optimization for practical design, especially for unconventional configurations where the engineering experience is largely unavailable.

1.2.1 Design Automation

If we consider the mesh generation and the design assessment more carefully in Figure 1.2, a more thorough aerodynamic design process including the optimization setup

and the design evaluation can be described, as illustrated in Figure 1.3. Although the aerodynamic optimization itself (green part) can be made highly automated, the problem setup (pink parts) and the design assessment (blue part) usually requires excessive human involvement. The optimization problem setup often involves choosing an optimization framework (green part), defining the optimization objective, imposing design constraints, and obtaining a starting computational mesh (pink part). Among these, generating a suitable starting mesh is typically the most time-consuming part, and has long time been a bottleneck even for standalone CFD simulations.

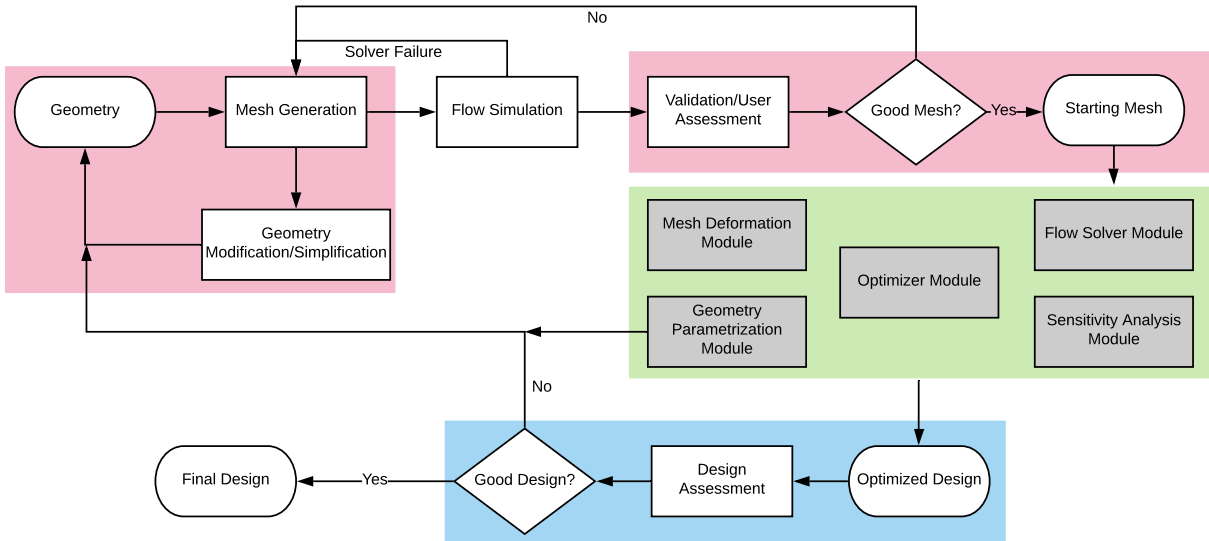


Figure 1.3: Aerodynamic design process including the problem setup and the design assessment.

Obtaining a good computational mesh for optimization, or even for a single CFD simulation is a nontrivial task, especially for complex geometries. It may take days or weeks to generate a suitable mesh around an aircraft, even by experienced meshing practitioners [54]. Enough resolution, *i.e.*, mesh elements, has to be put in regions that involve the most important flow features, *e.g.*, boundary layers, wakes and shocks, in order to achieve the required accuracy. However, these flow features are only known *a-posteriori*. As a result, mesh generation in practice extensively relies on experience or physical intuition of the possible solution field around the considered geometry. On the other hand, the generated mesh can be validated against existing experimental data to assess the quality of the mesh ². Due to the limited availability of validation data, mesh quality judgment through visual inspections by expert users is still the most common practice. The

²Assessing mesh quality, *i.e.*, numerical errors associated with the mesh, is in fact a verification exercise. Validating against experimental data to assess mesh quality assumes the physical model is well chosen.

heavy reliance on expert knowledge has made mesh generation a tedious task and a main bottleneck for many applications of CFD. Furthermore, this difficulty can be worse in aerodynamic optimization, since the available experience in meshing complex geometries, such as unconventional configurations, is not available in the first place. Furthermore, even when generated with best practice guidelines, the mesh has to go through many remeshing iterations based on extensive mesh/solution assessment by experienced meshing and aerodynamic experts, before it can be finally used to set up an optimization. In addition, the starting mesh is generated based on the initial design, with no guarantee of adequacy throughout the entire optimization search path. In order to improve the automation of the aerodynamic design process and to reduce the overall design cycle time, a better meshing paradigm should be developed to accelerate mesh generation, with allowance to ensure mesh quality during the optimization.

1.2.2 Design Assessment and Reliability

A natural and possibly most important question for aerodynamic shape optimization is: how good is the final optimized design? Due to the high costs associated with wind tunnel and flight tests, the design obtained using aerodynamic optimization has to be carefully examined before any further development is taken. Often in practice, this is investigated after the optimization run, as shown in Figure 1.3, judged by experienced aircraft designers or through high-fidelity simulations on the optimized design. For complex flow simulations or unconventional configurations, the available physical intuition is limited, such that the design assessment has to largely rely on high-fidelity simulations on the optimized design. The optimization objective functional and the constraints need to be checked on relatively finer computational meshes compared to the one used in the optimization.

To aid the assessment of aerodynamic optimization, a series of benchmark cases have been launched in the Aerodynamic Design Optimization Discussion Group (ADODG) organized by the American Institute of Aeronautics and Astronautics (AIAA). These benchmark cases are intended to provide identical test cases for comparison among state of the art optimization frameworks, which have driven many research efforts on the corresponding developments [55]. Unsurprisingly, evident differences on the optimized designs and the corresponding aerodynamic performance are observed despite the overall agreement, even on “simple” two-dimensional airfoil optimization cases [56, 57]. However, with so many components affecting the optimization, it is extremely difficult to isolate the influence of each individual component. Luckily, in general, the development costs associated with geometry parametrization and mesh deformation are relatively low, and

many optimization libraries are readily available for these tasks. Therefore, it is possible to eliminate the effects from these components, leaving the design uncertainty mainly in the accuracy of CFD solutions and the sensitivity calculations.

After the optimization has been set up, candidate designs are searched along the direction built on the sensitivity analysis. Design with sufficient performance improvements predicted by the CFD solver will be accepted and be used to advance further design updates. As we can see as well in Figure 1.4, the optimizer heavily depends on the information provided by the flow simulations and the sensitivity calculations. However, there are many error sources existing in these analyses, which can pollute the information sent to the optimizer and hence affect the optimization results. The errors include the modeling errors when the real-world system is modeled with simplified assumptions, *i.e.*, incompressible or inviscid flow, the numerical errors during the discretization of the flow governing equations on a finite-dimensional space, and the convergence errors when solving the discretized system of equations.

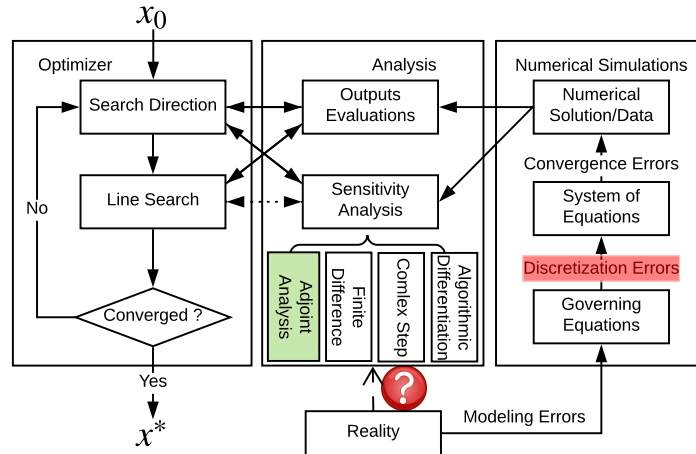


Figure 1.4: Numerical errors in aerodynamic optimization.

Modeling errors inherent to the governing equations can be reduced by model validation or by choosing more complex models. Although direct numerical simulation (DNS) and large eddy simulation (LES) have been developed for several decades, they are limited to moderate Reynolds numbers and relatively simple geometries, yet require extremely high computational costs. Reynolds-averaged Navier-Stokes (RANS) simulations with turbulence modeling are still the state of the art physical models used in aerodynamic optimization. Meanwhile, some simplified models such as inviscid Euler equations or even potential flow models are still used in aerodynamic optimization. The convergence errors, on the other hand, are easier to handle thanks to the developments in numerical algorithms and solution acceleration techniques. Therefore, the convergence errors are often

controlled to be sufficiently small compared to the errors from the other sources.

Given the availability of high-fidelity models (though some may not be computationally feasible yet) and the capability of solving the discretized system accurately and efficiently, another liability that needs to be dealt with is to ensure that the discretized system we are solving approximates the chosen physical models accurately. Compared to the physical intuition available to choose the models and the flexibility of setting convergence tolerances in the numerical solver, the choice of suitable discretization, especially the computational mesh, is not a trivial task even for experienced designers. As mentioned in Section 1.2.1, mesh generation is a trial-and-error process, requiring extensive human involvement. More importantly, the uncertainty associated with the discretization, *i.e.*, the numerical errors induced by the discretization, cannot be quantified easily just based on the user’s experience.

One thing to be noted is that these errors exist even in a single CFD simulation. Even before the ADODG, AIAA held a series of Drag Prediction Workshops (DPWs) [58, 59, 60, 61, 62], aiming to assess the state of the art computational methods as practical aerodynamic tools for aircraft force and moment prediction on industry-relevant geometries. Although the results were rather disappointing in the first workshop [58], it served as a reminder to the CFD community that CFD is not a matured field. As expected, the main differences in the submissions were due to modeling and discretization errors. Furthermore, an emerging consensus after the first three DPWs was that the dominant sources of error in aerodynamic drag prediction are related to spatial discretization error, which in turn stems largely from mesh resolution and mesh-quality issues [63]. Therefore, a greater emphasis has been placed on the mesh generation and mesh convergence studies in the fourth DPW. With more careful requirements on the computational meshes, the drag variation in DPW-IV was dramatically reduced, spanning only 41 drag counts if omitting the largest outlier, as shown in Figure 1.5 for the Common Research Model (CRM) geometry. In order to further reduce the mesh-induced error, a set of unified baseline families of multi-block structured meshes was provided in DPW-V, from which the overset and unstructured meshes are also defined. A even tighter scatter and smaller standard deviation were achieved compared to the results in DPW-IV, see Figure 1.5. However, the most recent DPW-VI, without using unified meshes, again shows comparable spread of drag predictions as DPW-IV [64, 65]. These facts reinforce the importance of mesh-related discretization errors and the difficulties in generating appropriate computational meshes in CFD applications. More importantly, without sufficient control of the discretization errors, improvements in physical models can be either inconsequential or impossible to assess in the presence of dominant spatial discretization errors [63].

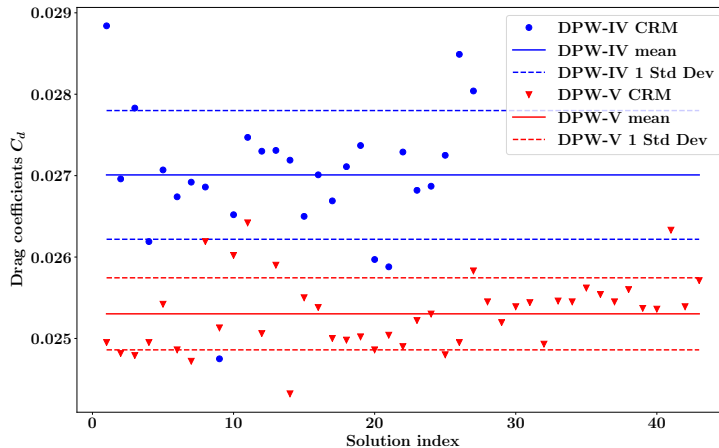


Figure 1.5: Drag coefficients scatter from DPW-IV and DPW-V, reproduced using data from [61, 62]. The largest outlier is removed from the original data set in both scatters. The offset of the mean values between DPW-IV and DPW-V is due to a slightly different CRM geometry, while the difference of the drag spread is mainly due to the computational meshes.

Rather than the absolute drag values, the incremental drag, such as the drag changes between two similar configurations, is of more importance in aerodynamic optimization. With best-practice meshing guidelines, a sufficiently fine mesh is able to capture the overall performance trend in the design space. Using the same CRM geometry, Lyu *et al.* [42] obtained similar designs using a set of meshes with different resolutions. However, the corresponding drag predictions and the chord-wise pressure distributions on the final designs are noticeably different. Moreover, in their study, the favorable “shock-free” design on a coarse mesh does not remain so when analyzed on finer meshes, indicating the discretization errors associated with the coarse mesh “shock-free” designs.

Although accurate incremental drag prediction is often less demanding than absolute drag prediction, it still depends heavily on the quality of the computational meshes. For some simulations involving complicated physics, even reliable incremental drag predictions can be difficult to obtain [63]. As shown in Figure 1.4, these inaccurate data will be used sequentially by the optimizer and can strongly affect the optimization results and the corresponding accuracy [66, 67]. Figure 1.6 shows two optimization examples where the results are strongly affected by the discretization errors, which will be studied in more details in Chapter 6. On the left, *i.e.*, Figure 1.6a, an optimization problem governed by a one-dimensional advection-diffusion partial differential equation (PDE) is considered. The numerical solution is obtained by solving the system on a coarse uniform spatial discretization. There is only one local minimum for the original continuous problem, while for the discrete numerical solution, we find a spurious local minimum purely caused by

numerical errors induced by the discretization. On the right, Figure 1.6b shows on top the optimized design of an airfoil optimization problem obtained on a coarse discretization, and on bottom the re-analysis of the same design on a refined discretization. Similar to the findings in [42], the “shock-free” design on the coarse discretization is associated with discretization errors, which indicates that the efforts to achieve “shock-free” features on the coarse discretization may be tied to discretization errors rather than physics. Therefore, if the discretization errors are not carefully controlled, the optimizer may (a) get stuck in a spurious optimum created by discretization errors, or (b) work on the discretization errors rather than on the physics and converge to an incorrect optimum.

Due to the strong influence of the discretization errors on both the absolute and incremental drag predictions, discretization errors have to be carefully controlled during optimization in order to converge to a reliable design. Another drawback of the current aerodynamic optimization is the open-loop after-design assessment paradigm, as shown in Figure 1.2. Despite the extensive human involvement in the design assessment, any design deficiency, including unmet performance or constraints, may not be easy to revert and may require a rerun of the whole optimization process with a better design or refined computational mesh. Considering the efforts required for mesh generation, the overall turnaround time may be undesirably factored. In order to avoid this as much as possible, the aerodynamic optimization paradigm needs to be improved such that the solution accuracy can be assessed during the optimization to ensure the reliability of the optimized design, at least in terms of the discretization errors.

1.2.3 Computational Efficiency

By virtue of the fast growing computational capacity, high-fidelity CFD simulations are now routinely carried out in aerodynamic optimizations. Nevertheless, high-fidelity aerodynamic optimization still remains computationally taxing because each evaluation of the objective function requires an expensive high-fidelity CFD solution. Although high-fidelity aerodynamic optimization enables aircraft engineers to perform detailed designs and to extend their intuition and experience in unconventional configurations, the computational resources needed for high-fidelity design preclude its wide-spread use. The term “high-fidelity” often refers to accurate physical models and high mesh resolutions. As mentioned in Section 1.2.2, high-fidelity models like DNS and LES are currently too expensive for optimization problems, RANS models considering transition effects and detached eddy simulation (DES) might be the next step of aerodynamic optimizations. On the other hand, “high-fidelity” refers to using sufficiently fine computational meshes. Due to the high solution sensitivity to the computational mesh, “high-fidelity” simulations

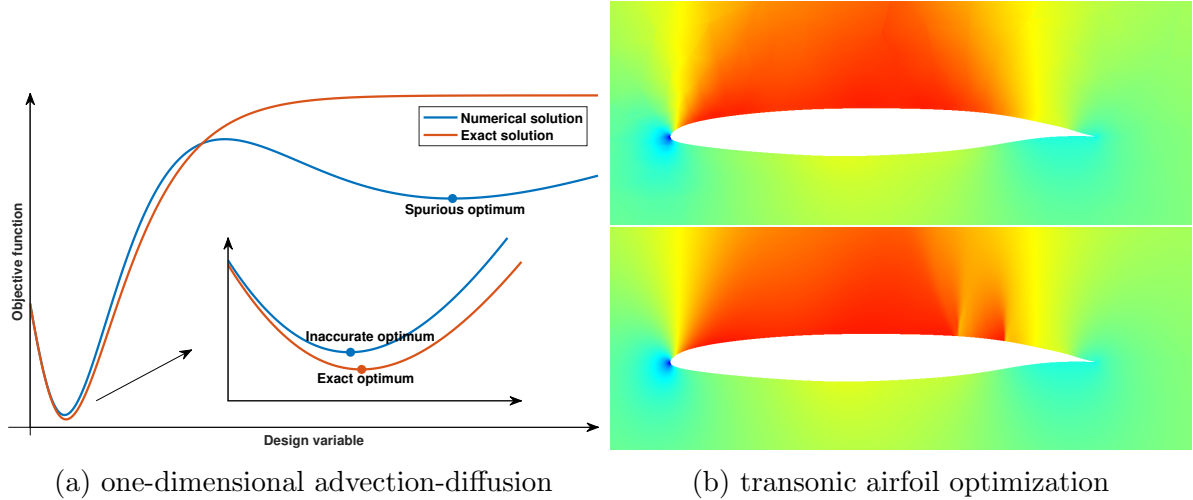


Figure 1.6: Examples of discretization error effects in optimization. The left figure shows the numerical and exact objective functional versus design variable in an optimization governed by a one-dimensional scalar advection-diffusion PDE. The right figure shows the Mach number contour of the optimized airfoil shape obtained on a coarse discretization, and its reanalyzed solution on a refined discretization; The Mach number contour range is $[0, 1.35]$.

often come with low confidence if no *a posteriori* analysis is performed. One alternative can be generating meshes that are extra fine to ensure solution accuracy. However, in this case, the computational expenses quickly makes large-scale aerodynamic optimization computationally intractable or impossible.

In order to save the computational burden of high-fidelity optimization, multifidelity optimization has been widely used because of its potential of reducing the overall computational costs by using cheaper low-fidelity simulations at the early stages of the optimization, either through lower-fidelity physical models or via coarse computational meshes. However, often there is no readily available knowledge about the confidence of the lower-fidelity simulations, such that the tolerances during the optimization are either always set to be the ultimate optimization tolerance, or varied purely based on users' experience or intuition. Although the multi-fidelity optimization is in general more efficient than traditional fixed-fidelity optimization, the current setup may hinder its potential of saving computational resources. On the one hand, if the low-fidelity optimization tolerance is set to be too loose, the optimizer will not be able to take advantage of cheaper low-fidelity computations to improve the design. On the other hand, if the optimization tolerance is set to be too tight at the lower fidelity, the optimizer may work on the numerical errors instead of physics to improve the design. Consequently, the optimization on the highest fidelity has to work on undoing these incorrect design modifications to bring the design

back to physically optimal, as shown in Figure 1.6b. In order to improve the computational efficiency in aerodynamic shape optimizations, a more appropriate strategy of actively varying the optimization tolerance during the optimization shall be developed.

1.3 Opportunities for Adaptive CFD in Aerodynamic Optimization

As discussed in the previous sections, several key challenges for current aerodynamic optimizations are: design automation, solution reliability, and computational efficiency. A cure for these obstacles is using high-order output-based adaptive CFD methods to achieve high-fidelity aerodynamic optimization. The favorable computational efficiency of high-order methods helps the optimization objective and constraint outputs converge faster than traditional second-order CFD methods. Output error estimates, on the other hand, provide a measure of confidence in the solutions during the design process. The capability of refining the mesh during optimization allows starting the optimization with fairly coarse meshes, significantly reducing the efforts in optimization setup. Moreover, the adaptive meshes offer various fidelity, combined with output error estimates, giving a systematic and robust way of controlling the fidelity switch in a multifidelity setting. This section gives a brief introduction of high-order CFD methods, output-based error estimation, and mesh adaptation, as well as the strategy for applying them in aerodynamic optimization problems.

1.3.1 High-Order CFD Methods

Generally speaking, the discretization error in a CFD solution is on the order of $\mathcal{O}(h^r)$, where h is the characteristic mesh size and r is the solution error convergence rate, *i.e.*, the order of the accuracy for the CFD method. A CFD method is considered high-order if it possesses an order of accuracy greater than two, $r > 2$. Therefore, in order to achieve similar accuracy, low-order methods will require more refined meshes, while high-order methods can use coarser meshes by taking advantage of higher convergence rates. The time step restriction, which is often inversely proportional to the mesh size, makes the high-order methods even more favorable in time-dependent simulations. Despite more complicated implementations, high-order methods are in general more efficient for smooth problems than traditional prevalent second-order finite-volume methods. For non-smooth problems, *e.g.*, flows involving shocks, high-order methods are not superior than low-order methods as the accuracy is vastly reduced due to the non-smoothness present in the solution. Fortunately, combined with solution-based adaptation, high-order methods are

able to recover the desired accuracy by isolating the non-smooth flow features. Although high-order methods are widely agreed to be more efficient than second-order finite volume methods, the lack of stability and robustness, as well as the software complexity when adaptation is adopted, have largely impeded their recognition in industry [68, 69]. On the other hand, the dominance of second-order methods in current CFD applications is primarily due to their robustness and high flexibility, which have been made possible by a considerable amount of research effort from the 1970s to the 1990s [29].

Although the current workhorse of CFD in aerospace applications, *e.g.*, aerodynamic optimization, is still the second-order finite-volume method, emerging high-fidelity requirements in many applications have brought to bear the importance of high-order methods. For example in aeroacoustic predictions and transition simulations, we are interested in small perturbations that are orders of magnitude smaller than the mean flow quantities. Over-dissipation of second-order methods tends to kill these small perturbations quickly and leads to inaccurate predictions. These are of particular interest for future aerodynamic optimization, since noise emission will be one of the important metrics in future aircraft design as mentioned in Section 1.1, and transitional modeling is essential to reduce the errors in physical models to achieve reliable designs.

During the past several decades, much attention in CFD developments has been paid to high-order methods, including spectral difference [70, 71, 72], spectral volume [73, 74], flux reconstruction or correction procedure via reconstruction [75, 76, 77, 78], continuous Galerkin (CG) [79, 80], and discontinuous Galerkin (DG) [81, 82, 83, 84, 85]. Among them, DG offers several advantages and is the discretization utilized in this work. First, compared to finite-volume and finite-difference methods, the variational formulation of the DG method lays a rigorous mathematical foundation for output error estimation and mesh adaptation. In addition, finite-volume like numerical fluxes adopted in DG buy its great stability for convection terms, which dominate in many aerospace engineering applications. Moreover, compact stencils offered by DG make it easily parallelizable on current multi-core processors. In addition, the high arithmetic intensity and local memory access pattern of DG make it well-suited for fast-paced graphics processing units (GPUs) and emerging heterogeneous high performance computing (HPC) architectures, which have been considered as a key to achieving petascale and exascale computing.

1.3.2 Output Error Estimation

As discussed in Section 1.2.2, quantifying the uncertainty due to discretization errors is essential for successful use of CFD in aerodynamic optimization. Unfortunately, this liability cannot be managed easily even by expert practitioners. In particular, an error

estimate of the outputs used in aerodynamic optimization ensures that the predictions of CFD solutions are only used to the limits of their accuracy, which can significantly reduce the chances of converging to spurious optima related to numerical errors. On the other hand, reliable error estimates give a measure of confidence for the designs obtained by aerodynamic optimization. However, error estimation itself cannot ensure the design reliability during optimization unless used in conjunction with adaptation to actively control the errors. Active error estimation and error control through adaptation are the keys to avoiding defect designs and after-design trade-offs. A brief review of using output-based error estimation and mesh adaptation in aerodynamic optimization is given in Section 1.3.4.

Since the exact solutions of flow governing equations are largely unavailable, discretization error in the numerical solution can only be estimated. Error estimation often falls into two categories: *a priori* and *a posteriori*. The former approach is heavily used in the early development of numerical methods, characterizing the convergence and stability of the methods. However, *a priori* error estimation often relies on parameters not known *a priori*, leaving the error bounds of limited usefulness in practice. In contrast, *a posteriori* error estimation is performed after obtaining the discrete numerical solution. The solution error can be derived through interpolation theory *a priori*, while the unknown parameters such as the solution Hessian can then be approximated by available numerical solutions [86]. Zienkiewicz and Zhu proposed a method to estimate the solution error by comparing the numerical solution with its reconstructed counterpart [87], which can be obtained by, for example, smoothing the solution gradient. Or alternatively, the solution error can be measured under the operators defined by the continuous governing equations or discretized ones on a finer discretization, often termed as residual errors [88, 89]. These methods all focus on estimating the solution error norms, which are also called *energy-based* estimators in the finite-element community. Such types of error estimators are covered in depth by Ainsworth and Oden [90, 91]. For the performance and robustness of different energy-based estimators, see studies conducted by Babuška *et al.* [92, 93].

The error estimated by the energy norm usually provides effective local error indicators for adaptation purposes. However, it relies on the assumption that local error only depends the local resolution, which holds well for elliptic problems such as those of the elasticity equations. However, for hyperbolic systems, such as high-speed flow simulations, errors can propagate by the convection-dominant nature of the system, making the prediction of regions requiring high resolution non-intuitive. To control local errors in hyperbolic systems, we need to trace the source of the error which may be far away, *e.g.*, some errors occurring down-stream may heavily depend on the accuracy of up-stream solutions.

Furthermore, often in engineering applications, we are only interested in some scalar outputs, *i.e.*, drag and lift, instead of the bulk flow solutions. Although outputs like drag and lift only depend on the solution integral on the geometry surface, they are sensitive to perturbations far away from the object surface, *e.g.*, ahead of the object and in its wake. Luckily, adjoint variables are able to identify the sensitivity of the output with respect to local solution errors, *i.e.*, flow equation residuals, providing a robust way of accounting for the error transport effects in hyperbolic systems. The adjoint variables weight the local flow residual to form an error measure of the output of interest, which can be used to provide error bounds or pure signed corrections of the output. Adjoint-based output error estimation, also known as the *dual weighted residual (DWR)* method, was pioneered by Becker and Rannacher [94], and Pierce, Giles, and Suli [95, 96]. The straightforward localization of the dual-weighted residual makes it easy to identify the regions that are important for accurate output predictions. Solution-adaptive methods via adjoint-based output error estimation have dramatically improved the accuracy and efficiency of CFD [85, 97, 98, 99, 100]. A detailed review focusing on aerospace applications is given by Fidkowski and Darmofal [101].

Despite the effectiveness of adjoint-based error estimation, it requires solving a dual linear system of the same size, or larger when solving on enriched spaces, as the flow primal problem. The additional memory and computational costs associated with the adjoint solutions, in addition to the implementation costs, hinder the effective use of adjoint-based error estimation methods in unsteady problems or in a many-query setting. More recently, error surrogate models based on machine learning techniques have received much attention, largely because of their non-intrusive nature and fast on-line evaluations. Several contributions have been made in error modeling for parameterized reduced-order models (ROMs) [102, 103], and the ideas have been extended to estimate discretization-induced errors [104]. Efforts have also been devoted to predicting the errors in flow solutions and the outputs of interest obtained on coarse computational meshes [105, 106], and the models have been used to guide the selection of a set of *a priori* meshes [107]. Nonetheless, in these studies, no output error indicator is provided to perform mesh adaptation. Manevitz *et al.* [108] used neural networks to predict the solution gradients in time-dependent problems, which then provided an indicator to drive the mesh adaptation. However, feature-based adaptive indicators are generally not as effective as adjoint-based indicators, especially for functional outputs and problems with discontinuities [109, 101]. Furthermore, these works rely on a set of user-selected local features (feature engineering) to construct the model, requiring either expert knowledge or fine tuning. Moreover, due to the local nature of the selected features (although some neighboring information comes

in with the gradient features), these models either largely ignore the error transport, and thus are not expected to be effective for convection-dominated problems, or still require the adjoint variables to bring in the global sensitivity information.

1.3.3 Mesh Adaptation

Error estimation gives a measure of confidence in the numerical solutions, and, if appropriately localized, is able to identify the regions that produce the majority of the errors. In order to control the solution accuracy, a natural idea is to enrich or modify the solution space in areas that are responsible for most of the errors. In high-order finite-element methods, the solution space depends on the local mesh resolution h and the approximation order p . Thus, solution approximation can often be improved through refining the local mesh resolution (h -adaptation), increasing the local approximation order (p -adaptation), or a combination of both (hp -adaptation). Standard adaptive CFD frameworks often adopt a feedback loop: the discretized governing equation is solved in the space defined by the current mesh and approximation order distribution; then the discretization error is estimated and the solution space is enriched or modified correspondingly, and a new solution is obtained on this space. This process repeats until some error or cost tolerance is met, ensuring solution reliability in an automatic way with minimal human interference.

p -adaptation is more efficient in smooth problems and requires less implementation effort compared to h -adaptation, as local approximation order can be easily increased. The drawbacks, however, are also evident. The accuracy gains are limited in regions where the solution exhibits discontinuities. Indeed, the order increase can make high-order methods even more vulnerable to stability issues around discontinuities. On the other hand, h -adaptation is more effective in isolating the discontinuities, though the order of accuracy is still bounded by the approximation order. Meanwhile, the mesh modification, either through local mesh operations or relying on global re-meshing, significantly increases the software complexity. hp -adaptation, a combination of the two, if implemented appropriately, is able to achieve great error convergence by increasing local approximation order, *i.e.*, p -adaptation, in regions where the solution is smooth while isolating the singularity or discontinuity through h -adaptation. However, automatic decision of performing p -adaptation or h -adaptation requires more algorithmic work.

Another important benefit of mesh adaptation, which is often understated, is that the adaptation procedure greatly reduces the time and effort required to generate appropriate meshes for complex geometries. Fairly coarse meshes that are easy to generate can be used to start the simulation. Adaptation will then put enough resolution in suitable

regions based on the localized output error estimates. Take the DPW-VI CRM geometry as an example. Creation of a sequence of fixed meshes requires an expert user over 3 weeks to match the best-practice meshing guidelines. In contrast, adapted meshes can be generated without human intervention, requiring less than two days of user time to generate a coarse initial mesh and start the adaptive run [54].

In this work, only h -adaptation is performed, while the approximation order p is kept fixed and uniform in the mesh. The benefits of p -adaptation and hp -adaptation will be investigated in future work. As in aerodynamic optimization, outputs like drag and lift are of more interest than the flow solution, and therefore adjoint-based output error estimation is used to drive the mesh adaptation. The localized output error can provide effective indicators for mesh adaptation, yet no solution anisotropy information can be directly obtained in adjoint-based estimators. In areas featuring strong directional phenomena, anisotropic mesh elements are necessary to achieve better adaptation efficiency. Two methods are considered in this work to detect solution anisotropy. One is through the incorporation of solution interpolation errors, and the other is using an optimization framework built on a continuous mesh formulation. More details on this are given in Chapter 4. To further increase the computational efficiency and to reduce the implementation cost of the traditional adjoint-based mesh adaptation, machine learning techniques are also introduced to accelerate the mesh adaptation, which can be found in Chapter 8 and Chapter 9.

1.3.4 Aerodynamic Optimization With Adaptive CFD

Given the capacity of adapting meshes automatically and putting necessary resolution in regions responsible for discretization errors, error estimation and mesh adaptation are able to significantly reduce the optimization setup efforts, *i.e.*, mainly the initial mesh generation [54]. The systematic feedback control of the discretization error, on the other hand, ensures the accuracy of the design analysis during optimization. Meanwhile, the error estimates give a measure of confidence for the optimizer, such that the CFD results are only used to the limits of their accuracy. This “in-design” assessment, enabled by error estimation and mesh adaptation can significantly reduce the chance of converging to undesired designs due to numerical errors and increase the reliability of the optimized design. A new paradigm for aerodynamic optimization with error estimation and mesh adaptation is depicted in Figure 1.7, in which the user only needs to provide a coarse mesh and the optimization will output the optimized design with discretization error controlled through automatic error estimation and mesh adaptation. The details of the coupling between different optimization components are discussed in Chapter 5.

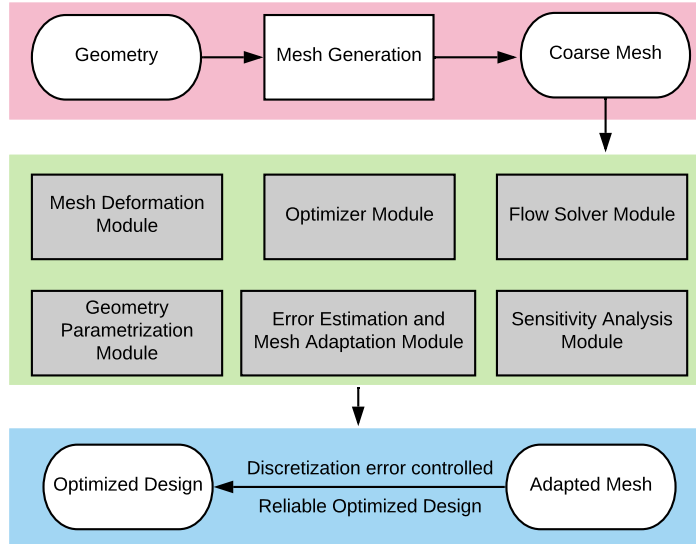


Figure 1.7: Aerodynamic optimization with error estimation and mesh adaptation.

The idea of coupling error estimation and mesh adaptation with gradient-based optimization is rather new. Energy-based estimators can be effective and robust for driving adaptation in elliptic PDEs like those of structural elasticity, and thus several contributions have been made to incorporate these approaches into structural optimization problems [110, 111, 112]. However for hyperbolic systems, such as those governing aerodynamic fluid flows, errors can propagate by the convection-dominant nature of the system, making energy-based type estimators inefficient, especially for scalar outputs. Therefore, adjoint weighted residual (AWR) is more popular for CFD applications, in which the sensitivity of the output to residual perturbation is identified by the adjoint solutions and the weighted residual is used to drive the adaptation.

The idea to combine adjoint-based output error estimation and mesh adaptation with traditional gradient-based optimization is natural, as both methods require output adjoint solutions, making the incorporation more efficient. Even though adjoint-based error estimation and mesh adaptation have been studied in depth and successfully demonstrated in many aerospace engineering problems, their application to optimization problems has received less attention. Lu [113] incorporated adjoint-based error estimation and p -adaptation into gradient-based optimization. The constraints are realized as simple quadratic penalty functions added to the objective. Progressive optimization is used with mesh adaptation based on the error of the penalized objective. Nemec and Aftosmis [114, 115] modified the penalty terms to avoid vanishing of the constraint error when the constraints are satisfied. Li and Hartmann [116] eliminated the constraint by trimming with an individual design variable, and introduced a multi-target adaptation

algorithm in which mesh adaptation targets the objective and constraint outputs equally on a fixed fidelity. Hicken and Alonso [66] used the gradient norm error as the refinement indicator to actively control the first-order optimality condition, while higher-order derivatives needed to be approximated. In most previous works on optimization combined with error estimation, mesh adaptation has only been used to add refinement. However, in order to control the discretization error during the optimization, the mesh may be adapted in many areas that are important for the intermediate designs but are not necessary for the final optimal design, which may decrease the efficiency of the high-fidelity optimization. Therefore, to fulfill the potential of adaptive CFD in aerodynamic optimization, more advanced adaptation mechanics have to be used, such that mesh resolution can be not only refined but also redistributed to avoid unnecessary refinements for intermediate designs.

1.4 Thesis Overviews

1.4.1 Major Contributions

In order to tackle the problems mentioned above, the goal of this thesis is to develop methods to increase the reliability, automation and efficiency of the optimization process in aircraft design. The main contributions of the current work include:

- Developed a more appropriate error estimator for optimization objective output, taking into account the errors from constraint outputs as well.
- Incorporated adaptive CFD with traditional gradient-based optimization, enabling automated and reliable optimization.
- Proposed multifidelity optimization frameworks suitable for both error-based and cost-based type adaptation mechanics.
- Demonstrated potential benefits of cost-based adaptation mechanics, especially mesh optimization, to reduce unnecessary refinements for intermediate designs.
- Developed a mesh anisotropy detection model based on artificial neural networks (ANNs), accelerating the mesh optimization procedure and reducing the implementation cost associated with adaptation.
- Explored the feasibility of a convolutional neural network (CNN)-based error estimation and mesh adaptation approach, offering adaptive CFD for problems where adjoint variables are not available or expensive to solve.

1.4.2 Thesis Outline

The thesis outline is summarized in Figure 1.8. The remainder of the thesis starts with the introduction of the governing equations and the DG discretization in Chapter 2. Chapter 3 provides a brief review of the adjoint-based methods in both error estimation and sensitivity computation. In Chapter 4, we propose a coupled-adjoint approach to estimate the objective error in an optimization problem, which also accounts for the constraint output errors. The proposed error estimator is then incorporated into traditional gradient-based optimization in Chapter 5, in which both error-based and cost-based approaches are discussed. The developed framework is first demonstrated in single-point airfoil optimization problems in Chapter 6, followed by extensions to multipoint optimizations in Chapter 7. In order to accelerate the error estimation and mesh adaptation process, machine learning techniques are utilized in Chapter 8 and Chapter 9. Finally, in Chapter 10 we conclude the present work and discuss future research directions.

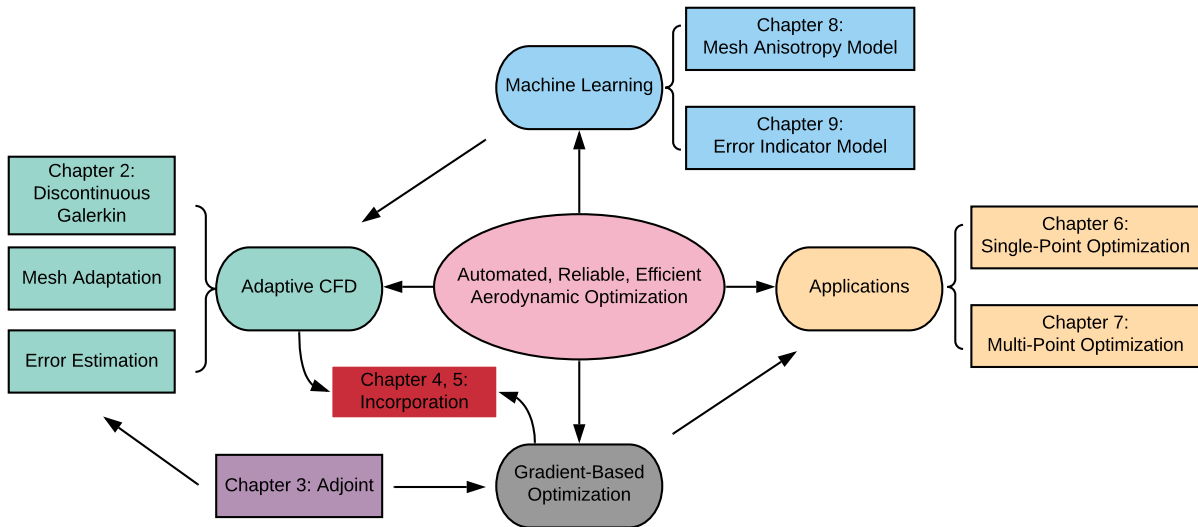


Figure 1.8: Thesis outline.

CHAPTER 2

Governing Equations and Discretization

Although the methods developed in this work are intended for fluid flow governing equations, specifically the compressible Navier-Stokes (NS) equations and their time averaged form, Reynolds-averaged Navier-Stokes (RANS) equations, they are formulated for a set of general conservation laws and can be applied to any system that admits a conservation form. In this chapter, we begin with a brief review of conservation laws, followed by the specific equations used for simulating the aerodynamic flows throughout this work. Then, we elaborate the high-order finite-element discretization method adopted for solving the equations of interest. The computational framework used in this work is based on previous and ongoing developments conducted in the computational fluid dynamics group (CFDG) ¹ at the University of Michigan, with a focus on output-based high-order adaptive CFD methods.

2.1 Equations and Notation

A general unsteady (time-dependent) conservation law can be written as a set of PDEs,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \quad (2.1)$$

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the state vector of rank s , which depends on the d -dimensional spatial coordinates $\vec{x} \in \mathbb{R}^d$ and the time $t \in \mathbb{R}$; $\vec{\mathbf{H}} \in \mathbb{R}^{d \times s}$ is the total flux and $\mathbf{S} \in \mathbb{R}^s$ is the source term when external forces or modeling terms are present, both possibly depending on the state \mathbf{u} and the state gradients $\nabla \mathbf{u}$. For systems involving both convection and diffusion, the total flux $\vec{\mathbf{H}}$ can be decomposed as

$$\vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u}) - \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u}), \quad (2.2)$$

¹<https://sites.google.com/a/umich.edu/cfdg>

where $\vec{\mathbf{F}} \in \mathbb{R}^{d \times s}$ is the inviscid or convective flux, and $\vec{\mathbf{G}} \in \mathbb{R}^{d \times s}$ is the viscous or diffusive flux. The viscous flux is often linearly dependent on the states gradient, such that the i^{th} component of $\vec{\mathbf{G}}$ can be written as

$$\mathbf{G}_i(\mathbf{u}, \nabla \mathbf{u}) = -\mathbf{K}_{ij} \partial_j \mathbf{u}, \quad i, j = 1, 2, \dots, d, \quad (2.3)$$

where $\mathbf{K}_{ij} \in \mathbb{R}^{s \times s}$ denotes the (i, j) component of the viscous diffusivity tensor. Eqn. 2.1-2.3 describe a general conservation law, where the quantity of the property remains unchanged unless external forces are exerted. In this work, we focus on physical conservation in fluid flow, specifically the compressible Navier-Stokes equations.

2.2 Compressible Navier-Stokes Equations

The NS equations describe the motion of a viscous fluid, in which the fluid properties are conserved during the transport of ordered motion of the flow *i.e.*, convection, and the random motion of the fluid particles, *i.e.*, diffusion. The NS equations arise from momentum conservation via Newton's second law of motion, considering both the convection and diffusion effects. Together with the conservation of mass and energy, the NS equations take the form of conservation laws and consist of $d + 2$ conserved states in d spatial dimensions. The original NS equations do not contain the source terms unless external sources are present, so that the equations can be written as

$$\begin{aligned} \partial_t \rho + \partial_j (\rho v_j) &= 0, & \text{mass conservation} \\ \partial_t (\rho v_i) + \partial_j (\rho v_i v_j + p \delta_{ij}) - \partial_j \tau_{ij} &= 0, & \text{momentum conservation} \\ \partial_t (\rho E) + \partial_j (\rho H v_j) - \partial_j (\tau_{ij} v_i - q_j) &= 0, & \text{energy conservation} \end{aligned} \quad (2.4)$$

where the variables are defined as

- ρ : density
- v_i : i^{th} component of velocity \vec{v}
- p : pressure
- E : specific (per unit mass) total energy
- H : specific (per unit mass) total enthalpy
- q_i : i^{th} component of heat flux \vec{q}
- τ_{ij} : viscous stress tensor

The heat flux \vec{q} is the conductive or diffusive heat flux, which obeys Fourier's law,

$$\vec{q}_i = -k_T \partial_i T, \quad (2.5)$$

where T is the temperature and k_T is the thermal conductivity. The specific total energy E and the specific total enthalpy H are defined as the fluid kinetic energy due to motion plus the fluid internal energy e or internal enthalpy h , respectively,

$$E = e + \frac{1}{2} |\vec{v}|^2, \quad (2.6)$$

$$H = h + \frac{1}{2} |\vec{v}|^2. \quad (2.7)$$

The internal energy e and enthalpy h , depend on the fluid material properties and the temperature T ,

$$e = c_v T, \quad (2.8)$$

$$h = e + \frac{p}{\rho} = c_p T, \quad (2.9)$$

$$\gamma = \frac{c_p}{c_v}, \quad (2.10)$$

where c_v and c_p are the specific heat capacity at constant volume and constant pressure respectively, and they can be related via a constant specific heat ratio γ .

The viscous stress tensor τ_{ij} was first derived by Stokes with analogy to Hookean elasticity,

$$\tau_{ij} = 2\mu \epsilon_{ij} + \lambda \delta_{ij} \partial_l v_l, \quad (2.11)$$

where λ is the *bulk viscosity coefficient* and μ is the *second viscosity coefficient*, or more often referred to as *dynamic viscosity*. The strain rates tensor, ϵ_{ij} , describes the deformations for the fluid, taking the form

$$\epsilon_{ij} = \frac{1}{2} (\partial_i v_j + \partial_j v_i). \quad (2.12)$$

To remove the ambiguous difference between the mean mechanic pressure of the fluid and its thermodynamic counterpart p , Stokes' hypothesis is adopted,

$$\begin{aligned} \bar{p} - p &= -(\lambda + \frac{2}{3}\mu) \partial_l v_l = 0, \\ \Rightarrow \lambda + \frac{2}{3}\mu &= 0. \quad \text{Stokes' hypothesis} \end{aligned} \quad (2.13)$$

With the thermodynamic and mechanic definitions above, the NS equations feature $d + 3$ unknowns (states ρ , ρv_i , ρE and the pressure p) with only $d + 2$ equations, the perfect gas law is used to close the system, *i.e.*, relating the pressure to the states,

$$p = \rho RT, \quad \text{perfect gas law} \quad (2.14)$$

$$\Rightarrow p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho|\vec{v}|^2\right), \quad (2.15)$$

where R is the gas constant. Eqn. 2.15 is derived from Eqn. 2.8–2.10, with the perfect gas law in Eqn. 2.14.

Following the notation presented in Section 2.1, we can write the NS equations in a vector form as Eqn. 2.1,

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho v_i \\ \rho E \end{bmatrix}, \quad \vec{\mathbf{F}} = \begin{bmatrix} \rho v_j \\ \rho v_i v_j + p \delta_{ij} \\ \rho H v_j \end{bmatrix}, \quad \vec{\mathbf{G}} = \begin{bmatrix} 0 \\ \tau_{ij} \\ \tau_{ij} v_i - q_j \end{bmatrix}.$$

In this work, air properties with temperature-dependent dynamic viscosity (Sutherland's law) are defined as follows:

$$\text{dynamic viscosity: } \mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{1.5} \left(\frac{T_{\text{ref}} + T_s}{T + T_s} \right)$$

$$T_{\text{ref}} = 188.15K, \quad T_s = 110K$$

$$\mu_{\text{ref}} = 1.789 \times 10^{-5} N \cdot s/m^2$$

$$\text{Prandtl number: } Pr = \frac{\mu c_p}{k_T} = 0.71$$

$$\text{thermal conductivity: } k_T = \frac{\mu c_p}{Pr}$$

$$\text{specific heat: } c_v = \frac{R}{\gamma - 1}, \quad c_p = \gamma c_v = \frac{\gamma R}{\gamma - 1}$$

$$\text{gas constant: } R = 8.314 J/(mol \cdot K)$$

$$\text{specific heat ratio: } \gamma = 1.4$$

Once the gas constant R is given, the fluid properties are fully defined given the formulas above. However, often in aerodynamic analysis, we are interested in non-dimensionalized quantities, *i.e.*, drag and lift coefficients, such that CFD codes do not necessary adopt the physical units. To establish a unit-independent system, two more non-dimensional

quantities are defined, namely the Reynolds number (Re) and the Mach number (M),

$$Re = \frac{\rho|\vec{v}|L}{\mu}, \quad (2.16)$$

$$M = \frac{|\vec{v}|}{a}, \quad a = \sqrt{\gamma RT} \quad (2.17)$$

where L is the reference length scale, *e.g.*, chord length of an airfoil, and a is the speed of sound which depends on the local temperature. For easier post-processing purposes, only the non-dimensional quantities like Pr , Re , and M are defined rather than the physical fluid properties. As long as R is defined, the unit system can be easily converted to the physical unit system.

2.3 Reynolds-Averaged Navier-Stokes Equations

The NS equations are easy to solve when the flow is laminar, such that the flow field is characterized by the object shape and dimension. As the Reynolds number increases, *i.e.*, the viscous effects become smaller compared to inertial forces, small perturbations cannot be damped out effectively and may get amplified, leading to unstable flow and eventually transition to turbulent flow. When turbulence commences, large variations in the velocity field create eddies that transport with the bulk flow motion and break down into smaller eddies until the length scale is small enough for viscous effects to dissipate the kinetic energy associated with them. This is often termed as the *energy cascade* process. The fluctuations in the velocity and pressure field caused by these eddies are inherently unsteady, and any initial condition perturbations can lead to chaotic results. Thus to solve the NS equations in high Reynolds number regimes, often of interest in aeronautical applications, the simulation has to resolve the eddies with different scales and trace their evolution, which often requires computational meshes that are intractable [69]. Therefore, for practical engineering simulations in high Reynolds number flows, turbulence has to be modeled.

In particular, Reynolds-averaged Navier-Stokes equations model turbulence by decomposing the primitive flow quantities into mean values and their time fluctuating parts, *e.g.*, the flow velocity \vec{v} can be decomposed into the mean velocity $\bar{\vec{v}}$ and the fluctuations \vec{v}' . With a time averaging process on the NS equations, Eqn. 2.4 becomes a set of PDEs governing the mean flow quantities, *i.e.*, the RANS equations. However, the time averaging decomposition does not provide closures for the high-order moments, *i.e.*, high-order means, for the fluctuating parts. The most important term that need to be closed is the Reynolds' stress term, $\overline{\rho v'_i v'_j}$, which results from flow momentum convection but behaves

like a stress term. In this work, the closure of the RANS equations is accomplished by the Spalart-Allmaras (SA) turbulence model [117] with negative viscosity modification [118]. More details for deriving and implementing the RANS-SA model can be found in the National Aeronautics and Space Administration (NASA) Turbulence Modeling Resource website ². Presently, we just give the essential equations of the RANS-SA model. The governing equations in Eqn. 2.4 are now with respect to the mean states and can be written as ³,

$$\begin{aligned}\partial_t \bar{\rho} + \partial_j(\bar{\rho} v_j) &= 0, \\ \partial_t(\bar{\rho} v_i) + \partial_j(\bar{\rho} v_i v_j + p \delta_{ij}) - \partial_j \tau_{ij}^{\text{RANS}} &= 0, \\ \partial_t(\bar{\rho} E) + \partial_j(\bar{\rho} H v_j) - \partial_j(\tau_{ij}^{\text{RANS}} v_i - q_j^{\text{RANS}}) &= 0.\end{aligned}\tag{2.18}$$

τ^{RANS} is the effective viscous stress tensor that accounts for both the physical viscous stress and the Reynolds' stress, while q_j^{RANS} is the effective heat flux considering the turbulent transport effects. These are related to mean flow quantities by

$$\tau_{ij}^{\text{RANS}} = 2(\mu + \mu_t) \left(\epsilon_{ij} - \frac{1}{3} \partial_l v_l \delta_{ij} \right),\tag{2.19}$$

$$q_j^{\text{RANS}} = -(k_T^{\text{RANS}}) \partial_j T = -c_p \left(\frac{\mu}{\text{Pr}} + \frac{\mu_t}{\text{Pr}_t} \right) \partial_j T,\tag{2.20}$$

where μ_t is the eddy viscosity by analogy to the molecular diffusion, and Pr_t is the turbulent Prandtl number. The SA model introduces a turbulent kinematic viscosity, $\tilde{\nu}$, to model the eddy viscosity,

$$\mu_t = \begin{cases} \rho \tilde{\nu} f_{v1}(\chi), & \tilde{\nu} \geq 0 \\ 0, & \tilde{\nu} < 0 \end{cases}, \quad f_{v1}(\chi) = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu},\tag{2.21}$$

where $\nu = \mu/\rho$ is the physical kinematic viscosity, and χ is the non-dimensionalized turbulent kinematic viscosity. To close the RANS-SA system, one transport equation for $\tilde{\nu}$ is given

$$\partial_t(\rho \tilde{\nu}) + \partial_j(\rho \tilde{\nu} v_j) = \frac{\rho}{\sigma} \partial_j(\eta \partial_j \tilde{\nu}) + \frac{c_{b2} \rho}{\sigma} \partial_j \tilde{\nu} \partial_j \tilde{\nu} + P - D,\tag{2.22}$$

or in a conserved form for $\rho \tilde{\nu}$, using Eqn. 2.18,

$$\partial_t(\rho \tilde{\nu}) + \partial_j(\rho \tilde{\nu} v_j) - \partial_j \left(\frac{\rho \eta}{\sigma} \partial_j \tilde{\nu} \right) = -\frac{\rho \eta}{\sigma} \partial_j \rho \partial_j \tilde{\nu} + \frac{c_{b2} \rho}{\sigma} \partial_j \tilde{\nu} \partial_j \tilde{\nu} + P - D,\tag{2.23}$$

²<https://turbmodels.larc.nasa.gov>

³The conserved states are all mean states in RANS equations; the mean value notation $\bar{\cdot}$ is omitted here for simplicity.

where P and D are, respectively, the production and destruction terms for $\tilde{\nu}$, c_{b2} and σ are model constants. $\rho\eta/\sigma$ resembles the diffusion coefficient for the turbulent kinematic viscosity, in which η is defined as

$$\eta = \nu + \nu', \quad \nu' = \begin{cases} \nu\chi, & \chi \geq 0 \\ \nu f_n(\chi), & \chi < 0 \end{cases}, \quad f_n(\chi) = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}. \quad (2.24)$$

The production term P and the destruction term D dictate, respectively, the increase and the decrease of the turbulent kinematic viscosity. The production term, P , is defined as

$$P = \begin{cases} c_{b1}\tilde{S}\rho\tilde{\nu}, & \chi \geq 0 \\ c_{b1}S\rho\tilde{\nu}, & \chi < 0 \end{cases}, \quad \tilde{S} = \begin{cases} S + \bar{S}, & \bar{S} \geq -c_{v2}S \\ S + \frac{S(c_{v2}^2S + c_{v3}\bar{S})}{\bar{S}(c_{v3} - 2c_{v2}) - \bar{S}}, & \bar{S} < -c_{v2}S \end{cases}. \quad (2.25)$$

where $S = \sqrt{2\Omega_{ij}\Omega_{ij}}$ is the magnitude of the vorticity tensor, $\Omega_{ij} = (\partial_i v_j + \partial_j v_i)/2$, and \tilde{S} is the modified vorticity magnitude with the near-wall correction \bar{S} given by

$$\bar{S} = \frac{\tilde{\nu}f_{v2}}{k^2d^2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \quad (2.26)$$

where d is the distance to the nearest wall. The destruction term D is modeled as

$$D = \begin{cases} c_{w1}f_w \frac{\rho\tilde{\nu}^2}{d^2}, & \chi \geq 0 \\ -c_{w1} \frac{\rho\tilde{\nu}^2}{d^2}, & \chi < 0, \end{cases} \quad (2.27)$$

where the coefficients are

$$f_w = g \left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right)^{1/6}, \quad g = r + c_{w2}(r^6 - r), \quad r = \frac{\tilde{\nu}}{\tilde{S}k^2d^2}. \quad (2.28)$$

The SA model constants (parameters) are summarized as follows,

$$\begin{array}{ll} c_{b1} & = 0.1355 & c_{w2} & = 0.3 \\ \sigma & = 2/3 & c_{w3} & = 2 \\ c_{b2} & = 0.622 & c_{v1} & = 7.1 \\ \kappa & = 0.41 & c_{v2} & = 0.7 \\ c_{w1} & = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma} & c_{v3} & = 0.9 \\ Pr_t & = 0.9 & c_{n1} & = 16 \end{array}$$

Scaling of $\tilde{\nu}$

The turbulent kinematic viscosity, $\tilde{\nu}$, will in general be orders of magnitude smaller than other state components, causing convergence issues for nonlinear solvers. In practice, $\tilde{\nu}$ is scaled to match the order of the other states [119],

$$\tilde{\nu}_s = \frac{\tilde{\nu}}{c_s}, \quad (2.29)$$

where c_s is the scaling factor, usually on the order of \sqrt{Re} . Eqn. 2.23 is also scaled by c_s ,

$$\partial_t(\rho\tilde{\nu}_s) + \partial_j(\rho\tilde{\nu}_s v_j) - \partial_j \left(\frac{\rho\eta}{\sigma} \partial_j \tilde{\nu}_s \right) = -\frac{\rho\eta}{\sigma} \partial_j \rho \partial_j \tilde{\nu}_s + c_s \frac{c_{b2}\rho}{\sigma} \partial_j \tilde{\nu}_s \partial_j \tilde{\nu}_s + \frac{P - D}{c_s}, \quad (2.30)$$

the scaling of the SA working variable $\tilde{\nu}$ helps converge the residual of the closure equation to the similar level of other components.

2.4 Discontinuous Galerkin Discretization

In order to solve the compressible NS and the RANS equations above, we need to discretize them in both space and time. In this work, we only consider steady state solutions, *i.e.*, $\partial/\partial t = 0$, although for clarity in exposition, we leave the unsteady term in the analysis. Finite element methods are used to discretize the system of equations in space. More specifically, a discontinuous Galerkin discretization is adopted to support high-order accuracy and mesh refinement. However, the framework developed in this thesis can be applied to other discretizations supporting output-based error estimation and mesh adaptation.

2.4.1 Solution Approximation

Consider a set of PDEs for states \mathbf{u} on a computational domain Ω , subject to well-posed boundary conditions. We first partition the domain into a tessellation \mathcal{T}_h , *i.e.*, the computational mesh, which consists of N_e non-overlapping elements, Ω_e ,

$$\mathcal{T}_h = \left\{ \Omega_e : \bigcup_{i=1}^{N_e} \Omega_e = \Omega, \bigcap_{i=1}^{N_e} \Omega_e = \emptyset \right\}. \quad (2.31)$$

An example unstructured computational mesh for flow over a smooth bump is shown in Figure 2.1. This mesh contains both linear and high-order curved elements to represent curved boundaries. High-order elements for curved boundary representations are often

necessary to fulfill the accuracy benefits of high-order discretizations [68] and to avoid spurious oscillations, especially in methods like DG where neighboring geometry information does not come into the local weak form directly [120].

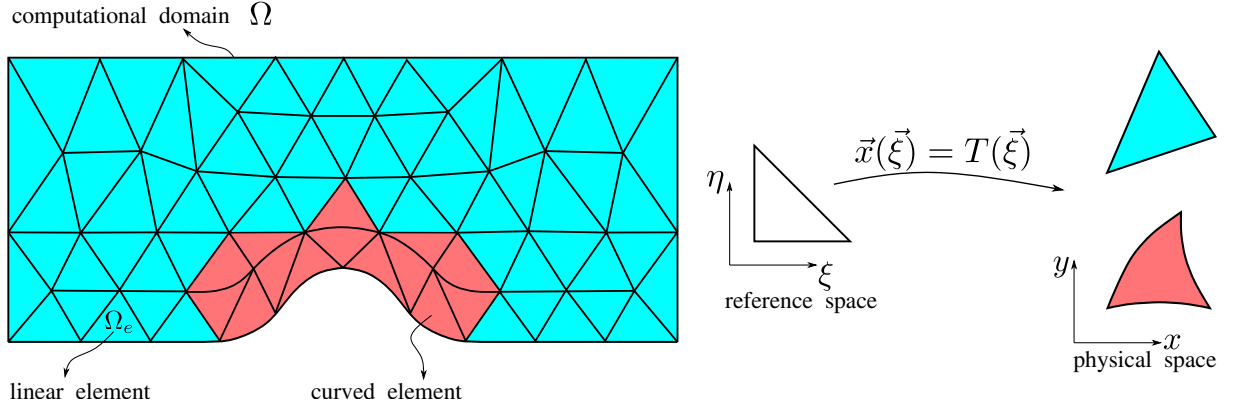


Figure 2.1: An example unstructured mesh \mathcal{T}_h for flow over a smooth bump in the computational domain Ω .

In the DG method, we seek a numerical representation of the solution, \mathbf{u}_h , in which each state component inside element Ω_e is approximated by a set of polynomials of order p_e , denoted by $\mathcal{P}^{p_e}(\Omega_e)$ ⁴. Formally, the solution \mathbf{u}_h lies in the *approximation space* \mathcal{V}_h , $\mathbf{u}_h \in \mathcal{V}_h = [\mathcal{V}_h]^s$, where

$$\mathcal{V}_h = \{v \in L^2(\Omega) : v|_{\Omega_e} \in \mathcal{P}^{p_e} \forall \Omega_e \in \Omega\}. \quad (2.32)$$

A key feature of DG is that solution continuity is not enforced across adjacent elements, *i.e.*, \mathcal{V}_h is only continuous inside an element. A comparison of DG approximation and its continuous counterpart, continuous Galerkin (CG) approximation, for a state component u , is shown in Figure 2.2 to highlight the discontinuous nature of DG methods. The elemental approximation order p_e is not necessarily uniform in the mesh, allowing for p -adaptation, while in this work we restrict to fixed order p approximation throughout the entire mesh. Thus for the rest of the paper, we denote the solution as \mathbf{u}_h^p where the subscript h specifically refers to the computational mesh and p is the approximation order which will be omitted for simpler illustration if necessary.

⁴The polynomials are defined in reference space and for curved elements they may not remain order p_e polynomials after the mapping to physical space, as shown in Figure 2.1, although additional mapping can be performed to recover the polynomial orders [121].

⁵The order p in the figures represent the minimum degree of polynomials required to represent the local solutions, although higher order polynomials can always recover the lower order profiles, *e.g.*, any linear solution profile can be represented by polynomials of order 1 or higher.

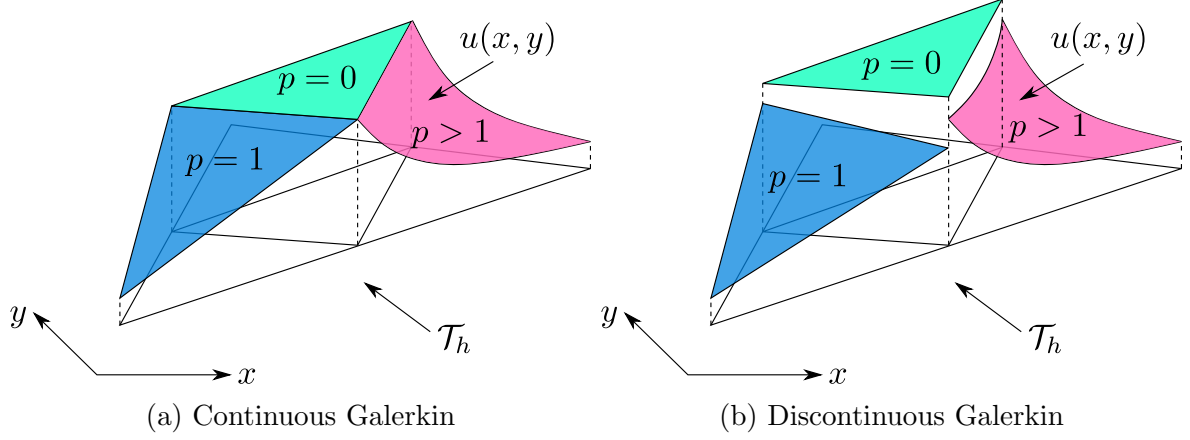


Figure 2.2: Solution approximation using the CG and DG methods ⁵. Though no continuity constraints are enforced at adjacent element boundaries, the inter-element flux is uniquely defined, as in finite volume methods.

2.4.2 Weak Form

With a properly defined approximation space, a DG method then seeks a solution, \mathbf{u}_h , that satisfies the *weak form* of the governing equations. The weak form enforces the orthogonality of the strong form PDEs with respect to *test functions*, \mathbf{w}_h , lying in the *test space* that is the same as the approximation space in the context of Galerkin methods, *i.e.*, $\mathbf{w}_h \in \mathcal{V}_h$. The weak form of Eqn. 2.1 follows from taking the inner products with test functions, integrating by parts and coupling elements via uniquely defined inter-element *numerical fluxes*. The resulting weak form can be written as

$$\sum_{e=1}^{N_e} \int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}_h}{\partial t} d\Omega + \mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h) = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h, \quad (2.33)$$

where $\mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h)$ is the semilinear form of the spatial residual term,

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h) = \sum_{e=1}^{N_e} [\mathcal{R}_{h,I}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) + \mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) + \mathcal{R}_{h,S}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e})] \quad (2.34)$$

where the subscripts I , V and S denote the contributions from the inviscid flux, viscous flux and the source terms, respectively.

2.4.2.1 Inviscid Discretization

Applying integration by parts for the inviscid flux term, we can obtain the inviscid residual as

$$\mathcal{R}_{h,I}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) = - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{F}} d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-) \cdot \vec{n} dS \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \quad (2.35)$$

On the elemental boundary $\partial\Omega_e$, a face in three dimensions or an edge in two dimensions, $\{\cdot\}^+$ and $\{\cdot\}^-$ terms denote, respectively, the quantities from inside the element or its neighbor element sharing $\partial\Omega_e$. $\{\cdot\} \cdot \vec{n}$ represents the uniquely defined normal numerical flux on the element interface, where \vec{n} is the outward-pointing normal vector on $\partial\Omega_e$. More specifically in Eqn. 2.35, $\widehat{\mathbf{F}} \cdot \vec{n}$ is the numerical inviscid flux depending on both the states from the element interior \mathbf{u}^+ and the states from its neighbor \mathbf{u}^- . The computation of numerical fluxes requires solving exactly or approximately the local Riemann problem on the element interface, which luckily has been well studied for finite volume methods during the 1980s [29]. We use the Roe approximate Riemann solver [122] with an entropy fix, which takes the form

$$\widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-) \cdot \vec{n} = \frac{1}{2}[\mathbf{F}(\mathbf{u}_h^+) + \mathbf{F}(\mathbf{u}_h^-)] - \frac{1}{2}\mathbf{R}|\mathbf{\Lambda}|\mathbf{L}(\mathbf{u}_h^+ - \mathbf{u}_h^-), \quad (2.36)$$

where \mathbf{L} and \mathbf{R} are the left and right eigenvector matrices corresponding to the eigenvalue matrix $\mathbf{\Lambda}$ of the flux Jacobian matrix $\partial\mathbf{F}/\partial\mathbf{u}$, which is linearized at the Roe-averaged state $\tilde{\mathbf{u}}(\mathbf{u}_h^+, \mathbf{u}_h^-)$.

2.4.2.2 Viscous Discretization

The treatment of the viscous flux is not straightforward for DG methods, as the diffusion terms favors a solution that is everywhere differentiable which does not naturally hold for DG solutions. In order to obtain a stable discretization for viscous terms, we use a “mixed” formulation which turns the viscous terms into first-order equations,

$$\vec{\mathbf{q}}_h = \vec{\mathbf{G}}_h(\mathbf{u}_h, \nabla\mathbf{u}_h) = -\mathbf{K}(\mathbf{u}_h)\nabla\mathbf{u}_h, \quad (2.37)$$

$$\mathbf{r}_{h,V}(\mathbf{u}_h) = -\nabla \cdot \vec{\mathbf{G}}_h(\mathbf{u}_h, \vec{\mathbf{q}}_h) = -\nabla \cdot \vec{\mathbf{q}}_h \quad (2.38)$$

where $\vec{\mathbf{q}}_h \in \mathcal{Q}_h$ is an approximation of the d -dimensional continuous viscous flux inside the element and $\mathbf{r}_{h,V}$ is the strong form of the viscous residual. Testing Eqn. 2.37 with

test functions $\vec{\tau}_h \in \mathcal{Q}_h$ gives

$$\int_{\Omega_e} \vec{\tau}_h^T \cdot \vec{\mathbf{q}}_h d\Omega = + \int_{\Omega_e} \mathbf{u}_h^T \nabla \cdot (\mathbf{K}^T \vec{\tau}_h) d\Omega - \int_{\partial\Omega_e} \mathbf{u}_h^{+T} \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS \quad \forall \vec{\tau}_h \in \mathcal{Q}_h. \quad (2.39)$$

Note that the viscous flux $\vec{\mathbf{q}}_h$ is discontinuous across the element interfaces. If directly used in Eqn. 2.38 to evaluate divergence, this discontinuity will cause stability issues for the viscous discretization. Instead, we use a “weakly continuized” flux introduced by the first approach of Bassi and Rebay [82], $\vec{\sigma}_h = \vec{\mathbf{q}}_h + \vec{\delta}_h$, defined as

$$\int_{\Omega_e} \vec{\tau}_h^T \cdot \vec{\sigma}_h d\Omega = + \int_{\Omega_e} \mathbf{u}_h^T \nabla \cdot (\mathbf{K}^T \vec{\tau}_h) d\Omega - \int_{\partial\Omega_e} (\hat{\mathbf{u}}_h)^T \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS \quad \forall \vec{\tau}_h \in \mathcal{Q}_h, \quad (2.40)$$

where $\hat{\mathbf{u}}_h$ is the numerical flux function for \mathbf{u}_h which brings in weak continuation for $\vec{\sigma}_h$. A natural choice of $\hat{\mathbf{u}}_h$ for diffusion problem is the averaged state defined at the element interface,

$$\hat{\mathbf{u}}_h = \{\mathbf{u}_h\} = \frac{\mathbf{u}_h^+ + \mathbf{u}_h^-}{2}. \quad (2.41)$$

Subtracting Eqn. 2.39 from Eqn. 2.40 defines the auxiliary variable $\vec{\delta}_h$ as

$$\begin{aligned} \int_{\Omega_e} \vec{\tau}_h^T \cdot \vec{\delta}_h &= - \int_{\partial\Omega_e} (\hat{\mathbf{u}}_h - \mathbf{u}_h^+)^T \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS \\ &= - \frac{1}{2} \int_{\partial\Omega_e} (\mathbf{u}_h^- - \mathbf{u}_h^+)^T \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS \\ &= - \frac{1}{2} \int_{\partial\Omega_e} \llbracket \mathbf{u}_h \rrbracket^T \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS \quad \forall \vec{\tau}_h \in \mathcal{Q}_h, \end{aligned} \quad (2.42)$$

where $\llbracket \cdot \rrbracket$ denotes the jumps at the element interfaces. Eqn. 2.42 can also be considered as a *lifting* operation which converts the states jumps (discontinuity) on the element interface into the surrounding elemental fluxes.

Now instead of taking the divergence of the original flux in Eqn. 2.38, we use the “continuized” flux in Eqn. 2.40. The resulting weak form can be written as

$$\mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) = + \int_{\Omega_e} \vec{\sigma}_h^T \cdot \nabla \mathbf{w}_h d\Omega_e - \int_{\partial\Omega_e} \mathbf{w}_h^T \hat{\vec{\sigma}}_h \cdot \vec{n} dS \quad \forall \sigma_h \in \mathcal{V}_h, \quad (2.43)$$

where $\hat{\vec{\sigma}}_h$ is the numerical flux function of $\vec{\sigma}_h$, which again takes the average on the element interfaces,

$$\hat{\vec{\sigma}}_h = \{\vec{\mathbf{q}}_h\} + \{\vec{\delta}_h\}. \quad (2.44)$$

However, this choice of numerical flux for $\vec{\sigma}_h$ together with the jump lifting operation

defined in Eqn. 2.42 results in a non-compact scheme. Moreover, this approach exhibits suboptimal convergence rate for odd approximation orders [83]. Hence, the second form of Bassi and Rebay (BR2) [83] is used in this work. In BR2 treatment, $\vec{\delta}_h$ is defined independently for each face $f \in \partial\Omega_e$,

$$\int_{\Omega_e} \vec{\delta}_{h,f} = -\frac{1}{2} \int_f [[\mathbf{u}_h]]^T \mathbf{K}^T \vec{\tau}_h \cdot \vec{n} dS, \quad \vec{\delta}_h|_{\Omega_e} = \sum_{f \in \partial\Omega_e} \vec{\delta}_{h,f}, \quad (2.45)$$

and accordingly the numerical flux for $\widehat{\sigma}_h$ is modified as

$$\widehat{\sigma}_{h,f} = \{\vec{\mathbf{q}}_h\} + \{\vec{\delta}_{h,f}\}. \quad (2.46)$$

If we choose the test function in Eqn. 2.40 such that $\vec{\tau}_h = \nabla \mathbf{w}_h$, we can substitute Eqn. 2.40 into Eqn. 2.43 to eliminate the first volume integral,

$$\begin{aligned} \mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) d\Omega = & + \int_{\Omega_e} \mathbf{u}_h^T \nabla \cdot (\mathbf{K}^T \nabla \mathbf{w}_h) d\Omega - \int_{\partial\Omega_e} \widehat{\mathbf{u}}_h^T \mathbf{K}^T \nabla \mathbf{w}_h \cdot \vec{n} dS \\ & - \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\sigma}_h \cdot \vec{n} dS. \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \end{aligned} \quad (2.47)$$

Integrating by parts for the first volume integral and plugging in the numerical fluxes defined in Eqn. 2.41 and Eqn. 2.46, we obtain the weak form of the viscous terms as

$$\begin{aligned} \mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) d\Omega = & - \int_{\Omega_e} \nabla \mathbf{w}_h^T \mathbf{K} \nabla \mathbf{u}_h d\Omega + \int_{\partial\Omega_e} (\mathbf{u}_h - \widehat{\mathbf{u}}_h)^T \mathbf{K}^T \nabla \mathbf{w}_h \cdot \vec{n} dS \\ & - \int_{\partial\Omega_e} \mathbf{w}_h^T \{\mathbf{K} \nabla \mathbf{u}\} \cdot \vec{n} dS - \sum_{f \in \partial\Omega_e} \int_f \mathbf{w}_h^T \{\vec{\delta}_{h,f}\} \cdot \vec{n} dS \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \end{aligned} \quad (2.48)$$

This discretization is *compact* since the numerical flux $\vec{\sigma}_h$ only depends on the two elements directly sharing the common faces. Moreover, if the auxiliary variable $\vec{\delta}_{h,f}$ is solved as $\vec{\delta}_{h,f} \cdot \vec{n}$ inside each element with local solution bases, Eqn. 2.48 provides a symmetric semilinear form and ensures *adjoint consistency* or so called *dual consistency*. See Chapter 3 for more details.

Often to ensure stability, the flux function $\vec{\sigma}$ is modified as $\widehat{\sigma}_h = \{\vec{\mathbf{q}}_h\} + \eta_f \{\vec{\delta}_{h,f}\}$. The method is provably stable when the non-dimensional stabilization factor η_f is chosen to be greater than or equal to the maximum number of faces on the adjacent elements [123]. We set it to be twice of the maximum number of faces in the current work. For solutions that exhibit discontinuities, *e.g.*, shocks in transonic regimes, additional stabilization are

added if necessary by increasing η_f .

2.4.2.3 Source Term Discretization

The source term is discretized by multiplying with test functions and integrating over the computational domain,

$$\mathcal{R}_{h,S}(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) = \int_{\Omega_e} \mathbf{w}_h^T \mathbf{S}(\mathbf{u}_h, \nabla \mathbf{u}_h). \quad (2.49)$$

The source discretization shown in Eqn. 2.49 is not dual-consistent, but additional terms can be added to the weak form to ensure dual consistency. Alternatively, asymptotic consistency can be achieved by replacing $\nabla \mathbf{u}$ with the ‘‘continuized’’ gradients $\vec{\sigma}_h$ in the source term [124, 125]. In our DG implementation, the output functional is modified to include the numerical flux $\widehat{\sigma}$ to achieve an asymptotically dual-consistent discretization.

2.4.3 Discrete Form and Solution Technique

By choosing a complete set of order p basis functions, ϕ_i , such that $\mathcal{V}_h^p|_{\Omega_e} = \text{span}\{\phi_i\}$, we can represent the solution as

$$\mathbf{u}_h = \sum_{e=1}^{N_e} \sum_{j=1}^{N_p} \mathbf{U}_{e,j} \phi_{e,j}(\vec{x}), \quad (2.50)$$

where $\mathbf{U}_{e,j}$ is the solution coefficients associated with the j^{th} out of N_p basis functions in element e . The basis functions are defined in the reference space as order p polynomials, $\phi(\vec{\xi})$ which takes the same form for each element while the geometry mapping inside different elements alter the approximation space, $\phi(\vec{x}(\vec{\xi}))$. In this work, we consider two sets of basis functions spanning the space covered by monomials on the reference space, $\prod_{i=1}^d \xi_i^{p_i}$: a full-order space in which $\sum_{i=1}^d p_i \leq p$, and a tensor-product space in which $\forall i \leq d, p_i \leq p$. The resulting dimensions of these spaces are

$$\text{full-order basis: } N_p = \binom{p+d}{d} = \frac{(p+d)!}{p!d!}, \quad \text{tensor-product basis: } N_p = (p+1)^d.$$

In each element, every state component is approximated using the same set of basis functions. In other words, the number of unknowns associated with each state component in one element is N_p , which is also called *degrees of freedom (DOF)* per element per state. The sum of degrees of freedom over elements is often used as a measure of the system cost. $\mathbf{U}_{e,j} \in \mathbb{R}^s$ includes the unknowns of all the state components associated with j^{th}

basis, and elemental solution vector $\mathbf{U}_e = [\mathbf{U}_{e,j}] \in \mathbb{R}^{N_p \times s}$ is an unrolled vector consists of the total DOF of element e . Therefore the weak form in Eqn. 2.34 can be written as a system of ordinary differential equations (ODEs),

$$\mathbf{M} \frac{d\mathbf{U}_h}{dt} + \mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}. \quad (2.51)$$

where $\mathbf{U}_h = [\mathbf{U}_e] \in \mathbb{R}^N$ is the unrolled vector consists of system total DOF, $N = N_e \times N_p \times s$. $\mathbf{M} \in \mathbb{R}^{N \times N}$ is a block-diagonal *mass matrix*,

$$\mathbf{M} = \text{diag}(\mathbf{M}_e), \quad \mathbf{M}_{e,ij} = \int_{\Omega_e} \phi_i \phi_j d\Omega. \quad (2.52)$$

Similarly, the global residual vector $\mathbf{R}_h \in \mathbb{R}^N$ is an unrolled vector of element-wise residuals $\mathbf{R}_e \in \mathbb{R}^{N_p \times s}$, which are defined as

$$\begin{aligned} \mathbf{R}_e &= [\mathbf{R}_{e,i}], \quad i = 1, 2, \dots, N_p; \\ \mathbf{R}_{e,i} &= [\mathbf{R}_{e,i,r}], \quad r = 1, 2, \dots, s; \\ \mathbf{R}_{e,i,r} &= \mathcal{R}_h(\mathbf{u}_h, \phi_i \mathbf{e}_r |_{\Omega_e}), \end{aligned} \quad (2.53)$$

where $\mathbf{e}_r \in \mathbb{R}^s$ is a vector of all zeros except a 1 in position r .

For steady problems considered in this work, $d\mathbf{U}_h/dt = 0$, while it is kept when *pseudo-transient continuation* is used to evolve the solution to a steady state. Particularly, an *implicit* backward Euler time integration scheme is used to evolve the system in time, such that at time step n the solution update follows

$$\mathbf{R}(\mathbf{U}^{n+1}) = \frac{\mathbf{M}}{\Delta t} (\mathbf{U}_h^{n+1} - \mathbf{U}_h^n) + \mathbf{R}_h(\mathbf{U}_h^{n+1}) = \mathbf{0}, \quad (2.54)$$

where the time step Δt is chosen based on Courant-Friedrichs-Lewy (CFL) number,

$$\Delta t_e = \text{CFL} \frac{h_e}{c_e}, \quad \Delta t = \min_{\Omega_e \in \mathcal{T}_h} \Delta t_e, \quad (2.55)$$

where h_e is a measure of the element grid spacing and c_e is the maximum convective wave speed over the element. The system in Eqn. 2.54 is linearized as

$$\left(\frac{\mathbf{M}}{\Delta t} + \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}} \Big|_{\mathbf{U}^k} \right) \Delta \mathbf{U}^k + \mathbf{R}(\mathbf{U}^k) = \mathbf{0}. \quad (2.56)$$

The linear system above is solved using a element-line preconditioned generalized minimal residual method (GMRES) Krylov subspace method [126, 121], in which k denotes the

k^{th} iteration in the nonlinear solve. The solution update obtained in Eqn. 2.56 is then limited based on physical realizability constraints of the equation set [127]. The CFL number is increased exponentially if a successful update is taken, otherwise the solution is rolled back and the CFL is decreased.

Shock Capturing High-order finite-element solutions are susceptible to oscillations when the underlying solution has discontinuities, *e.g.*, shocks in transonic regimes. The solver uses *artificial viscosity* to stabilize the solution. Currently, an element-based artificial viscosity is used, and more details can be found in [128].

CHAPTER 3

Adjoint-Based Sensitivity Analysis and Output Error Estimation

Since the early works of the adjoint method in aerodynamic optimization, it has been the predominant approach of computing output gradients due to its excellent scalability in large-scale problems. In addition, adjoint-based error estimation has also shown much success in quantifying solution errors and providing effective indicators for mesh adaptation purposes. In the current work, adjoints will be used in aerodynamic optimization problems for both evaluating the output gradients and quantifying the output accuracy. This chapter gives a formal introduction to adjoint methods, providing a brief review of the use of adjoint methods in optimization and error estimation, both in the context of finite element methods.

3.1 Adjoint and Duality

Although the entire solution field offers insight into the underlying physics of the PDE system we are solving, in most engineering applications such as design optimization, the quantities of most interest are often some scalar outputs, *e.g.*, drag or lift. For simplicity, we will start with a scalar PDE first and then generalize to systems of equations. Let \mathcal{J} be an output quantity derived from the solution u by applying a functional $\mathcal{J}(\cdot)$,

$$\mathcal{J}(u) = \int_{\Omega} g u d\Omega = \langle u, g \rangle, \quad (3.1)$$

where $\langle \cdot, \cdot \rangle$ is the inner product defined in $L^2(\Omega)$. Consider a linear differential PDE,

$$r(u) = Lu - f, \quad u \in \mathcal{V}, \quad (3.2)$$

where L is a linear differential operator, for instance the convection operator with velocity \vec{a} as $\vec{a} \cdot \nabla$, or the diffusion operator with diffusivity coefficient κ as $-\kappa \nabla^2$; f accounts for the source term if present in the system. Using an optimal control approach, we can consider a constrained optimization problem, which is equivalent to evaluating the output \mathcal{J} from the solution of $r(u) = 0$. The trivial optimization, often called the *primal* problem can be stated as

$$\begin{aligned} \mathcal{J}_p^* &= \min_{u \in \mathcal{V}} \mathcal{J}(u) \\ \text{s.t. } & r(u) = 0, \end{aligned} \quad (3.3)$$

where \mathcal{J}_p^* denotes the optimum of the primal problem. The above problem can then be written as searching for the stationary point of the Lagrangian function \mathcal{L} as

$$\begin{aligned} \mathcal{L}(u, \psi) &= \mathcal{J}(u) + \mathcal{R}(u, \psi) = \langle u, g \rangle + \langle r(u), \psi \rangle \\ &= \langle u, g \rangle + \langle Lu - f, \psi \rangle, \\ &= -\langle \psi, f \rangle + \langle L^* \psi + g, u \rangle, \end{aligned} \quad (3.4)$$

where $\psi \in \mathcal{V}$ is the Lagrange multiplier, whose inner product with $r(u)$ gives the weak form \mathcal{R} of the original PDE. L^* is the adjoint operator of L , which can be obtained by applying integration by parts on the weak form \mathcal{R} . The Lagrangian function introduces a corresponding *dual* problem,

$$\begin{aligned} J_d^* &= \max_{\psi \in \mathcal{V}} -\langle \psi, f \rangle \\ \text{s.t. } & d(\psi) = L^* \psi + g = 0. \end{aligned} \quad (3.5)$$

where J_d^* represents the optimum of the dual problem, $d(\psi)$ is the *dual equation* or often referred to as the *adjoint equation*, subject to corresponding boundary conditions. By definition, the optima of the primal and dual problems satisfy

$$J_p^*(u^*) = \inf_{u \in \mathcal{V}} \sup_{\psi \in \mathcal{V}} \mathcal{L}(u, \psi) \geq J_d^*(\psi^*) = \sup_{\psi \in \mathcal{V}} \inf_{u \in \mathcal{V}} \mathcal{L}(u, \psi). \quad (3.6)$$

The inequality is called *weak duality*, which always hold regardless of the form of the functionality and dual/primal equations. The equality, also termed as *strong duality*, holds when the optima J_p^* and J_d^* are finite and the corresponding primal solution u^* and adjoint solution ψ^* are feasible, *i.e.*, they are solutions of the primal equation in Eqn. 3.3 and the adjoint equation in Eqn. 3.5, respectively. An immediate necessary condition for strong duality is the Karush-Kuhn-Tucker (KKT) condition, which enforces the stationarity of

the Lagrangian function with respect to the primal and adjoint variables,

$$\mathcal{L}_\psi \delta\psi = \mathcal{R}'[\psi] \delta\psi = \mathcal{R}(u, \delta\psi) = 0 \quad \forall \delta\psi \in \mathcal{V} \quad (3.7)$$

$$\Rightarrow \mathcal{L}_\psi = r(u) = Lu - f = 0 \quad \text{primal equation;} \quad (3.8)$$

$$\mathcal{L}_u \delta u = \mathcal{J}'[u] \delta u + \mathcal{R}'[u] \delta u = 0 \quad \forall \delta u \in \mathcal{V} \quad (3.9)$$

$$\Rightarrow \mathcal{L}_u = d(\psi) = L^* \psi + g = 0 \quad \text{adjoint (dual) equation.} \quad (3.10)$$

Eqn. 3.7 and Eqn. 3.9 recover the weak forms of the primal and adjoint equations, respectively, while Eqn. 3.8 and Eqn. 3.10 present their strong forms. Strong duality relies on the convexity of the output functional and the existence of the primal and adjoint solutions, which can be ensured by the output definition and the well-posedness of the primal and adjoint problems.

For nonlinear systems we have $r(u) = N(u) - f$, where N is the nonlinear operator applied to u . Suppose \tilde{u} is the primal solution, *i.e.*, $r(\tilde{u}) = 0$, we can then linearize the system at \tilde{u} as

$$r(\tilde{u} + \delta u) = r(\tilde{u}) + N'[u] \delta u \quad \Rightarrow \quad L_{\tilde{u}} = N'[u], \quad (3.11)$$

where $L_{\tilde{u}}$ is the linear operator associated with the linearized system. The corresponding linearized output functional can be defined as

$$\mathcal{J}(\tilde{u} + \delta u) = \mathcal{J}(\tilde{u}) + \mathcal{J}'[u] \delta u \quad \Rightarrow \quad g_{\tilde{u}} = \mathcal{J}'[u], \quad (3.12)$$

where $g_{\tilde{u}}$ is the linearized output weight function. Then the adjoint equation can be written as

$$L_{\tilde{u}}^* \psi + g_{\tilde{u}} = 0. \quad (3.13)$$

Or alternatively, we can directly invoke the KKT condition for the nonlinear system to obtain both the primal and the adjoint equations,

$$\mathcal{L}_\psi \delta\psi = \mathcal{R}'[\psi](u, \delta\psi) = \mathcal{R}(u, \delta\psi) = 0 \quad \forall \delta\psi \in \mathcal{V} \quad \text{weak form primal equation} \quad (3.14)$$

$$\mathcal{L}_u \delta u = \mathcal{J}'[u] \delta u + \mathcal{R}'[u](\delta u, \psi) = 0 \quad \forall \delta u \in \mathcal{V} \quad \text{weak form adjoint equation} \quad (3.15)$$

A similar idea can be easily extended to systems of equations, where the strong form

residual $\mathbf{r}(\mathbf{u})$ and weak form residual \mathcal{R} are defined as

$$\begin{aligned}\mathbf{r}(\mathbf{u}) &= \mathcal{N}(\mathbf{u}) - \mathbf{f} = 0, \\ \mathcal{R}(\mathbf{u}, \mathbf{w}) &= \int_{\Omega} \mathbf{w}^T \mathbf{r}(\mathbf{u}) d\Omega = \int_{\Omega} w_k r_k(\mathbf{u}) d\Omega = \sum_{k=1}^s \langle w_k, r_k(\mathbf{u}) \rangle = 0 \quad \forall \mathbf{w} \in \mathcal{V} = [\mathcal{V}]^s.\end{aligned}\tag{3.16}$$

Again, by applying the stationarity condition and replacing the δ terms with a general perturbation \mathbf{w} in \mathcal{V} , we recover the primal equation and obtain the adjoint equation,

$$\mathcal{L}_{\psi} \mathbf{w} = \mathcal{R}(\mathbf{u}, \mathbf{w}) = 0 \quad \forall \mathbf{w} \in \mathcal{V}, \tag{3.17}$$

$$\mathcal{L}_{\mathbf{u}} \mathbf{w} = \mathcal{J}'[\mathbf{u}](\mathbf{w}) + \mathcal{R}'[\mathbf{u}](\mathbf{w}, \psi) = 0 \quad \forall \mathbf{w} \in \mathcal{V}. \tag{3.18}$$

3.2 Discrete Adjoint

Suppose the approximation space for both the primal and dual solutions is finite dimensional, defined by a computational mesh Ω and an approximation order p , $\mathcal{V}_h^p = [\mathcal{V}_h^p]^s$, or simply denoted as \mathcal{V}_h . The primal and adjoint equations can be written as

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h) = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h, \tag{3.19}$$

$$\mathcal{J}'_h[\mathbf{u}_h](\mathbf{w}_h) + \mathcal{R}'_h[\mathbf{u}_h](\mathbf{w}_h, \psi_h) = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \tag{3.20}$$

Eqn. 3.20 assumes the primal \mathbf{u}_h and the adjoint ψ_h lie in the same space \mathcal{V}_h . However, with Eqn. 3.18 a strong form of the adjoint equation can also be derived, which suggests that the adjoint problem can be solved in a different space, *i.e.*, discretized in different mesh or approximation order. This approach is called the *continuous adjoint* approach, in which the word “continuous” means obtaining the continuous adjoint equations before discretizing. Despite the flexibility of the continuous approach, its derivation requires manipulation on the primal governing equation and careful treatment of the boundary conditions of the adjoint equations.

On the other hand, the discrete adjoint can also be derived in a fully-discrete fashion, termed as *discrete adjoint* approach. Consider the fully-discrete form of the steady-state primal problem in Eqn. 2.51,

$$\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}. \tag{3.21}$$

The discretized output of interest can be written as $J_h(\mathbf{U}_h)$, where J_h is the discretized form of \mathcal{J} . Similarly, the evaluation of the output $J_h(\mathbf{U}_h)$ can be formulated as a mini-

mization problem with respect to the state solution vector \mathbf{U}_h , and the duality still holds under discretization [96]. Invoking KKT condition again recovers the discrete primal equation and defines the discrete adjoint equation.

Here, we derive the fully-discrete adjoint equation in another way, which involves more physical intuitions. In the continuous weak form of adjoint equation in Eqn. 3.18, the adjoint variable can be interpreted as the output sensitivity to an infinitesimal residual perturbation. Similarly, in a discrete form, we can define the adjoint vector $\boldsymbol{\Psi}_h$ to relate the output perturbation and the residual perturbation, $\delta\mathbf{R}_h \in \mathbb{R}^{N_h}$,

$$\delta J_h = J_h(\mathbf{U}_h + \delta\mathbf{U}_h) - J_h(\mathbf{U}_h) = \boldsymbol{\Psi}_h^T \delta\mathbf{R}_h. \quad (3.22)$$

The residual perturbation can come from the change of the boundary conditions in optimization problems or from different levels of discretizations in error estimation. To evaluate the output on the perturbed system, the state (perturbation) needs to be computed, which satisfies

$$\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h}\delta\mathbf{U}_h + \delta\mathbf{R}_h = \mathbf{0} \quad \Rightarrow \quad \delta\mathbf{U}_h = - \left[\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h} \right]^{-1} \delta\mathbf{R}_h. \quad (3.23)$$

The output perturbation can then be obtained using the linearization of the output starting with the state perturbation from Eqn. 3.23,

$$\delta J_h = \frac{\partial J_h}{\partial\mathbf{U}_h}\delta\mathbf{U}_h = - \frac{\partial J_h}{\partial\mathbf{U}_h} \left[\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h} \right]^{-1} \delta\mathbf{R}_h. \quad (3.24)$$

Comparing Eqn. 3.22 and Eqn. 3.24, we require the following identity for any permissible residual perturbation,

$$\boldsymbol{\Psi}_h^T = - \frac{\partial J_h}{\partial\mathbf{U}_h} \left[\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h} \right]^{-1}. \quad (3.25)$$

Rearranging the above equation, we arrive at the discrete adjoint equation,

$$\left[\frac{\partial\mathbf{R}_h}{\partial\mathbf{U}_h} \right]^T \boldsymbol{\Psi}_h + \left[\frac{\partial J_h}{\partial\mathbf{U}_h} \right]^T = \mathbf{0}. \quad (3.26)$$

A closer look at the adjoint solution vector reveals

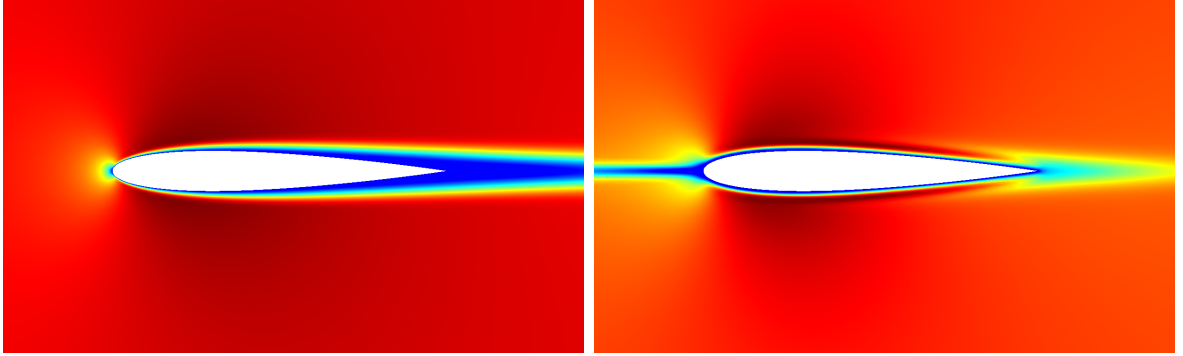
$$\begin{aligned}
\delta J_h &= \underbrace{\begin{bmatrix} g_{h,1} & g_{h,2} & \cdots & g_{h,N_h} \end{bmatrix}}_{\text{output weights } \mathbf{G}_h^T} \underbrace{- \begin{bmatrix} \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \end{bmatrix}^{-1}}_{\text{states perturbation } \delta \mathbf{U}_h} \underbrace{\begin{bmatrix} \delta R_{h,1} \\ \delta R_{h,2} \\ \vdots \\ \delta R_{h,N_h} \end{bmatrix}}_{\delta \mathbf{R}_h} \\
&= \mathbf{G}_h^T \boldsymbol{\alpha}_{h,1} \delta R_{h,1} + \mathbf{G}_h^T \boldsymbol{\alpha}_{h,2} \delta R_{h,2} + \cdots + \mathbf{G}_h^T \boldsymbol{\alpha}_{h,N_h} \delta R_{h,N_h} \\
&= \Psi_{h,1} \delta R_{h,1} + \Psi_{h,2} \delta R_{h,2} + \cdots + \Psi_{h,N_h} \delta R_{h,N_h} \\
&= \boldsymbol{\Psi}_h^T \delta \mathbf{R}_h,
\end{aligned} \tag{3.27}$$

where we can see that each column of the negative inverse Jacobian matrix, $\boldsymbol{\alpha}_{h,i}$, gives the sensitivity of the states to the residual perturbation, which weighted by the output linearization vector \mathbf{G}_h defines the adjoint vector that represents the output sensitivity with respect to the residual perturbation.

For discretizations taking a variational form (*i.e.*, a weak form), the continuous weak form of the adjoint equation, Eqn. 3.20 reduces to the equivalent discrete form as Eqn. 3.26 after choosing a set of basis function. Likewise, using the discrete adjoint vector solution and the corresponding basis set, a continuous adjoint field $\boldsymbol{\psi}_h$ can be constructed. For discretizations like finite volume and finite difference, the continuous adjoint approach and discrete adjoint approach may arrive at different fully-discrete algebraic systems. Often, in practice, the discrete adjoint approach is preferred if an implicit solver is employed in the primal flow problem, since the adjoint equation only requires a transpose of the primal residual Jacobian and can be solved in a similar framework with minimal change of the computational code. However, for explicit time integration methods, the Jacobian matrix may not be computed or stored, and a continuous adjoint approach is generally easier to implement [129, 130], despite the requisite derivation of the strong form adjoint equations.

In this work, the discrete adjoint approach is used. Figure 3.1 shows an example of compressible Navier-Stokes simulation for laminar flow over a symmetric National Advisory Committee for Aeronautics (NACA) 0012 Airfoil. The flow is moving from left to right horizontally, *i.e.*, the airfoil angle of attack is 0° , with a freestream Reynolds number of 5000 and Mach number of 0.5. The output of interest J_h is the drag of the airfoil, based on which the adjoint vector is solved using Eqn. 3.26. The left contour plot shows the

x -momentum component of the primal state, while the right one depicts the conservation of x -momentum component for the drag adjoint. Both the continuous primal and adjoint fields are constructed using the discrete primal and adjoint vectors respectively, with the same basis set. The adjoint field represents the sensitivity of the drag with respect to the residual, in Figure 3.1b particularly the x -momentum conservation residual.



(a) primal field x -momentum component (b) adjoint field x -momentum component

Figure 3.1: Sample primal and adjoint solutions of a laminar flow simulation over the NACA 0012 airfoil. The color scales are clipped to show interesting features. In the adjoint plot, red and blue regions identify the areas which the drag output is most sensitive to.

As shown in Figure 3.1b, the drag output is sensitive to the residual perturbation in the blue and red regions, namely the airfoil boundary and the stagnation streamline. Meanwhile, the drag is less sensitive to downstream regions. The adjoint fields share similarity compared to the primal fields, such as the presence of a boundary layer close to the airfoil boundary. Moreover, the stagnation streamline is clearly depicted in the adjoint field, which resembles the wake in the primal field, however presented in a reversed direction. These interesting similarities, including the adjoint boundary layer and the “reversed wake”, are inherent to the primal-adjoint symmetry rather than just coincidence. Let’s consider a model linear convection-diffusion problem, which is a linear version of the laminar flow problem,

$$Lu - f = \vec{v} \cdot \nabla u + \nu \nabla^2 u - f = 0, \quad (3.28)$$

where \vec{v} is the convection velocity and ν denotes the viscosity. Multiplying by a test function, integrating by parts, and assuming homogeneous boundary condition, we can obtain the weak form and hence the strong form of the adjoint equation,

$$L^* \psi + g = -\vec{v} \cdot \nabla \psi + \nu \nabla^2 \psi + g = 0. \quad (3.29)$$

The similarity in the adjoint and primal equations shown in Eqn. 3.28 and Eqn. 3.29

explains the similarity in the contour plots presented in Figure 3.1. More specifically, the negative sign of the convection velocity \vec{v} is responsible for the reversed convection of the adjoint field, while the same diffusion operator controls the similar boundary layers in both the primal and adjoint solutions. In a fully-discrete fashion, we can define the discrete form of Eqn. 3.28 as

$$\mathbf{R}_h(\mathbf{U}_h) = \mathbf{A}_h \mathbf{U}_h - \mathbf{F}_h = (\mathbf{C}_h + \mathbf{D}_h) \mathbf{U}_h - \mathbf{F}_h = \mathbf{0}, \quad (3.30)$$

where $\mathbf{A}_h \in \mathbb{R}^{N_h \times N_h}$ is the residual Jacobian matrix, while $\mathbf{C}_h \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{D}_h \in \mathbb{R}^{N_h \times N_h}$ are the Jacobian matrix contributions from the convection and diffusion respectively; $\mathbf{F}_h \in \mathbb{R}^{N_h}$ accounts for the source terms and the boundary conditions of the primal equation. Often \mathbf{C}_h is asymmetric accounting for the upwinding of the convection scheme, while \mathbf{D}_h is usually symmetric as the numerical viscous fluxes in general take an average form. The corresponding fully-discrete adjoint equation can be written as,

$$\mathbf{A}_h^T \boldsymbol{\Psi}_h + \mathbf{G}_h = (\mathbf{C}_h^T + \mathbf{D}_h^T) \mathbf{U}_h + \mathbf{G}_h = \mathbf{0}, \quad (3.31)$$

in which we can immediately observe that the transpose of \mathbf{C}_h resembles an upwind discretization for the adjoint variable propagating reversely as the primal variable, while the self symmetric (adjoint) diffusive Jacobian \mathbf{D}_h possesses the same structure in the adjoint equation, producing a similar boundary layer features in the adjoint solution. More formally, we can say that the transpose of the Jacobian matrix, $(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h})^T$ is a discrete operator (discretization) corresponding to the continuous adjoint operator L^* .

The adjoint field depicted in Figure 3.1b is a continuous representation, $\boldsymbol{\psi}_h$, of the discrete adjoint solution, $\boldsymbol{\Psi}_h$, obtained using the discrete adjoint approach, *i.e.*, solving Eqn. 3.26. It will only approach the exact adjoint solution of the continuous adjoint equation Eqn. 3.18, $\boldsymbol{\psi}$, if the discrete adjoint equation is consistent with the exact adjoint problem, or in other words, $(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h})^T$ is a consistent discretization of L^* . The consistency of the adjoint will be discussed with more details in Section 3.5.

3.3 Adjoint-Based Sensitivity Analysis

The strong duality of linear systems also suggests another approach of calculating the output, alleviating the repeated computation of the states for various boundary conditions. Consider an optimization problem using a discrete system of equations, as the design parameters vary the \mathbf{F}_h vector also changes accordingly. In order to evaluate the objective output $J_h = \mathbf{G}_h^T \mathbf{U}_h$ on the updated design, the system needs to be solved repeatedly for

different \mathbf{F}_h . A more efficient way of computing the outputs when exploring the design space is through the dual form, $J_h = -\Psi_h^T \mathbf{F}_h$, in which the variation of the design \mathbf{F}_h is directly converted to the output changes.

For a general nonlinear system, the strong duality still holds as long as the output functional is convex. Due to the nonlinearity of the operator, the output cannot be directly evaluated using the adjoint variables. However, the adjoint variables are still useful for estimating the output sensitivity or perturbations. Consider an optimization problem using a nonlinear system of discrete equations. Suppose the design space is parameterized with a set of design parameters, $\boldsymbol{\mu}$. The forward problem can be defined as

$$\underbrace{\boldsymbol{\mu}}_{\text{design parameters } \in \mathbb{R}^{N_\mu}} \rightarrow \underbrace{\mathbf{R}_h(\mathbf{U}_h, \boldsymbol{\mu})}_{N_h \text{ equations}} \rightarrow \underbrace{\mathbf{U}_h}_{\text{states } \in \mathbb{R}^{N_h}} \rightarrow \underbrace{J_h(\mathbf{U}_h)}_{\text{output } \in \mathbb{R}}. \quad (3.32)$$

In order to exploit gradient-based optimization algorithms, we need to first compute the gradient (sensitivity) of the output with respect to the design variables,

$$\frac{dJ_h}{d\boldsymbol{\mu}} \in \mathbb{R}^{N_\mu}. \quad (3.33)$$

A natural choice for computing the sensitivity is using the finite difference, which perturbs the parameters one at a time and solves the nonlinear system repeatedly to obtain the output perturbations, *i.e.*, invoke primal form of the output repeatedly. This process can be viewed as a feedback control system as shown in Figure 3.2, the red part. Given a parameter perturbation, the corresponding residual perturbation $\delta\mathbf{R}_{h,\mu}$ feeds a positive signal to the control system. In order to balance the system, the controller perturbs the states, $\delta\mathbf{U}_h$, to produce a residual perturbation $\delta\mathbf{R}_{h,\mathbf{U}}$ as a negative feedback signal until the nonlinear residual is balanced (zero) again. The response of the system is the perturbed output, which then gives the output perturbation. However, if the nonlinear system is expensive to solve, *i.e.*, N_h is large, the computational cost of the sensitivity calculation increases dramatically as the design space dimension N_μ increases. A more efficient approach is to use the adjoint, which bypasses the expensive nonlinear system solve (feedback control) to directly produce the output perturbations. By substituting the residual perturbation due to the variation of the design parameter, $\delta\mathbf{R}_{h,\mu}$, into Eqn. 3.23 and Eqn. 3.24, or by directly using the definition of adjoint, Eqn. 3.22, we can easily estimate the output perturbation,

$$\delta J_h = \Psi_h^T \delta\mathbf{R}_{h,\mu}. \quad (3.34)$$

This approach scales linearly with the dimension of the design space, since only one residual evaluation $\delta\mathbf{R}_{h,\mu}$ is required for each design parameter, which is much cheaper than solving the nonlinear system. The output sensitivity vector can then be computed as

$$\frac{dJ_h}{d\mathbf{U}_h} = \Psi_h^T \frac{\partial \mathbf{R}_h}{\partial \boldsymbol{\mu}}. \quad (3.35)$$

Sometimes the output definition directly involves the design parameter, then the sensitivity calculation should also include the partial derivative terms with respect to the parameter,

$$\frac{dJ_h}{d\boldsymbol{\mu}} = \frac{\partial J_h}{\partial \boldsymbol{\mu}} + \Psi_h^T \frac{\partial \mathbf{R}_h}{\partial \boldsymbol{\mu}}. \quad (3.36)$$

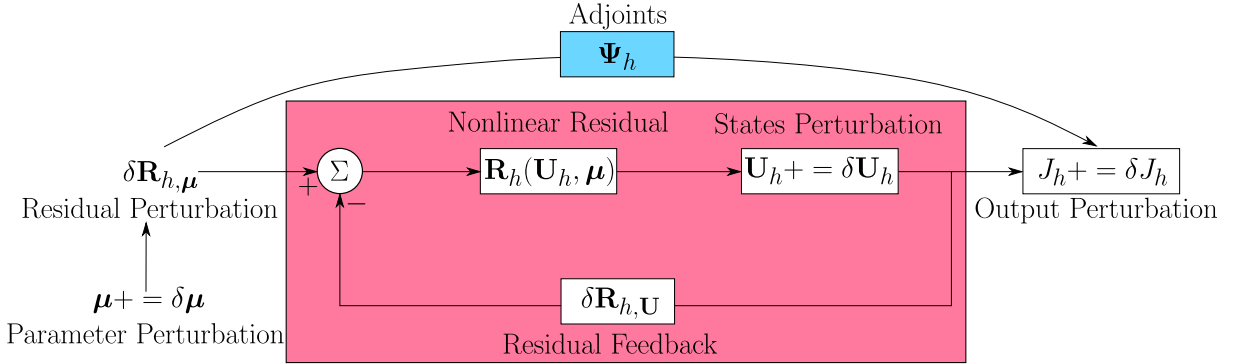


Figure 3.2: Nonlinear system solve as a feedback control system.

3.4 Adjoint-Based Output Error Estimation

Another application of adjoints is estimating the output error and providing effective indicators for mesh adaptation. Again, let's start with the duality of a linear system. Suppose \mathbf{u} and $\boldsymbol{\psi}$ are the exact primal and adjoint solutions respectively, while \mathbf{u}_h and $\boldsymbol{\psi}_h$ are their discrete counterparts. Then if we define the output error as the difference of the outputs evaluated with the discrete numerical solution and the exact solution, we have

$$\begin{aligned} \delta\mathcal{J} &= \mathcal{J}(\mathbf{u}_h) - \mathcal{J}(\mathbf{u}) = \langle \mathbf{u}_h, \mathbf{g} \rangle - \langle \mathbf{u}, \mathbf{g} \rangle \\ &= \langle \mathbf{u}_h - \mathbf{u}, \mathbf{g} \rangle = \langle \mathbf{u}_h - \mathbf{u}, -L^* \boldsymbol{\psi} \rangle \\ &= -\langle L(\mathbf{u}_h - \mathbf{u}), \boldsymbol{\psi} \rangle = -\langle L\mathbf{u}_h - \mathbf{f}, \boldsymbol{\psi} \rangle \\ &= -\langle r(\mathbf{u}_h), \boldsymbol{\psi} \rangle. \end{aligned} \quad (3.37)$$

Eqn. 3.37 provides a way to estimate the numerical error in the output induced by a finite dimensional discretization. The error is in a form of the residual error weighted by the adjoint solution, and thus it is often called the adjoint weighted residual (AWR) or dual weighted residual (DWR) approach. Nonetheless, the error bound is not directly computable as the exact adjoint solution $\boldsymbol{\psi}$ is in general unavailable and has to be approximated. A first try would be using the discrete adjoint solution on the same discretization, $\boldsymbol{\psi}_h$, as an approximation of the exact adjoint, which gives

$$\begin{aligned}
\delta\mathcal{J} &= -\langle r(\mathbf{u}_h), \boldsymbol{\psi}_h \rangle + \langle r(\mathbf{u}_h), \boldsymbol{\psi}_h - \boldsymbol{\psi} \rangle \\
&= - \underbrace{\mathcal{R}_h(\mathbf{u}_h, \boldsymbol{\psi}_h)}_{\text{discrete weak form}} + \underbrace{\langle r(\mathbf{u}_h), \boldsymbol{\psi}_h - \boldsymbol{\psi} \rangle}_{\text{remaining error bound}} \\
&= \langle r(\mathbf{u}_h), \boldsymbol{\psi}_h - \boldsymbol{\psi} \rangle.
\end{aligned} \tag{3.38}$$

The use of a discrete adjoint solution recovers the discrete weak form of the primal problem which is always zero if the discrete primal problem is appropriately solved (sometimes also referred to as *Galerkin orthogonality* in the context of error estimation). The remaining error is then the residual error weighted by the adjoint error. This form is quite interesting as it indicates that the output accuracy depends on not only the accuracy of the primal problem, but also on the accuracy of the adjoint solution, which may never be actually computed during the simulation. The remaining error offers a simple bound for the output error

$$\|\delta\mathcal{J}\| \leq \|r(\mathbf{u}_h)\| \|\boldsymbol{\psi}_h - \boldsymbol{\psi}\|, \tag{3.39}$$

where $\|\cdot\|$ can be any well-defined norm on $L^2(\Omega)$. Typically, the adjoint error $\boldsymbol{\psi}_h - \boldsymbol{\psi}$ is at the order of \mathcal{O}^{p+1} , while the residual error $r(\mathbf{u}_h)$ is $\mathcal{O}(h^{p+1-m})$, where m is the highest order of derivative involved in the operator L . Thus, the overall error in any output functional is at the order of $\mathcal{O}(h^{2p+2-m})$. This *superconvergent* property is one of the key benefits of using Galerkin type of finite element methods. Although the error bound breaks when the primal or adjoint solution exhibits singularities, the superconvergence rate can still be recovered by proper adaptations. For methods like finite difference and finite volume, the first term in Eqn. 3.38 does not vanish and the output error only converges at an order of $p + 1$ unless further postprocessing with the adjoint is employed, *i.e.*, subtracting the first adjoint correction term.

Galerkin orthogonality prevents the effective use of the current space adjoint solution to approximate the exact adjoint solution. A more expensive, but still affordable approach is using the adjoint solution from a finer discretization. Let's define a numerically computable error estimate for the output functional as the difference between the outputs

computed with a coarse space (discretization) solution \mathbf{u}_H and a fine space one \mathbf{u}_h ,

$$\begin{aligned}
\delta\mathcal{J} &= \mathcal{J}_H(\mathbf{u}_H) - \mathcal{J}_h(\mathbf{u}_h) = \mathcal{J}_h(\mathbf{u}_H) - \mathcal{J}_h(\mathbf{u}_h) \\
&= \mathcal{J}'_h[\mathbf{u}_h]\delta\mathbf{u}_h + \mathcal{O}(\delta\mathbf{u}_h^2) \quad \delta\mathbf{u}_h = \mathbf{u}_H - \mathbf{u}_h \\
&= -\mathcal{R}'_h[\mathbf{u}_h](\delta\mathbf{u}_h, \boldsymbol{\psi}_h) + \mathcal{O}(\delta\mathbf{u}_h^2) \\
&= -\underbrace{\mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_h)}_{\text{error estimate}} + \underbrace{\mathcal{R}_h(\mathbf{u}_h, \boldsymbol{\psi}_h)}_{\text{weak form} = 0} + \mathcal{O}(\delta\mathbf{u}_h^2) \\
\Rightarrow \delta\mathcal{J} &\approx -\mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_h).
\end{aligned} \tag{3.40}$$

Similar to Eqn. 3.38, we can apply Galerkin orthogonality,

$$\begin{aligned}
\delta\mathcal{J} &= -\mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_H) - \mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_h - \boldsymbol{\psi}_H) \\
&= -\mathcal{R}_H(\mathbf{u}_H, \boldsymbol{\psi}_H) - \mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_h - \boldsymbol{\psi}_H) \\
&= -\mathcal{R}_h(\mathbf{u}_H, \delta\boldsymbol{\psi}_h), \quad \delta\boldsymbol{\psi}_h = \boldsymbol{\psi}_h - \boldsymbol{\psi}_H.
\end{aligned} \tag{3.41}$$

This form of error estimate resembles the form of error in Eqn. 3.38, which again implies that the output error is high in regions where both the residual (primal error) and the adjoint error are large.

For fully-discrete systems, the state solution vector on the coarse space and fine space are of different dimensions, *i.e.*, $\dim(\mathbf{U}_H) = N_H < \dim(\mathbf{U}_h) = N_h$. In order to utilize the adjoint weighted residual method for estimating the output error, an injected state from the coarse space to the fine space needs to be defined. We assume that the fine approximation space contains the coarse approximation space, $\mathcal{V}_H \subset \mathcal{V}_h$, such that a lossless state injection exists,

$$\mathbf{U}_h^H = \mathbf{I}_h^H \mathbf{U}_H, \tag{3.42}$$

where \mathbf{U}_h^H is the injected state on the fine space and \mathbf{I}_h^H is the coarse-to-fine state injection (prolongation) operator, which can be obtained using an L_2 least-squares projection.

With the injected states, we can estimate the output error as

$$\begin{aligned}
\delta J &= J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) = J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \\
&= \frac{\partial J_h}{\partial \mathbf{U}_h} \delta \mathbf{U}_h + \mathcal{O}(\delta \mathbf{U}_h^2) \quad \delta \mathbf{U}_h = \mathbf{U}_h^H - \mathbf{U}_h \\
&= -\boldsymbol{\Psi}_h^T \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \delta \mathbf{U}_h + \mathcal{O}(\delta \mathbf{U}_h^2) \\
&= -\boldsymbol{\Psi}_h^T \delta \mathbf{R}_h + \mathcal{O}(\delta \mathbf{U}_h^2) \\
&= -\boldsymbol{\Psi}_h^T [\mathbf{R}_h(\mathbf{U}_h^H) - \mathbf{R}_h(\mathbf{U}_h)] + \mathcal{O}(\delta \mathbf{U}_h^2) \\
&= -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) + \mathcal{O}(\delta \mathbf{U}_h^2) \\
\Rightarrow \delta J &\approx -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H).
\end{aligned} \tag{3.43}$$

Again, invoking Galerkin orthogonality and assuming no loss of the projection from the coarse space to the fine space, we can rewrite Eqn. 3.43 as

$$\begin{aligned}
\delta J &= -(\boldsymbol{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H) - (\boldsymbol{\Psi}_h - \boldsymbol{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H) \\
&= -\boldsymbol{\Psi}_H^T \mathbf{R}_H(\mathbf{U}_H) - (\delta \boldsymbol{\Psi}_h)^T \mathbf{R}_h(\mathbf{U}_h^H) \\
&= -(\delta \boldsymbol{\Psi}_h)^T \mathbf{R}_h(\mathbf{U}_h^H),
\end{aligned} \tag{3.44}$$

where $\boldsymbol{\Psi}_h^H = \mathbf{I}_h^H \boldsymbol{\Psi}_H$ is the projected adjoint from the coarse space to the finer one.

Both the continuous form of the error estimate, Eqn. 3.40 and Eqn. 3.41, and the fully-discrete form of error estimate, Eqn. 3.43 and Eqn. 3.44, require the fine space adjoint solution, whose solution relies on the residual and output linearization at the fine space primal solution \mathbf{u}_h^* or \mathbf{U}_h^* ,

$$\begin{aligned}
\mathcal{J}'[\mathbf{u}_h]|_{\mathbf{u}_h^*}(\mathbf{w}_h) + \mathcal{R}'[\mathbf{u}_h]|_{\mathbf{u}_h^*}(\mathbf{w}_h, \boldsymbol{\psi}_h) &= 0 \\
\left(\frac{\partial \mathbf{J}_h}{\partial \mathbf{U}_h} \Big|_{\mathbf{U}_h^*} \right)^T + \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \Big|_{\mathbf{U}_h^*} \right)^T \boldsymbol{\Psi}_h &= \mathbf{0}.
\end{aligned} \tag{3.45}$$

If we solve the fine space primal problem to obtain \mathbf{u}_h^* or \mathbf{U}_h^* , the error estimate itself is of little use as the output difference can be computed exactly using the coarse and fine space states directly, however, effective error indicators can still be obtained for adaptation purposes. In practice, the fine space primal state is not solved exactly. Instead, an approximation of the fine space primal solution, $\tilde{\mathbf{u}}_h$ or $\tilde{\mathbf{U}}_h$, is used to obtain the output

and residual linearization. Then the adjoint equations are solved as

$$\begin{aligned} \mathcal{J}'[\mathbf{u}_h]|_{\tilde{\mathbf{u}}_h}(\mathbf{w}_h) + \mathcal{R}'[\mathbf{u}_h]|_{\tilde{\mathbf{u}}_h}(\mathbf{w}_h, \boldsymbol{\psi}_h) &= 0 \\ \left(\frac{\partial \mathbf{J}_h}{\partial \mathbf{U}_h} \Big|_{\tilde{\mathbf{U}}_h} \right)^T + \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \Big|_{\tilde{\mathbf{U}}_h} \right)^T \boldsymbol{\Psi}_h &= \mathbf{0}. \end{aligned} \quad (3.46)$$

In this work, the injected coarse space solution \mathbf{U}_h^H , is used after several smooth iteration on the fine space primal equation to approximate the fine space primal solution. Then the adjoint equation in Eqn. 3.46 is solved as a linear system using the GMRES linear solver mentioned in Section 2.4.3.

3.4.1 Error Localization

A key feature of the DWR method is the ability of localizing the output error to mesh elements by keeping track of the error contribution from each mesh element,

$$\delta J = -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) = -\sum_{e=1}^{N_e} \boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H), \quad (3.47)$$

where $\boldsymbol{\Psi}_{h,e}^T$ and $\mathbf{R}_{h,e}(\mathbf{U}_h^H)$ are the elemental adjoint and residual in element e , which are easily uncoupled in the DG method. For other methods which involve continuous constraints on the elemental interfaces, the error associated with the shared DOF between elements needs to be carefully distributed to adjacent elements. Therefore, DG methods are especially suitable for error localization and hence mesh adaptation.

The absolute value of the elemental error contribution often serves as a good indicator for adaptation purposes, although the possible error cancellation between elements is ignored. Formally, the elemental adaptive error indicator \mathcal{E}_e is defined as

$$\mathcal{E}_e = |\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)|, \quad \mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e, \quad (3.48)$$

where \mathcal{E} is the sum of the error indicator, which is sometimes used as a more conservative estimate of the output error.

In practice, the output error estimation and error localization procedure can be summarized as follows:

1. Solve the nonlinear discrete system in the current coarse space $\mathbf{R}_H(\mathbf{U}_H) = \mathbf{0}$ to obtain the coarse space solution \mathbf{U}_H .

2. Inject the coarse space solution to the fine space, $\mathbf{U}_h^H = \mathbf{I}_h^H \mathbf{U}_H$.
3. Obtain the fine space adjoint solution either by solving the fine space primal and adjoint problem exactly, or through an approximation as discussed in Eqn. 3.46.
4. Evaluate the fine space residual with the injected states, $\mathbf{R}_h(\mathbf{U}_h^H)$.
5. Weight the fine space residual with the fine space adjoint globally to produce the error estimate, $\delta J = -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)$, and locally to obtain the adaptive error indicator, $\mathcal{E}_e = |\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}|$.

A graphical representation of the above procedure applied to a subsonic laminar flow over a NACA 0012 airfoil is shown in Figure 3.3. The freestream flight condition is the same as those used in Figure 3.1. If we look at the fine space quantities involved in the error estimation procedure in Figures 3.3d–3.3f, the adaptive error indicator is high only if the residual is high and the output is sensitive to it, *i.e.*, adjoint is high. For example in regions behind the airfoil, the residual is high since the element size is large, however the error contribution from these areas is small as the adjoint is small in these regions.

The weighting provided by the adjoint solution brings in the global output sensitivity information, since the solution of the adjoint involves a residual Jacobian inverse which couples the system DOF together. For convection-dominated problems that are of particular interest in aerospace engineering, such as the flow problems at high-Reynolds numbers, the residual Jacobian inverse is usually a dense matrix even though the Jacobian itself is often a banded sparse matrix. Hence a local residual perturbation (inaccurate solution) can affect the approximation far away, *i.e.*, the error gets propagated through the convection nature of the system. More generally, if we consider the local residual error arising in the system as a source term, the *Green's* function of the original differential operator convolves with the residual source to produce a state solution error and thus an output error. The Green's function for elliptic problems decays logarithmically from the error source, and hence the residual error itself is often an effective indicator for adaptation. On the other hand, hyperbolic systems possess a Green's function which does not decay along the characteristic and hence the error is transmitted along the characteristic indefinitely [131]. Therefore, for convection-dominated problems in aerospace engineering, especially the flow problems involved in aerodynamic design, it is crucial to incorporate the error transport during the error localization and adaptation.

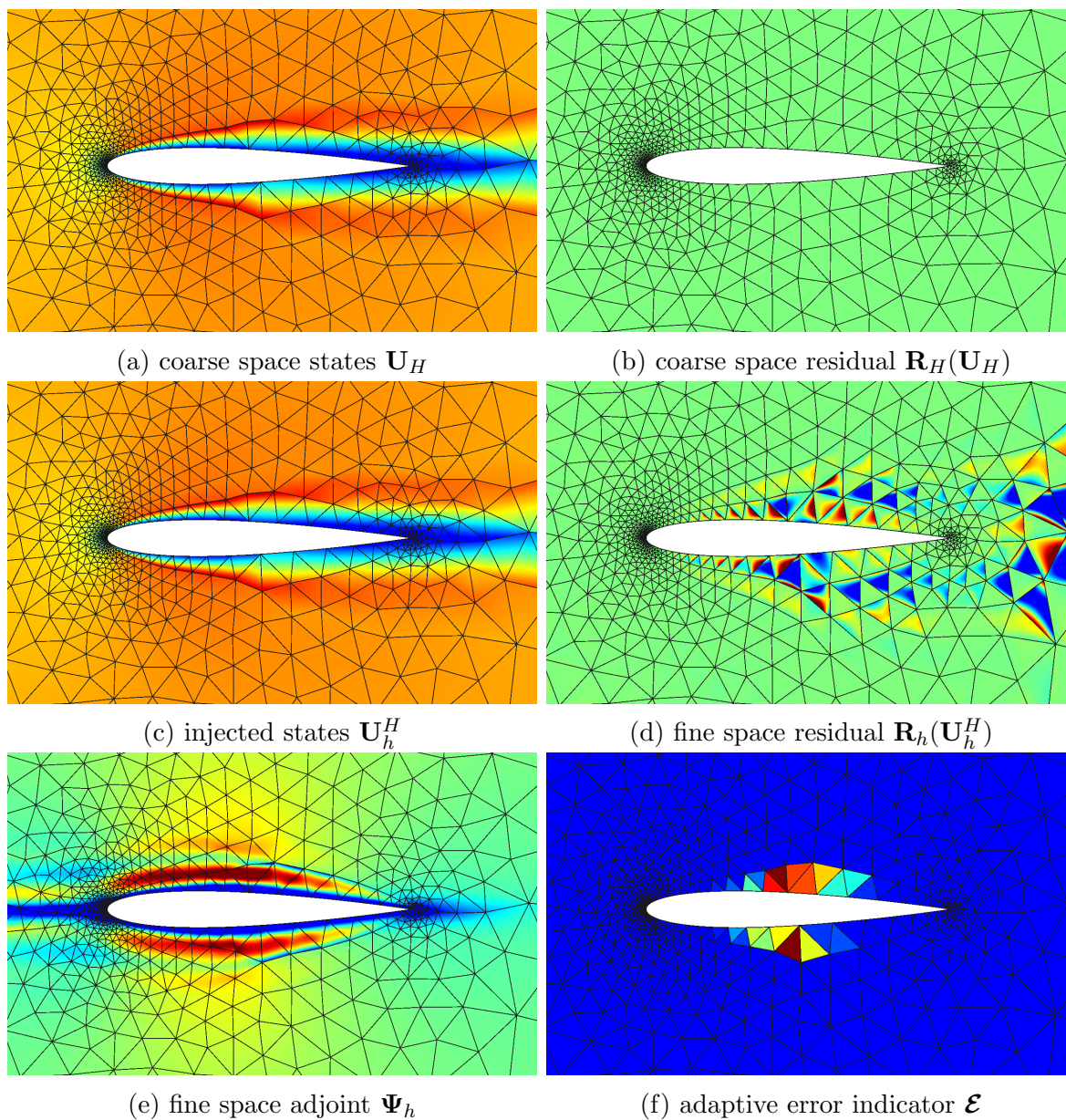


Figure 3.3: A summary of the error estimation and error localization procedure applied to a laminar flow simulation over a NACA 0012 airfoil. The airfoil angle of attack is $\alpha = 0^\circ$, and the freestream Reynolds number and Mach number are $Re_\infty = 5000$ and $M_\infty = 0.5$, respectively.

3.5 Adjoint Consistency

When we used the adjoint to either compute the output sensitivity or estimate the discretization induced output error, we assumed the discrete adjoint solution, $\boldsymbol{\psi}_h$ or $\boldsymbol{\Psi}_h$, obtained on a finite dimensional space \mathcal{V}_h , is a faithful representation of the exact adjoint solution $\boldsymbol{\psi}$. As discussed in Section 3.4, the convergence of the output functional depends on not only the primal problem but also the adjoint problem associated with the specific output. The discrete output obtained using the discrete solution only approaches the exact output if the adjoint error approaches zero as the discretization spacing vanishes, and so does the error in the discrete sensitivity analysis. The convergence of both the primal problem and the adjoint problem requires a consistent and stable discretization. The stability of the discretization is ensured by the numerical fluxes and appropriate solver setting described in Chapter 2, while presently, we are interested in the consistency. The impact of *adjoint consistency*, also referred to as *dual consistency*, has been studied by several authors in the context of DG methods [113, 124, 132]. We follow the definition proposed by Lu [113] in this work.

Primal consistency in variational form requires the exact primal solution $\mathbf{u} \in \mathcal{V}$ to satisfy the finite-dimensional weak form

$$\mathcal{R}_h(\mathbf{w}_h, \mathbf{u}) = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h, \quad (3.49)$$

where the finite dimensional approximation space is assumed to be a subset of the infinite dimensional space, $\mathcal{V}_h \subset \mathcal{V}$. Similarly, the output functional \mathcal{J}_h and the primal semilinear weak form \mathcal{R}_h are said to be adjoint-consistent or dual-consistent if the exact adjoint solution $\boldsymbol{\psi} \in \mathcal{V}$ is an admissible solution of the finite dimensional weak form of the adjoint equation

$$\mathcal{J}'_h[\mathbf{u}_h](\mathbf{w}_h) + \mathcal{R}'_h[\mathbf{u}_h](\mathbf{w}_h, \boldsymbol{\psi}) = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \quad (3.50)$$

Even if Eqn. 3.50 does not hold, a discretizations can still be *asymptotically dual consistent* if Eqn. 3.50 holds when the mesh spacing h vanishes,

$$\lim_{h \rightarrow 0} \left(\sup_{\mathbf{w}_h \in \mathcal{V}_h} \frac{|\mathcal{J}'_h[\mathbf{u}_h](\mathbf{w}_h) + \mathcal{R}'_h[\mathbf{u}_h](\mathbf{w}_h, \boldsymbol{\psi})|}{\|\mathbf{w}_h\|_{\mathcal{V}_h}} \right) = 0, \quad (3.51)$$

As indicated by both Eqn. 3.41 and Eqn. 3.44, adjoint solution error also affects the convergence of the primal approximation and is essential for output superconvergence [95, 123, 96, 113, 124, 125, 132]. The adjoint solution obtained from an adjoint-inconsistent

discretization bears irregular or spurious oscillations even if the exact solution is smooth, which provides inaccurate error estimates and hence drives mesh adaptation in incorrect areas [113, 124, 132]. Adjoint inconsistency in DG often occurs when incorrect interior treatment and boundary condition enforcement are adopted [113, 132], or when the source term discretization involves state gradients that require additional stabilization [124, 125], *e.g.*, the source term in RANS equations. In general, discretizations that are found to be adjoint inconsistent can often be made adjoint consistent by adding terms to either the semilinear weak form or the output functional.

CHAPTER 4

Aerodynamic Optimization Problem Formulation, Error Estimation and Mesh Adaptation

As discussed in Chapter 3, adjoint-based methods have shown much success in either output gradient computation for aerodynamic optimizations or output error estimation for standalone CFD simulations. However less work has been done to integrate these two applications, *i.e.*, employing adaptive CFD in an optimization problem. There are several main obstacles hampering the use of adjoint-based adaptive CFD in aerodynamic optimization, even though the reuse of adjoints in gradient-based optimization could potentially buy more efficiency in addition to the accuracy improvement. First of all, aerodynamic shape optimization often undergoes significant shape changes and hence vast flow field variations, which heavily relies on the robustness of the CFD solver to avoid solving failures during the optimization. Although more efficient than the prevalent second-order finite-volume methods, high-order finite-element methods such as DG, in which error estimation and mesh adaptation are often used, in general lack robustness, especially for steady-state problems due to less dissipation in the system. As a result, advanced adaptation techniques, such as anisotropic mesh adaptation or mesh optimization have to be used to not only improve the accuracy but also to help improve the robustness. Secondly, traditional error estimation and mesh adaptation techniques are often targeted for a single scalar output, while aerodynamic optimization problems usually involve multiple outputs, including the objective output and the constraint outputs. The errors in the constraint outputs may indirectly affect the accuracy of the objective and hence the optimization results. Therefore, a systematic way of incorporating the errors in both the objective and the constraints has to be developed for effectively estimating the error and adapting the mesh. Finally, the mesh topology and resolution changes in the mesh adaptation may be incompatible with traditional aerodynamic optimization frameworks. Thus, more carefully designed optimization frameworks should be developed to take advantage of the

various fidelity offered by adapted meshes and to achieve the best efficiency of adaptive CFD in optimizations.

The following two chapters will focus on tackling the issues mentioned above. In this chapter, we will formulate the constrained aerodynamic optimization problem with an adjoint-based approach in Section 4.1 and Section 4.2, based on which an objective error estimator is derived in Section 4.3 to take into account the effects of constraint output errors. A review of the anisotropic mesh adaptation techniques using the proposed error estimator is given in Section 4.4. The integration of the adaptive CFD and the traditional gradient-based optimization are covered in Chapter 5.

4.1 Continuous and Discrete Optimization

An aerodynamic shape optimization problem can be stated as a search for the design \mathbf{x} over the design space \mathcal{X} that minimizes a given objective function \mathcal{J} ,

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{J}(\mathbf{u}, \mathbf{x}) \quad \mathbf{u} \in \mathcal{V}, \mathbf{x} \in \mathcal{X}, \\ \text{s.t.} \quad & \mathcal{R}^e(\mathbf{u}, \mathbf{x}) = \mathbf{0}, \\ & \mathcal{R}^{ie}(\mathbf{u}, \mathbf{x}) \geq \mathbf{0}, \end{aligned} \tag{4.1}$$

where $\mathcal{J} : \mathcal{V} \times \mathcal{X} \rightarrow \mathbb{R}$ represents a scalar objective function, $\mathcal{R}^e : \mathcal{V} \times \mathcal{X} \rightarrow \mathbb{R}^{n_e}$ and $\mathcal{R}^{ie} : \mathcal{V} \times \mathcal{X} \rightarrow \mathbb{R}^{n_{ie}}$ denote n_e equality and n_{ie} inequality constraints, respectively. The objective and constraints are defined by the outputs (responses) of the flow equations, for example lift or drag, which consequently depend on the flow state variables \mathbf{u} . The state \mathbf{u} is the solution of the governing equations, lying in the solution space \mathcal{V} , which can be an infinite-dimensional space. The governing equations, often the Navier-Stokes equations or the Euler equations in the inviscid limit, can be represented by a semilinear form in a variational setting as presented in Chapter 2,

$$\mathcal{R}(\mathbf{u}, \mathbf{w}; \mathbf{x}) = 0, \quad \forall \mathbf{w} \in \mathcal{V}, \tag{4.2}$$

where the semilinear residual map $\mathcal{R} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ is the weak formulation of the flow equation, in which \mathbf{x} often presents as either boundary terms or source terms. The state $\mathbf{u} \in \mathcal{V}$ is solved within the design space \mathcal{X} to satisfy the governing equations, and this implicitly defines \mathbf{u} as a function of \mathbf{x} : $\mathbf{u} = \mathbf{u}(\mathbf{x})$. Moreover, the optimal design \mathbf{x} has to fall into the feasible space $\mathcal{F}(\mathcal{X}) = \{\mathbf{x} \in \mathcal{X} : \mathcal{R}^e(\mathbf{u}, \mathbf{x}) = \mathbf{0}, \mathcal{R}^{ie}(\mathbf{u}, \mathbf{x}) \geq \mathbf{0}\}$ that satisfies the constraints. Depending on the optimization algorithm, however, intermediate designs in an iterative process may not be in the feasible set.

Although the adjoint solution is not required in evaluating the objective output itself, it is needed for the output gradient calculations and error estimation purposes. Similarly, we rewrite the adjoint equation derived in Chapter 3,

$$\mathcal{J}'[\mathbf{u}](\mathbf{w}; \mathbf{x}) + \mathcal{R}'[\mathbf{u}](\mathbf{w}, \boldsymbol{\psi}; \mathbf{x}) = 0, \quad \forall \mathbf{w} \in \mathcal{V}. \quad (4.3)$$

which also implicitly defines the adjoint solution as a function of the design \mathbf{x} , $\boldsymbol{\psi} = \boldsymbol{\psi}(\mathbf{x})$. We generally cannot solve the PDEs in Eqn. 4.2 and Eqn. 4.3 analytically, and hence we discretize them on a finite-dimensional space over the computational domain, either in a weak formulation

$$\begin{aligned} \mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h; \mathbf{x}) &= 0, \quad \forall \mathbf{w}_h \in \mathcal{V}_h, \\ \mathcal{J}'_h[\mathbf{u}_h](\mathbf{w}_h; \mathbf{x}) + \mathcal{R}'_h[\mathbf{u}_h](\mathbf{w}_h, \boldsymbol{\psi}_h; \mathbf{x}) &= 0, \quad \forall \mathbf{w}_h \in \mathcal{V}_h; \end{aligned} \quad (4.4)$$

or in a fully-discrete form as

$$\begin{aligned} \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}) &= \mathbf{0}, \\ \left[\frac{\partial J_h}{\partial \mathbf{U}_h} \Big|_{\mathbf{U}_h, \mathbf{x}} \right]^T + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \Big|_{\mathbf{U}_h, \mathbf{x}} \right]^T \boldsymbol{\Psi}_h &= \mathbf{0}. \end{aligned} \quad (4.5)$$

In this work, we use the discrete adjoint approach, in which the adjoint equation is directly derived from the fully-discrete primal discretization as shown in Eqn. 4.5 rather than directly discretizing the exact adjoint equation Eqn. 4.3, *i.e.*, the continuous adjoint approach. Although both approaches should be consistent with each other for discretizations taking variational formulations, *e.g.*, finite element methods, and are expected to converge to the optimum design of the original continuous optimization problem Eqn. 4.1 if the discretization is both primal and dual consistent. However, subtle differences have been observed between the continuous and discrete adjoint approaches, especially for methods without a variational weak form, such as finite-difference and finite-volume methods. In short, the continuous approach provides an approximation of the gradient for the exact output functional \mathcal{J} , while the discrete adjoint approach offers an exact gradient (if the residual partial derivative is computed analytically) of the approximated output J_h . In practice, only the approximated output J_h is available in the optimization. The continuous adjoint approach may thus suffer from convergence problems as the approximated gradient of the exact output sometimes cannot predict the changes in the discretized output effectively. Therefore, the discrete adjoint approach is in general more favorable in practice besides its easier derivation and implementation in an implicit solver used in this

work.

In the rest of the exposition, we will only consider the fully-discrete form of the optimization problem,

$$\begin{aligned}
\min_{\mathbf{x}} \quad & J_h(\mathbf{U}_h, \mathbf{x}), \quad \mathbf{U}_h \in \mathbb{R}^N, \mathbf{x} \in \mathcal{X} \\
\text{s.t.} \quad & \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}) = \mathbf{0}, \\
& \mathbf{R}_h^e(\mathbf{U}_h, \mathbf{x}) = \mathbf{0}, \\
& \mathbf{R}_h^{\text{ie}}(\mathbf{U}_h, \mathbf{x}) \geq \mathbf{0}.
\end{aligned} \tag{4.6}$$

The equality and inequality constraints are also vectorized in the equation above. In this work, continuous and discrete optimization refer to the optimization governed by PDEs in continuous and discretized form, while in other contexts these terms may refer to the optimization with continuous and discrete design spaces.

4.2 Optimization Formulation via the Adjoint

Inactive inequality constraints, $\mathbf{R}_{\text{ia}}^{\text{ie}}$, do not affect the optimization explicitly, while the active ones, $\mathbf{R}_{\text{a}}^{\text{ie}}$, behave like equality constraints. We omit the subscript h here for simpler exposition. In general, the inequality constraints can also be transformed into equality constraints with non-negative slack variables [133]. For simplicity, we only consider the active inequality constraints and the equality constraints, put together into one vector of dimension N_t as trim constraints, $(\mathbf{R}^{\text{trim}})^T = [(\mathbf{R}^e)^T \ (\mathbf{R}_{\text{a}}^{\text{ie}})^T] \in \mathbb{R}^{N_t}$,

$$\mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) = \mathbf{J}^{\text{trim}}(\mathbf{U}, \mathbf{x}) - \bar{\mathbf{J}}^{\text{trim}} = \mathbf{0}, \tag{4.7}$$

where $\bar{\mathbf{J}}^{\text{trim}} \in \mathbb{R}^{N_t}$ is a set of N_t target trim outputs, for example, the target lift in an airfoil drag minimization problem. In order to distinguish the trim outputs from the objective output, we denote the latter by $J^{\text{adapt}} \in \mathbb{R}$, as the objective output is directly targeted for adaptation.

The Lagrangian function associated with Eqn. 4.6 that augments the flow equations and trim constraints can then be written as

$$\mathcal{L}(\mathbf{U}, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J^{\text{adapt}}(\mathbf{U}, \mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{R}(\mathbf{U}, \mathbf{x}) + \boldsymbol{\mu}^T \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}), \tag{4.8}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^N$ and $\boldsymbol{\mu} \in \mathbb{R}^{N_t}$ are the Lagrange multipliers associated with PDE constraints and the trim constraints, respectively. The dual problem for the original optimization problem can also be derived using the Lagrangian function in Eqn. 4.8. Although the

output function is often convex with respect to the states \mathbf{U} , the convexity with respect to the design variables \mathbf{x} is not guaranteed. Thus the strong duality is not guaranteed here, however, the *first-order necessary condition*, or the Karush-Kuhn-Tucker condition can still be applied assuming at least one local minimum exists for the discrete optimization problem,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} = \mathbf{0}, \quad (4.9a)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}} = \mathbf{0}, \quad (4.9b)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = \mathbf{R}(\mathbf{U}, \mathbf{x}) = \mathbf{0}, \quad (4.9c)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} = \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) = \mathbf{0}. \quad (4.9d)$$

The above optimality condition can be a large, coupled, nonlinear system, at least larger and more coupled compared to the primal and adjoint equations, especially for high-dimensional optimization problems. Solving this system in the full space is intractable for complex systems, when solving the original governing PDE is already challenging. One popular and most widely-used alternative in aerodynamic optimization is to solve the four subsystems individually and then coupled for the optimization, especially the primal and adjoint systems. These kind of methods are termed as *reduced space* methods in contrast to the *full space* methods, where the whole system is solved simultaneously. In this work, we used the former approach for several reasons. First of all, the PDE solver for large-scale primal and adjoint equations have been developed for decades and are now very efficient and robust. Another reason is that the whole system is often very ill-conditioned, whereas the four subsystems are typically better conditioned individually [134]. Additionally, most of the standard optimization packages fail with very high dimensional systems.

It's almost clear that the flow primal equations Eqn. 4.9c and the associated adjoint equation ¹ Eqn. 4.9b are solved by the flow solver, while the design stationarity conditions Eqn. 4.9a are handled by the optimizer. On the other hand, the trimming equations in Eqn. 4.9d can be either enforced by the optimizer or the flow solver. We will call the first approach optimizer-based trimming (OBT) and refer to the second one as solver-based trimming (SBT) in the rest of the thesis. We will cover both approaches here, however, most of the results in this thesis are obtained using the SBT approach, OBT results can be found in [67]. Although more expensive compared to the OBT approach, SBT has

¹We call this an adjoint equation since the derivation resembles the adjoint equations discussed earlier in Chapter 3.

been shown to be more robust, especially in multipoint optimizations [135]. Moreover, the OBT approach requires more knowledge in the optimizer when coupled with error estimation and mesh adaptation, while the SBT approach can be more easily coupled with an adaptive CFD framework. More details will be covered in Chapter 5.

4.2.1 Optimizer-Based Trimming

During the optimization, we solve the flow primal equations each time when the design updates. In other words, Eqn. 4.9c is always satisfied. Then we can choose the adjoint variables $\boldsymbol{\lambda}$ such that Eqn. 4.9b is enforced after each successful flow solve, *i.e.*, solving the adjoint equation,

$$\boldsymbol{\lambda}^T = - \left(\frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}} \right) \frac{\partial \mathbf{R}^{-1}}{\partial \mathbf{U}} = (\boldsymbol{\Psi}^{\text{adapt}} + \boldsymbol{\Psi}^{\text{trim}} \boldsymbol{\mu})^T. \quad (4.10)$$

We call both $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ coupled adjoint variables since they incorporate the adjoints of both the objective (adapt) and constraint (trim) outputs, $\boldsymbol{\Psi}^{\text{adapt}} \in \mathbb{R}^{N \times 1}$ and $\boldsymbol{\Psi}^{\text{trim}} \in \mathbb{R}^{N \times N_t}$, which are both solved by the flow solver using the discrete adjoint equations

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right)^T \boldsymbol{\Psi}^{\text{adapt}} + \left(\frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} \right)^T = \mathbf{0}, \quad \left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right)^T \boldsymbol{\Psi}^{\text{trim}} + \left(\frac{\partial \mathbf{J}^{\text{trim}}}{\partial \mathbf{U}} \right)^T = \mathbf{0}. \quad (4.11)$$

Now the optimality conditions in Eqn. 4.9 reduce to only two equations, Eqn. 4.9a and Eqn. 4.9d. Moreover, we can substitute Eqn. 4.10 into Eqn. 4.9a to evaluate the gradient of the Lagrangian function with respect to the design variables,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} \\ &= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \left[\frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \right] \\ &= \frac{dJ^{\text{adapt}}}{d\mathbf{x}} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}} = \mathbf{0}. \end{aligned} \quad (4.12)$$

The last equality is obtained via adjoint-based sensitivity analysis, where $d(\cdot)/d\mathbf{x}$ denotes the total derivative with respect to design variables by considering the states as an implicit function of \mathbf{x} , $\mathbf{U}(\mathbf{x})$.

Now the optimization problem has been reduced to finding an optimal design \mathbf{x} and the adjoint variable (Lagrange multiplier) $\boldsymbol{\mu}$ satisfying Eqn. 4.12 and Eqn. 4.9d. The

optimization process can then be equivalently written as

$$\begin{aligned}
 &\text{flow solver:} && \text{optimizer:} \\
 &\mathbf{R}(\mathbf{U}, \mathbf{x}) = \mathbf{0} && \frac{dJ^{\text{adapt}}}{d\mathbf{x}} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}} = \mathbf{0} \\
 &\frac{dJ^{\text{adapt}}}{d\mathbf{x}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} && \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) = \mathbf{0} \\
 &\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}} = \frac{\partial \mathbf{J}^{\text{trim}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} &&
 \end{aligned} \tag{4.13}$$

We can see above that in the OBT approach, both the design optimality (stationarity) condition and the trimming equations are handled in the optimizer, while the flow solver is only responsible for providing the output values and the sensitivity information. Note that although we have the expression for the adjoint variable $\boldsymbol{\lambda}$ in Eqn. 4.10, it's not explicitly solved or stored in the optimization. The flow solver solves both the flow primal and adjoint equations to evaluate the adapt and trim outputs as well as their gradients. The optimizer takes the output and gradient information to determine the optimal design and adjoint variables (Lagrange multipliers) $\boldsymbol{\mu}$ to satisfy the reduced optimality equation shown on the right in Eqn. 4.13. The optimization process is summarized by the flowchart shown in Figure 4.1.

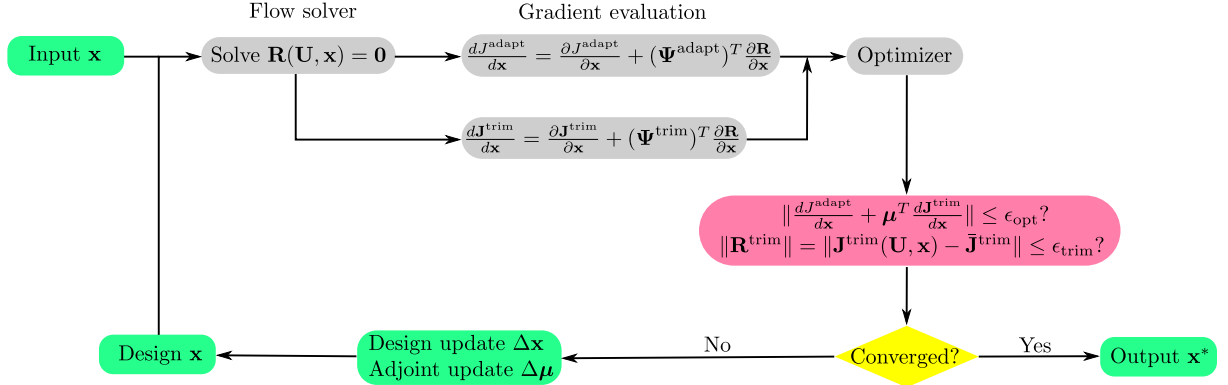


Figure 4.1: Flowchart of optimization using an optimizer-based trimming approach.

4.2.2 Solver-Based Trimming

In the SBT approach, both the flow equations and the trim equations are enforced by the flow solver. A set of design variables are dedicated to satisfying the trim constraints, denoted as trim variables \mathbf{x}_t , $\dim(\mathbf{x}_t) = \dim(\mathbf{J}^{\text{trim}}) = N_t$. For example, in aerodynamic optimizations, angle of attack can be the trim variable for the lift trimming constraint

while the wing or tail deflection can be the trim variable for the moment trimming constraint. The problem involved in the flow solver is then a coupled system,

$$\begin{aligned}\mathbf{R}(\mathbf{U}, \mathbf{x}_t, \mathbf{x}_s) &= \mathbf{0}, \\ \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}_t, \mathbf{x}_s) &= \mathbf{0},\end{aligned}\tag{4.14}$$

which implicitly defines both \mathbf{U} and \mathbf{x}_t as a function of the active design parameters \mathbf{x}_s , often the shape parameters in the original design parameter vector, $\mathbf{x} = [\mathbf{x}_t, \mathbf{x}_s]$. The trimming equation shown in Eqn. 4.14 is solved using a Newton-Raphson method, in which the trim variables are iteratively updated as shown Figure 4.2 to enforce the trimming equation.

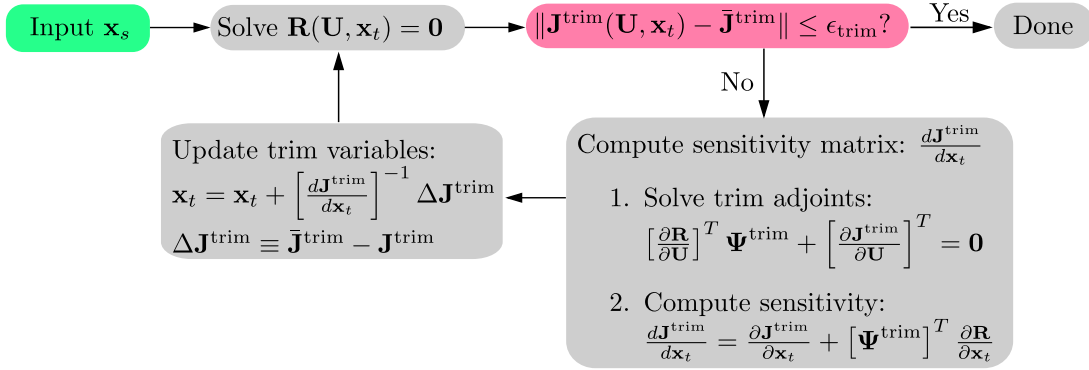


Figure 4.2: Trimming process using Newton-Raphson iteration.

This time, the KKT optimality condition in Eqn. 4.9 reduces to

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_s} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_s} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}_s} = \mathbf{0},\tag{4.15}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_t} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_t} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}_t} = \mathbf{0},\tag{4.16}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}} = \mathbf{0}.\tag{4.17}$$

Again, we can solve Eqn. 4.17 for the coupled adjoint $\boldsymbol{\lambda}$,

$$\boldsymbol{\lambda}^T = - \left(\frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}} \right) \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^{-1} = (\boldsymbol{\Psi}^{\text{adapt}} + \boldsymbol{\Psi}^{\text{trim}} \boldsymbol{\mu})^T.\tag{4.18}$$

Substituting the solution of $\boldsymbol{\lambda}$ into Eqn. 4.16, we can also obtain the solution of the

coupled adjoint variable $\boldsymbol{\mu}$ as

$$\begin{aligned}
\frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_t} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_t} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}_t} + \boldsymbol{\mu}^T (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_t} &= \mathbf{0} \\
\frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_t} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_t} + \boldsymbol{\mu}^T \frac{\partial \mathbf{J}^{\text{trim}}}{\partial \mathbf{x}_t} + \boldsymbol{\mu}^T (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_t} &= \mathbf{0} \\
\frac{dJ^{\text{adapt}}}{d\mathbf{x}_t} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} &= \mathbf{0} \\
\implies \boldsymbol{\mu}^T &= -\frac{dJ^{\text{adapt}}}{d\mathbf{x}_t} \left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} \right)^{-1}. \tag{4.19}
\end{aligned}$$

If we treat \mathbf{x}_t together with the flow states as the state vector for the coupled system in Eqn. 4.14, we can also derive $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as the adjoint variables associated with the states \mathbf{U} and \mathbf{x}_t [136]. This approach has also been used to couple different disciplines [137, 138, 139, 140, 141], for instance in an aerodynamic-structure coupled optimization problem, \mathbf{R} and \mathbf{R}^{trim} can be the flow and structure governing equations while \mathbf{U} and \mathbf{x}_t are the corresponding flow states and the structure states, respectively. If we substitute both Eqn. 4.18 and Eqn. 4.19 into Eqn. 4.15, the optimality conditions reduce to only one equation,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} &= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_s} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_s} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}_s} \\
&= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_s} + \boldsymbol{\mu}^T \left[\frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}_s} \right] \\
&= \frac{dJ^{\text{adapt}}}{d\mathbf{x}_s} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_s} \\
&= \frac{dJ^{\text{adapt}}}{d\mathbf{x}_s} - \frac{dJ^{\text{adapt}}}{d\mathbf{x}_t} \left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} \right)^{-1} \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_s} = \mathbf{0} \\
\implies \frac{DJ^{\text{adapt}}}{D\mathbf{x}_s} &= \underbrace{\frac{dJ^{\text{adapt}}}{d\mathbf{x}_s}}_{\text{adapt output gradients}} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_s} = \underbrace{\frac{dJ^{\text{adapt}}}{d\mathbf{x}_t} \left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} \right)^{-1} \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_s}}_{\text{trim correction}} = \mathbf{0}. \tag{4.20}
\end{aligned}$$

In Eqn. 4.20, $DJ^{\text{adapt}}/D\mathbf{x}_s$ is the total gradient of the adapt output with respect to the active design parameter \mathbf{x}_s when considering both the flow states and the trim variables as implicit functions of the active design \mathbf{x}_s . The derivatives $d(\cdot)/d\mathbf{x}_s$ or $d(\cdot)/d\mathbf{x}_t$, on the other hand, denote the output gradients with respect to the trim or active design parameters when the rest of the design variables are fixed. For instance, $dJ^{\text{adapt}}/d\mathbf{x}_t$ is computed via the adjoint method when \mathbf{x}_s is fixed and only considering the states \mathbf{U} as

an implicit function of the trim variables, $\mathbf{U}(\mathbf{x}_t)$, *i.e.*, in the trimming subproblem shown in Figure 4.2.

Now the original optimization problem has been reduced to an unconstrained optimization problem for the optimizer since both the PDE constraints and trim constraints are enforced by the flow solver. The optimization problem can then be equivalently written as

$$\begin{aligned}
 &\text{flow solver:} && \text{optimizer:} \\
 &\mathbf{R}(\mathbf{U}, \mathbf{x}_t, \mathbf{x}_s) = \mathbf{0} && \frac{DJ^{\text{adapt}}}{D\mathbf{x}_s} = \mathbf{0} \\
 &\mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}_t, \mathbf{x}_s) = \mathbf{0} && \\
 &\boldsymbol{\mu}^T = -\frac{dJ^{\text{adapt}}}{d\mathbf{x}_t} \left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} \right)^{-1} && \\
 &\frac{DJ^{\text{adapt}}}{D\mathbf{x}_s} = \frac{dJ^{\text{adapt}}}{d\mathbf{x}_s} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_s} && \tag{4.21}
 \end{aligned}$$

The optimization process using the SBT approach is summarized by the flowchart shown in Figure 4.3.

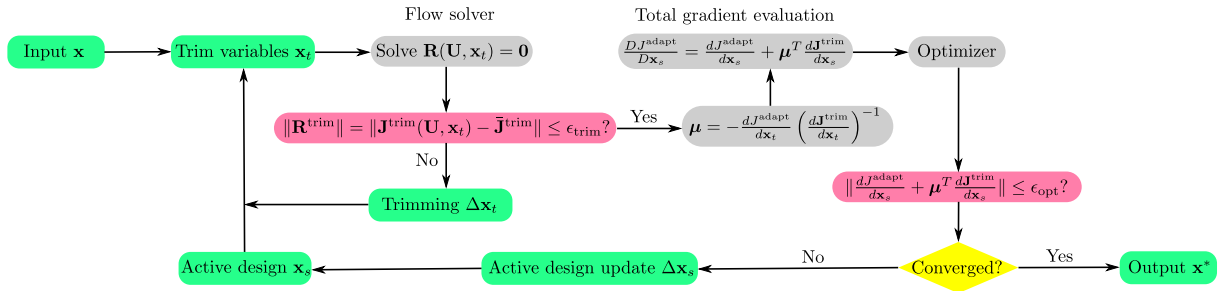


Figure 4.3: Flowchart of optimization using a solver-based trimming approach.

4.3 Output Error Estimation for Optimization

Optimization using either the OBT or SBT approach is expected to converge to the optimum of the discrete optimization problem in Eqn. 4.6 and hence approaches the optimum of the continuous optimization problem in Eqn. 4.1. However, since in a practical calculation, on a finite-dimensional space, the discretization error appears in both the flow equations and the adjoint equations, optimality cannot be guaranteed even when Eqn. 4.9 is satisfied. In order to control discretization errors during the optimization, we introduce adaptive CFD into gradient-based optimization which will be covered in details in the following sections.

4.3.1 Output Error Estimation for Standalone Simulations

We follow the definition used in Chapter 3 for the output error, which is the difference between outputs evaluated with the coarse and fine space solutions. Given a fixed design, if we consider two standalone simulations performed on both the coarse and fine spaces,

$$\begin{aligned} \text{Coarse space: } & \text{design } \mathbf{x} \rightarrow \mathbf{R}_H(\mathbf{U}_H, \mathbf{x}) = \mathbf{0} \rightarrow \mathbf{U}_H \rightarrow J_H(\mathbf{U}_H, \mathbf{x}) \\ \text{Fine space: } & \text{design } \mathbf{x} \rightarrow \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}) = \mathbf{0} \rightarrow \mathbf{U}_h \rightarrow J_h(\mathbf{U}_h, \mathbf{x}) \end{aligned} \quad (4.22)$$

we would normally expect difference in the outputs on the coarse and fine spaces. The difference can then be estimated and used as a surrogate for the output error,

$$\begin{aligned} \delta J &= J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \\ &= J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) = \frac{\partial J_h}{\partial \mathbf{U}_h} \delta \mathbf{U} \\ &= -\Psi_h^T \delta \mathbf{R}_h = -\Psi_h^T [\mathbf{R}_h(\mathbf{U}_h^H) - \mathbf{R}_h(\mathbf{U}_h)] \\ &= -\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H) \end{aligned} \quad (4.23)$$

where Ψ_h is the fine space adjoint and \mathbf{U}_h^H is the state injected into the fine space from the coarse one, which generally will not give a zero fine space residual, $\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}$. The error estimate given in Eqn. 4.23 originates from the small perturbation assumptions, and is valid for outputs whose definition does not change between the coarse and fine spaces, $J_H(\mathbf{U}_H) = J_h(\mathbf{U}_h^H)$. Detailed derivations of Eqn. 4.23 can be found in Chapter 3.

4.3.2 Output Error Estimation for Optimization Problems

Normally, error estimation is applied only to the output in which we are most interested, *i.e.*, the objective (adapt) output. However, our optimization problem requires simultaneous solutions of the flow equations and the trim equations, either loosely coupled by the optimizer in the OBT approach or strongly coupled by the flow solver in the SBT approach. Since the discretization errors exist in both the adapt and trim outputs, and the trim output errors may indirectly affect the calculation of the objective [136], to take this effect into account, the coupled adjoint should be used for the error estimation.

Consider a given fixed design in both the coarse and fine spaces, $\mathbf{x}_H = \mathbf{x}_h = \mathbf{x}$, which is of the same setting as in Eqn. 4.22. Since the design is fixed, the discretization error of the objective output only comes from the inexact states solution \mathbf{U}_H . We can estimate

the error using the linearization of the objective output,

$$\begin{aligned}
\delta J^{\text{adapt}}(\mathbf{x}_H) &= J_H^{\text{adapt}}(\mathbf{U}_H, \mathbf{x}_H) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_h) \\
&= J_h^{\text{adapt}}(\mathbf{U}_h^H, \mathbf{x}_H) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_H) \\
&= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_h} \delta \mathbf{x}^0 \\
&= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U}.
\end{aligned} \tag{4.24}$$

Since Eqn. 4.9a is always satisfied during the optimization as discussed earlier, we can substitute it into the equation above,

$$\begin{aligned}
\delta J^{\text{adapt}}(\mathbf{x}_H) &= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} = -\boldsymbol{\lambda}_h^T \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \delta \mathbf{U} - \boldsymbol{\mu}_h^T \frac{\partial \mathbf{R}_h^{\text{trim}}}{\partial \mathbf{U}_h} \delta \mathbf{U} \\
&= -\boldsymbol{\lambda}_h^T \delta \mathbf{R}_h - \boldsymbol{\mu}_h^T \delta \mathbf{R}_h^{\text{trim}} \\
&= -\boldsymbol{\lambda}_h^T [\mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_H)] - \boldsymbol{\mu}_h^T [\mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) - \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_H)] \\
&= -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T [\mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) - \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_H)] \\
&= -(\boldsymbol{\Psi}_h^{\text{adapt}} + \boldsymbol{\Psi}_h^{\text{trim}} \boldsymbol{\mu}_h)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) \\
&= -(\boldsymbol{\Psi}_h^{\text{adapt}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T (\boldsymbol{\Psi}_h^{\text{trim}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) \\
&= -(\boldsymbol{\Psi}_h^{\text{adapt}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) - \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) \\
&= -(\boldsymbol{\Psi}_h^{\text{adapt}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H),
\end{aligned} \tag{4.25}$$

where $\delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H)$ is a vector containing error estimates of the N_t trim outputs at the fixed design \mathbf{x}_H . Following the definition of the output error in Eqn. 4.23, we have

$$\begin{aligned}
\delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) &= \mathbf{J}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_H) - \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_H) = \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) - \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_H) \\
&= -(\boldsymbol{\Psi}_h^{\text{trim}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H).
\end{aligned} \tag{4.26}$$

Eqn. 4.25 is consistent with the previous analysis without the trim conditions, since we keep the design fixed between the coarse and fine spaces, and because we assume that the error only comes from the inexact state solution \mathbf{U}_H . In general, however, we need to deal with both the objective error and the constraints error. The problem becomes worse if we have high accuracy in the objective while little confidence in the constraint outputs, or vice versa. If we run the optimization on the fine space and the coarse space, even with the same target constraint outputs, we will generally obtain different designs. This difference may come from the deviation of both the design parameters and the flow states,

and separate error estimation and mesh adaptation for the objective and trim outputs can be inefficient.

Before estimating the objective error in an optimization problem, let's start with a simpler estimate, more specific to the SBT approach. Suppose that we have the fixed active design $\mathbf{x}_s = \mathbf{x}_{s,H} = \mathbf{x}_{s,h}$ on both the coarse and fine spaces. The trimming enforcement often results in different trim variables and flow states,

$$\begin{aligned} \text{Coarse space: active design } \mathbf{x}_s &\rightarrow \begin{cases} \mathbf{R}_H(\mathbf{U}_H, \mathbf{x}_t) = \mathbf{0} \\ \mathbf{R}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_t) = \mathbf{0} \end{cases} \rightarrow \mathbf{U}_H, \mathbf{x}_{t,H} \rightarrow J_H(\mathbf{U}_H, \mathbf{x}_{t,H}) \\ \text{Fine space: active design } \mathbf{x}_s &\rightarrow \begin{cases} \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_t) = \mathbf{0} \\ \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_t) = \mathbf{0} \end{cases} \rightarrow \mathbf{U}_h, \mathbf{x}_{t,h} \rightarrow J_h(\mathbf{U}_h, \mathbf{x}_{t,h}). \end{aligned} \quad (4.27)$$

Again, linearizing the adapt output gives

$$\begin{aligned} \delta J^{\text{adapt}}(\mathbf{x}_{s,H}) &= J_H^{\text{adapt}}(\mathbf{U}_H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_{t,h}, \mathbf{x}_{s,H}) \\ &= J_h^{\text{adapt}}(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_{t,h}, \mathbf{x}_{s,H}) \\ &= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_{t,h}} \delta \mathbf{x}_t + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_{s,h}} \delta \mathbf{x}_s \rightarrow \mathbf{0} \\ &= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_{t,h}} \delta \mathbf{x}_t. \end{aligned} \quad (4.28)$$

Substituting in both Eqn. 4.16 and Eqn. 4.17 which are always enforced in the SBT approach, we can write the error estimate above also in a dual weighted residual form,

$$\begin{aligned} \delta J^{\text{adapt}}(\mathbf{x}_{s,H}) &= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_{t,h}} \delta \mathbf{x}_t \\ &= -\boldsymbol{\lambda}_h^T \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_{t,h}} \delta \mathbf{x}_t \right) - \boldsymbol{\mu}_h^T \left(\frac{\partial \mathbf{R}_h^{\text{trim}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial \mathbf{R}_h^{\text{trim}}}{\partial \mathbf{x}_{t,h}} \delta \mathbf{x}_t \right) \\ &= -\boldsymbol{\lambda}_h^T \delta \mathbf{R}_h - \boldsymbol{\mu}_h^T \delta \mathbf{R}_h^{\text{trim}} \\ &= -\boldsymbol{\lambda}_h^T [\mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_{t,h}, \mathbf{x}_{s,H})] \\ &\quad - \boldsymbol{\mu}_h^T [\mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_{t,h}, \mathbf{x}_{s,H})] \\ &= -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \boldsymbol{\mu}_h^T \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}). \end{aligned} \quad (4.29)$$

Since the trim equations are solved on both the coarse and fine spaces, and as the definition of the outputs is often the same on the coarse space and the fine one, we expand the second

residual term above as

$$\begin{aligned}
\mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) &= \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \bar{\mathbf{J}}^{\text{trim}} \\
&= \mathbf{J}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \bar{\mathbf{J}}^{\text{trim}} \\
&= \mathbf{R}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) = \mathbf{0}.
\end{aligned} \tag{4.30}$$

Hence Eqn. 4.29 can be simplified as

$$\begin{aligned}
\delta J^{\text{adapt}}(\mathbf{x}_{s,H}) &= -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) \\
&= -(\boldsymbol{\Psi}_h^{\text{adapt}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) - \boldsymbol{\mu}_h^T (\boldsymbol{\Psi}_h^{\text{trim}})^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_{t,H}, \mathbf{x}_{s,H}) \\
&= \delta J^{\text{adapt}}(\mathbf{x}_{t,H}, \mathbf{x}_{s,H}) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_{t,H}, \mathbf{x}_{s,H}) \\
&= \delta J^{\text{adapt}}(\mathbf{x}_H) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H),
\end{aligned} \tag{4.31}$$

where $\delta J^{\text{adapt}}(\mathbf{x}_H)$ and $\delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H)$ are the standalone error estimates for the adapt output and trim outputs, respectively, assuming the error only comes from the inaccurate flow states. The first term $\delta J^{\text{adapt}}(\mathbf{x}_H)$ represents directly the error from the inaccurate states on the coarse space, \mathbf{U}_H , while the second term $\boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H)$ accounts for the correction from the inaccurate trim constraints satisfaction. This error estimate is valid for the trimming problem when the active design is fixed, or in an optimization problem ignoring the error in the objective due to inaccurate active design parameters. When we say inaccurate active design parameters, we have to measure them at the optimal design, *i.e.*, when the KKT conditions hold, since if we consider two separate optimization problems on the coarse and fine spaces, the search path or the intermediate active designs are by no means guaranteed to be the same or close, and thus the difference between the active design variables are meaningless if the design is not close to optimal.

If we consider two optimization problems on the coarse and fine spaces, the optimal designs \mathbf{x}^* (including \mathbf{x}_t and \mathbf{x}_s) are again expected to be different,

$$\begin{aligned}
\text{Coarse space: } & \text{initial design } \mathbf{x} \rightarrow \text{optimization} \rightarrow \mathbf{U}_H, \mathbf{x}_H^* \rightarrow J_H(\mathbf{U}_H, \mathbf{x}_H^*) \\
\text{Fine space: } & \text{initial design } \mathbf{x} \rightarrow \text{optimization} \rightarrow \mathbf{U}_h, \mathbf{x}_h^* \rightarrow J_h(\mathbf{U}_h, \mathbf{x}_h^*).
\end{aligned} \tag{4.32}$$

A complete error estimate for the objective output error for the optimal design on the

coarse space can be written as

$$\begin{aligned}
\delta J_{\text{opt}}^{\text{adapt}} &= J_H^{\text{adapt}}(\mathbf{U}_H, \mathbf{x}_H^*) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_h^*) \\
&= J_h^{\text{adapt}}(\mathbf{U}_h^H, \mathbf{x}_H^*) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_h^*) \\
&= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}_h} \delta \mathbf{x}.
\end{aligned} \tag{4.33}$$

The equation above also implies that the error should be measured around the optimal design, since the intermediate designs on the coarse and fine spaces can be very different such that the linearization above may not hold anymore. At the optimal designs ², we have that the KKT conditions Eqn. 4.9 all hold now and thus we can rewrite the error estimate by plugging in both Eqn. 4.9a and Eqn. 4.9b,

$$\begin{aligned}
\delta J_{\text{opt}}^{\text{adapt}} &= J_h^{\text{adapt}}(\mathbf{U}_h^H, \mathbf{x}_H^*) - J_h^{\text{adapt}}(\mathbf{U}_h, \mathbf{x}_h^*) \\
&= \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial J_h^{\text{adapt}}}{\partial \mathbf{x}} \delta \mathbf{x} \\
&= -\boldsymbol{\lambda}_h^T \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial \mathbf{R}_h}{\partial \mathbf{x}} \delta \mathbf{x} \right) - \boldsymbol{\mu}_h^T \left(\frac{\partial \mathbf{R}_h^{\text{trim}}}{\partial \mathbf{U}_h} \delta \mathbf{U} + \frac{\partial \mathbf{R}_h^{\text{trim}}}{\partial \mathbf{x}} \delta \mathbf{x} \right) \\
&= -\boldsymbol{\lambda}_h^T \delta \mathbf{R}_h - \boldsymbol{\mu}_h^T \delta \mathbf{R}_h^{\text{trim}} \\
&= -\boldsymbol{\lambda}_h^T [\mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) - \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_h^*)] - \boldsymbol{\mu}_h^T [\mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H^*) - \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_h^*)] \\
&= -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) - \boldsymbol{\mu}_h^T \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H^*)
\end{aligned} \tag{4.34}$$

Similarly, the second term vanishes if the trim equations are also enforced on the coarse space, which is true for the coarse space optimized design \mathbf{x}_H^* ,

$$\begin{aligned}
\mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H^*) &= \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H^*) - \bar{\mathbf{J}}^{\text{trim}} \\
&= \mathbf{J}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_H^*) - \bar{\mathbf{J}}^{\text{trim}} \\
&= \mathbf{R}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_H^*) = \mathbf{0}.
\end{aligned} \tag{4.35}$$

Therefore, the complete error estimate for the adapt output of the optimal design on the

²Here, we also assume that the discrete optimization problem only has one local optimum (convex) such that the optimal designs on the coarse and fine spaces are close.

coarse space can be simplified as

$$\begin{aligned}
\delta J_{\text{opt}}^{\text{adapt}} &= \boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) \\
&= - \left(\boldsymbol{\Psi}_h^{\text{adapt}} + \boldsymbol{\Psi}_h^{\text{trim}} \boldsymbol{\mu}_h \right)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) \\
&= - \left(\boldsymbol{\Psi}_h^{\text{adapt}} \right)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) - \boldsymbol{\mu}_h^T \left(\boldsymbol{\Psi}_h^{\text{trim}} \right)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H^*) \\
&= \delta J^{\text{adapt}}(\mathbf{x}_H^*) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H^*),
\end{aligned} \tag{4.36}$$

which is very similar to the error estimate given in Eqn. 4.31. However, Eqn. 4.31 assumes the same active design \mathbf{x}_s on both the coarse and fine spaces, which provides the estimate for the objective error in a constrained trimming problem, *i.e.*, the trimming condition is always satisfied, such as in the SBT approach or in OBT when the optimizer only searches along the feasible path. On the other hand, Eqn. 4.36 takes into account both the error from the states and the error from the complete design parameters, \mathbf{x}_t and \mathbf{x}_s , but is only valid at the optimal design point.

During the optimization process, the evaluations of Eqn. 4.31 and Eqn. 4.36 are in the equivalent form,

$$\delta J_{\text{eff}}^{\text{adapt}} = \delta J^{\text{adapt}}(\mathbf{x}_H) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H). \tag{4.37}$$

However, the interpretation is different. If in the form of Eqn. 4.31, we assume a fine space trimming flow simulation using the same active design, $\mathbf{x}_{s,h} = \mathbf{x}_{s,H}$, and we estimate the difference between the adapt outputs on the coarse and fine spaces. On the other hand, if in the form of Eqn. 4.36, it is neither the error of the objective nor the error of the trim constraints when the design is away from optimal due to different design update paths on the coarse and fine spaces; however, if measured at the coarse space optimal design \mathbf{x}_H^* , it estimates the objective error taking effects from both the trim parameter difference and the active design difference between the coarse and fine spaces. In the SBT approach, the flow equations and the trimming equations are always solved as a coupled system, such that the error estimate given by $\delta J_{\text{eff}}^{\text{adapt}}$ in Eqn. 4.37 is valid as an estimate for the constrained flow simulations. While in the OBT approach where the trimming equations are not necessary to hold for intermediate designs, yet $\delta J_{\text{eff}}^{\text{adapt}}$ in Eqn. 4.37 can still be used as a more effective estimate than the objective error alone, to measure the error level of the entire optimization and to provide effective indicators for adaptation in optimization problems [67]. A physical interpretation of the adjoint variable $\boldsymbol{\mu}_h$ is that it provides the sensitivity of the objective output with respect to the trim outputs if the trim constraints are enforced, as indicated by its solution in Eqn. 4.19, and therefore it provides effective weighting to include trim output errors when we estimate the objective

output error.

4.3.3 Implementation

With the advantages of adjoint-based error estimation, we avoid the expensive solutions of both the optimal design \mathbf{x}_h^* and the flow states \mathbf{U}_h , *i.e.*, the whole optimization process on the fine space. This is because the error estimate requires only the flow and trim residuals, which can be computed from \mathbf{x}_H and \mathbf{U}_H , without access to the fine-space flow states or the optimal design. However, the estimation requires the fine-space coupled adjoint, either $\boldsymbol{\lambda}_h$ or $\boldsymbol{\mu}_h$, to effectively combine the adapt output adjoint $\boldsymbol{\Psi}_h^{\text{adapt}}$ and the trim output adjoints $\boldsymbol{\Psi}_h^{\text{trim}}$. Although the adjoint variable $\boldsymbol{\lambda}_h$ appears in the derivation of the error estimates, it is not explicitly solved or stored in either the SBT or the OBT approach, instead only $\boldsymbol{\mu}_h$ is solved for the error estimation as shown in the final formula.

Depending on the handling of the trimming equations, $\boldsymbol{\mu}_h$ is approximated differently. In the OBT approach, the coupled adjoint is defined and updated via the optimizer, hence the coarse space adjoint $\boldsymbol{\mu}_H$ is extracted from the optimizer and used to approximate the fine space adjoint $\boldsymbol{\mu}_h$. In the SBT approach, the coupled adjoint $\boldsymbol{\mu}_H$ is solved inside the flow solver during the trimming iterations. Similar to the adjoint approximation shown by Eqn. 3.46 in Section 3.4, we can approximate the fine space adjoint $\boldsymbol{\mu}_h$ by solving the fine space adjoint problem Eqn. 4.19 linearized at a fine space smoothed states $\tilde{\mathbf{U}}_h$. In the current work, $\boldsymbol{\mu}_H$ is directly used in both the SBT and OBT approaches for simplicity.

With the solution of the coupled adjoint $\boldsymbol{\mu}_h$, the error estimate requires a combination of both the adapt and trim outputs errors, weighted by $\boldsymbol{\mu}_h$. The output error estimates require $1 + N_t$ fine space adjoint solutions, 1 for the adapt output and N_t for the trim outputs. For the current work, N_t is small such that the computational cost increase for the error estimate is relatively low. However, for optimization problems with a large set of trimming constraints, the fine space adjoint solutions can add a considerable amount of computational burden. Alternatively, we can define a combined output J^c as

$$J^c = J^{\text{adapt}} + \boldsymbol{\mu}_h^T \mathbf{J}^{\text{trim}}, \quad (4.38)$$

which resembles the Lagrangian function of the optimization problem. Then for error estimation, we can solve the adjoint only once for the combined output J^c , which is of similar cost to one normal output error estimate [142].

4.4 Mesh Adaptation Incorporating Anisotropy

4.4.1 Error Localization

For a single output of interest, the error estimate can be localized in each element and serves as an indicator for mesh adaptation. A common approach is to keep track of the elemental error contribution, taking its absolute value as the indicator. More details can be found in Section 3.4.1.

For the optimization problem, the error estimates given in Eqn. 4.37 can also be localized in each element to provide indicators for adaptation,

$$\begin{aligned}
 \mathcal{E}_e &= |\delta J_{h,e}^{\text{adapt}} + \boldsymbol{\mu}_h^T \delta \mathbf{J}_{h,e}^{\text{trim}}| \\
 &= |(\boldsymbol{\Psi}_{h,e}^{\text{adapt}})^T \mathbf{R}_{h,e}(\mathbf{U}_h^H, \mathbf{x}_H) + \boldsymbol{\mu}_h^T (\boldsymbol{\Psi}_{h,e}^{\text{trim}})^T \mathbf{R}_{h,e}(\mathbf{U}_h^H, \mathbf{x}_H)| \\
 &\leq |(\boldsymbol{\Psi}_{h,e}^{\text{adapt}})^T \mathbf{R}_{h,e}(\mathbf{U}_h^H, \mathbf{x}_H)| + |\boldsymbol{\mu}_h^T (\boldsymbol{\Psi}_{h,e}^{\text{trim}})^T \mathbf{R}_{h,e}(\mathbf{U}_h^H, \mathbf{x}_H)| \\
 &\leq \mathcal{E}_e^{\text{adapt}} + |\boldsymbol{\mu}_h^T| \mathcal{E}_e^{\text{adapt}} = \mathcal{E}_{e,\text{con}}
 \end{aligned} \tag{4.39}$$

where \mathcal{E}_e is the indicator that allows cancellations between objective and constraints error indicators, $\mathcal{E}_e^{\text{adapt}}$ and $\mathcal{E}_e^{\text{trim}}$, while $\mathcal{E}_{e,\text{con}}$ provides a more conservative error indicator for the optimization problem.

Given the localized error indicator, sometimes also called the adaptive indicator, there exist many strategies to modify/adapt the discretization, including the computational mesh and the approximation order, for reducing the output error. In CFD applications, the most popular adaptation strategy is h -adaptation, in which only the computational mesh is modified. h -adaptation often consists of mesh refinement, or possibly mesh coarsening if high efficiency is desired. Pure mesh node movement can also be used to reduce the error, including the manipulation of the geometry nodes (r -adaptation) [143, 144, 145] and high-order nodes in curved elements (q -adaptation) [146, 147]. For high-order methods, we can also enrich the approximation space locally by increasing the approximation order p , usually termed p -adaptation. For solutions exhibiting discontinuities, for example the shocks that are often seen in transonic aerodynamic optimizations, h -adaptation is more effective by isolating these regions. Although in the meantime, p -adaptation can be performed in areas where the solution is smooth to further improve the efficiency. In this work, we currently only consider standard h -adaptation where mesh refinement/coarsening is used to improve the accuracy. p -, q -, r -adaptation techniques will be investigated in future work.

In standard h -adaptation, the simplest way of adapting the mesh is through local mesh modification, which is applicable to both structured and unstructured meshes. Of-

ten the local mesh coarsening needs information of surrounding elements, which adds more complexity in the implementation. Thus, local mesh adaptation often only supports refinement where local element is subdivided to enrich the local approximation. Another approach to adapting the mesh is global re-meshing, in which a new mesh is generated for the entire computational domain. This approach is mostly used in unstructured mesh adaptation, as it often offers more flexibility in mesh generation and hence adaptation. In this work, we use unstructured simplex meshes and the adaptation is based on global re-meshing, where we allow mesh refinement and coarsening, as well as mesh DOF redistribution to achieve the best efficiency.

In CFD simulations, one featured phenomenon is the strong directional dependence of the flow solutions, such as the boundary layer or shocks, where the flow quantities vary dramatically along one direction while rarely change along another. In these scenarios, anisotropic mesh resolution, *i.e.*, appropriate mesh stretching, is essential to efficiently resolve the directional flow features. The error indicator in Eqn. 4.39, however, provides no directional information to support anisotropic mesh adaptation. As a result, the mesh anisotropy information has to be explicitly incorporated with the output error indicator to enable efficient output based mesh adaptation. The mesh anisotropy information can come either from directional solution error estimates through interpolation theory [148, 149, 150, 97, 151, 152, 153, 154, 155], or directly from anisotropic output error convergence properties via a sampling approach [156, 157]. We consider both approaches. More specifically, one adopts the solution Hessian-based anisotropy and the other uses the output sampling-based anisotropy, while both rely on global re-meshing in the adaptation. Detailed algorithms will be shown in Section 4.4.3 and Section 4.4.4, respectively.

4.4.2 Continuous Mesh Framework

Before we jump into the detailed anisotropic mesh adaptation algorithms, we first introduce the continuous mesh framework that supports global re-meshing. In global re-meshing, the original mesh, or background mesh, is used to store the desired mesh characteristics, including mesh sizes and stretching (anisotropy), which are different from the current mesh. During the adaptation, a global mesh regeneration is performed such that the new mesh matches the desired mesh characteristics. The mesh characteristic information is often stored using a *Riemannian metric field*.

A Riemannian metric field, $\mathcal{M}(\vec{x})$, is a smoothly varying field of symmetric positive definite (SPD) tensors that can be used to encode anisotropic information of the computational mesh, including desired mesh sizes and stretching directions. The metric tensor, $\mathcal{M}(\vec{x})$, provides a measure of distance under the metric: for any two arbitrary points a

and b , the length of the vector \vec{ab} under the metric, $l_{\mathcal{M}}(\vec{ab})$, is defined as

$$l_{\mathcal{M}}(\vec{ab}) = \int_0^1 \sqrt{\vec{ab}^T \mathcal{M}(\vec{x}_a + \vec{ab}s) \vec{ab}} ds, \quad \vec{ab} = \vec{x}_b - \vec{x}_a, \quad \forall \vec{x}_a, \vec{x}_b \in \mathbb{R}^d, \quad (4.40)$$

where \vec{x}_a and \vec{x}_b are the spatial coordinates of the point a and b . After choosing a Cartesian coordinate system and bases for the d -dimensional physical space, the metric field \mathcal{M} can be represented by a $d \times d$ SPD matrix. The set of points at the unit metric distance is an ellipse in two-dimension and an ellipsoid in three-dimension spaces: eigenvectors of \mathcal{M} give directions of its principal axes, while the length of each axis (stretching) is the inverse square root of the corresponding eigenvalue. A two-dimensional metric field is illustrated in Figure 4.4.

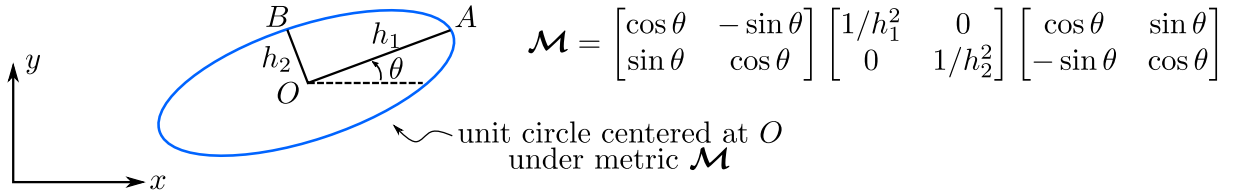


Figure 4.4: In two dimensions, the set of points equidistant to a point O under the metric measure \mathcal{M} defines an ellipse. The two principal axes OA and OB are on the directions of the eigenvectors of \mathcal{M} , while their lengths are defined by the corresponding eigenvalues.

Given a desired metric field, a mesh that conforms to a metric field is one in which each edge has the same length under the metric per Eqn. 4.40, to some tolerance. The *metric-conforming* mesh is not unique; however, a family of metric-conforming meshes have similar approximation properties [158, 156]. On the other hand, with a given computational mesh, the *mesh-implied metric* field can be obtained by solving a linear system for $d(d+1)/2$ independent entries at each element, which determines the elemental metric, \mathcal{M}_e . For simplex elements, the equations in this system enforce that each of the $d(d+1)/2$ edges has unit metric length, while for non-simplex elements least squares can be used to determine the elemental metric. The metric-conforming mesh and mesh-implied metric offer a way of converting between an anisotropic mesh and a Riemannian metric field, a so called *mesh-metric duality*. There are several mesh generators or libraries that support metric-conforming mesh adaptation/generation, for example the Bi-dimensional Anisotropic Mesh Generator (BAMG) [159] included in FreeFem ³, the Edge Primitive Insertion and Collapse (EPIC) [160] developed at Boeing and the Refine [160, 161] devel-

³Available at <https://freefem.org>.

oped at NASA ⁴. A more comprehensive list can be found in [162]. For two dimensional mesh adaptations considered in this work, we use the BAMG to perform global re-meshing in each adaptive iteration.

During mesh adaptation, either the desired metric or the metric change with respect to the current mesh-implied metric is specified at vertices of the current background mesh. If the latter approach is used, the current elemental mesh-implied metric is first averaged to vertices before applying the metric changes to get the desired metric at vertices. BAMG takes the desired metric field and the background mesh as input and then outputs a mesh that conforms to the desired metric, to some tolerance. An example of the metric-based global re-meshing is illustrated in Figure 4.5, where BAMG takes in an isotropic background mesh and generates an anisotropic adapted mesh according to the desired metric field.

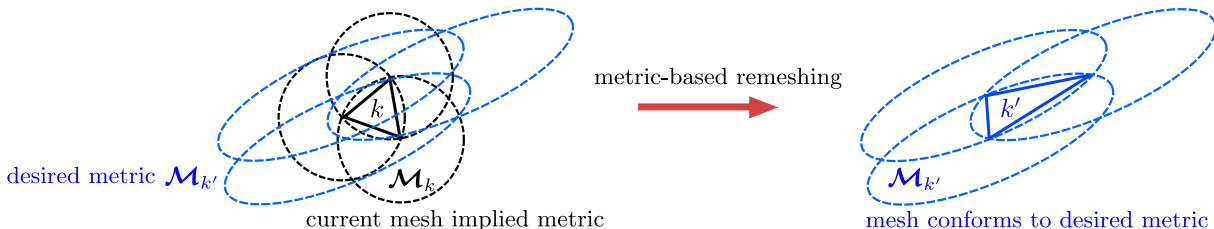


Figure 4.5: An illustration of metric-based global re-meshing in adaptation. The background isotropic mesh element (black) is shown on the left with the corresponding isotropic mesh-implied metric field. With the desired anisotropic metric (blue) specified on the background mesh, BAMG generates a metric-conforming anisotropic mesh, shown on the right.

4.4.3 Hessian-Based Anisotropy Detection

One dominant approach for detecting the anisotropy is to estimate the directional interpolation error of the solution [148, 149, 150, 152, 163, 164, 153, 154], where the mesh is adapted to control a well-defined norm of the interpolation error. More specifically, the choice of the norm has to be made such that the adaptation will not overly refine areas where the solution exhibits discontinuities or singularities. However, this approach is not aware of the output sensitivity with respect to the solution errors, which may be inefficient for convection dominated flow problems as mentioned in Chapter 3. In this work, we use an extension of this approach [97, 151] by incorporating the output adaptive error indicators obtained using Eqn. 4.39.

⁴Available at github.com/NASA/refine.

Suppose we have a linear solution profile, *i.e.*, approximation order $p = 1$, the interpolation error of a scalar solution u over an element edge E with unit tangent vector \vec{t} and length h , can be estimated by

$$\delta u_E \propto \left| \vec{t}^T \mathbf{H} \vec{t} \right| h^2, \quad (4.41)$$

where $\mathbf{H} \in \mathbb{R}^{d \times d}$ is the Hessian matrix of u in d -dimension,

$$H_{i,j} = \frac{\partial u}{\partial x_i \partial x_j}, \quad i, j \in \{1, 2, \dots, d\}. \quad (4.42)$$

The second-order derivatives can be estimated by a quadratic reconstruction of the linear solution. The scalar u used in this work is the Mach number as it has been found to be generally effective [151, 98], although other scalars or more sophisticated combined quantities can also be used [148].

A geometric interpretation of Eqn. 4.41 is that the interpolation error along an edge depends on the squared edge length under the measure of $|\mathbf{H}|$, which is an SPD matrix defined by taking the absolute value of the solution Hessian. Suppose that the mesh is conforming to a metric field \mathcal{M} , assumed constant along the current edge E . Then the edge is of unit length under the metric measure by definition,

$$l_{\mathcal{M}} = \sqrt{\vec{t}^T \mathcal{M} \vec{t}} h = 1. \quad (4.43)$$

Thus the interpolation error can be directly related to the metric field by Eqn. 4.41 and Eqn. 4.43,

$$\delta u_E \propto \frac{\left| \vec{t}^T \mathbf{H} \vec{t} \right|}{\vec{t}^T \mathcal{M} \vec{t}}. \quad (4.44)$$

A general principle in mesh adaptation is to equidistribute errors ⁵, which requires that

$$\frac{\vec{t}^T \mathcal{M} \vec{t}}{\left| \vec{t}^T \mathbf{H} \vec{t} \right|} = k, \quad (4.45)$$

where k is a constant defined by the desired equidistributed interpolation error. In order

⁵On an optimal mesh, the errors should be equidistributed per unit cost, which can be measured by edge length, element area or system DOF. Since otherwise, the mesh can always be modified by adding cost in regions where the error per unit cost is high and reducing cost elsewhere, such that the total error is reduced while the cost is fixed.

for Eqn. 4.45 to hold in any principal direction, \mathcal{M} can be chosen as

$$\mathcal{M}_{\mathbf{H}} = k|\mathbf{H}| = k\mathbf{Q}|\Lambda_{\mathbf{H}}|\mathbf{Q}^T = \mathbf{Q}\Lambda_{\mathcal{M}}\mathbf{Q}^T. \quad (4.46)$$

In Eqn. 4.46, \mathbf{Q} denotes the orthonormal matrix containing the eigenvectors (principal directions) of \mathbf{H} , while $\Lambda_{\mathbf{H}}$ and $\Lambda_{\mathcal{M}}$ are the diagonal matrices containing the eigenvalues of the Hessian matrix \mathbf{H} and the metric tensor \mathcal{M} , respectively. Eqn. 4.46 implies that the relative shape of the desired metric field and the mesh that conforms to the metric,

$$\frac{\lambda_{\mathcal{M},i}}{\lambda_{\mathcal{M},j}} = \left| \frac{\lambda_{\mathbf{H},i}}{\lambda_{\mathbf{H},j}} \right| \implies \frac{h_i}{h_j} = \sqrt{\frac{\lambda_{\mathcal{M},j}}{\lambda_{\mathcal{M},i}}} = \sqrt{\left| \frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}} \right|}, \quad (4.47)$$

where $\lambda_{\mathbf{H}}$ and $\lambda_{\mathcal{M}}$ represent the eigenvalues of \mathbf{H} and \mathcal{M} , respectively, i and j index the principal directions, and h_i and h_j denote the desired element sizes in the corresponding directions.

Anisotropy detection based on the standard Hessian matrix is not suited for higher order approximations, due to the linear solution profile assumption used in the derivation. Instead, the interpolation error is characterized by the $p + 1^{\text{st}}$ derivatives if order p approximation is used for the solution, then the first d largest directional derivatives can be used to determine the principal directions \mathbf{Q} and the corresponding stretching $\Lambda_{\mathbf{H}}$, *i.e.*, the generalized Hessian matrix \mathbf{H} [98, 165]. Consider two principal directions \vec{e}_i and \vec{e}_j from \mathbf{Q} , the error equidistribution yields a mesh stretching as

$$\frac{h_i}{h_j} = \left| \frac{u_{\vec{e}_j}}{u_{\vec{e}_i}} \right|^{1/(p+1)} = \left| \frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}} \right|^{1/(p+1)} = \sqrt{\frac{\lambda_{\mathcal{M},j}}{\lambda_{\mathcal{M},i}}}. \quad (4.48)$$

In Eqn. 4.48, the metric tensor shape on the direction \vec{e}_i is determined by $p+1^{\text{st}}$ derivatives of u along the same direction, $u_{\vec{e}_i}$, whose magnitude is characterized by the i^{th} eigenvalues of the generalized Hessian matrix $\lambda_{\mathbf{H},i}$; similar notations apply to the principal direction j . In this work, however, we only consider using the Hessian matrix of second-order derivatives, regardless of the solution approximation order ⁶.

Although the relative shape of the metric field is determined by the Hessian matrix, the absolute size of the metric, controlled by k in Eqn. 4.46 is still of arbitrary choice. The key idea of Hessian-based mesh adaptation used in this work, which incorporates output error estimation, is to use Eqn. 4.48 to control the mesh stretching (relative mesh

⁶For $p = 2$, the solution Hessian is readily available as a constant matrix, while $p < 2$, the Hessian matrix is estimated by a quadratic reconstruction of the solution, and for $p > 2$, the solution is projected down to $p = 2$ for evaluating second order derivatives.

size) while using the output error indicator in Eqn. 4.39 to determine the absolute mesh element sizes.

4.4.3.1 Element Sizing Using A Priori Rate Estimates

In order to perform mesh adaptation, we need to predict the desired element sizes, or the number of the elements N^f in the adapted (fine) mesh. Let n_e , not necessary an integer, be the number of adapted mesh elements contained in element e for the original mesh. Denoting the current element sizes by h_i^c and the requested element sizes by h_i^f , where i again indexes the principal directions; n_e can be approximated as

$$n_e = \prod_{i=1}^d (h_i^c / h_i^f). \quad (4.49)$$

The current sizes h_i^c are calculated as the singular values of the mapping from a unit equilateral triangle to element e . We assume that on the adapted mesh, the output error is equally distributed in the mesh elements as \mathcal{E}^f . We can then relate the growth in the number of elements to an error reduction factor through an *a priori* estimate,

$$n_e \mathcal{E}^f = \mathcal{E}_e^c \left(\frac{h_{\text{ref}}^f}{h_{\text{ref}}^c} \right)^{\bar{p}_e+1} = \mathcal{E}_e^c \left(\prod_{i=1}^d \frac{h_i^f}{h_i^c} \right)^{(\bar{p}_e+1)/d}, \quad (4.50)$$

where \mathcal{E}_e^c denotes the current error indicator in element e , $\bar{p}_e = \min(p, \gamma_e)$, and γ_e is the lowest order of any singularity within element e . h_{ref} is the reference element size that characterizes the convergence, defined in this work as the geometric mean of the edge lengths along the principal directions. Eqn. 4.49 and Eqn. 4.50 relate n_e to the desired equidistributed elemental error,

$$n_e \mathcal{E}^f = \mathcal{E}_e^c \left(\prod_{i=1}^d \frac{h_i^f}{h_i^c} \right)^{(\bar{p}_e+1)/d} = \mathcal{E}_e^c n_e^{-(\bar{p}_e+1)/d} \implies n_e^{1+(\bar{p}_e+1)/d} = \frac{\mathcal{E}_e^c}{\mathcal{E}^f}. \quad (4.51)$$

Substituting Eqn. 4.51 into $N^f = \sum_e n_e$, we can solve for both \mathcal{E}^f and n_e if N^f is given and \bar{p}_e is assumed equal everywhere in the mesh,

$$\mathcal{E}^f = \left(\frac{\sum_e (\mathcal{E}_e^c)^{\frac{1}{1+(\bar{p}_e+1)/d}}}{N^f} \right)^{1+(\bar{p}_e+1)/d}, \quad n_e = \left(\frac{\mathcal{E}_e^c}{\mathcal{E}^f} \right)^{\frac{1}{1+(\bar{p}_e+1)/d}}. \quad (4.52)$$

After obtaining n_e , the desired element size h_i^f can be calculated using Eqn. 4.49 with current element size information stored in the current mesh-implied metric. In this work, Hessian-based adaptation is made error-based or cost-based depending on how the total number of elements N^f is specified:

- Cost-based Hessian adaptation: a fixed total cost, *i.e.*, number of elements N^f is given. The mesh element sizes and stretching are updated iteratively to match the desired metric field.
- Error-based Hessian adaptation: an error tolerance on the output is given as τ . At each adaptive iteration, the current number of elements N^c is increased by a fixed-growth factor, *i.e.*, $N^f = f^{\text{growth}} N^c$, until the output error estimate is below the tolerance, $\delta J \leq \tau$.

4.4.4 Sampling-Based Anisotropy Detection

Goal-oriented Hessian-based anisotropic mesh adaptation has been shown to successfully detect solution anisotropy in many applications [97, 151, 98]. However, it relies on a scalar solution u , which should be carefully chosen to correlate to the specific output of interest. Also, an inflection in u may lead to inappropriate mesh stretching, and inadequate resolution may occur where the magnitude of the Hessian is close to zero. In order to maximize the approximation potential of a mesh with a given cost, we consider a more sophisticated h -adaptation method: unstructured mesh optimization via error sampling and synthesis (MOESS). In MOESS, the mesh adaptation is formulated as an optimization problem in which the optimal change of the metric field is iteratively determined based on a prescribed metric-cost model and a sampling-inferred metric-error relationship. This method was first developed by Yano and Darmofal for the DG discretization [156, 157], and has been extended to CG [166, 167] and hybridized methods such as hybridized discontinuous Galerkin (HDG) and embedded discontinuous Galerkin (EDG) [168, 169]. We briefly review this method and discuss its modifications in this section.

4.4.4.1 Error Convergence Model

In MOESS, the mesh adaptation is formulated as an optimization problem in which the discrete computational mesh, described by a continuous metric field, is optimized to minimize the output error. During the optimization process, the optimal changes to the current mesh-implied metric, $\mathcal{M}_0(\vec{x})$, is determined iteratively. An affine-invariant [170]

change of the metric can be described by a symmetric local *step matrix*, $\mathbf{S} \in \mathbb{R}^{d \times d}$,

$$\mathbf{M} = \mathbf{M}_0^{1/2} \exp(\mathbf{S}) \mathbf{M}_0^{1/2}. \quad (4.53)$$

where \mathbf{M} and \mathbf{M}_0 denote the modified metric and the current one. Eqn. 4.53 immediately defines a logarithmic map from the current metric to the modified one

$$\mathbf{S} = \log(\mathbf{M}_0^{-1/2} \mathbf{M} \mathbf{M}_0^{-1/2}). \quad (4.54)$$

Now the mesh optimization problem can be stated as searching for the optimal changes of the metric, $\mathbf{S}(\vec{x})$, to minimize the total error,

$$\begin{aligned} \min_{\mathbf{S}} \quad \mathcal{E} &= \sum_{e=1}^{N_e} \mathcal{E}_e(\mathbf{S}) \\ \text{s.t.} \quad C &= \sum_{e=1}^{N_e} C_e(\mathbf{S}) = \text{const}, \end{aligned} \quad (4.55)$$

where \mathcal{E} and C are the total error indicator and the total cost associated with the mesh, the subscript e denote the elemental quantities in element e . The main ingredients of the optimization are the elemental error convergence model $\mathcal{E}_e(\mathbf{S})$ and the elemental cost model $C_e(\mathbf{S})$, which will be discussed in this and the next sections, respectively.

The main problem with heuristic anisotropy detection is that the anisotropy is from the interpolation error while the element sizes are from output error, which may not be compatible. Moreover, the *a priori* rate estimate used in Section 4.4.3.1 assumes a constant, *a priori* and isotropic rate of error convergence, which contradicts the idea of anisotropic adaptation. Therefore, in MOESS, Yano [156] proposed a more generalized error convergence model which allows anisotropic convergence and is built *a posteriori* during the adaptation. Recall the isotropic error convergence used in Eqn. 4.50,

$$\mathcal{E}_e = \mathcal{E}_{e,0} \left(\frac{h}{h_0} \right)^r = \mathcal{E}_{e,0} \exp \left(r \log \left(\frac{h}{h_0} \right) \right), \quad (4.56)$$

where h/h_0 measures the change between current and the new element reference sizes, and r denotes the *a priori* isotropic convergence rate. In the metric-based setting, the element size changes are measured by the local step matrix \mathbf{S}_e . In addition, a symmetric *rate tensor* \mathbf{R}_e is defined to allow anisotropic directional error convergence. A generalized model that dictates the error change due to an elemental step matrix \mathbf{S}_e applied on the

current metric \mathcal{M}_e can then be chosen as

$$\mathcal{E}_e = \mathcal{E}_{e,0} \exp[\text{tr}(\mathcal{R}_e \mathcal{S}_e)] \quad \Rightarrow \quad \frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e} = \mathcal{E}_e \mathcal{R}_e. \quad (4.57)$$

In Eqn. 4.57, the step matrix \mathcal{S}_e plays the role of the size change as $\log(h/h_0)$ while the rate tensor \mathcal{R}_e replaces the scalar isotropic rate r , when compared to Eqn. 4.56. The rate tensor \mathcal{R}_e is difficult to determine *a priori*, and in practice it is inferred through a sampling approach separately for each element, *i.e.*, the anisotropy is determined *a posteriori* through sampling.

4.4.4.2 Local Error Sampling

The generalized anisotropic error model in Eqn. 4.57 relies on the local error convergence rate tensor, \mathcal{R}_e , which has $d(d+1)/2$ unknowns. In order to estimate the rate tensor, Yano [156] proposed a sampling approach, in which several predefined refinement options are sampled to detect the anisotropic error convergence properties. For a triangular element used in this work, four refinement options are considered as shown in Figure 4.6. For each refinement option i , we can determine the corresponding step matrix $\mathcal{S}_{e,i}$ with respect to the original configuration using Eqn. 4.54. In order to estimate the rate tensor, we would like to know how the error changes under each refinement option i , *i.e.*, each step matrix $\mathcal{S}_{e,i}$. One expensive option is to refine the element with the proposed cut, re-solve the primal and fine-space adjoint problems globally, and re-compute the error estimate. Though accurate, this would be impractically expensive. Another option is to only solve the primal/adjoint problems on a subset of the original mesh: the current element and its neighbors. This approach, taken in the original MOESS algorithm, is less accurate but still performs very well as globally-exact primal/adjoint states are not necessary to estimate the error rate tensor. In this work we further simplify the estimation by not solving additional primal/adjoint problems, even on a local patch of elements. Instead, we use an element-local projection method [171] to approximate the fine-space adjoint in semi-refined spaces associated with each refinement option.

Consider one element Ω_e in the current mesh, shown as the original configuration in Figure 4.6. The fine space adjoint, $\Psi_{h,e}$, provides an estimate of the output error in the current order p solution, as measured relative to the $p+1$ solution, denoted as $\mathcal{E}_{e,0}$. Now, suppose that we are looking at refinement option i in Figure 4.6: this local refinement creates a solution space that is finer than the original one, which is called a semi-refined space, although we assume that the semi-refined spaces considered are not as fine as the order $p+1$ space. If we have an order p adjoint on this semi-refined space, $\Psi_{H,i}$, where

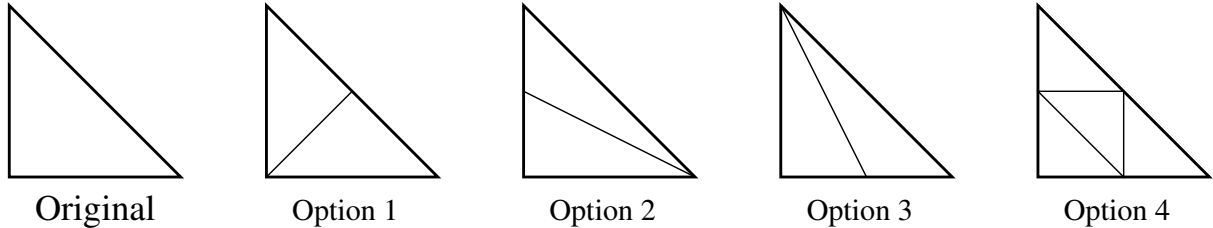


Figure 4.6: Four refinement options for a triangle. Each one is considered implicitly during error sampling, though the elements are never actually refined.

H is the order p space and i indicates that we are considering refinement option i , we can compute an error indicator $\Delta\mathcal{E}_{e,i}$, which estimates the error between the original coarse space and the i^{th} semi-refined space. The remaining error associated with refinement option i is then given by the difference,

$$\mathcal{E}_{e,i} \equiv \mathcal{E}_{e,0} - \Delta\mathcal{E}_{e,i}. \quad (4.58)$$

In other words, we treat the order $p + 1$ space as the “true space”, $\Delta\mathcal{E}_{e,i}$ is the error that is not covered on the semi-refined space when measured relative to the “true space”.

Calculating $\Delta\mathcal{E}_{e,i}$ requires an adjoint-weighted residual evaluation on the element refined under option i . To avoid actually refine the element, we project $\Psi_{H,i}$ back into the $p + 1$ space on the original element and evaluate the adjoint weighted residual there ⁷. The error estimate can be written as

$$\Delta\mathcal{E}_{e,i} \equiv (\Psi_{h,e}^{H,i})^T \mathbf{R}_{h,e}(\mathbf{U}_h^H), \quad (4.59)$$

where $\Psi_h^{H,i}$ is the order p adjoint solution $\Psi_{H,i}$, projected from the i^{th} semi-refined space into order $p + 1$ space on the original configuration. Finally, instead of solving the adjoint on the semi-refined space, as we never refine the element anyway, we project the order $p + 1$ adjoint solution on the original configuration to order p under the refinement option i as a surrogate of $\Psi_{H,i}$. This projection causes information loss as the semi-refined space is not as rich as the “true space” (order $p + 1$ space on the original configuration), while it gives the best possible adjoint (closest to the adjoint on the “true space”) on the semi-refined space and produces the uncovered error $\Delta\mathcal{E}_{e,i}$.

In summary, the error uncovered by refinement option i , $\Delta\mathcal{E}_{e,i}$, is estimated by the adjoint-weighted residual in Eqn. 4.59, with all calculations occurring at order $p + 1$ on the original element. Using least-squares projections in reference space, the combination

⁷Again, we assume the order $p + 1$ space on the original configuration is finer/richer than the semi-refined order p space, so the projection is loss-less and is able to recover the error estimate.

of projections can be encapsulated into one transfer matrix that converts Ψ_h into $\Psi_h^{H,i}$, both represented in the order $p + 1$ space on the original element:

$$\begin{aligned}\Psi_h^{H,i} &= \mathbf{T}_i \Psi_h, \\ \mathbf{T}_i &= [\mathbf{M}_0(\phi_0^{p+1}, \phi_0^{p+1})]^{-1} \sum_{k=1}^{n_i} \mathbf{T}_{ik}, \\ \mathbf{T}_{ik} &= \mathbf{M}_k(\phi_0^{p+1}, \phi_k^p) [\mathbf{M}_k(\phi_k^p, \phi_k^p)]^{-1} \mathbf{M}_k(\phi_k^p, \phi_k^{p+1}) [\mathbf{M}_k(\phi_k^{p+1}, \phi_k^{p+1})]^{-1} \mathbf{M}_k(\phi_k^{p+1}, \phi_0^{p+1}).\end{aligned}\tag{4.60}$$

In these equations, n_i is the number of sub-elements in refinement option i , k is an index over these sub-elements, ϕ_k^p, ϕ_k^{p+1} are order p and $p + 1$ basis functions on sub-element k , ϕ_0^p, ϕ_0^{p+1} are order p and $p + 1$ basis functions on the original element, and components of the mass-like matrices are defined as

$$\mathbf{M}_k(\phi_l, \phi_m) = \int_{\Omega_k} \phi_l \phi_m d\Omega, \quad \mathbf{M}_0(\phi_l, \phi_m) = \int_{\Omega_0} \phi_l \phi_m d\Omega,\tag{4.61}$$

where Ω_k is sub-element k and Ω_0 is the original element. Note that the transfer matrix \mathbf{T}_i can be calculated for each refinement option i once in reference space and then used for all elements, so that the calculation of $\Delta \mathcal{E}_{e,i}$ consumes minimal additional cost – and most importantly, no solves or residual evaluations are needed on the refined element, as these generally require cumbersome data management and transfer.

After calculating the element error indicators, $\mathcal{E}_{e,0}$, and the errors remaining after each refinement option i , $\mathcal{E}_{e,i}$, according to Eqn. 4.58, we perform a regression to determine the local rate tensor, \mathcal{R}_e . Note that we have 4 refinement options and only 3 independent entries in the symmetric \mathcal{R}_e tensor. Using Eqn. 4.57, we formulate the regression task as a least-square problem which minimizes the following error, summed over refinement options,

$$\mathcal{R}_e = \arg \min_{\mathcal{Q}} \sum_i \left[\log \frac{\mathcal{E}_{e,i}}{\mathcal{E}_{e,0}} - \text{tr}(\mathcal{Q} \mathcal{S}_{e,i}) \right]^2.\tag{4.62}$$

In this equation, $\mathcal{S}_{e,i}$ is the step matrix associated with refinement option i , given by (from Eqn. 4.54),

$$\mathcal{S}_{e,i} = \log \left(\mathcal{M}_0^{-1/2} \mathcal{M}_i \mathcal{M}_0^{-1/2} \right),\tag{4.63}$$

where \mathcal{M}_i is the affine-invariant metric average [170] of the mesh-implied metrics of all sub-elements in refinement option i . Differentiating Eqn. 4.62 with respect to the independent components of \mathcal{R}_e yields a linear system for these components, *i.e.*, the normal equations for the least-square problem.

4.4.4.3 Cost Model

To measure the cost of refinement, we use degrees of freedom (DOF), which on each element just depends on the approximation order p_e , assumed in this work as constant and equal to p over the elements. The formula for calculating the cost associated with element Ω_e is given in Section 2.4.3. Consider a proposed metric step matrix \mathcal{S}_e applied on current metric $\mathcal{M}_{e,0}$, the cost allocation for the new configuration is inversely proportional to the element area, which can be inferred from the metric change as,

$$\begin{aligned} \mathcal{C}_e &= \mathcal{C}_{e,0} \sqrt{\frac{\det(\mathcal{M}_e)}{\det(\mathcal{M}_{e,0})}} = \mathcal{C}_{e,0} \sqrt{\frac{\det(\mathcal{M}_{e,0}^{1/2} \exp(\mathcal{S}_e) \mathcal{M}_{e,0}^{1/2})}{\det(\mathcal{M}_{e,0})}} \\ &= \mathcal{C}_{e,0} \sqrt{\det(\exp(\mathcal{S}_e))} = \mathcal{C}_{e,0} \exp\left[\frac{1}{2}\text{tr}(\mathcal{S}_e)\right] \\ \Rightarrow \quad \frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e} &= \mathcal{C}_e \frac{1}{2} \mathcal{I}, \end{aligned} \tag{4.64}$$

where \mathcal{C}_e is the expected cost over the original element area after applying \mathcal{S}_e to the original metric $\mathcal{M}_{e,0}$, and \mathcal{I} is the identity matrix. The total cost over the mesh is the sum of the elemental costs over the mesh, $\mathcal{C} = \sum_{e=1}^{N_e} \mathcal{C}_e$.

4.4.4.4 Mesh Optimization Algorithm

Given a current mesh with its mesh-implied metric ($\mathcal{M}_0(\vec{x})$), elemental error indicators $\mathcal{E}_{e,0}$, and elemental rate tensor estimates, \mathcal{R}_e , the mesh optimization problem is to determine the optimal step matrix field, $\mathcal{S}(\vec{x})$, that minimizes the error at a fixed cost,

$$\begin{aligned} \min_{\mathcal{S}_v} \quad \mathcal{E} &= \sum_{i=1}^{N_e} \mathcal{E}_e(\mathcal{S}_v) \\ \text{s.t.} \quad \mathcal{C} &= \sum_{i=1}^{N_e} \mathcal{C}_e(\mathcal{S}_v) = \text{const.} \end{aligned} \tag{4.65}$$

where the step matrix field is parametrized at mesh vertices as \mathcal{S}_v ⁸, which are arithmetically-averaged to adjacent elements⁹:

$$\mathcal{S}_e = \frac{1}{|V_e|} \sum_{v \in V_e} \mathcal{S}_v, \tag{4.66}$$

⁸A node-based step matrix is used since BAMG accepts a node-based metric \mathcal{M}_v as input.

⁹There is no need for an affine-invariant average because entries of \mathcal{S} are coordinate system independent.

where V_e is the set of vertices ($|V_e|$ is the number of them) adjacent to element e .

The first order optimality condition requires

$$\frac{\partial \mathcal{E}}{\partial \mathbf{S}_v} + \lambda_v \frac{\partial \mathcal{C}}{\partial \mathbf{S}_v} = \mathbf{0}, \quad (4.67)$$

where λ_v is the Lagrange multiplier associated with the fixed cost constraint. The error and cost derivatives in Eqn. 4.67 can be obtained by applying chain rules on Eqn. 4.57 and Eqn. 4.64 with Eqn. 4.66,

$$\frac{\partial \mathcal{E}}{\partial \mathbf{S}_v} = \sum_{e \in E_v} \frac{\partial \mathcal{E}_e}{\partial \mathbf{S}_e} \frac{\partial \mathbf{S}_e}{\partial \mathbf{S}_v}, \quad \frac{\partial \mathcal{C}}{\partial \mathbf{S}_v} = \sum_{e \in E_v} \frac{\partial \mathcal{C}_e}{\partial \mathbf{S}_e} \frac{\partial \mathbf{S}_e}{\partial \mathbf{S}_v} \quad (4.68)$$

where E_v is the set of elements sharing the vertex v . We further notice that the cost only depends on the *trace* of the step matrix, in other words, the trace-free part of \mathbf{S}_e only stretches the element but does not alter the area. On the other hand, both the trace and trace-free part of \mathbf{S}_e affect the error due to anisotropic convergence. Thus the step matrix can be divided into trace and trace-free parts as

$$\mathbf{S}_v = \underbrace{s_v \mathbf{I}}_{\text{trace}} + \underbrace{\tilde{\mathbf{S}}_v}_{\text{trace-free}}, \quad s_v = \frac{\text{tr}(\mathbf{S}_v)}{d}. \quad (4.69)$$

Then the derivatives shown in Eqn. 4.68 can be broken down into

$$\frac{\partial \mathcal{E}}{\partial s_v} = \text{tr} \left(\frac{\partial \mathcal{E}}{\partial \mathbf{S}_v} \right), \quad \frac{\partial \mathcal{E}}{\partial \tilde{\mathbf{S}}_v} = \frac{\partial \mathcal{E}}{\partial \mathbf{S}_v} - \frac{\partial \mathcal{E}}{\partial s_v} \frac{\mathbf{I}}{d}; \quad (4.70)$$

$$\frac{\partial \mathcal{C}}{\partial s_v} = \frac{\partial \mathcal{C}}{\partial \mathbf{S}_v}, \quad \frac{\partial \mathcal{C}}{\partial \tilde{\mathbf{S}}_v} = \mathbf{0}. \quad (4.71)$$

Meanwhile, the optimality condition in Eqn. 4.67 can be rewritten accordingly

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial s_v} + \lambda_v \frac{\partial \mathcal{C}}{\partial s_v} &= 0, \\ \frac{\partial \mathcal{E}}{\partial \tilde{\mathbf{S}}_v} &= \mathbf{0}. \end{aligned} \quad (4.72)$$

We do not solve above system exactly, since it would be a very high-dimensional problem which may require extremely high computational effort, especially in an optimization problem where the optimal mesh changes as the design varies. Furthermore, the error model based on the empirical local sampling may not represent the error exactly. Therefore, solving the mesh optimization problem exactly is inefficient and unnecessary. Instead,

we follow the approach proposed by Yano [156] where the optimization is done locally, which converges to a solution of the original optimization problem. The optimization algorithm is summarized as follows:

1. Given a mesh, solution, and adjoint, calculate $\mathcal{E}_e, \mathcal{C}_e, \mathcal{R}_e$ for each element e .
2. Set $\delta s = \delta s_{\max}/n_{\text{step}}, \mathcal{S}_v = 0$.
3. Begin loop: $i = 1 \dots n_{\text{step}}$
 - (a) Calculate \mathcal{S}_e from (4.66), $\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$ from (4.57), and $\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}$ from (4.64).
 - (b) Calculate derivatives of \mathcal{E} and \mathcal{C} with respect to s_v and $\tilde{\mathcal{S}}_v$.
 - (c) At each vertex form the ratio $\lambda_v = \frac{\partial \mathcal{E}/\partial s_v}{\partial \mathcal{C}/\partial s_v}$ and
 - Refine the metric for 30% of the vertices with the largest $|\lambda_v|$: $\mathcal{S}_v = \mathcal{S}_v + \delta s \mathcal{I}$
 - Coarsen the metric for 30% of the vertices with the smallest $|\lambda_v|$: $\mathcal{S}_v = \mathcal{S}_v - \delta s \mathcal{I}$
 - (d) Update the trace-free part of S_v to enforce stationarity with respect to shape changes at fixed area: $\mathcal{S}_v = \mathcal{S}_v - \delta s (\partial \mathcal{E}/\partial \tilde{\mathcal{S}}_v)/(|\partial \mathcal{E}/\partial s_v|)$.
 - (e) Rescale $\mathcal{S}_v \rightarrow \mathcal{S}_v + \beta \mathcal{I}$, where β is a global constant calculated from (4.64) to constrain the total cost to the desired DOF value: $\beta = \frac{2}{d} \log \frac{\mathcal{C}_{\text{target}}}{\mathcal{C}}$, where $\mathcal{C}_{\text{target}}$ is the target cost.

Note, λ_v is a Lagrange multiplier in the optimization, which can be interpreted as the ratio of the marginal error to marginal cost of a step matrix trace increase (*i.e.*, mesh refinement). The above algorithm iteratively equidistributes λ_v globally so that, at optimum, all elements have the same marginal error to cost ratio. Constant values that work generally well in the above algorithm are $n_{\text{step}} = 20$ and $\delta s_{\max} = 2 \log 2$. In practice, the mesh optimization and flow/adjoint solution are performed several times at a given target cost, $\mathcal{C}_{\text{target}}$, until the error stops changing. Then the target cost is increased to reduce the error further if desired.

CHAPTER 5

Aerodynamic Optimization Framework with Adaptive CFD

As mentioned earlier in Chapter 1, the optimization framework relies on an effective integration of several components, including the geometry parameterization, mesh deformation, numerical optimization algorithm, CFD flow solver, sensitivity analysis and also in this work the error estimation and mesh adaptation component. The last three components have been discussed in detail in Chapter 2–4, and the remaining parts will be covered in the current chapter. We first introduce in Section 5.1 the problem of interest: two-dimensional airfoil shape optimization, in which the geometry parameterization method is also presented. Section 5.2 focuses on the mesh deformation techniques used in this work. Section 5.3 starts with a description of the gradient-based optimizer adopted, followed by which detailed discussions are given on the effective integration of the traditional gradient-based optimization with an adaptive CFD framework.

5.1 Two-Dimensional Airfoil Optimization

5.1.1 Problem Statement

For demonstration, two-dimensional airfoil shape optimizations are considered in this work. In particular, the problem studied here is to search for an optimal design, including the airfoil shape and the angle of attack to minimize the drag subject to a fixed lift constraint and a minimum airfoil area. The optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & c_d(\mathbf{U}, \mathbf{x}) \\ \text{subject to:} \quad & R^e = c_\ell(\mathbf{U}, \mathbf{x}) - \bar{c}_\ell = 0, \\ & R^{ie} = A(\mathbf{x}) - A_{\min} \geq 0, \end{aligned} \tag{5.1}$$

where c_d is the optimization objective, c_ℓ and \bar{c}_ℓ denote the current and the target lift coefficients, A and A_{\min} represent the current and minimum allowable areas of the airfoil; \mathbf{x} is the design parameter vector and \mathbf{U} is the conservative state vector of the governing PDEs. The objective output, *i.e.*, c_d , and the constraint output, *i.e.*, c_ℓ , depend on both the design \mathbf{x} and the flow states \mathbf{U} , while the airfoil area is solely determined by the design parameters \mathbf{x} . Following the notation in Chapter 4, we denote the drag coefficient c_d as the adapt output and the lift coefficient c_ℓ as the trim output. The inequality volume constraint, independent of the flow states, is assumed to be measured exactly and handled by the optimizer as normal inequality constraints.

The design variables considered are the angle of attack α , and the airfoil shape parameterized by a discrete shape vector \mathbf{x}_s , $\mathbf{x} = [\alpha, \mathbf{x}_s]$. When the trim condition (fixed-lift constraint) is enforced by the optimizer, *i.e.*, in the OBT approach, the whole design vector is updated through the optimizer during the optimization; On the other hand, if SBT is used, the angle of attack α serves as a trim variable \mathbf{x}_t to satisfy the trim condition and only the active design \mathbf{x}_s is available for the optimizer in the optimization.

5.1.2 Geometry Parametrization

The airfoil shape is parameterized using the deformation method proposed by Hicks and Henne [28], in which the deformation of airfoil instead of the geometry itself is parameterized. The geometry change is parameterized by a set of “bump” functions, also referred to as Hicks-Henne basis functions, such that a new airfoil shape is defined by adding a linear combination of the Hicks-Henne basis functions to its original baseline shape,

$$z(x) = z_{\text{base}}(x) + \sum_{i=1}^{N_s} a_i \phi_i(x), \quad (5.2)$$

where x denotes the position along the airfoil chord, and z and z_{base} represent the upper or lower surface vertical coordinates of the new airfoil and the original one. The upper and lower surfaces are considered separately in Eqn. 5.2, *i.e.*, independent design variables are used for the upper and the lower surfaces. The summation term in Eqn. 5.2 is a linear combination of the Hicks-Henne basis $\phi_i(x)$ with different coefficients a_i , which constitute the design parameters along with the angle of attack, $\mathbf{x} = [\alpha, a_1, a_2, \dots, a_{N_s}]^T$. One widely used basis set is the sine functions [52],

$$\phi_i(x) = \sin^{t_i}(\pi x^{m_i}), \quad m_i = \frac{\ln(0.5)}{\ln(x_{M_i})}, \quad (5.3)$$

where x_{M_i} is the predefined location of the maxima for each basis function and t_i controls the width of the bump function. We adopt an optimized basis set [172] where

$$t_i = 4, \quad x_{M_i} = \frac{1}{2} \left[1 - \cos \left(\frac{i\pi}{n+1} \right) \right], \quad i = 1, 2, \dots, n. \quad (5.4)$$

5.1.3 Angle of Attack Handling

In most flow simulations, the angle of attack comes into the system as a parameter determining the boundary condition, more specifically the free-stream flow angles. In the present work, however, we will focus on possibly separate angles of attack for multiple components. To allow relative angle changes between different components, the angle of attack variations are implemented through a mesh motion strategy based on the Arbitrary Lagrangian Eulerian (ALE) formulation [173, 174]. The idea of ALE is to map the original governing PDE on a deforming physical domain to a modified PDE on a static reference domain. In other words, the original problem on the physical domain involves relative mesh motions/deformations, which can be viewed as a modified PDE on the undeformed reference domain. Although the mesh motion can also be achieved by standard mesh deformation techniques which will be covered in Section 5.2, the ALE framework provides a relatively easier treatment for boundary conditions and is more flexible when multiple geometry components are involved. However, the ALE method requires an analytically defined mapping between the reference and physical domains, thus analytical mesh motions/deformations, such as flow angle variations, are often preferred. On the other hand, the airfoil geometry modification, which is more local and nonlinear, is still managed by standard mesh deformation techniques in the current work.

In the ALE formulation, the angle of attack is applied as a parameter, and the sensitivity can be obtained using standard adjoint method as mentioned in Chapter 3. Depending on how the trimming condition is enforced, the sensitivities are used by the optimizer or the flow solver to satisfy the trimming equations.

5.2 Mesh Deformation

At each iteration in the optimization, the objective output and the constraint outputs need to be re-evaluated on the updated geometry, which in turn requires an updated computational mesh that conforms to the geometry. Regeneration of a mesh especially for a complex geometry or with high resolution, could be time-consuming and non-trivial ¹.

¹Although in a mesh adaptation setting, the adapted meshes are obtained from global re-meshing, a valid mesh is still required on the updated geometry first, such that the adaptation itself is much cheaper

And more importantly, the mesh topology or structure changes may be inconsistent with the sensitivity obtained on the original mesh, see Section 5.3.3 for details. Therefore, an efficient way to update the computational mesh is needed. In this work, two mesh deformation techniques are used to propagate the airfoil boundary deformation into interior mesh nodes: one is based on an explicit inverse distance weighting (IDW) interpolation scheme [175], and the other is based on radial basis function (RBF) interpolation [176, 177]. The former approach is explicit and easy to implement, which is used for inviscid flow meshes in this work; while the latter approach often results in a higher-quality mesh, at higher cost, and is applied in viscous simulations such that the near-wall mesh quality is better-preserved.

5.2.1 Inverse Distance Weighting Interpolation

The deformation of the computational mesh can be viewed as a projection of deformations and rotations from the boundary surface into the interior mesh nodes. Assume mesh displacement fields are prescribed separately at each boundary node, and that the effects on the interior nodes depend inversely on the distance to the corresponding boundary node. Then at each interior mesh node, the displacement can be estimated by summing over the effects of displacement fields prescribed at each boundary node.

For general mesh deformations, we can decompose the displacement into rotation and translation. Once the displacement on a given boundary node is determined, we can estimate the prescribed displacement field as

$$\vec{d}_i(\vec{x}) = \mathbf{M}_i\vec{x} + \vec{t}_i - \vec{x}, \quad (5.5)$$

where \mathbf{M}_i is the rotation matrix, \vec{t}_i is the translation vector associated with the i^{th} boundary node, \vec{x} is the coordinate vector of an arbitrary node in the original mesh. The rotation matrix is determined by the local normal vector rotation. In practice, it is estimated by finding the closest rotation matrix that best matches the rotation of adjacent edges on the boundary nodes. Once the rotation matrix is determined, the translation field \vec{t}_i can be easily computed given the displacement at the i^{th} boundary node.

The displacement field on any interior mesh nodes can then be approximated through a weighted average of all boundary node displacement fields,

$$\vec{d}(\vec{x}) = \frac{\sum_{i=1}^{N_b} w_i(\vec{x})\vec{d}_i(\vec{x})}{\sum_{i=1}^{N_b} w_i(\vec{x})}, \quad (5.6)$$

than generating a high-quality mesh.

where N_b is the number of the boundary nodes and w_i is the weight function, which depends inversely on the distance to the i^{th} boundary node. We adopt the two-exponent form of the weighting function proposed by Luke *et al.* [175], which preserves the near-boundary deformations while providing a smooth transition in the interpolation. The weight also includes the areas of adjacent faces such that mesh refinements of a specific region will not increase its influence in the interpolation. The resulting weighting function can be written as

$$w_i(\vec{x}) = A_i * \left[\left(\frac{L_{\text{def}}}{\|\vec{x} - \vec{x}_i\|} \right)^a + \left(\frac{\alpha L_{\text{def}}}{\vec{x} - \vec{x}_i} \right)^b \right], \quad (5.7)$$

where A_i is the edge length weight or the face area weight assigned to node i , L_{def} is the estimated length of the deformation region, α defines the size of the near body influence region as a fraction of L_{def} , and a and b are user-defined exponents controlling the decay of the weights. We take the values suggested in [175] that $a = 3$ and $b = 5$, and L_{def} as the maximum distance from any mesh node to the origin. The parameter α controls the relative effects of nearby boundary nodes compared to more distant ones. In general, this factor should increase if the surface displacement variation on different nodes is large. Thus we can determine α by estimating the deviation of the boundary node displacement from the mean boundary displacement \vec{d}_{mean} ,

$$\alpha = \frac{\eta}{L_{\text{def}}} \max_{i=1}^{N_b} \|\vec{d}_i(\vec{x}_i) - \vec{d}_{\text{mean}}\|, \quad (5.8)$$

where η is a scaling factor which takes a value of $\eta = 5$ as suggested by numerical experiments [175], \vec{d}_{mean} is the mean boundary displacement defined as

$$\vec{d}_{\text{mean}} = \sum_{i=1}^{N_b} a_i \vec{d}_i(\vec{x}_i), \quad a_i = \frac{A_i}{\sum_{j=1}^{N_b} A_j}. \quad (5.9)$$

In practice, α is limited to be greater or equal to 0.1 to guarantee rigid deformation near the boundary for better viscous mesh quality. The current implementation closely follows the method presented in [175] and the IDWarp² package.

5.2.2 Radial Basis Function Interpolation

A radial basis function (RBF) is a real valued function that only depends on the distance from the origin, or some specified center points, \vec{c} , such that $\phi(\vec{x}) = \phi(\|\vec{x}\|)$ or $\phi(\vec{x}) = \phi(\|\vec{x} - \vec{c}\|)$. Assume that we are given the displacements $\mathbf{d}_b = [\vec{d}_{b,1}, \vec{d}_{b,2}, \dots, \vec{d}_{b,N_b}]$ of

²Available at <https://github.com/mdolab/idwarp>.

the N_b airfoil boundary nodes $\mathbf{x}_b = [\vec{x}_{b,1}, \vec{x}_{b,2}, \dots, \vec{x}_{b,N_b}]$, which are the center points in the RBF interpolation. Then we can use the sum of the RBFs and a polynomial to interpolate the original displacement function,

$$\vec{d}(\vec{x}) = \sum_{i=1}^{N_b} \vec{r}_i \phi_i(\|\vec{x} - \vec{x}_{b,i}\|) + \vec{p}(\vec{x}), \quad (5.10)$$

where \vec{r}_i is the coefficient for the i^{th} RBF, and $\vec{p}(\vec{x})$ is an order p polynomial. The coefficients \vec{r}_i and the polynomial \vec{p} are determined by the interpolation conditions

$$\vec{d}(\vec{x}_{b,i}) = \vec{d}_{b,i} \quad \forall \vec{x}_{b,i} \in \mathbf{x}_b, \quad (5.11)$$

and the additional requirements to recover any polynomial $q(\vec{x})$ with an order less than or equal to order p ,

$$\sum_{i=1}^{N_b} \vec{r}_i q(\vec{x}_{b,i}) = \mathbf{0}. \quad (5.12)$$

Eqn. 5.11 and Eqn. 5.12 lead to a linear system whose size is $\mathcal{O}(N_b)$, which is small compared to the number of nodes in the entire mesh. Moreover, the connectivity of the mesh is not required for the RBF interpolation. By constructing the interpolation, we can easily propagate the mesh deformation from the boundary to any interior mesh nodes through Eqn. 5.10. More implementation details can be found in the previous work of Jakobsson and Amoignon [176] and de Boer *et al.* [177].

5.3 Optimization Algorithm

5.3.1 Gradient-Based Optimizer

Although the optimization problem is formulated in an augmented Lagrangian form in Chapter 4, any gradient-based optimization algorithm can be used if the trimming condition is handled by the flow solver, since the Lagrange multiplier associated with the trimming constraints, *i.e.*, the coupled adjoint $\boldsymbol{\mu}$, are obtained during the trimming process. However, if the trimming constraints are handled by the optimizer, then an optimization algorithm that involves Lagrange multipliers should be used in order to provide them for error estimation and mesh adaptation purposes [67].

In this work, we use the Sequential Least Squares Programming (SLSQP) [178, 179] algorithm which is originally written in Fortran as a module for optimal control calcula-

tions ³. It can be used to solve general nonlinear programming (NLP) problems,

$$\begin{aligned}
\mathbf{NLP}: \quad & \min_{\mathbf{x}} f(\mathbf{x}) \\
\text{subject to:} \quad & g_j(\mathbf{x}) = 0, \quad j = 1, \dots, m_e \\
& g_j(\mathbf{x}) \geq 0, \quad j = m_e + 1, \dots, m, \\
& \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u.
\end{aligned} \tag{5.13}$$

We follow the notation that is generally used by the optimization community in Eqn. 5.13, where \mathbf{x} stands for the design variables with lower and upper bounds denoted as \mathbf{x}_l and \mathbf{x}_u , $f(\mathbf{x})$ and $g(\mathbf{x})$ are the objective and constraints respectively, m_e and m denote the dimension of equality constraints and the total number of the constraints. One of the most efficient methods of solving the NLP problem is the sequential quadratic programming (SQP), in which the original problem is sequentially approximated as a quadratic programming (QP) problem. First, we write the original minimization problem in a Lagrangian form,

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) = f(x) + \sum_{j=1}^m \mu_j g_j(\mathbf{x}), \tag{5.14}$$

where $\boldsymbol{\mu} \in \mathbb{R}^m$ is the Lagrange multipliers whose first m_e entries are associated with the equality constraints while the rest of them are for the inequality constraints. The SQP method sequentially minimize a QP subproblem which uses a quadratic approximation of the Lagrangian function while a linear approximation of the constraints,

$$\begin{aligned}
\mathbf{QP}: \quad & \min_{\mathbf{d}} f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k) \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{B}^k \mathbf{d}, \\
\text{subject to:} \quad & \nabla g_j(\mathbf{x}^k) \mathbf{d} + g_j(\mathbf{x}^k) = 0, \quad j = 1, \dots, m_e, \\
& \nabla g_j(\mathbf{x}^k) \mathbf{d} + g_j(\mathbf{x}^k) \geq 0, \quad j = m_e + 1, \dots, m.
\end{aligned} \tag{5.15}$$

where the \mathbf{B} matrix is the Hessian of the Lagrangian function, $\mathbf{B} \equiv \nabla_{xx}^2 \mathcal{L}(\mathbf{x}, \boldsymbol{\mu})$.

Instead of directly approximating the Hessian matrix \mathbf{B} by taking second-order derivatives, SLSQP adopts a quasi-Newton type method where the \mathbf{B} matrix is approximated using only first-order information. Particularly, the Broyden-Fletcher-Goldfarb-Shanno (BFGS)-type formula is used for updating the \mathbf{B} matrix sequentially along with the QP problem. In the implementation, the QP subproblem is replaced by a least squares with equality and inequality constraints (LSEI) problem, using the Cholesky decomposition of

³Available as the original Fortran code at <http://www.netlib.org/toms/733> or wrapped in the optimization module in the SciPy library at <https://www.scipy.org>.

the Hessian, $\mathbf{B} = \mathbf{LDL}^T$,

$$\begin{aligned} \text{LSEI: } \quad & \min_{\mathbf{d}} \|(\mathbf{D}^k)^{1/2}(\mathbf{L}^k)^T \mathbf{d} + (\mathbf{D}^k)^{-1/2}(\mathbf{L}^k)^{-1} \nabla f(\mathbf{x}^k)\| \\ \text{subject to: } \quad & \nabla g_j(\mathbf{x}^k) \mathbf{d} + g_j(\mathbf{x}^k) = 0, \quad j = 1, \dots, m_e, \\ & \nabla g_j(\mathbf{x}^k) \mathbf{d} + g_j(\mathbf{x}^k) \geq 0, \quad j = m_e + 1, \dots, m; \end{aligned} \quad (5.16)$$

where the superscript k denotes the iteration number. The above least squares problem is solved by a slightly modified version of the least squares solver of Lawson and Hanson [180]. When the solution is far from optimal, the LSEI problem only gives a descent direction, which should be incorporated with a line search algorithm to ensure sufficient decrease of the objective function in each QP problem. A back-tracking line search, which originates from Brent's localmin algorithm [181], is used.

Due to the equivalence of solving the QP subproblem in Eqn. 5.15 and the LSEI subproblem in Eqn. 5.16, SLSQP is sometimes interpreted as Sequential Least Squares Quadratic Programming. In our implementation, the original SLSQP Fortran code is wrapped in C functions, and if the trim constraints are handled by the optimizer, the coupled adjoint $\boldsymbol{\mu}$ is directly extracted from the optimizer for error estimation and mesh adaptation use.

5.3.2 Incorporation with Output Error Estimation and Mesh Adaptation

In traditional aerodynamic optimization, a single fixed mesh is often used. Thus if high accuracy is required, a fine discretization, such as a high order discretization or a fine computational mesh or both, has to be chosen. As mentioned in Chapter 1, this high-fidelity requirement poses several challenges in the optimization. First of all, generating a high-fidelity computational mesh is by no means a trivial task, especially for complex geometries or unconventional configurations in which limited experience or knowledge is available. The overhead added by mesh generation will significantly slow down the optimization setup and can thus be the bottleneck of the design automation. In addition, the starting mesh is generated based on the initial design, with no guarantee to remain adequate throughout the entire optimization process. Secondly, the quality of the design and the numerical errors are typically only investigated via mesh convergence study for the initial and the final designs, before and after the optimization, which can potentially lead to inaccurate or spurious optima during the design process. Finally, the computational burden associated with high-fidelity meshes often makes high-dimensional optimization problems computationally taxing. In this section, we will tackle these problems by incorporating the output error estimation and mesh adaptation within aerodynamic optimiza-

tion problems to improve the automaticity, reliability and efficiency of the optimization.

In order to aid practical aerodynamic design and to reduce the optimization cost, automated adapted meshes are introduced into the design process. The designer only needs to provide a relatively coarse background mesh, which is much easier to generate, to start the optimization run. Then, the computational mesh is adapted in necessary regions based on the output error estimates, with active control of numerical errors during the optimization. Output error estimates in this case not only guides the mesh adaptation, but also provides a way to quantify the uncertainty of the design due to numerical errors, *i.e.*, assess the quality of the design on the fly rather than after the optimization. The sensitivity calculation requires the output adjoint on the current mesh (coarse space), while the error estimation needs the adjoint solution on the enriched space ($p + 1$ fine space). However, the coarse space adjoint is often injected to the fine space to provide a good initial guess for the fine space adjoint solution. Therefore, the error estimation can reuse the adjoint solution obtained in the sensitivity analysis, making the incorporation more efficient.

A multifidelity optimization framework is built into the proposed method, taking advantage of the variable fidelity offered by adaptive meshes. Although different levels of *a priori* meshes can also be used to enable multifidelity optimization, the mesh generation and optimization setup in such a scenario rely more on users' experience to exploit the benefits of various fidelity levels. Despite the difficulties in generating good meshes for each fidelity, the uncertainty due to discretization errors is not quantified at each fidelity in traditional optimization. As a result, the optimization tolerance in each fidelity is either always set to be the ultimate optimization tolerance, or varied purely based on users' experience or intuition, which may degenerate the potential benefits of variable fidelity. For example, on the one hand, if the optimization tolerance at low fidelities is set to be too high, the optimizer will not be able to fully exploit the current fidelity to improve the design. In this case, the current mesh is unnecessarily fine for the prescribed optimization tolerance, which we will call "over-refinement" in this work. On the other hand, however, if the optimization tolerance is set too tightly at low fidelities, the optimizer may work on numerical errors instead of physics to improve the design, *i.e.*, the optimizer is over-optimizing with inaccurate information. In consequence, the optimization on the high fidelity has to do extra work for undoing these incorrect design modifications to bring the design back to physically optimal. By incorporating the output error estimation and mesh adaptation, a better balance of the solution accuracy and the optimization tolerance can be achieved.

Two possible ways to incorporate the mesh adaptation and design optimization are

considered here: optimization-driven adaptation and adaptation-driven optimization. In the former approach, the optimization tolerance at each fidelity is prescribed by the user. The objective function is first evaluated on a relatively coarse mesh, and then the error estimation and mesh adaptation are performed to control the discretization error to be below the optimization tolerance at the current fidelity. The allowable discretization error decreases as the optimization fidelity increases. For the latter approach, several mesh levels, *i.e.*, DOF, are defined before the optimization. Again, we start with a fairly coarse mesh, and then the mesh is adapted/optimized for each design, subject to the given DOF level, to achieve the best accuracy. Once the objective change or the gradient norm is smaller than the objective error estimate, the optimization terminates at the current cost level and the fidelity increases through mesh adaptation with a higher cost. Depending on the information specified at the optimization setup, we therefore refer to the optimization-driven adaptation as the error-based approach, while we denote the adaptation-driven optimization as the cost-based approach. The error-based optimization needs an error-based mesh adaptation method: here we use error-based Hessian adaptation ⁴; while the cost-based one requires cost-based adaptation mechanics, which can be either cost-based Hessian adaptation or MOESS. More details about the adaptation mechanics can be found in Chapter 4.

Compared with the fixed-fidelity optimization, unnecessarily fine meshes at the early stages of the shape optimization are avoided in the two proposed multifidelity frameworks. Moreover, the areas that introduce most of the error may differ a lot for different designs during the optimization. Both approaches reduce the chance of over-refining areas that are not relatively important for the final design, which is important for problems where the initial and final designs are significantly different. Compared with the multifidelity optimization without error estimation and mesh adaptation, the optimization tolerance and the error estimate are tightly coupled in the proposed method to actively control the optimization at each fidelity and avoid the waste of low-fidelity convergence. Therefore, we expect that the two proposed methods can effectively prevent over-optimizing on a coarse mesh, or over-refining on an unintended design.

5.3.3 Consistent Objective-Sensitivity Analysis

One should note that in the proposed frameworks, even at the same fidelity, the mesh is not necessarily fixed as in the traditional design method. Rather, the mesh is also adapted

⁴MOESS can also be made error-based, however, the optimal step matrix tensor field is determined iteratively with a fixed cost as MOESS is formulated, making the error-based approach expensive and inefficient due to the cost changes during the adaptation. Therefore, only the cost-based adaptation strategy is considered for MOESS in this work.

if needed, *e.g.*, refined in the error-based approach or optimized in the cost-based one, to control the discretization error. Recall the discrete optimization problem in Eqn. 4.6, omitting the additional constraints for simplicity,

$$\begin{aligned} \min_{\mathbf{x}} \quad & J_h(\mathbf{U}_h(\mathbf{x}), \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{R}_h(\mathbf{U}_h(\mathbf{x}), \mathbf{x}) = \mathbf{0} \end{aligned} \tag{5.17}$$

The above problem formulation infers a dependence on the discretization h , *i.e.*, the computational mesh if the order is fixed. One could prove that by refining the discretization, the discrete optimization problem converges to the continuous one. However, it should also be mentioned that, the discrete optimization is an independent problem, which is characterized by the behavior of both the continuous problem and the discretization induced error,

$$J_h(\mathbf{u}_h, \mathbf{x}) = \mathcal{J}(\mathbf{u}, \mathbf{x}) + e_h(\mathbf{u} - \mathbf{u}_h, \mathbf{x}) \quad \Rightarrow \quad \frac{dJ_h}{d\mathbf{x}} = \frac{d\mathcal{J}}{d\mathbf{x}} + \frac{de_h}{d\mathbf{x}}, \tag{5.18}$$

where e_h is the exact discretization error of the output, assumed to be continuous with respect to the design variable \mathbf{x} .

In general, the discretized objective is incorrect compared to the exact continuous objective due to the numerical error induced by discretization. Consequently, the discretized objective gradient also involves the gradient of the error, which is unique in different discretizations. If the mesh is not adapted, then the numerical error may lead to a substantial deviation of the objective. Indeed, even if the mesh is adapted, but only on the objective and not its gradient, convergence to the true optimum of the continuous optimization problem may be hampered due to inaccuracy of the gradient approximation [66]. However luckily, the discretization error does not affect the convergence of the discrete optimization problem, since the discretized gradient analysis is consistent with the discretized objective function. In fact, if exact differentiation is used in Eqn. 5.18, *e.g.*, using algorithmic differentiation, one obtains the exact gradient of the discretized objective functional. Due to the consideration above, controlling the error of the sensitivity seems to be of limited importance for the optimization process. However, objective sensitivity is commonly used in the stopping criterion for the optimization problem, controlling the sensitivity error helps ensure the optimality condition. Hicken and Alonso [66] proposed a method to estimate the error of the output gradient norm. However, high-order derivatives need to be approximated and the computational cost is higher than output error estimation. Moreover, there exist scenarios where the gradient is more accurate than the output value

itself, *i.e.*, incremental output predictions are less computationally intensive than absolute output predictions as suggested in [63]. In these cases, only controlling the gradient errors may yield accurate designs, however, the final output values may not be accurate enough for practical use. In addition, apart from the optimality, the trim constraints also rely much on accurate absolute output computations to determine the design feasibility. Therefore, in this work we focus on controlling the error in the output predictions instead of the errors in the sensitivity calculations.

The proposed multifidelity optimization methods approximate the continuous optimization by generating a sequence of designs and discretizations, which converges to the optimal design of the continuous optimization problem,

$$\{J_h\} = \{J_{h_0}(\mathbf{x}_0), J_{h_1}(\mathbf{x}_1), \dots, J_{h_N}(\mathbf{x}_N)\}, \quad \lim_{h_N \rightarrow 0} |J_{h_N}(\mathbf{x}_N) - \mathcal{J}(\mathbf{x}^*)| = 0. \quad (5.19)$$

The convergence can be guaranteed by the error estimation given in Eqn. 4.36 if the optimization problem is smooth and convex. However, in a gradient-based optimization, a feasible path has to exist along these design and mesh pairs for successful convergence. In the proposed frameworks, inconsistent objective-sensitivity analysis may occur since the discretized optimization problem changes every time the mesh is adapted. Although all of the discretized problems (discrete optimizations) are approximations to the same continuous optimization problem, each of them has its own behavior associated with the embedded discretization error. There may not exist a feasible gradient-based update path between two design and discretization pairs, since the gradient depends on the discretization and hence cannot guide the update between different discretizations. This can lead to convergence issues, similar to optimization that utilizes the continuous adjoint for sensitivity analysis. Therefore, for the sake of consistent objective-gradient analysis, we update the design on the same mesh, and then perform mesh adaptation if needed. In other words, after obtaining the objective gradient on the current mesh, the line search is performed on the same mesh, and, when necessary, adaptation is only applied after the design update in each major iteration. From the point of view of the optimizer, each QP or LSEI subproblem is solved on the same mesh, while they are sequentially built on different adapted meshes. This approach yields a sequence of designs and discretizations that avoids possible convergence difficulties,

$$\{J_h\} = \{J_{h_0}(\mathbf{x}_0), J_{h_0}(\mathbf{x}_1), J_{h_1}(\mathbf{x}_1), \dots, J_{h_{N-1}}(\mathbf{x}_N), J_{h_N}(\mathbf{x}_N)\}, \quad \lim_{h_N \rightarrow 0} |J_{h_N}(\mathbf{x}_N) - \mathcal{J}(\mathbf{x}^*)| = 0. \quad (5.20)$$

5.3.4 Multifidelity Optimization Algorithm Overviews

The proposed optimization frameworks with error estimation and mesh adaptation are summarized in Algorithm 5.1 and Algorithm 5.2, using error-based and cost-based approaches, respectively ⁵. Optimization tolerance levels and cost levels are specified by the user, driving the mesh adaptation to actively control the numerical errors. The general idea of the algorithms is to use the information from the flow solution only to its accuracy limit by setting the optimization tolerance to the discretization error or controlling the error to be below the optimization tolerance. In this work, we assume that the error estimation is sufficiently accurate to represent the “true” numerical error, which may be inappropriate when the adjoint is not well-resolved or when the problem is highly nonlinear. On the other hand, there exist cases in practice when the low fidelity meshes are able to predict incremental objective differences more accurately than absolute objective values as discussed earlier. In other words, the objective functional shape with respect to the design parameters are well-predicted while the absolute output values are off. Therefore in the proposed optimization frameworks, a safety factor η is defined to control reliance on the error estimation and to allow optimization with less accurate information. However, a good choice of η can be problem dependent. Nonetheless, if the trim constraints are involved, allowing more optimization when the constraints are not accurately enforced can be dangerous if they are essential, such as some failure constraints. Furthermore, avoiding over-optimization is more important in multifidelity optimization if the optimization is not restarted, which will be discussed in the next section. Finally, in this work, we are interested in both accurate optimal design and accurate prediction of the corresponding output values. Therefore, the optimization tolerance and the discretization error estimate are required to be close enough to achieve better efficiency, *i.e.*, better accuracy at a fixed cost. More detailed discussion on this is given in Chapter 6.

5.3.4.1 Hessian Reuse/Restart

When implementing a multifidelity optimization, there is a choice of whether or not the optimization should be restarted when the fidelity is increased. In a quasi-Newton method, restarting the optimization will reinitialize the Hessian matrix \mathbf{B} , which then throws away all of the curvature information built on the low-fidelity optimization. This gives more flexibility for the design update on the higher fidelity. However, the Hessian approximation needs to be rebuilt which significantly slows down the high-fidelity convergence.

⁵Only the algorithms with the solver-based trimming approach are shown here, for the optimizer-based trimming approach, see [67].

Algorithm 5.1 Optimization with error estimation and mesh adaptation (error-based)

input : initial design \mathbf{x}_0 , initial coarse mesh \mathcal{T}_h , optimization tolerance levels $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n$, safety factor η

output: optimized design \mathbf{x}^*
 adapted mesh \mathcal{T}_h with controlled objective error $\delta J^{\text{adapt}} \leq \mathcal{O}_n$

```

1 for  $l = 1, 2, \dots, n$  do
2   set the total error tolerance as  $\tau_l = \eta \mathcal{O}_l$ 
3   while not converged do ▷ optimization algorithm
4     while  $\delta J^{\text{adapt}} > \tau_l$  do
5       adapt the mesh  $\mathcal{T}_h$  with refinements ▷ Hessian adaptation
6       update  $\mathbf{x}_{t,l}$  to meet trim constraints ▷ trimming process
7       compute the objective  $J^{\text{adapt}}$  and its error estimate  $\delta J^{\text{adapt}}$ 
8     end
9     calculate the objective gradient  $DJ^{\text{adapt}}/D\mathbf{x}_{s,l}$ , per Eqn. 4.20
10    update the active design  $\mathbf{x}_{s,l}$  with meshes  $\mathcal{T}_h$  fixed ▷ line search
11  end
12  finish optimization at level  $l$ ,  $\mathbf{x}_{l+1} = \mathbf{x}_l$ 
13 end

```

Algorithm 5.2 Optimization with error estimation and mesh adaptation (cost-based)

input : initial design \mathbf{x}_0 , initial coarse mesh \mathcal{T}_h , cost levels C_0, C_1, \dots, C_n , safety factor η

output: optimized design \mathbf{x}^*
 optimized accuracy/minimized objective error at given cost C_n

```

1 for  $l = 0, 1, \dots, n$  do
2   while not converged do ▷ optimization algorithm
3     for  $i = 1, \dots, N_{\text{adapt}}$  do
4       adapt/optimize mesh at fixed cost  $C_l$  ▷ Hessian adaptation/MOESS
5       update  $\mathbf{x}_{t,l}$  to meet trim constraints ▷ trimming process
6       compute objective  $J^{\text{adapt}}$  and its error estimate  $\delta J^{\text{adapt}}$ 
7     end
8     set optimization tolerance  $\mathcal{O}_l = \eta \delta J_h$ 
9     calculate the objective gradient  $DJ^{\text{adapt}}/D\mathbf{x}_{s,l}$ , per Eqn. 4.20
10    update the active design  $\mathbf{x}_{s,l}$  with meshes  $\mathcal{T}_h$  fixed ▷ line search
11  end
12  finish optimization at level  $l$ ,  $\mathbf{x}_{l+1} = \mathbf{x}_l$ 
13 end

```

On the other hand, the Hessian matrix approximation can be reused and kept during the fidelity increase. The current implementation in this work takes this approach. Nevertheless, in this approach, the design flexibility of the high fidelity is limited, as the search direction or the maximum step size given in the QP or LSEI problem is controlled by the Hessian. Therefore, over-optimization should be avoided as much as possible in this setting to prevent the design from being trapped around sub-optimal regions. As a result, in order to avoid over-optimization in the current work, the optimization tolerance is set to be equal to the estimated objective error in the cost-based approach, or the estimated error is always controlled to be below the optimization tolerance in the error-based method.

CHAPTER 6

Application to Aerodynamic Optimization

In this chapter, we test the frameworks developed in Chapter 5 to optimization problems governed by conservation-form PDEs, with a focus on aerodynamic shape optimizations. The key idea is to balance the discretization error and the optimization tolerance in the optimization problem as presented in Chapter 5, which is elaborated more in practical optimization problems in the current chapter. We first discuss in Section 6.1 the effects of the solution accuracy, *i.e.*, the discretization error, and its relation with the optimization tolerance in an optimization problem. Concrete examples are shown in Section 6.2, in which a simple optimization governed by one-dimensional advection-diffusion PDE and a more complicated airfoil optimization governed by Euler equations are considered. In order to improve the accuracy and the efficiency in these examples, optimization with error estimation and mesh adaptation, *i.e.*, adaptive CFD, is used to solve these problems again in Section 6.3. The two optimization frameworks proposed in Chapter 5 are both investigated to demonstrate their effectiveness of improving the design and their efficiency of reducing the errors. After showing benefits over traditional methods, we test our proposed frameworks on unconventional design problems and more practical turbulent cases in Section 6.3.3 and Section 6.3.4. In addition to accuracy and efficiency gains, benefits on mesh generation and optimization setup have also been shown. The observations in these problems are summarized in Section 6.4.

6.1 Solution Accuracy and Optimization Tolerance

If we have the exact solution to the PDEs that govern the system, optimization results will only depend on the optimization tolerance. Suppose that the continuous optimization is convex and smooth. The optimizer is guaranteed to converge to a solution within the tolerance compared to the exact optimum. However, in practice, the exact solution

is largely unavailable and thus a discretized flow solution is used to evaluate the objective functional and the sensitivity, *i.e.*, discrete optimization is adopted. Due to the discretization errors associated with the discrete optimization, the optimizer is working on a discrete functional $J_h(\mathbf{U}_h, \mathbf{x})$ rather than the exact continuous functional $\mathcal{J}(\mathbf{u}, \mathbf{x})$. The difference between these is the output discretization error, which can be estimated using the output adjoint as discussed in Chapter 4. Therefore, the optimization results now depend also on the discretization errors, which affect solution accuracy.

As mentioned in Section 5.3.3, the discrete optimization problem is characterized by not only the corresponding continuous optimization problem but also the numerical error embedded in the discretization. Suppose that the exact optimal design of the continuous optimization problem is \mathbf{x}^* , while the exact optimal design of the discrete optimization problem is \mathbf{x}_h^* . The difference between \mathbf{x}^* and \mathbf{x}_h^* is a measure of the solution accuracy, yet difficult to estimate. Instead, the objective error estimate, $\|J_h(\mathbf{x}_h^*) - \mathcal{J}(\mathbf{x}^*)\|$, using the adjoint-based approach as shown in Chapter 4, can serve as an indicator of the solution accuracy¹. For a primal-dual consistent discretization we have $\lim_{h \rightarrow 0} \|\mathbf{x}_h^* - \mathbf{x}^*\| = 0$ and $\lim_{h \rightarrow 0} \|J_h(\mathbf{x}_h^*, \mathbf{U}_h) - \mathcal{J}(\mathbf{x}^*, \mathbf{u})\| = 0$, and hence the discrete optimization result will converge to the exact optimum of the continuous optimization problem. However, by using a gradient-based optimizer, the optimization results also depend on the optimization tolerance, and very often we converge to a design within the tolerance compared to the exact optimum of the discrete optimization problem, $\|\tilde{\mathbf{x}}_h^* - \mathbf{x}_h^*\| \leq \epsilon$ or $\|J_h(\tilde{\mathbf{x}}_h^*, \mathbf{U}_h) - J_h(\mathbf{x}_h^*, \mathbf{U}_h)\| \leq \epsilon$, where $\tilde{\cdot}$ terms denote practically computable results we get from numerical optimization. In conclusion, the exact error in the computable result of a practical numerical optimization comes from two sources, one is the numerical error associated with the discretization and the other is the error due to incomplete convergence of the numerical optimizer controlled by the optimization tolerance. If we denote the solution accuracy or the numerical error as \mathcal{E} , and the optimization tolerance as ϵ , a strict bound for the total error in the optimization result can be defined as

$$\|J_h(\tilde{\mathbf{x}}_h^*) - \mathcal{J}(\mathbf{x}^*)\| \leq \|J_h(\tilde{\mathbf{x}}_h^*) - J_h(\mathbf{x}_h^*)\| + \|J_h(\mathbf{x}_h^*) - \mathcal{J}(\mathbf{x}^*)\| = \epsilon + \mathcal{E}. \quad (6.1)$$

From the equation above, we can clearly see that the quality of the optimization results depends on both the discretization error and the prescribed optimization tolerance. The computational cost required to reduce the discretization error directly depends on the computational mesh, while the cost associated with a reduction of optimization tolerance

¹More precisely, standard adjoint method only estimates the error due to the solution inaccuracy assuming the same design, *i.e.*, $\|J_h(\mathbf{x}_h^*, \mathbf{U}_h) - J(\mathbf{x}_h^*, \mathbf{u})\|$; while the coupled adjoint approach introduced in Chapter 4 includes design inaccuracy as well, $\|J_h(\mathbf{x}_h^*, \mathbf{U}_h) - J(\mathbf{x}^*, \mathbf{u})\|$.

depends on both the computational mesh and the dimension of the design space. The optimal error distribution between \mathcal{E} and ϵ is fairly complicated, but we assume the error equidistribution principle still holds. For instance, the over-refining scenario mentioned in Section 5.3.2 can now be mathematically defined as $\mathcal{E} \ll \epsilon$, which means the optimizer cannot fully utilize the current solution accuracy to optimize the design, *i.e.*, the total error is bounded by the optimization tolerance. Hence we can reduce the cost in the mesh, *i.e.*, coarsen the mesh, and meanwhile tighten the optimization tolerance, *i.e.*, perform more optimization iterations, to improve the design quality without increasing the overall computational cost. On the other hand, in the over-optimizing cases, the discretization error is high and the optimization tolerance is relatively low, $\epsilon \ll \mathcal{E}$, such that the optimizer may work on the discretization error instead of the physics to improve the design, *i.e.*, the total error is dominated by the discretization error \mathcal{E} . In these cases, we can refine the mesh while loosening the optimization tolerance to improve the design quality without sacrificing the overall computational efficiency.

Although over-refining and over-optimizing are both computationally inefficient for optimization problems, the latter one is more often seen in practice since the optimization tolerance can be easily reduced in the optimization setup and is often set to be tight enough in most problems despite possible inefficiency. For example in transonic airfoil optimization, a physical intuition of the optimization is to remove the shocks that appear on the airfoil surface, which often requires very tight optimization tolerance even on a coarse mesh when the solution is inaccurate. Thus the optimization will often converge to a “shock-free” design on the current mesh, however, the very detailed design modification to remove the shock may be associated with the discretization error instead of the physics ². Although not computationally efficient, the optimized design $\tilde{\mathbf{x}}_h^*$ will be close to the true optimal design \mathbf{x}^* as long as the discretization error is not overly large. However, on the other hand, if the discretization error is too high, then the discrete optimization problem is not anymore a good approximation to the corresponding continuous optimization problem. As a result, over-optimizing on current mesh may point to the numerically exact optimum, \mathbf{x}_h^* , that is far from the true physical optimum, \mathbf{x}^* , as the discretization error is high. In some worse scenarios, large discretization errors can even create spurious optimum or saddle points. More dangerously, these numerical artifacts do not affect the convergence of the discrete optimization, while the design issues can only be found after the optimization.

²In fact at some high Mach numbers, if we refine the mesh, shocks will most likely appear again as the “shock-free” design is an isolated point which will switch to weak shock solutions if any perturbation presents [182], including boundary condition perturbations or mesh changes in adaptation. Lyu *et al.* [42] have observed the shock reappearing in aircraft wing optimizations and similar behavior has also been found in airfoil optimization problems considered in the current work as shown later in Section 6.2.2.

Although in a multifidelity setting, either through an *a priori* fixed fine mesh or via mesh adaptation, the design can be pulled back to physically optimal by undoing the incorrect design modifications from the lower fidelity, yet the convergence on the high-fidelity optimization slows down and the potential benefits of multifidelity optimization is thus limited. Furthermore, if the multifidelity optimization uses a quasi-Newton method and the Hessian approximation is reused, over-optimizing is even more undesirable as the design update can be limited by the inaccurate Hessian approximation constructed by inaccurate information from the lower fidelity.

To summarize, aerodynamic optimization in practice is often over-optimizing on the given mesh such that the design quality (total error) is restricted by the discretization error. If the discretization error is high, the optimized design might be numerically optimal, *i.e.*, close to the true optimum of the discrete optimization problem \mathbf{x}_h^* , but not physically optimal, *i.e.*, the optimized design is far from the true optimum of the continuous optimization problem or can even be a spurious optimum. On the other hand, computational accuracy and efficiency stall when either the solution accuracy or the optimization tolerance dominates the optimization.

6.2 On the Effects of Discretization Errors in Optimization

6.2.1 One-Dimensional Scalar Advection-Diffusion

In this section, we take a closer look at the example shown in Figure 1.6a, which is an optimization problem governed by a one-dimensional scalar advection-diffusion equation,

$$\begin{aligned} a \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} &= 0, & x \in [0, L]; \\ u(0) &= 0, & u(L) = 1, \end{aligned} \tag{6.2}$$

where a is the advection velocity, ν denotes the viscosity and L is the domain length. The optimization problem is formulated as seeking an optimal Péclet number (Pe), defined as $Pe = aL/\nu$, to minimize the negative scalar gradient at a specified location,

$$\min_{Pe} \left. -\frac{\partial u}{\partial x} \right|_{x=0.76L}. \tag{6.3}$$

The objective functional $\mathcal{J} = -\frac{\partial u}{\partial x}|_{x=0.76L}$ is an implicit function of the design variable $\mathbf{x} = [Pe]$, defined by the underlying governing equation in Eqn. 6.2. We discretize the continuous optimization problem using DG with approximation order of $p = 2$ on a fixed computational mesh with $N_e = 8$ elements. The mesh elements are uniformly distributed

in the computational domain. As the governing equation in Eqn. 6.2 is relatively simple and the analytical solution can be obtained, we compare the discretized objective functional $J_h(\mathbf{x})$, *i.e.*, numerical solution, and the continuous one, *i.e.*, analytical solution, in Figure 6.1a. We can see that the naive uniformly distributed mesh produces a spurious optimum besides the expected one close to the true optimum, due to the discretization error. As mentioned in Section 6.1, the discrete optimization problem is characterized by both the continuous problem (near the exact optimum) and the discretization error (near the spurious optimum). Moreover, the discrete optimization problem is smooth around the spurious optimum which will not affect the optimizer convergence even though it is purely numerical artifact. Therefore, the optimization performed on the uniform mesh can heavily depend on the starting point, especially for gradient-based methods. The optimizer may converge to the spurious local optimum if the descent direction is pointing to it. In order to more closely study the spurious optimum, we plot the corresponding state solution $u(x)$ in Figure 6.1b. We can observe a severe numerical oscillation near the location of interest for the solution on the uniform mesh, while the analytical solution is almost flat nearby. The numerical oscillation shown in Figure 6.1b is only one of the possible sources that may cause spurious optima. Many physical features that are sensitive to discretization errors, such as boundary layers or shocks in flow problems, can also potentially create spurious optima.

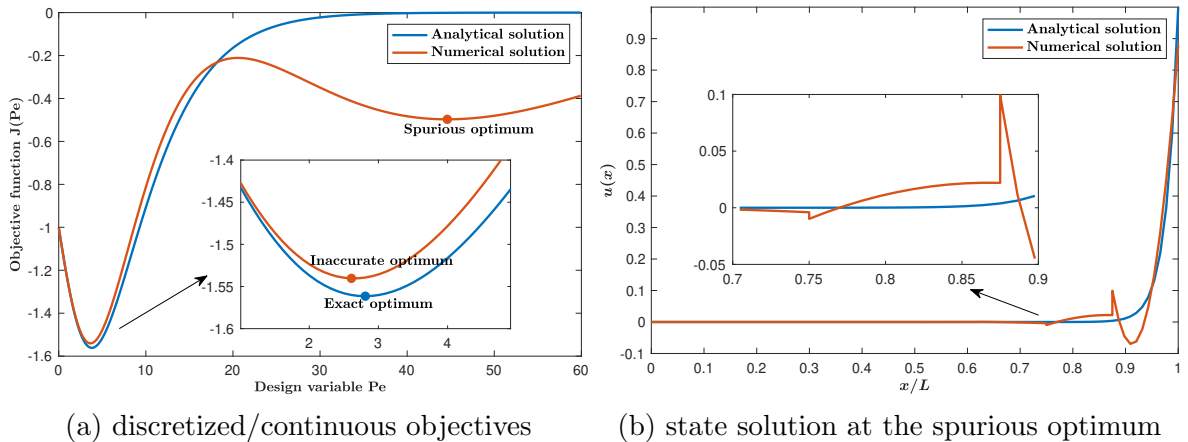


Figure 6.1: Optimization governed by a one-dimensional scalar advection-diffusion PDE on a uniform coarse mesh.

We perform two optimizations, both with a very tight optimization tolerance to eliminate the uncertainty due to optimization tolerance, such that we can see more clearly the effects of discretization error in this case. Starting from two different Pe numbers,

the results are listed in Table 6.1 ³. As expected, the optimized design converges to the spurious optimum when the initial point is close to it, while it converges to the more correct one if the initial design is close to the true optimum. Both the exact error measured with respect to the true solution and the error estimate using the adjoint-based method are shown in Table 6.1 as well. If the optimizer converges to an inaccurate optimum that is close to the exact one, the error estimate gives a good prediction of the optimization accuracy. On the other hand, if the optimizer converges to the spurious optimum caused by the discretization error, the error estimates at the spurious optimum, while not very effective, are generally large enough to indicate that the optimization is not converging correctly. Thus even if mesh adaptation is not performed, the objective error estimate by itself still gives a good assessment of the design quality. Results incorporating with mesh adaptation are shown in Section 6.3.1.

Table 6.1: Optimization results (advection-diffusion PDE) on a uniform mesh. The discretization uses DG with $p = 2$ and $N_e = 8$ elements, the optimization tolerance is set to be $\epsilon = 10^{-10}$.

Initial design	\mathbf{x}_h^*	$J_h(\mathbf{x}_h^*)$	$\delta J_h(\mathbf{x}_h^*)$	$\ \mathbf{x}_h - \mathbf{x}^*\ $	$\ J_h(\mathbf{x}_h^*) - \mathcal{J}(\mathbf{x}^*)\ $
$\mathbf{x}_0 = 40$	44.60287	-0.49700	-0.15555	40.79686	1.06446
$\mathbf{x}_0 = 20$	3.64579	-1.54037	0.02063	0.16022	0.02109

6.2.2 Inviscid Transonic RAE 2822 Airfoil Optimization

In this section, we consider a higher-dimension optimization problem governed by more complicated PDEs, namely the transonic airfoil optimization problem. Although less practical than three-dimensional wing or aircraft design optimization, airfoil optimization has long been a representative problem in aerodynamic optimization since the aerodynamic performance of an aircraft wing is strongly affected by span-wise airfoil shapes. Transonic airfoil optimization is of particular interest as the cruise speed of modern business jets often falls into the transonic regime. Moreover, the drag coefficient of the airfoil, often used as the optimization objective, is very sensitive to the design variations due to the involvement of shocks. Consequently, transonic optimization problem is a good testing problem for both the optimization algorithms and the CFD flow simulations.

We consider here the optimization of a Royal Aircraft Establishment (RAE) 2822 airfoil [183]. The goal of the optimization is to seek an optimal airfoil shape and angle of

³Since the optimization tolerance is very small compared to the discretization error, *i.e.*, $\epsilon \ll \mathcal{E}$, we assume $\tilde{\mathbf{x}}_h^* = \mathbf{x}_h^*$ in the table.

attack to minimize the drag coefficient, subject to a fixed lift trim condition and a minimum area constraint. This optimization problem originates from the second case of a set of benchmark cases proposed by ADODG ⁴, although for simplicity we ignore the moment constraint here. Furthermore, inviscid flow simulation (Euler equation) is used with an element-based artificial viscosity model [128] for shock capturing. A fully-turbulent version of this problem will be investigated later in Section 6.3.4. The optimization problem starts with a RAE 2822 airfoil at a freestream Mach number of 0.734, the target lift of the trimming condition is $\bar{c}_\ell = 0.824$ and the minimum area constraint is set to be the initial area of RAE 2822. The airfoil is parameterized with 16 Hicks-Henne basis functions (8 each for the upper and lower surfaces), and cubic curved mesh elements are used to represent the boundary.

In this problem, although error estimates are used to assess the design quality during the optimization, the computational mesh is not adapted. We manually generate a mesh around the RAE 2822 airfoil, which consists of 2009 elements as shown in Figure 6.2. The farfield is a square that is 2000 chords away from the airfoil and the airfoil boundary is represented by cubic curved elements. Four optimizations with order p from 1 to 4 are performed independently, all starting with this same mesh on the RAE 2822 airfoil. The optimization tolerance is set to be around 5×10^{-8} for all of the optimizations ⁵, which is expected to be much smaller than the discretization error on the current mesh with the specified approximation orders. The objective convergence, with estimated objective error is compared in Figure 6.3a, while the final optimized shapes are compared in Figure 6.3b. As we expected, the estimated objective error is extremely high for lower p . Although as p increases the error drops down, yet the error estimates for higher orders are still much larger than the optimization tolerance. In Figure 6.3b, we can see that the final optimized shapes using lower orders $p = 1, 2$ are significantly different from the optimized shapes obtained using higher orders, $p = 3, 4$. However, surprisingly, the Mach number contours and the pressure coefficient distributions on all of these designs using the associated solution order exhibit “shock-free” features as shown in Figure 6.4. Since here we have high-order solutions available, *e.g.*, the $p = 3, 4$ solutions, we would trust more their solutions and prefer the resulting supercritical-kind designs which are in general favorable for transonic regimes. However, in practice, given a fixed mesh and a low-order solution, it is difficult to assess the design by just examining the objective convergence and the solution field.

In order to further examine the designs obtained using lower-orders, we use $p = 4$ on

⁴Available at <https://sites.google.com/view/mcgill-computational-aerogroup/adodg>

⁵In fact, the tolerances are adjusted until a nearly “shock-free” design is obtained on each order.

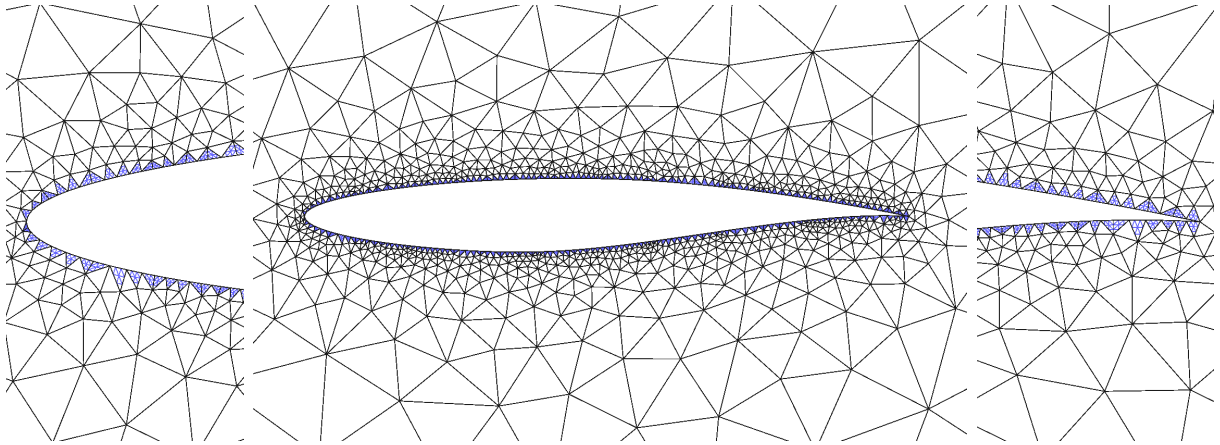


Figure 6.2: Fixed manually generated mesh on the RAE 2822 airfoil. The left and right figures show the mesh zoomed into the leading edge and the trailing edge, respectively. The blue nodes denote the high-order nodes in the curved boundary elements, while the rest of the elements are linear.

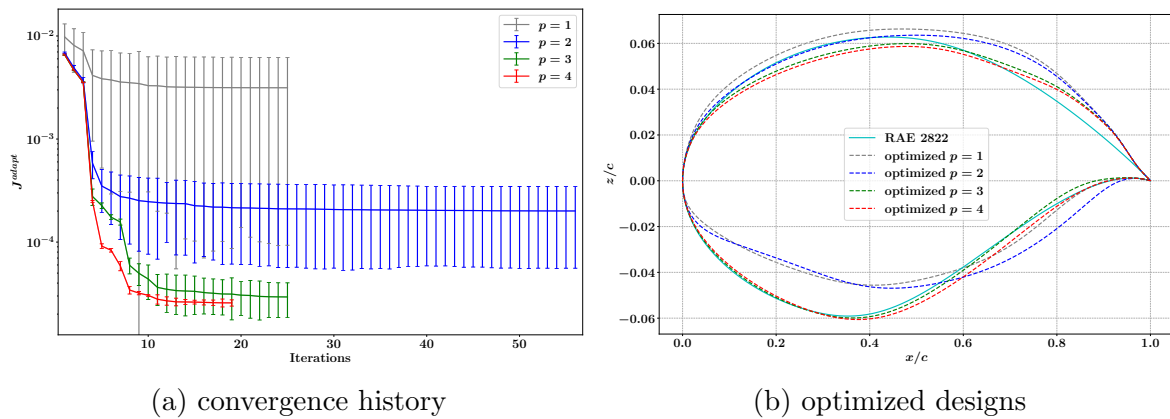


Figure 6.3: Comparison of optimizations on a fixed mesh with various approximation orders. In the convergence history plot, the error bars show the error estimates including the constraint output error, per Eqn. 4.37.

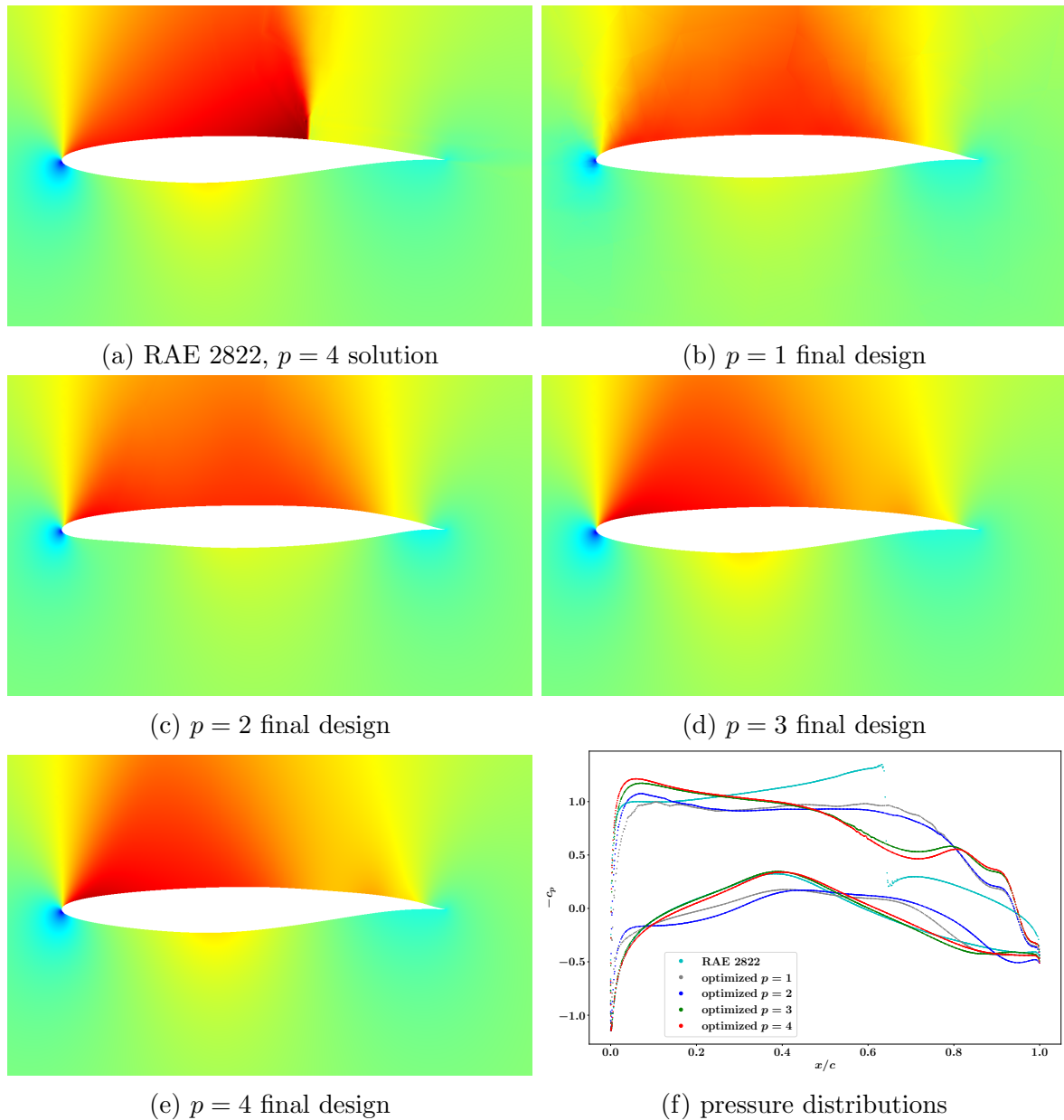


Figure 6.4: Mach contour and pressure distribution of the optimized designs on a fixed mesh with various approximation orders. The Mach number contour on the initial RAE 2822 airfoil is obtained using a $p = 4$ solution, while the Mach contours on the optimized designs are from solutions on the corresponding orders, *e.g.*, the Mach contour on the $p = 2$ optimized design is from the $p = 2$ flow solution. All of the Mach contour plots have the same color limits, $[0, 1.35]$.

the current mesh as the “exact” space, and re-solve the flow and trimming equations ⁶ on the final designs obtained from lower-order solutions. Moreover, some intermediate designs from the low-order optimizations are also re-examined using the “exact” $p = 4$ space. The re-analysis results for $p = 1, 2, 3$ are shown in Figures 6.5–6.7, and the detailed objective values are summarized in Table 6.2. As we can see in these figures, the “shock-free” designs on the original low-order spaces all produce new shocks when the approximation order is increased, *i.e.*, when the discretization error is reduced. Dramatic flow field changes can be observed on the final designs from the $p = 1$ and $p = 2$ optimizations, while the flow field on the $p = 3$ final design is similar when increasing the order, yet weak shocks are still observed when the flow field is more resolved. More importantly, for these low-order optimizations, we are able to find some intermediate designs that exhibit better performance, *i.e.*, lower drag coefficients under the same trim condition, than the original optimized design, when measured on the $p = 4$ space. Also, the shapes of these intermediate designs are found to be closer to the “true” $p = 4$ optimized design. However, due to the discretization errors in the original low-order approximation space, they are not considered superior to subsequent designs and are aborted by the optimizer. This implies that the optimizer, after these intermediate designs, is most likely working on the discretization error instead of the physics to improve the design. This type of over-optimization can lead to incorrect designs when the discretization error is high such as the $p = 1$ and $p = 2$ optimizations in this case, which is similar to what we have found in Section 6.2.1; on the other hand, when the discretization error is not very high but still much higher than the optimization tolerance, such as $p = 3$ in this case, the over-optimization reduces the efficiency of the optimization as considerable computational resources are wasted on unnecessary convergence that is dominated by the discretization errors. In addition, both scenarios can hamper the effective use of a multifidelity optimization framework, as in the former case the low-fidelity optimization will provide the high-fidelity optimization a bad starting design which slows down the high-fidelity convergence; while the latter one will waste much computational time on unnecessary low-fidelity convergence although the high-fidelity optimization is not much affected ⁷.

As depicted in Figure 6.3a, the low-order optimization is still effective when the ob-

⁶When analyzing the designs, we always enforce the lift constraint. In other words, as we increase the p to $p = 4$, the trimming condition is satisfied by varying the angle of attack while keeping the shape fixed.

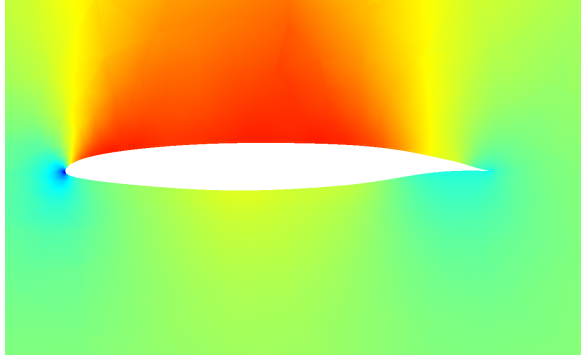
⁷In multifidelity frameworks where the optimizer is not restarted/reinitialized when increasing the fidelity, *e.g.*, the Hessian matrix is not reinitialized in this work, over-optimization in low fidelity might limit the design update on the high fidelity although the starting design for the high fidelity is not affected much.

jective difference is lower than the estimated objective error, which means picking an optimization tolerance slightly lower than the objective error estimate might have potential benefits, especially when the objective error is not too high, *e.g.*, the $p = 3$ optimization in Figure 6.3a. In those cases, the discretization error might still be high compared to the optimization tolerance, while the gradient (objective relative shape) might be accurate enough for the prescribed optimization tolerance. If we only care about the final design, this benefit might be useful but more study is required to investigate the optimal optimization tolerance given a discretization error level. However, if we are concerned about the final accuracy in the objective as well, as suggested in Eqn. 6.1, the final accuracy is always bounded by the discretization error if the optimization tolerance is low. In these scenarios, it is always more beneficial to shift the cost from the optimizer to the flow solver side, *i.e.*, increasing the optimization tolerance while refining the mesh.

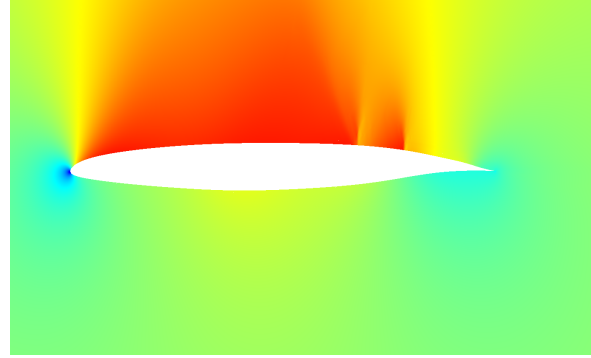
On the other hand, as we can see in Table 6.2, the objective error estimate is fairly accurate in most cases, though the errors on the $p = 3$ designs tend to be a little over-estimated. Therefore, it is in general effective to use the objective error estimate to guide the optimization, or in turn to use the optimization tolerance to restrict the allowable objective error estimate. In this work, if mesh adaptation is applied, we refine the mesh until the objective error estimate is below the optimization tolerance in an error-based approach, while if the cost-based approach is used, we set the optimization tolerance to be always equal to the estimated objective error. More details of the proposed optimization frameworks can be found in Chapter 5 and the results of applying the proposed adaptive CFD approaches on this problem are given in Section 6.3.2.

Table 6.2: Optimization results (inviscid transonic RAE 2822 airfoil optimization) on a fixed mesh with various approximation orders. The design index is the iteration number of the considered design, J_H denotes the objective evaluated on the current space, while J_h is the objective obtained on the “true” $p = 4$ space; the estimated δJ_h represents the error estimate using the adjoint-weighted residual, per Eqn. 4.37, while the “true” δJ measures the difference between the objectives on the current and the “true” space.

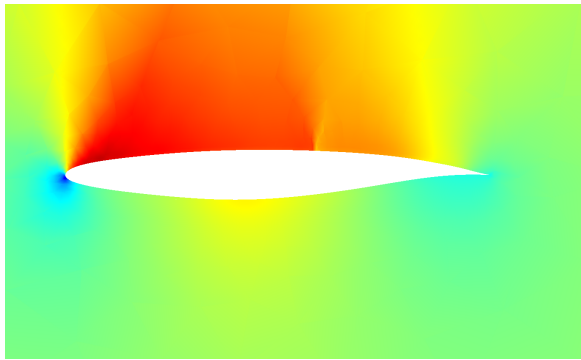
p	design index	J_H	J_h	“true” δJ	estimated δJ
1	6	3.74607×10^{-3}	1.89360×10^{-4}	3.55671×10^{-3}	3.45309×10^{-3}
1	25	3.14681×10^{-3}	2.75378×10^{-4}	2.87143×10^{-3}	3.05369×10^{-3}
2	8	2.66938×10^{-4}	5.23784×10^{-5}	2.14560×10^{-4}	1.71277×10^{-4}
2	56	2.00560×10^{-4}	6.10801×10^{-5}	1.39480×10^{-4}	1.44399×10^{-4}
3	19	3.12910×10^{-5}	2.56854×10^{-5}	5.60560×10^{-6}	1.36850×10^{-5}
3	25	2.93655×10^{-5}	2.60104×10^{-5}	3.35510×10^{-6}	1.09624×10^{-5}



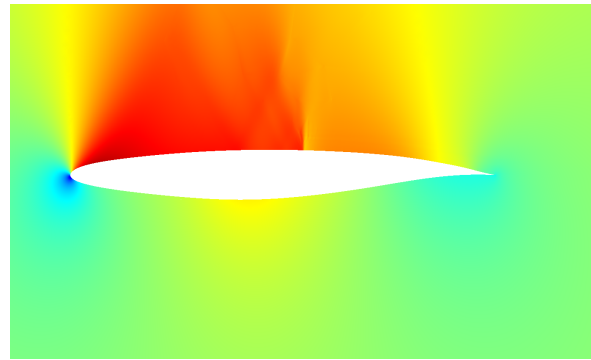
(a) $p = 1$ final design, $p = 1$ solution



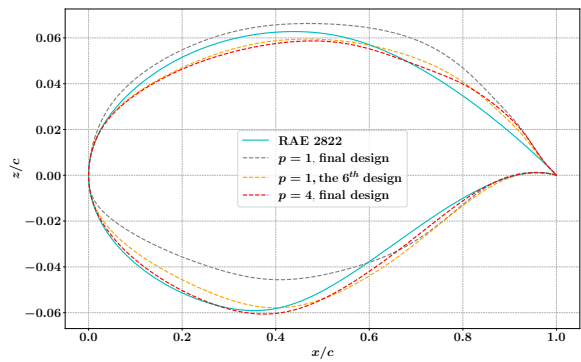
(b) $p = 1$ final design, $p = 4$ solution



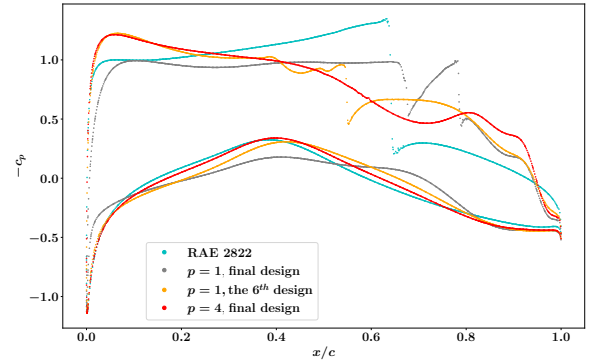
(c) $p = 1$, the 6th design, $p = 1$ solution



(d) $p = 1$, the 6th design, $p = 4$ solution

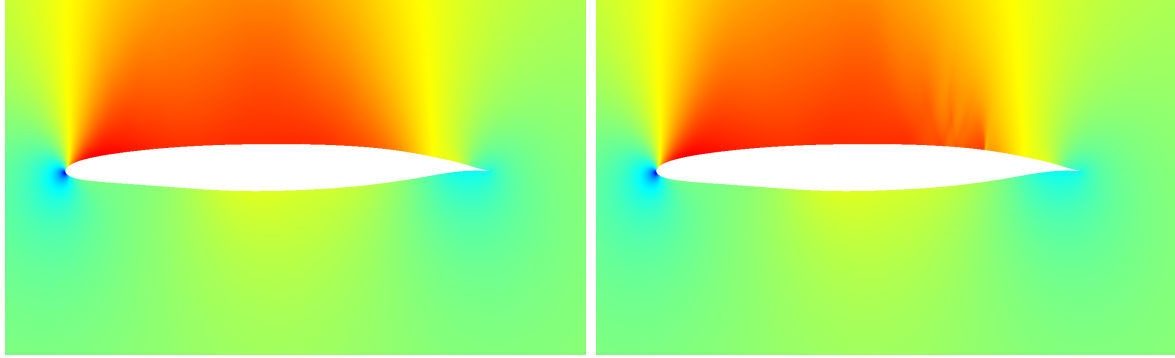


(e) design comparison



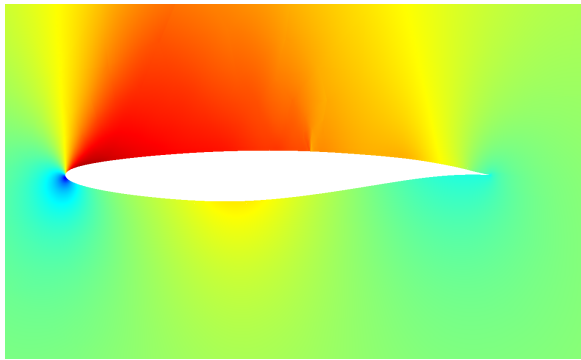
(f) pressure distributions

Figure 6.5: Re-analysis of the $p = 1$ designs on the $p = 4$ “true” space. All of the Mach contour plots have the same color limits, $[0, 1.35]$.

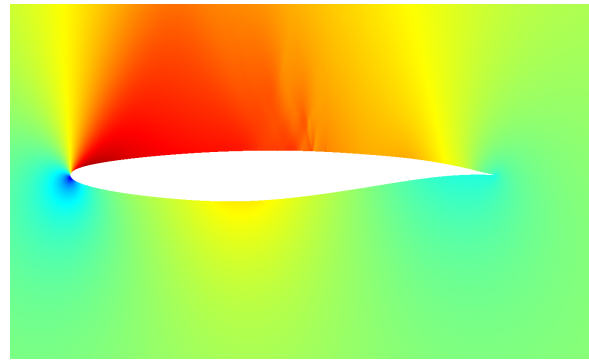


(a) $p = 2$ final design, $p = 2$ solution

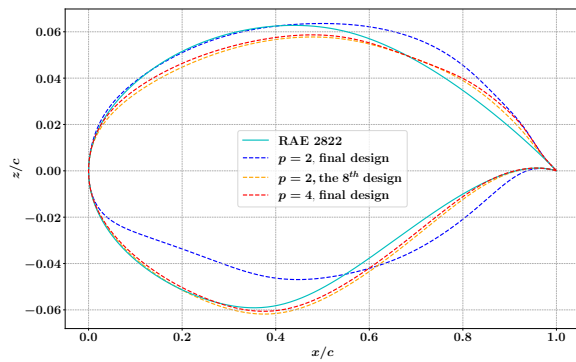
(b) $p = 2$ final design, $p = 4$ solution



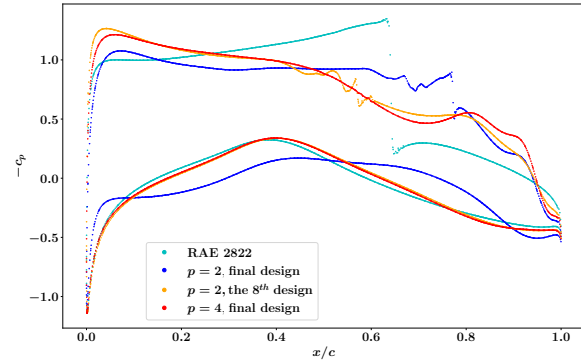
(c) $p = 2$, the 8th design, $p = 2$ solution



(d) $p = 2$, the 8th design, $p = 4$ solution



(e) design comparison



(f) pressure distributions

Figure 6.6: Re-analysis of the $p = 2$ designs on the $p = 4$ “true” space. All of the Mach contour plots have the same color limits, $[0, 1.35]$.

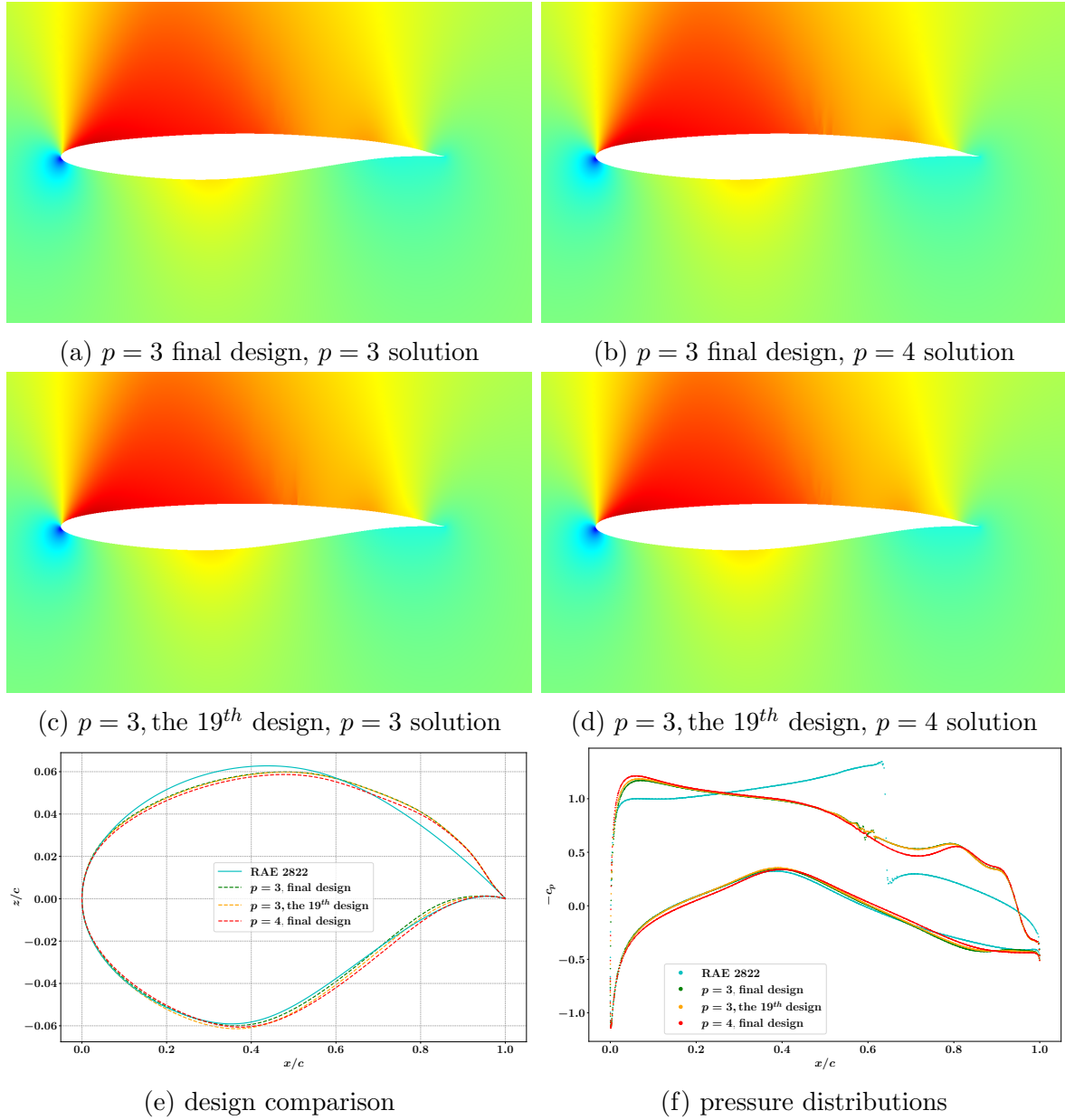


Figure 6.7: Re-analysis of the $p = 3$ designs on the $p = 4$ “true” space. All of the Mach contour plots have the same color limits, $[0, 1.35]$.

6.3 Optimizations with Adaptive CFD

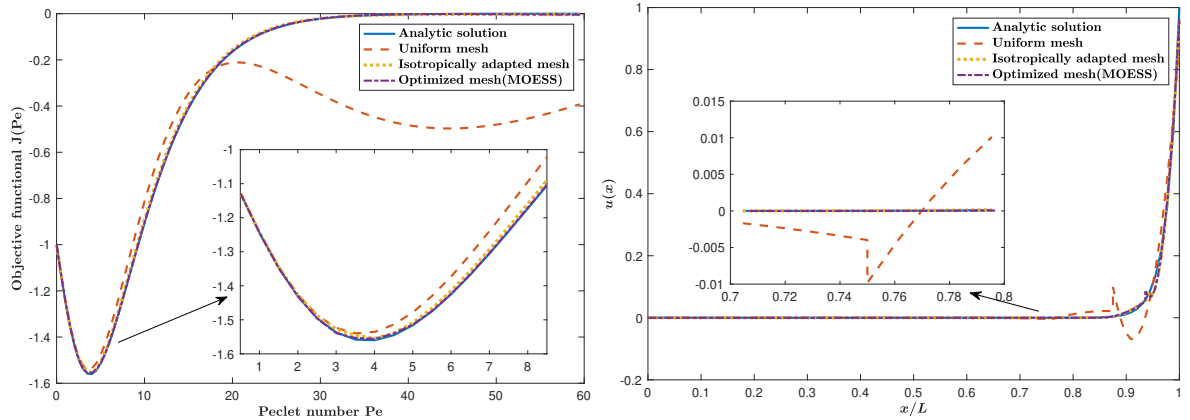
As we can see from the examples in Section 6.2, the discretization error can have detrimental effects on optimization. Although by estimating the discretization error during the optimization, the design quality can be effectively assessed, yet the error is still not controlled and the optimization has to restart with a better mesh to avoid poor designs. In order to actively control the discretization error and to better balance the discretization error and optimization tolerance, the optimization frameworks using adaptive CFD introduced in Chapter 5 are applied to improve the accuracy and efficiency of the optimization. Again, we start with the one-dimensional scalar advection-diffusion problem, followed by two-dimensional airfoil optimization problems.

6.3.1 Revisit of the One-Dimensional Advection-Diffusion Problem

The same optimization problem as considered in Section 6.2.1 is studied here. Instead of optimizing on a fixed uniform mesh, we this time use two adapted meshes: one isotropically-refined mesh based on localized output errors and one optimized mesh obtained from MOESS⁸. To compare with the fixed uniform mesh used in Section 6.2.1, we enforce the DOF of the adapted meshes to be the same, *i.e.*, $N_e = 8$, however the mesh is actively adapted during the optimization. Meanwhile, the optimization tolerance is also kept the same as used in Section 6.2.1. The discretized objectives on different meshes are compared in Figure 6.8a. By actively adapting the mesh during the optimization, the spurious optimum is eliminated as shown in Figure 6.8b and thus a reasonably accurate objective functional profile over the entire design space is obtained. Therefore, the optimization performed with adapted meshes are more robust with respect to the starting point, as the objective shape over the entire design space is better preserved by reducing the discretization error. Since the optimization tolerance is much smaller than the discretization error in this case, the design quality is mostly restricted by the discretization error around the exact optimum. By zooming into the region near the exact optimum, we can see that the uniform mesh has the highest error and thus leads to a most inaccurate optimum. The optimized mesh tends to give better accuracy compared to the isotropically-adapted mesh, as the former approach has more flexibility of redistributing the mesh nodes while the latter one only isotropically refines the elements with the highest error. The mesh difference is also well-reflected in Figure 6.8c. The numerical

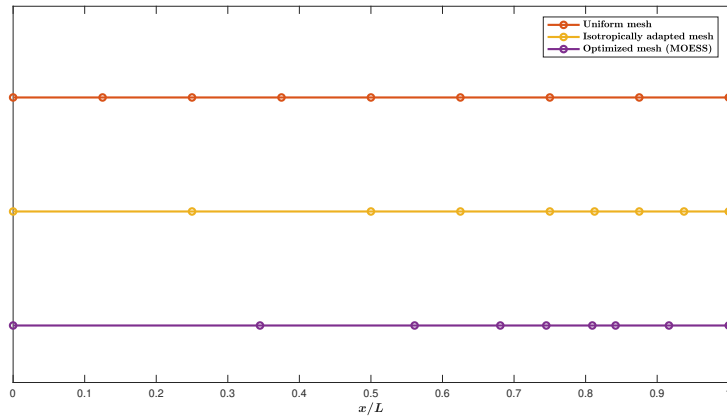
⁸In the isotropic adaptation, we start with a coarse mesh and isotropically refine the elements with highest objective error indicators until the target cost is met. In one dimension, there is no directional anisotropy, while Hessian adaptation and MOESS can still be used to only control the element sizing. For simplicity, we only test MOESS here.

oscillation observed in this problem represents only one of the possible sources that may cause spurious optima. Many physical features that are sensitive to discretization errors, such as boundary layers or shocks in flow problems, can also potentially create spurious optima. Thus output-based error estimation and mesh adaptation are expected to be more important in complex optimization problems, especially for novel configurations in which little knowledge or experience is available *a priori*.



(a) discretized/continuous objectives

(b) state solution at the spurious optima



(c) mesh node distribution of the fixed and adapted meshes

Figure 6.8: Optimization governed by the one-dimensional scalar advection-diffusion PDE on different meshes.

Again, we perform two optimizations starting from different Pe numbers as Section 6.2.1, but with adapted meshes this time. The results are summarized in Table 6.3. As expected, the optimization on the fixed uniform mesh converges to the spurious optimum when the initial design is close to it, whereas the adapted meshes produce more accurate results that are more robust to the starting point. Furthermore, optimization with meshes produced by MOESS exhibits better accuracy since the optimized meshes tend to obtain a better approximation of the objective over the design space. Also, similar

to what we have seen in Section 6.2.1, objective error estimates on all of the meshes give an accurate prediction of the optimization accuracy if the design converges close to the true optimum. On the other hand, error estimates around the spurious optima are often large enough to indicate poor design quality.

Table 6.3: Optimization results (advection-diffusion PDE) on uniform and adapted meshes. The discretization uses DG with $p = 2$ and $N_e = 8$ elements, the optimization tolerance is set to be $\epsilon = 10^{-10}$.

Initial design	Mesh	\mathbf{x}_h^*	$J_h(\mathbf{x}_h^*)$	$\delta J_h(\mathbf{x}_h^*)$	$\ \mathbf{x}_h - \mathbf{x}^*\ $	$\ J_h(\mathbf{x}_h^*) - \mathcal{J}(\mathbf{x}^*)\ $
$\mathbf{x}_0 = 40$	Uniform	44.60287	-0.49700	-0.15555	40.79686	1.06446
	Iso-adapted	3.72145	-1.54724	0.01309	0.08455	0.01423
	Optimized	3.81001	-1.56218	-0.00063	0.00400	0.00072
$\mathbf{x}_0 = 20$	Uniform	3.64579	-1.54037	0.02063	0.16022	0.02109
	Iso-adapted	3.84404	-1.55622	0.00505	0.03803	0.00525
	Optimized	3.81001	-1.56218	-0.00063	0.00400	0.00072

6.3.2 Revisit of the Inviscid Transonic Optimization on the RAE 2822 Airfoil

In this section, we revisit the inviscid transonic airfoil optimization problem that was considered in Section 6.2.2. As shown earlier, the optimization results heavily depend on the accuracy of the solution, *i.e.*, the discretization errors. High discretization errors induced by either coarse meshes or low approximation orders can potentially lead the optimizer converging to an undesired design. On the other hand, given limited accuracy on the objective output, tight optimization tolerance might force the optimizer to work on the discretization error rather than the actual physics. Therefore, either the mesh needs to be adapted to meet the current optimization requirement or the optimization tolerance needs to be loosened to avoid excessive design exploration using inaccurate information.

From previous study in Section 6.2.2, the initial RAE 2822 airfoil features a strong shock at the specified trimming condition, thus an initial mesh can be carefully designed or even adapted to capture the shock effectively. However, this only helps the analysis on the original shape, which will be surpassed by other designs quickly in the optimization. Particularly, we expect the design to be improved such that the shock is significantly weakened or removed. Any substantial refinement on the initial shock location will thus not effectively increase the accuracy but instead add considerable computational cost to the optimization. As a result, a computationally affordable mesh with specific refinements around the airfoil is generally used in this optimization.

In order to improve the accuracy and efficiency of the optimization, we applied the

optimization frameworks with adaptive CFD proposed in Chapter 5. Both the error-based and the cost-based approaches as described in Algorithm 5.1 and Algorithm 5.2 are tested in this problem. The error-based approach uses the error-based Hessian adaptation, while the cost-based approach adopts either the MOESS or the cost-based Hessian adaptation. Thanks to the automated adaptation process, a fairly coarse mesh can be used, which alleviates the efforts on mesh generation and thus accelerates the design setup. The starting mesh for the optimization with adaptive CFD consists of 393 triangular elements as shown in Figure 6.9a. Again, the farfield is 2000 chords away from the airfoil and cubic curved elements are used to represent the airfoil boundary. In the error-based optimization, a set of optimization tolerance levels is specified with an ultimate tolerance of 0.02 drag counts, *i.e.*, 2×10^{-6} . On the other hand, the cost-based optimization starts with a fairly low cost level, and degrees of freedom are added once the optimization converges at the current cost level, until the final optimization tolerance, set to be equal to the output error estimate, is close to 0.02 drag counts. To compare with traditional optimization method, we also run a fixed-fidelity optimization on a fixed mesh as shown in Figure 6.9b, which has comparable DOF as the final meshes obtained by cost-based mesh adaptation. All the simulations use the same solver setting as Section 6.2.2, while to focus on mesh adaptation, a fixed order $p = 2$ is used.

The adapted final meshes obtained using different methods are shown in Figures 6.9c–6.9e. All of the adapted meshes have similar refinement patterns: isotropic elements are put around the leading and trailing edges while mesh anisotropy is evident around the shocks on the final designs (will be shown in the Mach contours later). However, Hessian based adaptation has more refinements around the shocks than the MOSS adapted mesh. On the other hand, MOESS mesh features high anisotropy along the stagnation streamline, where only little isotropic refinement is allocated in Hessian adapted ones. This type of distinctions are expected and are common in practice, as the stagnation streamline is more of an adjoint solution feature, which is not often presented in the primal solutions and thus cannot be effectively detected by the Mach number Hessian. As mentioned in Chapter 3, the accuracy of the output is determined by the accuracy of both the primal and adjoint solutions. Consequently, MOESS in general achieves a better accuracy than Hessian-based adaptation with the same cost by resolving both the primal and adjoint features. This accuracy benefit will be discussed more later.

The objective convergence and the mesh evolution in the optimization are shown in Figure 6.10. As each optimization converges to the optimum with very similar number of iterations, we plot the objective verses the aggregated total DOF instead of the iterations to separate the convergence curves. The aggregated DOF in each optimization are

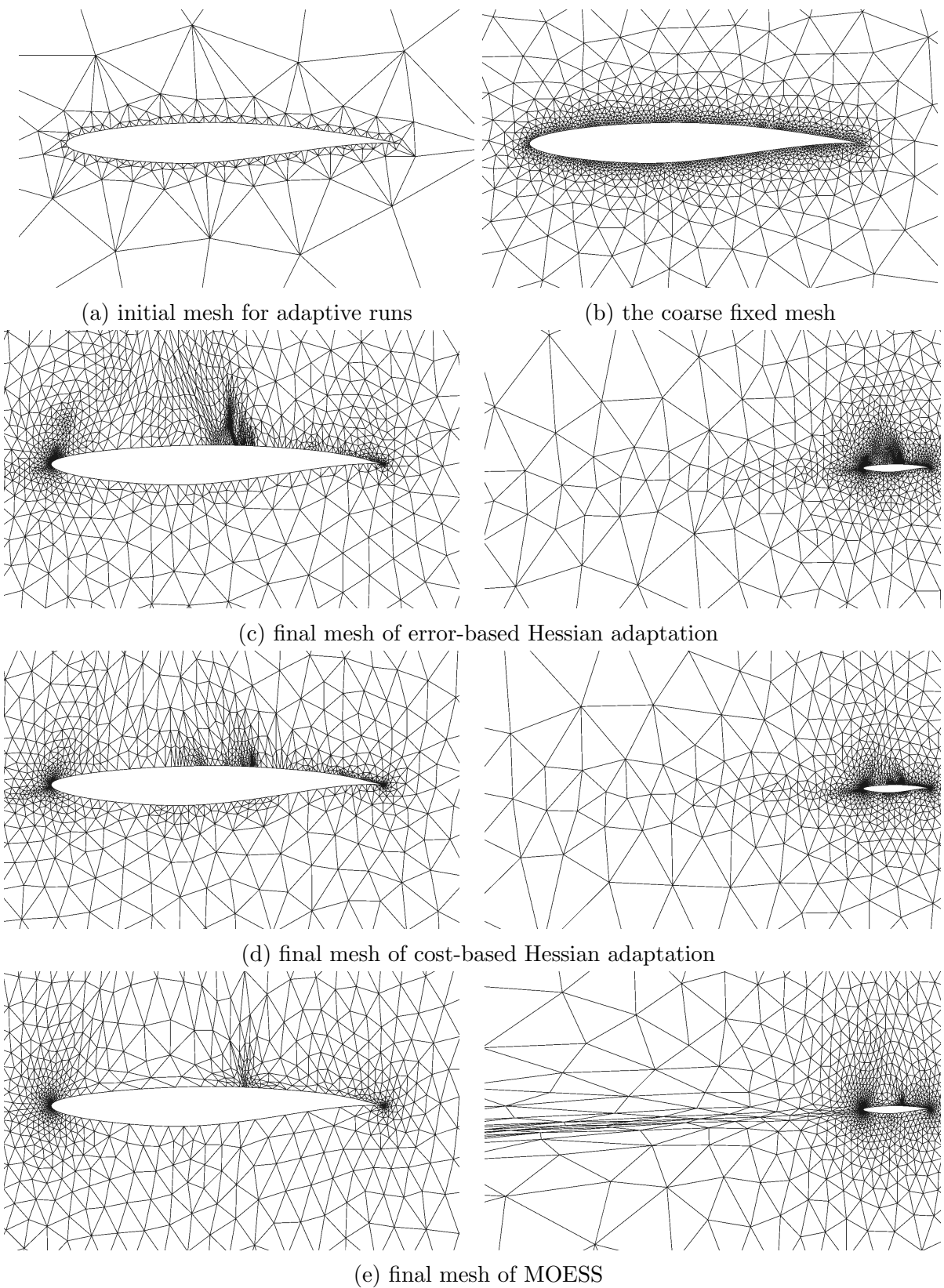


Figure 6.9: Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: meshes for multifidelity and fixed-fidelity optimizations. Only the final adapted meshes for the multifidelity optimization are shown, with a zoom-in plot on the left and a zoom-out one on the right.

accumulated at each major iteration with the current mesh DOF, *i.e.*, once a new design is accepted, the aggregated DOF get updated by adding the current mesh DOF. We can see in Figure 6.10a that the estimated discretization error of the objective is always above the optimization tolerance in the fixed-fidelity (fixed-mesh) optimization. At the end of the optimization, the objective errors are much larger than the objective changes between different designs. As a result, the optimization accuracy (accuracy of objective on the optimized design) is dominated by the discretization error due to the imbalance between the discretization error and the optimization tolerance. In these scenarios, the optimizer may work on the numerical error instead of the physics to minimize the drag and thus converges to undesired designs. In contrast, the discretization error is always controlled to be below the optimization tolerance in the error-based optimization approach, or the optimization tolerance is adjusted to be equal to the discretization error in the cost-based methods. On the other hand, the multifidelity optimizations built on the adapted meshes all require much less iterations on the highest fidelity thanks to the better starting design obtained from the lower-fidelity optimization.

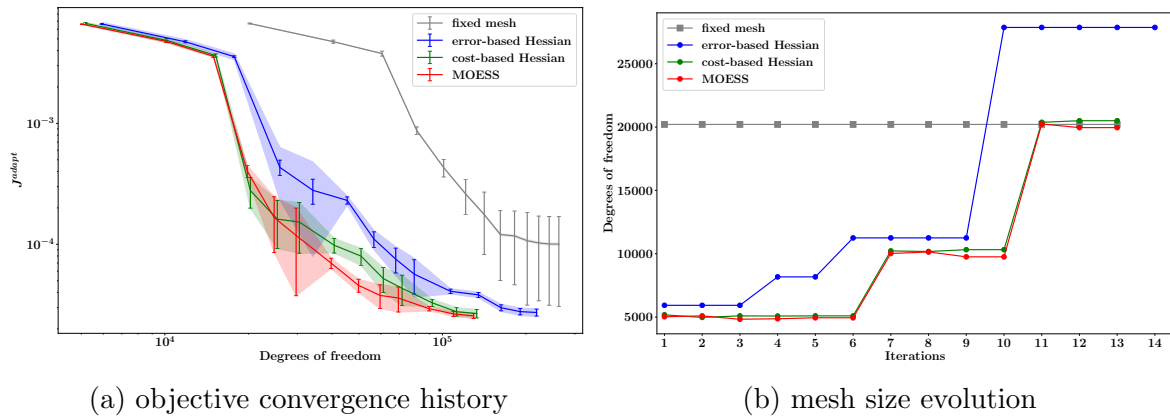
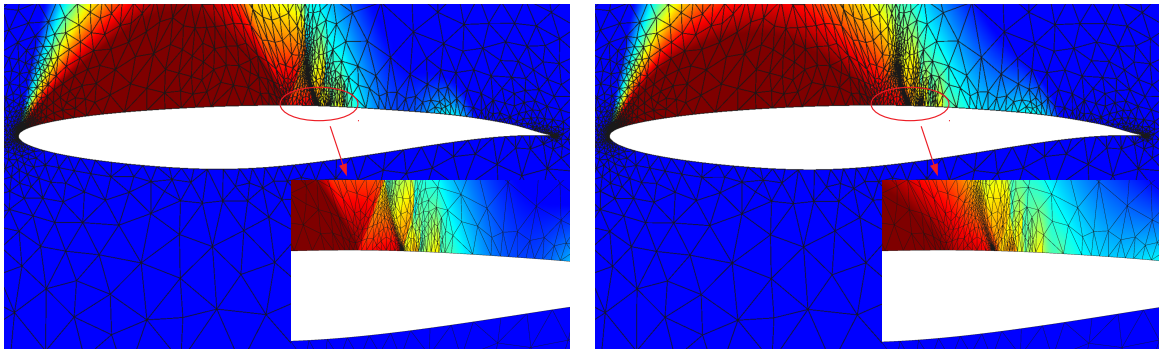


Figure 6.10: Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: objective convergence history and mesh size evolution for different methods. In the objective convergence plot, the error bars denote the objective error estimates while the shaded area represent the optimization tolerance. In the mesh evolution plot, for the cost-based optimization the mesh cost at each fidelity can be easily read; while the error-based optimization has the first 5 iterations in the lowest fidelity, 4 iterations in the middle fidelity and 5 more iterations on the highest fidelity.

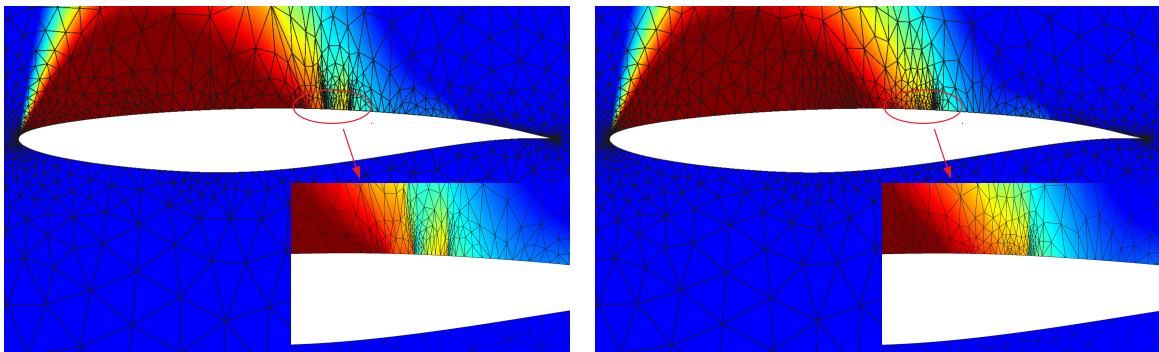
In the mesh evolution plot shown in Figure 6.10b, we observe the differences between various adaptation mechanics. For the error-based optimization approach, the error is controlled to be below the optimization tolerance at each fidelity. Therefore, in the same fidelity, the mesh can be refined if the objective error is above the optimization tolerance on a new design, otherwise the mesh stays fixed. For example, in the optimization with

error-based Hessian adaption (blue line), the first 3 designs are obtained on the same mesh as the objective errors are below the optimization tolerance, while at the 4th design the mesh is refined since the error increases on the new design and is above the optimization tolerance; however, the optimization fidelity stays the same until it converges at the 5th design. On the other hand, the cost-based approaches, both the cost-based Hessian adaptation and MOESS, keep adapting the meshes with the fixed cost at each fidelity. As we can see, the error-based approaches and the cost-based approaches have very similar mesh sizes at the low fidelities; nevertheless, the error-based approach has much higher DOF on the highest fidelity. There are several reasons for this difference. First of all, the cost-based approaches frequently adapt the mesh by redistributing the mesh resolution, which in general results in a higher accuracy at low fidelities compared to the error-based adaptation and thus better designs. As a result, the starting design on the highest fidelity for cost-based approaches has weaker shocks and hence requires lower DOF to achieve similar accuracy compared to the error-based ones. Furthermore, as the error-based adaptation keeps the mesh fixed as long as the error is still below the optimization, the error-based adaptation tends to over-refine areas that are important for some intermediate designs but not necessary for the final design. On the other hand, cost-based adaptation avoids the over-refinement although it requires several adaptive iterations with the fixed DOF at each major design iteration. Some meshes on the highest fidelity for different adaptation methods are shown in Figure 6.11. We can see that the meshes for the starting designs on the highest fidelity for different adaptation mechanics all have mesh elements well-aligned to the weak shocks on the upper surface since all of the them are adapted on the current design. However, for later designs, the error-based adaptation keeps the mesh fixed since the objective error still meets the optimization tolerance. As a result, the refinement around the original shock persists even though it is almost eliminated on later designs. Inefficiency arises as some of the approximation capacity of these degrees of freedom is lost. In contrast, the cost-based approaches actively redistribute the mesh elements to always have the refinements aligned to the shocks even when their location and strength change during the optimization as shown in Figure 6.11. Therefore, cost-based adaptation utilizes the mesh DOF more effectively and the benefits can be more significant in problems with dramatic design changes or with complex physics.

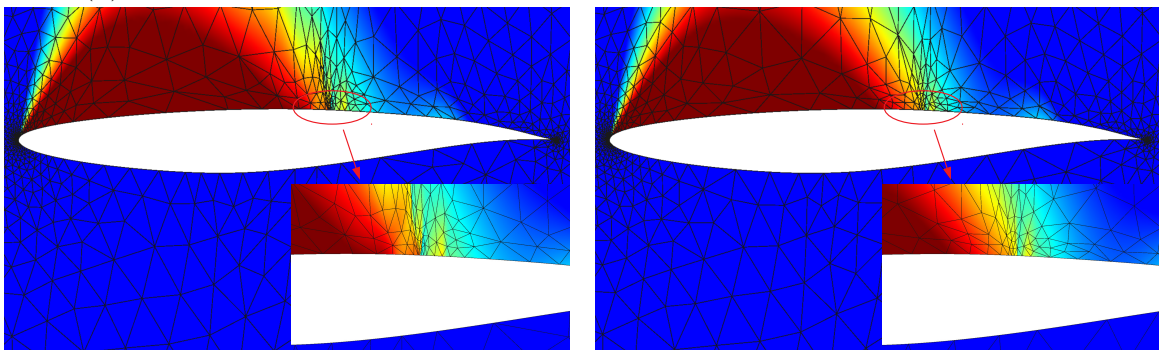
The accuracy benefits of the proposed methods are summarized in Table 6.4. Due to high discretization errors, the optimization using the fixed mesh converges to a design that has much higher drag compared to the designs obtained using adapted meshes. On the other hand, the optimizations using adapted meshes converge to similar designs with close drag values. Furthermore, the objective error estimates on the final designs are fairly



(a) error-based Hessian adaptation, the 10th and 14th designs from left to right



(b) cost-based Hessian adaptation, the 11th and 12th designs from left to right



(c) MOESS, the 11th and 13th designs from left to right

Figure 6.11: Mach contours on the adapted meshes from different adaptation methods. The color limit is clipped to $[0.9, 1.1]$ to show the shocks.

accurate, assuring the effectiveness of using error estimation to control the optimization. The final designs are compared in Figure 6.12, in which we can see that the optimized design on the fixed mesh is noticeably different from the designs on adapted meshes, which are close to each other despite small differences. The design differences are also reflected in the Mach number contours and the pressure distributions, where the fixed-mesh design features stronger shocks compared to other designs. Note that all of the final designs possess complex shock structures as shown in the Mach contours, which are obtained using much finer adapted meshes. On the one hand, the shocks are present even on the original working meshes used in the optimization, both the fixed and adapted ones. We can tighten the tolerance to possibly achieve “shock-free” designs on the original working meshes by ignoring the imbalance with respect to the error estimates, however, the discretization errors will dominate the design update. On the other hand, the shocks may appear on the finer adapted meshes even if the design does not produce a shock on the original working mesh. In a word, in an optimization tightly coupled with error estimation and mesh adaptation, “shock-free” designs are not required and not necessary given a relatively high error level, *i.e.*, loose optimization tolerance.

In addition to accuracy benefits, computational efficiency gains of the proposed methods compared to the fixed-mesh optimization are shown in Table 6.5. As expected, multifidelity optimizations with adapted meshes converge much faster on the highest fidelity with better designs from the lower fidelity. Although the error-based adaptation yields a mesh that is even finer than the fixed mesh we used, the computational time is around 25% less compared to the fixed-mesh optimization. Due to lower efficiency with the error-based approach as mentioned earlier, the computational costs are higher than the cost-based approaches. On the other hand, both using a cost-based approach, MOESS achieves slightly higher accuracy on the objective calculations as shown in both Table 6.4 and Table 6.5 by better resolving the adjoint features such as stagnation streamlines. Slightly lower computational time of the optimization using MOESS over cost-based Hessian is observed, we attribute it to faster iterative convergence in the flow solver since the mesh anisotropy is better aligned to both primal and adjoint features.

Table 6.4: Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: optimization results summary on different meshes.

	Final mesh DOF	J_m^{adapt}	J_m^{adapt} (“true”)
Fixed mesh	20208	$1.003 \times 10^{-4} \pm 6.981 \times 10^{-5}$	3.178×10^{-5}
Error-based Hessian	27864	$2.730 \times 10^{-5} \pm 1.797 \times 10^{-6}$	2.586×10^{-5}
Cost-based Hessian	20502	$2.679 \times 10^{-5} \pm 2.154 \times 10^{-6}$	2.484×10^{-5}
MOESS	19956	$2.556 \times 10^{-5} \pm 1.220 \times 10^{-6}$	2.436×10^{-5}

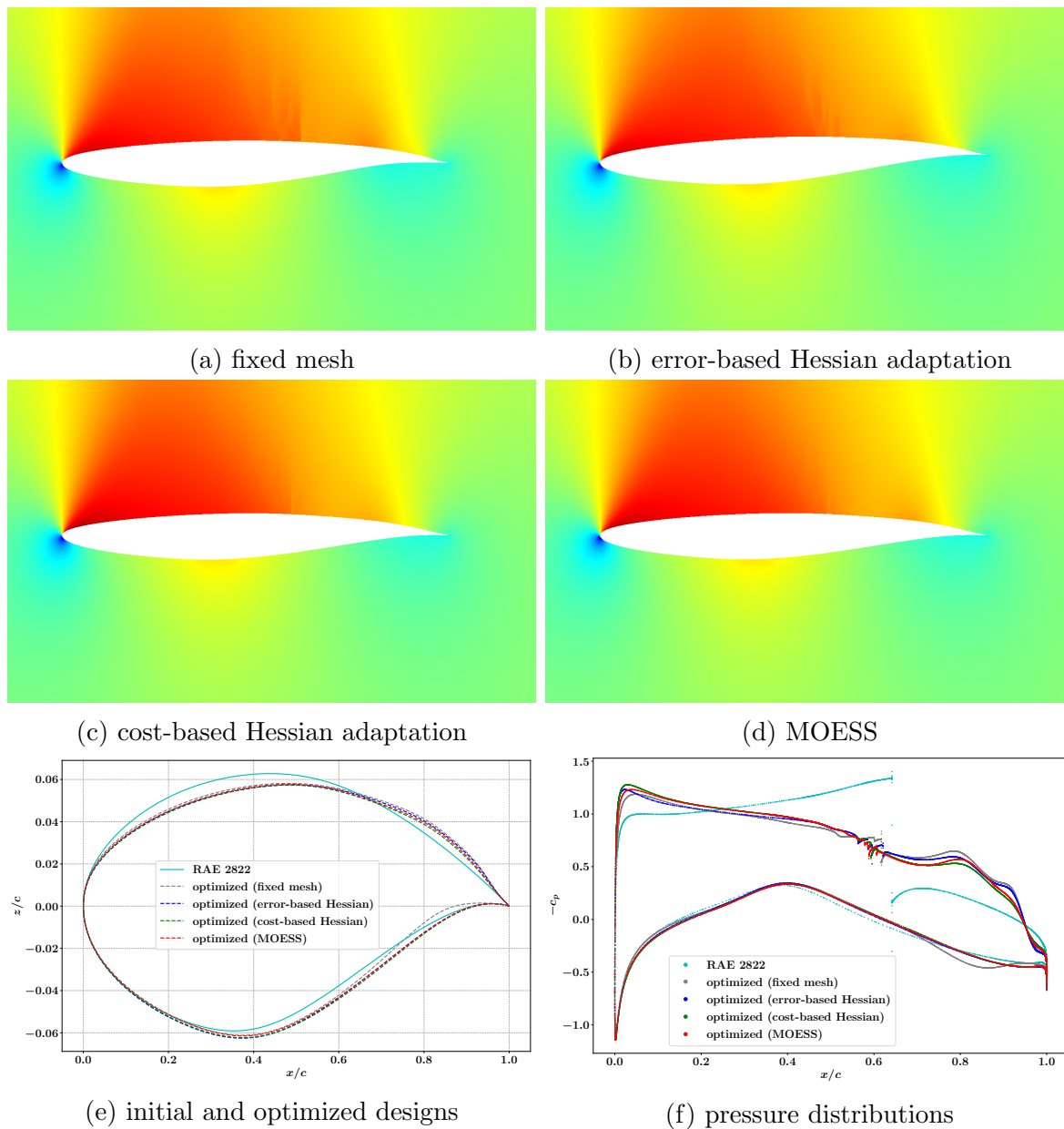


Figure 6.12: Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: final design comparison. The Mach contours (range is 0.0 ~ 1.35) and the pressure distributions are obtained on much finer adapted meshes.

Table 6.5: Revisit of the inviscid transonic optimization on the RAE 2822 airfoil: computational cost comparison. In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity. The computational time results are obtained on the same HPC cluster (3.0 GHz Intel Xeon Gold 6154) using parallel runs with 16 cores and 16GB RAM.

Optimization methods	Optimization level	Optimization tol (Drag count)	CPU time (s)
Fixed-fidelity (fixed mesh)	L3	0.020	4.014×10^3
	L1	2.000	5.287×10^2
Multifidelity (error-based Hessian)	L2	0.200	4.271×10^2
	L3	0.020	2.101×10^3
Multifidelity (cost-based Hessian)	L1	≈ 0.710	3.814×10^2
	L2	$\delta J^{\text{adapt}} \approx 0.121$	6.408×10^2
	L3	≈ 0.022	1.237×10^3
Multifidelity (MOESS)	L1	≈ 0.808	4.200×10^2
	L2	$\delta J^{\text{adapt}} \approx 0.084$	3.547×10^2
	L3	≈ 0.012	7.893×10^2

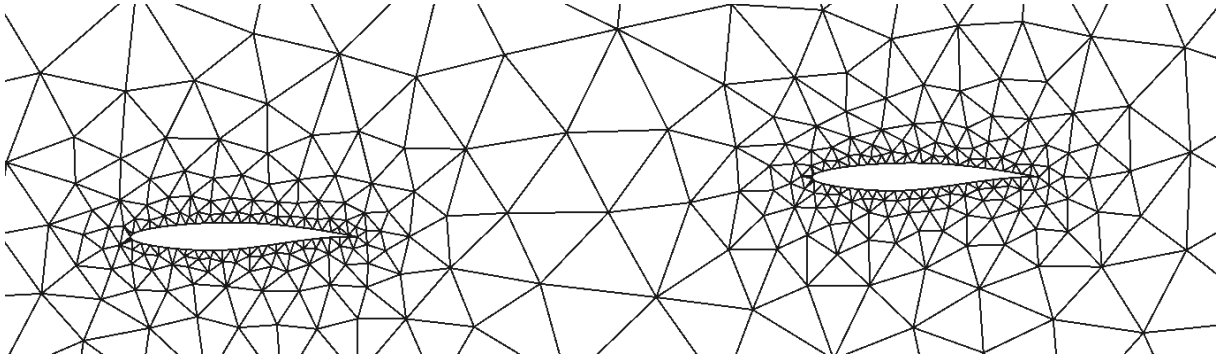
6.3.3 Tandem RAE 2822 Airfoils

As seen in the problems considered above, the proposed methods improve both the efficiency and the accuracy by a multifidelity framework with active discretization error control. In terms of the adaptation and optimization integration, the cost-based approach outperforms the error-based one by more frequent adaptation and less over-refinement on undesired designs. In this section, we apply the cost-based approach to a more complicated case which involves two airfoils close by. Although less common and not well-studied as the single airfoil optimization, this case is representative of the design of some unconventional configurations such as the strut-braced wing and the joined-wing aircraft shown in Figure 1.1.

We again use the RAE 2822 airfoil, and the configuration setup is shown in Figure 6.13a. The two RAE 2822 airfoils are placed two chords away and the height difference is around a quarter chord. The freestream flow condition is the same as the single RAE 2822 airfoil optimization considered before. Figure 6.13a shows a very coarse hand-generated mesh around the two airfoils, where the farfield is a square that is 100 chords away from the airfoils. Figure 6.13b shows the Mach number contour and Figure 6.13c depicts the conservation of the x -momentum component for the drag adjoint. We can see in the Mach contour that a strong shock is present on the first airfoil upper surface, while the second airfoil features two shocks: one weak shock close to the leading edge and another stronger one around the middle chord. A significant impact of the first airfoil wake on the second airfoil is also observed in the Mach number contour. On the other hand, the adjoint field features λ -like adjoint “shock” structures over the two airfoils, as well as a high variation along the leading-edge stagnation streamline from the second airfoil. The complicated primal and adjoint fields pose challenges for mesh generation. Due to limited knowledge about the flow field *a priori*, much effort may be required to obtain a high-quality mesh for this problem. Enough resolution is needed near both of the airfoil boundaries, and the area between them may also require specific refinement. Extra refinement for shocks is in general impossible *a priori* and not necessary either since the shock strength and location will vary significantly during the optimization. In contrast to the challenging meshing task in traditional optimization, a fairly coarse mesh such as the one in Figure 6.13a can be used to start the optimization if adaptive CFD is used.

For simplicity, we only consider the shape optimization of the second airfoil while keeping the first airfoil shape fixed. The optimization seeks the optimal shape and the angle of attack ⁹ to minimize the total drag of the two airfoils, *i.e.*, $J^{\text{adapt}} = c_d = c_{d,1} + c_{d,2}$,

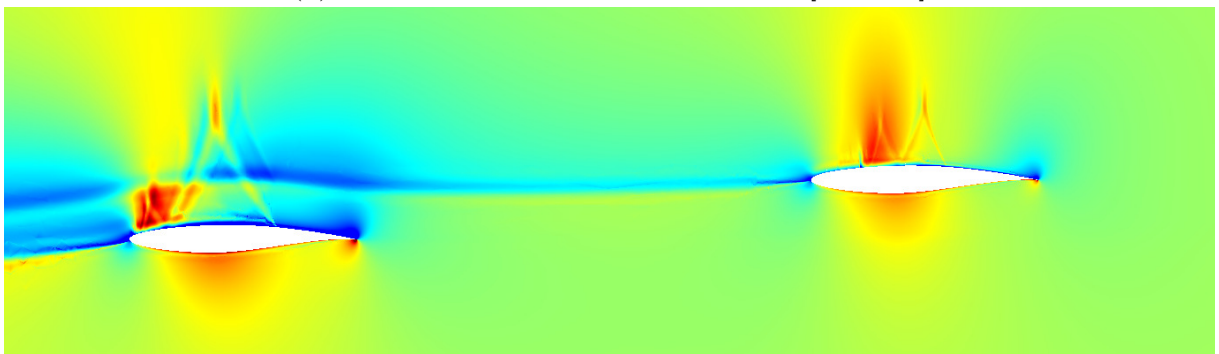
⁹No relative angle between the two airfoils.



(a) initial coarse mesh



(b) Mach number contour, color limit is $[0.3, 1.35]$



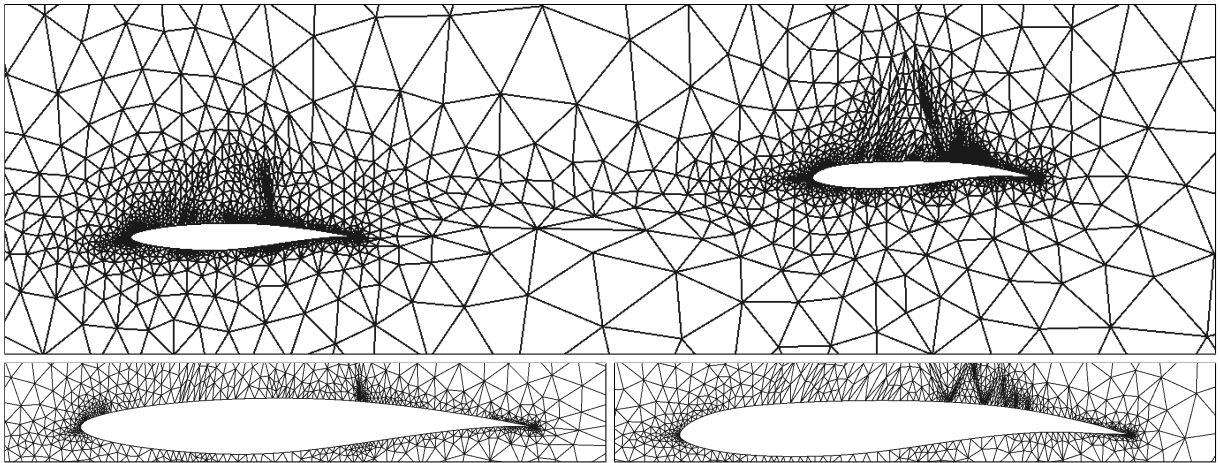
(c) x -moment component of the drag adjoint, color limit is $[-1.0, 0.8]$

Figure 6.13: Tandem RAE 2822 airfoils: initial configuration and the corresponding primal and adjoint fields.

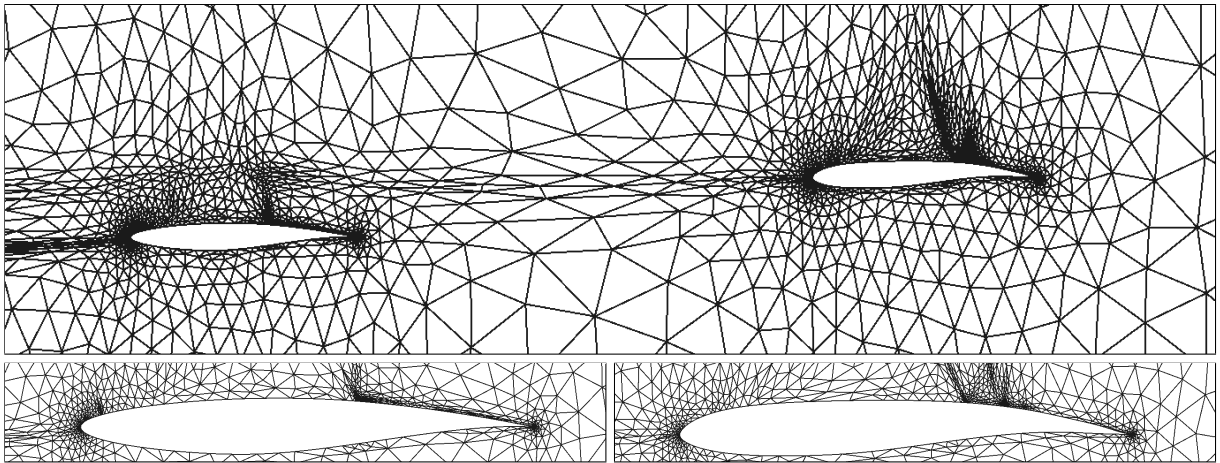
where $c_{d,1}$ and $c_{d,2}$ denote the drag coefficient of the first and second airfoils, respectively. The trim constraint is the total lift of the two airfoils, *i.e.*, $J^{\text{trim}} = c_\ell = c_{\ell,1} + c_{\ell,2}$, whose target value is set to be 1.5. The airfoil shape is parameterized with 16 Hicks-Henne basis functions and the airfoil area is again restricted to be no smaller than the original value.

We apply the cost-based optimization approach in this problem, starting with the coarse initial mesh shown in Figure 6.13a. Both the cost-based Hessian adaptation and MOESS are considered here. The final adapted meshes are compared in Figure 6.14. Both the Hessian adaptation and MOESS refine the shocks on the two airfoils with anisotropic elements, while the MOESS mesh features more anisotropic resolution of both the stagnation streamlines and the post shock locations. Hessian adaptation has also targeted these areas, however, with nearly isotropic elements, since these anisotropy is tied to the adjoint field which can not be obtained directly from the Mach number Hessian. As discussed earlier in Chapter 3, the accuracy of the output depends on the accuracy of both the primal and the adjoint solutions, MOESS has thus a better accuracy in this case. This accuracy gain is also reflected in the objective convergence and mesh evolution plots shown in Figure 6.15. At the lowest fidelity, the two methods have similar convergence as the objective changes are much larger than the discretization errors. However, at the medium fidelity, only two design iterations are allowed (optimization tolerance set to be equal to the estimated error) on Hessian adapted meshes due to the high discretization errors, while the MOESS has a much smaller error which permits more design updates with reliability. The better accuracy of MOESS is also observed at the highest fidelity, leading to a lower allowable optimization tolerance. However, optimization with MOESS meshes still converge faster on the highest fidelity due to a better design obtained from the medium fidelity. This performance benefit of MOESS can also be observed in the total computational time as listed in Table 6.6, where the optimization with MOESS achieves around 24% time saving compared to the one using Hessian-based adaptation.

The optimized designs obtained from the two methods are compared in Figure 6.16. As the shape of the first airfoil is fixed, we only compare the optimized shapes of the second airfoil in Figure 6.16e. Both methods try to shift up the second airfoil such that the strong down-wash caused by the first airfoil wake is significantly reduced, which can be observed by comparing the final Mach contour plots in Figures 6.16a–6.16b and the initial one in Figure 6.13b. Meanwhile, the optimizations flatten the upper surface to reduce the shock while curve the aft section of the lower surface to maintain the lift trimming condition. Despite the overall agreements on the two optimized shapes, the detailed designs on the upper and lower surfaces are different. The differences are also presented in the Mach number contours and the pressure distributions as shown in Figure 6.16. The



(a) final mesh of Hessian adaptation



(b) final mesh of MOESS

Figure 6.14: Tandem RAE 2822 airfoils: final adapted meshes on the optimized designs.

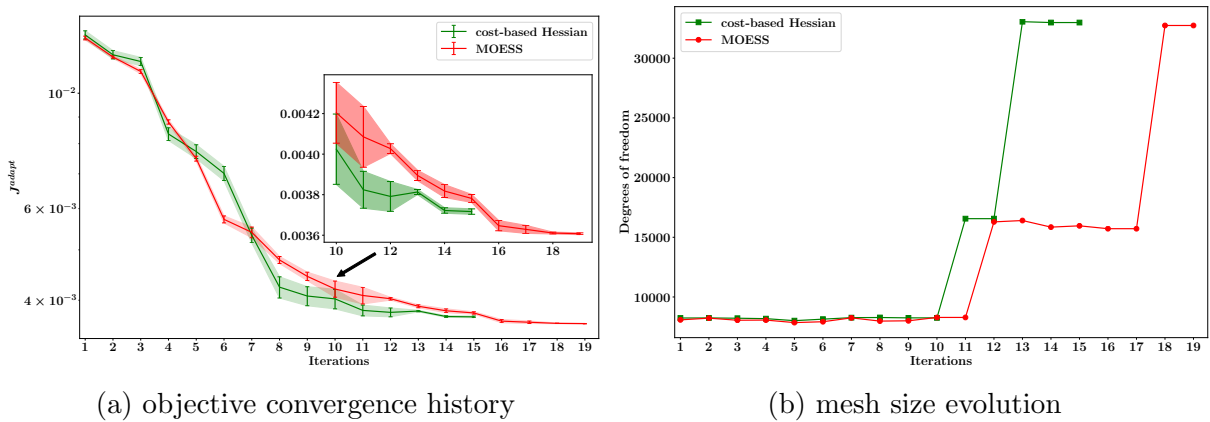


Figure 6.15: Tandem RAE 2822 airfoils: objective convergence and mesh size evolution.

Table 6.6: Tandem RAE 2822 airfoils: computational cost comparison. In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity. Results are obtained on the same HPC cluster (3.0 GHz Intel Xeon Gold 6154) using parallel runs with 36 cores and 36GB RAM.

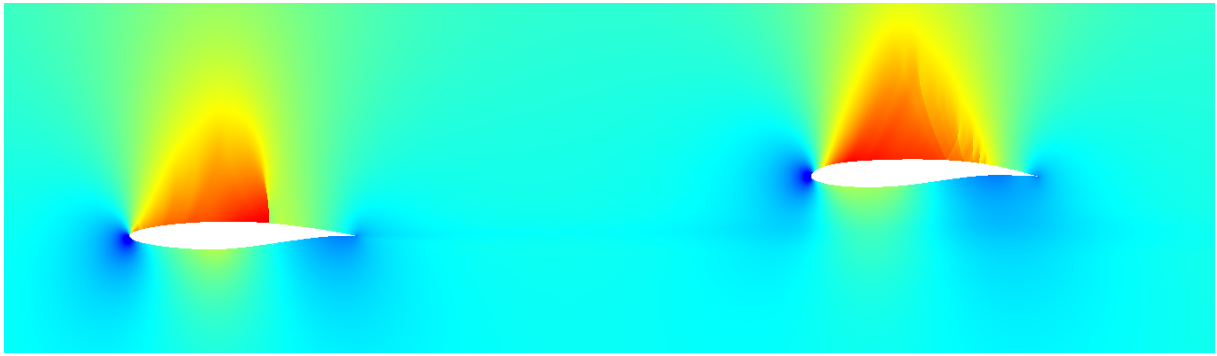
Adaptation methods	Optimization level	Optimization tol (Drag count)	CPU time (s)
cost-based Hessian	L1	≈ 1.733	8.727×10^2
	L2	$\delta J^{\text{adapt}} \approx 0.735$	2.355×10^2
	L3	≈ 0.136	2.686×10^3
MOESS	L1	≈ 1.800	8.645×10^2
	L2	$\delta J^{\text{adapt}} \approx 0.196$	9.329×10^2
	L3	≈ 0.044	1.111×10^3

two optimized designs both feature a complex shock structure on the second airfoil, while the MOESS design has slightly stronger shocks. On the other hand, the normal shock structure is preserved on the first airfoil for both designs but a weaker shock is found on the MOESS design. The final objective values are summarized in Table 6.7, where we can see that the MOESS design has a one count lower drag compared to the Hessian design. Again, the objective error estimates are fairly accurate which can be used to evaluate the accuracy even without running more expensive verification flow simulations on finer discretizations.

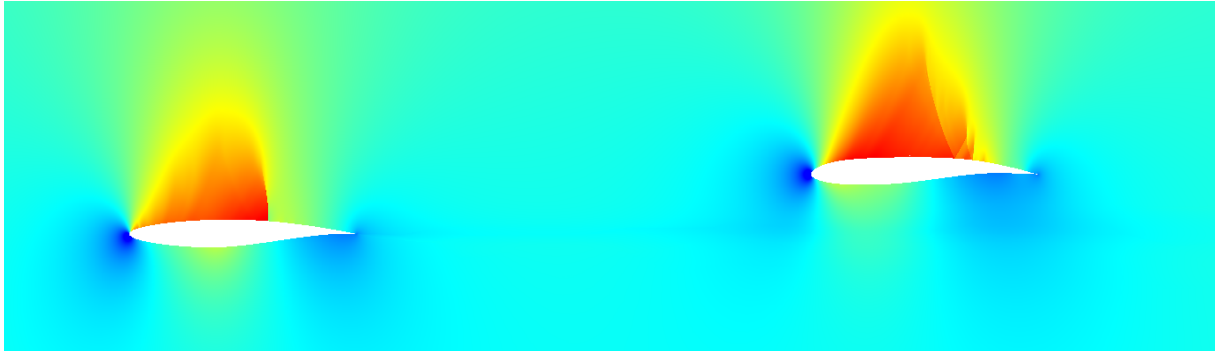
Table 6.7: Tandem RAE 2822 airfoils: optimized objective on different meshes. The “true” objective are obtained using order increment from p to $p + 1$.

	Final mesh DOF	J^{adapt}	J^{adapt} (“true”)
Cost-based Hessian	32988	$3.717 \times 10^{-3} \pm 1.365 \times 10^{-5}$	3.704×10^{-3}
MOESS	32742	$3.607 \times 10^{-3} \pm 4.396 \times 10^{-6}$	3.604×10^{-3}

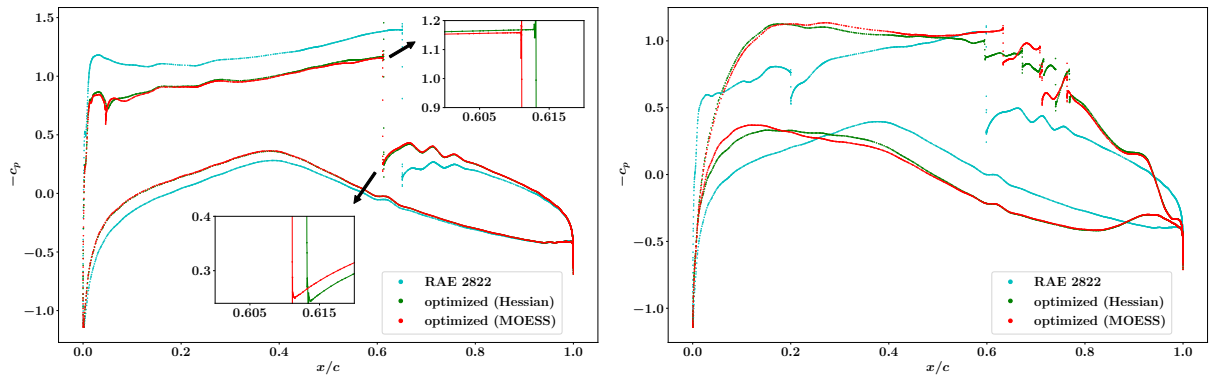
Although some physical intuitions can still be drawn from the Mach number contours and the pressure distributions, it is in general hard to assess the design just by looking at the solutions as the physics becomes more and more complicated. On the other hand, the objective value is used as the main metric for assessing the design quality. Therefore, reliable objective predictions are essential for optimization that involves complicated physics, such as the design for unconventional configurations. We expect the proposed optimization method with error estimation and mesh adaptation to be more beneficial in these problems.



(a) Mach number contour on the optimized design from Hessian adaptation

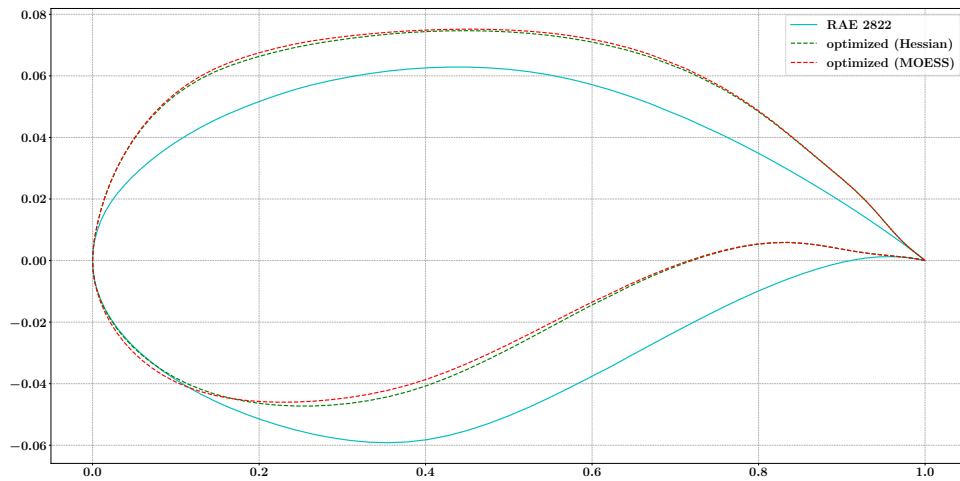


(b) Mach number contour on the optimized design from MOESS



(c) first airfoil pressure distributions

(d) second airfoil pressure distributions



(e) initial and optimized airfoil shapes

Figure 6.16: Tandem RAE 2822 airfoils: comparison between the optimized designs from Hessian adaptation and MOESS. Mach number contours has the same scale, $[0.3, 1.35]$.

6.3.4 ADODG Case 2: Turbulent Transonic Optimization on RAE 2822

After showing success in inviscid transonic problems, we test our methods in a more complicated turbulent case. The problem considered here is a fully-turbulent version of the optimization problem studied in Section 6.3.2, which is proposed by ADODG as a benchmark problem for two-dimensional airfoil optimization. RANS equations with the SA turbulence model introduced in Chapter 2 are used as the flow governing equations. For transonic turbulent simulations, anisotropic resolution is vital for efficiently resolving both the boundary layers and possibly existing shocks. However, due to the linear velocity profile in the viscous sub-layer, adaptations using Hessian-based anisotropy is in general ineffective as the Mach number Hessian is close to zero in these regions. Therefore, only MOESS with the cost-based optimization approach is used here.

With automatic error estimation and mesh adaptation, the starting mesh for the optimization can be fairly coarse even for turbulent simulations. We use the same starting mesh as the one used in Section 6.3.2, which is shown in Figure 6.17 together with the adapted meshes obtained during the optimization. For this turbulent case, the mesh adaptation always prioritizes the boundary layer over other flow features since it is the most influential region for accurate output (both drag and lift) predictions. As a result, many degrees of freedom are put into the boundary layer with anisotropic elements around the airfoil boundary. Only after appropriately resolving the boundary layer, more refinements are added to capture stream-wise features like shocks and the stagnation streamlines.

The mesh size and the objective value are collected at each optimization step as shown in Figure 6.18. In the same fidelity, *i.e.*, mesh cost level, we can see in Figure 6.18a that the objective errors are close even for different designs since the meshes are optimized on each of them. The initial design, *i.e.*, the first optimization iteration, however, has a much higher error than the other designs in the same fidelity mainly due to the much stronger shock presented on the upper surface. Nonetheless, the accuracy is enough for the optimizer to effectively update the initial design. As the shape optimization proceeds, the objective error estimates are more stable with similar mesh DOF as the design approaching optimal. At the highest fidelity, we expect only small changes in the shape, and the objective error remains almost the same, which means that both the shape and the mesh converge to an optimum. The initial and optimized airfoils are compared in Figure 6.19. As depicted in Figure 6.19c, the optimization flattens the upper surface near the forward section, while curves the lower surface and increases the thickness in the aft section. The curvature reduction on the top surface is trying to smooth the flow acceleration region to weaken the shock. The thickened lower surface and aft section are required to maintain the lift and area constraints. This is also reflected in the Mach number contours

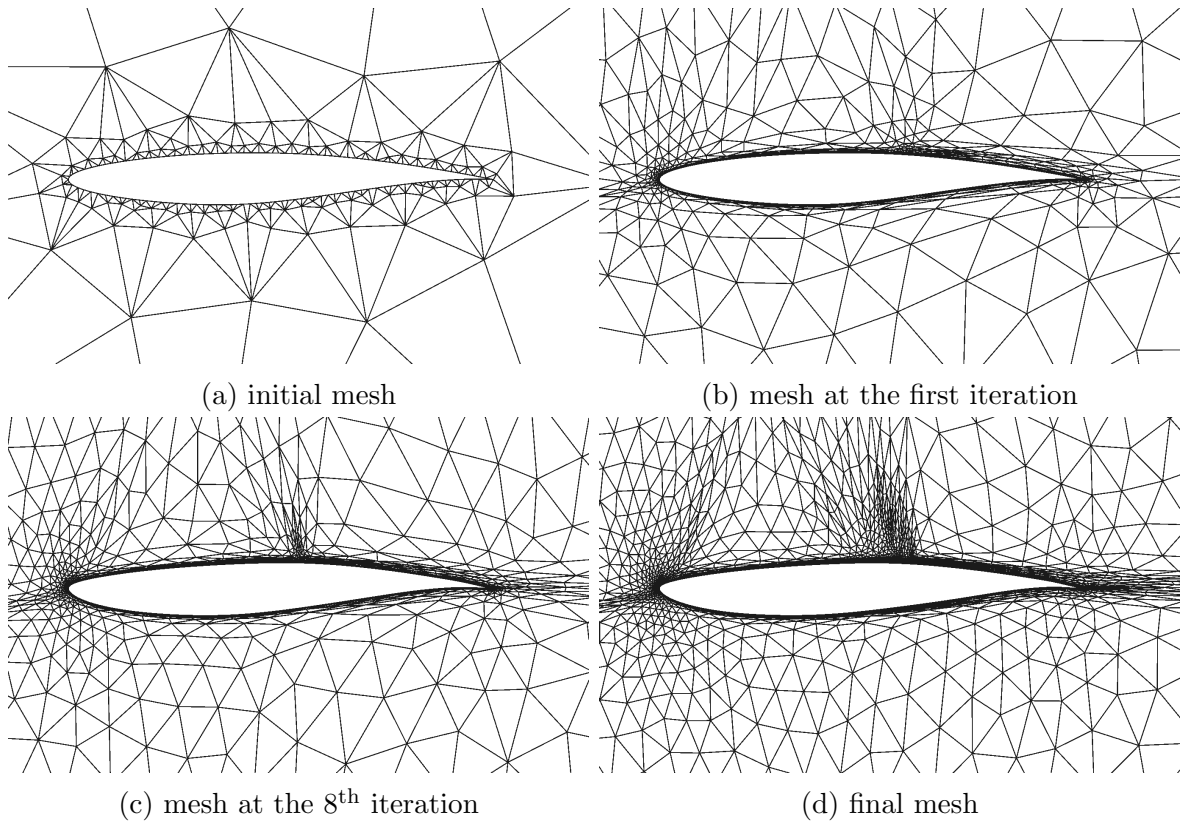


Figure 6.17: ADODG case 2: the initial mesh and adapted meshes in the optimization.

shown in Figures 6.19a–6.19b where the initial strong shock is significantly reduced. As a result, the strong discontinuity presented in the original pressure distribution is absent on the optimized design in Figure 6.19d. Although some very weak shocks on the optimized design can still be observed in both the Mach number contour and the pressure distribution plots, those shocks are insignificant under the current optimization tolerance (tied to the objective accuracy) and further optimization might be dominated by the discretization errors as we have seen in Section 6.2.2. The final design yields a drag coefficient of $c_{d,\text{opt}} = 102.52 \pm 0.0275$ counts, achieving a $45.17\% \pm 0.014\%$ drag reduction compared to the initial RAE 2822 airfoil which produces a drag of $c_{d,0} = 186.97$ counts¹⁰. The overall behavior is similar to the optimization results obtained in inviscid cases shown in Section 6.3.2, however, the detailed optimized shape are very different due to different physical models used. Therefore, in order to achieve a reliable design, both the discretization errors and the modeling errors need to be quantified and controlled. The methods developed in this work can assure the design quality as long as a reliable physical model is chosen during the optimization; on the other hand, they can also be used to help identify the impact of different physical models on the optimizations by well controlling the discretization errors.

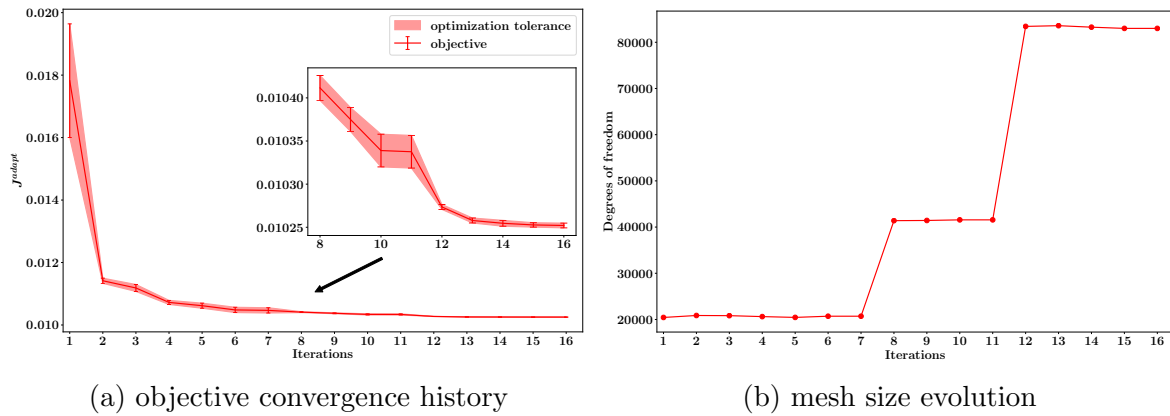
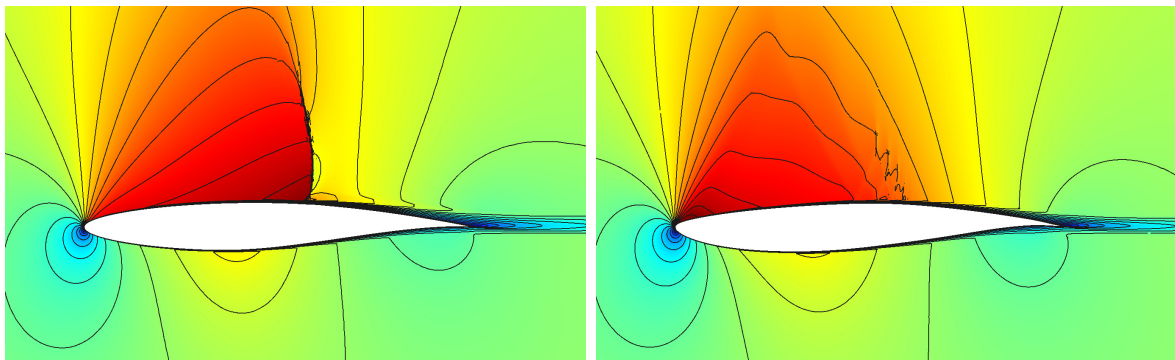


Figure 6.18: ADODG case 2: objective convergence history and mesh size evolution for different methods.

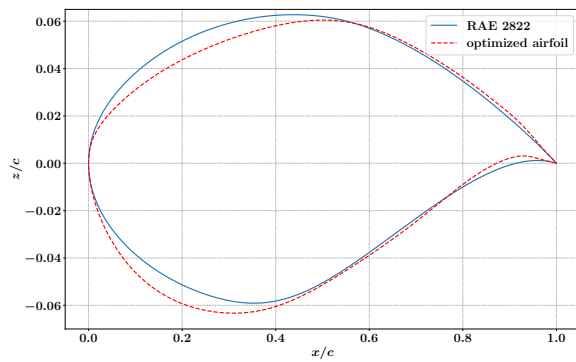
As this testing case is a fairly standard benchmark for viscous airfoil optimization, we also compare our results to those reported by other research groups in Table 6.8. As shown in the table, the drag values of the baseline RAE 2822 airfoil show large variance in different results. The optimized airfoils exhibit even larger difference in the final drag values. There are several reasons for the spread of the drag coefficients. First of all, the

¹⁰The initial drag coefficient is obtained with a much finer optimized mesh on the original RAE 2822 airfoil.

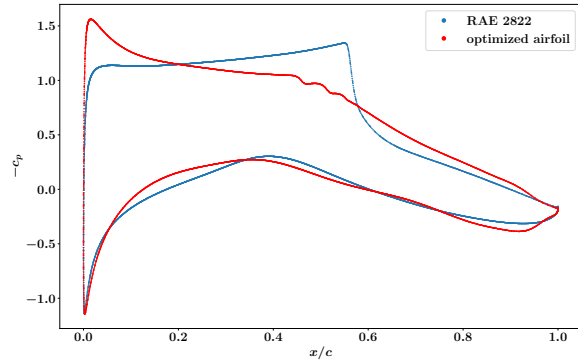


(a) RAE 2822

(b) optimized airfoil



(c) initial and optimized shapes



(d) pressure distributions

Figure 6.19: ADODG case 2: local Mach number (0-1.3) and pressure distributions for the initial and the optimized designs.

physical (turbulence) models and the numerical discretizations (discretization methods and the meshes) are different among these results. Secondly, the geometry parametrization adopted in different optimizations differs, resulting in different design spaces in the first place. Finally, the different mesh deformation methods and the various numerical optimizers also contribute to the drag difference for the same optimization problem ¹¹. By better controlling the discretization errors (difference) in the outputs during the optimization, the author believes the current optimization frameworks with adapted meshes will be important for identifying other dominant components of a standard optimization framework.

Result	Mesh type	Mesh Size	N_s	$c_{d,0}$	$c_{d,opt}$	Δc_d
Present work	Unstructured	13834 [*]	16	186.97	102.52	-84.45
Bisson <i>et al.</i> [184]	Structured	32193 [*]	16	177.80	102.30	-75.50
Carrier <i>et al.</i> [185]	Structured	153300 [†]	10	202.50	111.10	-91.40
Lee <i>et al.</i> [186]	Structured	47824 [†]	34	234.44	131.81	-102.63
Poole <i>et al.</i> [187]	Structured	98304 [*]	10	174.30	93.00	-81.30
Yang and Ronch [188]	Structured	24576 [*]	20	241.24	151.35	-89.89
He <i>et al.</i> [189]	Structured	131072 [*]	40	194.40	108.91	-85.49

Table 6.8: Comparison of the drag coefficients (in drag counts) for RANS optimizations of the RAE 2822 airfoil. The mesh sizes are reported in the number of elements (with superscript ^{*}) or the number of nodes (with superscript [†]) in different works.

6.4 Summary

In this chapter, we study the effects of discretization errors in optimization problems. The optimization process strongly depends on the objective output accuracy and the prescribed optimization tolerance. If not well-balanced, the design update will either be severely polluted by the discretization error or be largely limited by the optimization tolerance, leading to accuracy or efficiency losses in the optimization. In most aerodynamic optimizations, we use tight optimization tolerances without good control of the discretization error during the optimization. As a result, the optimizer may arrive at a sub-optimal design or even at an incorrect spurious optimum with inaccurate information provided by the flow solver and gradient analysis, as shown in the test cases. Thus, discretization error in the objective should be carefully controlled to ensure convergence to the “true” optimal design at a prescribed optimization tolerance. Or alternatively, the optimization

¹¹The problem formulations are actually different as some groups handle the constraints through penalties while others use Lagrange multipliers.

tolerance should be set according to the objective error level to terminate the optimization appropriately, avoiding unreliable design updates.

In order to achieve better accuracy and efficiency, we actively balance the discretization error and the optimization tolerance, by either controlling the error to be below the prescribed optimization tolerance, or adjusting the optimization tolerance according to the objective error. Adjoint-based output error estimation and mesh adaptation are thus introduced to provide a robust way of quantifying and controlling the objective errors. With the variable fidelity offered by adapted meshes, multifidelity optimization frameworks are proposed to efficiently integrate the error estimation and mesh adaptation with traditional gradient-based optimization algorithms, using either an error-based or a cost-based approach. The cost-based approach is shown to be more efficient at a similar accuracy level. By more frequent adaptation compared to the error-based approach, over-refinements are avoided in areas that are important for intermediate designs but not the final design. In terms of adaptation mechanics, MOESS has shown better accuracy and efficiency over Hessian-based adaptation, especially for problems featuring strong anisotropy for both the primal and adjoint solutions. This benefit can become more significant when higher fidelity is required, or when highly anisotropic physics governs the system.

With more judicious considerations of the objective functions and constraints, and additional design parameters, the new methods can provide realistic configurations in practical design scenarios. The fidelity increase is presently driven by an optimization tolerance or a computational cost that decreases or increases by a fixed factor each time, or has to be specified by the user. However, for more practical problems, without *a priori* knowledge of the objective convergence, an improved and automated fidelity increase strategy should be developed to fulfill the potential of the present optimization frameworks to increase the accuracy and efficiency for aerodynamic optimization problems. Furthermore, only mesh adaptation (*h*-adaptation) is considered here to control the discretization error. More efficient adaptation mechanics such as approximation order increment (*p*-adaptation), and combinations (*hp*-adaptation) can also be applied to the proposed methods in the future.

CHAPTER 7

Extension to Multipoint Aerodynamic Optimization

Ideally, an aerodynamic design is expected to retain favorable performance over a wide range of operating conditions. Optimization at one specific cruise condition as shown in Chapter 6 can lead to mediocre performance on the overall mission profile, and/or poor performance at off-design conditions. Therefore, aerodynamic optimization in practice must take into account various flight conditions in both the objective and the constraints [35, 190, 191, 192, 44], making the traditional setup for high-fidelity aerodynamic optimization even more challenging. In order to achieve high accuracy for every design point (flight condition), the single mesh used for all points has to be able to capture all of the important flow features over a wide range of operating conditions. The designer, either expert in meshing or not, cannot reliably generate a mesh appropriate for all cruise conditions. This situation can be even worse when the geometry is complex or as the number of design points increases. Also, the resulting mesh may be quite fine, making the computational cost needed for high-fidelity multipoint aerodynamic optimization prohibitive in practice. On the other hand, numerical errors, if not quantified or well-controlled in multipoint optimization, will be even more dangerous than in single-point ones, as the output errors at different design conditions might accumulate during the optimization. In order to aid practical multipoint aerodynamic design and to reduce the risk of converging to incorrect optima, we extend the framework proposed in Chapter 5 to multipoint optimizations.

7.1 Multipoint Optimization Problem

7.1.1 Weighted-Sum Approach

Consider a multipoint optimization problem involving N_m design points, *i.e.*, flight conditions, which are typically specified by different Mach numbers. To combine the

objective outputs at each design, we use the weighted sum method in this work,

$$J_m^{\text{adapt}} = \sum_{i=1}^{N_m} J_{m,i}^{\text{adapt}} = \sum_{i=1}^{N_m} \omega_i J_i^{\text{adapt}}(\mathbf{U}_i, \mathbf{x}) \quad (7.1)$$

where J_m^{adapt} is the scalar composite objective used in the optimization, which is a sum of the weighted objective component $J_{m,i}^{\text{adapt}}$ at each design point. $J_{m,i}^{\text{adapt}}$ is defined as the objective at i^{th} design point $J_i^{\text{adapt}}(\mathbf{U}_i, \mathbf{x})$, weighted by ω_i that is specified by the user or from the quadrature rules. The weighted objective at each design point $J_{m,i}^{\text{adapt}}$ depends on the design \mathbf{x} shared by all the design points and the individual flow state solutions $\mathbf{U}_i \in \mathbb{R}^{N_{f,i}}$.

The trim conditions at each design point can be the same or can be specified separately, such that

$$\mathbf{R}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}) = \mathbf{J}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}) - \bar{\mathbf{J}}_i^{\text{trim}} = \mathbf{0}, \quad (7.2)$$

where $\mathbf{J}_i^{\text{trim}} \in \mathbb{R}^{N_{t,i}}$ and $\bar{\mathbf{J}}_i^{\text{trim}} \in \mathbb{R}^{N_{t,i}}$ are the trim outputs and their target values at i^{th} design point, respectively. The governing PDE constraints can be written as

$$\mathbf{R}_i(\mathbf{U}_i, \mathbf{x}) = \mathbf{0}, \quad (7.3)$$

in which the states at each design point are independent, but are coupled through the shared design variables \mathbf{x} .

7.1.2 Adjoint and Design Equations

The Lagrangian function that augments the flow equations with trim constraints can be written as

$$\mathcal{L} = \sum_{i=1}^{N_m} \omega_i J_i^{\text{adapt}}(\mathbf{U}_i, \mathbf{x}) + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \mathbf{R}_i(\mathbf{U}_i, \mathbf{x}) + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \mathbf{R}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}), \quad (7.4)$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}^{N_{f,i}}$ and $\boldsymbol{\mu}_i \in \mathbb{R}^{N_{t,i}}$ are the Lagrange multipliers associated with the flow equations and the trim constraints, respectively. The first-order optimality conditions are

obtained by setting the partial derivatives of \mathcal{L} to zero,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}} = \mathbf{0}, \quad (7.5)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_i} = \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} + \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} + \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i} = \mathbf{0}, \quad i = 1, \dots, N_m; \quad (7.6)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}_i} = \mathbf{R}_i(\mathbf{U}_i, \mathbf{x}) = \mathbf{0}, \quad i = 1, \dots, N_m; \quad (7.7)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_i} = \mathbf{R}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}) = \mathbf{0}, \quad i = 1, \dots, N_m. \quad (7.8)$$

As we solve the flow equations for a given design \mathbf{x} each time at every design point, Eqn. 7.7 is always satisfied during the optimization. Again, the trim constraints can be handled either the by the flow solver in the SBT approach or by the optimizer in the OBT approach; for simplicity, we only discuss the former method here. A set of design variables are dedicated to satisfying the trim constraints, denoted as trim variables \mathbf{x}_t , $\dim(\mathbf{x}_t) = \dim(\mathbf{J}^{\text{trim}}) = N_t$, where \mathbf{J}^{trim} is a vector concatenated with the trim outputs at different flight conditions, such that $N_t = \sum_{i=1}^{N_m} N_{t,i}$. Since now Eqn. 7.8 is satisfied by the variation of the trim variables, Eqn. 7.5 breaks down into,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} = \mathbf{0}, \quad (7.9)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_t} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_t} = \mathbf{0}, \quad (7.10)$$

where \mathbf{x}_s is the set of active design parameters in the optimization. We can now choose $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ at each point such that Eqn. 7.6 and Eqn. 7.10 are enforced after each flow solve,

$$\begin{aligned} \boldsymbol{\lambda}_i^T &= - \left(\omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} + \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i} \right) \left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} \right)^{-1} = (\omega_i \boldsymbol{\Psi}_i^{\text{adapt}} + \boldsymbol{\Psi}_i^{\text{trim}} \boldsymbol{\mu}_i)^T, \\ \boldsymbol{\mu}^T &= [\boldsymbol{\mu}_1^T, \dots, \boldsymbol{\mu}_{N_m}^T] = - \left(\sum_{i=1}^{N_m} \omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_t} \right) \left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} \right)^{-1}. \end{aligned} \quad (7.11)$$

There may exist a set of trim variables which can eliminate the coupling of the trim constraints among different design points (*i.e.*, make $d\mathbf{J}^{\text{trim}}/d\mathbf{x}_t$ diagonal), so that $\boldsymbol{\mu}_i$ only depends on the i^{th} design point. In lift-constrained optimization problems discussed in Chapter 6, angle of attack is such a choice. Eqn. 7.11 defines coupled adjoint variables $\boldsymbol{\lambda}_i$

and $\boldsymbol{\mu}_i$, which can then be used to evaluate the total objective gradients with respect to the active design variables, starting from Eqn. 7.9,

$$\begin{aligned}
\frac{DJ_m^{\text{adapt}}}{D\mathbf{x}_s} &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} \\
&= \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \omega_i (\boldsymbol{\Psi}_i^{\text{adapt}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \left[\frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{trim}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[\frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{adapt}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{trim}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] \\
&= \sum_{i=1}^{N_m} \omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_s} \\
&= \sum_{i=1}^{N_m} \left(\omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_s} + \boldsymbol{\mu}_i^T \frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_s} \right).
\end{aligned} \tag{7.12}$$

Eqn. 7.11 and Eqn. 7.12 follow the notation in Chapter 4: $d(\cdot)/d\mathbf{x}_t$ is the sensitivity measured with respect to the trim variables \mathbf{x}_t while keeping the active design \mathbf{x}_s fixed, *i.e.*, in the trimming subproblem; $d(\cdot)/d\mathbf{x}_s$ represents the sensitivity with respect to the active design variables with fixed trim variables. On the other hand, $DJ_m^{\text{adapt}}/D\mathbf{x}_s$ is the total gradient of the composite objective with respect to the active design \mathbf{x}_s , which drives the optimization.

Now the optimization problem has been reduced to finding an optimal design \mathbf{x}_s that drives to zero the gradients in Eqn. 7.12. However, in a practical calculation, on a finite-dimensional space, discretization errors appear in both the flow equations and the adjoint equations, so that infinite-dimensional optimality cannot be guaranteed even when the finite-dimensional optimality condition is satisfied. The following sections will focus on extending the error estimation and mesh adaptation methods to multipoint optimization problems to control the discretization errors.

7.1.3 Output Error Estimation

Normally, the error estimation is applied only to the output in which we are most interested, *i.e.*, the objective. However, our optimization problem requires simultaneous solutions of the flow and adjoint equations. The numerical error of the trim outputs may indirectly affect the calculation of the objective and hence the optimization result. To take this effect into account, coupled adjoints should be used for error estimates as discussed

earlier in Chapter 4.

Consider a given design \mathbf{x}_s , again, on a coarse space H and a finer space h . The error in the objective comes from both the inexact solution $\mathbf{U}_{H,i}$ and the inexact trim constraints satisfaction (inexact trim variables $\mathbf{x}_{t,H}$). We can estimate the error in the composite objective using the linearization given by Eqn. 7.6 and Eqn. 7.10,

$$\begin{aligned}
\delta J_m^{\text{adapt}} &= \sum_{i=1}^{N_m} \omega_i \left[J_{H,i}^{\text{adapt}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H}) - J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}, \mathbf{x}_{t,h}) \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}, \mathbf{x}_{t,h}) \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[\frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} \delta \mathbf{U}_i + \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_t} \delta \mathbf{x}_t \right] \tag{7.13} \\
&= - \sum_{i=1}^{N_m} \left[\boldsymbol{\lambda}_{h,i}^T \delta \mathbf{R}_{h,i} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{R}_{h,i}^{\text{trim}} \right] \\
&= - \sum_{i=1}^{N_m} \left[\boldsymbol{\lambda}_{h,i}^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) + \boldsymbol{\mu}_{h,i}^T \mathbf{R}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \right].
\end{aligned}$$

For the second term in Eqn. 7.13, we can expand the trim residual as

$$\begin{aligned}
\mathbf{R}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) &= \mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \bar{\mathbf{J}}_i^{\text{trim}} \\
&= [\mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H}) - \bar{\mathbf{J}}_i^{\text{trim}}] + [\mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H})]. \tag{7.14}
\end{aligned}$$

The first term above is automatically driven to zero because of the trimming on the coarse space. For the second term, if the definition of the trim outputs does not depend on the approximation space, then we have $\mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) = \mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H})$. Hence, the second term in Eqn. 7.13 is often negligible, resulting a simpler form of the error estimate for the

composite objective,

$$\begin{aligned}
\delta J_m^{\text{adapt}} &= - \sum_{i=1}^{N_m} \boldsymbol{\lambda}_{h,i}^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \\
&= - \sum_{i=1}^{N_m} (\omega_i \boldsymbol{\Psi}_{h,i}^{\text{adapt}} + \boldsymbol{\Psi}_{h,i}^{\text{trim}} \boldsymbol{\mu}_{h,i})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \\
&= \sum_{i=1}^{N_m} \left[-\omega_i (\boldsymbol{\Psi}_{h,i}^{\text{adapt}})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \boldsymbol{\mu}_{h,i}^T (\boldsymbol{\Psi}_{h,i}^{\text{trim}})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \right] \\
&= \sum_{i=1}^{N_m} (\omega_i \delta J_i^{\text{adapt}} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{J}_i^{\text{trim}}),
\end{aligned} \tag{7.15}$$

where $\delta J_i^{\text{adapt}}$ is the objective error and $\delta \mathbf{J}_i^{\text{trim}}$ stands for the trim output error, using standard adjoint-based error estimates (Eqn. 3.43). The weighting by $\boldsymbol{\mu}_{h,i}$ accounts for the effects of trim output error on the objective calculations. More discussions on the objective error estimation with trim conditions and its implementation can be found in Chapter 4.

7.2 Mesh Adaptation

If we would like to use the same mesh for all of the design points, then Eqn. 7.15 can be directly used to localize the error to each element, which then serves as the indicator for mesh adaptation. However, this can be inefficient when the flow features change significantly at different operating conditions, *e.g.*, from subsonic to supersonic regimes. To achieve certain accuracy in such cases, the mesh should be adapted in the areas important for all of the design points, and hence unnecessary computational effort is added to each flow solve if using a single mesh.

In the present work, we allow different meshes for different design points. The objective error in Eqn. 7.15 is first localized to each design point as $\delta J_{m,i}^{\text{adapt}} = \omega_i \delta J_i^{\text{adapt}} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{J}_i^{\text{trim}}$. Then a common approach for obtaining an error indicator is to take the absolute value of the elemental error contribution. When trim outputs are involved, we do not allow cancellation between objective and trim output error indicators, so that the final error indicator on element e at flight condition i , $\mathcal{E}_{i,e}$, is given by

$$\mathcal{E}_{i,e} = \omega_i |\delta J_{i,e}^{\text{adapt}}| + |\boldsymbol{\mu}_{h,i}^T| |\delta \mathbf{J}_{i,e}^{\text{trim}}| = \omega_i \mathcal{E}_{i,e}^{\text{adapt}} + |\boldsymbol{\mu}_{h,i}^T| \mathcal{E}_{i,e}^{\text{trim}}, \tag{7.16}$$

where $\mathcal{E}_{i,e}^{\text{adapt}} \in \mathbb{R}_{>0}$ and $\mathcal{E}_{i,e}^{\text{trim}} \in \mathbb{R}_{>0}^{N_t}$ are the non-negative error indicators for the objective

output and the trim outputs, respectively.

At each operating condition, the mesh adaptation can be performed as a refinement process to achieve certain accuracy, or as a modification or optimization process at a given cost to improve the accuracy. We will denote the former approach as error-based adaptation while the latter as cost-based adaptation, following the notation in Chapter 5. Although each adaptation strategy has been well-studied for fixed configurations, adaptation involving multiple operating conditions has seldom been investigated. In our implementation, mesh adaptation for multipoint flow simulations is a two-stage process: the desired error/cost is first allocated to each design point; common adaptation techniques are then applied to each design point. The sections below describe the error/cost allocation strategies and the adaptation methods adopted at individual design points.

7.2.1 Error/Cost Allocation for Multipoint Mesh Adaptation

In error-based mesh adaptation, a target error tolerance is specified to drive mesh refinement. Thus as a first step, we would like to determine the error tolerance at each design point. This high-level error split relies on an error convergence model that dictates how the output errors behave with respect to changes in cost, usually measured by the system degrees of freedom (DOF). Here, an *a priori* error-cost model is assumed,

$$|\delta J_{m,i}^{\text{adapt}}| \propto C_i^{-r_i/d}, \quad (7.17)$$

where C_i is the cost at the i^{th} flight condition, r_i is the corresponding output error convergence rate, and d is the spatial dimension. The convergence rate at each design point depends on the approximation order and the smoothness of the problem. We use the ideal super-convergent rate of outputs in an adjoint-consistent setting to prevent too aggressive refinement or coarsening (cost redistribution), though lower convergence rates should be expected for under-resolved meshes.

We follow the idea of equally distributing the error-to-cost ratios [156, 193], *i.e.*, the marginal error reduction per cost increase. This is considered optimal as we can otherwise further reduce the error without adding cost by just reallocating degrees of freedom among different design points. For one adaptive iteration, the change in the error due to cost redistribution is

$$\left| \delta J_{m,i}^{\text{adapt}} \right| = \left| \delta J_{m,i}^{\text{adapt},0} \right| \left(\frac{C_i}{C_i^0} \right)^{-r_i/d}, \quad (7.18)$$

where the superscript 0 indicates values in the unadapted meshes. The marginal error-

to-cost ratio at each design point can be obtained by

$$\begin{aligned}
\frac{\partial |\delta J_{m,i}^{\text{adapt}}|}{\partial C_i} &= -\frac{r_i}{d} \left| \delta J_{m,i}^{\text{adapt},0} \right| \left(\frac{C_i}{C_i^0} \right)^{-r_i/d-1} \frac{1}{C_i^0} \\
&= -\frac{r_i}{d} \left| \delta J_{m,i}^{\text{adapt},0} \right| \left(\frac{C_i}{C_i^0} \right)^{-r_i/d} \frac{1}{C_i} \\
&= -\frac{r_i}{d} \frac{|\delta J_{m,i}^{\text{adapt}}|}{C_i}, \quad i = 1, \dots, N_m.
\end{aligned} \tag{7.19}$$

Further, we define the desired error ratios f_i^δ and cost ratios f_i^C as

$$f_i^\delta = \frac{\delta J_{m,i}^{\text{adapt}}}{\delta J_{m,1}^{\text{adapt}}}, \quad f_i^C = \frac{C_i}{C_1}, \quad i = 1, \dots, N_m. \tag{7.20}$$

By equidistributing the error-to-cost ratios among different design points, the desired error ratios f_i^δ and cost ratios f_i^C can be determined using Eqn. 7.19. In this work, we further assume identical convergence rates ($r_i = r, i = 1, 2, \dots, N_m$) if the same approximation order is used and the meshes are well-adapted. Eqn. 7.19 then implies that the desired error ratios and cost ratios are equal, and can be solved as

$$f_i^\delta = f_i^C = \left| \frac{\delta J_{m,i}^{\text{adapt},0}}{\delta J_{m,1}^{\text{adapt},0}} \right|^{\frac{1}{r/d+1}} \left[\frac{C_i}{C_1} \right]^{\frac{r/d}{r/d+1}}, \quad i = 1, 2, \dots, N_m. \tag{7.21}$$

The desired ratios in Eqn. 7.21 can then guide the error redistribution in error-based multipoint mesh adaptation, or cost redistribution in cost-based adaptation. Ideally, specifying desired cost ratios and error ratios are equivalent if the *a priori* error-cost model in Eqn. 7.17 is perfect, which is not the case in general. Furthermore, an error-based mesh adaptation procedure in practice does not focus on matching exactly the target error, since the *a posteriori* output error may deviate from the *a priori* estimation. Instead, an error tolerance τ_i is specified in the error-based adaptation, and the adaptation (refinement) reduces the *a posteriori* error until it is below the error tolerance, $|\delta J_{m,i}^{\text{adapt}}| \leq \tau_i$. On the other hand, the desired cost can be specified in cost-based adaptation such that the adaptation redistributes the degrees of freedom on the computational mesh to reduce the error while keeping the cost fixed. Therefore, cost-based and error-based adaptation with equal desired ratios result in different meshes even if the same adaptation method is used.

- For cost-based multipoint mesh adaptation, given a fixed total cost C , the desired

cost at each design point is then redistributed as,

$$C_i = \frac{f_i^C}{\sum_{j=1}^{N_m} f_j^C} C. \quad (7.22)$$

At each design point, the mesh adaptation then refines areas more important for output prediction and coarsens elsewhere to keep the cost fixed (within some tolerance).

- In error-based multipoint mesh adaptation, given a total error tolerance τ of the composite objective J_m^{adapt} , we first distribute the tolerance to each design point according to the desired error ratios,

$$\tau_i = \frac{f_i^\delta}{\sum_{j=1}^{N_m} f_j^\delta} \tau. \quad (7.23)$$

At each design point, the mesh is refined until the objective error component is below the error tolerance, $|\delta J_{m,i}^{\text{adapt}}| \leq \tau_i$. Thus the cost at each design point C_i increases as the mesh is adapted, although the local adaptation may support coarsening.

7.2.2 Mesh Adaptation at Individual Design Points

Similar to single point adaptation as discussed in Chapter 5, at each design point, the mesh is adapted with either cost-based or error-based strategies. During the mesh adaptation, the error indicators in Eqn. 7.16 provide information on local refinement or coarsening. Meanwhile, the mesh elements are stretched according to the desired anisotropy, which is either from solution Hessian in Hessian-based adaptation, or from a sampling and optimization procedure in MOESS. More details of the adaptation mechanics can be found in Chapter 4. We consider three adaptation methods here: cost-based and error-based adaptation with Hessian-based anisotropy and standard cost-based MOESS. The accuracy and efficiency of these adaptation approaches are compared in Section 7.4.

7.3 Optimization Algorithms

We consider again two-dimensional airfoil optimization here, in which the airfoil shape and angles of attack are optimized to minimize the overall drag under a range of flight conditions, subject to fixed lift coefficients and a minimum airfoil area. The optimization setup follows the framework developed in Chapter 5. The lift constraints are treated as the trimming constraints, and the angle of attack at each design point α_i is chosen as the trim

variable, $\mathbf{x}_t = [\alpha_1, \alpha_2, \dots, \alpha_{N_m}]$, to decouple the trim conditions at various points. During each flow solve, the trim constraints are enforced by a trimming process as presented in Chapter 4. On the other hand, the inequality area constraint is assumed to be measured exactly and handled by the optimizer as it does not depend on the flow solutions. The active design variables are the parameters controlling the airfoil shape, which in this work are the coefficients of the Hicks-Henne basis functions [28, 172], $\mathbf{x}_s = [a_1, a_2, \dots, a_{N_s}] \in \mathbb{R}^{N_s}$. Table 7.1 summarizes the multipoint aerodynamic optimization problem considered in this work. During the optimization, the mesh deformation is performed using IDW or RBF

Table 7.1: Multipoint aerodynamic shape optimization problem

	Function/Variable	Description	Quantity
Minimize	$\sum_{i=1}^{N_m} \omega_i c_{d,i}$	Weighted drag coefficients sum	1
With respect to	\mathbf{x}_s	Hicks-Henne basis function coefficients	N_s
	\mathbf{x}_t	Angles of attack	N_m
Subject to	$c_{\ell,i} - \bar{c}_{\ell,i} = 0$	Lift constraints	N_m
	$A - A_{\min} \geq 0$	Area constraint	1

interpolations as discussed in Chapter 5.

To avoid over-optimization and over-refinement, the objective error, including the trim effects, is estimated to incorporate mesh adaptation and shape optimization. The multifidelity algorithms proposed in Chapter 5, both error-based and cost-based, are extended to multipoint problems and are summarized in Algorithm 7.1 and Algorithm 7.2, respectively. Again, mesh adaptation is avoided during the line-search to ensure objective-sensitivity consistency.

7.4 Results

As a simple demonstration of the proposed multipoint optimization frameworks, we consider two-dimensional airfoil optimization problems in transonic flow regimes, over a range of flight conditions. The goal of the optimization is to search for an optimal airfoil shape and angles of attack to minimize the drag coefficients, subject to fixed lift trim conditions and a minimum area constraint. We only consider the discretization errors in drag and lift calculations, and the airfoil area measurements are assumed to be exact. Furthermore, the trim constraint tolerances are always set to be sufficiently small to make sure the sensitivity calculation in Eqn. 7.12 is accurate. The airfoil shape is parameterized with 16 Hicks-Henne basis functions, and the design parameter vector includes both the shape parameters and the angle of attack at each design point. The shape parameters

Algorithm 7.1 Optimization with error estimation and mesh adaptation (error-based)

input : initial design \mathbf{x}_0 , initial coarse mesh \mathcal{T}_h , optimization tolerance levels $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n$, safety factor η

output: adapted meshes at each design point $\mathcal{T}_{h,i}$
 optimized design \mathbf{x}^* with controlled objective error $\delta J_{m,h}^{\text{adapt}} \leq \mathcal{O}_n$

```

1 for  $l = 1, 2, \dots, n$  do
2   set the total error tolerance as  $\tau_l = \eta \mathcal{O}_l$ 
3   while not converged do ▷ optimization algorithm
4     distribute the total error tolerance  $\tau_l$  at each design point as  $\tau_{l,i}$ , using Eqn. 7.23
5     for  $i = 1, \dots, N_m$  do
6       while  $\delta J_{m,i}^{\text{adapt}} > \tau_{l,i}$  do
7         adapt the mesh  $\mathcal{T}_{h,i}$  with refinements ▷ Hessian adaptation
8         update  $\mathbf{x}_{t,l}$  to meet trim constraints ▷ trimming process
9         compute the objective component  $J_{m,i}^{\text{adapt}}$  and its error estimate  $\delta J_{m,i}^{\text{adapt}}$ 
10        end
11      end
12      update the composite objective  $J_m^{\text{adapt}} = \sum_{i=1}^{N_m} J_{m,i}^{\text{adapt}}$ 
13      calculate the composite objective gradient  $DJ_m^{\text{adapt}}/D\mathbf{x}_{s,l}$ , per Eqn. 7.12
14      update the active design  $\mathbf{x}_{s,l}$  with meshes  $\mathcal{T}_{h,i}$  fixed ▷ line search
15    end
16  finish optimization at level  $l$ ,  $\mathbf{x}_{l+1} = \mathbf{x}_l$ 
17 end

```

Algorithm 7.2 Optimization with error estimation and mesh adaptation (cost-based)

input : initial design \mathbf{x}_0 , initial coarse mesh \mathcal{T}_h , cost levels C_1, C_2, \dots, C_n , safety factor η **output**: optimized mesh at each design point $\mathcal{T}_{h,i}$ with total cost C_n optimized design \mathbf{x}^* with optimized accuracy at given total cost C_n

```
16 for  $l = 1, 2, \dots, n$  do
17   distribute the total cost  $C_l$  among various design points as  $C_{l,i}$ , using Eqn. 7.22
   while not converged do                                     ▷ optimization algorithm
18     for  $i = 1, \dots, N_m$  do
19       for  $j = 1, \dots, N_{adapt}$  do
20         adapt the mesh  $\mathcal{T}_{h,i}$  with DOF redistribution           ▷ Hessian/MOESS
21         update  $\mathbf{x}_{t,l}$  to meet the trim constraints             ▷ trimming process
22         compute the objective component  $J_{m,i}^{adapt}$  and its error estimate  $\delta J_{m,i}^{adapt}$ 
23       end
24     end
25     update the composite objective  $J_m^{adapt} = \sum_{i=1}^{N_m} J_{m,i}^{adapt}$ 
26     calculate the composite objective gradient  $DJ_m^{adapt}/D\mathbf{x}_{s,l}$ , per Eqn. 7.12
27     set the optimization tolerance  $\mathcal{O}_l = \eta \sum_{i=1}^{N_m} \delta J_{m,i}^{adapt}$ 
28     update the active design  $\mathbf{x}_{s,l}$  with meshes  $\mathcal{T}_{h,i}$  fixed           ▷ line search
29   end
30   finish optimization at level  $l$ ,  $\mathbf{x}_{l+1} = \mathbf{x}_l$ 
31 end
```

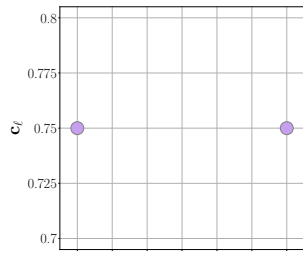
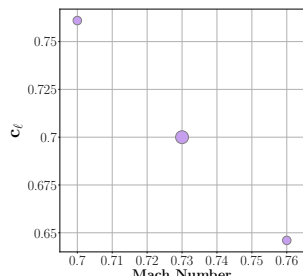
are the active design variables in the optimization, while the angles of attack are used as trim variables to enforce the trim constraints, as described in Section 7.3. Unstructured triangular meshes and DG $p = 2$ approximation are used for the discretization. The airfoil boundary is represented by cubic curved mesh elements. We first test our proposed methods on a two-point, inviscid airfoil optimization problem, following which a more practical turbulent case including three flight conditions is considered. A detailed description of the two cases are given in Table 7.2.

7.4.1 Two-Point Inviscid Transonic Airfoil Optimization

In this test case, the two-point optimization starts with an RAE 2822 airfoil and seeks an optimal shape and angles of attack to minimize the weighted drag coefficient, subject to fixed lift constraints and nondecreasing airfoil area. The two operating conditions including the corresponding lift trim constraints are listed in Table 7.2.

Under the high lift trim condition, flow around the original RAE 2822 airfoil features a strong shock on the upper surface, the location and strength of which vary depending on the operating conditions, *i.e.*, Mach number in this case. Without any priori knowledge about the flow fields around the airfoil at each design point, a fairly fine mesh with

Table 7.2: Operating conditions for multipoint optimization problems

Case	Point	Weights w_i	Mach	c_ℓ	Reynolds number	$M - c_\ell$ plot
7.4.1	1	0.50	0.70	0.750	—	
	2	0.50	0.76	0.750	—	
7.4.2	1	0.25	0.70	0.761	4.79×10^6	
	2	0.50	0.73	0.700	5.00×10^6	
	3	0.25	0.76	0.646	5.21×10^6	

specific refinement around the airfoil is generally used in optimization. Effort can be put into generating meshes suitable for capturing the shocks effectively, either based on experience or output-based error estimates. However, this only helps the analysis on the original shape. If the shock moves or its strength reduces as the optimization proceeds, the specific resolution for the initial design is wasted. Particularly, we expect in this case for the shape to be modified such that the shock strength is significantly weakened. Any substantial refinement on the initial shock location will thus not effectively increase the accuracy but instead add considerable computational cost to the optimization.

We test both the error-based and cost-based optimization frameworks as described in Algorithm 7.1 and Algorithm 7.2, with various adaptation methods. For error-based optimization, we use error-based Hessian adaptation; while for the cost-based optimization, both MOESS and cost-based Hessian adaptation are used. All these three optimizations start with the same initial mesh that was used in the single point optimization in Chapter 6, which consists of 393 triangular elements and is shown in Figure 7.1a. In the error-based optimization, a set of optimization tolerance levels is specified with an ultimate tolerance of 0.02 drag counts, *i.e.*, 2×10^{-6} . On the other hand, the cost-based optimization starts with a fairly low cost level, and degrees of freedom are added once the optimization converges at current cost level, until the final optimization tolerance is around 0.02 drag counts. To compare with traditional methods, we also run fixed-fidelity optimization on fixed meshes. As suggested by the single-point case considered in Chapter 6, uniform refinements around the airfoil boundary might not lead to a good mesh for

this type of flow fields. We thus generate a fixed mesh with specific refinements around the leading and trailing edges following the adapted meshes, while still keeping enough resolution around the airfoil boundary. Again, the mesh has comparable DOF to the adapted meshes. Moreover, a finer fixed mesh with around twice the cost is also generated to further compare fixed-mesh optimization with our proposed methods. The optimization tolerances on the fixed meshes are also set to be 0.02 drag counts. The meshes used in these different optimizations are summarized in Figures 7.1b–7.1h. Only the coarse mesh used in the fixed-fidelity optimization is shown for conciseness, as the finer one has more elements but similar refinement patterns around the airfoil boundary.

The objective convergence and mesh evolution are shown in Figure 7.2. In Figure 7.2a, we plot the objectives versus the aggregated total degrees of freedom, which are only accumulated at each optimization major iteration, *i.e.*, not including the line search. We see in the plot that the estimated discretization error of the objective is always above the optimization tolerance in the fixed-fidelity (fixed-mesh) optimization. On the coarse fixed mesh, the discretization error is large and sometimes even comparable to the objective values. Although the objective error decreases as the finer fixed mesh is used, it is still fairly large compared to the optimization tolerance. In these scenarios, the optimizer may work on the numerical error instead of the physics to minimize the drag, leading to inaccurate designs. On the other hand, discretization error is always controlled to be below the optimization tolerance, or the optimization tolerance is adjusted to be equal to the discretization error in the proposed methods. Furthermore, the variable-fidelity optimizations with different adaptation methods all converge faster at the highest fidelity by virtue of a better starting point obtained from the lower fidelity. Significant computational resources can be saved with fast, low-fidelity optimizations.

We can also observe from the mesh evolution plot in Figure 7.2b that the mesh sizes, if adapted, required to achieve similar accuracy on different operating conditions are different. For all of the mesh adaptation methods considered, the final mesh size for Mach number of 0.70 is smaller than the one required for Mach number 0.76. Mesh adaptation prevents unnecessarily fine meshes from being used for relatively simple operating conditions. The distinction among these methods comes from the difference between error-based and cost-based approaches, and the difference between Hessian adaptation and MOESS. The error-based approach refines the mesh every time the error is above the optimization tolerance, keeping it fixed otherwise, and hence it tends to over-refine areas that are important for some intermediate designs but not necessary for the final design. On the other hand, the cost-based approach always adapts the mesh while keeps the cost fixed at the same fidelity, so that the redistribution of the degrees of freedom avoids

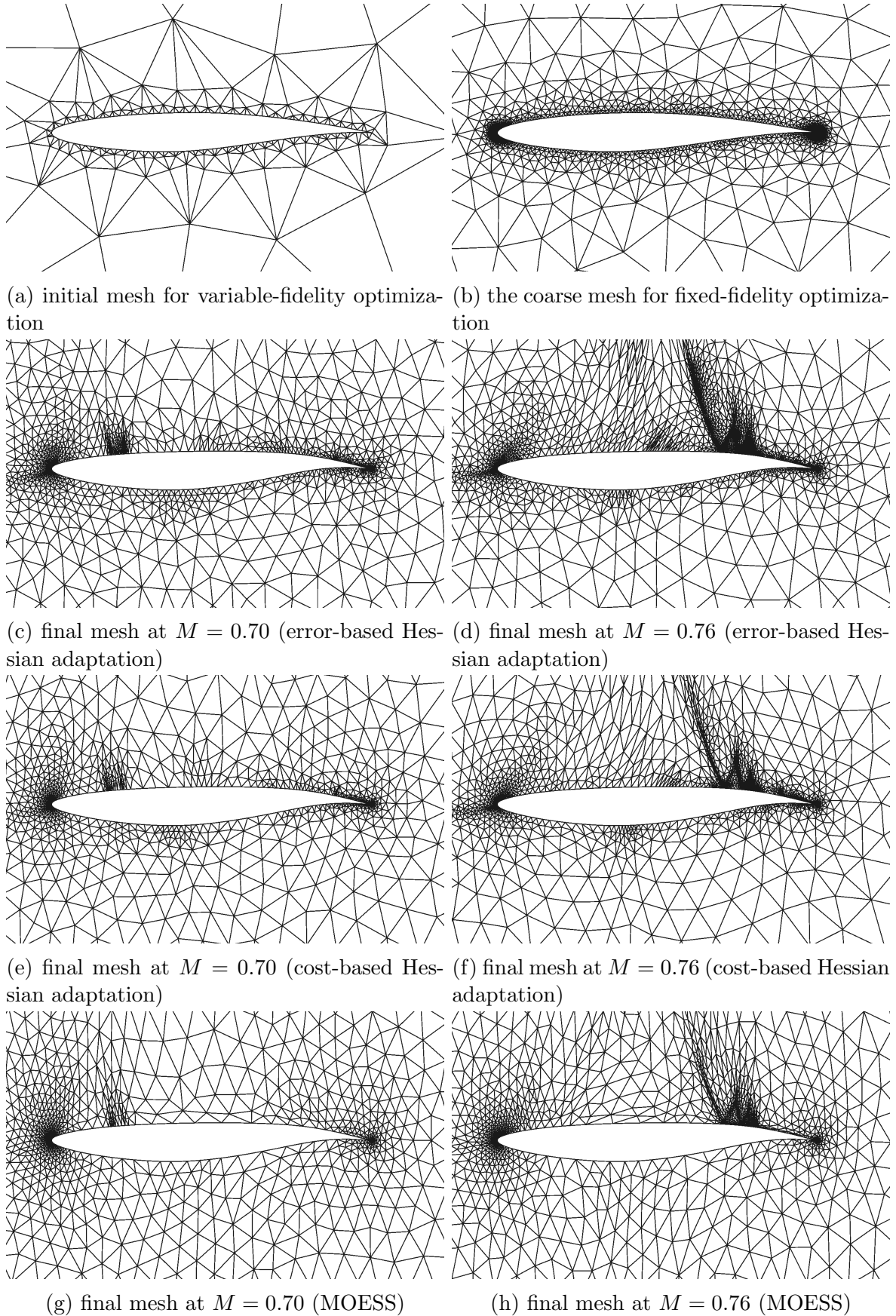


Figure 7.1: Two-point inviscid transonic airfoil optimization: meshes for variable-fidelity and fixed-fidelity optimization.

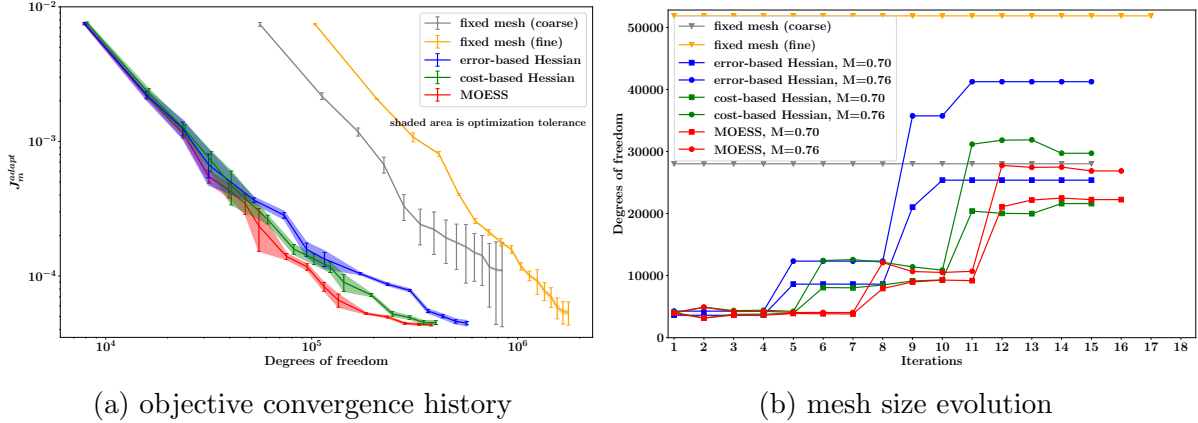


Figure 7.2: Two-point inviscid transonic airfoil optimization: convergence history and mesh size evolution for different methods.

over-refinement and improves the accuracy. If we look at the error-based Hessian adaptation and the cost-based Hessian adaptation (blue and green lines) in Figure 7.2, they have very similar convergence and costs at low fidelities. However at the highest fidelity, the error-based approach has more refinements, which are added for some intermediate designs. Those extra refinements do not affect the final accuracy much as we can see both methods have similar final accuracy, however, unnecessary computational costs are added for the optimization. This suggests using the cost-based optimization framework with cost-based adaptation methods. Although it requires several adaptive iterations with fixed DOF at each major optimization iteration, it prevents extremely fine meshes from being used at the highest fidelity, which is always the main overhead in the optimization. In terms of adaptation methods, both using cost-based approach, MOESS benefits from more appropriate anisotropy detection, resulting in lower cost and better accuracy. As we can see in Figure 7.1f and Figure 7.1h, MOESS meshes tend to have more anisotropic elements in the post shock locations, though they have similar refinement at the shocks. If we look at the upstream region of these two meshes, as shown in Figure 7.3, the difference is more evident: Hessian adaptation only has isotropic refinement along the stagnation streamline since it is important for the output prediction but the Mach number field is isotropic across it; However, MOESS is able to detect the anisotropy through sampling and puts anisotropic resolution along the stagnation streamline. Therefore, we can see in Figure 7.2 that at the low fidelities, with similar cost, MOESS achieves lower objective error, hence better convergence and better design at the low fidelities. With a better starting design, the optimization at the highest fidelity has smoother convergence, as the sharp objective change at the highest fidelity that occurs in both error-based and cost-based Hessian adaptation is not observed in MOESS. Since the flow solve in the optimization

always restarts from the solution on the previous design, MOESS converges faster and consumes less CPU time compared to cost-based Hessian adaptation due to smaller design changes and better mesh anisotropy alignment to both primal and adjoint features, even though the aggregated total degrees of freedom are close. The computational cost saving is reflected in Table 7.3. The proposed variable-fidelity optimization frameworks with adapted meshes achieve substantial time savings compared to fixed-fidelity optimization with fixed meshes. As mentioned, the cost-based algorithm outperforms the error-based one with considerable time savings on the highest fidelity; cost-based optimization with MOESS achieves the most time savings, around 2 times and 7 times speedup compared to the optimizations on the coarse and fine fixed meshes, respectively.

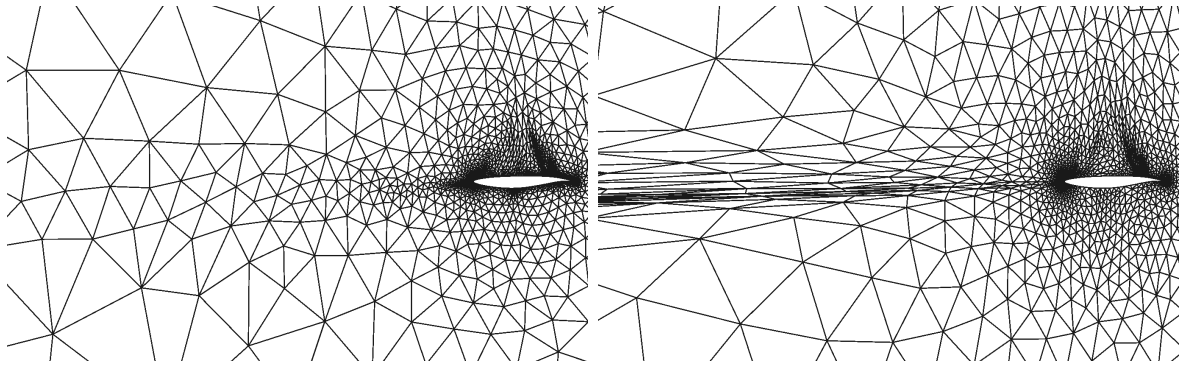


Figure 7.3: Meshes around the stagnation streamline, the left mesh is from cost-based Hessian adaptation, the right one is the MOESS adapted mesh, both at $M = 0.76$.

The initial and optimized airfoils are compared in Figure 7.4, while the final objective values are collected in Table 7.4, in which the corresponding “true” objective values are also obtained via adapted meshes on the final designs, with discretization error controlled to be small compared to the final optimization tolerance. All of the optimizations flatten the upper surface near the forward section, while curving and increasing the thickness at the lower surface. The curvature reduction on the top surface smooths the flow acceleration region to weaken the shock, while the thickened lower surface and curved aft section are required to maintain the lift and area constraints. Therefore, the strong shocks are significantly reduced at both operating conditions, as shown in the pressure distributions in Figures 7.4a–7.4b. In the optimization runs with adapted meshes, areas around the airfoil leading and trailing edges are significantly refined, and many elements are dedicated to the shocks and the stagnation streamline. However, in the optimization with fixed meshes, elements are not efficiently distributed, and areas that are important for output predictions are not well-resolved, which as a result causes high objective error as seen in Figure 7.2a. When the numerical error is too high, for example on the coarse

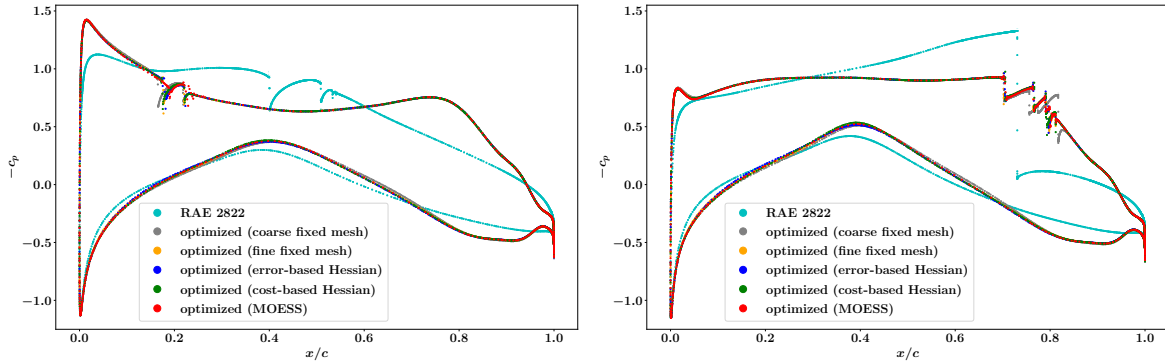
Table 7.3: Two-point inviscid transonic airfoil optimization: computational cost comparison. In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity. Computational time results are obtained using parallel runs with 8 threads on the same machine (Intel Core i7-3770 3.40 GHz CPU with 16GB total RAM).

	Optimization level	Optimization tol (Drag count)	CPU time (s)
Fixed-fidelity (coarse fixed mesh)	L3	0.020	5.243×10^4
Fixed-fidelity (fine fixed mesh)	L3	0.020	1.639×10^5
Variable-fidelity (error-based Hessian)	L1	2.000	7.880×10^2
	L2	0.200	7.153×10^3
	L3	0.020	3.668×10^4
Variable-fidelity (cost-based Hessian)	L1	≈ 1.329	2.149×10^3
	L2	$\delta J_m^{\text{adapt}} \approx 0.130$	8.241×10^3
	L3	≈ 0.015	2.113×10^4
Variable-fidelity (MOESS)	L1	≈ 0.822	2.334×10^3
	L2	$\delta J_m^{\text{adapt}} \approx 0.074$	5.550×10^3
	L3	≈ 0.007	1.630×10^4

fixed mesh, the optimization converges to a noticeably different design compared to the designs obtained from other optimizations, as shown in Figure 7.4c. Thus the “true” objective value for the optimized design on the coarse fixed mesh is much higher compared to designs obtained on the other meshes. In the optimization with the fixed fine mesh, the discretization error is still high, but the optimization is able to converge to a similar design compared to designs produced by optimizations with discretization error control, as shown in Figure 7.4c. Although the “true” objective value is also close to (still slightly higher) the objective values of our proposed methods with mesh adaptation (the difference among these methods is below or comparable to the optimization tolerance, which means that the optimization on these meshes converges correctly), the final objective value reported on the fixed mesh is far from accurate for practical design and the cost is extremely high compared to our proposed methods, which is observed in both Table 7.3 and Table 7.4. In contrast, the proposed methods with mesh adaptation are able to obtain a reasonable design, and the associated error estimation is also sufficiently accurate to provide confidence in the final design and the computed output quantities.

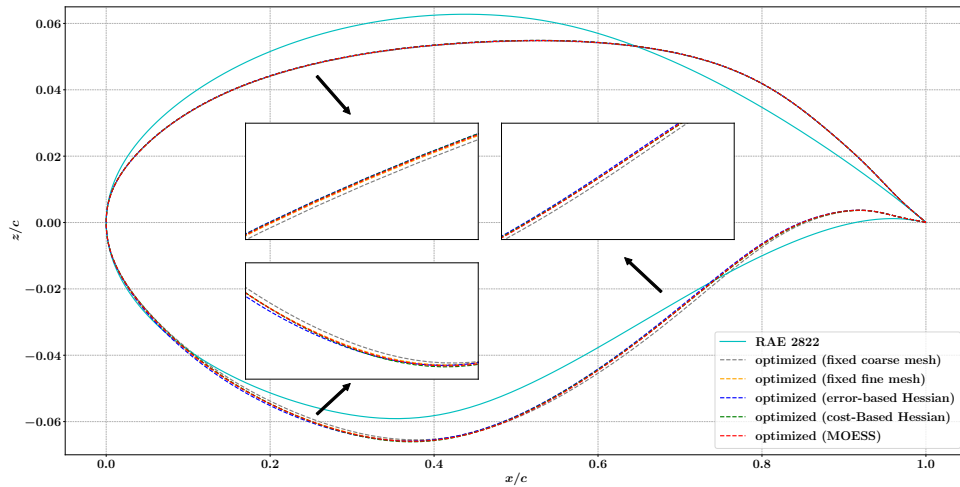
To help better understand the pressure distribution in Figures 7.4a–7.4b, Mach number contours at each design point are shown here in Figures 7.5–7.6. As we can see in Figure 7.5, for the lower Mach number considered, all of the optimized designs reduce the distance between the two original weak shocks and push them towards the leading edge. The final distance between the two weak shocks is a little larger on the design obtained

from the fixed coarse mesh compared to the designs from other meshes, resulting in a higher drag in this flight condition. The shock location and strength are pretty much the same in the other designs. For the higher Mach number operating condition in Figure 7.6, all of the final designs feature a complex shock structure near the trailing edge. The shock locations and the shock structures are all similar, except that an extra weak shock is present after the complex shock structure on the fixed coarse mesh design, again, resulting in a higher drag compared to other designs.



(a) pressure distribution at $M = 0.70$

(b) pressure distribution at $M = 0.76$



(c) initial and optimized shape

Figure 7.4: Two-point inviscid transonic airfoil optimization: pressure distribution for the initial and optimized designs.

7.4.2 Three-Point Turbulent Transonic Airfoil Optimization

Another problem considered here is a more sophisticated fully-turbulent case. We set up a three-point optimization problem, analogous to minimizing the integrated drag coefficients over a range of Mach numbers at a fixed aircraft weight and altitude. The

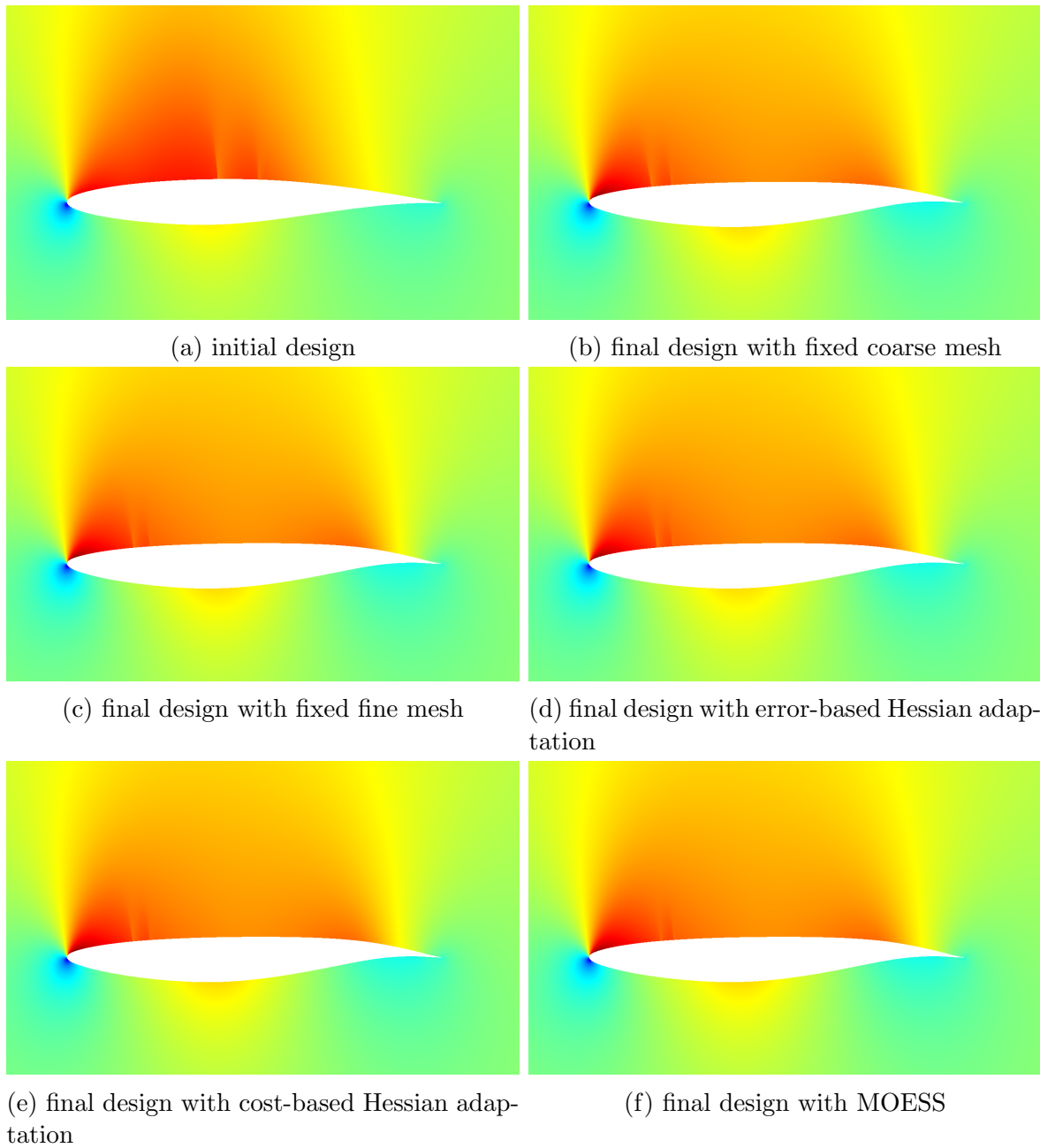


Figure 7.5: Two-point inviscid transonic airfoil optimization: Mach contours at $M = 0.70$. The color limit is $[0, 1.28]$ for all of the contours.

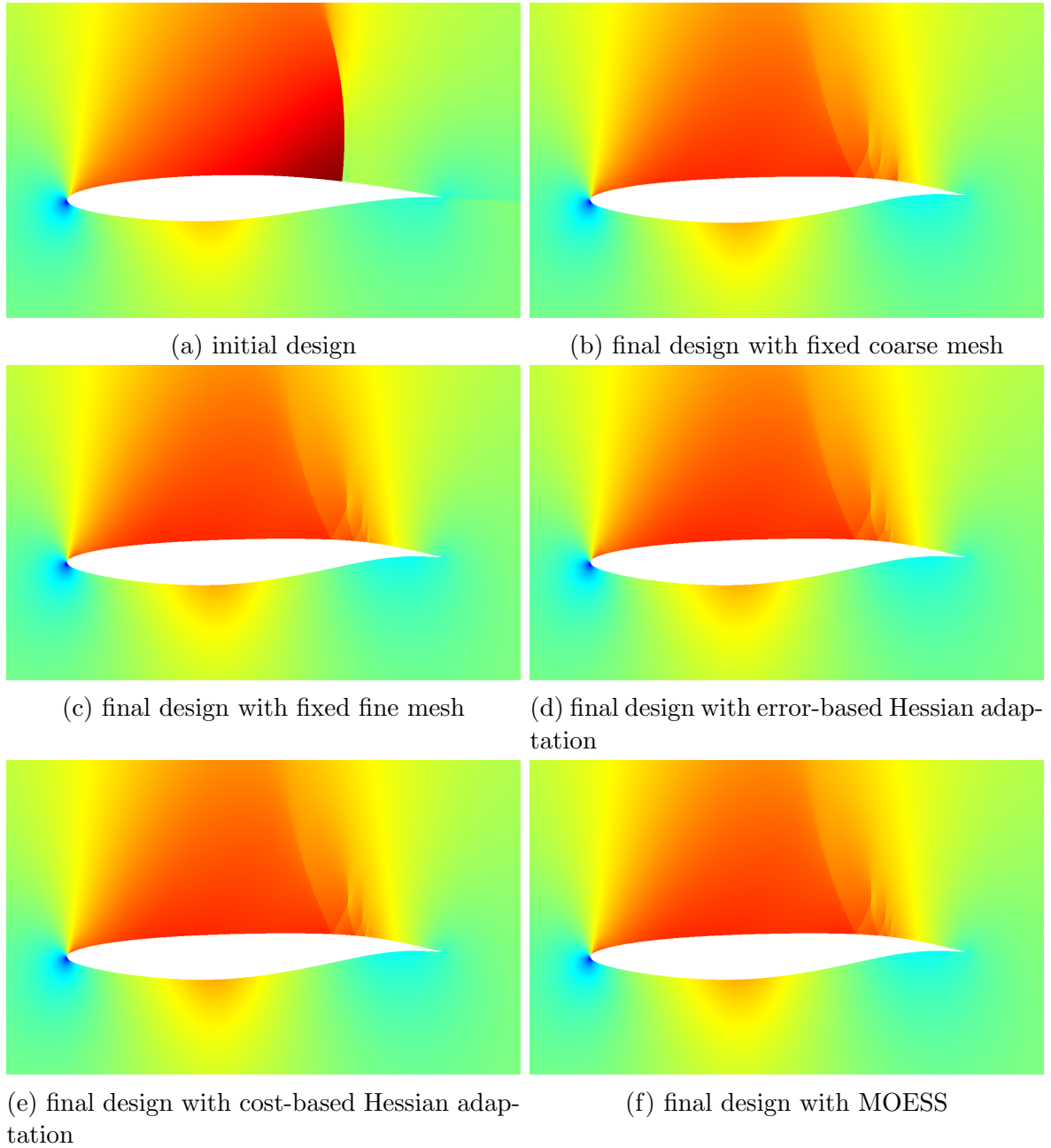


Figure 7.6: Two-point inviscid transonic airfoil optimization: Mach contours at $M = 0.76$. The color limit is $[0, 1.4]$ for all of the contours.

Table 7.4: Two-point inviscid transonic airfoil optimization: results on different meshes.

	Final mesh DOF	J_m^{adapt}	J_m^{adapt} (“true”)
Fixed mesh (coarse)	56040	$1.098 \times 10^{-4} \pm 6.778 \times 10^{-5}$	5.031×10^{-5}
Fixed mesh (fine)	103716	$5.366 \times 10^{-5} \pm 1.045 \times 10^{-5}$	4.389×10^{-5}
Error-based Hessian	66630	$4.466 \times 10^{-5} \pm 1.311 \times 10^{-6}$	4.338×10^{-5}
Cost-based Hessian	51324	$4.513 \times 10^{-5} \pm 1.504 \times 10^{-6}$	4.383×10^{-5}
MOESS	49116	$4.333 \times 10^{-5} \pm 7.202 \times 10^{-7}$	4.278×10^{-5}

optimization again starts with the RAE 2822 airfoil, and seeks an optimal shape and angles of attack to minimize the composite drag coefficients with fixed lift constraints and non-decreasing airfoil area; the details of the case setup are given in Table 7.2.

For turbulent flow simulations at high Reynolds numbers, one of the key flow features is the thin boundary layer. Due to the linear velocity profile in the viscous sub-layer, Hessian-based mesh adaptation is usually inefficient since the Mach number Hessian is close to zero within this region. Therefore, only MOESS with cost-based multifidelity optimization is used in this case. The starting and final mesh at each flight condition are compared in Figure 7.7. At all of the flight conditions considered, MOESS is able to effectively detect strong directional flow features in both the primal and adjoint solutions, putting highly-anisotropic elements around the airfoil boundary, at shock locations on the top surface, along the stagnation streamline and also near the wake region.

The objective and mesh size are collected at each optimization step as shown in Figure 7.8. In the convergence plot, we can see that the composite objective errors are close even for different designs at the same optimization fidelity, as the total degrees of freedom are optimally distributed among different flight conditions, and the meshes are optimized individually at each of them. The degrees of freedom assigned to the design point at Mach number of 0.73 are consistently higher than at the other two points, mainly due to the higher weight used during the optimization. On the other hand, the cost distribution between Mach number of 0.70 and 0.76 changes as the design varies. As the optimization progresses, the meshes get refined and optimized, and more detailed design improvement is then made with smaller objective error and tighter optimization tolerance, *i.e.*, both the design and meshes converge to the optimum.

Figures 7.9a–7.9c shows the initial and final pressure distributions at each design point. The corresponding final airfoil shape is shown in Figure 7.9e. As in the inviscid case, the upper surface of the airfoil is flattened, while more curvature is added to the lower surface aft section. As we can see in the pressure distribution plots, both the location and the strength of the original strong shock at each design point are modified. The optimizer

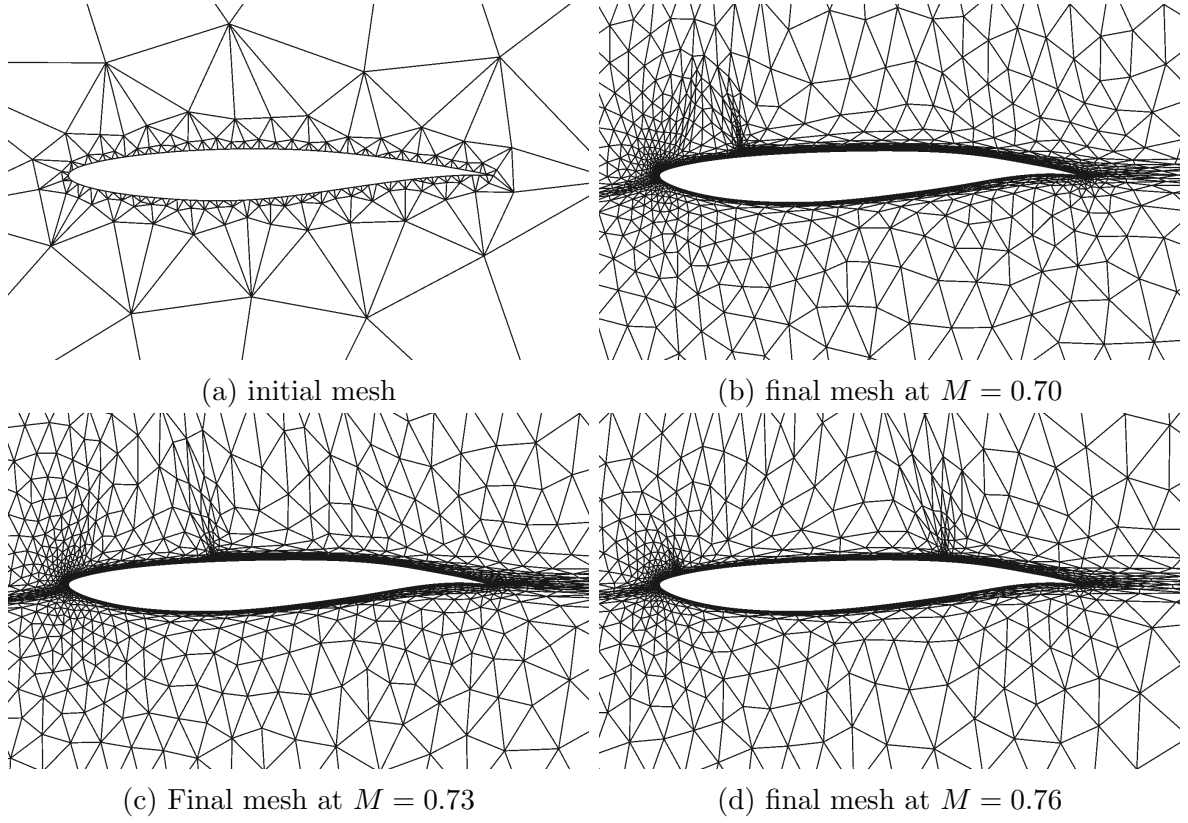


Figure 7.7: Three-point turbulent transonic airfoil optimization: initial mesh and final meshes during the optimization.

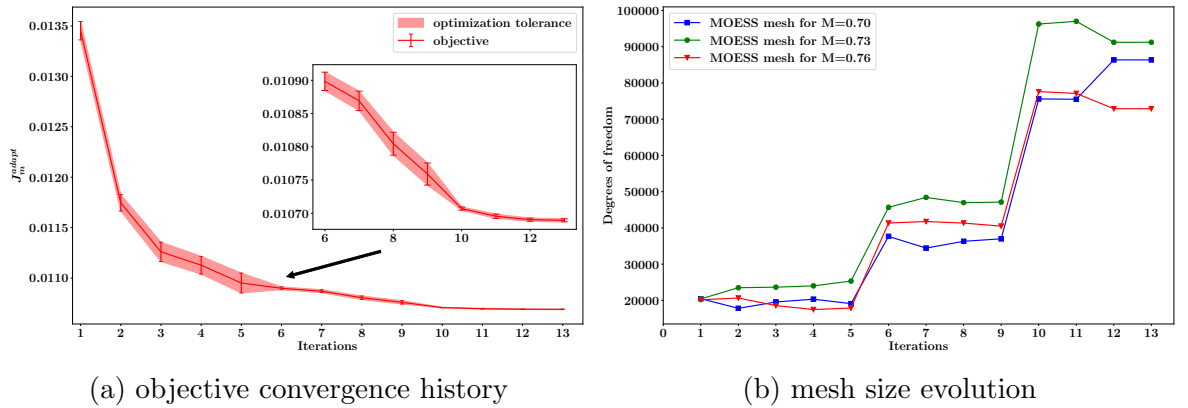


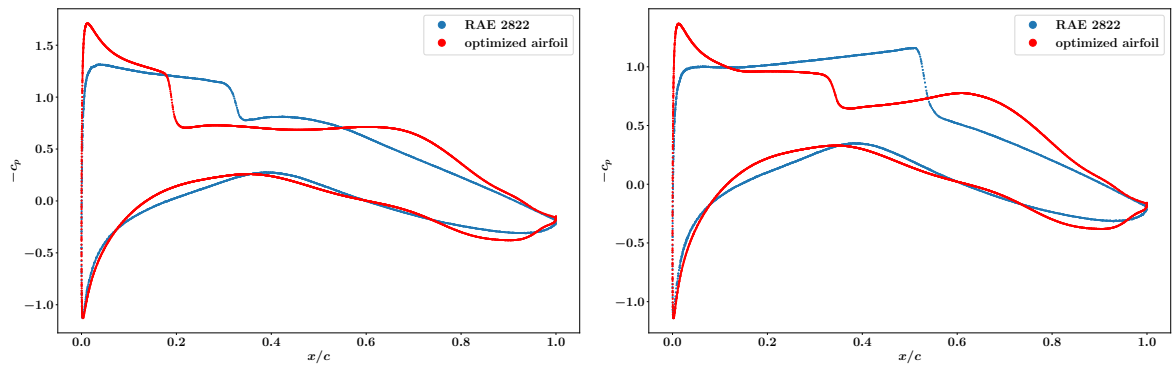
Figure 7.8: Three-point turbulent transonic airfoil optimization: objective convergence history and mesh size evolution for different methods.

weakens the shock at both Mach numbers of 0.73 and 0.76, while slightly strengthens the shock at the Mach number of 0.70, resulting in a significant reduction in the composite drag coefficient as shown in Figure 7.8a. These changes are also well-reflected in the Mach number contours on the initial and optimized designs as shown in Figure 7.10. The drag divergence curves for the original RAE 2822 airfoil and the optimized design around the nominal flight conditions are presented in Figure 7.9d. Despite some sacrifice in the performance at low cruise speeds, the new design achieves significantly lower drag values for Mach numbers above 0.73 and is able to maintain good performance over a much wider range of Mach numbers compared to the original design.

7.5 Summary

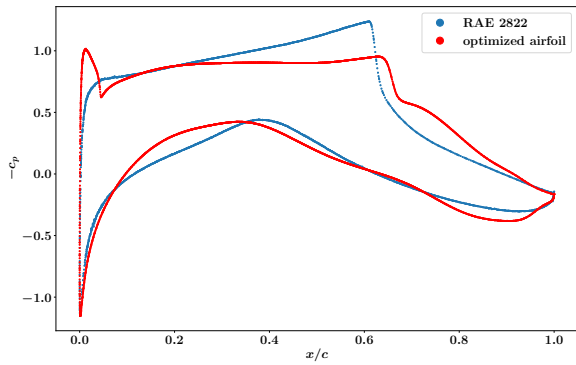
In practical aerodynamic design processes, the optimization problem has to be posed such that a range of operating conditions, including off-design points, are considered in the objective as well as the constraints. To ensure convergence to the “true” optimal design, the numerical error at each design point has to be carefully controlled. As the flow conditions involved can vary dramatically, *a priori* meshes appropriate for all the design points can be hard to generate and are generally not sufficient for the requirements of high-fidelity optimization.

In this chapter, we extend the multifidelity optimization framework with output-based error estimation and mesh adaptation developed in Chapter 5 to multipoint optimization problems. The proposed frameworks can considerably facilitate the optimization setup and accelerate the design process. The designer only needs to input an initial mesh, which can be fairly coarse and easy to generate. The mesh adaptation (fidelity increase) is then tightly coupled with the optimization algorithm either with an error-based or a cost-based strategy. The variable-fidelity optimization framework driven by mesh adaptation is capable of preventing over-optimizing and over-refining, as shown in the test cases. Similar to the single point optimization problem, cost-based optimization approach outperforms the error-based one. In addition, MOESS is in general more efficient than Hessian-based adaptation by more effective anisotropy detection. This benefit can become more significant when higher fidelity is required, or when highly anisotropic physics governs the system.

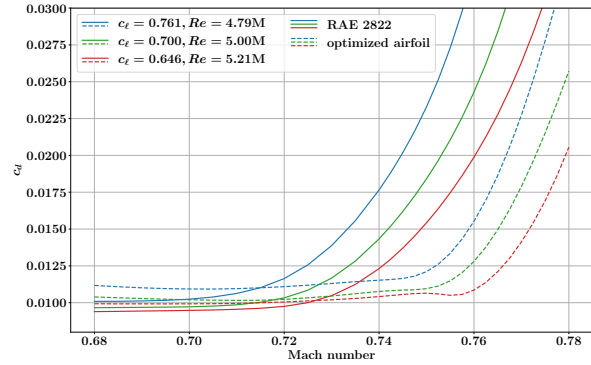


(a) pressure distribution at $M = 0.70$

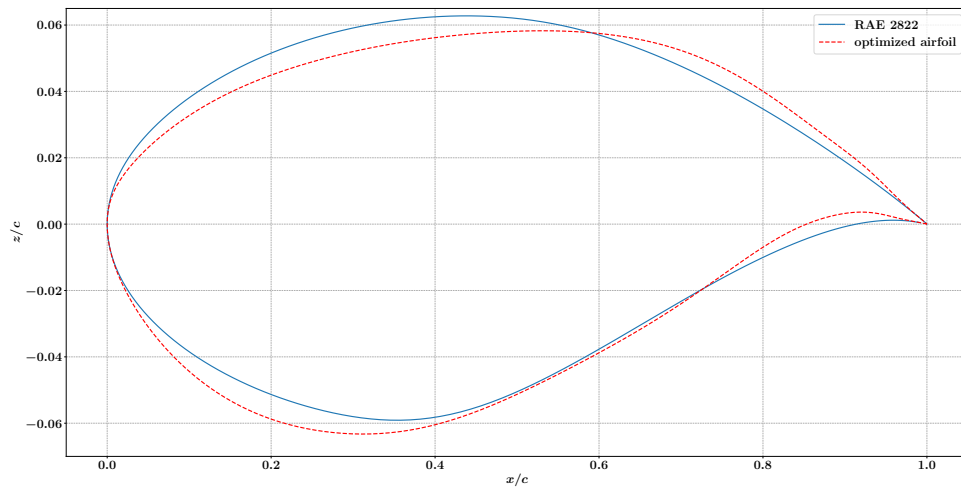
(b) pressure distribution at $M = 0.73$



(c) pressure distribution at $M = 0.76$



(d) drag divergence curves



(e) Initial and optimized shapes

Figure 7.9: Three-point turbulent transonic airfoil optimization: pressure distribution for the initial and optimized designs.

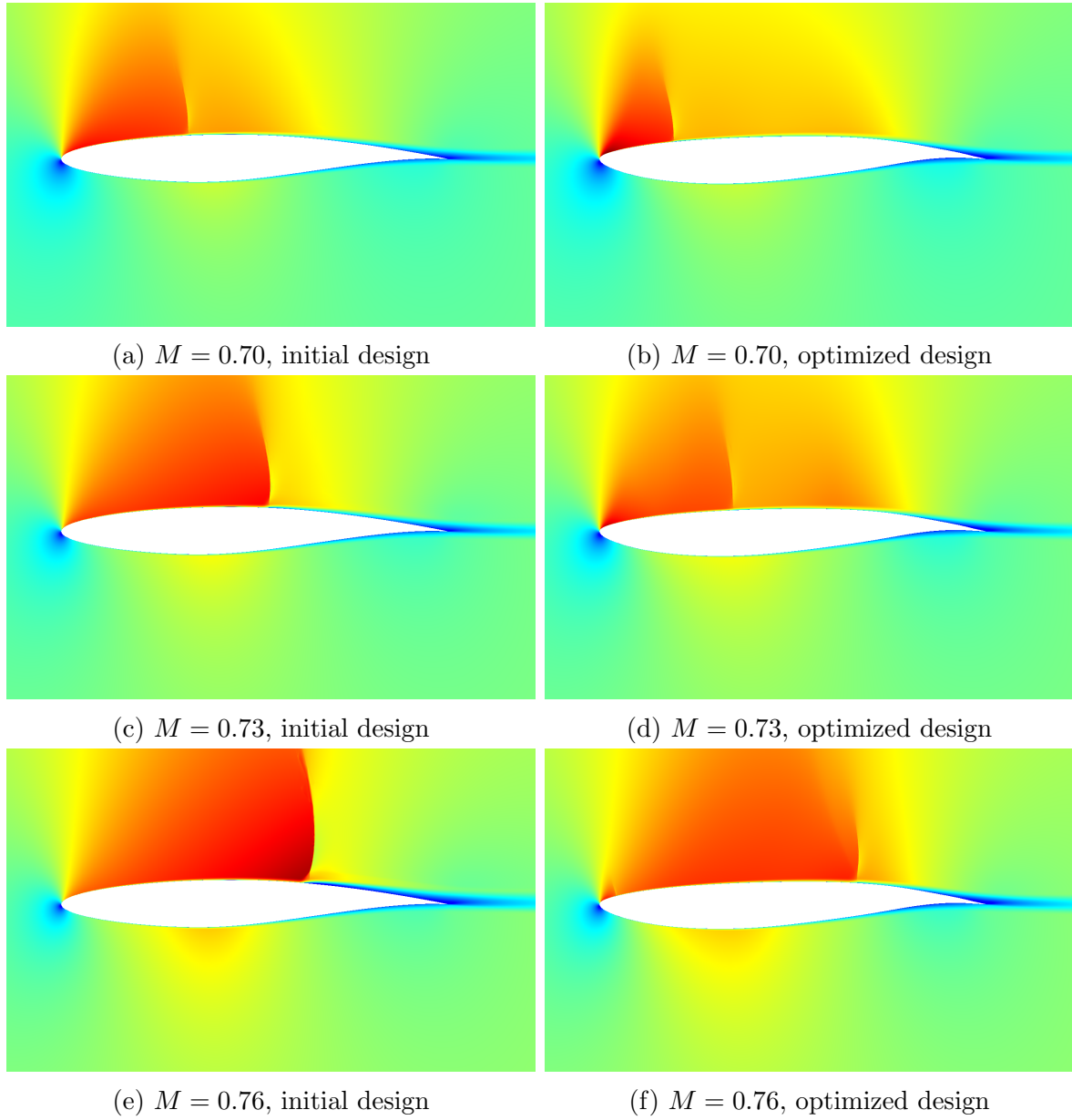


Figure 7.10: Three-point turbulent transonic airfoil optimization: Mach contours. The color limit is $[0, 1.4]$ for all of the contours.

CHAPTER 8

Mesh Adaptation Acceleration Techniques Based on Machine Learning

As we have seen in Chapter 6 and Chapter 7, incorporating output-based error estimation and mesh adaptation with aerodynamic optimization provides a way of improving the optimization accuracy and efficiency. Although MOESS has shown benefits of optimally distributing the mesh DOF and aligning the mesh to desired solution anisotropy, it requires a local sampling process for the unknown convergence rates which is computationally more complicated. Depending on the sampling implementation, it can be more expensive than Hessian-based adaptation although better mesh efficiency is in general found in MOESS meshes. For example, the sampling algorithm involves non-trivial mesh operations (refinement options as shown in Figure 4.6) for its implementation, or in this work requires complex element-local projections. On the other hand, if the adjoint and primal are re-solved (not in this work) for the proposed sampling cuts, the computation cost also grows fast. Hessian-based anisotropy detection, on the other hand, is computationally less intensive as it relies on readily available Hessian information, yet the performance is not generally optimal for a wide range of flow fields and approximation orders.

In this chapter, we investigate the idea of using a machine-learning approach to predict the optimal mesh anisotropy. The goal is to bypass the sampling and optimization process but still achieve similar performance as MOESS. Instead of relying on the generalized error convergence model (Eqn. 4.57) and the optimization process proposed in MOESS, a regression-based surrogate model is constructed by machine learning techniques to directly predict the optimal anisotropy. Without loss of generality, MOESS can also be viewed as a regression-based model, but is built on the fly during the simulation and uses limited amount of data. In the meantime, the model performance is limited by the mathematical form of the proposed error convergence model. By using more advanced machine learning

based regression models, we increase the model power despite the lack of interpretability. In addition, the model is trained before the simulation with a large data set such that no further sampling or optimization is needed in the adaptation. Also, due to the increase amount of data used when building the model, the resulting model is less biased to local irregular behaviors, such as adjoint or primal singularity.

Machine learning techniques have the potential to accurately and efficiently model responses of highly-nonlinear problems over a wide range of input features. In particular, we use artificial neural networks (ANNs) in this work. ANNs are a popular machine learning technique for modeling complex nonlinear mappings between input parameters and target outputs. They emulate the biological systems through connected neurons that are activated under sufficiently strong input activation. Although originally popularized in computer vision and pattern recognition [194], they have received much attention for decades in computational science and engineering, primarily for interpolation and data-driven modeling [195, 196, 197, 198, 199], both of which involve mapping input features to output targets ¹. Despite the fact that physical constraints can be added when constructing the ANN map (often termed physics-informed network [200, 201]), it does not contain the physics of the problem by itself, *i.e.*, the underlying physics cannot be directly extracted from the model. Nevertheless, the map can effectively “learn” relationships between inputs and outputs that may be difficult to discern mathematically from first principles.

The rest of this chapter is adapted from [202]. We first introduce the new machine learning approach for identifying mesh anisotropy in Section 8.1, followed by the model training details in Section 8.2. Section 8.3 presents the deployment results of the proposed model and compare the performance with standard MOESS and Hessian-based adaptation.

8.1 Machine-Learning Anisotropy Detection

As an alternative to mesh subdivision sampling and regression to determine the anisotropy information as shown in Section 4.4.4, we present an approach that uses a neural network to determine anisotropy from relevant features of the primal and adjoint solutions. A key choice in the design of the neural network is the set of input features from which the target anisotropy information can be detected and hence the desired adapted mesh can be obtained. For generality across problems, the input features should

¹We follow the convention of machine learning here, where the network inputs are denoted as features and the outputs are referred to as targets. These terms will be used interchangeably for the rest of the thesis.

be insensitive to scaling of the problem and the choice of physical units. In the Hessian-based anisotropy detection approach, the Mach number Hessian matrix provides such information from the primal solution. In MOESS, both the primal and adjoint solutions are combined via sampling of the adjoint-weighted residual to produce the most efficient anisotropy distribution. Note that in both methods, the element size relies on localized output error estimate, either through direct scaling of the solution Hessian matrix in Hessian-based adaptation, or through the step matrix (trace part) applied on the current mesh metric in MOESS.

The output error estimation formula involves both the primal (via the residual) and the adjoint (as a weight) solutions, and we therefore seek to incorporate both sets of features into the anisotropy measure. Our choice for the features is motivated by the success of the Mach number Hessian, even for high-order solutions. Properly normalized, the Hessian matrix provides information about the relative importance of principal orthogonal directions computed from a scalar field. In order to not exclude any primal or adjoint information, we take as features the Hessian matrices computed from all primal and adjoint state components.

8.1.1 Primal and Adjoint Features

For element e , denote by \mathbf{H}_k^u and \mathbf{H}_k^ψ the Hessian matrices of the k^{th} state and adjoint variables, respectively,

$$[\mathbf{H}_k^u]_{i,j} = \frac{\partial u_k^2}{\partial x_i \partial x_j}, \quad [\mathbf{H}_k^\psi]_{i,j} = \frac{\partial \psi_k^2}{\partial x_i \partial x_j}, \quad i, j \in [1, \dots, d], \quad (8.1)$$

where $k \in [1, \dots, s]$ ranges over the state rank. We do not include the subscript e to denote that this is an elemental quantity, as this will be assumed for all calculations in this section. The second derivatives in the Hessian are computed from the polynomial approximation of the state inside element e . When the approximation order is $p = 2$, the Hessian is uniquely defined for the entire element. To obtain a unique definition for orders $p > 2$, we project the field, via least squares, to $p = 2$. For $p < 2$, which we do not consider in this work, we would obtain a $p = 2$ field using patch reconstruction from neighboring elements.

In two spatial dimensions, to normalize the Hessian, we compute the aspect ratio (AR) and direction of the first eigenvector,

$$AR = \sqrt{\lambda_2^H / \lambda_1^H}, \quad \theta = \arg(\vec{v}_1), \quad (8.2)$$

where $\lambda_1^H \leq \lambda_2^H$ are the Hessian eigenvalues and \vec{v}_1 is the eigenvector corresponding to λ_1^H . The normalized elemental Hessian metric, for either a primal or adjoint field k , is then defined as

$$\bar{\mathbf{H}}_k = \begin{bmatrix} \lambda_1 \cos^2 \theta + \lambda_2 \sin^2 \theta & (\lambda_1 - \lambda_2) \sin \theta \cos \theta \\ (\lambda_1 - \lambda_2) \sin \theta \cos \theta & \lambda_1 \sin^2 \theta + \lambda_2 \cos^2 \theta \end{bmatrix}, \quad \text{where } \lambda_1 = \frac{1}{AR}, \lambda_2 = AR. \quad (8.3)$$

Note that this matrix does not encode sizing, which is not a meaningful quantity from the Hessian matrix in the context of output-based adaptation, and hence has only two independent quantities (e.g. AR and θ). This gives a total of $2s$ primal features and $2s$ adjoint features for use in the machine-learning algorithm.

Inputs into a neural network should often be normalized such that the training convergence is not significantly slowed down by extreme cases, *e.g.*, elements of large aspect ratio. In addition, care must be taken when using an angle as an input, due to the periodic nature of trigonometric functions, such as sine and cosine functions. To address both problems, we do not use the normalized Hessian metric as defined in Eqn. 8.3 directly as an input. Nor do we use AR and θ , which can have very different scales. Instead, we take the natural logarithm of the normalized Hessian metric, which is equivalent to a step matrix \mathbf{S}_k computed relative to the identity matrix. Since the eigenvalues of $\bar{\mathbf{H}}_k$ are multiplicative inverses, the resulting step matrix has zero trace and hence only two independent components in two dimensions,

$$\mathbf{S}_k = \log(\bar{\mathbf{H}}_k) = \begin{bmatrix} -\log(AR) \cos(2\theta) & -\log(AR) \sin(2\theta) \\ -\log(AR) \sin(2\theta) & \log(AR) \cos(2\theta) \end{bmatrix}. \quad (8.4)$$

We use the first row values as the independent components, *i.e.*, input features for the network:

$$S_{k,1} \equiv (\mathbf{S}_k)_{1,1}, \quad S_{k,2} \equiv (\mathbf{S}_k)_{1,2}. \quad (8.5)$$

These two values per scalar field result in a total of $4s$ inputs from the primal and adjoint anisotropy information.

8.1.2 Error Indicator Features

In addition to the primal and adjoint features, state-component contributions to the error indicator can also inform the element anisotropy calculation. The elemental error indicator \mathcal{E} in Eqn. 3.48 (again, we drop the subscript e for conciseness) is a scalar that is already used to determine element size. However, for systems of equations, this indicator

can be broken down into contributions from each equation,

$$\mathcal{E} = \sum_{k=1}^s \mathcal{E}_k, \quad \mathcal{E}_k = |\Psi_{h,k}^T \mathbf{R}_{h,k}(\mathbf{U}_h^H)|, \quad (8.6)$$

where k again indexes the equations (state components), such that \mathcal{E}_k , $\Psi_{h,k}$ and $\mathbf{R}_{h,k}$ represent the local error indicator, adjoint and residual vectors respectively associated with the k^{th} equation. The relative magnitudes of \mathcal{E}_k can then be incorporated into the anisotropy calculation, as these values could arguably influence the importance of Hessian anisotropy information from the various state and adjoint components. Note that we do not assume a direct weighting by these indicators, but rather let the neural network determine the effect of these values on the output. For input into the neural network, we define the normalized error contributions as

$$\bar{\mathcal{E}}_k \equiv \mathcal{E}_k \left(\sum_{k=1}^s \mathcal{E}_k^2 \right)^{-1/2}. \quad (8.7)$$

8.1.3 Surrogate Model Using Neural Networks

8.1.3.1 Artificial Neural Networks

The idea of ANN is to emulate the response of neurons to input stimulation signals, and thus they were also referred to as multi-layer perceptrons (MLPs) in the early days of machine learning and neural science [203]. A simple single-layer ² ANN is shown in Figure 8.1a. It receives an input vector \mathbf{x} (input layer) and applies an affine transformation followed by a nonlinear activation function (hidden layer) to produce an output vector \mathbf{y} (output layer). The map between \mathbf{x} and \mathbf{y} can be written as

$$\begin{aligned} \mathbf{y} &= f(\Theta^{\text{out}}\mathbf{h}), \quad \mathbf{h} = \sigma(\mathbf{z}), \quad \mathbf{z} = \Theta^{\text{in}}\mathbf{x}, \\ \Theta^{\text{in}}\mathbf{x} &\equiv \mathbf{W}^{\text{in}}\mathbf{x} + \mathbf{b}^{\text{in}}, \quad \Theta^{\text{out}}\mathbf{h} \equiv \mathbf{W}^{\text{out}}\mathbf{h} + \mathbf{b}^{\text{out}}. \end{aligned} \quad (8.8)$$

\mathbf{z} is an affine transformation of the input \mathbf{x} with parameters Θ^{in} , which contains both the linear map weights $\mathbf{W}^{\text{in}} \in \mathbb{R}^{\dim(\mathbf{z}) \times \dim(\mathbf{x})}$, and a translation or bias term $\mathbf{b}^{\text{in}} \in \mathbb{R}^{\dim(\mathbf{z})}$. A nonlinear activation function σ then maps \mathbf{z} element-wise to the hidden units \mathbf{h} , often referred to as the hidden *neurons*. The nonlinear activation σ provides the power of modeling complex phenomena and are often defined *a priori*, such as sigmoid or rectified linear unit (ReLU) [204] functions. From the hidden layer to the output, we only showed

²We only count the hidden layer here.

an affine map Θ^{out} in Figure 8.1a. However, one more nonlinear or linear activation f can also be applied.

The complexity and approximation power of a network increase as the number of neurons increases. One can also stack the hidden layers to increase the approximation capacity, resulting in a multi-layer network. Deep ANNs are usually obtained by both increasing the number of hidden layers and the number of neurons in each layer, as shown in Figure 8.1b. For a neural network of L hidden layers, the corresponding model can be written as,

$$\begin{aligned} \mathbf{h}^1 &= \sigma(\Theta^1 \mathbf{x}) \in \mathbb{R}^{N_1}; \\ \mathbf{h}^l &= \sigma(\Theta^l \mathbf{h}^{l-1}) \in \mathbb{R}^{N_l}, \quad l = 2, 3, \dots, L; \\ \mathbf{y} &= f(\Theta^{L+1} \mathbf{h}^L). \end{aligned} \quad (8.9)$$

The number of hidden layers L , and the dimension of each hidden layer N_l , are hyper-parameters of the network, which can be fine-tuned to achieve higher efficiency and better performance. The network trainable parameters (weights and bias) Θ^l , $l = 1, \dots, L+1$, are obtained by minimizing an objective function, often called the *loss function*, measuring the deviation between the model outputs and the target values from the observed data.

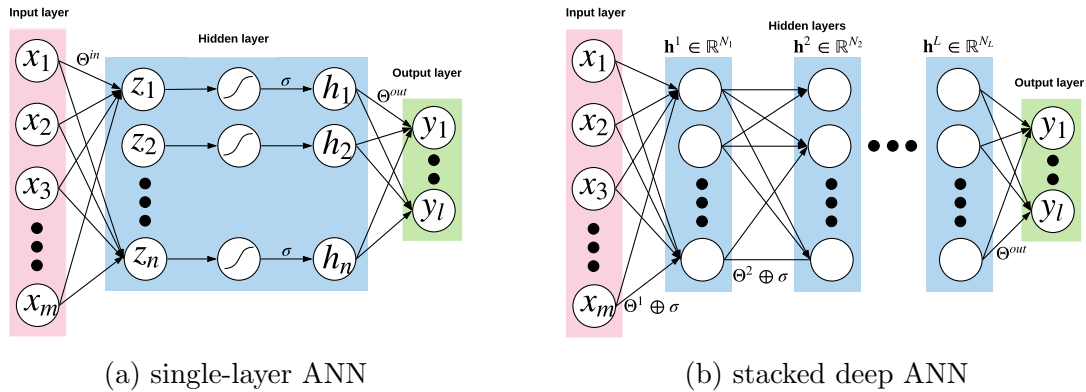


Figure 8.1: Structures of artificial neural networks (ANNs).

8.1.3.2 Proposed Network Architectures

In this work we consider artificial neural networks for mapping the input primal and adjoint features into the desired output, which contains the optimal mesh anisotropy. Suppose we have an optimal mesh, it can then be represented by a mesh-implied metric field \mathbf{M}_{out} which encodes both the element size and stretching information. Following the idea presented in Section 8.1.1, we first drop the sizing information to get the normalized

mesh metric, $\overline{\mathbf{M}}_{\text{out}}$; then the step matrix of normalized mesh metric with respect to the identity metric, *i.e.*, the natural logarithm of $\overline{\mathbf{M}}_{\text{out}}$, is used to encode the mesh anisotropy. Thus the target output of the neural network, \mathbf{y}_{out} , is defined as

$$\mathbf{S}_{\text{out}} = \log(\overline{\mathbf{M}}_{\text{out}}) \Rightarrow \mathbf{y}_{\text{out}} = [(\mathbf{S}_{\text{out}})_{1,:}]^T. \quad (8.10)$$

Again, the step matrix \mathbf{S}_{out} is trace-free, thus only the first row of the step matrix is used to define the output vector.

The network architectures considered are simple ANNs consist of one or two hidden layers, $\mathbf{h}_1, (\mathbf{h}_2)$, between the input layer, *i.e.*, the features, \mathbf{x} , and the output layer, *i.e.*, the targets \mathbf{y}_{out} . Four network configurations, labeled as network A-D, are tested in this work. They differ in the size of the input layer and the number and size of the hidden layers, which are compared in Table 8.1. The inputs to all networks contain the primal and adjoint Hessian information, through the $4s$ values in $\mathbf{S}_{k,1,:}^u$ and $\mathbf{S}_{k,1,:}^\psi$. Networks A-C also include the s relative errors, \mathcal{E}_k in the input. Networks A and D both have one hidden layer of $10s$ neurons, network B has two hidden layers of $40s$ neurons each, and network C has one hidden layer of $2s$ neurons. The associated numbers of weights and biases are given in the last two columns of Table 8.1. In summary, network A is the baseline network, B is a large/fine network, C is a small/coarse network, and D is the baseline without the relative error indicators.

Table 8.1: Four neural network architectures for anisotropy detection.

Network	Input Layer	Hidden Layers	# Weights	# Biases
A	$\mathbf{x}_0 \in \{\mathbf{S}_{k,1,:}^u, \mathbf{S}_{k,1,:}^\psi, \mathcal{E}_k\} \in \mathbb{R}^{5s}$	$\mathbf{x}_1 \in \mathbb{R}^{10s}$	$50s^2 + 20s$	$10s + 2$
B	$\mathbf{x}_0 \in \{\mathbf{S}_{k,1,:}^u, \mathbf{S}_{k,1,:}^\psi, \mathcal{E}_k\} \in \mathbb{R}^{5s}$	$\mathbf{x}_1 \in \mathbb{R}^{40s}, \mathbf{x}_2 \in \mathbb{R}^{40s}$	$1800s^2 + 80s$	$80s + 2$
C	$\mathbf{x}_0 \in \{\mathbf{S}_{k,1,:}^u, \mathbf{S}_{k,1,:}^\psi, \mathcal{E}_k\} \in \mathbb{R}^{5s}$	$\mathbf{x}_1 \in \mathbb{R}^{2s}$	$10s^2 + 4s$	$2s + 2$
D	$\mathbf{x}_0 \in \{\mathbf{S}_{k,1,:}^u, \mathbf{S}_{k,1,:}^\psi\} \in \mathbb{R}^{4s}$	$\mathbf{x}_1 \in \mathbb{R}^{10s}$	$40s^2 + 20s$	$10s + 2$

The parameters associated with the network consist of all of the weights and biases, Θ^l , $l = 1, 2, 3$. The values of these parameters are determined using an optimization procedure, that minimizes the mean squared error loss function between predicted and actual output layer values. The actual values come from training data, which are obtained from multiple MOESS iterations on prototypical cases. Each MOESS iteration produces one “training point” for each element of the mesh, so for meshes with many elements, a large amount of data can be obtained with a relatively small number of MOESS iterations.

Once a network is trained, it is implemented in the adaptive code as a replacement for the Hessian-only metric anisotropy and stretching calculation in the output-based mesh

adaptation method described in Section 4.4.3. More specifically, we use the same adaptation code used for Hessian-based adaptation in Section 4.4.3, but the desired mesh metric is now directly from the network model rather than Mach number Hessian. However, due to the normalization we had for the input and output of the network, it only contains the mesh stretching while the element sizing still follows Section 4.4.3.1. Figure 8.2 illustrates the calculation process from the features: primal, adjoint, and error indicators, to the desired element anisotropy, encoded by the metric $\bar{\mathbf{M}}_{\text{out}}$ presented on the lower-left in the figure. This metric then replaces the Mach Hessian in an otherwise equivalent adaptive procedure presented in Section 4.4.3.

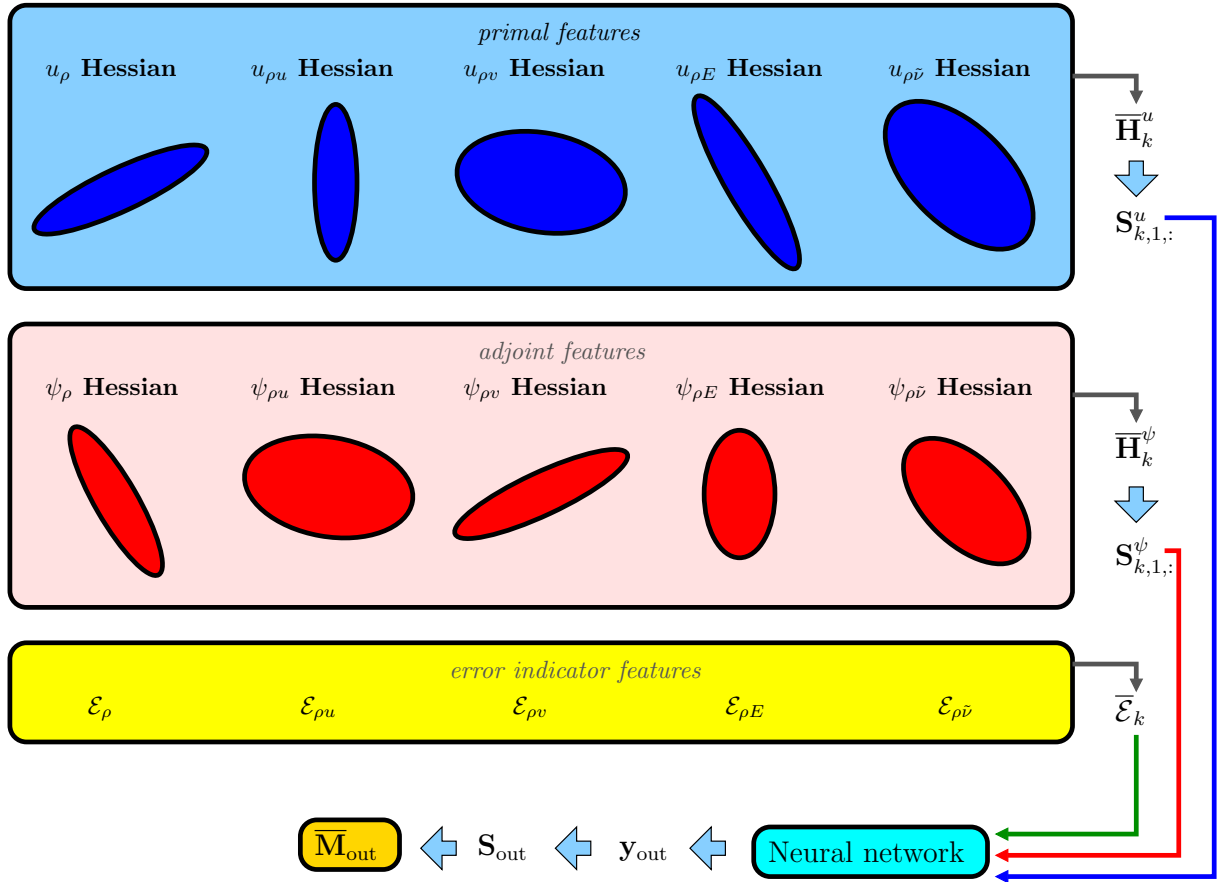


Figure 8.2: Flowchart of the neural-network implementation.

8.2 Neural Network Training

Several prototypical aerodynamic flow cases are run to provide training data for the anisotropy prediction neural network. Table 8.2 describes these cases and includes figures of the primal and adjoint solutions, and of the optimized meshes. All cases are two-

dimensional and use the RANS-SA equations as presented in Section 2.3. A variety of flow Mach numbers are included, ranging from subcritical to supersonic. The Reynolds numbers are representative of aircraft flight conditions, $\mathcal{O}(10^6)$. Force outputs, drag and lift, are considered for error estimation and adaptation. In some cases, for a given flow condition, both outputs yield two different adaptation sequences.

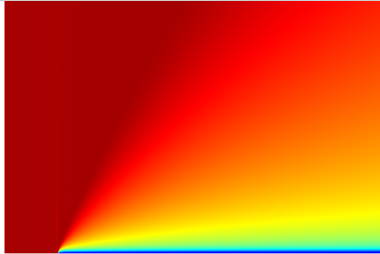
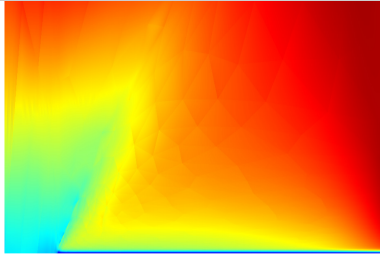
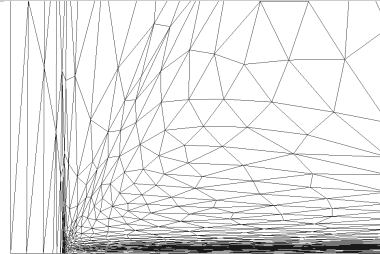
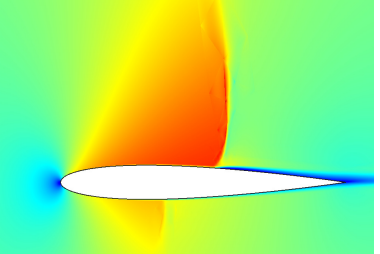
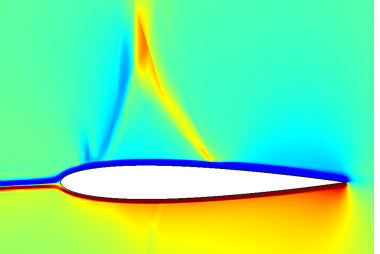
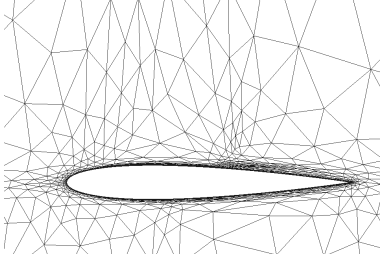
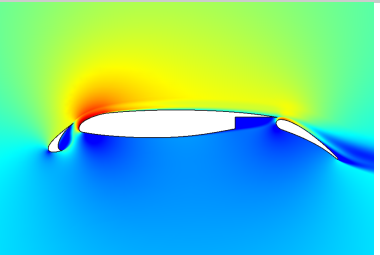
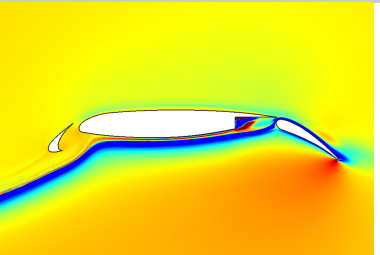
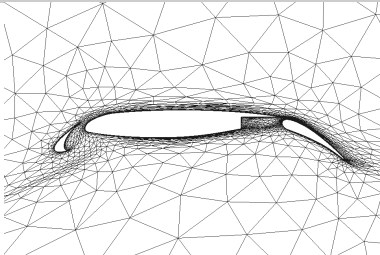
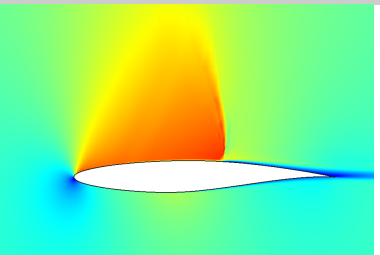
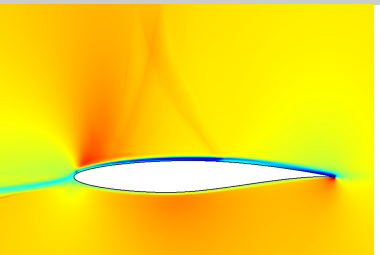
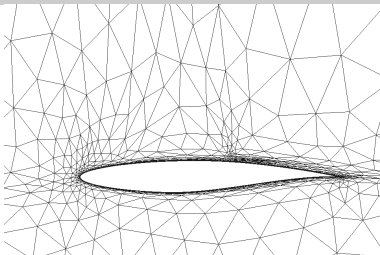
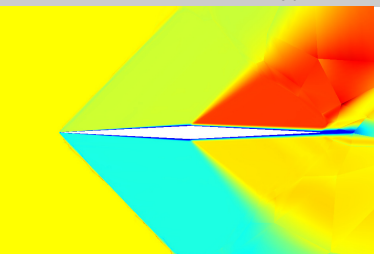
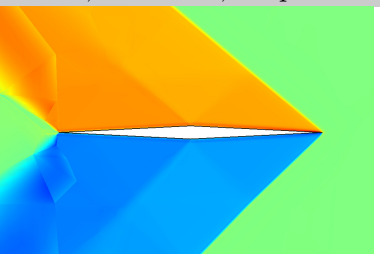
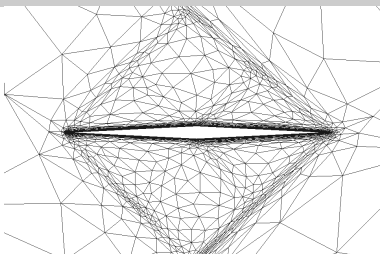
Adapted meshes are generated at a chosen target degree-of-freedom cost using MOESS, at a solution approximation order of $p = 2$ and separately at $p = 3$ (for use in the results in Section 8.3.6). The meshes for the different cases are not all made to be of the same size, but the number of adaptive iterations collected is chosen such that the total amount of training data (product of mesh size and number of iterations) is approximately the same among the cases. Data collection begins once the mesh size and outputs stabilize following initialization of MOESS iterations with a user-generated, sub-optimal mesh. Specifically, data from the first ten MOESS iterations are discarded before collection.

The input data, which consist of the normalized primal and adjoint variable Hessian matrices, are computed from the current-space primal and adjoint solutions for each element. For $p = 2$, these matrices are constant inside elements, whereas for $p > 2$, the matrices would be averaged across the element interiors. The ground truth for the output data are computed using the normalized mesh-implied metric, since we are using MOESS to generate the meshes, converted to a step matrix via a natural logarithm as discussed in Section 8.1.3.2.

The total number of training data points (elements) over all cases and adaptive iterations is 121,840. These data are randomized and split 70%/30% into training and validation categories. The training data are used to drive the optimization, whereas the validation data are used to monitor the loss on untrained data. The networks proposed in Table 8.1 are all implemented in TensorFlow [205] and the network parameters are optimized (trained) using the adaptive moment (Adam) estimation algorithm [206]. The training data are broken into mini-batches of size 1000 for the optimizer, and the learning rate is set to 0.001. Several tens of thousands of optimization iterations typically lead to a stabilization of the mean-squared error, as shown in Figure 8.3. Typically, an order of magnitude drop in the loss is observed, without a significant difference between training and validation data loss. This indicates that we are not over-fitting the data, which would be difficult to do with the small neural network size presently considered. Furthermore, the results of the training were not found to be overly sensitive to the choices of the mini-batch size or the learning rate.

Training of the networks yields weight matrices and bias vectors. Interpretation of these values is not straightforward, especially for large networks. However, some insight

Table 8.2: Neural network training cases.

Flat plate: $M_\infty = 0.2, 0.5$, $Re = 10^5, 10^6$, output = drag <i>(images scaled 100x vertically)</i>		
		
<i>Mach contours</i>	<i>drag adjoint (x-momentum)</i>	<i>Optimized mesh (700 elem)</i>
NACA 0012: $M_\infty = 0.8$, $\alpha = 1.25^\circ$, $Re = 5 \times 10^6$, outputs = drag, lift		
		
<i>Mach contours</i>	<i>lift adjoint (x-momentum)</i>	<i>Optimized mesh (2700 elem)</i>
MDA 30P/30N: $M_\infty = 0.2$, $\alpha = 5^\circ$, $Re = 10^6$, outputs = drag, lift		
		
<i>Mach contours</i>	<i>lift adjoint (x-momentum)</i>	<i>Optimized mesh (5300 elem)</i>
RAE 2822: $M_\infty = 0.734$, $\alpha = 2.79^\circ$, $Re = 6.5 \times 10^6$, outputs = drag, lift		
		
<i>Mach contours</i>	<i>drag adjoint (x-momentum)</i>	<i>Optimized mesh (2600 elem)</i>
Diamond airfoil: $M_\infty = 1.5$, $\alpha = 2^\circ$, $Re = 10^6$, output = lift		
		
<i>Mach contours</i>	<i>lift adjoint (energy)</i>	<i>Optimized mesh (3900 elem)</i>

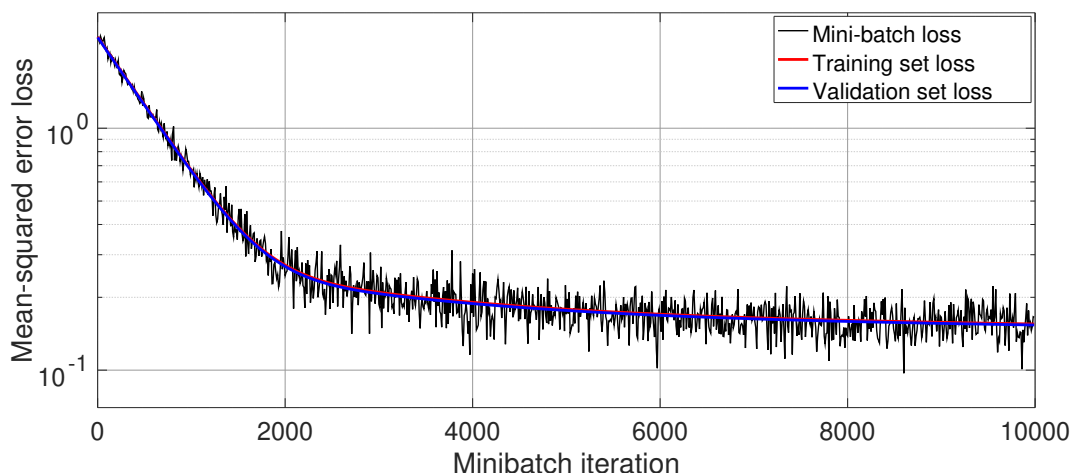


Figure 8.3: Neural network A training loss history, using a 70% training, 30% validation split.

can be gained from a graphical representation, as shown in Figure 8.4 for networks A,C, and D (network B is too large to clearly visualize). These have all been trained using $p = 2$ solutions. Note that the hidden neurons have been sorted based on importance, calculated from incoming and outgoing weight magnitudes, with the most important/active neurons at the top. We see that in network A, the primal and adjoint features contribute similarly to the final output (based on the weights of connections emanating from the input blocks), and that the error indicators (particularly the fourth, energy component) are active. In network C, which has a small hidden layer, the primal features contribute somewhat more than the adjoint features, and the error indicators are quite active (large magnitude weights). Finally, for network D, which does not use the error indicators, the primal features are more active than the adjoint features. In addition, in networks A and C, the bias magnitudes are large for the important hidden layer neurons, whereas this is not the case for network D.

8.3 Adaptive Simulation Results

This section presents results obtained by implementing the trained neural network in an adaptive solution process and using it to generate adapted meshes for various flow cases at different degrees of freedom. When using the network to determine anisotropy information, the same *a priori* element sizing technique, described in Section 4.4.3.1, is used as in the Hessian-based approach. The neural network results are compared to both Mach number Hessian-based adaptation and MOESS.

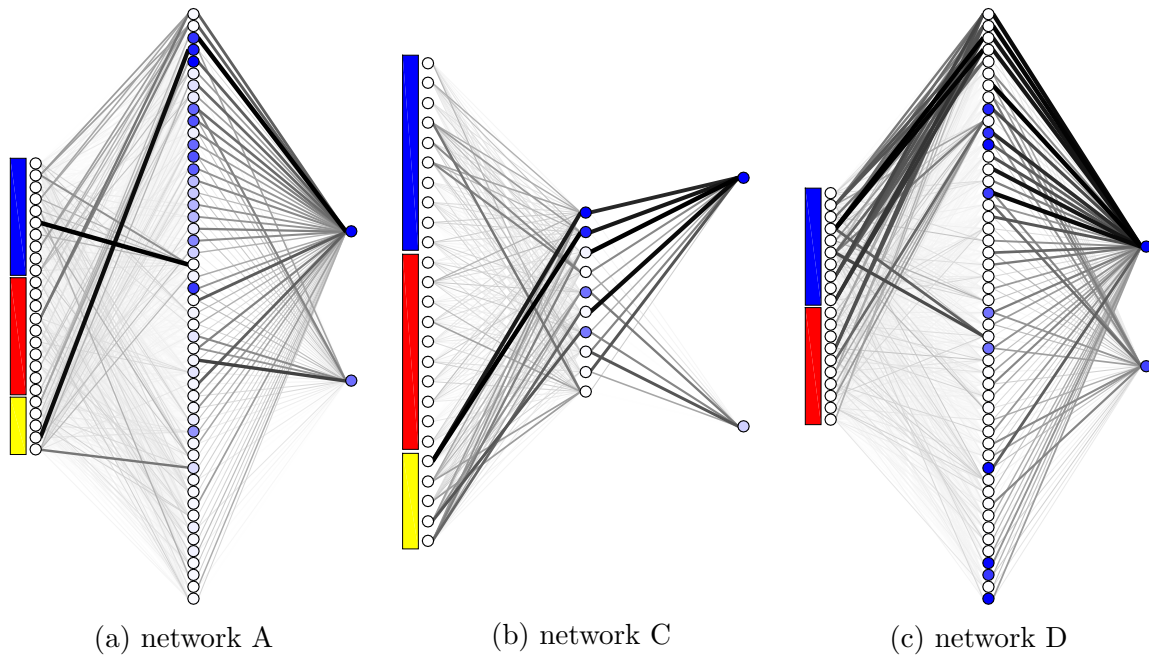


Figure 8.4: Visualization of the trained neural networks. The input highlighting indicates primal (blue), adjoint (red), and error indicator (yellow) neurons. The darkness and width of connecting lines indicate weight magnitudes, and the color of the neurons indicates bias magnitude. The hidden layer neurons are sorted by importance, using the product of the sum of incoming weight magnitudes and the sum of outgoing weight magnitudes. The two outputs are $S_{\text{out},1,1}$ and $S_{\text{out},1,2}$.

8.3.1 NACA 0012 Airfoil

This test case consists of RANS fully-turbulent flow over a NACA 0012 airfoil at $M_\infty = 0.8$, $\alpha = 1.25^\circ$, and $Re = 5 \times 10^6$. The computational domain consists of a square farfield boundary 100 chords away from the airfoil. An adiabatic no-slip wall boundary conditions is imposed on the airfoil. Shock capturing is performed using element-based artificial viscosity [128], and the output of interest is the drag coefficient on the airfoil. A hand-generated initial isotropic mesh of 2800 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at $p = 2$ solution approximation order for three target degrees of freedom (DOF) costs: 8000, 16000, 32000. At each target DOF, 10 adaptive iterations are run, using the final mesh from the previous target DOF as the starting mesh at each iteration. The adaptive methods compared are MOESS, Mach-number Hessian, and the neural-network approaches.

Figure 8.5 presents the output convergence histories obtained from the adaptive runs. For each method, only one data point is shown at a given DOF target: this is the average output at the average DOF value computed over the last 4 adaptive iterations at that target DOF. An “exact” value is also indicated, obtained by running a simulation at an approximation order of $p = 3$ on a uniformly-refined version of the final MOESS adapted mesh.

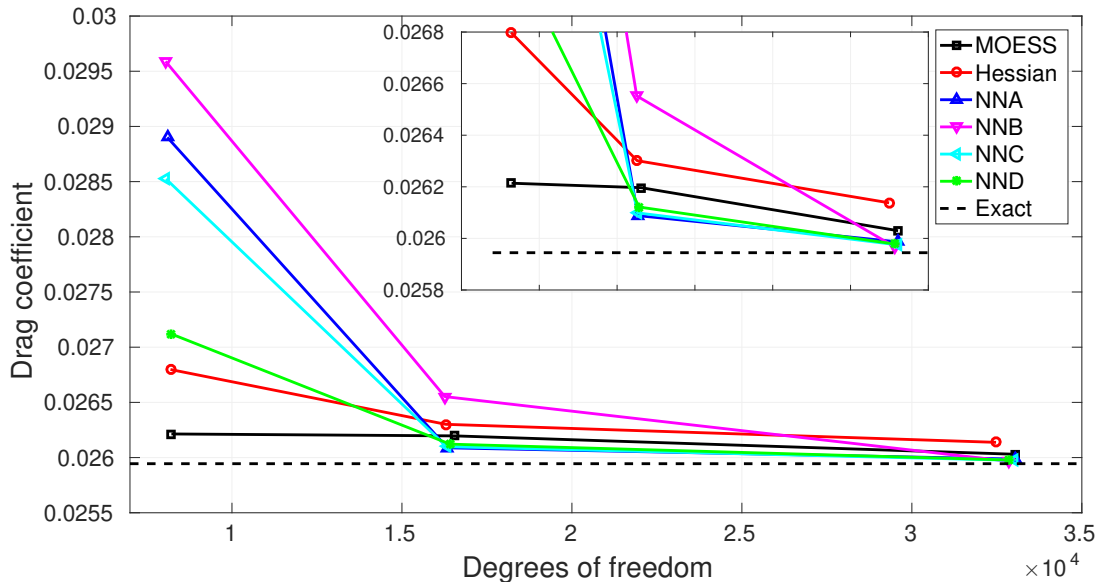


Figure 8.5: NACA 0012, $M_\infty = 0.8$, $\alpha = 1.25^\circ$, $Re = 5 \times 10^6$: output convergence history.

We see that after the coarsest target DOF, the adaptive results using most of the neural-network methods are closer to the exact value compared to those using the Mach number

Hessian anisotropy. By the finest target DOF, this is true for all of the network methods. As both methods are driven by the same output-based element sizing information, the observed error reduction is made possible by more efficient meshes resulting from improved anisotropy information. Specifically, correct anisotropy identification allows for optimal use of degrees of freedom to reduce the output error.

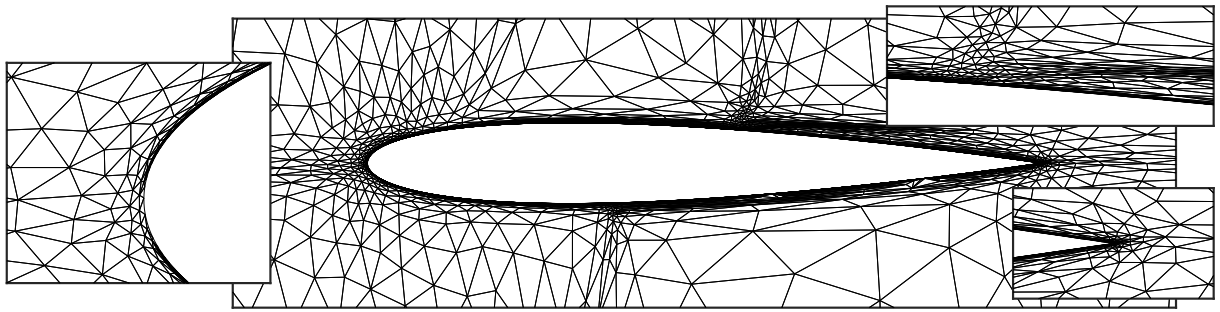
The neural network approaches can achieve more optimal mesh anisotropy because they incorporate information not only from the primal solution, but also from the adjoint. The networks were trained to reproduce the element stretching obtained from MOESS, but they also include a degree of regularization, which leads to a slightly-improved performance over MOESS. This point is discussed further in the subsequent results.

Figure 8.6 shows the final adapted meshes at the highest target DOF for three of the adaptive approaches tested. The neural network meshes are similar, and hence only results from one network (A) are shown. The flow is transonic with a strong shock on the upper surface, a weaker shock on the lower surface, and rapid boundary-layer growth in the vicinity of the shocks. From the three figures, we see that the resolution of the shock, which is minimal to start with in the output-based setting, is similar among the methods, with the neural-network indicator exhibiting slightly less primal-based anisotropy compared to the Hessian indicator and MOESS.

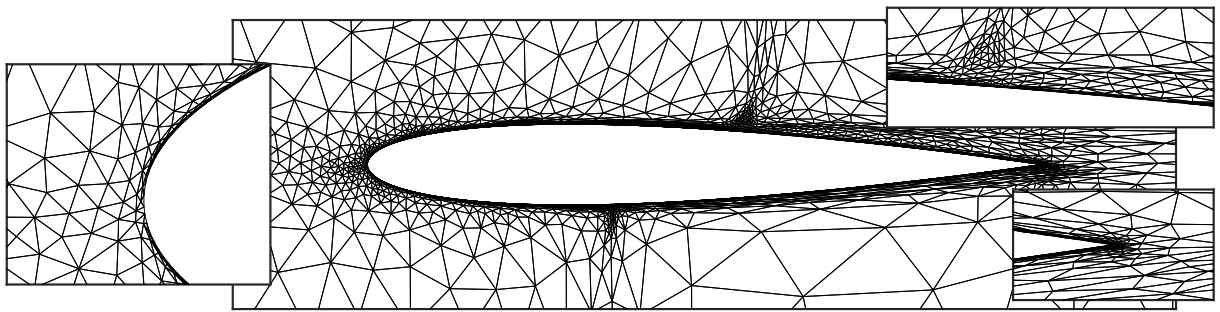
As shown in Table 8.2, the adjoint variable possesses a λ structure variation on the upper surface, and both the neural-network anisotropy measure and MOESS show evidence of its resolution. The Mach number Hessian measure does not align elements to this structure, which is specific to the adjoint. The most notable difference is the leading-edge stagnation streamline anisotropy, which is again an adjoint feature that is resolved by MOESS and the neural networks. Although the extent to which this feature needs to be resolved for accurate output prediction is still a point of debate, it is reassuring to see the neural network reproduce the MOESS behavior, for which it was trained. Finally, near the trailing edge, the Hessian-based measure shows more flow-aligned anisotropy, due to the primal wake, which is not as apparent in MOESS and the neural-network approach.

8.3.2 Diamond Airfoil

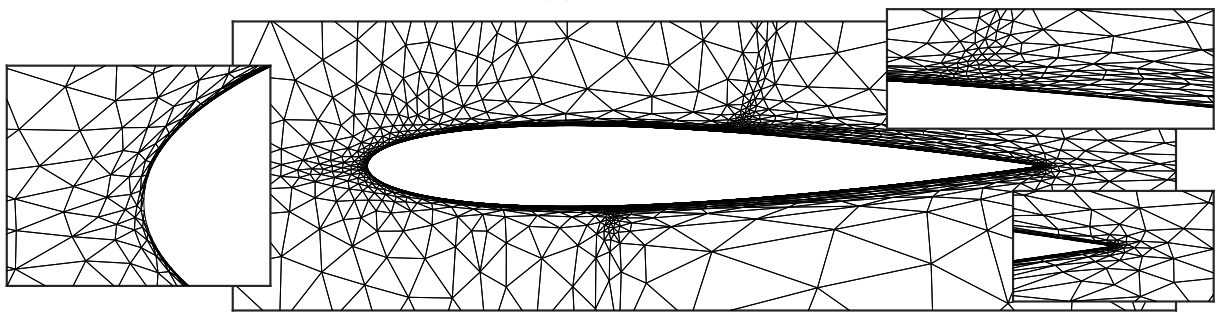
This test case consists of RANS supersonic flow over a diamond airfoil at $M_\infty = 1.5$, $\alpha = 2^\circ$, and $Re = 1 \times 10^6$. The leading and trailing edge corner angles are both $2 \tan^{-1}(.05)$. The computational domain consists of a square farfield boundary 100 chords away from the diamond. An adiabatic no-slip wall boundary conditions is imposed on the airfoil. Shock capturing is again performed using element-based artificial viscosity [128]. The output of interest is the lift coefficient on the airfoil. A hand-generated initial isotropic



(a) MOESS



(b) Hessian



(c) neural network A

Figure 8.6: NACA 0012: final adapted meshes.

mesh of 1500 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at $p = 2$ solution approximation order for three target DOF costs: 10000, 20000, 40000. At each target DOF, 10 adaptive iterations are performed, using the final mesh from the previous target DOF as the starting mesh. As in the previous case, MOESS, Mach-number Hessian, and the neural-network anisotropy prediction methods are compared.

Figure 8.7 presents the output convergence history obtained from the adaptive runs, where again averaging has been performed over the last 4 iterations at each target DOF. The “exact” value is also indicated, obtained from a $p = 3$ run on a uniformly-refined version of the final MOESS adapted mesh.

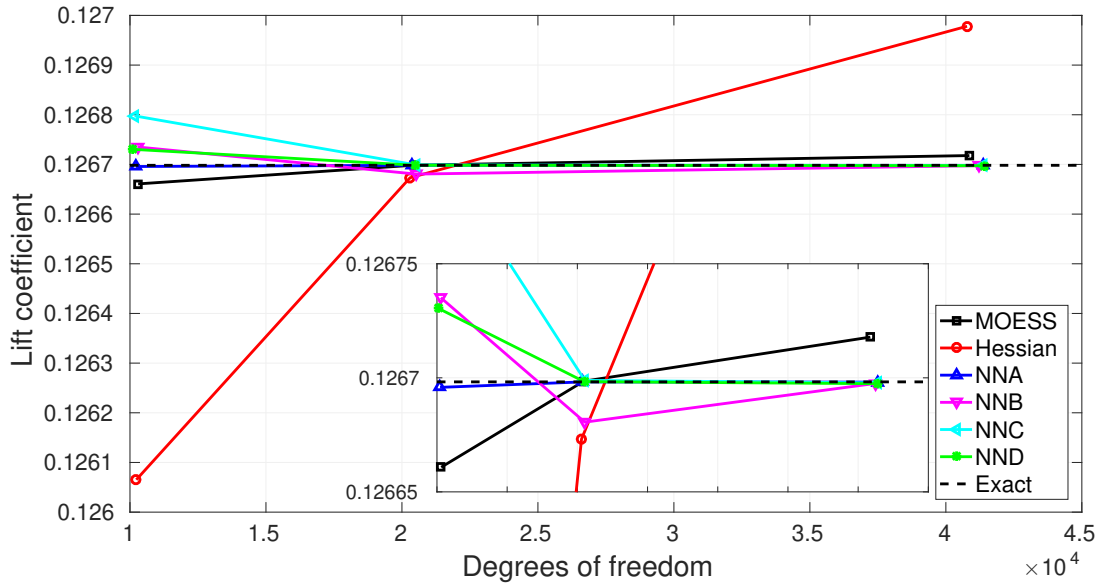


Figure 8.7: Diamond airfoil, $M_\infty = 1.5$, $\alpha = 2^\circ$, $Re = 1 \times 10^6$: output convergence history.

We see that at all target DOF, the adaptive results using neural-network anisotropy are much closer to the exact value compared to those using the Mach number Hessian anisotropy. In this case, the difference between Hessian-based anisotropy and MOESS is significant, and the machine-learning approaches follow the MOESS results closely. As in the previous case, the accuracy improvement is made possible by a more optimal distribution of degrees of freedom due to correct anisotropy in regions where the output error is more sensitive to resolution in one direction than another. In this case, the performance of MOESS and the neural-network are close in terms of output accuracy versus degrees of freedom, and in fact, on the finest mesh, the neural networks perform better, as shown in the zoomed-in portion of Figure 8.7. This behavior will be seen in

subsequent results, and it is likely due to a regularization effect of the neural networks: their anisotropy predictions are less sensitive to outlier primal and adjoint input features, such as lines of primal or adjoint discontinuity.

Figure 8.8 shows the final adapted meshes at the highest target DOF for the adaptive approaches tested. The neural network adapted meshes are similar, and again only one (A) is given. The flow over the diamond airfoil is supersonic, with oblique shocks emanating from

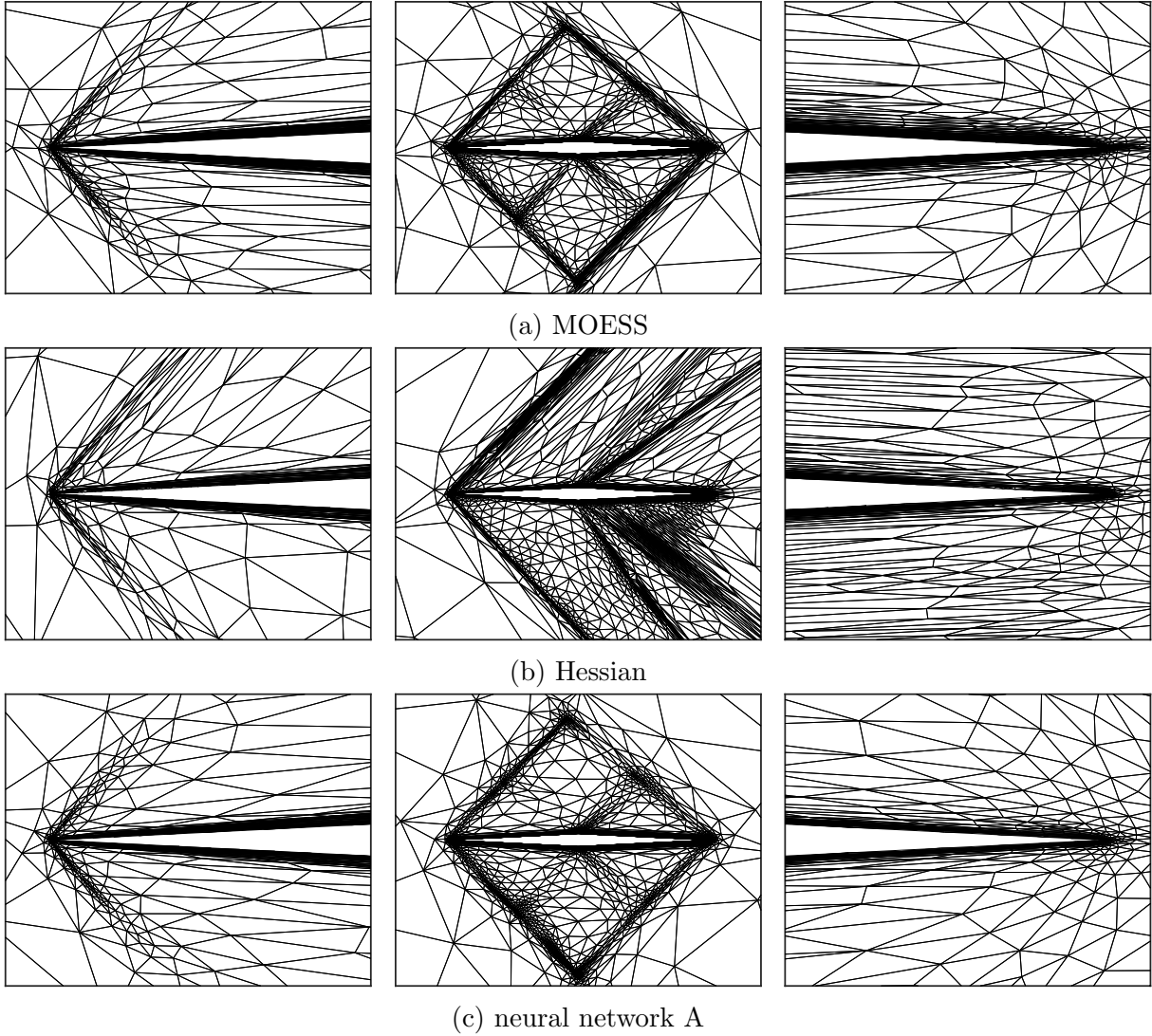


Figure 8.8: Diamond airfoil: final adapted meshes.

the leading and trailing edges, and expansions originating from the mid-chord corners. The Hessian-based primal anisotropy detection approach places shock/expansion-aligned stretched elements in these areas, as expected based on the Mach number variations. In contrast, MOESS places anisotropic elements in not only these areas, but also the corresponding adjoint features, which mimic the primal shock/expansion structure but

in reverse (right to left instead of left to right). The result is a primal/adjoint resolving anisotropic mesh, with somewhat higher emphasis evident for the adjoint features. Finally, the neural network anisotropy detection methods, like MOESS, also target the primal and adjoint features, but with a lower degree of anisotropy in the primal and adjoint shocks/discontinuities – the anisotropy remains high in the boundary layer.

We note top/bottom asymmetry of the resolution is expected due to the presence of a nonzero angle of attack. In addition, all methods target the boundary layer with anisotropic elements, although the Hessian-based approach places more emphasis on anisotropy away from the wall near the trailing edge/wake compared to the other two methods. Combined with the excessive resolution of the shocks and expansions due to a primal-only anisotropy measure, particularly on the bottom aft portion of the airfoil, Hessian adaptation produces mesh anisotropy that is nearly orthogonal to the required one, which leads to a less efficient distribution of degrees of freedom and hence larger error for a given target DOF.

8.3.3 MDA 30P/30N Airfoil

The third test case consists of subcritical RANS flow over the McDonnell Douglas Aerospace (MDA) 30P/30N three-element airfoil at $M_\infty = 0.2$, $\alpha = 5^\circ$, and $Re = 5 \times 10^6$. The computational domain consists of a C-shaped farfield boundary that is 100-200 main-element chords away from the airfoil. An adiabatic no-slip wall boundary conditions is imposed on the airfoil main element, the leading-edge slat, and the trailing-edge flap. The flow is subcritical and hence no shock capturing is employed. The output of interest is the lift coefficient on the airfoil. An Euler-flow adapted initial isotropic mesh of 3000 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at $p = 2$ solution approximation order for three target DOF costs: 8000, 16000, 32000. At each target DOF, 10 adaptive iterations are performed, using the final mesh from the previous target DOF as the starting mesh. Again, MOESS, Mach-number Hessian anisotropy, and the neural-network anisotropy methods are compared.

Figure 8.9 presents the output convergence history obtained from the adaptive runs, where averaging has been performed over the last 4 iterations at each target DOF. The “exact” value is also indicated, obtained from a $p = 3$ run on the final MOESS adapted mesh, uniformly-refined.

We see that the Hessian-based anisotropy detection method yields meshes that have the lift output closer to the exact value compared to MOESS, although the prediction

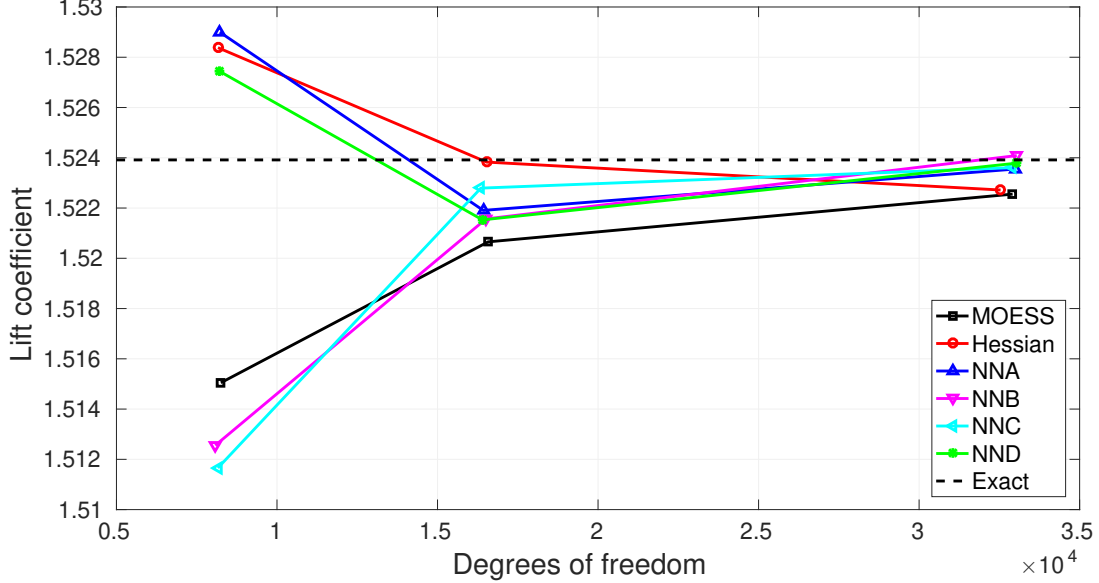


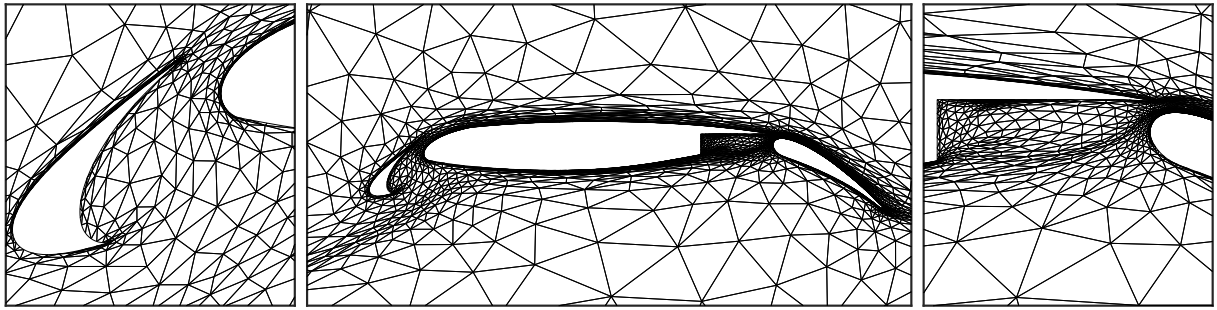
Figure 8.9: MDA 30P/30N, $M_\infty = 0.2$, $\alpha = 5^\circ$, $Re = 5 \times 10^6$: output convergence history.

undershoots the exact value on the way to convergence. On the other hand, the neural network approaches all perform better than MOESS by the second DOF target and hone in closest to the exact value on the finest meshes compared to Hessian and MOESS anisotropy. We note that the neural network results are not identical, and that the output differences on the coarsest DOF target are large.

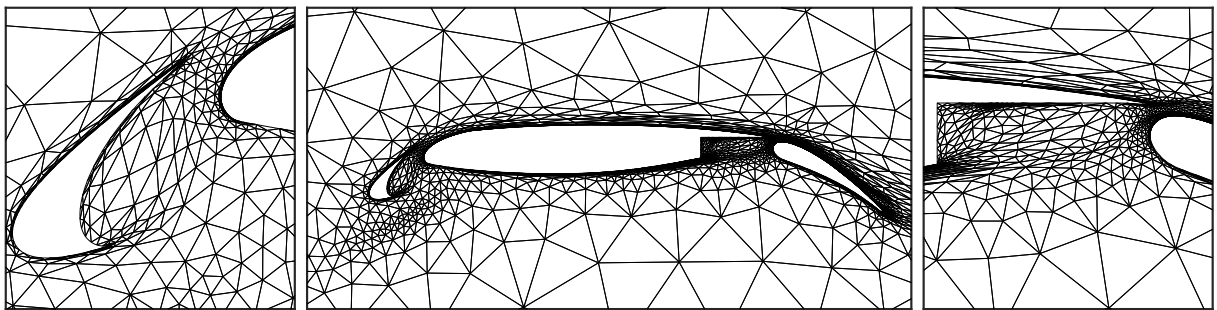
Figure 8.10 shows the final adapted meshes at the highest target DOF for the adaptive approaches tested. The most notable differences are in the placement of anisotropy on the leading-edge stagnation streamline and near the lower corner of the main-element cove. Anisotropy on the leading-edge streamline is strongest with MOESS, diminishes slightly with the neural-network method, and is nonexistent for the Hessian approach. This is consistent with the observation that the anisotropy in this region is driven by features of the adjoint, to which the Mach-number Hessian is agnostic. On the other hand, the Hessian approach places anisotropic elements in the region of flow coming off the lower surface of the main element, into the cove. This anisotropy is driven by the edge of the boundary layer, in which the Mach number variation is large. MOESS and the neural network approaches do not fixate on this region, indicating that high element stretching is not required there.

8.3.4 Extrapolation: Tandem NACA 5410 Airfoils

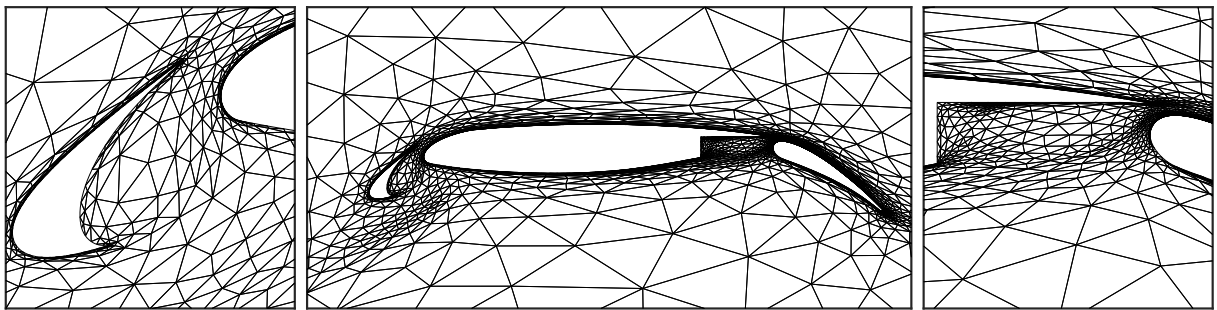
The previous tests demonstrated the ability of the neural network approaches to correctly detect anisotropy in cases used for training. Given the complexity of the training



(a) MOESS



(b) Hessian



(c) neural network A

Figure 8.10: MDA 30P/30N airfoil: final adapted meshes.

flow fields and the relatively small sizes of the networks, the successful performance of the networks was by no means guaranteed. Nevertheless, the generalizability of the network must be assessed on an extrapolation case, which is one for which the network was not trained.

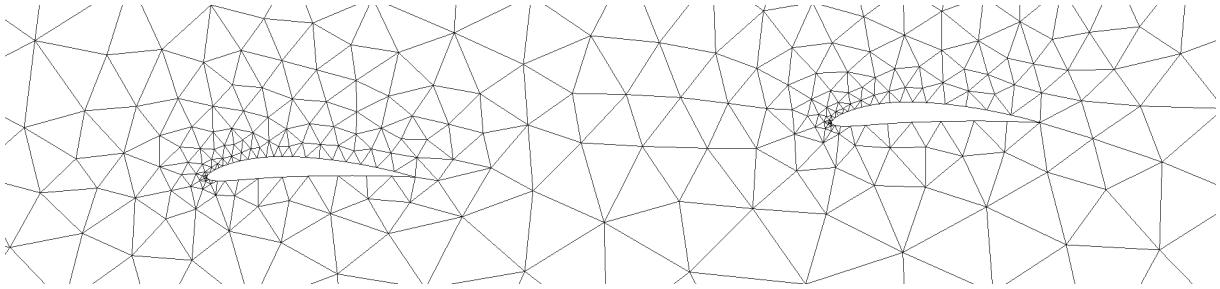
For this test, we choose a completely new geometry: two NACA 5410 airfoils separated by approximately two chord lengths, as shown in Figure 8.11. The farfield boundary is a square, 100 chords away from the airfoils. The flow conditions are also different from the previous tests: $M_\infty = 0.3$, $\alpha = 5^\circ$, and $Re = 2 \times 10^6$. A hand-generated initial isotropic mesh, shown in Figure 8.11a, of 1250 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution. The output of interest is the lift coefficient on the second airfoil. Figure 8.11b shows the Mach contour and Figure 8.11c shows the conservation of x -momentum component of the corresponding adjoint. Note the high variation of the adjoint along the leading-edge stagnation streamline of the second airfoil.

Adaptive simulations are performed at $p = 2$ solution approximation order for three target DOF costs: 8000, 16000, 32000. At each target DOF, 10 adaptive iterations are performed, using the final mesh from the previous target DOF as the starting mesh. As in the previous cases, we compare MOESS, Mach-number Hessian anisotropy, and the neural-network approaches.

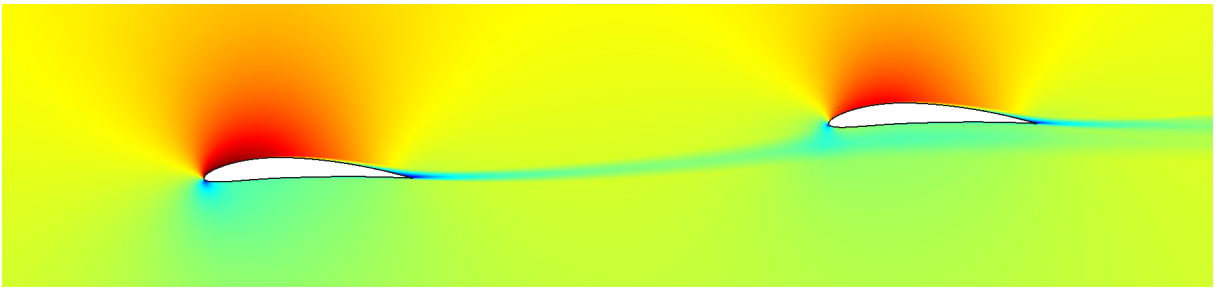
Figure 8.12 presents the output convergence history obtained from the adaptive runs, where again averaging has been performed over the last 4 iterations at each target DOF. The “exact” value is also indicated, obtained from a four-times finer-mesh run at a higher approximation order of $p = 3$.

We see that at all target DOF, the adaptive results using the neural-network approaches are closer to the exact value compared to those using MOESS and the Mach number Hessian anisotropy. Indeed, in this case, both MOESS and the Hessian-based methods perform comparably, at least in terms of output convergence, even though the adapted meshes, shown next, are different. The clustering of the neural network approaches indicates similar performance.

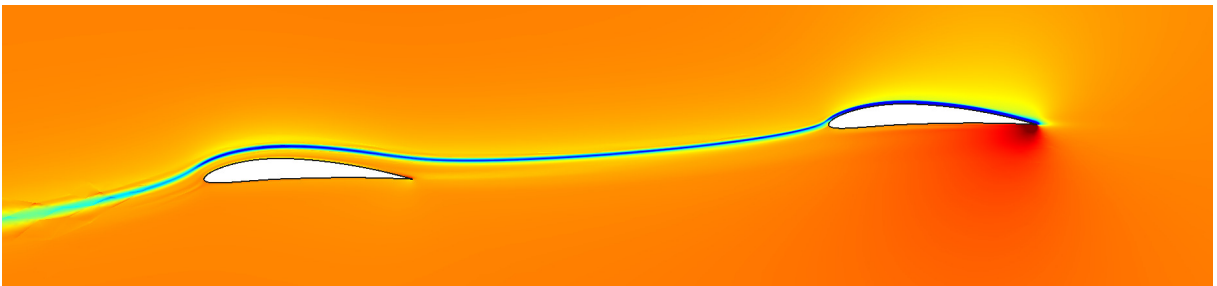
Figure 8.13 shows the final adapted meshes at the highest target DOF for several of the adaptive approaches tested. As in the previous tests, the meshes among the approaches differ in their placement of anisotropic elements. The Mach number Hessian approach focuses on the primal anisotropy, in the boundary layer and wake regions. It particularly targets with anisotropy the initial wake of the second airfoil, an area not as much resolved by the other methods. On the other hand, MOESS addresses both the primal and adjoint anisotropy features. MOESS particularly targets the leading-edge stagnation streamline



(a) initial mesh



(b) Mach number



(c) conservation of x -momentum adjoint

Figure 8.11: Tandem NACA 5410 airfoils: initial mesh and primal/adjoint solutions.

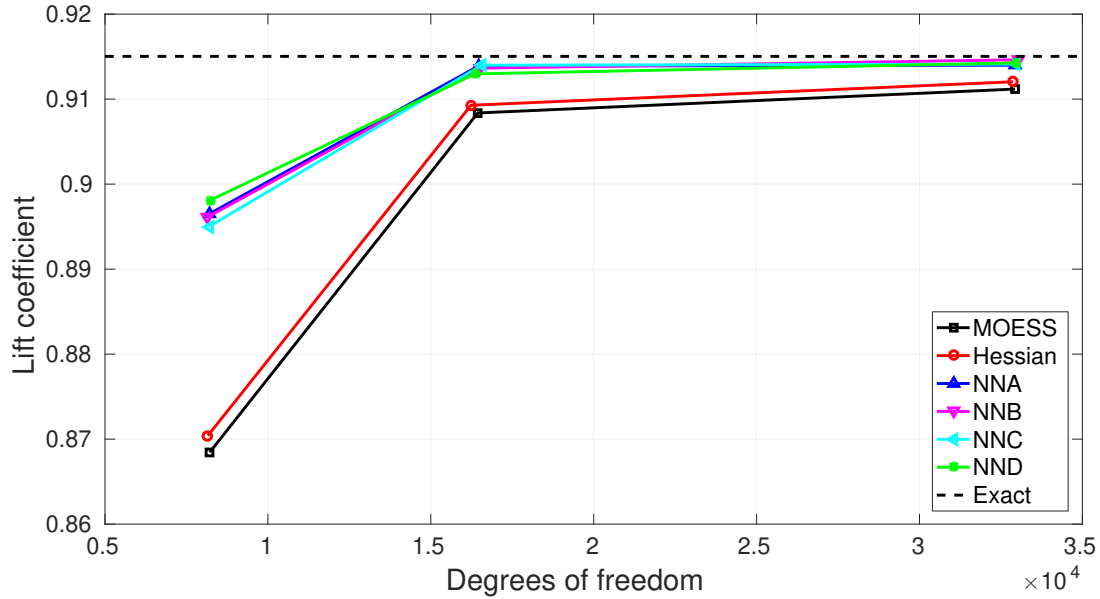


Figure 8.12: Tandem NACA 5410 airfoils: output convergence history.

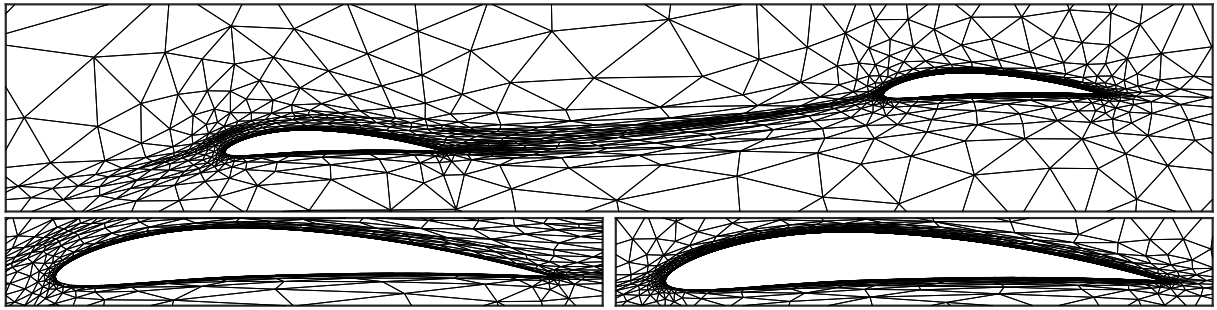
of the aft airfoil, shown in the adjoint contours in Figure 8.11c, with anisotropic elements. The Hessian-based method places isotropic elements in these regions, with anisotropy between the airfoils driven only by the Mach number variation in the wake of the first airfoil.

Finally, the neural network approaches, which yield similar meshes among themselves, produce anisotropy distributions that are similar to MOESS, but with somewhat lower emphasis on the adjoint anisotropy in the leading-edge stagnation streamline. This is likely due to a regularization property of the neural networks, which are small relative to the amount of training data, and which employ an activation function that saturates. As a result, their predictions are not as sensitive to outlier inputs, such as very high adjoint or primal anisotropy.

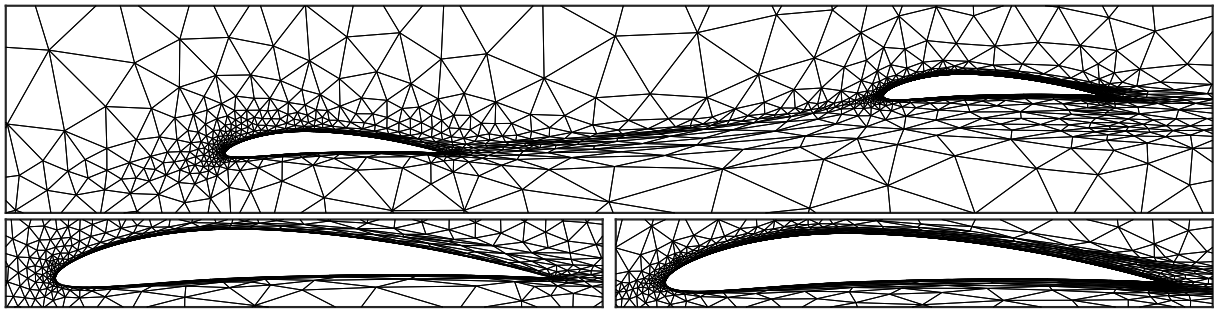
8.3.5 Adaptive Iteration Comparison

A practical consideration of adaptive methods is their cost. The final meshes produced can be highly optimized, but their generation requires multiple adaptive iterations. Each such iteration requires primal and adjoint solutions, which are relatively inexpensive on coarse meshes but grow in cost as refinement proceeds. Ideally, an adaptive method should not need many solutions on fine meshes before honing in on the optimal mesh. In this section, we compare the various adaptive methods in terms of this cost measure.

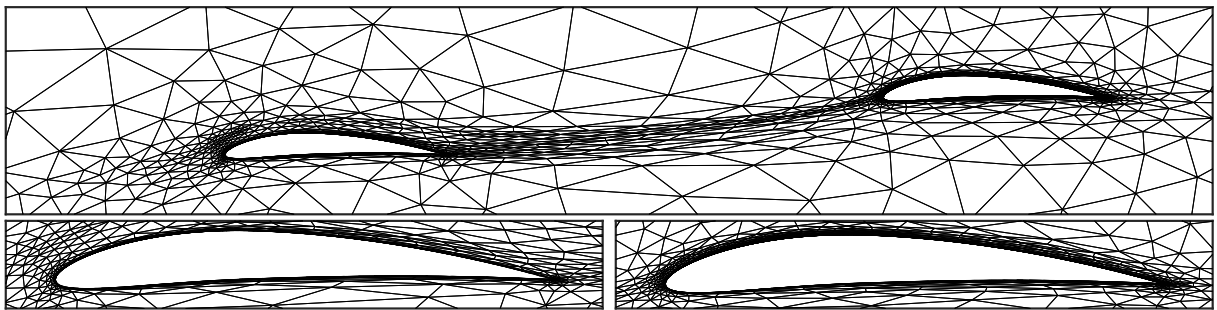
The solver settings, target degrees of freedom, and mesh growth factors are the same for all methods tested. Therefore, the measure of cost reduces to the number of iterations



(a) MOESS



(b) Hessian



(c) neural network A

Figure 8.13: Tandem NACA 5410 airfoils: final adapted meshes.

required to attain a certain error level. In the results thus far, 10 iterations were chosen empirically by monitoring outputs and ensuring that they had stabilized by the point of data collection, which was over the last four iterations. Restarts from previous target degrees of freedom provide good starting meshes, which limit the number of extra adaptive iterations needed.

In a production setting, of practical interest is the expense of generating the final adapted solution starting from a coarse initial mesh. We study this cost in terms of the number of adaptive iterations, using the same mesh growth factors for all methods. The demonstration case is the tandem airfoil simulation from the previous section, with the lift on the second airfoil as the output. The starting mesh is the coarse one shown in Figure 8.11a, which has 7.5k DOF for $p = 2$, the target mesh size is 32k DOF, and the growth factor set to 2.0.

Figure 8.14 shows the convergence of the output with adaptive iterations for all of the methods. Both the actual output and the error relative to a truth solution are shown. The output has not stabilized by the last four iterations for all methods, as this calculation proceeds straight from the initial mesh to the finest DOF target.

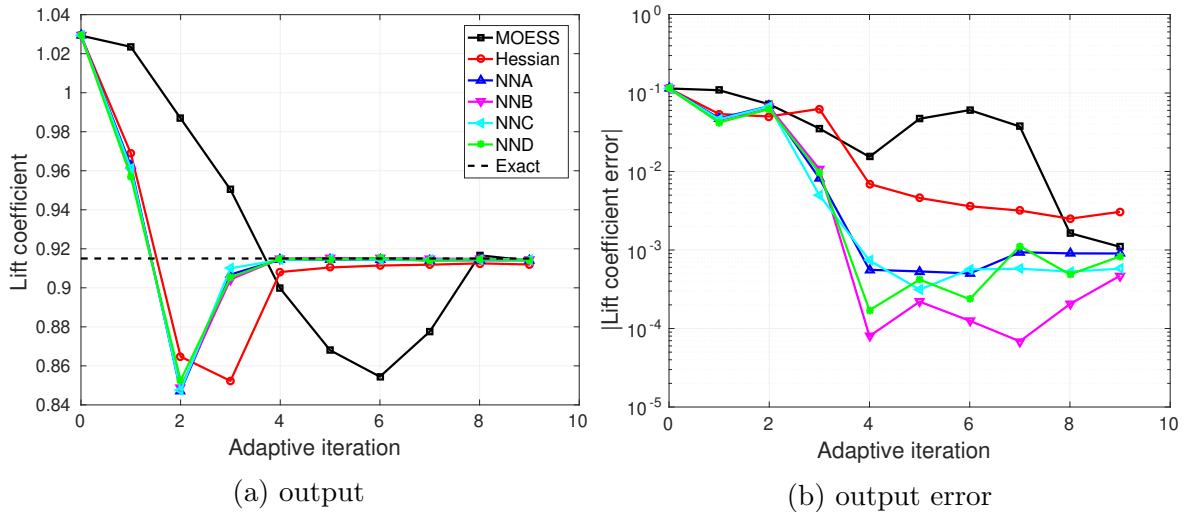


Figure 8.14: Tandem NACA 5410 airfoils: adaptive iteration performance.

From the figure, we see that MOESS exhibits the slowest output convergence. MOESS discovers the error model, specifically the rate tensor that governs anisotropy, through a sampling procedure. This discovery takes multiple adaptive iterations, as the error model is based on asymptotic analysis, and samples are computed relative to an incrementally finer space. On the other hand, both the Hessian and neural-network-based approaches discern the anisotropy from a direct mapping of the primal/adjoint solution features. As soon as the primal and adjoint fields are reasonably resolved, the anisotropy prediction

becomes accurate and drives the meshes more quickly to the final optimized mesh. We note, however, that the Hessian-based approach converges to a larger error compared to the neural network approaches, as expected from the data in the previous section. The neural network methods hone in on the optimal meshes quickly, by the fourth adaptive iteration in this test case, minimizing the number of solutions required at the finest DOF target.

Figure 8.15 compares the first four adapted meshes between MOESS and the neural network A approach. The side-by-side meshes in each row are at similar DOF. We see that MOESS discovery of anisotropy proceeds more slowly, with largely isotropic elements in the joining wake until the fourth adaptive iteration. On the other hand, anisotropic elements are already present in the neural network meshes by the second adaptive iteration. Combined with the higher resolution in the vicinity of the airfoils, the result is of faster iterative convergence to the optimal mesh.

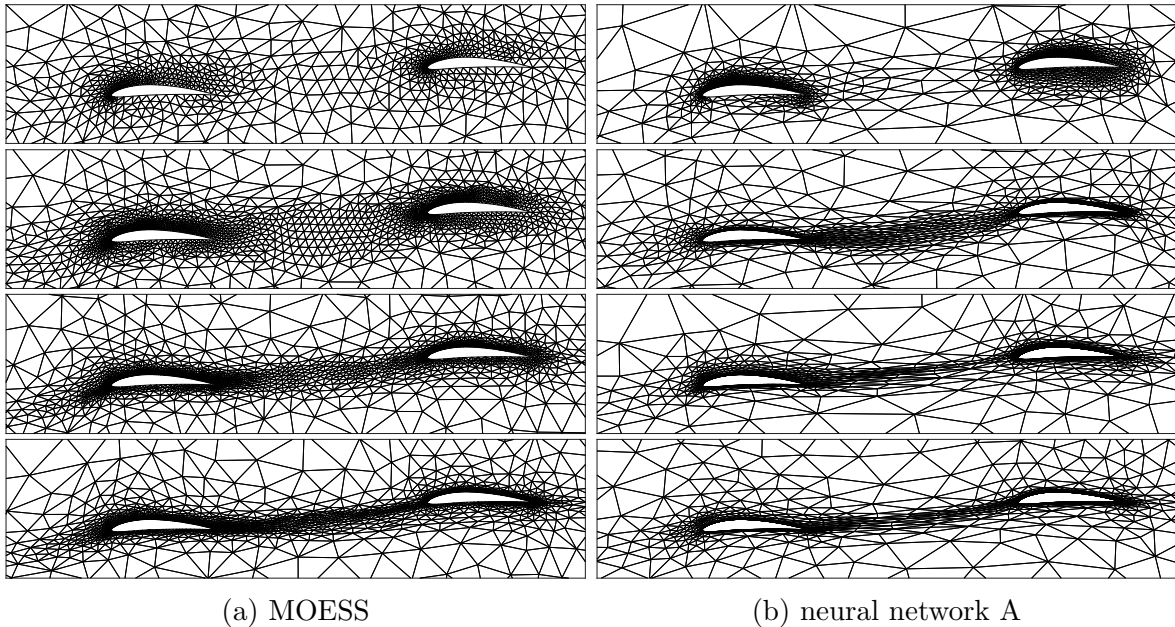


Figure 8.15: Tandem NACA 5410 airfoils: meshes at adaptive iterations 1 (top) through 4 (bottom).

8.3.6 $p = 3$ Results

All results so far were obtained using an approximation order of $p = 2$ for both training and deployment simulations. In this section we present, as an extension, a subset of corresponding $p = 3$ results. Recall that for $p > 2$, the state and adjoint fields are first least-squares projected to $p = 2$ approximation order within each element, in order to

obtain constant elemental Hessian matrices. This occurs during training and deployment feature calculations. The relative error indicators remain unaltered from the standard adjoint-weighted residuals, computed at order $p + 1$.

8.3.6.1 Network Training

The same flow cases that were used for training at $p = 2$, described in Section 8.2, are also used at $p = 3$. The training algorithm, including the optimizer, batch size, learning rate, and number of iterations, also remains the same. The resulting training loss history is similar to that shown in Figure 8.3. Figure 8.16 illustrates three out of the four trained networks, A, C, D. Note that the number of inputs, outputs, and hidden layer sizes does not change with p .

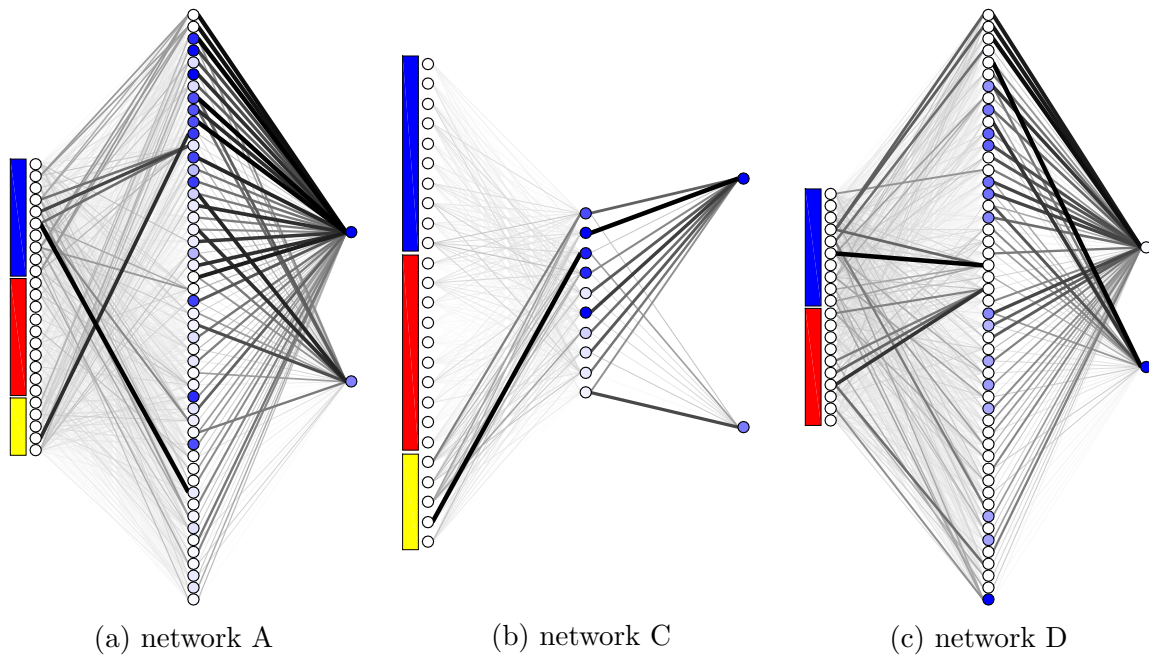


Figure 8.16: Visualization of the neural networks for $p = 3$. The format is the same as the $p = 2$ version in Figure 8.4.

Based on the connecting weights, we see that in network A, the baseline, the primal and error indicator features contribute relatively more to the output prediction than the adjoint features. In network C, which has the small hidden layer, the error indicator features become more important. Finally, in network D, which does not use the error indicators, the contributions from the primal and adjoint features are similar.

8.3.6.2 Tandem NACA 5410 airfoil test

We present one $p = 3$ result, the same extrapolation test case as in Section 8.3.4. The flow conditions, starting mesh, output of interest, and DOF targets remain unchanged. Figure 8.17 shows the convergence of the output, the lift coefficient on the aft airfoil averaged over the last four adaptive iterations, with degrees of freedom. The “exact” value is computed at $p = 4$ on a uniformly-refined version of the finest MOESS-adapted mesh.

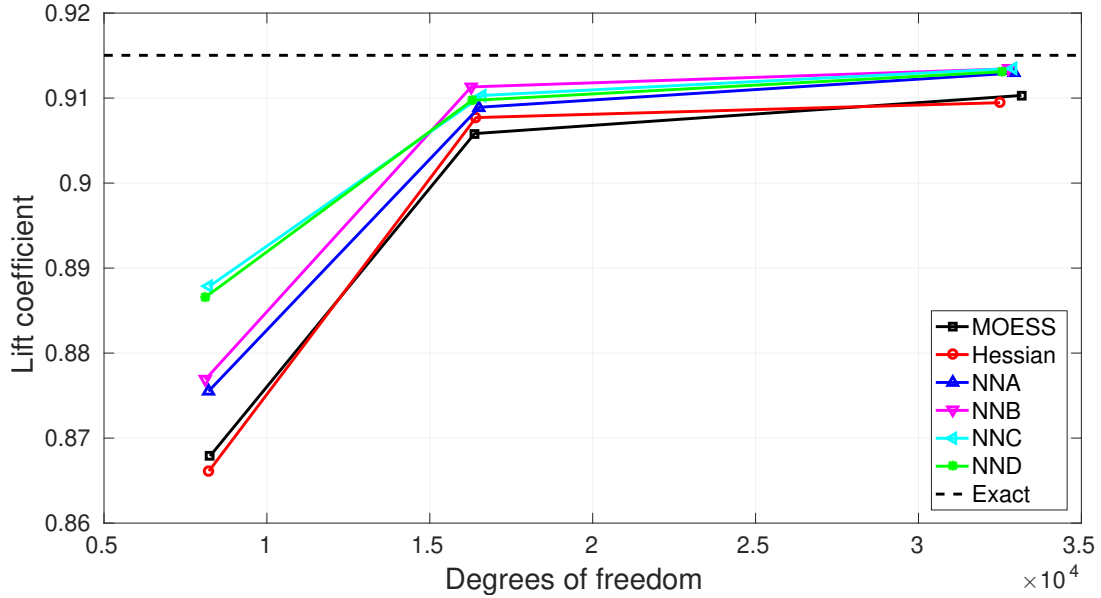


Figure 8.17: Tandem NACA 5410 airfoils: $p = 3$ output convergence history.

We see that the neural network results are similar and that they yield outputs closer to the exact result for all DOF targets. Both MOESS and the Hessian-based methods perform comparably, even though the adapted meshes, are different.

Figure 8.18 shows the final adapted meshes at the highest target DOF. These are coarser than the $p = 2$ ones in Figure 8.13, due to the higher DOF per element for $p = 3$. We see similar differences in the anisotropy placement. First, MOESS heavily targets the stagnation streamline for the aft airfoil, with high aspect ratio elements. The neural network mesh exhibits less anisotropy in this area, and the Hessian mesh places isotropic elements there due to a lack of anisotropy in the Mach number ahead of the airfoil. At the trailing edge of the aft airfoil, the Hessian-based mesh exhibits heavy anisotropic refinement, due to Mach number variations in the boundary layer and wake. On the other hand, the meshes from MOESS and the neural network show less refinement due to lower anisotropy. Overall, as for $p = 2$, the neural network approaches, which again yield similar meshes among themselves, produce anisotropy distributions that are similar

to MOESS, but with lower emphasis on the adjoint anisotropy features.

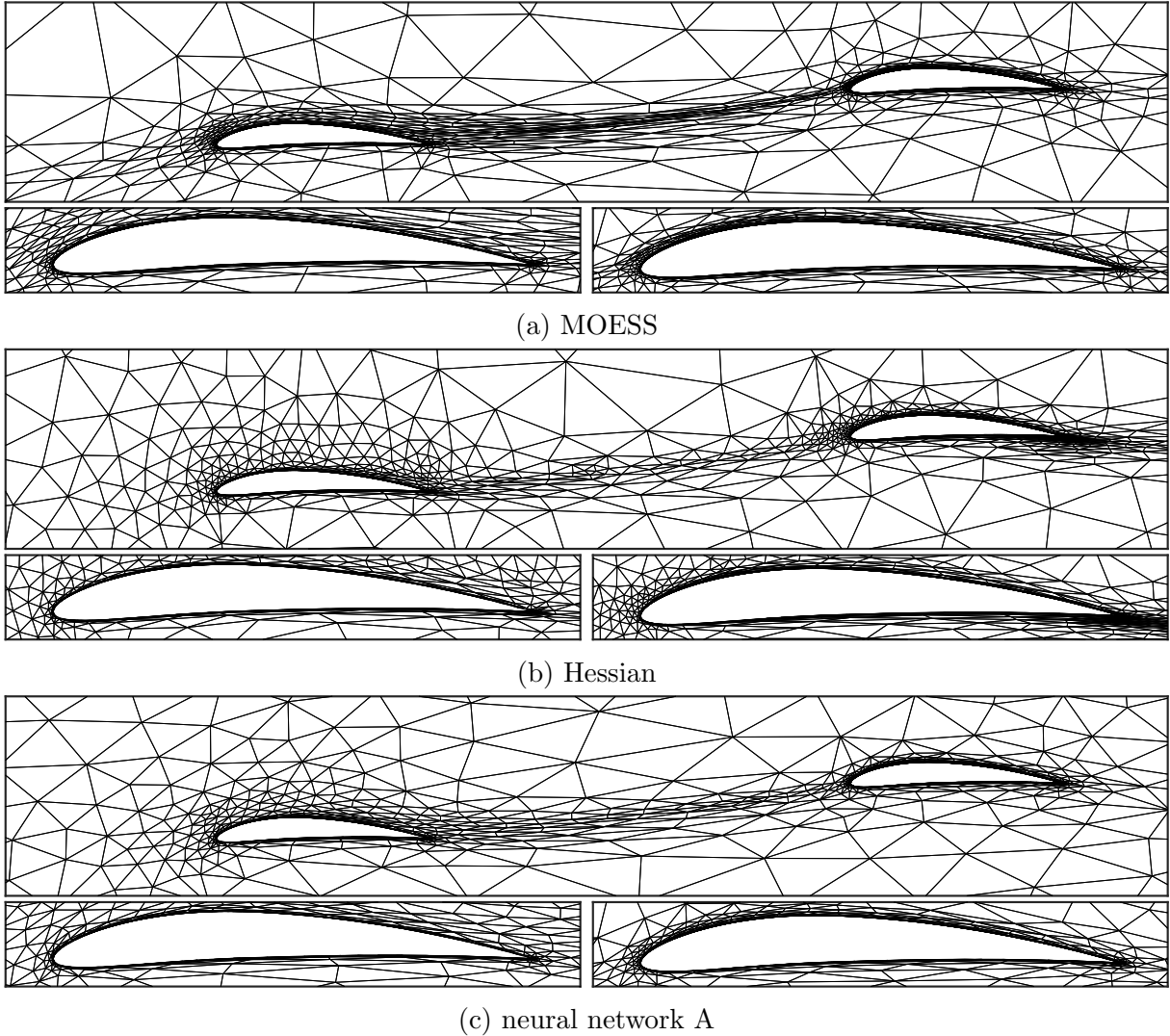


Figure 8.18: Tandem NACA 5410 airfoils: $p = 3$ final adapted meshes.

As in Section 8.3.5, we also test the rate at which the different methods converge to the final meshes for $p = 3$. The initial mesh has 12.5k DOF for $p = 3$, and the target mesh size is 32k DOF, with a growth factor of 2.0. Figure 8.19 shows the convergence of the aft airfoil lift output with adaptive iterations. We again see a more rapid convergence of the output with the neural network methods compared to MOESS, and we attribute this to the direct feature-to-anisotropy map in the neural-network methods, as opposed to the error model discovery of MOESS.

Figure 8.20 shows a comparison of the adapted meshes for MOESS and the neural network A approach. As in the case of $p = 2$, we see that MOESS discovery of anisotropy proceeds more slowly, with largely isotropic elements in the joining wake until the fourth

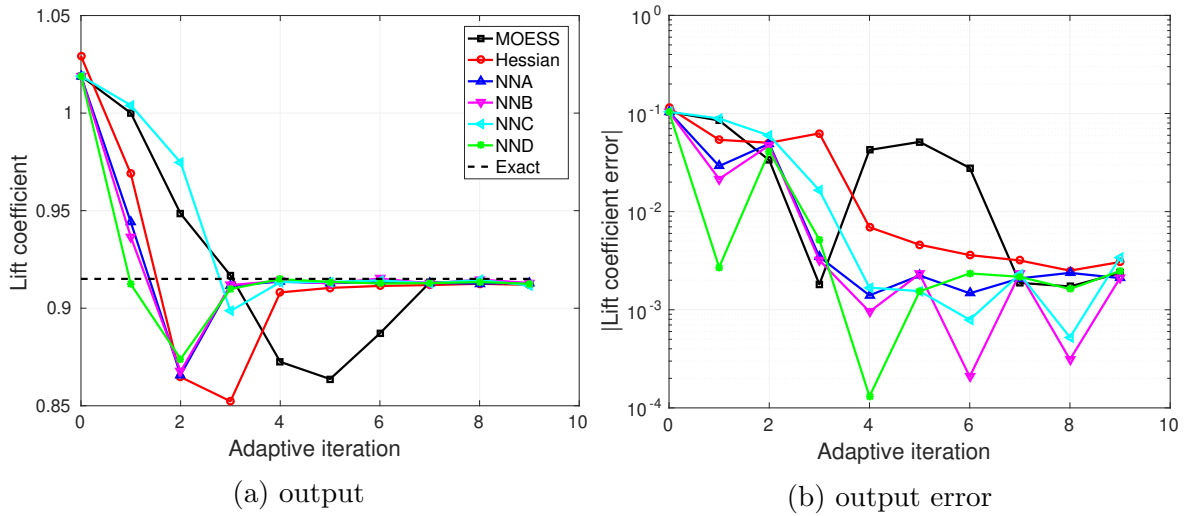


Figure 8.19: Tandem NACA 5410 airfoils: $p = 3$ adaptive iteration performance.

adaptive iteration. In contrast, anisotropic elements are present in the neural network meshes by the second adaptive iteration, resulting in faster iterative convergence to the optimal mesh.

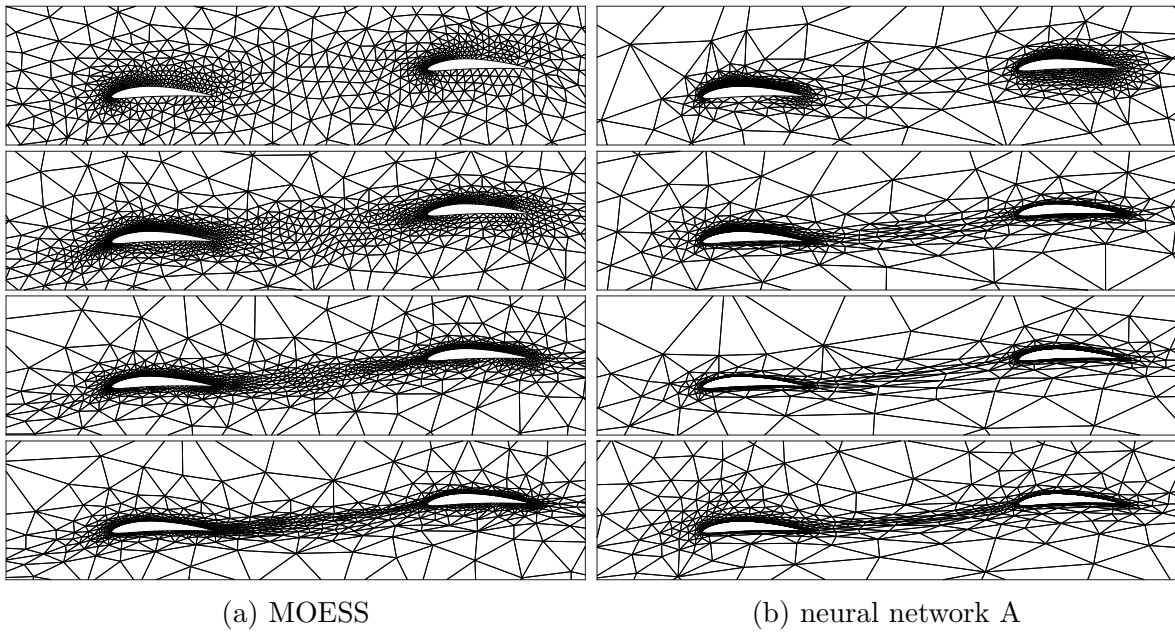


Figure 8.20: Tandem NACA 5410 airfoils: $p = 3$ meshes at adaptive iterations 1 (top) through 4 (bottom).

8.4 Summary

In this chapter, we introduce a machine-learning approach for determining optimal mesh anisotropy in an output-based adaptive setting. An artificial neural network is used to predict the desired element stretching ratio and direction from features of the primal and adjoint solutions. The network is trained to reproduce anisotropy calculated by MOESS, using features that are more easily calculated than the mesh-refinement sampling procedure of MOESS. These features consist of aspect ratio and direction data computed from the Hessian of each of the primal and adjoint variables, and from normalized error indicators of each of the state components. The features are invariant to scaling of the equations or outputs, and hence generalizable across different flow conditions, outputs, and unit choices.

Trained on a large amount of data from prototypical aerodynamic flow cases, the network model is able to closely follow the MOESS results in the deployment, *i.e.*, producing similar meshes and close output errors in an adaptive process, which is in general more effective than pure Mach number Hessian-based anisotropy detection. In many of the cases tested, the neural networks yielded outputs that, for a given degree-of-freedom target, were even more accurate than MOESS. This is in spite of the fact that the networks were trained with MOESS data. The explanation for this is that the networks possess a regularization property: the networks are small relative to the amount of training data and employ an activation function that saturates. As a result, their predictions are not as sensitive to outlier inputs, such as very high adjoint or primal anisotropy. This generally results in lower aspect ratios in regions of large primal or adjoint variation.

Based on the analysis of the adapted meshes, we observe that MOESS appears to over-emphasize regions of adjoint anisotropy, such as leading-edge stagnation streamlines, whereas the Mach number Hessian-based approach appears to over-emphasize regions of primal anisotropy, such as boundary layers and wakes. Although the element sizing information is driven by a common error estimate, a sub-optimal anisotropy prediction will yield inefficient flow-field resolution in these regions, requiring more mesh elements, or equivalently higher error for a fixed degrees of freedom. The neural network approaches, being less sensitive to sharp primal/adjoint features, can then perform better than either MOESS or the Hessian approach.

Furthermore, the neural network methods can more rapidly converge to the final adapted meshes compared to MOESS, at an expense close the Hessian-based adaptation. Whereas MOESS discovers the error model, specifically the rate tensor that governs anisotropy, through a sampling procedure, which can take multiple adaptive iterations,

both the Hessian and neural-network-based approaches discern the anisotropy from a direct mapping of the primal/adjoint solution features. When these fields are reasonably resolved, the anisotropy prediction becomes accurate and drives the meshes more quickly to the final optimized mesh. These final meshes are more accurate for the neural network methods compared to the Hessian approach.

CHAPTER 9

Adjoint-Free Error Estimation and Mesh Adaptation Using Convolutional Neural Networks

So far the error estimation and mesh adaptation that have been used in this work are all based on the output adjoint solutions. Although in Chapter 8, artificial neural networks (ANNs) are used to predict the mesh anisotropy, the element sizing still relies on the adjoint weighted residual (AWR). Due to the effectiveness of localizing the output error and identifying the important regions for adaptation, AWR has been widely used in aerospace CFD applications [85, 97, 98, 99, 115, 101, 207] to improve the simulation accuracy and efficiency. Despite the great success in CFD applications, the additional computational cost and implementation complexity associated with adjoint-based methods cannot be neglected. On the one hand, AWR requires solving a dual linear system (adjoint equations) of the same size (current space), or larger when solving on enriched (fine) spaces. Although this cost can be mitigated in problems where the adjoint solutions on current space are solved regardlessly, such as gradient-based optimization; yet for problems like unsteady simulations, uncertainty quantification or optimization without gradients, additional costs are added and can be accumulated in these problems when the adjoint is repeatedly solved. On the other hand, the implementation of the adjoint method requires the transpose of the residual Jacobian matrix, which is not always available in explicit solvers or Jacobian-free methods [208]. In these circumstances, either the continuous adjoint equations should be derived and directly discretized [129] or special implementation efforts are required [209], adding considerable costs and efforts in the development. The additional computational costs associated with the adjoint solves, in addition to the implementation efforts, have largely hindered the effective use of adjoint-based error estimation and the corresponding adaptation techniques in practice.

Motivated by the success of predicting mesh anisotropy using artificial neural networks, we explore the possibility of estimating the output error and adapting the mesh using

machine learning techniques, without solving for the adjoint variables. Given the data of adaptive flow simulations guided by adjoint-based error estimation, an error surrogate model is trained to predict the output error as well as the localized error indicator field, with only the low-fidelity solution as an input. For simplicity, mesh anisotropy is not considered here, although it can also be incorporated through Mach number Hessian or can be built in the surrogate model, which will be studied in the future. The goal for the current work is to generalize the error modeling knowledge from the adaptive simulation data at hand. More recently, error surrogate models based on machine learning techniques have received much attention, largely because of their non-intrusive nature and fast on-line evaluations. Several contributions have been made in error modeling for parameterized reduced-order models (ROMs) [102, 103], and the ideas have been extended to estimate discretization-induced errors [104]. Efforts have also been devoted to predicting the errors in flow solutions and the outputs of interest obtained on coarse computational meshes [105, 106], and the models have been used to guide the selection of a set of *a priori* meshes [107]. Nonetheless, in these studies, no output error indicator is provided to perform mesh adaptation. Manevitz *et al.* used neural networks to predict the solution gradients in time-dependent problems, which then provided an indicator to drive the mesh adaptation [108]. However, feature-based adaptive indicators are generally not as effective as adjoint-based indicators, especially for functional outputs and problems with discontinuities [109, 101]. Furthermore, these works rely on a set of user-selected local features (feature engineering) to construct the model, requiring either expert knowledge or fine tuning. Moreover, due to the local nature of the selected features (although some neighboring information comes in with the gradient features), these models either largely ignore the error transport, and thus are not expected to be effective for convection-dominated problems, or still require the adjoint variables to bring in the global sensitivity information.

In this chapter, we focus on inferring the output error for a CFD simulation, as well as the corresponding localized error indicator field to drive mesh adaptation, directly from the solution field without access to the adjoint variables. Suppose that we want to predict elemental error indicator given a solution field, artificial neural networks can be potentially used to construct a local surrogate model, which takes in some local features defined *a priori* and output the local error indicator. However, in order to take into account the error transport, the network input features ideally should be chosen along the characteristics, which makes it almost impossible to pick a consistent set of local features *a priori* for nonlinear problems. Alternatively, we can use the entire solution field as the ANN input, and build a global error model to predict the entire error indicator

field. Nevertheless, the dimension (number of parameters) of the network grows fast as the input and output dimensions increase due to the fully-connected network structure. As a result, the network training quickly becomes computationally infeasible for problems with high-dimensional input and output.

As mentioned in Chapter 3, the adjoint variables can be regarded as a generalized *Green's function*, which convolves a residual source to produce an output perturbation (error estimate). In order to emulate the adjoint operator, network architectures that involves convolution operations, namely the convolutional neural networks (CNNs), are used in this work to map the solution field to the error indicator field. Particularly, a set of linear convolution operators is trained to approximate the generalized *Green's function* which convolves the discretized solution to produce the corresponding error with respect to a refined space. In other words, the network can be regarded as an approximate adjoint-weighted-residual operator applied on the solution field, which produces the whole error indicator field as well as the total output error. More importantly, the convolution operators preserve the spatial locality and are shared for the input field, *i.e.*, the network is not fully connected. As a result, the dimension of the free parameters in the network model scales well for large-scale problems, making it well-suited for the high-dimensional map between the input solution and the output error indicator fields.

A CNN architecture that is specifically efficient for constructing this type of maps, which involves high-dimensional input and output vectors, is the encoder-decoder type CNN. It has shown excellent performance for image semantic segmentation and feature extraction in computer vision tasks [210, 211, 212, 213, 214], and has recently received much attention in physical modeling applications [215, 216, 217, 218] as well. The network is composed of two subnetworks: an encoder convolutional neural network (CNN) that extracts a low-dimensional representation (code) from the input data, *i.e.*, the solution field, followed by a decoder CNN that reconstructs the high-dimensional output field, *i.e.*, the adaptive error indicator field. The ability of CNN to automatically learn internal invariant features and multi-scale feature hierarchies alleviates the need for a tedious, hand-crafted feature engineering process, making this approach more flexible and robust. Instead of using the network output field to obtain the total output error, we connect the codes (low-dimensional representations) extracted from the input field to a fully connected network (FCN) to predict the total output error. The network training is supervised by both the adaptive error indicator field and the total output error to capture both the local and global features related to the numerical error. Since the two regression tasks are trained simultaneously, separate models and additional training costs are avoided.

The remainder of this chapter proceeds as follows. We describe the CNN-based er-

ror surrogate model in Section 9.1, including the details of the network architecture and the training procedure. The proposed network architectures are then applied in a simple two-dimensional advection-diffusion problems shown in Section 9.2. More complicated aerodynamic flow simulation over airfoils are considered in Section 9.3, and a brief summary of the present work is in Section 9.4.

9.1 CNN-Based Model for Output Error Estimation

9.1.1 Parameterized PDEs and Output Error Estimation

In this work, we consider parameterized governing PDEs in a fully-discretized form following the notation in Chapter 2,

$$\mathbf{R}_h(\mathbf{U}_h(\boldsymbol{\mu}); \boldsymbol{\mu}) = \mathbf{0}, \quad (9.1)$$

where $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$ is a vector of parameters sampled from the parameter space \mathcal{D}_μ , characterizing the physics of the system, *e.g.*, initial and boundary conditions, material properties, or shape parameters in a design optimization problem; $\mathbf{U}_h \in \mathbb{R}^{N_u}$ denotes the flow state vector, uniquely defining the continuous flow state field $\mathbf{u}_h \in \mathcal{V}_h$, where \mathcal{V}_h is the approximation space defined by a finite-dimensional discretization, denoted by h ; and $\mathbf{R}_h : \mathbb{R}^{N_u} \times \mathbb{R}^{N_\mu} \rightarrow \mathbb{R}^{N_u}$ is a nonlinear residual vector, which implicitly defines \mathbf{U}_h as a function of the parameter vector, $\mathbf{U}_h(\boldsymbol{\mu}) : \mathbb{R}^{N_\mu} \rightarrow \mathbb{R}^{N_u}$.

Often in engineering applications, the quantities of particular interest are the scalar outputs such as drag or lift, defined as,

$$J_h \equiv J_h(\mathbf{U}_h(\boldsymbol{\mu}), \boldsymbol{\mu}) = S_h(\boldsymbol{\mu}). \quad (9.2)$$

$J_h : \mathbb{R}^{N_u} \times \mathbb{R}^{N_\mu} \rightarrow \mathbb{R}$ represents the explicit map to the scalar output from the discrete state vector and the parameter vector; while $S_h(\boldsymbol{\mu})$ denotes the direct map between the parameter vector and the output, whose form is fairly complicated and generally intractable explicitly. As discretization error always appears in Eqn. 9.1 on a finite-dimensional space and affects the calculation of the state vector \mathbf{U}_h , the resulting error in the output has to be quantified and the mesh has to be adapted accordingly to ensure the accuracy of the output of interest.

In practice, it is generally not possible to obtain the true discretization error of an output, since the exact infinite-dimensional solution is often inaccessible. Instead, followed the definition in Chapter 3, we use the difference between outputs evaluated on a coarse

approximation space \mathcal{V}_H and on a relatively finer space \mathcal{V}_h as an estimate of the output error,

$$\text{output error: } \delta J \equiv J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h). \quad (9.3)$$

The subscripts H and h denote the coarse and fine spaces, respectively. However, the error estimate in Eqn. 9.3 is hardly used in practice, as it requires the state vector solution on the finer space, and more importantly the resulting error cannot be localized to guide the mesh adaptation. Instead, an adjoint variable is used to bypass the expensive solve for \mathbf{U}_h on the finer space, and to provide localized error in each mesh element to drive the mesh adaptation.

For a given output, the associated adjoint vector, $\Psi_h \in \mathbb{R}^{N_u}$, can be used to weight the residual perturbation to produce an output perturbation, such that the output error can be estimated as,

$$\begin{aligned} \delta J &= J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \\ &= J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx \frac{\partial J_h}{\partial \mathbf{U}_h} \delta \mathbf{U} \\ &= -\Psi_h^T \delta \mathbf{R}_h = -\Psi_h^T [\mathbf{R}_h(\mathbf{U}_h^H) - \mathbf{R}_h(\mathbf{U}_h)] \\ &= -\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H), \end{aligned} \quad (9.4)$$

where \mathbf{U}_h is the (hypothetical) exact solution on the fine space, and \mathbf{U}_h^H is the coarse state injected into the fine space, which generally will not give a zero fine-space residual, $\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}$. Eqn. 9.4 gives a first-order approximation of the output error and is valid when the residual perturbations are small. Furthermore, the output definition is assumed to be unchanged between the coarse and fine spaces, *i.e.*, $J_H(\mathbf{U}_H) = J_h(\mathbf{U}_h^H)$. In our implementation, Galerkin orthogonality, *i.e.*, $\Psi_H^T \mathbf{R}_H(\mathbf{U}_H) = \mathbf{0}$, is assumed to be consistent during the projection, and is subtracted from Eqn. 9.4 to account for remaining convergence errors in both the primal and adjoint solves on the coarse space,

$$\begin{aligned} \delta J &= -\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H) \\ &= -[\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H) - \Psi_H^T \mathbf{R}_H(\mathbf{U}_H)] \\ &= -[\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H) - (\Psi_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H)] \\ &= -[\Psi_h - \Psi_h^H]^T \mathbf{R}_h(\mathbf{U}_h^H) \\ &= -\delta \Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H). \end{aligned} \quad (9.5)$$

This form of the adjoint-weighted residual indicates that in regions where the fine space adjoint is well approximated by the coarse space, the contribution of local residual errors

to the output error will be small. More details about adjoint variables and the associated output error estimation can be found in Chapter 3.

9.1.2 Hanging-Node Mesh Adaptation

The inner product in Eqn. 9.5 can be localized to each element using the local adjoint vector to weight the local residual (perturbation) vector, which provides a measure of elemental contributions to the total output error,

$$\delta J = -\delta \Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H) = -\sum_{e=1}^{N_e} \delta \Psi_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \Rightarrow \mathcal{E}_e = |\delta \Psi_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)|, \quad (9.6)$$

where N_e is the total number of elements in the mesh, and the subscript e indicates the product restriction to element e . The adaptive error indicator \mathcal{E}_e is obtained by taking the absolute value of the elemental error contribution. The error indicator can then be used to drive mesh adaptation, actively controlling the discretization error to ensure output accuracy.

In this work, mesh adaptation is performed using a hanging-node refinement strategy starting from an initially structured quadrilateral mesh [219, 119]. At each adaptive iteration, a fixed fraction of elements with the highest error indicators is targeted for refinement. Presently, we only consider isotropic refinement in which each quadrilateral element is subdivided uniformly into four sub-elements, although the method can be extended to anisotropic mesh adaptation as well [219]. The subdivision is done in the reference space and the physical mesh refinement is then determined through the reference-to-physical mapping. For high-order curved elements, the refined elements inherit the geometry order of the original unadapted element. For simplicity, we use curved elements of the same geometry order throughout the computational domain. Note that elements created in a hanging-node refinement can be marked for subsequent adaptation again. In these cases, neighbors of the adapted elements will also be cut to keep a maximum of one level of refinement difference between adjacent elements. Figure 9.1 presents an illustration of the adaptation mechanics used in this work.

Although effective for output error estimation and mesh adaptation purposes, it requires solving the linear adjoint equation, Eqn. 3.26, exactly or approximately on the finer space, which has the same dimension as the fine-space flow problem. These additional solves can add non-negligible costs in unsteady problems or in a many-query setting. Moreover, Eqn. 3.26 requires the transpose of the residual Jacobian matrix, $\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}$, which is not always available in Jacobian-free methods or if explicit time integration schemes are

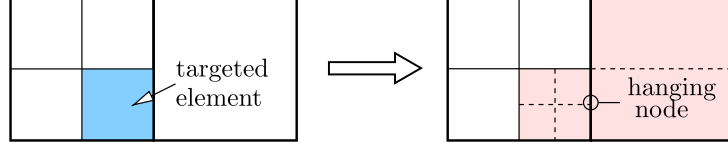


Figure 9.1: Hanging-node adaptation for a quadrilateral mesh, figure reproduced from [220]. The blue element on the left mesh is targeted for adaptation, while to keep a maximum of one level of refinement difference, the adjacent element is also refined as shown in the right mesh.

used. As a result, dedicated adjoint implementation efforts are required for these systems. In this work, we avoid the adjoint implementation and reduce the computational cost by directly constructing the maps from the injected flow state vector \mathbf{U}_h^H to the output error δJ and the error indicator field $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{N_e}\}$.

9.1.3 Surrogate Model as a Regression Problem

The error surrogate model can be treated as two regression problems: given the input solution vector from a CFD simulation $\mathbf{U} \in \mathbb{R}^{N_u}$, we would like to predict the scalar output error δJ as well as the adaptive indicator field \mathcal{E} over the entire mesh. Here we omit the subscript h for simpler exposition. The output and input dimensions can be very different; for example in a finite-element simulation, the state vector can be post-processed into several state components of the same dimension $\mathbf{U}^T = [\mathbf{U}_1^T, \mathbf{U}_2^T, \dots, \mathbf{U}_{N_f}^T]$, where N_f is the state rank. We call these state components *channels* following the convention in computer vision. For each channel we have $\mathbf{U}_i \subseteq \mathbb{R}^{N_p \times N_e}, \forall i = 1, 2, \dots, N_f$, where N_e is the number of elements in the mesh and N_p is the degrees of freedom (DOF) per element of approximation order p (assumed to be the same everywhere in the mesh). The solution vector arrangement and dimension details for DG can be found in Chapter 2. On the other hand, the error indicator field \mathcal{E} is of the same dimension as the mesh size, $\mathcal{E} \in \mathbb{R}^{N_e}$. The input (each channel) and output can be made to have the same dimension, either by averaging the state vector over each mesh element on every individual channel i , $\hat{\mathbf{U}}_i \equiv \mathbf{P}\mathbf{U}_i \in \mathbb{R}^{N_e}$ (\mathbf{P} is the averaging or projection operator), or by further localizing the error estimate in Eqn. 9.6 into each degree of freedom in every mesh element. Nonetheless, their dimensions are not required to be the same in the proposed method.

Although the solution is obtained in a vector form, it is usually interpreted as a field variable on the computational domain. Consider a flow problem solved on a two-dimensional rectangular mesh with H elements in height and W elements in width, *i.e.*,

$H \times W = N_e$, the regression functions we are seeking can be written as

$$\widetilde{\delta J} = f_{\text{error}}(\mathbf{U}) : \mathbb{R}^{N_f \times H_{\text{in}} \times W_{\text{in}} \times N_p} \rightarrow \mathbb{R}; \quad \widetilde{\mathcal{E}} = \mathbf{f}_{\text{indicator}}(\mathbf{U}) : \mathbb{R}^{N_f \times H_{\text{in}} \times W_{\text{in}} \times N_p} \rightarrow \mathbb{R}^{H_{\text{out}} \times W_{\text{out}}}, \quad (9.7)$$

where H_{in} and W_{in} are the height and width of each input channel, while N_f (state rank) denotes the number of channels, or alternatively denoted as depth of the input D_{in} , $D_{\text{in}} = N_f$. The input dimension of each channel depends on the mesh size, the approximation order, and the operator \mathbf{P} if projection is applied. For the former output error model in Eqn. 9.7, the model output is a scalar; while for the latter adaptive indicator model, the model output is a single-channel field of height H_{out} and width W_{out} since the output error is localized into a scalar in each element. If other types of error localization are used, the model output can have multiple channels or the dimension of each channel can be higher than the mesh size. If we interpret the solution field of each component (channel) as an image, then the first map f_{error} is an image-wise prediction often considered in image classification problems, while the latter map $\mathbf{f}_{\text{indicator}}$ is a pixel-wise prediction in image semantic segmentation tasks. The main difference is that in computer vision applications, the inputs and outputs are often integer-valued, while they are generally real-valued in physical systems. In this work, the two regression tasks are combined in a single encoder-decoder type CNN and are trained simultaneously, which is presented in details in Section 9.1.4 and Section 9.1.5.

9.1.4 Convolutional Neural Networks

Traditional ANNs/FCNs can be inefficient for high-dimensional problems as each neuron is directly connected to all the neurons in the previous layer and the layer after, *i.e.*, the network is fully connected. In other words, the fully-connected structure forces the hidden neurons to learn global features spanning the entire visual field (output from the previous layer), which introduces redundancy in the network parameters and poses challenges in the network training. Convolutional neural networks (CNNs) were introduced in the 1990's as a variant of traditional ANNs/FCNs, by taking into account the input spatial information [221]. The convolution operations are designed to discover localized features that are spatially-invariant. Hence, only a small region of the previous layer (receptive field) is connected to each neuron and the corresponding weights and bias are shared over the entire visual field. As a result, the dimension of the free parameters (weights and biases) in the network does not depend on the dimension of the inputs and outputs and thus CNNs often scale well for high-dimensional problems. With the ability of automatically learning spatially-invariant features and its good computational efficiency

for high-dimensional problems, CNNs have demonstrated state of the art performance in many computer vision benchmarks and have become the dominant approach in pattern recognition [222, 223, 224].

A traditional CNN architecture is defined similarly to the FCNs in Section 8.1.3.1, with the difference that each fully-connected hidden layer is replaced with a layer containing a linear convolution with nonlinear activation (convolutional layer), and very often followed with a feature pooling layer. The essential convolutional layer follows the equation below,

$$\mathbf{h}_i^l = \sigma(\mathbf{W}_i^l \otimes \mathbf{h}^{l-1} + b_i^l) \equiv \sigma(\Theta_i^l \otimes \mathbf{h}^{l-1}), \quad i = 1, 2, \dots, D_l, \quad l = 1, 2, \dots, L; \quad (9.8)$$

where \otimes is the discretized convolution operation. Assume the dimension of hidden units (often called feature maps in CNN) from the previous layer \mathbf{h}^{l-1} is $H_{l-1} \times W_{l-1} \times D_{l-1}$, where H_{l-1} , W_{l-1} and D_{l-1} are the height, width and depth (channels) of the feature maps; The i^{th} convolutional filter \mathbf{W}_i^l of dimension $F_H \times F_W \times D_{l-1}$ is applied to \mathbf{h}^{l-1} with a shared bias term b_i^l (a scalar), where F_W , F_H and D_{l-1} characterize the filter height, width and the depth. Followed by a nonlinear activation σ similar to FCNs, the convolution layer produces a new feature map of the output layer (one single channel) $\mathbf{h}_i^l \in \mathbb{R}^{H_l \times W_l \times 1}$. The number of convolutional filters at layer l , D_l , determines the channels of the output feature maps, such that the feature maps at layer l is of dimension $H_l \times W_l \times D_l$. The convolution operation has the flexibility to deal with various input and output dimensions. The filter (receptive field) slides over the input domain with stride s to perform the convolution, which often down-samples the input layer \mathbf{h}^{l-1} . Zeros can be padded around the borders of the input layer to adjust the width and the height of the output feature maps. An example of a $3 \times 3 \times 1$ Laplacian-like convolution filter with bias $b = 1$ and stride $s = 1$ applied on a $4 \times 4 \times 1$ input feature map is demonstrated in Figure 9.2. No padding is applied in this case. A pooling operation is often used right after a convolutional layer, which further down-samples the input feature maps by extracting the max values (max pooling) or the averaged values (average pooling) of subregions (pooling filter) sliding through the input features with strides larger than 1. A traditional CNN uses nonlinear activation after the pooling layer while modern models often do not [225]. As the convolutional layer has the ability to do the down-sampling with bigger strides and less padding, the pooling layer is not always required and is not used in the current work.

Although CNNs are featured by the convolution operations, fully-connected layers are also used in most CNN architectures. In traditional CNNs, the last convolutional or pooling layer (last feature map) is reshaped to a vector and is connected to several fully-connected layers to perform the final classification or regression tasks. However, in

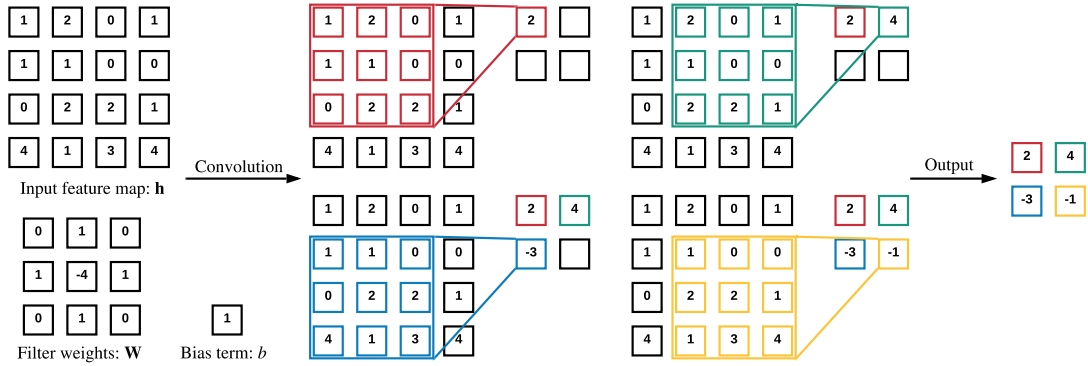


Figure 9.2: An example of convolution operations.

our error indicator prediction task, we would like to reconstruct an indicator field which requires an image-to-image regression. A paradigm for this type of problems in semantic segmentation [212, 213, 214] is the encoder-decoder network architecture shown in Figure 9.3. The intuition is that the high-dimensional inputs often lie on an embedded low-dimensional nonlinear manifold or latent space, specifically representative of the high-dimensional output field. Hence, an efficient way to find the map between high-dimensional inputs and outputs is to go through the latent space, featuring an encoder subnetwork to extract the high-level features (codes) from the input field, and a decoder subnetwork to construct the output field from the low-dimensional codes. The encoder subnetwork is a down-sampling process, often through convolution and pooling operations or sole convolutions. To reconstruct the high-dimensional output field, an up-sampling or deconvolution process has to be performed, either through transposed convolution [226], or using nearest-neighbor interpolation or bilinear interpolation [227]. The transposed convolution approach is used in this work.

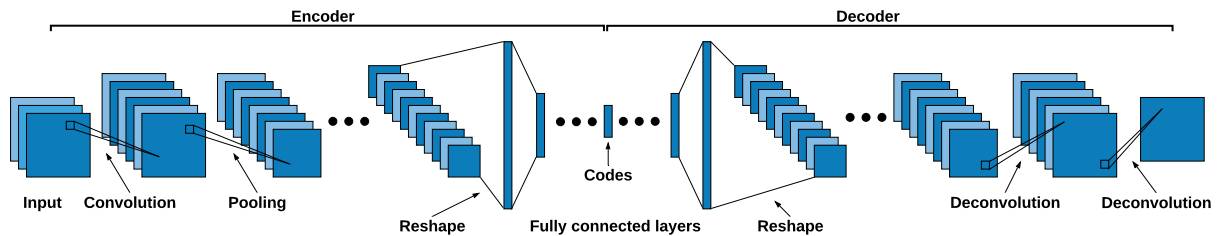


Figure 9.3: An example of encoder-decoder convolutional neural networks. The input dimension is reduced through convolution and pooling operations, followed by fully-connected layers to further reduce the feature dimension until the low-dimensional codes are obtained. The subnetwork performing this dimension reduction is the encoder part; the decoder part performs the opposite operations with fully-connected layers and deconvolution operations, increasing the dimension to reconstruct the output.

9.1.5 Proposed Architecture and Network Training

In the output error estimation and mesh adaptation problem, we would like to predict the error in the output as well as the localized error indicator field. Instead of constructing and training models separately for these two tasks, we propose a network architecture capable of learning the two maps simultaneously, as shown in Figure 9.4. The network consists of an encoder-decoder CNN to reconstruct the error indicator field and a FCN connected to the latent layer (codes) of the CNN for output error estimation. The encoder-decoder CNN is used to learn the latent features (codes) representative of the indicator field, while the regression FCN guides the learning of the latent space and the total output error as well. The network design is based on a simple assumption that the total output error and the error indicator field should share some embedded features in the inputs. The network is trained to minimize the loss of the reconstruction task in the decoder CNN, and the loss of the regression task in the FCN, together with a L_2 regularization penalty to avoid excessive over-fitting. The training process is then an optimization problem formulated as

$$\begin{aligned}\Theta^* &= \arg \min_{\Theta} L_{\text{net}} + \lambda_{\text{reg}} L_{\text{reg}} \\ &= \arg \min_{\Theta} L_{\mathcal{E}} + \lambda_{\delta} L_{\delta} + \lambda_{\text{reg}} L_{\text{reg}},\end{aligned}\tag{9.9}$$

where L_{net} denotes the total loss of the network, including the indicator prediction loss $L_{\mathcal{E}}$ and the output error prediction loss L_{δ} ; L_{reg} represents the regularization loss that penalizes all the network weights¹, including the linear map weights and the convolution filters, to avoid over-fitting. λ_{δ} and λ_{reg} are the weights for the output error prediction loss and the regularization loss, which are hyper-parameters² of the model. Θ in Eqn. 9.9 are the network trainable parameters, which consist of the encoder parameters Θ^{en} , decoder parameters Θ^{de} and the parameters in the fully-connected regression layer, Θ^{δ} . The superscript $*$ denotes the optimized parameters to the optimization problem. Consider N_d sample solutions, the indicator loss $L_{\mathcal{E}}$ and the output error loss L_{δ} can be written as

$$L_{\mathcal{E}} = \frac{1}{N_d \times N_e} \|\tilde{\mathcal{E}}^i - \mathcal{E}^i\|_F^2 = \frac{1}{N_d \times N_e} \|\mathbf{f}_{\text{indicator}}(\mathbf{U}^i; \Theta^{\text{en}}, \Theta^{\text{de}}) - \mathcal{E}^i\|_F^2,\tag{9.10}$$

$$L_{\delta} = \frac{1}{N_d} \|\tilde{\delta J}^i - \delta J^i\|_2^2 = \frac{1}{N_d} \|f_{\text{error}}(\mathbf{U}^i; \Theta^{\text{en}}, \Theta^{\delta}) - \delta J^i\|_2^2,\tag{9.11}$$

¹Sometimes the biases terms are also penalized, yet in this work we only penalize the network weights.

²Hyper-parameters are predefined and are not optimized during the training. They often have a big impact on the model performance and can be optimally chosen through cross-validation.

where quantities with $\tilde{\cdot}$ indicate the predicted values of the model, while those without $\tilde{\cdot}$ are the ground-truth values from the training data. The regularization loss L_{reg} takes the form of

$$L_{\text{reg}} = \sum_l \frac{\|\mathbf{W}^l\|_F^2}{\dim(\mathbf{W}^l)}, \quad (9.12)$$

which penalizes the weights in each layer of the network, \mathbf{W}^l . The gradients of the loss function are calculated using back-propagation [228], and the parameters are updated using stochastic gradient descent algorithms.

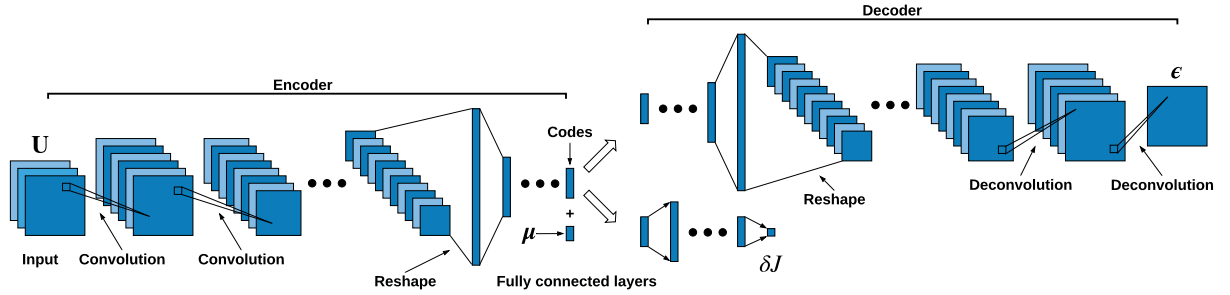


Figure 9.4: Proposed network architecture. The network is composed of an encoder network, a decoder network, and a fully-connected regression network; the decoder and the regression networks share the latent layer to improve the efficiency and to avoid using separate models. The dimension reduction in the encoder only uses the convolution operation, no pooling layer is used. The parameter vector $\boldsymbol{\mu}$ is added into the latent layer as additional codes to help the training.

In contrast to computer vision tasks, often in physical modeling we know a set of low-dimensional codes beforehand: the parameters $\boldsymbol{\mu}$ that govern the system, *e.g.*, the Reynolds number and the Mach number in a flow simulation. Although it is conceptually helpful to add these parameters into the codes in the training process, and this is implemented in the present work, it does not improve the model performance much in our tests. The author believes that the network should be able to extract the parameters information directly from the state input, while these extra codes may be more helpful if the training dataset size is limited.

9.1.6 Fixed Network for Adaptive Simulation on General Domains

Since traditional CNN models are often trained with data of fixed input and output dimensions, they are not readily useful for adaptive simulations as the dimensions of the state and error indicator fields both change as the mesh gets adapted. In order to generalize the model for adaptive simulations, we use a fixed reference mesh to do the error estimation for adaptive simulations. In other words, the fine space h in Eqn. 9.5 is

achieved using a fixed reference mesh that is much finer than the current mesh. At each adaptive iteration, the states are solved on the current mesh and then projected to the reference fine mesh to obtain a fixed-dimension state vector \mathbf{U}_h^H . After applying Eqn. 9.5 on the reference mesh, we obtain a fixed-dimension error-indicator field, \mathcal{E}_h , which is then projected back to the current mesh to drive the mesh adaptation, as shown in Figure 9.5. Since the injected states \mathbf{U}_h^H and the fine space error indicators \mathcal{E}_h are of fixed dimensions, the proposed CNN network can be easily constructed to build the map between them. This error estimation procedure is different from standard adjoint-based error estimation, where the fine space is usually achieved with approximation order increment, p to $p + 1$. During an adaptive simulation, the state data \mathbf{U}_h^H and the error indicator data \mathcal{E}_h are collected on the same reference mesh at each adaptive iteration, resulting in multiple samples for every complete adaptive simulation.

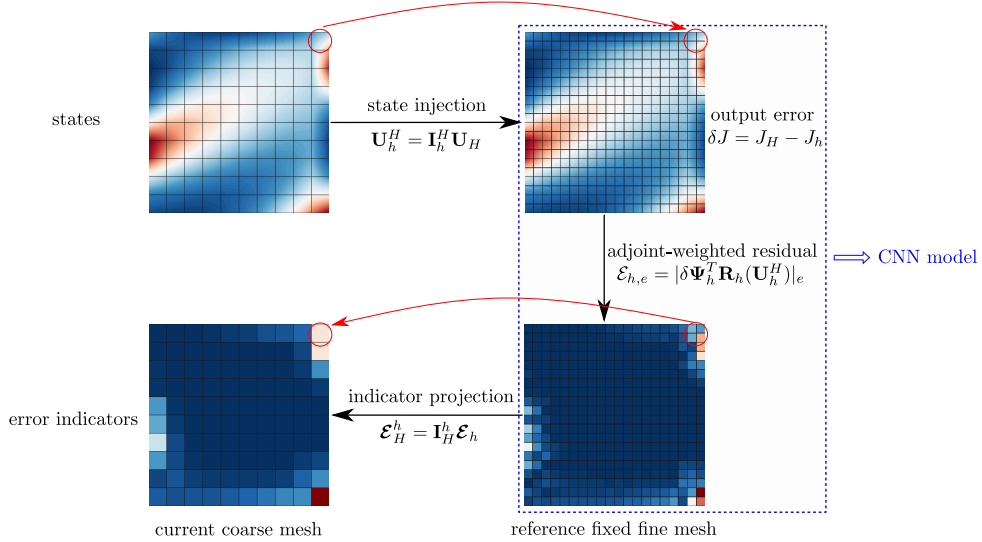


Figure 9.5: An example of building the proposed CNN model on rectangular domains. This example shows an adaptive simulation for the scalar advection-diffusion problem studied in Section 9.2, where the red color denotes large magnitude while the blue regions are of small magnitudes.

This treatment is straightforward for rectangular computational domains without interior geometries. However, practical CFD simulations often involve complex geometries and irregular computational domains, especially in an adaptive setting. Therefore, we seek in this work a topology map from the current computational domain to a rectangular reference domain first, then the projection procedure in Figure 9.5 is applied. Particularly, for the airfoil problems that will be considered in this work, we use the most common single-block topology mapping from a “C-mesh” around the airfoil to a rectangular Cartesian mesh. The error estimation and mesh adaptation can be summarized in Figure 9.6:

the state vector on the current space, \mathbf{U}_H , is first injected to a fixed reference “C-mesh” where the adjoint weighted residual is evaluated and localized, then the localized error indicator can be transferred back to the working mesh for adaptation; Since the fixed fine mesh, *i.e.*, the reference “C-mesh”, is topologically equivalent to a fixed Cartesian mesh on the reference space, we can transfer the states vector and the error indicator vector from the reference “C-mesh” on the physical space to the reference Cartesian mesh on the reference space, where our proposed network architecture can be built in a traditional fashion. Two key benefits of this approach compared to other CNN models that treat the physical inputs directly as image pixels [215, 229, 217] (sampling the inputs on a rectangular mesh) are:

- The input and output are smoother than treating them directly as image pixels, which may cause some visual artifacts in the output field. This is even more preferable if the output fields are physical quantities.
- This approach has the potential to deal with multi-scale physics, as the reference “C-mesh” can be anisotropic while the mesh stretching is embedded in the physical-reference space mapping, *i.e.*, the mapping Jacobian. Nonetheless, the Jacobian field has not been used as an input for this work yet.

The proposed network architecture is first applied in a simple two-dimensional scalar advection-diffusion problem in Section 9.2, where the computational domain is a regular rectangular domain. More complicated aerodynamic simulations over airfoils are considered in Section 9.3. The reference space fine mesh as shown in Figure 9.6 is adopted to handle the geometry and the non-rectangular domain. For simplicity, structured quadrilateral meshes are used for both problems. Although the CNN model relies on quadrilateral meshes on rectangular domains, which directly restricts the reference fine meshes (on both the physical and reference spaces), yet the current coarse working mesh can be arbitrary as the state injection and indicator projection can be done even between different mesh structures.

9.2 Two-Dimensional Advection-Diffusion Problem

We first test our proposed network architecture on a scalar advection-diffusion problem, with the governing equation written as

$$\vec{V} \cdot \nabla u - \nu \nabla^2 u = 0, \quad (x, y) \in \Omega, \quad (9.13)$$

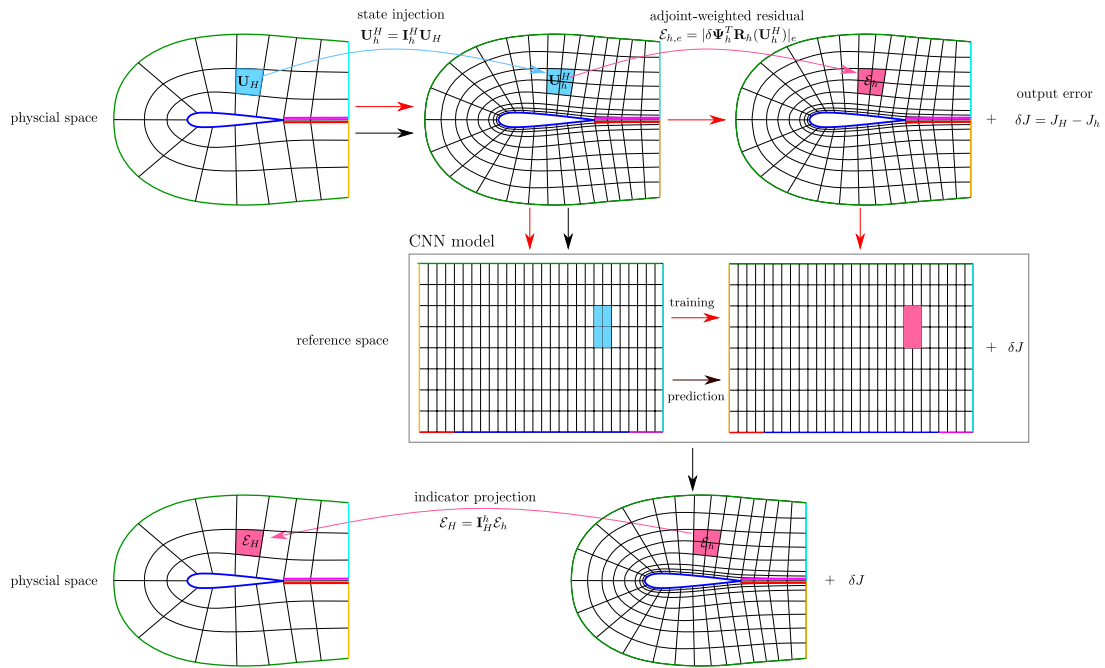


Figure 9.6: An example of building the proposed CNN model on general domains. The red arrows describe the steps taken in the offline model training process, where the data from the adjoint-based approach is used. On the other hand, the black arrows depict the online prediction phase, where both the overall output error and the adaptive error indicator field are directly predicted by the CNN model. Note in this work, we use the readily available output difference between the coarse and fine spaces as the output error truth data, rather than the adjoint-based estimation.

where \vec{V} denotes the advection velocity, ν is the viscosity, and u is the scalar state. In this problem, we consider a simple unit square computational domain. A Dirichlet boundary condition is used on all four boundaries,

$$u = \exp(0.5 \sin(-4x + 6y) - 0.8 \cos(3x - 8y)), \quad (x, y) \in \partial\Omega. \quad (9.14)$$

The governing equation is discretized with a DG method as presented in Chapter 2, which results in a discretized algebraic equations in the form of Eqn. 9.1,

$$\mathbf{R}(\mathbf{U}; \boldsymbol{\mu}) = 0, \quad \boldsymbol{\mu} = \{\vec{V}, \nu\}. \quad (9.15)$$

The output of interest J is the integral of the viscous flux, $-\nu\nabla u$, at the right boundary of the domain.

9.2.1 Data Generation and Preprocessing

In the test problem, we restrict the advection velocity magnitude to be unit, $|\vec{V}| = 1$; and we use the non-dimensional Péclet number (Pe) instead of the viscosity to parameterize the system. Thus, the parameter space is reduced to two dimensions, $\boldsymbol{\mu} = \{\alpha, Pe\}$, where α is the advection angle defined by $\vec{V} = [\cos \alpha, \sin \alpha]$, and the Péclet number is defined as $Pe \equiv |\vec{V}|L/\nu$, where L is the domain length. We generate the data by uniformly sampling 21 points in the advection angle space $\alpha \in [0, 60]$ degrees and 50 points in the Péclet number space $Pe \in [1, 50]$, resulting in a data set of 1050 adaptive simulations, $\mathcal{D}_\mu = \{\boldsymbol{\mu}^i\}, i = 1, 2, \dots, N_\mu$ ($N_\mu = 1050$).

At each parameter point $\boldsymbol{\mu}^i$, the governing equation is solved with a DG $p = 1$ discretization on a uniform mesh starting with 5×5 elements. Then the output error indicator field is obtained with the adjoint-based method as shown in Figure 9.5, with a reference fine mesh consisting of 320×320 elements. Both the adjoint and state vectors are solved exactly on the fine reference mesh for the error estimation and error localization. Since the state vector is solved exactly on the fine mesh, we use the exact difference between the outputs on the current mesh and the reference mesh as the ground truth value for the output error in our data. If the state and adjoint vectors are solved approximately on the reference mesh, the adjoint-weighted residual can be used as the true output error instead. In each adaptive simulation, 19 mesh adaptations are performed, resulting in 20 data points including the data on the initial mesh. Therefore, the entire dataset contains $N_d = N_\mu \times 20 = 21000$ samples. The dataset is then randomly shuffled and split into a training dataset of 14700 samples (70%), a validation dataset of 4200 samples (20%), and

a testing dataset of 2100 samples (10%).

Since the error indicator is localized to a scalar per element, the state vector (per channel) with order $p > 0$ will have higher dimension compared to the error indicator field. Nonetheless, the network can be carefully designed to handle the dimension mismatch. However, the state non-uniqueness at element interfaces in the DG method makes the network design and training cumbersome. In the current implementation, we average the states at the element interfaces to make the solution “continuous”. This can also be considered in the CNN point of view as a down-sampling or convolution operation only on the element interfaces. However, this filter is defined *a-priori*, which may not be optimal in our regression tasks. For approximation order $p = 1$, the averaging process results a state vector of the same size as the reference mesh nodes, while the adaptive error indicator field has the same size as the reference mesh elements. Therefore, the network input is of size 321×321 , while the network output is a 320×320 single-channel indicator field and a scalar output error predication. A logarithm transformation is applied to the indicator field $\log(|\mathcal{E}|)$ and the output error $\log(|\delta J|)$ before training, as the transformed output has lower variance and generally helps the training [102, 103]. Several samples from the dataset are shown in Figure 9.18, in which the second column shows the projected state fields (network inputs), and the fourth column presents the error indicator fields (network outputs), both on the fine reference mesh.

9.2.2 Network Implementation and Training

The network is designed following the architecture proposed in Section 9.1.5, and the detailed structure is summarized in Table 9.1. The training loss is defined according to Eqn. 9.9 with $\lambda_\delta = 64$ and $\lambda_{reg} = 0.001$ ³. λ_δ is large since the output error modeling is found to be more difficult than the indicator field prediction though the latter has much higher dimension. The output error allows cancellations of the errors in different elements such that its behavior is more oscillatory compared to the more conservative error indicator field. The network is implemented in TensorFlow [205] and trained with the adaptive moment estimation (Adam) algorithm [206]. The starting learning rate is set to be 0.0001, and 500 total epochs with mini-batch size of 20 are run in the training. The training and validation losses are recorded in Figure 9.8, and the performance of the resulting model on both the training and validation datasets is shown in Figure 9.9 and Figure 9.10. As expected, the model performs well on the training data which the

³ λ_δ and λ_{reg} are hyper-parameters that can be further tuned to achieve better performance. The outputs have to be normalized to make the tuned hyper-parameters more generalizable, yet this is not performed in our training.

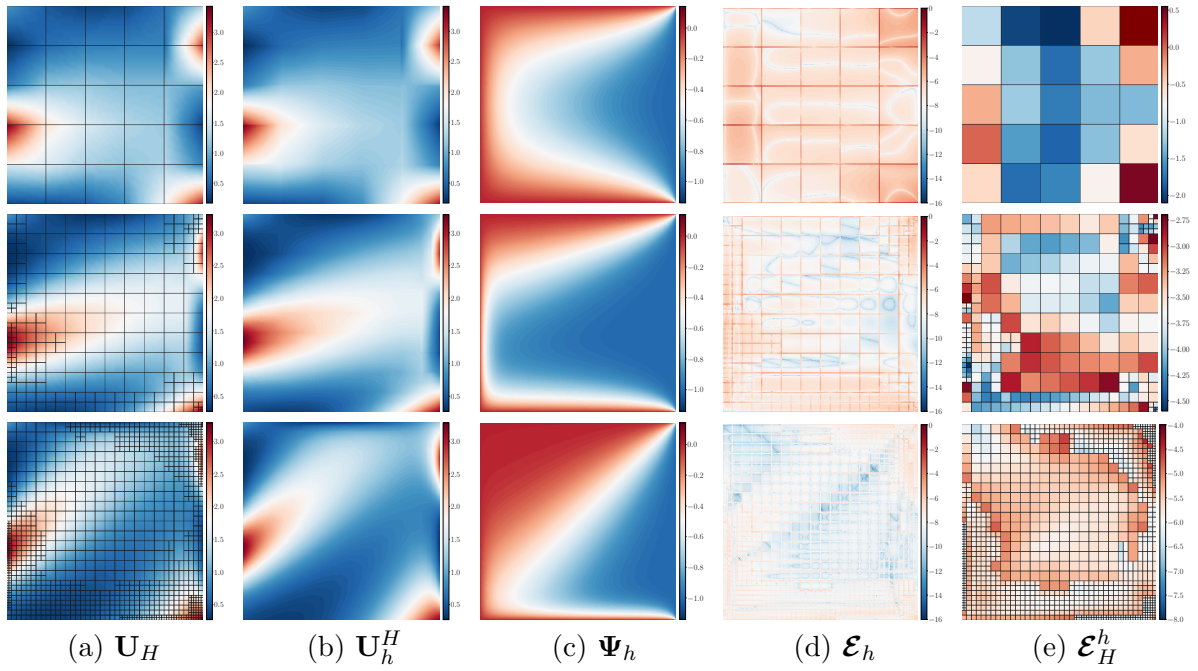


Figure 9.7: Three samples from the dataset of the scalar advection-diffusion problem. The first and the last columns show the state and the indicator fields on current adapted meshes. The second and the fourth columns are the projected states and the error indicators on the reference mesh, used as inputs and outputs respectively in our network model. The third column depicts the adjoint variables on the reference mesh. The reference mesh has 320×320 elements, and the mesh lines are not shown to make the contours clearer.

model is trained with. Meanwhile, the model also shows good predictions for both the adaptive error indicator fields and the total output errors on the validation set, as shown in Figure 9.9 and Figure 9.10, indicating a good generalization of the model ⁴. The model performance will be investigated in details on the testing dataset in Section 9.2.3.

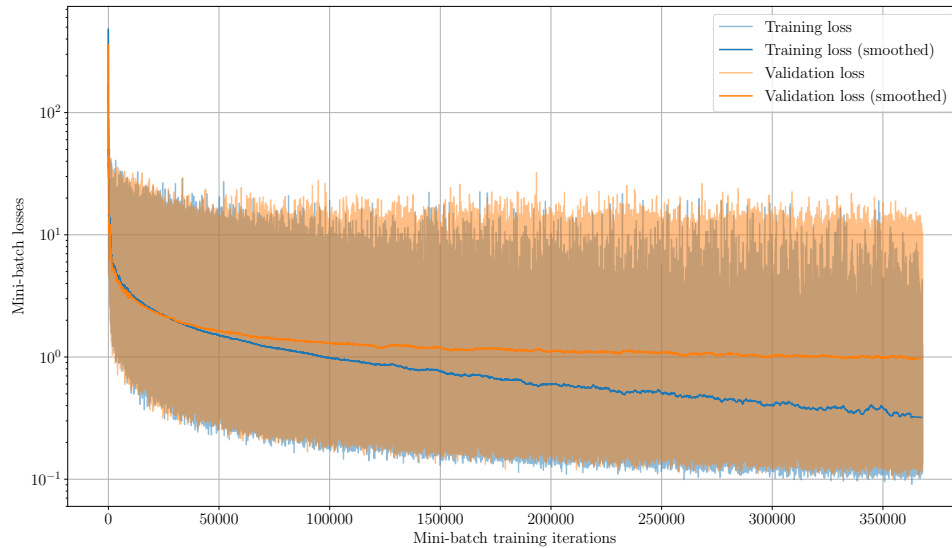


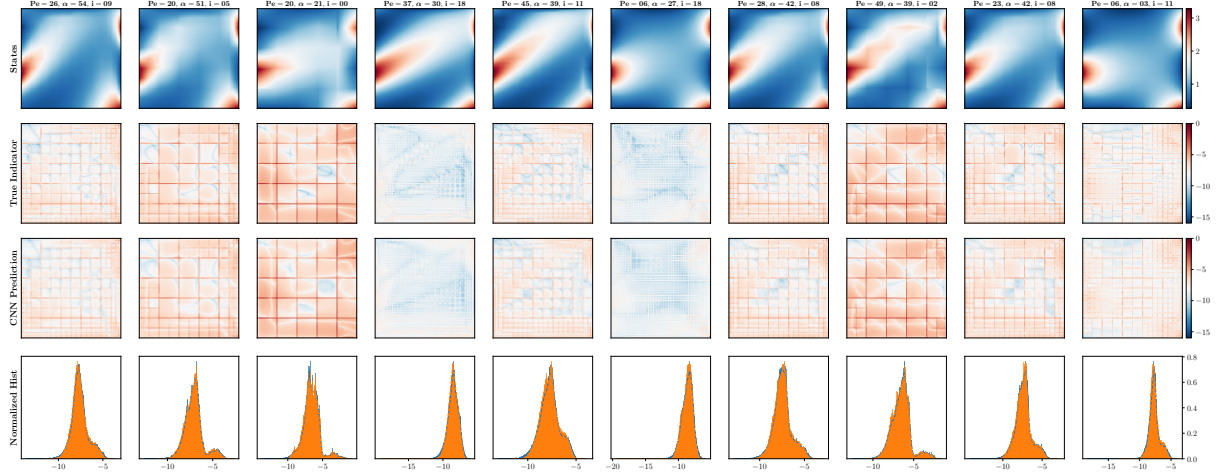
Figure 9.8: Training history of the model for the scalar advection-diffusion problem.

⁴In general, the validation dataset is used to monitor the model generalization and tune the model hyper-parameters during the training, and thus is not suitable as testing data although the model does not see the validation data during the training.

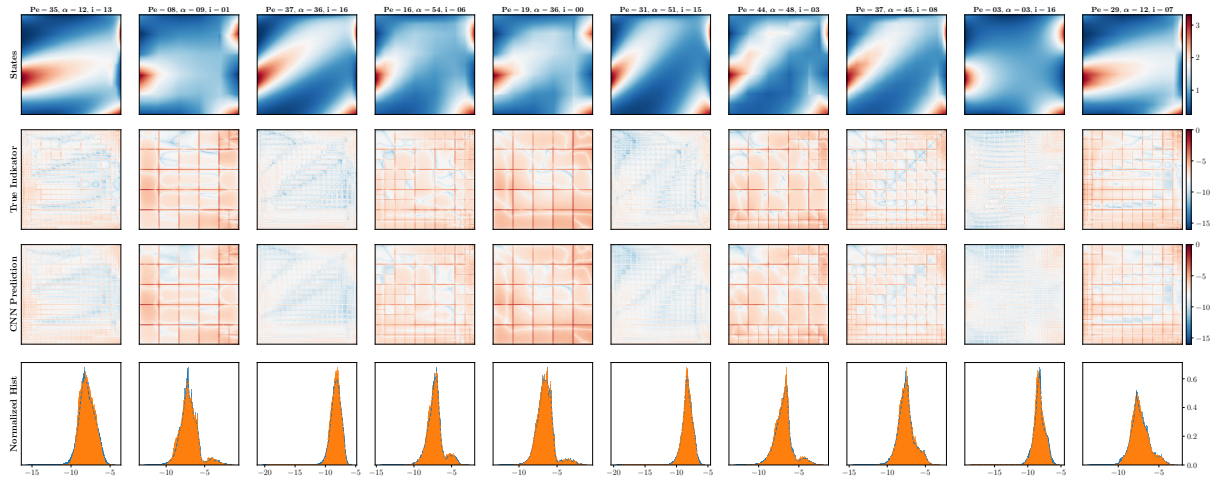
Table 9.1: Network architecture for the scalar advection-diffusion problem.

Subnetwork	Sublayer	Input layer	Operation	Output dim	Activation
Input	States			$321 \times 321 \times 1$	
	Pe			1	
	α			1	
Encoder	Conv1	States	Convolution ($F = 2 \times 2 \times 128, s = 1$)	$321 \times 321 \times 128$	ReLU
	Conv2	Conv1	Convolution ($F = 2 \times 2 \times 128, s = 2$)	$161 \times 161 \times 128$	ReLU
	Conv3	Conv2	Convolution ($F = 2 \times 2 \times 128, s = 2$)	$81 \times 81 \times 128$	ReLU
	Conv4	Conv3	Convolution ($F = 4 \times 4 \times 128, s = 4$)	$21 \times 21 \times 128$	ReLU
	Conv5	Conv4	Convolution ($F = 4 \times 4 \times 64, s = 4$)	$6 \times 6 \times 64$	ReLU
	Flat	Conv5	Reshape	2304×1	None
	Compress	Flat	Fully-connected	800×1	ReLU
	Codes	Compress, Pe, α	Concatenate	802×1	None
Decoder	Decompress	Codes	Fully-connected	1600×1	ReLU
	Unflat	Decompress	Reshape	$5 \times 5 \times 64$	None
	Deconv1	Unflat	Convolution ^T ($F = 4 \times 4 \times 128, s = 4$) ⁵	$20 \times 20 \times 128$	ReLU
	Deconv2	Deconv1	Convolution ^T ($F = 4 \times 4 \times 128, s = 4$)	$80 \times 80 \times 128$	ReLU
	Deconv3	Deconv2	Convolution ^T ($F = 2 \times 2 \times 128, s = 2$)	$160 \times 160 \times 128$	ReLU
	Deconv4	Deconv3	Convolution ^T ($F = 2 \times 2 \times 128, s = 2$)	$320 \times 320 \times 128$	ReLU
	IndPred	Deconv4	Convolution ($F = 2 \times 2 \times 1, s = 1$)	$320 \times 320 \times 1$	None
	Regressor	Dense1	Codes	Fully-connected	400×1
Dense2		Dense1	Fully-connected	200×1	ReLU
Dense3		Dense2	Fully-connected	100×1	ReLU
ErrEst		Dense3	Fully-connected	1	None

⁵This is a transposed convolution operation [226].



(a) training set



(b) validation set

Figure 9.9: Model performance of error indicator field predictions on the training and validation sets (scalar advection-diffusion). The top row shows the inputs to the network; the top caption shows the parameters of the current data point, in which i indicates the index of the current adaptive iteration, starting from 0. The second row presents the ground truth, while the third row contains the predictions made by the network model. The last row compares the normalized histograms of the predictions (orange) and the ground truth (blue).

9.2.3 Network Testing and Model Deployment

The model obtained in Section 9.2.2 is first tested on the testing set generated in Section 9.2.1 to assess the generalization power of the model on unseen data. This can also be considered as an interpolation test since the testing data and the training data show strong similarity as they are both sampled from the same dataset generated in

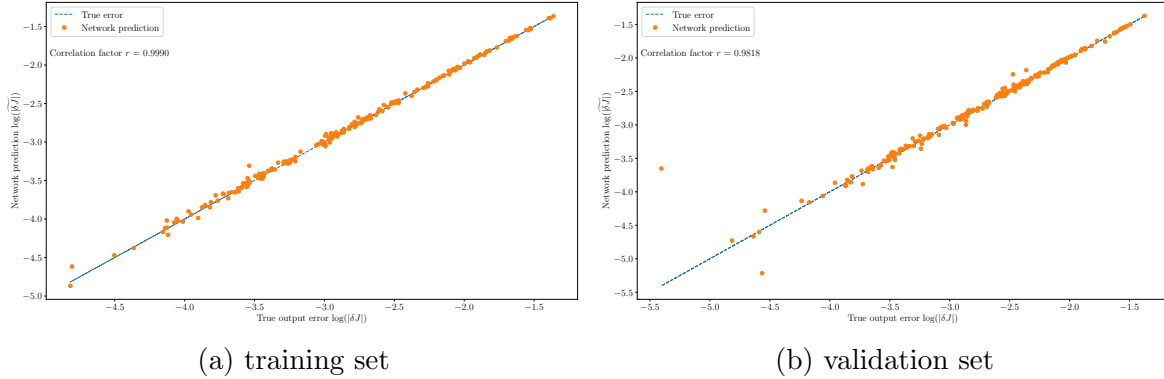
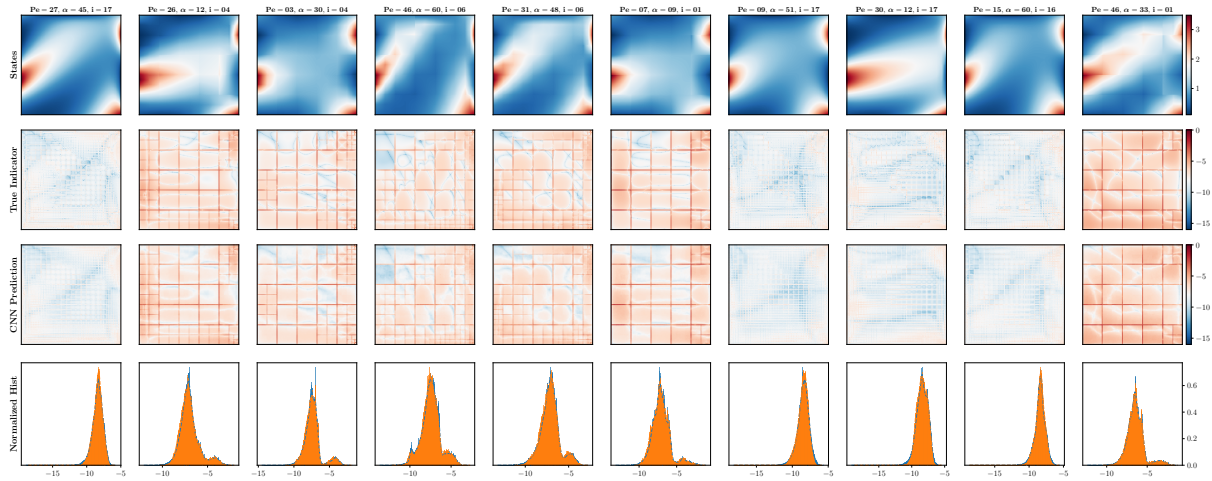


Figure 9.10: Model performance of output error predictions on the training and validation sets (scalar advection-diffusion). Each plot is generated using 200 samples randomly sampled from the training and validation sets.

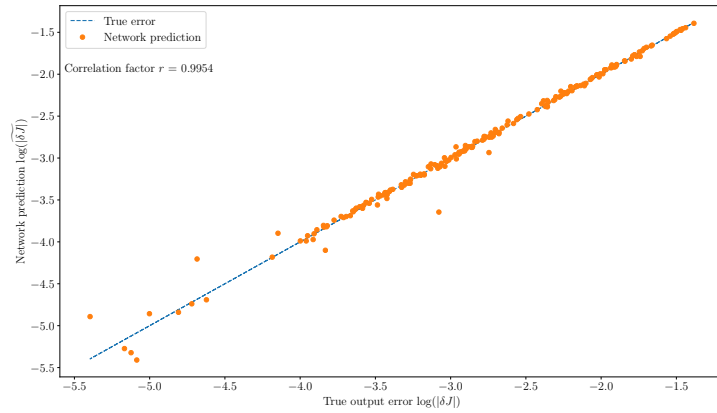
Section 9.2.1. To further study how well the model generalizes, we also generated more data with parameters that are out of the parameter space sampled in Section 9.2.1 and tested the model on them, which we called extrapolation tests in this work. On each testing set, we also deploy the trained model in the flow solver to validate the effectiveness of the model predictions in real-time simulations.

9.2.3.1 Interpolation on Unseen Data

In this test, the testing samples are from the testing set generated in Section 9.2.1. The performance of the model is shown in Figure 9.11, from which we can see that the model achieves good accuracy on both the adaptive error indicator and output error predictions. The model is then deployed to perform two adaptive simulations using the parameters chosen from Figure 9.11a. Standard adjoint-based error estimation and mesh adaptation are also performed on these two cases. All the simulations start with the same initial mesh with 5×5 elements. The final adapted meshes and the output error convergence are compared using our model and the standard adjoint-based approach, as shown in Figure 9.12. We can see that the trained model is able to identify important regions for the output prediction and produce similar final adapted meshes compared to the adjoint-based method, although the true output errors are slightly higher than the adjoint-based method. On other other hand, the CNN model also gives acceptable error estimation on the true output error, with accuracy comparable to the adjoint-based approach.



(a) Interpolation test of the error indicator field predictions on the testing set.



(b) Interpolation test of the output error predictions on the testing set (200 samples).

Figure 9.11: Model interpolation test on the testing dataset (scalar advection-diffusion). Refer to Figure 9.9 and Figure 9.10 for a detailed figure interpretation.

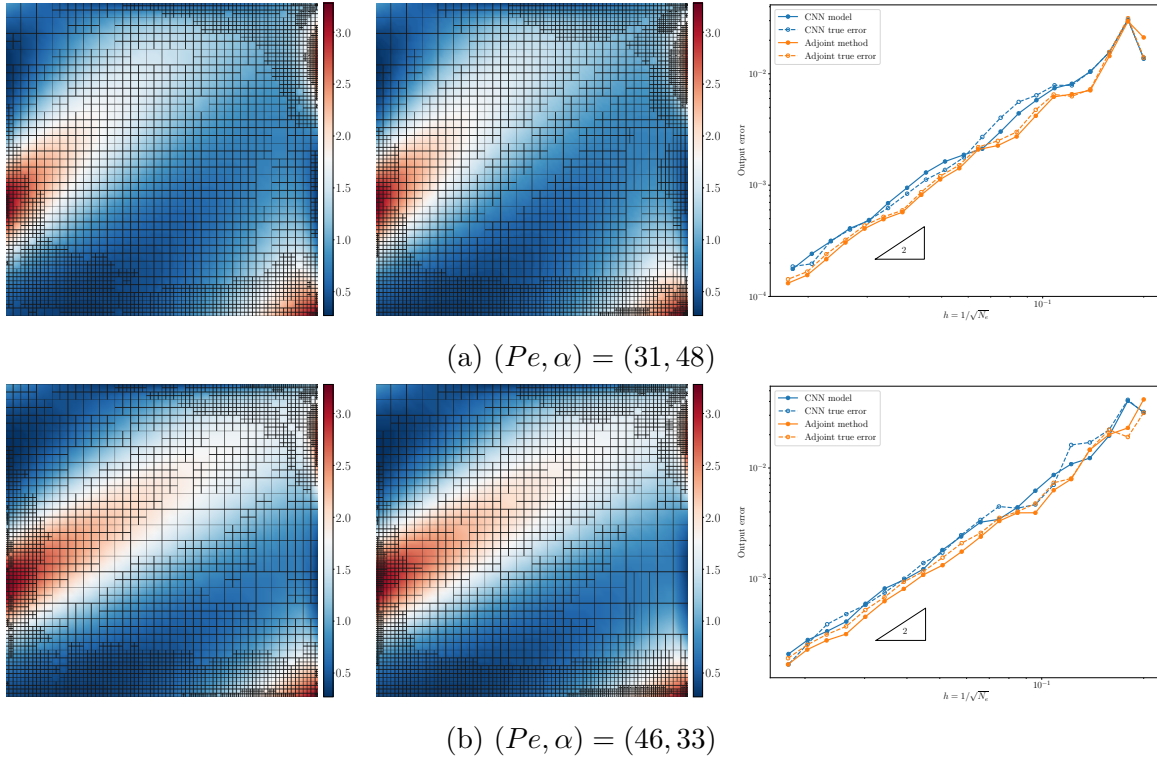
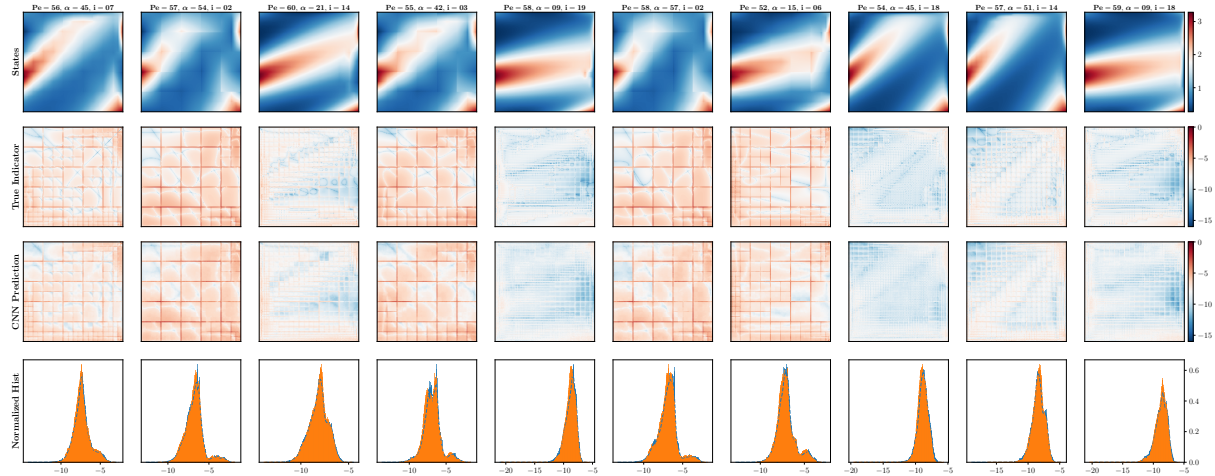


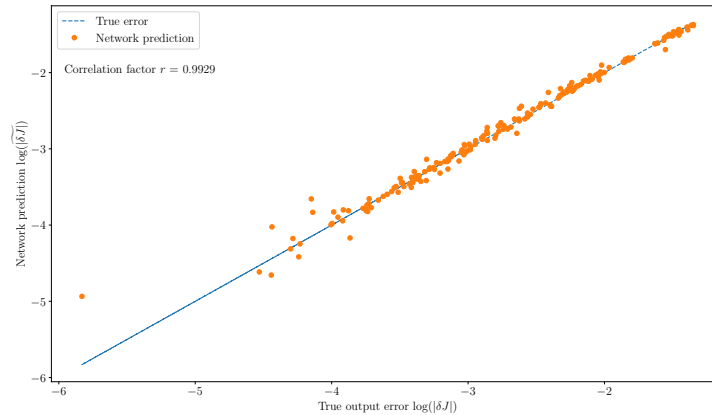
Figure 9.12: Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion). The first column shows the states on the final adaptive meshes from the CNN model, while the second column presents the ones from the adjoint method. The last column shows the output error convergence history in these two methods. The solid lines are the error estimates computed by the CNN model and the adjoint method, while the dashed lines are the “true” output error compared to the “true” outputs obtained on the reference fine mesh with order increment, *i.e.*, $p \rightarrow p + 1 = 2$.

9.2.3.2 Extrapolation on Unseen Data

Unseen Péclet Numbers Pe For the extrapolation test, we first test the model on unseen Péclet numbers. Keeping the advection angles $\alpha \in [0, 30]$, the testing data is randomly sampled from $Pe \in [51, 60]$. The testing results are shown in Figure 9.13. The model is able to predict the adaptive error indicator fields and the output errors accurately on the testing data, indicating good generalizations on the Pe space. Again, adaptive simulations using the CNN model and the adjoint-based approach are compared on two samples chosen from Figure 9.13a. The comparison is shown in Figure 9.14, from which we can see that the trained model is able to effectively drive the mesh adaptation, though the true output error is higher than the adjoint-based method. The error estimation provided by the CNN model is again accurate, and the accuracy is close to the adjoint-based method.



(a) Extrapolation test of the error indicator field predictions on the testing set (unseen Pe)



(b) Extrapolation test of the output error predictions on the testing set (unseen Pe , 200 samples).

Figure 9.13: Model extrapolation test on the testing data (scalar advection-diffusion, unseen Pe). Refer to Figure 9.9 and Figure 9.10 for a detailed figure interpretation.

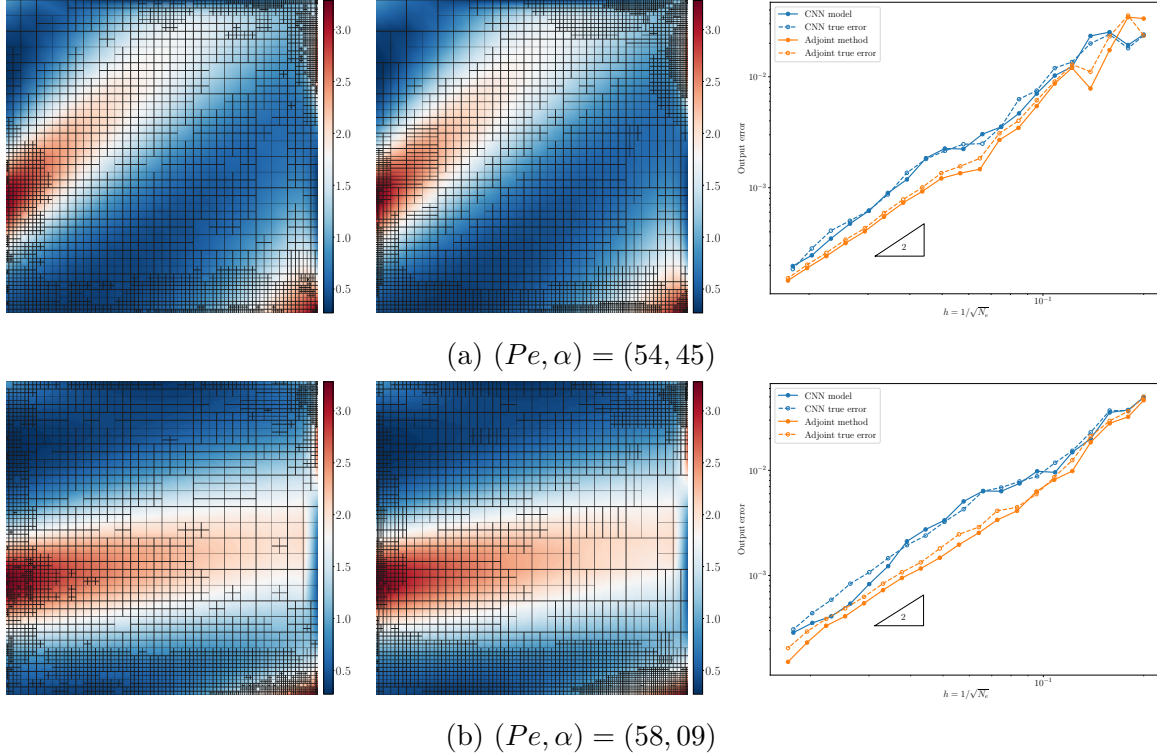
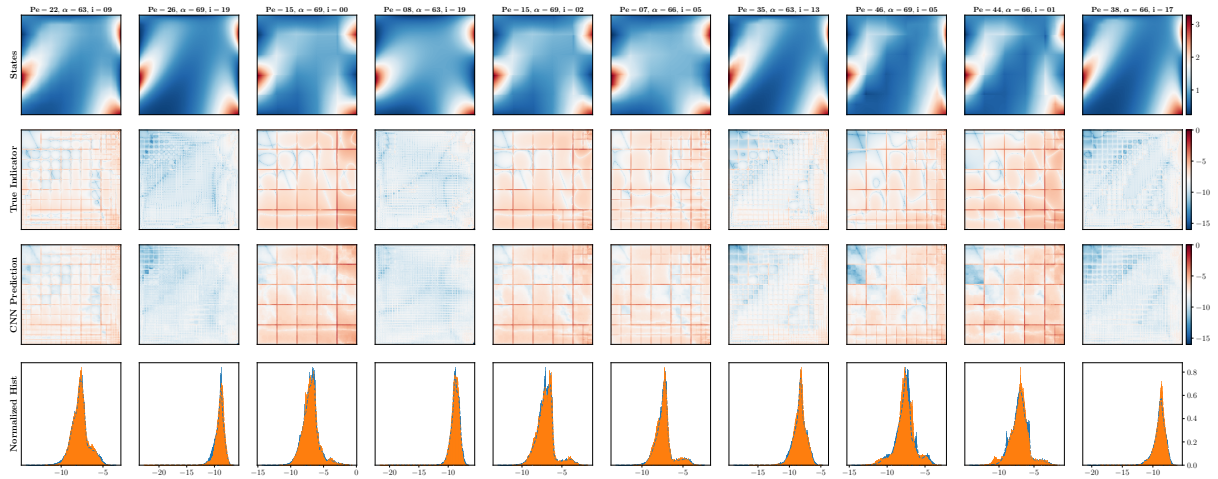
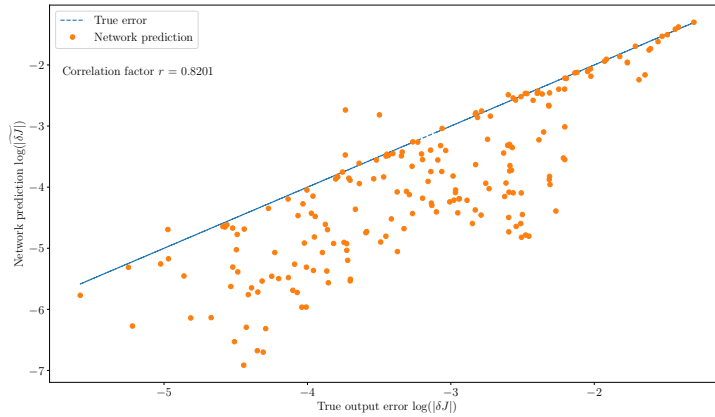


Figure 9.14: Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion, unseen Pe). The figures from left to right are: final CNN adapted meshes, final adjoint-based adapted meshes, and the output error convergence. Refer to Figure 9.12 for a more detailed figure interpretation.

Unseen Advection Angles α In this extrapolation test, the model is tested on unseen advection angles. The testing data is sampled from $\alpha \in \{63, 66, 69\}$, while keeping $Pe \in [1, 50]$. The model performance on the error indicator prediction and the output error prediction is presented in Figure 9.15. We see that the model again generalizes well on the error indicator field predictions but tends to underestimate the output errors in this test. Adaptive simulations on two samples from Figure 9.15, are again compared in Figure 9.16. As expected, the adaptation performance is comparable to the adjoint-based method (similar “true” output error convergence on the adapted meshes), while the error estimation provided by the CNN model is unreliable. We expect that more training samples in the advection angle space should help generalize the model, as currently the sample points in α space are very limited (only 21 points compared to 50 points in the Pe space). On the other hand, given the fact that the error estimation of the adjoint-based method in this test dataset is also less accurate, the CNN model, which is trained using data from the adjoint-based method, is not expected to perform well in this test region in the first place.

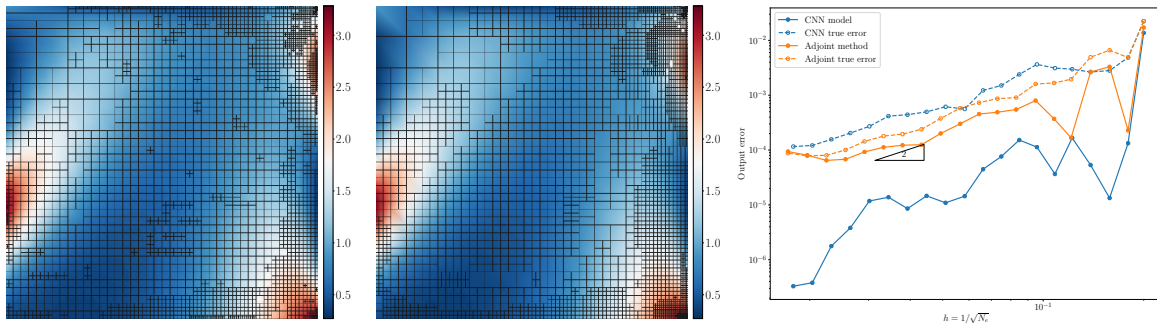


(a) Extrapolation test of the error indicator field predictions on the testing set (unseen α).

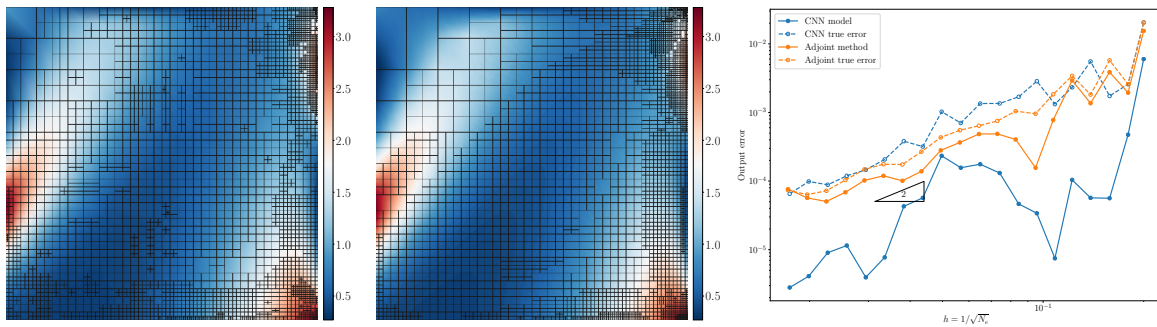


(b) Extrapolation test of the output error predictions on the testing set (unseen α , 200 samples).

Figure 9.15: Model extrapolation test on the testing data (scalar advection-diffusion, unseen α). Refer to Figure 9.9 and Figure 9.10 for a more detailed figure interpretation.



(a) $(Pe, \alpha) = (26, 69)$



(b) $(Pe, \alpha) = (44, 66)$

Figure 9.16: Comparison of the CNN model and the standard adjoint-based method in adaptive simulations (scalar advection-diffusion, unseen α). The figures from left to right are: final CNN adapted meshes, final adjoint-based adapted meshes, and the output error convergence. Refer to Figure 9.12 for a more detailed figure interpretation.

9.3 Application in Aerodynamic Simulations Over Airfoils

In this section, we consider a more complicated problem: transonic aerodynamic flow simulations over airfoils, which involves geometries and irregular computational domains. Inviscid Euler equations are used as the flow governing equation. Again, we use DG with an element-based artificial viscosity [128] to discretize the system of equations. The fully discretized system can be written in the form of Eqn. 9.1,

$$\mathbf{R}(\mathbf{U}; \boldsymbol{\mu}) = 0, \quad \boldsymbol{\mu} = \{\mathbf{x}_s, M, \alpha\}. \quad (9.16)$$

The system is parameterized by the airfoil shape parameters, \mathbf{x}_s , the freestream Mach number M , and the angle of attack α . The output of interest J is the drag coefficient of the airfoil, c_d .

9.3.1 Data Generation and Preprocessing

In this work, we use the NACA 4-digit airfoil series [230] with closed trailing edge to parameterize the airfoil shape. The airfoil geometry is controlled by the maximum camber, C , in percentage of the chord, the location of the maximum camber, P , in tenths of the chord, and the maximum airfoil thickness, T (two digits), also in percentage of the chord. Therefore the parameter space is in five dimensions, $\boldsymbol{\mu} = \{C, P, T, M, \alpha\}$. We generate the data by sampling from $C \in \{0, 2, 4\}$, $P \in \{0, 4, 6\}$ and $T \in \{10, 12, 14\}$ in the airfoil shape space, resulting in 15 shapes in total since C and P can only be both zeros or non-zeros. For the angle of attack, we uniformly sample 16 points from $\alpha \in [0, 3]$ degrees. The Mach numbers are sampled from $M \in [0.54, 0.68]$. As the flow fields become more complex for higher Mach numbers, we uniformly sampled 4 points from $[0.54, 0.60]$ and 8 points from $[0.61, 0.68]$. In conclusion, the data set consists of $N_\mu = 15 \times 16 \times 12 = 2880$ parameters.

At each parameter point $\boldsymbol{\mu}^i$, we solve the flow equations with DG $p = 1$ discretization adaptively starting with a coarse “C-mesh” consisting of 10 points in height (airfoil surface to farfield) and 24 points in width (airfoil chord and wake). The adjoint-weighted residual is then calculated on a reference fixed fine mesh (also “C-mesh”) on the physical space with 129 points in height and 1281 points in width (128×1280 elements). The starting coarse mesh and the reference fine mesh are compared in Figure 9.17. We solve the state vector and the adjoint vector again exactly on the reference fine mesh, thus the exact output difference is recorded as the truth for the output error and the localized adjoint-weighted residual is collected as the truth for the indicator field. In each adaptive simulation,

9 mesh adaptation iterations are performed, resulting in 10 data points including the data on the initial mesh. Thus, the entire dataset has $N_d = N_\mu \times 10 = 28800$ samples. The dataset is randomly shuffled and split into a training set of 20160 samples (70%), a validation set of 5760 samples (20%) and a testing set of size 2880 (20%).

Similar to the scalar case considered in Section 9.2, we first average the state vector (for every component) to reduce it to the same size as of the reference mesh nodes, *i.e.*, 129×1281 . Since the state vector this time involves four components, density, x and y momentums, and energy, we treat them as separate channels. In other words, the input of the network will be in the dimension of $H \times W \times D = 129 \times 1281 \times 4$. On the other hand, the error indicator field (network output) is a single channel output of the same size as the reference mesh elements $H \times W \times D = 128 \times 1280 \times 1$. Meanwhile, the output error is a scalar output of size 1. Again, a logarithmic transformation is applied to the indicator field $\log(|\mathcal{E}|)$ and the output error $\log(|\delta J|)$ before training to scale the outputs. Several samples from the dataset are shown in Figure 9.18, in which the second column shows the projected state solution fields (network inputs), and the fourth column presents the error indicator fields (network outputs), both on the fine reference mesh.

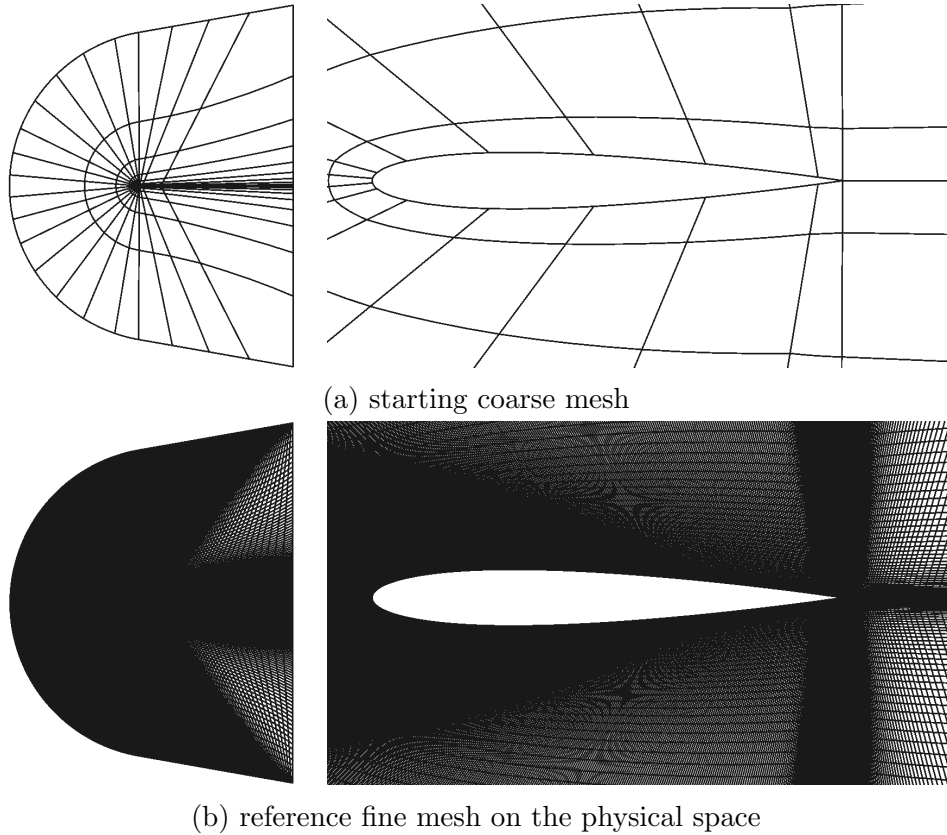


Figure 9.17: The starting coarse mesh for the adaptive simulation and the fixed reference fine mesh for adjoint-weighted residual calculations. The left figures are the zoom-out view of the meshes, while the right ones are the meshes around the airfoils. For different airfoil shapes, the starting coarse mesh and the reference fine mesh are generated using the same setting, *e.g.*, mesh spacing. The reference mesh similarity among different airfoil shapes is especially important here since the physical-reference mapping Jacobian is not used as an input. The starting mesh, on the other hand, can be arbitrary but are made similar for simplicity in this work.

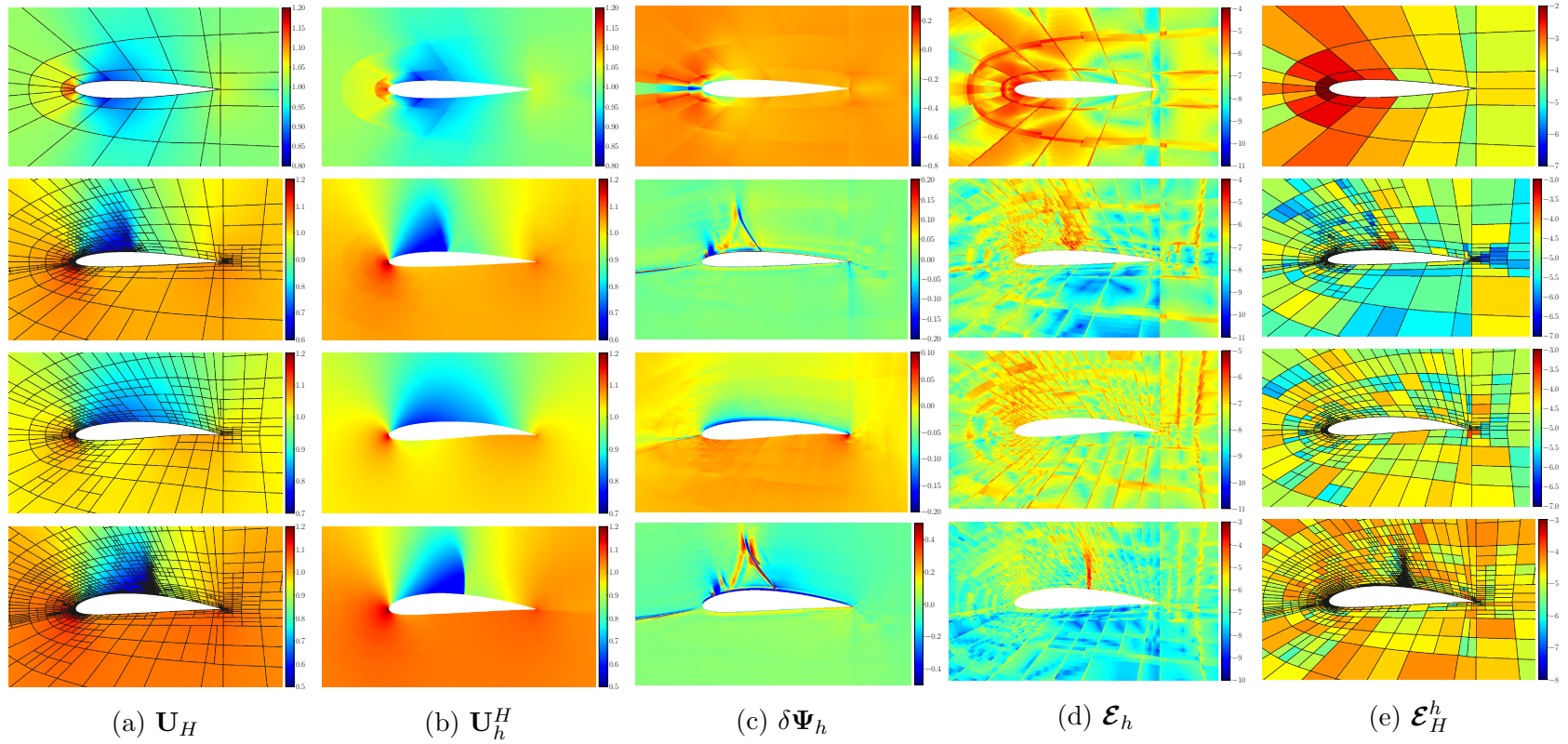


Figure 9.18: Samples from the dataset of the airfoil problem. The first and the last columns show the state and the indicator fields on current adapted meshes. The second and the fourth columns are the projected states and the error indicators on the reference mesh, used as inputs and outputs respectively in our network model. The third column depicts the adjoint variables on the reference mesh. Only the density component of the states and the corresponding adjoint component are shown here.

9.3.2 Network Implementation and Training

The network is built on a rectangular Cartesian mesh on the reference space, which is topologically equivalent to the reference fine mesh on the physical space as shown in Figure 9.17. The network design follows the architecture proposed in Section 9.1.5, and the detailed structure is summarized in Table 9.2. The network is implemented in TensorFlow [205] and trained with the adaptive moment estimation (Adam) algorithm [206]. The starting learning rate is set to be 0.0001, and 500 total epochs with mini-batch size of 20 are run in the training. The training and validation losses are recorded in Figure 9.19, and the performance of the resulting model on both the training and validation datasets is shown in Figure 9.20 and Figure 9.21.

The model shows good predictions for both the adaptive error indicator fields and the total output errors on the training and validation set, as shown in Figure 9.20 and Figure 9.21, indicating a good generalization of the model on the unseen validation dataset. The model performance on the testing dataset and the detailed model deployment will be investigated in the following section.

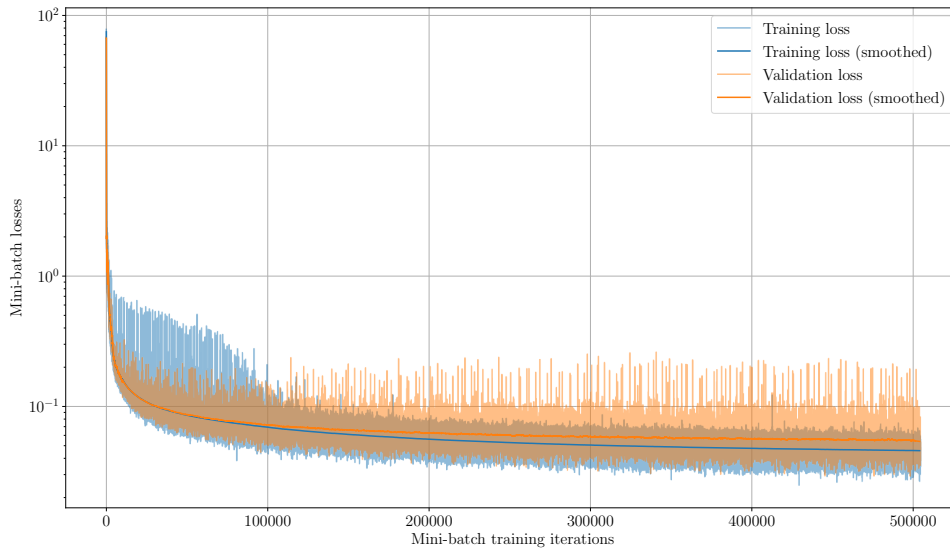
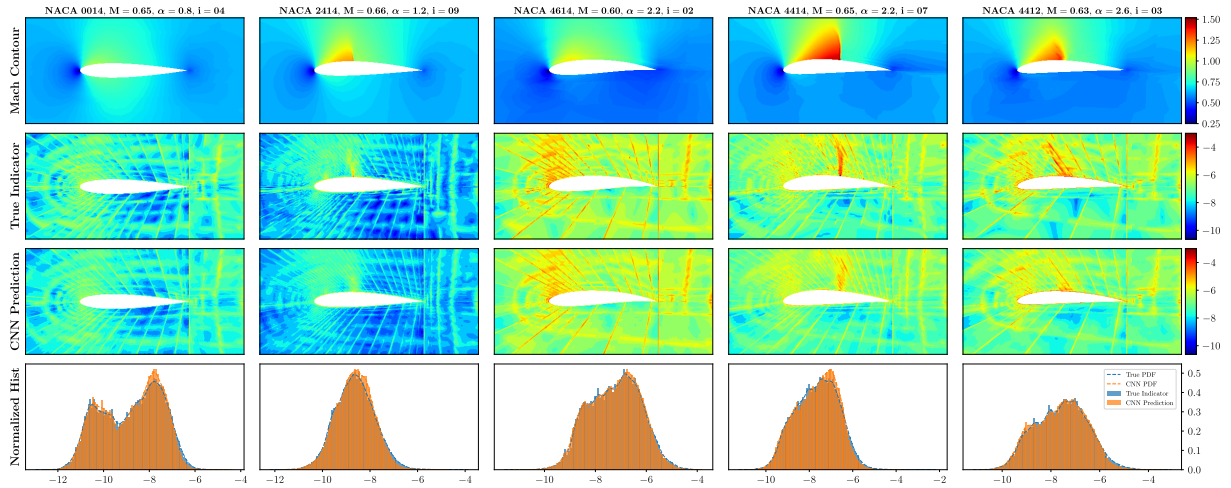


Figure 9.19: Training history of the model for airfoil simulations.

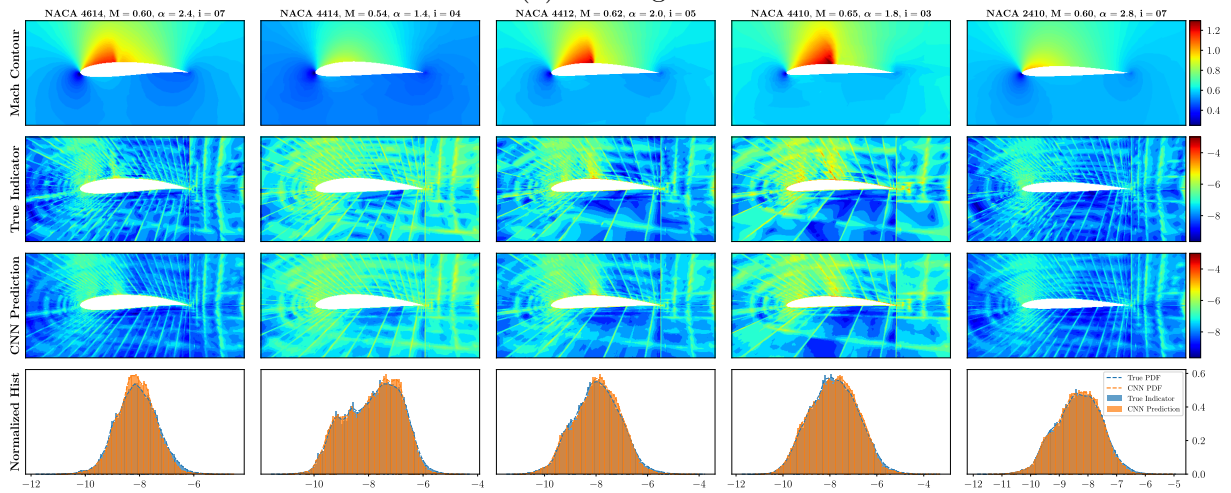
Table 9.2: Network architecture for aerodynamic simulations over airfoils

Subnetwork	Sublayer	Input layer	Operation	Output dim	Activation
Input	States			$129 \times 1281 \times 4^6$	
	C			1	
	P			1	
	T			1	
	M			1	
	α			1	
Encoder	Conv1	States	Convolution ($F = 2 \times 2 \times 128, s = [1, 1]$)	$129 \times 1281 \times 128$	ReLU
	Conv2	Conv1	Convolution ($F = 2 \times 4 \times 128, s = [2, 4]$)	$65 \times 321 \times 128$	ReLU
	Conv3	Conv2	Convolution ($F = 2 \times 4 \times 128, s = [2, 4]$)	$33 \times 81 \times 128$	ReLU
	Conv4	Conv3	Convolution ($F = 2 \times 4 \times 128, s = [2, 4]$)	$17 \times 21 \times 128$	ReLU
	Conv5	Conv4	Convolution ($F = 4 \times 4 \times 64, s = [4, 4]$)	$5 \times 6 \times 64$	ReLU
	Flat	Conv5	Reshape	1920×1	None
	Compress	Flat	Fully-connected	800×1	ReLU
	Codes	Compress, C, P, T, M, α	Concatenate	805×1	None
Decoder	Decompress	Codes	Fully-connected	1280×1	ReLU
	Unflat	Decompress	Reshape	$4 \times 5 \times 64$	None
	Deconv1	Unflat	Convolution ^T ($F = 4 \times 4 \times 128, s = [4, 4]$)	$16 \times 20 \times 128$	ReLU
	Deconv2	Deconv1	Convolution ^T ($F = 2 \times 4 \times 128, s = [2, 4]$)	$32 \times 80 \times 128$	ReLU
	Deconv3	Deconv2	Convolution ^T ($F = 2 \times 4 \times 128, s = [2, 4]$)	$64 \times 320 \times 128$	ReLU
	Deconv4	Deconv3	Convolution ^T ($F = 2 \times 4 \times 128, s = [2, 4]$)	$128 \times 1280 \times 128$	ReLU
	IndPred	Deconv4	Convolution ($F = 2 \times 2 \times 1, s = [1, 1]$)	$128 \times 1280 \times 1$	None
Regressor	Dense1	Codes	Fully-connected	400×1	ReLU
	Dense2	Dense1	Fully-connected	200×1	ReLU
	Dense3	Dense2	Fully-connected	100×1	ReLU
	ErrEst	Dense3	Fully-connected	1	None

⁶State rank = 4



(a) training set



(b) validation set

Figure 9.20: Model performance of error indicator field predictions on the training and validation sets (aerodynamic flow over airfoils). The top row shows the Mach number contours of the input solutions (the network inputs are the state solution fields, *i.e.*, state components, not the Mach number) to the network; the top caption shows the parameters of the current data point, in which i indicates the index of the current adaptive iteration, starting from 0. The second row presents the ground truth, while the third row contains the predictions made by the network model. The last row compares the normalized histograms of the predictions (orange) and the ground truth (blue).

9.3.3 Network Testing and Model Deployment

The model obtained in Section 9.3.2 is first tested on the testing set generated in Section 9.3.1 to assess the generalization power of the model on unseen data. Then the model is deployed in the adaptive flow solver to validate the effectiveness of the model predictions in real-time simulations. Extrapolation testes are also performed on parameters

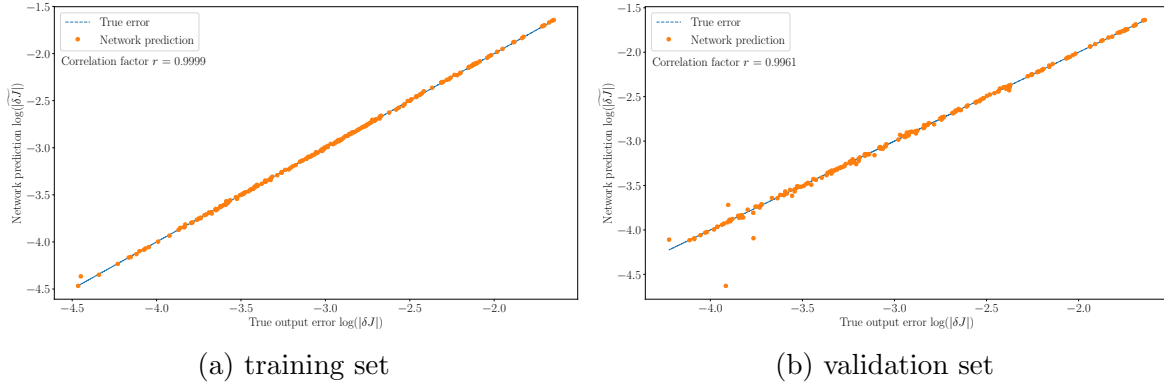


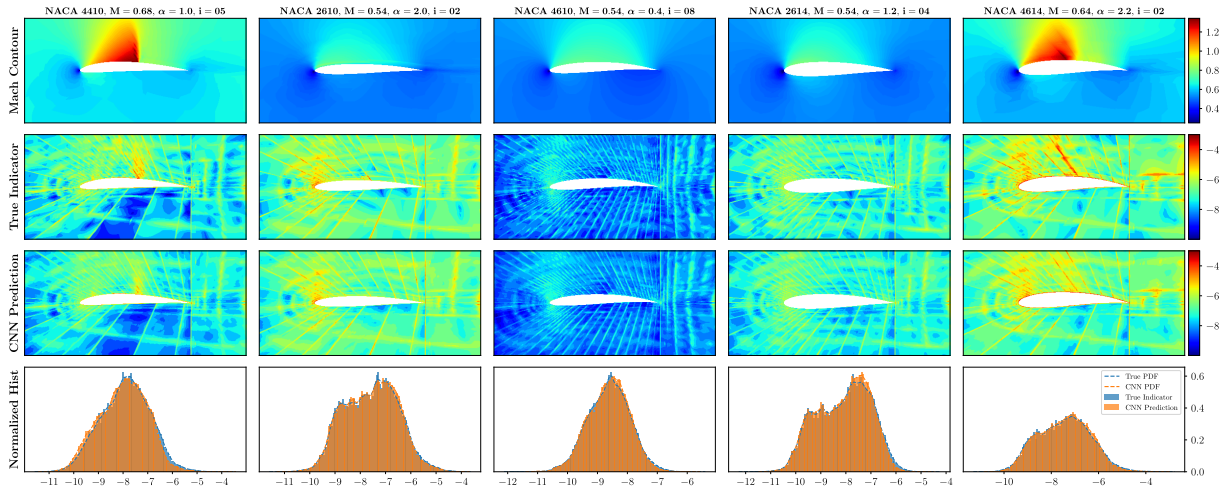
Figure 9.21: Model performance of output error predictions on the training and validation sets (aerodynamic flow over airfoils). Each plot is generated using 200 samples randomly sampled from the training and validation sets.

that are out of the original sampling space to further assess the model generalization.

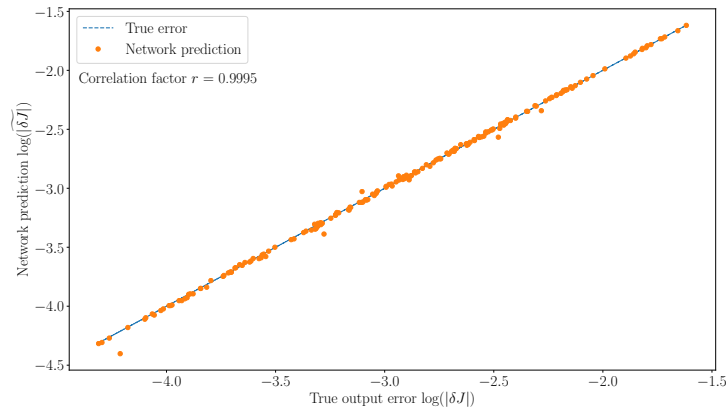
9.3.3.1 Interpolation on Unseen Data

The Testing Dataset We first test the model on the original testing set generated in Section 9.3.1. The performance of the model is shown in Figure 9.22, from which good accuracy on both the adaptive error indicator field and the output error has been observed. The model is then deployed to perform two adaptive simulations using the parameters chosen from Figure 9.22a. Standard adjoint-based error estimation and mesh adaptation are also performed on these two cases. Both methods start with the same initial mesh (10×24 mesh nodes) which possesses similar resolution as the coarse mesh shown in Figure 9.17.

The first testing sample is a subsonic flow over a NACA 2614 airfoil at $M = 0.54$ and $\alpha = 1.2^\circ$. The initial mesh and the final adapted meshes are compared in Figure 9.23, together with the output error convergence plot. We can see that the trained model is able to effectively identify important regions for the output prediction and produce a similar final adapted mesh compared to the adjoint-based method. Although the true output errors are slightly higher than the adjoint-based method, the same optimal superconvergence rate is obtained with the CNN model. On other other hand, the CNN model also gives good error estimation on the true output error, with accuracy comparable to the adjoint-based approach. On the coarse meshes, *i.e.*, first several adaptations, the CNN model gives even better error estimates where the adjoint-based method is in general inaccurate. On these meshes, the adjoint is often not well-resolved such that the adjoint method is ineffective, while the network is trained on the “true” error measured



(a) Interpolation test of the error indicator field predictions on the testing set.



(b) Interpolation test of the output error predictions on the testing set (200 samples).

Figure 9.22: Model interpolation test on the testing dataset (aerodynamic flow over airfoils). Refer to Figure 9.20 and Figure 9.21 for a detailed figure interpretation.

with respect to the fine reference mesh, which turns out to be more accurate and robust on coarse meshes.

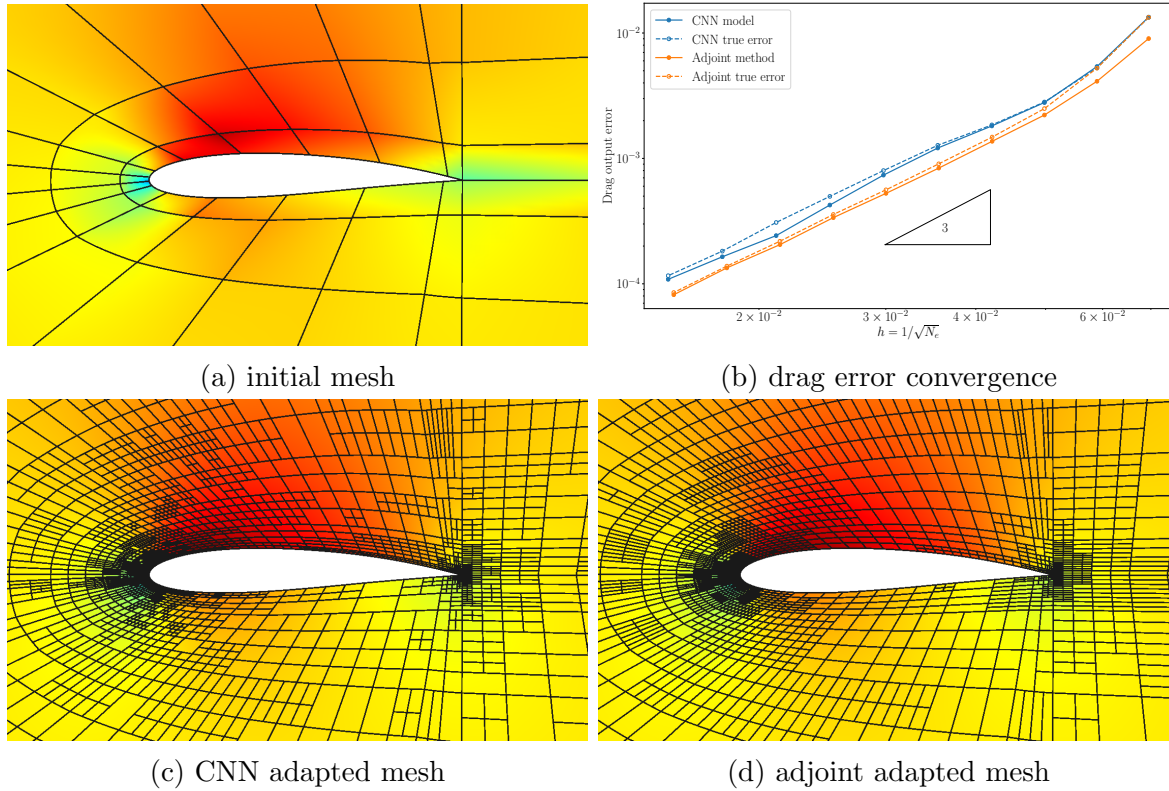


Figure 9.23: Model deployment in adaptive simulations: NACA 2614 airfoil, $M = 0.54$, $\alpha = 1.2^\circ$. The Mach contour scale is $[0, 0.8]$. In the output error convergence plot, the solid lines are the error estimates computed by the CNN model and the adjoint method, while the dashed lines are the “true” output error measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

The second testing sample is a transonic flow over a NACA 4410 airfoil at $M = 0.68$ and $\alpha = 1.0^\circ$. The flow field features a strong shock on the top surface around the middle chord, which should be well-resolved to achieve good accuracy. The performance of the network model is compared with adjoint-based method in Figure 9.24. The CNN model, again, is able to guide the adaptation with a focus on important areas for the output prediction. Similar mesh refinements around the stagnation streamline, trailing edge and the shock location are observed on the final CNN adapted mesh when compared to the adjoint adapted mesh, although the refinements around the shock are a little more spread out in the CNN adapted mesh. For this problem, both methods tend to underestimate the output error during the adaptation and the convergence rate reduces to suboptimal, as shown in the convergence plot. Besides the poor performance on the coarse mesh, the adjoint-based error estimates are also less accurate on the adapted meshes compared

to the CNN model. Since the adjoint-based error estimates are based on linearization, it is usually less accurate in problems that are highly nonlinear, such as the transonic flow with strong shocks. The CNN model, on the other hand, is trained with the “true” output error measured with respect to the fine reference mesh, and thus gives better error estimates in highly nonlinear problems.

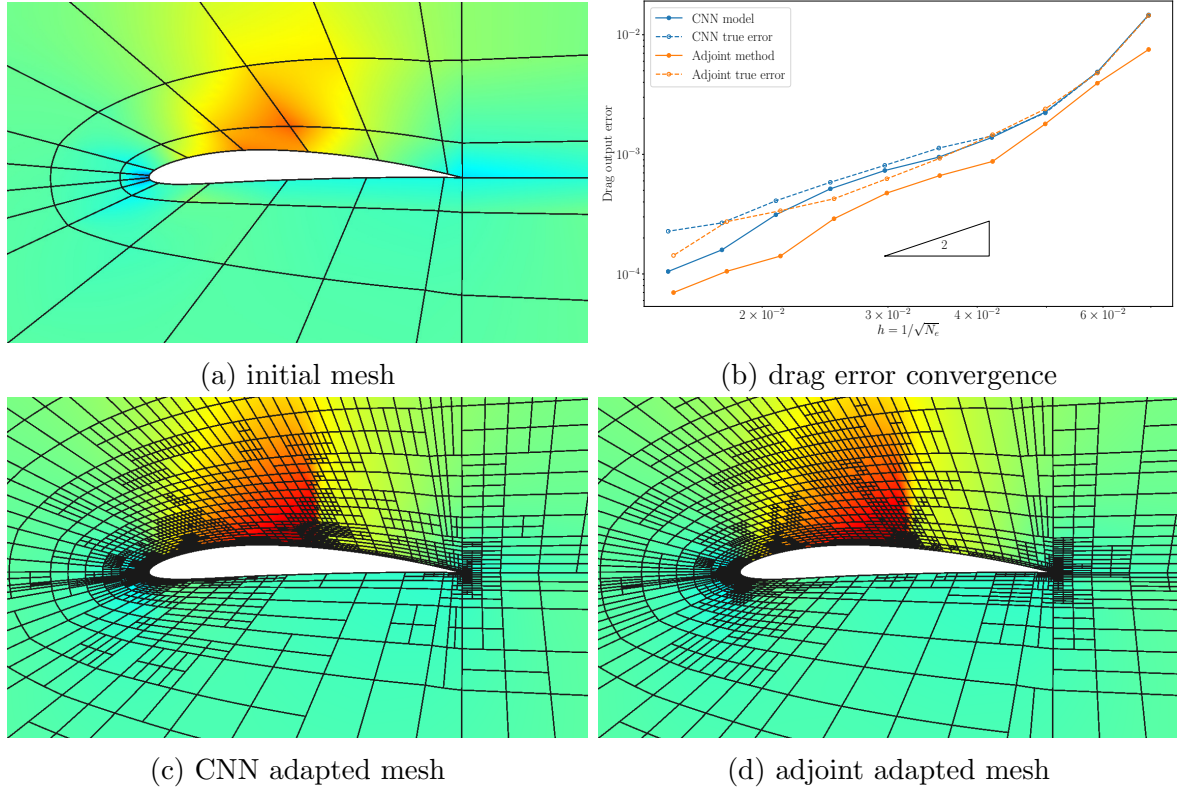


Figure 9.24: Model deployment in adaptive simulations: NACA 4410 airfoil, $M = 0.68, \alpha = 1.0^\circ$. The Mach contour scale is $[0, 1.4]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

9.3.3.2 Extrapolation on Unseen Data

Unseen Mach Numbers For the extrapolation test, we first consider Mach numbers that are out of the original sampling space used in Section 9.3.1. The first case tested is a subsonic flow over a NACA 0012 airfoil at $M = 0.50$ and $\alpha = 2.0^\circ$. We again compare the CNN model with the standard adjoint-based method in an adaptive simulation as shown in Figure 9.25. The CNN model closely follows the adaptation pattern of the adjoint-based one, although subtle difference on the leading-edge refinement can be found. Superconvergence is observed for both the CNN and adjoint adapted meshes, while adjoint-

based meshes have consistent lower drag errors in the adaptation sequence. Nevertheless, the CNN model still predicts the output error fairly well, despite some underestimations during the adaptation sequence.

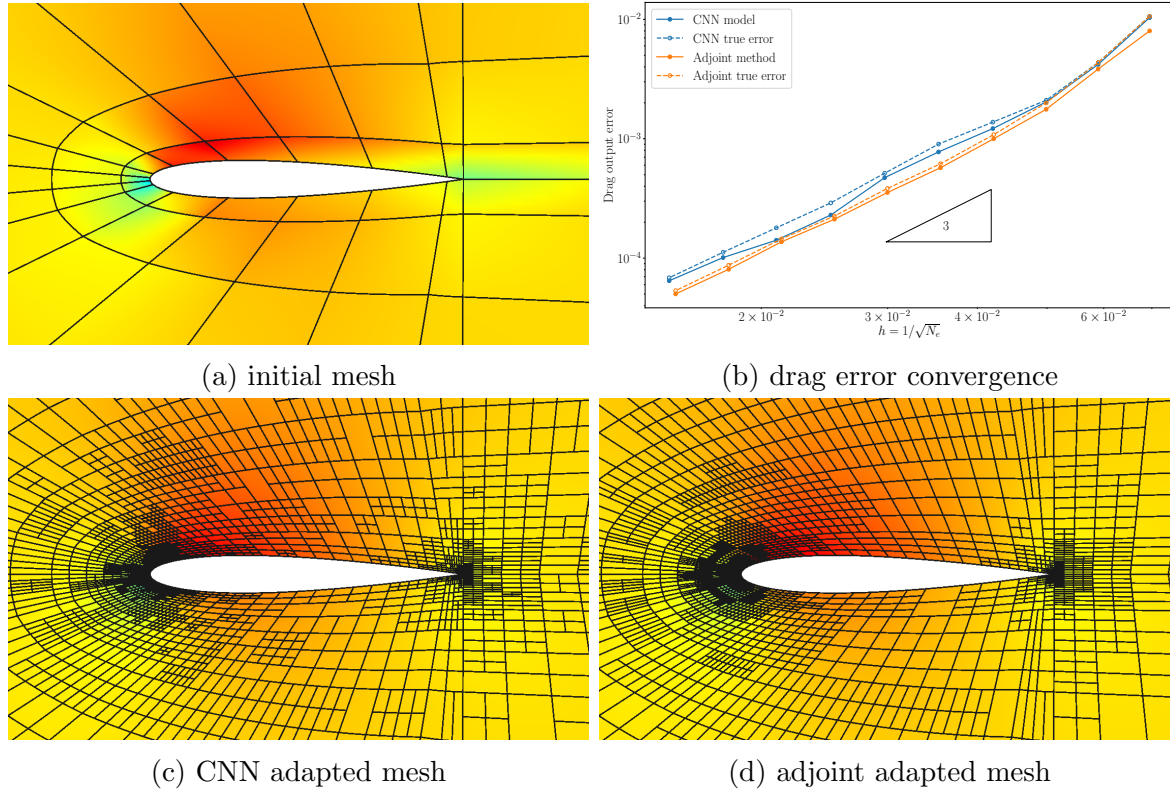


Figure 9.25: Model deployment in adaptive simulations: NACA 0012 airfoil, $M = 0.50, \alpha = 2.0^\circ$. The Mach contour scale is $[0, 0.75]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

The second case considered is a transonic flow simulation over a NACA 2412 airfoil, at $M = 0.70$ and $\alpha = 1.0^\circ$. The adaptive performance of the trained CNN model and the standard adjoint method is compared in Figure 9.26. Both methods converge suboptimally in this case due to the strong shock involved. Although the CNN adapted meshes still produce higher output errors than the adjoint adapted ones, the output error estimates of the CNN model are very accurate in this case. Compared to the transonic case considered earlier in the original testing dataset, the shock strength here is actually weaker due to the smaller camber. Since the flow problem is less nonlinear, the error estimates provided by both the CNN and the adjoint methods are more accurate, although the output convergences are still suboptimal. Note that the accuracy of the CNN-based error estimates is higher than the adjoint-based ones along the entire adaptation sequence. In

terms of the final adapted meshes, the CNN adapted mesh puts more refinement at the post-shock location, while the adjoint-based one has more refinement at the trailing edge, which turns out to be more effective as seen in the convergence plot.

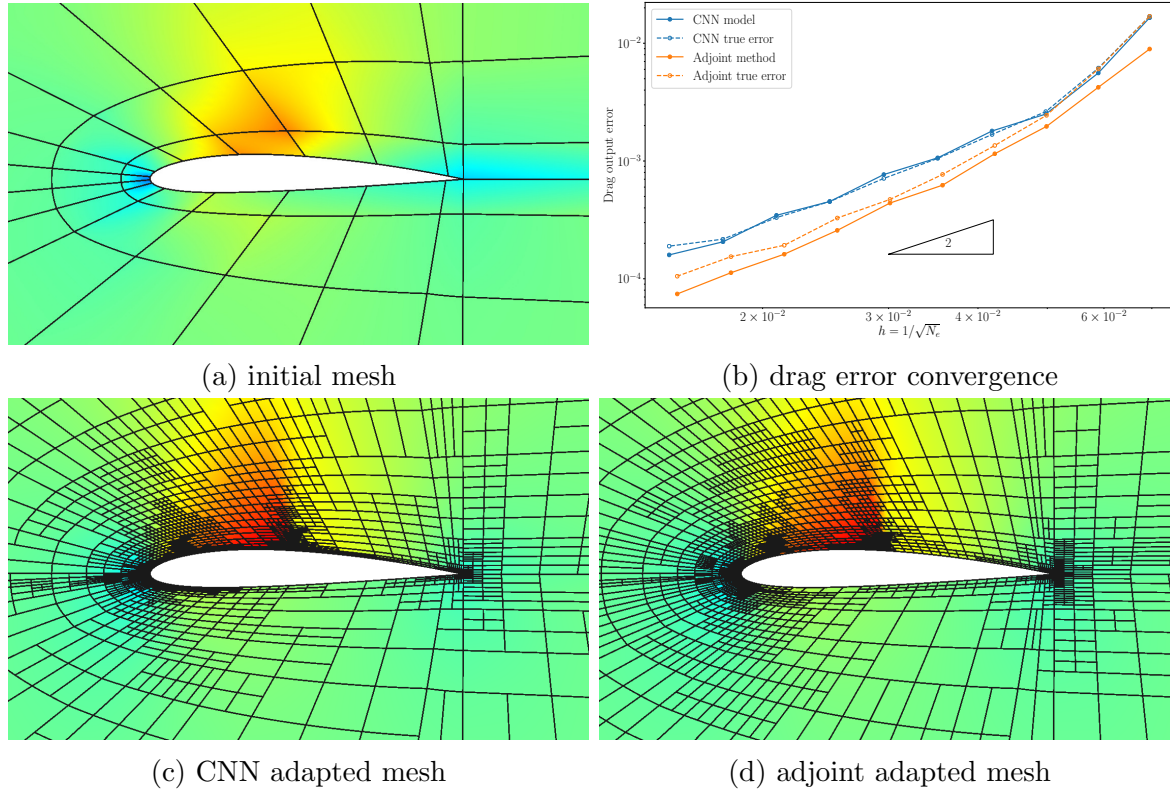


Figure 9.26: Model deployment in adaptive simulations: NACA 2412 airfoil, $M = 0.70, \alpha = 1.0^\circ$. The Mach contour scale is $[0, 1.4]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

Unseen Angles of Attack α Here we test the CNN model against angles of attack that are not seen in the original sampling space. The CNN model and the adjoint-based method are again compared in adaptive simulations. The first problem of interest is the flow over a symmetric NACA 0010 airfoil at $M = 0.60$ and $\alpha = -1.0^\circ$. Two adaptive simulation starts with the same initial mesh, using the CNN model and the adjoint method respectively, are compared in Figure 9.27. Although the current angle of attack is in the region where the network is largely agnostic of, the final CNN adapted mesh still closely follows the adaptation pattern of the adjoint method, especially the asymmetric refinements around the trailing edge. However, CNN adapted mesh tends to put more sharp refinements (most likely due to projection loss), resulting in sharp mesh

size changes around adjacent elements; while the adjoint-based adaptation produces an adapted mesh with more continuous mesh size transitions. This difference is also present in all of the tests performed above, although much more subtle compared to the current case. Despite the difference in the adapted meshes, CNN model still provides acceptable estimation for the output error.

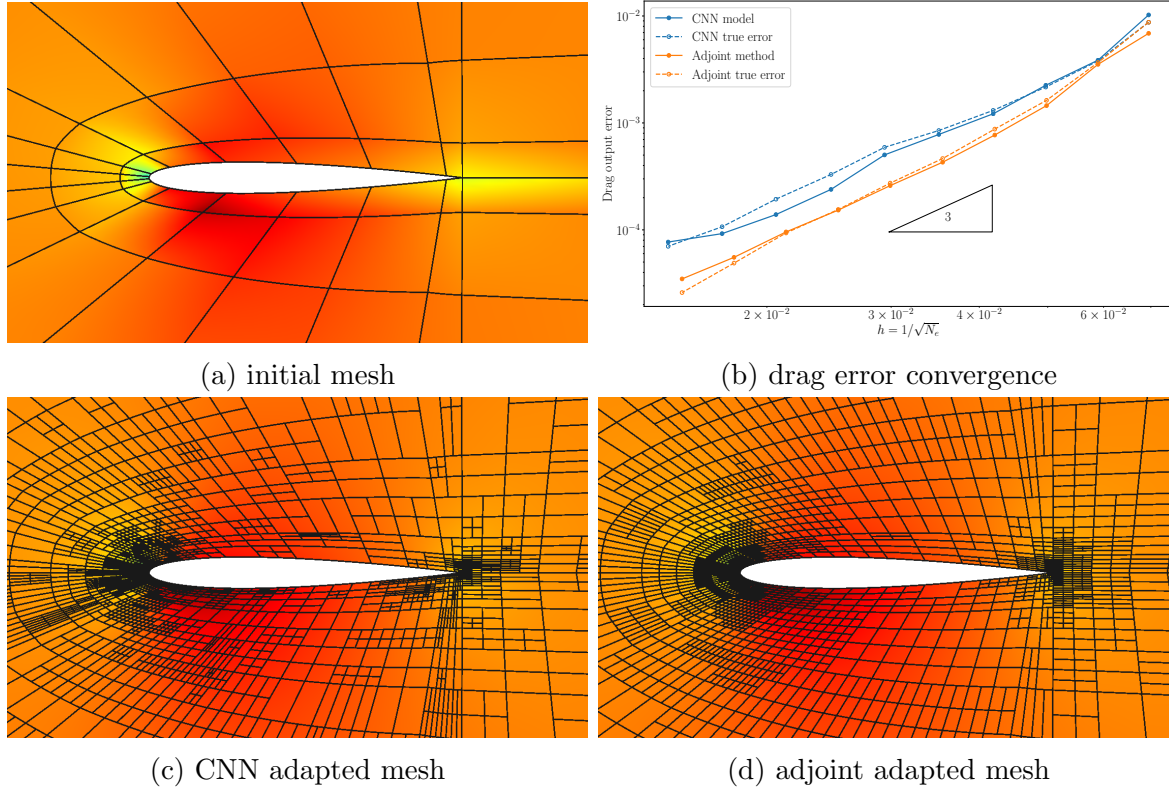


Figure 9.27: Model deployment in adaptive simulations: NACA 0010 airfoil, $M = 0.60, \alpha = -1.0^\circ$. The Mach contour scale is $[0, 0.8]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

The other case considered here is a transonic flow simulation over a NACA 4412 airfoil at $M = 0.62$ and $\alpha = 4.0^\circ$. A strong shock is present on the upper surface, which is significantly refined in both the adjoint and the CNN methods as shown in Figure 9.28. In this case, the CNN mesh features more refinements along the stagnation streamline, while the adjoint method resolves the λ -structured adjoint “shock” with considerable refinements. Although the adaptation is less effective in the CNN adapted meshes (higher output error), the output error estimation is more accurate than the adjoint-based method, especially on the coarse meshes.

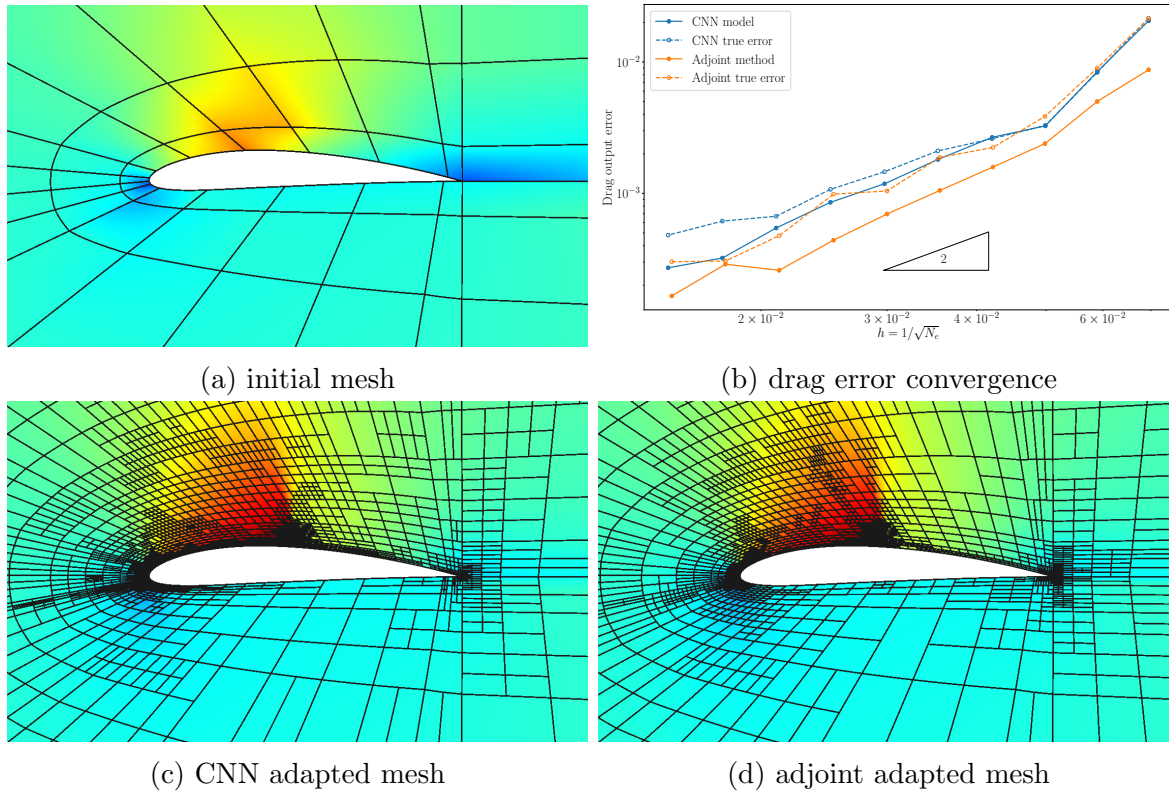


Figure 9.28: Model deployment in adaptive simulations: NACA 4412 airfoil, $M = 0.62, \alpha = 4.0^\circ$. The Mach contour scale is $[0, 1.4]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

Unseen Airfoil Shapes The last set of extrapolation tests are on the airfoil shapes that are out of the sampling space used to generate the original data. We first test the model on a NACA 3709 airfoil, which features a high camber around the aft section. The flight condition is $M = 0.66$ and $\alpha = 0^\circ$. The model performance is summarized in Figure 9.29. The CNN model is able to produce a pretty similar final adapted mesh as the adjoint adapted one, again with higher output error. As the flow field is very smooth due to a zero incidence angle, both methods provide good error estimates, while the CNN model consistently features higher accuracy in the estimates during the adaptation.

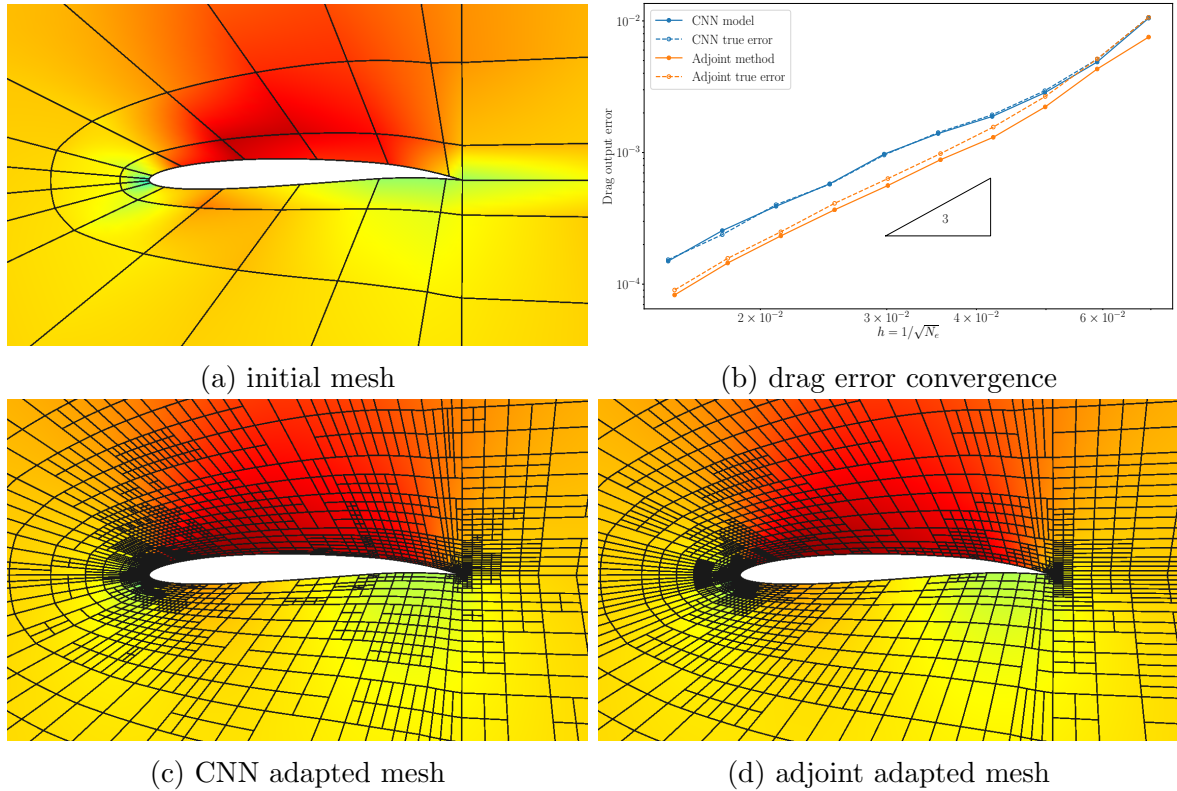


Figure 9.29: Model deployment in adaptive simulations: NACA 3709 airfoil, $M = 0.66, \alpha = 0^\circ$. The Mach contour scale is $[0, 0.96]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

The last test is performed on a NACA 5715 airfoil, at $M = 0.62$ and $\alpha = 1.0^\circ$. Final adapted meshes and output error convergence for the CNN model and the adjoint-based method are compared in Figure 9.30. Although the airfoil possesses a greater thickness and a higher camber, the shock strength is not too strong as the Mach number and the angle of attack considered are relatively low. As a result, both methods produce acceptable estimates for the output error, while the CNN model in general has a higher accuracy in

the estimates. In terms of the adapted meshes, both methods refine the areas around the strong shock on the upper surface as well as the re-acceleration region caused by the high aft-section comb. However, the CNN model focuses more on the re-acceleration region while the adjoint method resolves more the shock. The refinements at the leading and trailing edges are similar for both methods.

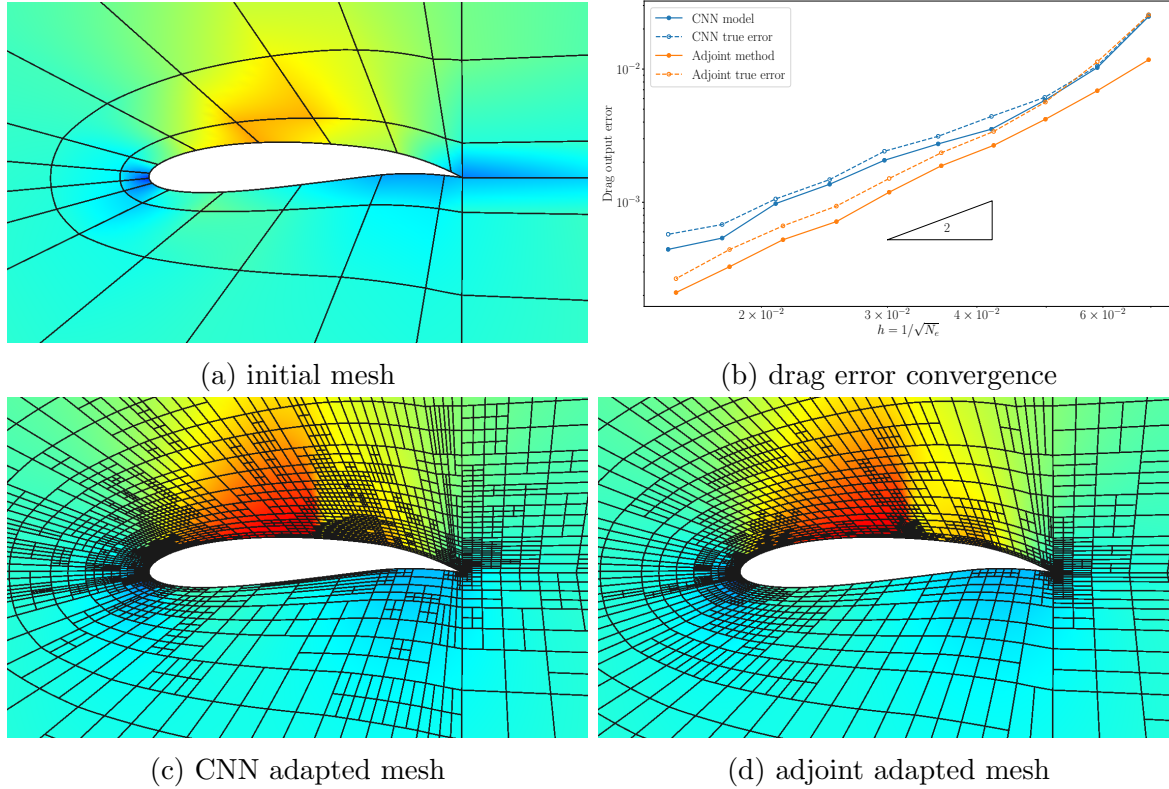


Figure 9.30: Model deployment in adaptive simulations: NACA 5715 airfoil, $M = 0.62, \alpha = 1.0^\circ$. The Mach contour scale is $[0, 1.4]$. The true output error in the convergence plot is measured with respect to the “true” output obtained on a much finer adapted mesh with $p = 2$.

9.4 Summary

Most of the output-based adaptation techniques strongly depend on the output adjoint solution as it provides a robust approach to quantify the output error and to localize the error for adaptation purposes. However, besides the high implementation cost, the additional computational cost associated with the adjoint solves cannot be ignored. We proposed a new method based on machine learning techniques for output error estimation and mesh adaptation to avoid solving for adjoint variables. The goal is to directly

predict the output error and the adaptive error indicator field from the solution field on a coarse mesh. A composite encoder-decoder type convolutional neural network, containing both convolutional and fully-connected layers, is used to construct the surrogate model. Traditional CNNs are often built on fixed Cartesian grids and thus are not readily useful for irregular domains or unstructured meshes or in an adaptive setting. To address this incompatibility, topology mapping is applied to transfer the model learning process into the reference space where a fixed Cartesian mesh can always be constructed. The feasibility of the proposed method has been demonstrated in both scalar advection-diffusion problems and inviscid flow simulations over airfoils.

The CNN model is trained using data from adjoint-based adaptation, in which a fixed fine mesh in the physical space is used as the finer space instead of order increments. A properly-trained model is able to accurately predict the adaptive error indicator field as well as the output error on unseen data in the interpolation test. The generalization of the adaptive error indicator prediction in the extrapolation regions is fairly good for both the scalar advection diffusion and the airfoil flow problems. The CNN adapted meshes closely follow the adaptation pattern of the adjoint-based method, although the mesh effectiveness is consistently lower than the adjoint-based ones, *i.e.*, higher output error. The generalization of the adaptive error indicator strongly depends on the performance of the adjoint-based method, and the model performance reduces if the adjoint-based method is not effective, *e.g.*, in highly nonlinear problems. This is expected as the model is trained on the adjoint-based adaptation data. In contrast, the output error model uses the exact difference between the coarse and fine spaces as the training data. As a result, its performance mostly depends on the data sampling and the problem complexity. When the parameter space is not well sampled or if the extrapolation point is far from the sampled training data, the model does not generalize well in the output error estimation, which has been found in the scalar advection diffusion problem. On the other hand, for complex problems, such as highly nonlinear cases, the accuracy of the error prediction also decreases. However, the performance of the CNN error estimation has been found to be better in these problems compared to the adjoint-based error estimates. The reason is that the adjoint method uses output linearization and assumes small residual perturbations for the estimates, which are often invalid for highly nonlinear problems or when the mesh is coarse; The CNN model, on the other hand, directly learns the output error model from the exact difference measured with respect to a very fine mesh. Therefore, the CNN model tends to have more accurate error estimates on coarse meshes and are more robust in highly nonlinear problems.

Presently, the network is not finely tuned, better performance may be obtained with

more tuning. Additionally, advanced training techniques such as batch normalization and dropout can also be used to improve the training efficiency and to improve the model performance. Furthermore, the symmetry of the encoder and decoder sub-networks suggests sharing the network parameters through the corresponding layers, which can substantially reduce the number of parameters and improve the training efficiency. Sparsity constraints of the latent space codes can also be added into the training loss to force the network to learn independent embedded representations. These research directions will be investigated in the future.

CHAPTER 10

Conclusions and Future Work

10.1 Summary and Conclusions

The work presented in this dissertation was motivated by the need for automated, reliable, and efficient computational fluid dynamic tools in aerodynamic design problems. With the emerging need for high-fidelity optimization and the development of unconventional configurations, the meshing process is often considered as a bottleneck for design automation. Furthermore, it is in general hard to evaluate the solution quality on a computational mesh and hence the optimized design due to unquantified discretization errors, even when the physical models are well chosen. Without properly quantifying or controlling the discretization errors, the optimization might converge to incorrect optima or work on the discretization errors instead of the physics, which significantly reduces the reliability and the efficiency of the optimization. Solution adaptive techniques, specifically the output-based mesh adaptation driven by adjoint-weighted residual are therefore introduced to address these issues. With automatic mesh adaptation, fairly coarse meshes can be used to start the optimization. Meanwhile, the discretization error can be actively quantified and controlled during the optimization with output-based error estimation and mesh adaptation. Although the optimization setup can be significantly accelerated and the solution accuracy can be greatly improved, the error estimation and mesh adaptation themselves add extra computational cost by solving the adjoint problem on an enriched space. Luckily, if adjoint-based sensitivity analysis is used, this additional computational burden can be mitigated by reuse of the adjoint solutions. In order to further improve the computational efficiency, multifidelity optimization frameworks are developed on the adapted meshes to efficiently incorporate the adaptive CFD methods with traditional gradient-based optimization. The proposed optimization frameworks with adaptive CFD techniques are demonstrated in several single-point airfoil optimization problems, showing desirable accuracy and efficiency when compared to traditional methods. Extensions

of the proposed frameworks are then made to tackle more complicated multipoint optimization problems. Individually adapted meshes on different operating conditions are adopted, with global mesh degrees of freedom redistribution among different operating conditions to further improve the overall accuracy and efficiency.

Although adjoint-based adaptive CFD has shown success in many aerospace applications, including the aerodynamic optimization problems considered in this work, the additional computational costs of the error estimation and mesh adaptation cannot be ignored, especially when the adjoint solution is not required by the original problem itself, such as gradient-free optimizations, or when the adjoint is unavailable. On the other hand, software complexity, *i.e.*, the implementation cost, has prevented the widespread use of output-based error estimation and mesh adaptation in practice. For example, the anisotropic mesh optimization method used in the current work requires a complicated sampling procedure, which either involves cumbersome mesh operations or complex projections. In order to accelerate the adaptive CFD techniques and to reduce the implementation complexity, the second half of the dissertation focuses on developing alternatives for standard mesh adaptation algorithms, where non-intrusive models based on machine learning techniques are introduced. To facilitate the anisotropic mesh adaptation, a mesh anisotropy detection model built on artificial neural networks (ANNs) is first developed. Trained using the data from standard mesh optimization techniques, the model is able to predict the mesh anisotropy directly from the primal and adjoint solution features without the sampling process. Good performance is observed on a wide range of problems involving different geometries and flight conditions, although the adjoint-weighted residual is still needed to provide adaptive error indicators for mesh sizing control. Inspired by the success of the anisotropy detection, more advanced convolutional neural networks (CNNs) are used to build a surrogate model to predict both the output error and the localized error indicator field directly from the solution field, without access to adjoint solutions. An encoder-decoder type CNN is adopted for constructing the high-dimensional map from the solution field to the error indicator field, together with fully-connected layers for output error predictions. The network trained with data from adjoint-based adaptations shows good performance and generalizes well when used in real-time adaptive simulations.

As a summary, the main observations and contributions of the current work are:

- The optimization accuracy is bounded by both the discretization error and the optimization tolerance, which should be well balanced to achieve optimal accuracy and efficiency.
- Discretization errors in the objective output calculations can produce spurious op-

tima, and, if not properly controlled, may lead to undesired designs purely caused by the numerical errors.

- The optimization tolerance, without reliable estimate of the discretization errors, is often set to be very tight in the optimization. As a result, the optimizer in general converges to a design close to the true optimum as long as the discretization error is not overly high. However, the detailed design made by the optimizer might be working on the discretization error instead of physics when the discretization errors dominate the objective changes.
- In order to achieve optimal accuracy and efficiency, an equidistribution strategy between the discretization error and the optimization tolerance is adopted in this work. Although more computational efficiency gains can possibly be achieved by allowing more optimizations on a given error level, it usually only helps improve the design while not the objective accuracy. If the objective accuracy is also important for subsequent design analysis, the equidistribution strategy is considered optimal.
- An adjoint-weighted residual is introduced to provide reliable estimates for objective errors in optimization problems. To account for the effects of constraint output errors on objective predictions, coupled adjoints are used to estimate the objective error and guide the mesh adaptation.
- Thanks to automatic mesh adaptation, coarse meshes that are much easier to generate can be used to start the optimization, even for problems involve fairly complicated physics. Furthermore, the mesh adaptation driven by error estimation actively controls the objective errors in the optimization, preventing convergence to undesired designs due to numerical errors.
- Multifidelity optimization frameworks are developed to efficiently integrate the error estimation and mesh adaptation with gradient-based optimization, taking advantage of the variable fidelity offered by adaptive meshes. The proposed frameworks follow the equidistribution of the optimization tolerance and the discretization error to achieve optimal accuracy with a given cost.
- Extensions to multipoint problems are made to accommodate optimization problems involving multiple flight conditions. Individual mesh at each flight condition is used, starting from the same coarse mesh, but is adapted independently at each flight condition. Global mesh degrees of freedom are redistributed among different flight conditions to achieve optimal accuracy and efficiency.

- The proposed frameworks showed good performance on both the single-point and the multipoint aerodynamic optimization problems considered. Both accuracy and efficiency gains are achieved compared to traditional optimization with fixed meshes.
- Cost-based integration of the mesh adaptation and optimization in general outperforms the error-based approach. In terms of the adaptation mechanics, mesh optimization via error sampling and synthesis (MOESS) often outperforms Hessian-based adaptation in more complex problems due to better anisotropy detection.
- To reduce the implementation cost and mitigate the possible computational overhead in the sampling procedure of MOESS, a mesh anisotropy model that predicts the optimal mesh anisotropy directly from the solution and adjoint features are developed using ANNs. With carefully design of the input and output features, the ANN anisotropy model generalizes well in a wide range of geometries and flight conditions.
- Although trained using the data from MOESS, the ANN model consistently achieve better output error convergence compared to MOESS in real-time adaptive simulations. This performance gains are most likely attributed to the regularization effects of the model such that it is less biased to extreme values or singularities in the primal and adjoint solutions.
- Although the ANN anisotropy model can effectively guide the anisotropic adaptation, it still relies on the adjoint-weighted residual to provide localized error indicators for element sizing control. An encoder-decoder type of CNN architecture is hence introduced to construct a direct map from the solution field to adaptive error indicators as well as the total output error, without solving for adjoint variables.
- Topology mapping is used to handle irregular computational domains that are often present in physical modeling but are incompatible with traditional CNN models. The proposed network structure has the potential to handle multi-scale problems with the physical-reference mapping.
- Trained on the data from adjoint-based adaptive simulations, the CNN model is able to provide effective error indicators in adaptive simulations and generalizes well on unseen geometries and flow conditions.

10.2 Future Work

In this dissertation, we have demonstrated the accuracy and efficiency benefits of adaptive CFD in aerodynamic optimization problems. To help reduce the possible computational and implementation overhead, non-intrusive models based on machine learning techniques have also been developed to accelerate the error estimation and mesh adaptation. In the meantime, several issues and potential improvements have also been identified during the current work, which are considered essential to fully exploit the potential of adaptive CFD in practical design process. They are summarized below:

- **Optimal Discretization Error and Optimization Tolerance Balance**

In this work the focus is not only the accuracy on the design itself but also the accuracy on the final objective values, hence an equidistribution strategy between the optimization tolerance and the discretization error is adopted. But for general optimizations in practice, sometimes the objective accuracy is less of a concern. In these scenarios, more optimization should be allowed given an estimated error level. However, design feasibility may bear less discretization errors than the design optimality, since the constraint satisfactions depend more on the absolute output predictions. Therefore, the optimal balance between the discretization error and the optimization tolerance is rather complicated and more detailed investigation is required.

- **Mesh Deformation for Highly-Anisotropic Meshes**

Excessive anisotropic mesh clustering is often found in boundary layers for high Reynolds number turbulent flows, which poses challenges for the mesh deformation during the optimization, especially when curved unstructured meshes are used. The mesh deformation techniques used in the current work has already exhibited problems such as negative volumes in our test cases. The situation can be even worse for curved boundary meshes in three-dimensional problems. To make anisotropic adaptive meshes more robust in aerodynamic optimization, more efforts are needed for improving the mesh deformation techniques, specifically for high-order anisotropic unstructured meshes.

- **More Advanced Mesh Adaptation Mechanics**

A representative flow field of transonic aerodynamic optimization problems often features both strong discontinuities, *i.e.*, shocks, and smooth flow features like flow acceleration regions and wakes. Incorporating order and mesh adaptation, *i.e.*, h - p adaptation, is expected to be more effective than the h adaptation itself. However,

more algorithmic work has to be done for efficient integration of the adaptation methods with optimization frameworks.

- **Mesh Size Predictions in Anisotropic Adaptation**

In the ANN mesh anisotropy model developed in this work, the mesh sizing control still relies on the adjoint-weighted residual. Although the implementation cost is largely reduced by avoiding the tedious sampling procedure, the computational cost saving is still limited. Element size predictions embedded in the anisotropy model based on ANNs will be more favorable in practice.

- **Handling Multi-Scale Problems in CNN Models**

For complicated flow problems such as high Reynolds number turbulent flows, the ability of handling multi-scale flow features is essential. Although the CNN model proposed in the current work has the ability to handle multi-scale problems through the physical-reference mapping, it requires the mapping information as input features and is not implemented yet. Moreover, the physical-reference mapping information, *e.g.*, the mapping Jacobian, is most likely to vary dramatically in the computational domain. Proper normalization is required to accelerate the model training and help generalize the model.

- **Non-Intrusive Model Deployment in Aerodynamic Optimization**

The non-intrusive models based on machine learning techniques have been tested and shown effectiveness in standalone adaptive simulations, while the performance when coupling with aerodynamic optimization problems still requires more studies.

BIBLIOGRAPHY

- [1] de Juniac, A., “IATA Annual Review 2019,” Tech. Rep. 75th Annual General Meeting, International Air Transport Association, Seoul, Korea, June 2019, Retrieved from <https://www.iata.org/en/publications/annual-review> (2020-02-21).
- [2] “Commercial Market Outlook 2019-2038,” Tech. rep., Boeing Company, Chicago, United States, 2019, Retrieved from <https://www.boeing.com/commercial/market/commercial-market-outlook> (2020-02-21).
- [3] “Global Market Forecast 2019-2038: Cities, Airports & Aircraft,” Tech. rep., Airbus, Blagnac Cedex, France, Aug. 2019, Retrieved from <https://www.airbus.com/aircraft/market/global-market-forecast.html> (2020-02-21).
- [4] “Market Outlook 2019-2038,” Tech. rep., Embraer, São José dos Campos, Brazil, 2019, Retrieved from <https://www.embraermarketoutlook2019.com> (2020-02-21).
- [5] “Economic Impacts of COVID-19 on Civil Aviation,” Tech. rep., International Civil Aviation Organization, Montréal, Canada, 2019, Retrieved from <https://www.icao.int/sustainability/Pages/Economic-Impacts-of-COVID-19.aspx> (2020-05-23).
- [6] “Safely Restarting Aviation: ACI and IATA Joint Approach,” Tech. rep., International Civil Aviation Organization and Airports Council International, 2020, Retrieved from <https://www.iata.org/contentassets/5c8786230ff34e2da406c72a52030e95/safely-restart-aviation-joint-aci-iata-approach.pdf> (2020-05-23).
- [7] “ICAO Environmental Report 2019: Aviation and Environment,” Tech. rep., International Civil Aviation Organization, Montréal, Canada, 2019, Retrieved from <https://www.icao.int/environmental-protection/Pages/envrep2019.aspx> (2020-02-21).

- [8] “Fact Sheet: Climate Change & CORSIA,” Tech. rep., International Air Transport Association, Montréal, Canada, May 2018, Retrieved from <https://www.iata.org/en/iata-repository/pressroom/fact-sheets/fact-sheet---industry-statistics> (2020-02-21).
- [9] Graver, B., Zhang, K., and Rutherford, D., “CO₂ emissions from commercial aviation, 2018,” Tech. rep., The International Council on Clean Transportation, Sept. 2019, Retrieved from https://theicct.org/sites/default/files/publications/ICCT_CO2-commercl-aviation-2018_20190918.pdf (2020-02-21).
- [10] “Fact Sheet: Industry Statistics,” Tech. rep., International Air Transport Association, Montréal, Canada, Dec. 2019, Retrieved from <https://www.iata.org/en/iata-repository/pressroom/fact-sheets/fact-sheet---climate-change> (2020-02-21).
- [11] Borsky, P. N., “Community Reactions to Sonic Booms in the Oklahoma City Area,” Tech. rep., National Opinion Research Center, New York, United States, Feb. 1965, Retrieved from <https://apps.dtic.mil/docs/citations/AD0613620> (2020-02-22).
- [12] Rutherford, D., Graver, B., and Chen, C., “Noise and climate impacts of an unconstrained commercial supersonic network,” Tech. rep., The International Council on Clean Transportation, Jan. 2019, Retrieved from https://theicct.org/sites/default/files/publications/Supersonic_Impact_Working_Paper_20190130.pdf (2020-02-21).
- [13] “Beginner’s Guide to Aviation efficiency,” Tech. rep., Air Transport Action Group, Geneva, Switzerland, Nov. 2010, Retrieved from <https://www.atag.org/our-publications/latest-publications.html> (2020-02-21).
- [14] “ICAO Environmental Report 2010: Aviation and Climate Change,” Tech. rep., International Civil Aviation Organization, Montréal, Canada, 2010, Retrieved from <https://www.icao.int/environmental-protection/Pages/EnvReport10.aspx> (2020-02-21).
- [15] Liebeck, R. H., “Design of the Blended Wing Body Subsonic Transport,” *Journal of Aircraft*, Vol. 41, No. 1, Jan. 2004, pp. 10–25, doi:[10.2514/1.9084](https://doi.org/10.2514/1.9084).
- [16] Wolkovitch, J., “The Joined Wing: An Overview,” *Journal of Aircraft*, Vol. 23, No. 3, March 1986, pp. 161–178, doi:[10.2514/3.45285](https://doi.org/10.2514/3.45285).

- [17] Pfenninger, W., “Design Considerations of Large Subsonic Long Range Transport Airplanes with Low Drag Boundary Layer Suction,” Tech. Rep. NAI-58-529, Northrop Aircraft, Inc., 1958.
- [18] Drela, M., “Development of the D8 Transport Configuration,” *29th AIAA Applied Aerodynamics Conference*, AIAA Paper 2011-3907, June 2011, doi:[10.2514/6.2011-3970](https://doi.org/10.2514/6.2011-3970).
- [19] Clark, L. and Gerhold, C., “Inlet noise reduction by shielding for the blended-wing-body airplane,” *5th AIAA/CEAS Aeroacoustics Conference and Exhibit*, AIAA Paper 1999-1937, May 1999, doi:[10.2514/6.1999-1937](https://doi.org/10.2514/6.1999-1937).
- [20] Lyu, Z. and Martins, J. R. R. A., “Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft,” *Journal of Aircraft*, Vol. 51, No. 5, Sept. 2014, pp. 1604–1617, doi:[10.2514/1.c032491](https://doi.org/10.2514/1.c032491).
- [21] Peigin, S. and Epstein, B., “Computational Fluid Dynamics Driven Optimization of Blended Wing Body Aircraft,” *AIAA Journal*, Vol. 44, No. 11, Nov. 2006, pp. 2736–2745, doi:[10.2514/1.19757](https://doi.org/10.2514/1.19757).
- [22] Gallman, J. W., Smith, S. C., and Kroo, I. M., “Optimization of joined-wing aircraft,” *Journal of Aircraft*, Vol. 30, No. 6, Nov. 1993, pp. 897–905, doi:[10.2514/3.46432](https://doi.org/10.2514/3.46432).
- [23] Gagnon, H. and Zingg, D. W., “Euler-Equation-Based Drag Minimization of Unconventional Aircraft Configurations,” *Journal of Aircraft*, Vol. 53, No. 5, Sept. 2016, pp. 1361–1371, doi:[10.2514/1.c033591](https://doi.org/10.2514/1.c033591).
- [24] Ivaldi, D., Secco, N. R., Chen, S., Hwang, J. T., and Martins, J., “Aerodynamic Shape Optimization of a Truss-Braced-Wing Aircraft,” *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA Paper 2015-3436, June 2015, doi:[10.2514/6.2015-3436](https://doi.org/10.2514/6.2015-3436).
- [25] Secco, N. R. and Martins, J. R. R. A., “RANS-Based Aerodynamic Shape Optimization of a Strut-Braced Wing with Overset Meshes,” *Journal of Aircraft*, Vol. 56, No. 1, Jan. 2019, pp. 217–227, doi:[10.2514/1.c034934](https://doi.org/10.2514/1.c034934).
- [26] Yutko, B. M., Titchener, N., Courtin, C., Lieu, M., Wirsing, L., Tylko, J., Jeffrey, C. T., Roberts, T. W., and Church, C. S., “Conceptual Design of a D8 Commercial Aircraft,” *17th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA Paper 2017-3509, June 2017, doi:[10.2514/6.2017-3590](https://doi.org/10.2514/6.2017-3590).

- [27] Hicks, R. M., Murman, E. M., and Vanderplaats, G. N., “An assessment of airfoil design by numerical optimization,” Tech. Rep. NASA-TM-X-3092, NASA Ames Research Center, Moffett Field, California, United States, July 1974, Available at <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19740020369.pdf>.
- [28] Hicks, R. M. and Henne, P. A., “Wing Design by Numerical Optimization,” *Journal of Aircraft*, Vol. 15, No. 7, July 1978, pp. 407–412, doi:[10.2514/3.58379](https://doi.org/10.2514/3.58379).
- [29] Hussaini, M. Y., van Leer, B., and Rosendale, J. V., editors, *Upwind and High-Resolution Schemes*, Springer Berlin Heidelberg, 1997, doi:[10.1007/978-3-642-60543-7](https://doi.org/10.1007/978-3-642-60543-7).
- [30] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., “The complex-step derivative approximation,” *ACM Transactions on Mathematical Software (TOMS)*, Vol. 29, No. 3, Sept. 2003, pp. 245–262, doi:[10.1145/838250.838251](https://doi.org/10.1145/838250.838251).
- [31] Griewank, A. and Walther, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Society for Industrial and Applied Mathematics, Jan. 2008, doi:[10.1137/1.9780898717761](https://doi.org/10.1137/1.9780898717761).
- [32] Jameson, A., “Aerodynamic design via control theory,” *Journal of Scientific Computing*, Vol. 3, No. 3, September 1988, pp. 233–260, doi:[10.1007/bf01061285](https://doi.org/10.1007/bf01061285).
- [33] Jameson, A., “Computational Aerodynamics for Aircraft Design,” *Science*, Vol. 245, No. 4916, July 1989, pp. 361–371, doi:[10.1126/science.245.4916.361](https://doi.org/10.1126/science.245.4916.361).
- [34] Jameson, A., “Automatic design of transonic airfoils to reduce the shock induced pressure drag,” *31st Israel Annual Conference on Aviation and Aeronautics*, Tel Aviv, Feb. 1990, pp. 5–17, Available at <http://aero-comlab.stanford.edu/Papers/jameson.133.pdf>.
- [35] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., “Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1,” *Journal of Aircraft*, Vol. 36, No. 1, January 1999, pp. 51–60, doi:[10.2514/2.2413](https://doi.org/10.2514/2.2413).
- [36] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., “Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2,” *Journal of Aircraft*, Vol. 36, No. 1, January 1999, pp. 61–74, doi:[10.2514/2.2414](https://doi.org/10.2514/2.2414).

- [37] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., “High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet,” *Journal of Aircraft*, Vol. 41, No. 3, May 2004, pp. 523–530, doi:[10.2514/1.11478](https://doi.org/10.2514/1.11478).
- [38] Anderson, W. and Venkatakrisnan, V., “Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation,” *Computers & Fluids*, Vol. 28, No. 4-5, May 1999, pp. 443–480, doi:[10.1016/s0045-7930\(98\)00041-3](https://doi.org/10.1016/s0045-7930(98)00041-3).
- [39] Nielsen, E. J. and Anderson, W. K., “Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations,” *AIAA Journal*, Vol. 37, No. 11, Nov. 1999, pp. 1411–1419, doi:[10.2514/2.640](https://doi.org/10.2514/2.640).
- [40] Buckley, H. P. and Zingg, D. W., “Approach to Aerodynamic Design Through Numerical Optimization,” *AIAA Journal*, Vol. 51, No. 8, Aug. 2013, pp. 1972–1981, doi:[10.2514/1.j052268](https://doi.org/10.2514/1.j052268).
- [41] Osusky, L., Buckley, H., Reist, T., and Zingg, D. W., “Drag Minimization Based on the Navier–Stokes Equations Using a Newton–Krylov Approach,” *AIAA Journal*, Vol. 53, No. 6, June 2015, pp. 1555–1577, doi:[10.2514/1.j053457](https://doi.org/10.2514/1.j053457).
- [42] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., “Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark,” *AIAA Journal*, Vol. 53, No. 4, April 2015, pp. 968–985, doi:[10.2514/1.j053318](https://doi.org/10.2514/1.j053318).
- [43] Cliff, S. E., Reuther, J. J., Saunders, D. A., and Hicks, R. M., “Single-Point and Multipoint Aerodynamic Shape Optimization of High-Speed Civil Transport,” *Journal of Aircraft*, Vol. 38, No. 6, November 2001, pp. 997–1005, doi:[10.2514/2.2886](https://doi.org/10.2514/2.2886).
- [44] Kenway, G. K. W. and Martins, J. R. R. A., “Multipoint Aerodynamic Shape Optimization Investigations of the Common Research Model Wing,” *AIAA Journal*, Vol. 54, No. 1, January 2016, pp. 113–128, doi:[10.2514/1.j054154](https://doi.org/10.2514/1.j054154).
- [45] Nadarajah, S. K. and Jameson, A., “Optimum Shape Design for Unsteady Flows with Time-Accurate Continuous and Discrete Adjoint Method,” *AIAA Journal*, Vol. 45, No. 7, July 2007, pp. 1478–1491, doi:[10.2514/1.24332](https://doi.org/10.2514/1.24332).
- [46] Hicken, J. E. and Zingg, D. W., “Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement,” *AIAA Journal*, Vol. 48, No. 2, Feb. 2010, pp. 400–413, doi:[10.2514/1.44033](https://doi.org/10.2514/1.44033).

- [47] Zingg, D. W., Nemec, M., and Pulliam, T. H., “A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization,” *European Journal of Computational Mechanics*, Vol. 17, No. 1-2, Jan. 2008, pp. 103–126, doi:[10.3166/remn.17.103-126](https://doi.org/10.3166/remn.17.103-126).
- [48] Lyu, Z., Xu, Z., and Martins, J., “Benchmarking optimization algorithms for wing aerodynamic design optimization,” *8th International Conference on Computational Fluid Dynamics*, ICCFD8-2014-0203, Chengdu, Sichuan, China, July 2014.
- [49] Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., “pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization,” *Structures and Multidisciplinary Optimization*, Vol. 45, No. 1, 2012, pp. 101–118, doi:[10.1007/s00158-011-0666-3](https://doi.org/10.1007/s00158-011-0666-3).
- [50] Gill, P. E., Murray, W., and Saunders, M. A., “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization,” *SIAM Review*, Vol. 47, No. 1, Jan. 2005, pp. 99–131, doi:[10.1137/s0036144504446096](https://doi.org/10.1137/s0036144504446096).
- [51] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and Contributors, S. . ., “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, Vol. 17, 2020, pp. 261–272, doi:<https://doi.org/10.1038/s41592-019-0686-2>.
- [52] Masters, D. A., Taylor, N. J., Rendall, T., Allen, C. B., and Poole, D. J., “Review of Aerofoil Parameterisation Methods for Aerodynamic Shape Optimisation,” *53rd AIAA Aerospace Sciences Meeting*, AIAA Paper 2015-0716, Jan. 2015, doi:[10.2514/6.2015-0761](https://doi.org/10.2514/6.2015-0761).
- [53] MM, S. and RP, K., “Mesh Deformation Approaches – A Survey,” *Journal of Physical Mathematics*, Vol. 7, No. 2, 2016, doi:[10.4172/2090-0902.1000181](https://doi.org/10.4172/2090-0902.1000181).
- [54] Michal, T., Babcock, D., Kamenetskiy, D., Krakos, J., Mani, M., Glasby, R., Erwin, T., and Stefanski, D. L., “Comparison of Fixed and Adaptive Unstructured Grid Results for Drag Prediction Workshop 6,” *Journal of Aircraft*, Vol. 55, No. 4, July 2018, pp. 1420–1432, doi:[10.2514/1.c034491](https://doi.org/10.2514/1.c034491).

- [55] Martins, J., “Perspectives on aerodynamic design optimization,” *AIAA Scitech 2020 Forum*, AIAA Paper 2020-0043, Jan. 2020, doi:[10.2514/6.2020-0043](https://doi.org/10.2514/6.2020-0043).
- [56] LeDoux, S. T., Vassberg, J. C., Young, D. P., Fugal, S., Kamenetskiy, D., Huffman, W. P., Melvin, R. G., and Smith, M. F., “Study Based on the AIAA Aerodynamic Design Optimization Discussion Group Test Cases,” *AIAA Journal*, Vol. 53, No. 7, July 2015, pp. 1910–1935, doi:[10.2514/1.j053535](https://doi.org/10.2514/1.j053535).
- [57] Destarac, D., Carrier, G., Anderson, G. R., Nadarajah, S., Poole, D. J., Vassberg, J. C., and Zingg, D. W., “Example of a Pitfall in Aerodynamic Shape Optimization,” *AIAA Journal*, Vol. 56, No. 4, April 2018, pp. 1532–1540, doi:[10.2514/1.j056128](https://doi.org/10.2514/1.j056128).
- [58] Levy, D. W., Zickuhr, T., Vassberg, J., Agrawal, S., Wahls, R. A., Pirzadeh, S., and Hensch, M. J., “Data Summary from the First AIAA Computational Fluid Dynamics Drag Prediction Workshop,” *Journal of Aircraft*, Vol. 40, No. 5, Sept. 2003, pp. 875–882, doi:[10.2514/2.6877](https://doi.org/10.2514/2.6877).
- [59] Laffin, K. R., Klausmeyer, S. M., Zickuhr, T., Vassberg, J. C., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Rakowitz, M. E., Tinoco, E. N., and Godard, J.-L., “Data Summary from Second AIAA Computational Fluid Dynamics Drag Prediction Workshop,” *Journal of Aircraft*, Vol. 42, No. 5, Sept. 2005, pp. 1165–1178, doi:[10.2514/1.10771](https://doi.org/10.2514/1.10771).
- [60] Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Einfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Laffin, K. R., and Mavriplis, D. J., “Abridged Summary of the Third AIAA Computational Fluid Dynamics Drag Prediction Workshop,” *Journal of Aircraft*, Vol. 45, No. 3, May 2008, pp. 781–798, doi:[10.2514/1.30572](https://doi.org/10.2514/1.30572).
- [61] Vassberg, J., Tinoco, E., Mani, M., Rider, B., Zickuhr, T., Levy, D., Brodersen, O., Einfeld, B., Crippa, S., Wahls, R., Morrison, J., Mavriplis, D., and Murayama, M., “Summary of the Fourth AIAA CFD Drag Prediction Workshop,” *28th AIAA Applied Aerodynamics Conference*, AIAA Paper 2010-4547, June 2010, doi:[10.2514/6.2010-4547](https://doi.org/10.2514/6.2010-4547).
- [62] Levy, D. W., Laffin, K. R., Tinoco, E. N., Vassberg, J. C., Mani, M., Rider, B., Rumsey, C. L., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Crippa, S., Mavriplis, D. J., and Murayama, M., “Summary of Data from the Fifth Computational Fluid Dynamics Drag Prediction Workshop,” *Journal of Aircraft*, Vol. 51, No. 4, July 2014, pp. 1194–1213, doi:[10.2514/1.c032389](https://doi.org/10.2514/1.c032389).

- [63] Mavriplis, D. J., Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Einfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Levy, D., and Murayama, M., “Grid Quality and Resolution Issues from the Drag Prediction Workshop Series,” *Journal of Aircraft*, Vol. 46, No. 3, May 2009, pp. 935–950, doi:[10.2514/1.39201](https://doi.org/10.2514/1.39201).
- [64] Tinoco, E. N., Brodersen, O. P., Keye, S., Laffin, K. R., Feltrop, E., Vassberg, J. C., Mani, M., Rider, B., Wahls, R. A., Morrison, J. H., Hue, D., Roy, C. J., Mavriplis, D. J., and Murayama, M., “Summary Data from the Sixth AIAA CFD Drag Prediction Workshop: CRM Cases,” *Journal of Aircraft*, Vol. 55, No. 4, July 2018, pp. 1352–1379, doi:[10.2514/1.c034409](https://doi.org/10.2514/1.c034409).
- [65] Derlaga, J. M. and Morrison, J. H., “Statistical Analysis of the Sixth AIAA Drag Prediction Workshop Solutions,” *Journal of Aircraft*, Vol. 55, No. 4, July 2018, pp. 1388–1400, doi:[10.2514/1.c034938](https://doi.org/10.2514/1.c034938).
- [66] Hicken, J. E. and Alonso, J. J., “PDE-constrained optimization with error estimation and control,” *Journal of Computational Physics*, Vol. 263, April 2014, pp. 136–150, doi:[10.1016/j.jcp.2013.12.050](https://doi.org/10.1016/j.jcp.2013.12.050).
- [67] Chen, G. and Fidkowski, K. J., “Discretization error control for constrained aerodynamic shape optimization,” *Journal of Computational Physics*, Vol. 387, No. 1, June 2019, pp. 163–185, doi:[10.1016/j.jcp.2019.02.038](https://doi.org/10.1016/j.jcp.2019.02.038).
- [68] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., “High-order CFD methods: current status and perspective,” *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, Jan. 2013, pp. 811–845, doi:[10.1002/flid.3767](https://doi.org/10.1002/flid.3767).
- [69] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., “CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences,” Tech. Rep. NASA/CR-2014-218178, NASA, March 2014, Available at <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140003093.pdf>.
- [70] Liu, Y., Vinokur, M., and Wang, Z., “Spectral difference method for unstructured grids I: Basic formulation,” *Journal of Computational Physics*, Vol. 216, No. 2, Aug. 2006, pp. 780–801, doi:[10.1016/j.jcp.2006.01.024](https://doi.org/10.1016/j.jcp.2006.01.024).

- [71] Wang, Z. J., Liu, Y., May, G., and Jameson, A., “Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations,” *Journal of Scientific Computing*, Vol. 32, No. 1, Dec. 2006, pp. 45–71, doi:[10.1007/s10915-006-9113-9](https://doi.org/10.1007/s10915-006-9113-9).
- [72] Liang, C., Jameson, A., and Wang, Z., “Spectral difference method for compressible flow on unstructured grids with mixed elements,” *Journal of Computational Physics*, Vol. 228, No. 8, May 2009, pp. 2847–2858, doi:[10.1016/j.jcp.2008.12.038](https://doi.org/10.1016/j.jcp.2008.12.038).
- [73] Wang, Z., “Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids. Basic Formulation,” *Journal of Computational Physics*, Vol. 178, No. 1, May 2002, pp. 210–251, doi:[10.1006/jcph.2002.7041](https://doi.org/10.1006/jcph.2002.7041).
- [74] Wang, Z., Zhang, L., and Liu, Y., “Spectral (finite) volume method for conservation laws on unstructured grids IV: extension to two-dimensional systems,” *Journal of Computational Physics*, Vol. 194, No. 2, March 2004, pp. 716–741, doi:[10.1016/j.jcp.2003.09.012](https://doi.org/10.1016/j.jcp.2003.09.012).
- [75] Huynh, H. T., “A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods,” *18th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2007, doi:[10.2514/6.2007-4079](https://doi.org/10.2514/6.2007-4079).
- [76] Vincent, P. E., Castonguay, P., and Jameson, A., “A New Class of High-Order Energy Stable Flux Reconstruction Schemes,” *Journal of Scientific Computing*, Vol. 47, No. 1, Sept. 2010, pp. 50–72, doi:[10.1007/s10915-010-9420-z](https://doi.org/10.1007/s10915-010-9420-z).
- [77] Huynh, H., Wang, Z., and Vincent, P., “High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids,” *Computers & Fluids*, Vol. 98, July 2014, pp. 209–220, doi:[10.1016/j.compfluid.2013.12.007](https://doi.org/10.1016/j.compfluid.2013.12.007).
- [78] Vermeire, B. C., Nadarajah, S., and Tucker, P. G., “Implicit large eddy simulation using the high-order correction procedure via reconstruction scheme,” *International Journal for Numerical Methods in Fluids*, Vol. 82, No. 5, Jan. 2016, pp. 231–260, doi:[10.1002/flid.4214](https://doi.org/10.1002/flid.4214).
- [79] Hughes, T. J. R., “A simple scheme for developing ‘upwind’ finite elements,” *International Journal for Numerical Methods in Engineering*, Vol. 12, No. 9, 1978, pp. 1359–1365, doi:[10.1002/nme.1620120904](https://doi.org/10.1002/nme.1620120904).

- [80] Brooks, A. N. and Hughes, T. J., “Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, No. 1-3, Sept. 1982, pp. 199–259, doi:[10.1016/0045-7825\(82\)90071-8](https://doi.org/10.1016/0045-7825(82)90071-8).
- [81] Reed, W. and Hill, T., “Triangular mesh methods for the neutron transport equation,” Tech. rep., Los Alamos Scientific Lab, October 1973, Available: <https://www.osti.gov/servlets/purl/4491151>.
- [82] Bassi, F. and Rebay, S., “A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations,” *Journal of Computational Physics*, Vol. 131, No. 2, March 1997, pp. 267–279, doi:[10.1006/jcph.1996.5572](https://doi.org/10.1006/jcph.1996.5572).
- [83] Bassi, F. and Rebay, S., “GMRES Discontinuous Galerkin Solution of the Compressible Navier-Stokes Equations,” *Lecture Notes in Computational Science and Engineering*, Springer Berlin Heidelberg, 2000, pp. 197–208, doi:[10.1007/978-3-642-59721-3_14](https://doi.org/10.1007/978-3-642-59721-3_14).
- [84] Cockburn, B. and Shu, C.-W., “Runge–Kutta discontinuous Galerkin methods for convection-dominated problems,” *Journal of Scientific Computing*, Vol. 16, No. 3, September 2001, pp. 173–261, doi:[10.1023/a:1012873910884](https://doi.org/10.1023/a:1012873910884).
- [85] Hartmann, R. and Houston, P., “Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations,” *Journal of Computational Physics*, Vol. 183, No. 2, December 2002, pp. 508–532, doi:[10.1006/jcph.2002.7206](https://doi.org/10.1006/jcph.2002.7206).
- [86] Demkowicz, L., Devloo, P., and Oden, J., “On an h-type mesh-refinement strategy based on minimization of interpolation errors,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 53, No. 1, Oct. 1985, pp. 67–89, doi:[10.1016/0045-7825\(85\)90076-3](https://doi.org/10.1016/0045-7825(85)90076-3).
- [87] Zienkiewicz, O. C. and Zhu, J. Z., “A simple error estimator and adaptive procedure for practical engineering analysis,” *International Journal for Numerical Methods in Engineering*, Vol. 24, No. 2, Feb. 1987, pp. 337–357, doi:[10.1002/nme.1620240206](https://doi.org/10.1002/nme.1620240206).
- [88] Eriksson, K. and Johnson, C., “Adaptive Finite Element Methods for Parabolic Problems I: A Linear Model Problem,” *SIAM Journal on Numerical Analysis*, Vol. 28, No. 1, Feb. 1991, pp. 43–77, doi:[10.1137/0728003](https://doi.org/10.1137/0728003).

- [89] Demkowicz, L., Oden, J., and Strouboulis, T., “Adaptive finite elements for flow problems with moving boundaries. part I: Variational principles and a posteriori estimates,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 46, No. 2, Oct. 1984, pp. 217–251, doi:[10.1016/0045-7825\(84\)90063-x](https://doi.org/10.1016/0045-7825(84)90063-x).
- [90] Ainsworth, M. and Oden, J., “A posteriori error estimation in finite element analysis,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 142, No. 1-2, March 1997, pp. 1–88, doi:[10.1016/s0045-7825\(96\)01107-3](https://doi.org/10.1016/s0045-7825(96)01107-3).
- [91] Ainsworth, M. and Oden, J. T., *A Posteriori Error Estimation in Finite Element Analysis*, John Wiley & Sons, Inc., Aug. 2000, doi:[10.1002/9781118032824](https://doi.org/10.1002/9781118032824).
- [92] Babuška, I., Strouboulis, T., and Upadhyay, C., “A model study of the quality of a posteriori error estimators for linear elliptic problems. Error estimation in the interior of patchwise uniform grids of triangles,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 114, No. 3-4, April 1994, pp. 307–378, doi:[10.1016/0045-7825\(94\)90177-5](https://doi.org/10.1016/0045-7825(94)90177-5).
- [93] Babuška, I., Strouboulis, T., Upadhyay, C. S., Gangaraj, S. K., and Copps, K., “Validation of a posteriori error estimators by numerical approach,” *International Journal for Numerical Methods in Engineering*, Vol. 37, No. 7, April 1994, pp. 1073–1123, doi:[10.1002/nme.1620370702](https://doi.org/10.1002/nme.1620370702).
- [94] Becker, R. and Rannacher, R., “An optimal control approach to a posteriori error estimation in finite element methods,” *Acta Numerica*, Vol. 10, May 2001, pp. 1–102, doi:[10.1017/s0962492901000010](https://doi.org/10.1017/s0962492901000010).
- [95] Pierce, N. A. and Giles, M. B., “Adjoint Recovery of Superconvergent Functionals from PDE Approximations,” *SIAM Review*, Vol. 42, No. 2, Jan. 2000, pp. 247–264, doi:[10.1137/s0036144598349423](https://doi.org/10.1137/s0036144598349423).
- [96] Giles, M. B. and Süli, E., “Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality,” *Acta Numerica*, Vol. 11, Jan. 2002, pp. 145–236, doi:[10.1017/s096249290200003x](https://doi.org/10.1017/s096249290200003x).
- [97] Venditti, D. A. and Darmofal, D. L., “Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows,” *Journal of Computational Physics*, Vol. 176, No. 1, Feb. 2002, pp. 40–69, doi:[10.1006/jcph.2001.6967](https://doi.org/10.1006/jcph.2001.6967).
- [98] Fidkowski, K. J. and Darmofal, D. L., “A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier–Stokes equations,” *Journal of*

Computational Physics, Vol. 225, No. 2, August 2007, pp. 1653–1672, doi:[10.1016/j.jcp.2007.02.007](https://doi.org/10.1016/j.jcp.2007.02.007).

- [99] Nemec, M. and Aftosmis, M., “Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes,” *18th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2007-4187, June 2007, doi:[10.2514/6.2007-4187](https://doi.org/10.2514/6.2007-4187).
- [100] Wang, L. and Mavriplis, D. J., “Adjoint-based h–p adaptive discontinuous Galerkin methods for the 2D compressible Euler equations,” *Journal of Computational Physics*, Vol. 228, No. 20, Nov. 2009, pp. 7643–7661, doi:[10.1016/j.jcp.2009.07.012](https://doi.org/10.1016/j.jcp.2009.07.012).
- [101] Fidkowski, K. J. and Darmofal, D. L., “Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics,” *AIAA Journal*, Vol. 49, No. 4, April 2011, pp. 673–694, doi:[10.2514/1.j050073](https://doi.org/10.2514/1.j050073).
- [102] Drohmann, M. and Carlberg, K., “The ROMES Method for Statistical Modeling of Reduced-Order-Model Error,” *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 3, No. 1, Jan. 2015, pp. 116–145, doi:[10.1137/140969841](https://doi.org/10.1137/140969841).
- [103] Moosavi, A., Ștefănescu, R., and Sandu, A., “Multivariate predictions of local reduced-order-model errors and dimensions,” *International Journal for Numerical Methods in Engineering*, Vol. 113, No. 3, Oct. 2017, pp. 512–533, doi:[10.1002/nme.5624](https://doi.org/10.1002/nme.5624).
- [104] Freno, B. A. and Carlberg, K. T., “Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 348, May 2019, pp. 250–296, doi:[10.1016/j.cma.2019.01.024](https://doi.org/10.1016/j.cma.2019.01.024).
- [105] Rauser, F., Korn, P., and Marotzke, J., “Predicting goal error evolution from near-initial-information: A learning algorithm,” *Journal of Computational Physics*, Vol. 230, No. 19, Aug. 2011, pp. 7284–7299, doi:[10.1016/j.jcp.2011.05.029](https://doi.org/10.1016/j.jcp.2011.05.029).
- [106] Hanna, B. N., Dinh, N. T., Youngblood, R. W., and Bolotnov, I. A., “Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD),” *Progress in Nuclear Energy*, Vol. 118, Jan. 2020, pp. 103140, doi:[10.1016/j.pnucene.2019.103140](https://doi.org/10.1016/j.pnucene.2019.103140).

- [107] Bao, H., Dinh, N. T., Lane, J. W., and Youngblood, R. W., “A data-driven framework for error estimation and mesh-model optimization in system-level thermal-hydraulic simulation,” *Nuclear Engineering and Design*, Vol. 349, Aug. 2019, pp. 27–45, doi:[10.1016/j.nucengdes.2019.04.023](https://doi.org/10.1016/j.nucengdes.2019.04.023).
- [108] Manevitz, L., Bitar, A., and Givoli, D., “Neural network time series forecasting of finite-element mesh adaptation,” *Neurocomputing*, Vol. 63, Jan. 2005, pp. 447–463, doi:[10.1016/j.neucom.2004.06.009](https://doi.org/10.1016/j.neucom.2004.06.009).
- [109] Balasubramanian, R. and Newman, J. C., “Comparison of adjoint-based and feature-based grid adaptation for functional outputs,” *International Journal for Numerical Methods in Fluids*, Vol. 53, No. 10, Oct. 2007, pp. 1541–1569, doi:[10.1002/flid.1361](https://doi.org/10.1002/flid.1361).
- [110] Kikuchi, N., Chung, K. Y., Torigaki, T., and Taylor, J. E., “Adaptive Finite Element Methods for Shape Optimization of Linearly Elastic Structures,” *The Optimum Shape*, Springer US, 1986, pp. 139–169, doi:[10.1007/978-1-4615-9483-3_6](https://doi.org/10.1007/978-1-4615-9483-3_6).
- [111] Banichuk, N. V., Barthold, F. J., Falk, A., and Stein, E., “Mesh refinement for shape optimization,” *Structural Optimization*, Vol. 9, No. 1, Feb. 1995, pp. 46–51, doi:[10.1007/bf01742644](https://doi.org/10.1007/bf01742644).
- [112] Schleupen, A., Maute, K., and Ramm, E., “Adaptive FE-procedures in shape optimization,” *Structural and Multidisciplinary Optimization*, Vol. 19, No. 4, July 2000, pp. 282–302, doi:[10.1007/s001580050125](https://doi.org/10.1007/s001580050125).
- [113] Lu, J., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2005, Available: <http://hdl.handle.net/1721.1/34134>.
- [114] Nemec, M. and Aftosmis, M., “Output Error Estimates and Mesh Refinement in Aerodynamic Shape Optimization,” *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2013-865, January 2013, doi:[10.2514/6.2013-865](https://doi.org/10.2514/6.2013-865).
- [115] Nemec, M., Aftosmis, M., and Wintzer, M., “Adjoint-based adaptive mesh refinement for complex geometries,” *46th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan 2008, p. 725, doi:[10.2514/6.2014-2576](https://doi.org/10.2514/6.2014-2576).

- [116] Li, D. and Hartmann, R., “Adjoint-based airfoil optimization with discretization error control,” *International Journal for Numerical Methods in Fluids*, Vol. 77, No. 1, January 2015, pp. 1–17, doi:[10.1002/flid.3971](https://doi.org/10.1002/flid.3971).
- [117] SPALART, P. and ALLMARAS, S., “A one-equation turbulence model for aerodynamic flows,” *30th Aerospace Sciences Meeting and Exhibit*, AIAA Paper 1992-0439, Jan. 1992, doi:[10.2514/6.1992-439](https://doi.org/10.2514/6.1992-439).
- [118] Allmaras, S., Johnson, F., and Spalart, P., “Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model,” Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.
- [119] Ceze, M. and Fidkowski, K. J., “Drag Prediction Using Adaptive Discontinuous Finite Elements,” *Journal of Aircraft*, Vol. 51, No. 4, July 2014, pp. 1284–1294, doi:[10.2514/1.c032622](https://doi.org/10.2514/1.c032622).
- [120] Bassi, F. and Rebay, S., “High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations,” *Journal of Computational Physics*, Vol. 138, No. 2, Dec. 1997, pp. 251–285, doi:[10.1006/jcph.1997.5454](https://doi.org/10.1006/jcph.1997.5454).
- [121] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., “p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations,” *Journal of Computational Physics*, Vol. 207, No. 1, July 2005, pp. 92–113, doi:[10.1016/j.jcp.2005.01.005](https://doi.org/10.1016/j.jcp.2005.01.005).
- [122] Roe, P., “Approximate Riemann solvers, parameter vectors, and difference schemes,” *Journal of Computational Physics*, Vol. 43, No. 2, October 1981, pp. 357–372, doi:[10.1016/0021-9991\(81\)90128-5](https://doi.org/10.1016/0021-9991(81)90128-5).
- [123] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., “Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems,” *SIAM Journal on Numerical Analysis*, Vol. 39, No. 5, Jan. 2002, pp. 1749–1779, doi:[10.1137/s0036142901384162](https://doi.org/10.1137/s0036142901384162).
- [124] Oliver, T. A., *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2008, Available: <http://hdl.handle.net/1721.1/46818>.

- [125] Oliver, T. A. and Darmofal, D. L., “Analysis of Dual Consistency for Discontinuous Galerkin Discretizations of Source Terms,” *SIAM Journal on Numerical Analysis*, Vol. 47, No. 5, Jan. 2009, pp. 3507–3525, doi:[10.1137/080721467](https://doi.org/10.1137/080721467).
- [126] Saad, Y. and Schultz, M. H., “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,” *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, July 1986, pp. 856–869, doi:[10.1137/0907058](https://doi.org/10.1137/0907058).
- [127] Ceze, M. and Fidkowski, K. J., “Constrained pseudo-transient continuation,” *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 11, March 2015, pp. 1683–1703, doi:[10.1002/nme.4858](https://doi.org/10.1002/nme.4858).
- [128] Persson, P.-O. and Peraire, J., “Sub-Cell Shock Capturing for Discontinuous Galerkin Methods,” *44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2006-0112, Jan. 2006, doi:[10.2514/6.2006-112](https://doi.org/10.2514/6.2006-112).
- [129] Nadarajah, S. and Jameson, A., “A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization,” *38th Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2000-667, January 2000, doi:[10.2514/6.2000-667](https://doi.org/10.2514/6.2000-667).
- [130] Nadarajah, S., *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*, Ph.D. thesis, Stanford University, Stanford, California, USA, 2003, Available at <http://aero-comlab.stanford.edu/Papers/nadarajah.thesis.pdf>.
- [131] Süli, E. and Houston, P., “Adaptive Finite Element Approximation of Hyperbolic Problems,” *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*, Springer Berlin Heidelberg, 2003, pp. 269–344, doi:[10.1007/978-3-662-05189-4_6](https://doi.org/10.1007/978-3-662-05189-4_6).
- [132] Hartmann, R., “Adjoint Consistency Analysis of Discontinuous Galerkin Discretizations,” *SIAM Journal on Numerical Analysis*, Vol. 45, No. 6, Jan. 2007, pp. 2671–2696, doi:[10.1137/060665117](https://doi.org/10.1137/060665117).
- [133] Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004, doi:[10.1017/cbo9780511804441](https://doi.org/10.1017/cbo9780511804441).
- [134] Biros, G. and Ghattas, O., “Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver,” *SIAM Journal on Scientific Computing*, Vol. 27, No. 2, Jan. 2005, pp. 687–713, doi:[10.1137/s106482750241565x](https://doi.org/10.1137/s106482750241565x).

- [135] Ilić, Č., “Comparison of Optimizer-Based and Flow Solver-Based Trimming in the Context of High-Fidelity Aerodynamic Optimization,” *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Springer International Publishing, Oct. 2017, pp. 455–465, doi:[10.1007/978-3-319-64519-3_41](https://doi.org/10.1007/978-3-319-64519-3_41).
- [136] Rothacker, B. A., Ceze, M., and Fidkowski, K., “Adjoint-Based Error Estimation and Mesh Adaptation for Problems with Output Constraints,” *32nd AIAA Applied Aerodynamics Conference*, AIAA Paper 2014-2576, June 2014, doi:[10.2514/6.2014-2576](https://doi.org/10.2514/6.2014-2576).
- [137] Martins, J. R. R. A., *A coupled-adjoint method for high-fidelity aero-structural optimization*, Ph.D. thesis, Stanford University, Stanford, California, USA, 2002, Available at <http://aero-comlab.stanford.edu/Papers/martins.thesis.pdf>.
- [138] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., “A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design,” *Optimization and Engineering*, Vol. 6, No. 1, March 2005, pp. 33–62, doi:[10.1023/b:opte.0000048536.47956.62](https://doi.org/10.1023/b:opte.0000048536.47956.62).
- [139] Barcelos, M., Bavestrello, H., and Maute, K., “A Schur–Newton–Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, No. 17-18, March 2006, pp. 2050–2069, doi:[10.1016/j.cma.2004.09.013](https://doi.org/10.1016/j.cma.2004.09.013).
- [140] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., “Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations,” *AIAA Journal*, Vol. 52, No. 5, May 2014, pp. 935–951, doi:[10.2514/1.j052255](https://doi.org/10.2514/1.j052255).
- [141] Sanchez, R., Albring, T., Palacios, R., Gauger, N. R., Economon, T. D., and Alonso, J. J., “Coupled adjoint-based sensitivities in large-displacement fluid-structure interaction using algorithmic differentiation,” *International Journal for Numerical Methods in Engineering*, Vol. 113, No. 7, Nov. 2017, pp. 1081–1107, doi:[10.1002/nme.5700](https://doi.org/10.1002/nme.5700).
- [142] Hartmann, R., “Multitarget Error Estimation and Adaptivity in Aerodynamic Flow Simulations,” *SIAM Journal on Scientific Computing*, Vol. 31, No. 1, Jan. 2008, pp. 708–731, doi:[10.1137/070710962](https://doi.org/10.1137/070710962).

- [143] Zegeling, P. A., “r-refinement for evolutionary PDEs with finite elements or finite differences,” *Applied Numerical Mathematics*, Vol. 26, No. 1-2, Jan. 1998, pp. 97–104, doi:[10.1016/s0168-9274\(97\)00086-x](https://doi.org/10.1016/s0168-9274(97)00086-x).
- [144] McRae, D., “r-Refinement grid adaptation algorithms and issues,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 189, No. 4, Sept. 2000, pp. 1161–1182, doi:[10.1016/s0045-7825\(99\)00372-2](https://doi.org/10.1016/s0045-7825(99)00372-2).
- [145] Ding, K. and Fidkowski, K., “Output Error Control Using r-Adaptation,” *23rd AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2017, doi:[10.2514/6.2017-4111](https://doi.org/10.2514/6.2017-4111).
- [146] Sanjaya, D. P. and Fidkowski, K. J., “Improving High-Order Finite Element Approximation Through Geometrical Warping,” *AIAA Journal*, Vol. 54, No. 12, Dec. 2016, pp. 3994–4010, doi:[10.2514/1.j055071](https://doi.org/10.2514/1.j055071).
- [147] Sanjaya, D. P., *Towards Automated, Metric-Conforming, Mesh Optimization For High-Order, Finite-Element Methods*, Ph.D. thesis, University of Michigan, Ann Arbor, Michigan, USA, 2019, Available: <http://hdl.handle.net/2027.42/151619>.
- [148] Castro-Díaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., “Anisotropic unstructured mesh adaption for flow simulations,” *International Journal for Numerical Methods in Fluids*, Vol. 25, No. 4, August 1997, pp. 475–491, doi:[10.1002/\(sici\)1097-0363\(19970830\)25:4<475::aid-flf575>3.0.co;2-6](https://doi.org/10.1002/(sici)1097-0363(19970830)25:4<475::aid-flf575>3.0.co;2-6).
- [149] Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M.-G., “Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles,” *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, March 2000, pp. 725–744, doi:[10.1002/\(sici\)1097-0363\(20000330\)32:6<725::aid-flf935>3.0.co;2-4](https://doi.org/10.1002/(sici)1097-0363(20000330)32:6<725::aid-flf935>3.0.co;2-4).
- [150] Frey, P. and Alauzet, F., “Anisotropic mesh adaptation for CFD computations,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 48-49, November 2005, pp. 5068–5082, doi:[10.1016/j.cma.2004.11.025](https://doi.org/10.1016/j.cma.2004.11.025).
- [151] Venditti, D. A. and Darmofal, D. L., “Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows,” *Journal of Computational Physics*, Vol. 187, No. 1, May 2003, pp. 22–46, doi:[10.1016/s0021-9991\(03\)00074-3](https://doi.org/10.1016/s0021-9991(03)00074-3).

- [152] Loseille, A., Dervieux, A., and Alauzet, F., “Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations,” *Journal of Computational Physics*, Vol. 229, No. 8, April 2010, pp. 2866–2897, doi:[10.1016/j.jcp.2009.12.021](https://doi.org/10.1016/j.jcp.2009.12.021).
- [153] Rangarajan, A., Balan, A., and May, G., “Mesh Optimization for Discontinuous Galerkin Methods Using a Continuous Mesh Model,” *AIAA Journal*, Vol. 56, No. 10, Oct. 2018, pp. 4060–4073, doi:[10.2514/1.j056965](https://doi.org/10.2514/1.j056965).
- [154] Balan, A., Park, M. A., and Anderson, W. K., “Adjoint-based Anisotropic Mesh Adaptation for a Stabilized Finite-Element Flow Solver,” *AIAA Aviation 2019 Forum*, AIAA Paper 2019-2949, June 2019, doi:[10.2514/6.2019-2949](https://doi.org/10.2514/6.2019-2949).
- [155] Balan, A., Park, M. A., Wood, S., and Anderson, W. K., “Verification of Anisotropic Mesh Adaptation for Complex Aerospace Applications,” *AIAA Scitech 2020 Forum*, AIAA Paper 2020-0675, Jan. 2020, doi:[10.2514/6.2020-0675](https://doi.org/10.2514/6.2020-0675).
- [156] Yano, M., *An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2012, Available: <http://hdl.handle.net/1721.1/76090>,.
- [157] Yano, M. and Darmofal, D. L., “An optimization-based framework for anisotropic simplex mesh adaptation,” *Journal of Computational Physics*, Vol. 231, No. 22, September 2012, pp. 7626–7649, doi:[10.1016/j.jcp.2012.06.040](https://doi.org/10.1016/j.jcp.2012.06.040).
- [158] Loseille, A. and Alauzet, F., “Continuous Mesh Model and Well-Posed Continuous Interpolation Error Estimation,” Research Report RR-6846, INRIA, Domaine de Voluceau, Rocquencourt, France, March 2009, Available at <https://hal.inria.fr/inria-00370235/PDF/rrcontmesh.pdf>.
- [159] Hecht, F., “BAMG: Bidimensional Anisotropic Mesh Generator,” INRIA–Rocquencourt, France, 1998, Available: <https://www.ljll.math.upmc.fr/hecht/ftp/bamg/>.
- [160] Michal, T. and Krakos, J., “Anisotropic Mesh Adaptation Through Edge Primitive Operations,” *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2012-159, Jan. 2012, doi:[10.2514/6.2012-159](https://doi.org/10.2514/6.2012-159).

- [161] Park, M. and Darmofal, D., “Parallel Anisotropic Tetrahedral Adaptation,” *46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2008-0917, Jan. 2008, doi:[10.2514/6.2008-917](https://doi.org/10.2514/6.2008-917).
- [162] Ibanez, D., Barral, N., Krakos, J., Loseille, A., Michal, T., and Park, M., “First benchmark of the Unstructured Grid Adaptation Working Group,” *Procedia Engineering*, Vol. 203, 2017, pp. 154–166, doi:[10.1016/j.proeng.2017.09.800](https://doi.org/10.1016/j.proeng.2017.09.800).
- [163] Loseille, A. and Alauzet, F., “Continuous Mesh Framework Part I: Well-Posed Continuous Interpolation Error,” *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, January 2011, pp. 38–60, doi:[10.1137/090754078](https://doi.org/10.1137/090754078).
- [164] Loseille, A. and Alauzet, F., “Continuous Mesh Framework Part II: Validations and Applications,” *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, January 2011, pp. 61–86, doi:[10.1137/10078654x](https://doi.org/10.1137/10078654x).
- [165] Fidkowski, K. J., *A Simplex Cut-Cell Adaptive Method for High-order Discretizations of the Compressible Navier-Stokes Equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2007, Available at <http://hdl.handle.net/1721.1/39701>.
- [166] Carson, H. A., Huang, A. C., Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., “Mesh Optimization via Error Sampling and Synthesis: An Update,” *AIAA Scitech 2020 Forum*, AIAA Paper 2020-0087, Jan. 2020, doi:[10.2514/6.2020-0087](https://doi.org/10.2514/6.2020-0087).
- [167] Carson, H. A., Huang, A. C., Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., “Anisotropic mesh adaptation for continuous finite element discretization through mesh optimization via error sampling and synthesis,” *Journal of Computational Physics*, Vol. 420, Nov. 2020, pp. 109620, doi:[10.1016/j.jcp.2020.109620](https://doi.org/10.1016/j.jcp.2020.109620).
- [168] Dahm, J. P., *Toward Accurate, Efficient, and Robust Hybridized Discontinuous Galerkin Methods*, Ph.D. thesis, University of Michigan, Ann Arbor, Michigan, USA, 2017, Available at <http://hdl.handle.net/2027.42/137150>.
- [169] Fidkowski, K. J. and Chen, G., “Output-based mesh optimization for hybridized and embedded discontinuous Galerkin methods,” *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 5, March 2020, pp. 867–887, doi:[10.1002/nme.6248](https://doi.org/10.1002/nme.6248).

- [170] Pennec, X., Fillard, P., and Ayache, N., “A Riemannian Framework for Tensor Computing,” *International Journal of Computer Vision*, Vol. 66, No. 1, January 2006, pp. 41–66, doi:[10.1007/s11263-005-3222-z](https://doi.org/10.1007/s11263-005-3222-z).
- [171] Fidkowski, K., “A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization,” *54th AIAA Aerospace Sciences Meeting*, AIAA Paper 2016-0835, January 2016, doi:[10.2514/6.2016-0835](https://doi.org/10.2514/6.2016-0835).
- [172] Wu, H.-Y., Yang, S., Liu, F., and Tsai, H.-M., “Comparisons of Three Geometric Representations of Airfoils for Aerodynamic Optimization,” *16th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2003-4095, June 2003, doi:[10.2514/6.2003-4095](https://doi.org/10.2514/6.2003-4095).
- [173] Persson, P.-O., Bonet, J., and Peraire, J., “Discontinuous Galerkin solution of the Navier–Stokes equations on deformable domains,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, No. 17-20, April 2009, pp. 1585–1595, doi:[10.1016/j.cma.2009.01.012](https://doi.org/10.1016/j.cma.2009.01.012).
- [174] Kast, S. M. and Fidkowski, K. J., “Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains,” *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494, doi:[10.1016/j.jcp.2013.06.007](https://doi.org/10.1016/j.jcp.2013.06.007).
- [175] Luke, E., Collins, E., and Blades, E., “A fast mesh deformation method using explicit interpolation,” *Journal of Computational Physics*, Vol. 231, No. 2, January 2012, pp. 586–601, doi:[10.1016/j.jcp.2011.09.021](https://doi.org/10.1016/j.jcp.2011.09.021).
- [176] Jakobsson, S. and Amoignon, O., “Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization,” *Computers & Fluids*, Vol. 36, No. 6, July 2007, pp. 1119–1136, doi:[10.1016/j.compfluid.2006.11.002](https://doi.org/10.1016/j.compfluid.2006.11.002).
- [177] de Boer, A., van der Schoot, M., and Bijl, H., “Mesh deformation based on radial basis function interpolation,” *Computers & Structures*, Vol. 85, No. 11-14, June 2007, pp. 784–795, doi:[10.1016/j.compstruc.2007.01.013](https://doi.org/10.1016/j.compstruc.2007.01.013).
- [178] Kraft, D., “A software package for sequential quadratic programming,” Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center–Institute for Flight Mechanics, Köln, Germany, 1988.
- [179] Kraft, D., “Algorithm 733: TOMP—Fortran modules for optimal control calculations,” *ACM Transactions on Mathematical Software*, Vol. 20, No. 3, September 1994, pp. 262–281, doi:[10.1145/192115.192124](https://doi.org/10.1145/192115.192124).

- [180] Lawson, C. L. and Hanson, R. J., *Solving Least Squares Problems*, Society for Industrial and Applied Mathematics, Jan. 1995, doi:[10.1137/1.9781611971217](https://doi.org/10.1137/1.9781611971217).
- [181] Brent, R. P., *Algorithms for minimization without derivatives*, Dover Publications, Inc., Mineola, New York, 2013.
- [182] Morawetz, C. S., “On the non-existence of continuous transonic flows past profiles I,” *Communications on Pure and Applied Mathematics*, Vol. 9, No. 1, Feb. 1956, pp. 45–68, doi:[10.1002/cpa.3160090104](https://doi.org/10.1002/cpa.3160090104).
- [183] Cook, P. H., McDonald, M. A., and Firmin, M. C. P., “Aerofoil RAE 2822 pressure distributions and boundary layer and wake measurements,” Experimental Data Base for Computer Program Assessment AR-138, AGARD, 1979.
- [184] Bisson, F., Nadarajah, S., and Shi-Dong, D., “Adjoint-Based Aerodynamic Optimization Framework,” *52nd Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Jan. 2014, doi:[10.2514/6.2014-0412](https://doi.org/10.2514/6.2014-0412).
- [185] Carrier, G., Destarac, D., Dumont, A., Meheut, M., Din, I. S. E., Peter, J., Khelil, S. B., Brezillon, J., and Pestana, M., “Gradient-Based Aerodynamic Optimization with the elsA Software,” *52nd Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Jan. 2014, doi:[10.2514/6.2014-0568](https://doi.org/10.2514/6.2014-0568).
- [186] Lee, C., Koo, D., Telidetzki, K., Buckley, H., Gagnon, H., and Zingg, D. W., “Aerodynamic Shape Optimization of Benchmark Problems Using Jetstream,” *53rd AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Jan. 2015, doi:[10.2514/6.2015-0262](https://doi.org/10.2514/6.2015-0262).
- [187] Poole, D. J., Allen, C. B., and Rendall, T., “Control Point-Based Aerodynamic Shape Optimization Applied to AIAA ADODG Test Cases,” *53rd AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Jan. 2015, doi:[10.2514/6.2015-1947](https://doi.org/10.2514/6.2015-1947).
- [188] Yang, G. and Ronch, A. D., “Aerodynamic Shape Optimisation of Benchmark Problems Using SU2,” *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Jan. 2018, doi:[10.2514/6.2018-0412](https://doi.org/10.2514/6.2018-0412).
- [189] He, X., Li, J., Mader, C. A., Yildirim, A., and Martins, J. R. R. A., “Robust aerodynamic shape optimization—From a circle to an airfoil,” *Aerospace Science and Technology*, Vol. 87, April 2019, pp. 48–61, doi:[10.1016/j.ast.2019.01.051](https://doi.org/10.1016/j.ast.2019.01.051).

- [190] Nemec, M., Zingg, D. W., and Pulliam, T. H., “Multipoint and Multi-Objective Aerodynamic Shape Optimization,” *AIAA Journal*, Vol. 42, No. 6, June 2004, pp. 1057–1065, doi:[10.2514/1.10415](https://doi.org/10.2514/1.10415).
- [191] Zingg, D. W. and Elias, S., “Aerodynamic Optimization Under a Range of Operating Conditions,” *AIAA Journal*, Vol. 44, No. 11, November 2006, pp. 2787–2792, doi:[10.2514/1.23658](https://doi.org/10.2514/1.23658).
- [192] Kenway, G. K. W. and Martins, J. R. R. A., “Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration,” *Journal of Aircraft*, Vol. 51, No. 1, January 2014, pp. 144–160, doi:[10.2514/1.c032150](https://doi.org/10.2514/1.c032150).
- [193] Fidkowski, K. J., “Output-Based Space-Time Mesh Optimization for Unsteady Flows Using Continuous-in-Time Adjoint,” *Journal of Computational Physics*, Vol. 341, July 2017, pp. 258–277, doi:[10.1016/j.jcp.2017.04.005](https://doi.org/10.1016/j.jcp.2017.04.005).
- [194] Bishop, C. M. et al., *Neural networks for pattern recognition*, Oxford University Press, Inc., New York, United States, Nov. 1995.
- [195] Milano, M. and Koumoutsakos, P., “Neural Network Modeling for Near Wall Turbulent Flow,” *Journal of Computational Physics*, Vol. 182, No. 1, Oct. 2002, pp. 1–26, doi:[10.1006/jcph.2002.7146](https://doi.org/10.1006/jcph.2002.7146).
- [196] Huang, R., Hu, H., and Zhao, Y., “Nonlinear Reduced-Order Modeling for Multiple-Input/Multiple-Output Aerodynamic Systems,” *AIAA Journal*, Vol. 52, No. 6, June 2014, pp. 1219–1231, doi:[10.2514/1.j052323](https://doi.org/10.2514/1.j052323).
- [197] Ling, J., Kurzawski, A., and Templeton, J., “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” *Journal of Fluid Mechanics*, Vol. 807, Oct. 2016, pp. 155–166, doi:[10.1017/jfm.2016.615](https://doi.org/10.1017/jfm.2016.615).
- [198] Singh, A. P., Medida, S., and Duraisamy, K., “Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils,” *AIAA Journal*, Vol. 55, No. 7, July 2017, pp. 2215–2227, doi:[10.2514/1.j055595](https://doi.org/10.2514/1.j055595).
- [199] Pan, S. and Duraisamy, K., “Long-Time Predictive Modeling of Nonlinear Dynamical Systems Using Neural Networks,” *Complexity*, Vol. 2018, Dec. 2018, pp. 1–26, doi:[10.1155/2018/4801012](https://doi.org/10.1155/2018/4801012).

- [200] Raissi, M., Perdikaris, P., and Karniadakis, G., “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving non-linear partial differential equations,” *Journal of Computational Physics*, Vol. 378, Feb. 2019, pp. 686–707, doi:[10.1016/j.jcp.2018.10.045](https://doi.org/10.1016/j.jcp.2018.10.045).
- [201] Wang, J.-X., Wu, J.-L., and Xiao, H., “Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data,” *Physical Review Fluids*, Vol. 2, No. 3, March 2017, pp. 034603, doi:[10.1103/physrevfluids.2.034603](https://doi.org/10.1103/physrevfluids.2.034603).
- [202] Fidkowski, K. J. and Chen, G., “Metric-based, goal-oriented mesh adaptation using machine learning,” *Journal of Computational Physics*, 2020, Manuscript submitted for publication.
- [203] Rosenblatt, F., *Principles of neurodynamics; perceptrons and theory of brain mechanics*, Spartan Books, Washington, D.C., 1962.
- [204] Nair, V. and Hinton, G. E., “Rectified linear units improve restricted boltzmann machines,” *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [205] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015, Software available from tensorflow.org.
- [206] Kingma, D. P. and Ba, J., “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [207] Shimizu, Y. S., *Output-Based Error Estimation and Model Reduction for Chaotic Flows*, Ph.D. thesis, University of Michigan, Ann Arbor, Michigan, USA, 2019, Available at <http://hdl.handle.net/2027.42/149954>.
- [208] Knoll, D. and Keyes, D., “Jacobian-free Newton–Krylov methods: a survey of approaches and applications,” *Journal of Computational Physics*, Vol. 193, No. 2, Jan. 2004, pp. 357–397, doi:[10.1016/j.jcp.2003.08.010](https://doi.org/10.1016/j.jcp.2003.08.010).

- [209] Kenway, G. K., Mader, C. A., He, P., and Martins, J. R., “Effective adjoint approaches for computational fluid dynamics,” *Progress in Aerospace Sciences*, Vol. 110, Oct. 2019, pp. 100542, doi:[10.1016/j.paerosci.2019.05.002](https://doi.org/10.1016/j.paerosci.2019.05.002).
- [210] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y., “Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition,” *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, June 2007, doi:[10.1109/cvpr.2007.383157](https://doi.org/10.1109/cvpr.2007.383157).
- [211] Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J., “Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction,” *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 52–59, doi:[10.1007/978-3-642-21735-7_7](https://doi.org/10.1007/978-3-642-21735-7_7).
- [212] Ronneberger, O., Fischer, P., and Brox, T., “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *Lecture Notes in Computer Science*, Springer International Publishing, 2015, pp. 234–241, doi:[10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28).
- [213] Long, J., Shelhamer, E., and Darrell, T., “Fully convolutional networks for semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
- [214] Noh, H., Hong, S., and Han, B., “Learning deconvolution network for semantic segmentation,” *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.
- [215] Guo, X., Li, W., and Iorio, F., “Convolutional neural networks for steady flow approximation,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 481–490.
- [216] Zhu, Y. and Zabararas, N., “Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification,” *Journal of Computational Physics*, Vol. 366, Aug. 2018, pp. 415–447, doi:[10.1016/j.jcp.2018.04.018](https://doi.org/10.1016/j.jcp.2018.04.018).
- [217] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S., “Prediction of aerodynamic flow fields using convolutional neural networks,” *Computational Mechanics*, Vol. 64, No. 2, June 2019, pp. 525–545, doi:[10.1007/s00466-019-01740-0](https://doi.org/10.1007/s00466-019-01740-0).

- [218] Winovich, N., Ramani, K., and Lin, G., “ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains,” *Journal of Computational Physics*, Vol. 394, Oct. 2019, pp. 263–279, doi:[10.1016/j.jcp.2019.05.026](https://doi.org/10.1016/j.jcp.2019.05.026).
- [219] Ceze, M. and Fidkowski, K. J., “Anisotropic hp-Adaptation Framework for Functional Prediction,” *AIAA Journal*, Vol. 51, No. 2, Feb. 2013, pp. 492–509, doi:[10.2514/1.j051845](https://doi.org/10.2514/1.j051845).
- [220] Fidkowski, K. J., “Output-Based Error Estimation and Mesh Adaptation for Steady and Unsteady Flow Problems,” *38th Advanced CFD Lectures Series; Von Karman Institute for Fluid Dynamics (September 14–16 2015)*, edited by H. Deconinck and T. Horvath, von Karman Institute for Fluid Dynamics, 2015.
- [221] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, Vol. 1, No. 4, Dec. 1989, pp. 541–551, doi:[10.1162/neco.1989.1.4.541](https://doi.org/10.1162/neco.1989.1.4.541).
- [222] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “ImageNet Classification with Deep Convolutional Neural Networks,” *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Curran Associates, Inc., 2012, pp. 1097–1105, Available at <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [223] Simonyan, K. and Zisserman, A., “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [224] He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [225] LeCun, Y., Kavukcuoglu, K., and Farabet, C., “Convolutional networks and applications in vision,” *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, IEEE, 2010, pp. 253–256.
- [226] Dumoulin, V. and Visin, F., “A guide to convolution arithmetic for deep learning,” *arXiv preprint arXiv:1603.07285*, 2016.

- [227] Odena, A., Dumoulin, V., and Olah, C., “Deconvolution and Checkerboard Artifacts,” *Distill*, 2016, doi:[10.23915/distill.00003](https://doi.org/10.23915/distill.00003).
- [228] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., “Learning representations by back-propagating errors,” *Nature*, Vol. 323, No. 6088, Oct. 1986, pp. 533–536, doi:[10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [229] Sekar, V., Zhang, M., Shu, C., and Khoo, B. C., “Inverse Design of Airfoil Using a Deep Convolutional Neural Network,” *AIAA Journal*, Vol. 57, No. 3, March 2019, pp. 993–1003, doi:[10.2514/1.j057894](https://doi.org/10.2514/1.j057894).
- [230] Jacobs, E. N., Ward, K. E., and Pinkerton, R. M., “The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel,” Tech. Rep. NACA-TR-460, PB-177874, National Advisory Committee for Aeronautics, Washington, DC, United States, Jan. 1933, Available at <https://ntrs.nasa.gov/search.jsp?R=19930091108>.