EXPLORATORY HIGH-FIDELITY AEROSTRUCTURAL OPTIMIZATION USING AN EFFICIENT MONOLITHIC SOLUTION METHOD

by

Jenmy Zimi Zhang

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

© Copyright 2017 by Jenmy Zimi Zhang
Abstract

Exploratory High-Fidelity Aerostructural Optimization using an Efficient Monolithic Solution Method

Jenny Zimi Zhang
Doctor of Philosophy
Graduate Department of Aerospace Engineering
University of Toronto
2017

This thesis is motivated by the desire to discover fuel efficient aircraft concepts through exploratory design. An optimization methodology based on tightly integrated high-fidelity aerostructural analysis is proposed, which has the flexibility, robustness, and efficiency to contribute to this goal.

The present aerostructural optimization methodology uses an integrated geometry parameterization and mesh movement strategy, which was initially proposed for aerodynamic shape optimization. This integrated approach provides the optimizer with a large amount of geometric freedom for conducting exploratory design, while allowing for efficient and robust mesh movement in the presence of substantial shape changes. In extending this approach to aerostructural optimization, this thesis has addressed a number of important challenges. A structural mesh deformation strategy has been introduced to translate consistently the shape changes described by the geometry parameterization to the structural model. A three-field formulation of the discrete steady aerostructural residual couples the mesh movement equations with the three-dimensional Euler equations and a linear structural analysis. Gradients needed for optimization are computed with a three-field coupled adjoint approach. A number of investigations have been conducted to demonstrate the suitability and accuracy of the present methodology for use in aerostructural optimization involving substantial shape changes.

Robustness and efficiency in the coupled solution algorithms is crucial to the success of an exploratory optimization. This thesis therefore also focuses on the design of an effective monolithic solution algorithm for the proposed methodology. This involves using a Newton-Krylov method for the aerostructural analysis and a preconditioned Krylov subspace method for the coupled adjoint solution. Several aspects of the monolithic solution method have been investigated. These include appropriate strategies for scaling and matrix-vector product evaluation, as well as block preconditioning techniques that preserve the modularity between subproblems. The monolithic solution method is applied to problems with varying degrees of fluid-structural coupling, as well as a wing span optimization study. The monolithic solution algorithm typically requires 20%–70% less computing time than its partitioned counterpart. This advantage increases with increasing wing flexibility. The performance of the monolithic solution method is also much less sensitive to the choice of the solution parameters.
Acknowledgements

First and foremost I would like to express my gratitude to my supervisor, Prof. David Zingg, for his supervision and guidance. His attention to detail and insightful questions have always led me to new understandings of the material. I am also grateful for his willingness to contribute his time whenever it is needed despite his busy schedule. His patience and support have allowed and encouraged me to be as thorough as possible in my work.

I would like to thank my doctoral examination committee members, Prof. Prasanth Nair and Prof. Craig Steeves, for their insights and comments during DEC meetings and the thesis revision process. I am further thankful to everyone on my final oral examination committee, which also includes Prof. Karen E. Willcox from the Massachusetts Institute of Technology and Prof. Masayuki Yano, for their valuable time and efforts in evaluating this work. Finally, I would like to thank Prof. Clinton Groth for his early influence on my decision to pursue computational fluid dynamics as a research topic, and Prof. Joaquim Martins at the University of Michigan, Ann Arbor for sharing his multidisciplinary optimization framework for the purpose of constructing this methodology.

I have had the pleasure to work closely with Shahriar Khosravi throughout my PhD studies. His insights on the applications of aerostructural optimization have contributed directly to the design of numerous studies presented in this thesis. I have also benefited greatly from our countless discussions. I would like to thank Timothy Leung, as well as members of the former Multidisciplinary Optimization Group at UTIAS, from whom I have received a lot of guidance near the start of this project. I would like to thank my colleagues and friends at the Computational Aerodynamics Group, who have always been helpful whenever I need assistance with a less familiar topic, or simply someone to bounce ideas off. In particular, I would like to thank Hugo Gagnon for his help with concepts related to B-splines and the linear elasticity mesh movement, Thomas Reist for assisting me with the region design variables and the optimization interface, and David Brown for his insights about the nonlinear and linear flow solvers. I would like to thank the past and present CFD lab administrators for taking time out of their research to look after the computers. I would like to thank Joan DaCosta and Sarah Ramji, as well as many others I met throughout my graduate studies for their valuable friendship and support. I am grateful to have been surrounded by the talented students and researchers, and wonderful administrative staff at the institute, who have made my past few years at UTIAS a pleasant and memorable experience.

I would like to thank Chris Lam for his love and friendship, and for sharing with me the good and the not-so-good days during this PhD. I am further grateful to my parents, who have not only fostered in me the curiosity and passion for learning, but made unconditional sacrifices so that I can have a better future. I am especially indebted to my mom who is also my closest friend. My achievements would not have been possible without her continuous support and encouragement.

Finally, I am thankful for the funding provided by the Zonta International Amelia Earhart Fellowships, the Ontario Graduate Scholarship, and the Kenneth M. Molson Foundation Fellowship. Computations were performed on the GPC supercomputer at the SciNet HPC Consortium. SciNet is funded by the Canada Foundation for Innovation under the auspices of Compute Canada, the Government of Ontario, Ontario Research fund - Research Excellence, and the University of Toronto.

Jenny Zimi Zhang

University of Toronto Institute for Aerospace Studies
December, 2016
# Contents

1 Introduction ................................. 1
   1.1 Background and Motivation .................. 1
   1.2 Exploratory High-Fidelity Aerostructural Optimization ....... 2
   1.3 Meeting the Challenges in Shape Optimization ....... 3
   1.4 Robust and Efficient Solution to Coupled Problems ....... 6
      1.4.1 Partitioned Methods .................. 6
      1.4.2 Monolithic Methods ................. 7
   1.5 Objectives and Thesis Outline ......... 12

2 Aerostructural Optimization Problem Overview .......... 15

3 Aerodynamics and Structural Modules ........... 17
   3.1 Aerodynamics Module .................. 17
      3.1.1 Transformed Three-Dimensional Euler Equations .... 17
      3.1.2 Discretization and Solution to the Nonlinear Problem .... 19
      3.1.3 Distributed Solution to the Linearized Flow Problem .... 20
      3.1.4 Inviscid Surface Traction Calculation .......... 22
      3.1.5 Aerodynamic Functionals and the Flow Adjoint Equation .... 23
   3.2 Structural Module .................. 24
      3.2.1 Shell Element Formulation .......... 24
      3.2.2 Parallel Finite-Element Analysis ........ 25
      3.2.3 Force and Displacement Transfer ........ 26
      3.2.4 Structural Functionals ............. 28
      3.2.5 Calculation of Design Variable Sensitivities .... 29
   3.3 Chapter Summary .................. 29

4 Integrated Geometry Parameterization and Grid Movement ...... 31
   4.1 Geometry and Flow Grid Parameterization using B-Spline Surfaces and Volumes ..... 31
   4.2 Flow Grid Movement based on Linear Elasticity .......... 33
   4.3 Parallelizing the Flow Grid Movement Calculations ....... 35
      4.3.1 Distributed Data Storage ............. 36
      4.3.2 Local Factorization and the Additive Schwarz Preconditioning .... 37
      4.3.3 Approximate Schur Preconditioning ....... 38
      4.3.4 Scaling .................. 39
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.5</td>
<td>Parallel Mesh Solution Performance</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Structural Mesh Deformation</td>
<td>41</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Inverse FFD</td>
<td>42</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Surface-based FFD</td>
<td>44</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Comparison between Inverse and Surface-based FFD</td>
<td>47</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Other Possible Alternatives</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Geometric Gradient Calculations for Aerodynamic Shape Optimization</td>
<td>49</td>
</tr>
<tr>
<td>4.6</td>
<td>Chapter Summary</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Aerostructural Analysis and Coupled Adjoint Formulation</td>
<td>51</td>
</tr>
<tr>
<td>5.1</td>
<td>Steady-State Aerostructural Analysis</td>
<td>51</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Nonlinear Block Gauss-Seidel Iterations</td>
<td>53</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Modified Displacement Transfer</td>
<td>54</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Flow Grid Movement for Aerostructural Optimization</td>
<td>55</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Force and Energy Conservation</td>
<td>58</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Validation Based on the HIRENASD Wing</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Gradient Calculations by the Coupled Adjoint Method</td>
<td>62</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The Discrete Coupled Adjoint Problem</td>
<td>63</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Calculation of the Partial Derivative Terms</td>
<td>65</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Gradient Verification</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>Chapter Summary</td>
<td>70</td>
</tr>
<tr>
<td>6</td>
<td>Monolithic Solution to Aerostructural Analysis and Coupled Adjoint Problems</td>
<td>71</td>
</tr>
<tr>
<td>6.1</td>
<td>Aerostructural Analysis using a Newton-Krylov Method</td>
<td>72</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Scaling</td>
<td>75</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Matrix-Vector Product</td>
<td>78</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Preconditioner</td>
<td>82</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Detailed Cost Comparison between Monolithic and Partitioned Analysis</td>
<td>88</td>
</tr>
<tr>
<td>6.2</td>
<td>Monolithic Coupled Adjoint Solution</td>
<td>91</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Scaling</td>
<td>92</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Preconditioner</td>
<td>94</td>
</tr>
<tr>
<td>6.3</td>
<td>Verification</td>
<td>97</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Gradient Verification</td>
<td>97</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Inviscid Transonic Wing Sweep Optimization</td>
<td>97</td>
</tr>
<tr>
<td>6.4</td>
<td>Three-Field Formulation and Monolithic Solution Method</td>
<td>100</td>
</tr>
<tr>
<td>6.5</td>
<td>Chapter Summary</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>Results and Applications</td>
<td>103</td>
</tr>
<tr>
<td>7.1</td>
<td>The Performance of Partitioned and Monolithic Solution Methods at Different Design Conditions</td>
<td>103</td>
</tr>
<tr>
<td>7.2</td>
<td>Span Optimization Study</td>
<td>108</td>
</tr>
<tr>
<td>7.3</td>
<td>Chapter Summary</td>
<td>113</td>
</tr>
</tbody>
</table>
## 8 Conclusions, Contributions, and Future Work

<table>
<thead>
<tr>
<th>8.1 Conclusions and Contributions</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.1 Aerostructural Optimization with an Integrated Geometry Parameterization and Mesh Movement Algorithm</td>
<td>115</td>
</tr>
<tr>
<td>8.1.2 Monolithic Solution to Coupled Aerostructural Problems</td>
<td>116</td>
</tr>
<tr>
<td>8.1.3 Results and Applications</td>
<td>117</td>
</tr>
<tr>
<td>8.2 Contributions</td>
<td>118</td>
</tr>
<tr>
<td>8.3 Future Work</td>
<td>120</td>
</tr>
</tbody>
</table>

## References

121

## Appendices

<table>
<thead>
<tr>
<th>A A Two-Field Formulation with Algebraic Grid Movement</th>
<th>135</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Aerostructural Analysis</td>
<td>135</td>
</tr>
<tr>
<td>A.2 Optimization and Gradient Calculations</td>
<td>136</td>
</tr>
<tr>
<td>B Recovering the Two-Field Formulation</td>
<td>139</td>
</tr>
<tr>
<td>B.1 Coupled Adjoint Problem and the Jacobian Matrix of $\mathbf{R}_{AS}$</td>
<td>139</td>
</tr>
<tr>
<td>B.2 Gradient Calculations</td>
<td>140</td>
</tr>
<tr>
<td>C A Modular Schur Preconditioner</td>
<td>143</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Benchmarking the performance of the parallel PCG algorithm against PETSc</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Preconditioning parameters for parallel mesh solutions</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>Scaling in the mesh movement stiffness matrix</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Different block row and column scaling factors considered for $\hat{A}$</td>
<td>78</td>
</tr>
<tr>
<td>6.2</td>
<td>Computational time comparison for different matrix-vector product calculations</td>
<td>81</td>
</tr>
<tr>
<td>6.3</td>
<td>Coupled nonlinear iteration details during monolithic and partitioned analyses</td>
<td>89</td>
</tr>
<tr>
<td>6.4</td>
<td>The effects of scaling on the results of monolithic coupled adjoint solution</td>
<td>94</td>
</tr>
<tr>
<td>6.5</td>
<td>Computational time comparison between monolithic and partitioned adjoint solutions</td>
<td>96</td>
</tr>
<tr>
<td>6.6</td>
<td>Normalized differences in functional values between partitioned and monolithic methods</td>
<td>98</td>
</tr>
<tr>
<td>6.7</td>
<td>Results from the transonic sweep optimization study</td>
<td>99</td>
</tr>
<tr>
<td>7.1</td>
<td>Ten $q_\infty/E$ values used to vary the level of aerostructural coupling</td>
<td>104</td>
</tr>
<tr>
<td>7.2</td>
<td>Results from a span optimization study</td>
<td>111</td>
</tr>
<tr>
<td>7.4</td>
<td>Efficiency improvements from the monolithic method for span optimization</td>
<td>112</td>
</tr>
<tr>
<td>7.5</td>
<td>Computational time and coupled solution iterations per design iteration</td>
<td>113</td>
</tr>
</tbody>
</table>
List of Figures

4.1 Illustration of flow grid parameterization using B-spline volumes ........................................... 33
4.2 A trilinear element defined from eight control points in the B-spline volume ................................ 34
4.3 FFD volume used in the inverse FFD approach ........................................................................... 43
4.4 Linking a structural mesh point with the OML for surface-based FFD ......................................... 45
4.5 Surface-based FFD at a junction .................................................................................................. 46
4.6 The equivalent volumetric parameterization of surface-based FFD ............................................. 46
4.7 Structural mesh deformation from an initially planar wing to a C-wing ........................................ 47
5.1 Two ways to move the flow grid during aerostructural optimization ........................................... 56
5.2 Distribution of pressure coefficient on a deflected C-wing geometry ........................................... 57
5.3 Cell aspect ratio distribution of the flow grids after substantial shape changes .............................. 58
5.4 Cell orthogonality distribution of the flow grids after substantial shape changes ............................ 58
5.5 Initial planform for optimization .................................................................................................. 59
5.6 Initial structural layout within the OML ........................................................................................ 59
5.7 Grid convergence of lift and drag coefficients .............................................................................. 60
5.8 Error in the force and energy transfer ........................................................................................... 61
5.9 Comparison with experimental data from the HIRENASD project .............................................. 63
5.10 Geometry and structural thickness distribution for gradient verification ..................................... 69
5.11 Normalized difference between computed gradients and finite-difference approximations ........... 69
6.1 Test geometry used for the results throughout this chapter .......................................................... 74
6.2 Structural layout for the results throughout this chapter ............................................................... 74
6.3 Relative scaling in the blocks of the coupled Jacobian matrix ...................................................... 77
6.4 Nonlinear convergence without any Jacobian block scaling ....................................................... 77
6.5 Nonlinear convergence with different Jacobian block scaling .................................................... 78
6.6 Nonlinear convergence with different Jacobian matrix-vector product approximations ............... 81
6.7 Different block preconditioning parameters for the block Jacobi preconditioner ............................ 85
6.8 Different block preconditioning parameters for the block Gauss-Seidel preconditioner .................. 87
6.9 Nonlinear convergence with monolithic and partitioned solution methods .................................... 89
6.10 Estimated scaling in the transposed coupled Jacobian matrix before and after scaling .................. 93
6.11 Coupled adjoint convergence for the lift functional ..................................................................... 93
6.12 Coupled adjoint convergence for the KS2 functional .................................................................... 93
6.13 Preconditioning tolerances for the monolithic coupled adjoint solution ...................................... 95
6.14 Normalized differences in gradient values between partitioned and monolithic methods ............... 98
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.15 Optimization convergence using partitioned and monolithic methods</td>
<td>99</td>
</tr>
<tr>
<td>6.16 Resulting planforms from a transonic sweep optimization study</td>
<td>99</td>
</tr>
<tr>
<td>6.17 Optimized lift distributions from the transonic sweep optimization</td>
<td>100</td>
</tr>
<tr>
<td>7.1 Surface patches on the geometry used for the analysis and adjoint calculations</td>
<td>105</td>
</tr>
<tr>
<td>7.2 The component thickness distribution in the structural model.</td>
<td>105</td>
</tr>
<tr>
<td>7.3 Deflected wing shapes at different $q_{\infty}/E$ values</td>
<td>105</td>
</tr>
<tr>
<td>7.4 Aerostructural analysis time at different $q_{\infty}/E$ values</td>
<td>106</td>
</tr>
<tr>
<td>7.5 Coupled adjoint solution time at different $q_{\infty}/E$ values</td>
<td>106</td>
</tr>
<tr>
<td>7.6 Tip deflections and computational savings from using the monolithic solution method</td>
<td>107</td>
</tr>
<tr>
<td>7.7 Structural layout used for the span optimization</td>
<td>109</td>
</tr>
<tr>
<td>7.8 Wing geometry for the span optimization with the planform variables</td>
<td>109</td>
</tr>
<tr>
<td>7.9 Span optimization history for $\beta = 0.5$</td>
<td>110</td>
</tr>
<tr>
<td>7.10 Optimized thickness distribution on the upper and lower skins for different $\beta$ values</td>
<td>112</td>
</tr>
<tr>
<td>7.11 Spanwise lift distribution for different $\beta$ values</td>
<td>112</td>
</tr>
<tr>
<td>7.12 Percentage of design iterations requiring a certain number of iterations for analysis and coupled adjoint solution</td>
<td>114</td>
</tr>
</tbody>
</table>
# List of Abbreviations and Symbols

## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-dimensional</td>
</tr>
<tr>
<td>AOA</td>
<td>Angle of attack</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer-aided design</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of freedom</td>
</tr>
<tr>
<td>FD</td>
<td>Finite-difference</td>
</tr>
<tr>
<td>FFD</td>
<td>Free-form deformation</td>
</tr>
<tr>
<td>FGMRES</td>
<td>Flexible Generalized Minimal Residual method</td>
</tr>
<tr>
<td>FSI</td>
<td>Fluid-structural interaction</td>
</tr>
<tr>
<td>GCROT</td>
<td>Generalized conjugate residual with inner orthogonalization and outer truncation</td>
</tr>
<tr>
<td>GMRES</td>
<td>Generalized minimal residual method</td>
</tr>
<tr>
<td>IC</td>
<td>Incomplete Cholesky</td>
</tr>
<tr>
<td>ILU</td>
<td>Incomplete lower-upper factorization</td>
</tr>
<tr>
<td>KS</td>
<td>Kreisselmeier-Steinhauser function for stress constraint aggregation</td>
</tr>
<tr>
<td>LHS</td>
<td>Left-hand side</td>
</tr>
<tr>
<td>LU</td>
<td>Lower-upper factorization</td>
</tr>
<tr>
<td>MITC</td>
<td>Mixed interpolation of tensorial components</td>
</tr>
<tr>
<td>NK</td>
<td>Newton-Krylov</td>
</tr>
<tr>
<td>OML</td>
<td>Outer mold line</td>
</tr>
<tr>
<td>PCG</td>
<td>Preconditioned conjugate gradient</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial differential equation</td>
</tr>
<tr>
<td>RANS</td>
<td>Reynolds-Averaged Navier-Stokes equations</td>
</tr>
</tbody>
</table>
RHS  Right-hand side
RMS  Root mean square

Greek

$\Delta \hat{b}_\Delta$ The scaled Newton update to the mesh state vector

$\Delta b_{sc}^*, \Delta b_{se}^*, \Delta b_{si}^*$ Changes in the B-spline surface control point coordinates at the corners, edges, and interior of a surface patch due to structural deflections

$\delta_{FD}$ Finite-difference stepsize used to approximate the Jacobian matrix-vector product

$\delta$ An update to the coupled solution vector at each Newton iteration

$\epsilon_{AS}$ Convergence tolerance for the coupled aerostructural analysis problem

$\epsilon_A, \epsilon_M, \epsilon_S$ Convergence tolerance for the aerodynamic, mesh, and structural subproblems

$\theta$ Under-relaxation factor for partitioned solution methods

$\Theta$ Mid-surface rotations inside a shell element

$\xi, \eta, \zeta$ Parametric coordinates

$\nabla \xi$ Vector of all grid metric terms over the flow domain

$\Psi_A$ Aerodynamic adjoint vector

$\Psi_{MJ}, \Psi_{M\Delta}$ Mesh adjoint vector corresponding to the jig shape and the deflected shape

$\Psi_S$ Structural adjoint vector

Alphanumeric

$A$ Flow Jacobian matrix

$A_1$ First-order approximation to the flow Jacobian matrix

$\Lambda$ Jacobian of the coupled aerostructural residual

$\hat{\Lambda}$ Scaled aerostructural Jacobian matrix

$B$ Coordinates of a single control point

$b$ Vector of control point coordinates

$b_J$ $b$ for the jig shape

$b_\Delta$ The mesh state vector, or $b$ for the deflected geometry

$b_s$ Coordinates of surface control points from all increments

$b_{sJ}, b_{s\Delta}$ $b_s$ for the jig shape and the deflected geometry

$b_s^*$ Coordinates of surface control points describing the final geometry
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b^*_s$</td>
<td>for the jig shape and the deflected geometry</td>
</tr>
<tr>
<td>$D_{(i)}$</td>
<td>Scaling matrix containing the diagonal entries in $K_{M\Delta}^{(i)}$ at the $i^{th}$ mesh movement increment</td>
</tr>
<tr>
<td>$E$</td>
<td>Young’s modulus in the structural material</td>
</tr>
<tr>
<td>$f_A$</td>
<td>Aerodynamic surface traction</td>
</tr>
<tr>
<td>$f_M$</td>
<td>Force vector in the mesh equations</td>
</tr>
<tr>
<td>$f_{M\Delta}$</td>
<td>for the jig shape and the deflected geometry</td>
</tr>
<tr>
<td>$f_S$</td>
<td>Force vector in the structural equations</td>
</tr>
<tr>
<td>$G$</td>
<td>Vector containing all nodes on the flow grid</td>
</tr>
<tr>
<td>$G_{sJ}$</td>
<td>Coordinates of the surface grid nodes on the jig shape</td>
</tr>
<tr>
<td>$G_{s\Delta}$</td>
<td>Coordinates of the surface grid nodes on the deflected shape (in a two-field formulation)</td>
</tr>
<tr>
<td>$J$</td>
<td>Metric Jacobian for the curvilinear transformation applied to the flow problem</td>
</tr>
<tr>
<td>$J$</td>
<td>A scalar functional computed during optimization</td>
</tr>
<tr>
<td>$J_A$, $J_S$</td>
<td>Aerodynamic and structural functional</td>
</tr>
<tr>
<td>$K_M$</td>
<td>Stiffness matrix for the mesh equations</td>
</tr>
<tr>
<td>$K_{Mf}$</td>
<td>Stiffness matrix used to compute the implicit force vector in the mesh equation</td>
</tr>
<tr>
<td>$K_{M\Delta}$</td>
<td>for the jig shape and the deflected geometry</td>
</tr>
<tr>
<td>$K_S$</td>
<td>Stiffness matrix for the structural equations</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>Lagrangian function</td>
</tr>
<tr>
<td>$M_A$, $M_M$, $M_S$</td>
<td>Block preconditioners for the flow, mesh, and structural subproblems</td>
</tr>
<tr>
<td>$M_\infty$</td>
<td>Freestream Mach number</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of mesh movement increments</td>
</tr>
<tr>
<td>$m_{\Delta}$</td>
<td>$m$ used in mesh movement for the jig shape and the deflected geometry</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of nodes in the flow grid</td>
</tr>
<tr>
<td>$N_i$</td>
<td>Element shape functions in the structural solver</td>
</tr>
<tr>
<td>$n_A$, $n_M$, $n_S$</td>
<td>Total number of unknowns in the flow, mesh, and structural subproblems</td>
</tr>
<tr>
<td>$n_{Ms}$</td>
<td>Total number of surface control points</td>
</tr>
<tr>
<td>$n_i$</td>
<td>Surface normal at a surface node in the flow grid</td>
</tr>
</tbody>
</table>
\( P_i \)  
Element injection operator for the structural solver

\( P_s \)  
Restriction operator mapping from the vector of all mesh unknowns to those at the surface of the geometry

\( p \)  
Pressure

\( q_{\infty} \)  
Freestream dynamic pressure

\( q \)  
Aerodynamic state vector

\( R_{AS} \)  
Aerostructural residual vector

\( R_A \)  
Aerodynamic residual vector

\( R_{MJ}, R_{M\Delta} \)  
Mesh residual vector for the jig shape and the deflected shape

\( R_S \)  
Structural residual vector

\( \hat{R}_{AS}, \hat{R}_A, \hat{R}_{M\Delta} \)  
Scaled aerostructural residual, flow residual, and mesh residual vectors for monolithic analysis

\( r_{\text{scl}, \star}, c_{\text{scl}, \star} \)  
Jacobian block row and column scaling values used for monolithic analysis

\( r \)  
All rigid link vectors

\( S, S_i \)  
Global and local Schur-complement matrices

\( S \)  
Computed estimate of the relative scaling between the Jacobian blocks in \( \hat{\mathbf{A}} \)

\( U_0 \)  
Mid-surface displacements inside a shell element

\( u \)  
Structural state vector describing the structural deflections

\( u_A \)  
Changes in the aerodynamic surface coordinates due to structural deflections

\( u_{Ac}, u_{Ae}, u_{Ai} \)  
\( u_A \) at the corners, edges, or interior of a surface patch

\( v \)  
Design variables

\( v_A, v_G, v_S \)  
Aerodynamic design variables, geometric, and structural design variables

\( X_S \)  
Vector containing all nodes on the structural mesh/model

\( x = (x, y, z) \)  
Cartesian coordinates of a point in physical space

\( x_i^e \)  
Nodal coordinates of a structural element
Chapter 1

Introduction

1.1 Background and Motivation

Air travel today plays a crucial role in socioeconomic development around the globe. The amount of air traffic is predicted to double by the year 2030 [1]. While such a rapid growth rate is expected to continue well into the future, it is not sustainable unless the aviation industry takes appropriate actions. From an economic perspective, the profitability of the aviation industry is largely affected by fuel costs. For example, when the oil price underwent a steep increase by a factor of three during the years between 2002 and 2006, the industry lost 29.1 billion dollars worldwide [2]. The cost of fuel has since surpassed the labour cost in having the largest share of the operating budget for major airlines, at over 30% [3]. Due to the limit in oil supply and the expected increase in demand, the price of fuel will continue to be highly volatile [2], putting the aviation industry in a vulnerable position. In addition, there are environmental challenges. Air travel currently contributes 2% of global anthropogenic carbon-dioxide emissions [1]. Moreover, there are potential effects of cirrus cloud enhancement due to aviation [2]. Air traffic can therefore contribute two to four times more to radiative forcing than ground travel with the same fuel consumption. To mitigate the environmental impact of aviation, especially in light of its projected growth rate, the International Civil Aviation Organization has defined ambitious goals for the industry: to achieve a 2% annual fuel efficiency improvement, and to stabilize the global carbon-dioxide emission at its 2020 level.

Sustainable alternative fuels, such as biofuels, have the potential to address both challenges associated with the price of jet fuel and the environment. However, research into alternative fuel sources is a work in progress, and there is not yet sufficient data to predict its success with sufficient confidence [1]. In fact, ensuring sustainable growth of the aviation industry requires research efforts and investments on multiple fronts, including various technology improvements, operational changes, alternative fuels, and market-based measures. Given the large margin of uncertainty in the predicted demand in future aviation [1], this is especially important if the above goals are to be met in the highest-demand scenario.

In terms of technology improvements, the challenges faced by the aviation industry require a step change in the fuel efficiency of aircrafts via new airframe designs. Conventional tube-and-wing designs are highly optimized and offer limited room for substantial improvements. Unconventional design options must be explored in order to achieve a sufficient gain in fuel efficiency [1]. This in turn requires an evolution from the traditional cut-and-try approach to aircraft design, which no longer suffices because
it relies heavily on the knowledge and experience of the designer. Not only is such knowledge and experience not generally available for unconventional configurations, it may actually be desirable to look past conventional design wisdom to foresee new possibilities that are truly groundbreaking. Fortunately, the reliance on designers’ experience can be gradually reduced due to the emergence of numerical optimization. Numerical optimization performs an automated search of the design space for the best design, where the rapid assessment of any configuration is enabled by physics-based numerical analysis. There have been rapid advancements in numerical methods and high-performance computing facilities, which are now capable of performing large-scale calculations using from hundreds to thousands of processors. This allows for the development of powerful numerical optimization algorithms that can accelerate the discovery of next-generation novel aircraft concepts.

1.2 Exploratory High-Fidelity Aerostructural Optimization

The potential of numerical optimization for the design of unconventional aircraft has been successfully demonstrated by aerodynamic shape optimization based on high-fidelity analysis [5, 6, 7]. Examples of high-fidelity aerodynamic analysis include methods which model the three-dimensional (3D) flow around the aircraft using either the Euler or Navier-Stokes equations. Although computationally more expensive than low-fidelity models, such as those based on linear theories or empirical relations, they are capable of capturing the nonlinear effects under conditions where low-fidelity models can be inaccurate. This is especially important in the design of unconventional concepts, as it allows for the correct assessment of a novel design feature even in the absence of the designers’ past experience.

Aerodynamic shape optimization has revealed several promising design concepts that lead to reductions in drag [5, 7]. What remains an interesting and important question is how much of these benefits are offset by the possible increase in structural weight. Although some recent applications of high-fidelity aerodynamic optimization have included simplified weight models [6, 7, 8, 9, 10], the tradeoff between drag and weight is more accurately captured with a full stress analysis based on aerodynamic loading. Coupling aerodynamic and structural analysis automatically accounts for the effects of the structural deflections on the aerodynamic performance [11]. It also provides a more accurate indication of possible structural failure, reducing the reliance on artificial geometric constraints that can prevent the optimizer from finding the most efficient design [11]. Hence, optimization based on tightly integrated high-fidelity aerostructural analysis is an important step towards taking full advantage of numerical optimization.

This work is driven by the desire to combine the power of high-fidelity aerostructural analysis with an exploratory optimization capability. An exploratory optimization is characterized by a large amount of geometric freedom given to the optimizer to identify novel designs with the highest efficiency benefits. As the objective is to assess the potential of competing design concepts, it is often not necessary to consider all of the aspects involved in the detailed design of an aircraft. Exploratory optimization works with hundreds and sometimes thousands of design variables, leading to a large design space. This, together with the cost of large-scale high-fidelity calculations, makes gradient-based optimization methods the preferred option [12]. The cost of optimization can be further reduced by the use of adjoint methods, where the cost of gradient calculations is almost independent of the number of design variables [13, 14]. High-fidelity exploratory optimization can be an especially valuable tool for the potential discovery of unconventional design alternatives.

Although the use of gradient-based optimization and adjoint methods for gradient calculations are
1.3 Meeting the Challenges in Shape Optimization

Shape optimization requires a method to parameterize the geometry of interest and a control mechanism to accomplish shape changes. Geometry parameterization refers to the way that the geometry is defined in space. Perhaps the most obvious example is the discrete parameterization technique where the geometry is defined using the individual grid nodes. Other examples include parameterization via analytical shape functions such as those proposed by Hicks and Henne \cite{Hicks_Henne}, PARSEC \cite{PARSEC}, Non-Uniform Rational B-Spline (NURBS) \cite{NURBS}, or its variations such as B-spline or Bézier curves \cite{B-spline_Bezier}. In contrast, geometry control more precisely refers to the way shape changes are applied to the geometry during optimization. For instance, a geometry may be parameterized by B-spline surfaces, but controlled with free-form-deformation (FFD) volumes that make changes to the B-spline control points rather than the discrete set of surface grid nodes \cite{FFD}.

An ideal geometry parameterization describes the geometry with a compact set of design variables, yet at the same time gives the optimizer sufficient flexibility to develop design features of interest to the designer. For gradient-based optimization, the availability of geometric gradients with respect to design variables is an important consideration \cite{Geometric_Grad}. Computer-aided-design (CAD) tools are powerful for creating complex geometries in aircraft design \cite{CAD_Aircraft}. A CAD-based optimization approach uses the CAD software to make changes to the original CAD model. However, the design variable sensitivities of the geometry are often unavailable for at least the two following reasons: proprietary code within the CAD software and a geometry that does not necessarily vary smoothly due to the use of a patch topology \cite{CAD_Software_Patch}. In contrast, CAD-free methods have been developed that do not involve the use of CAD software. These avoid the above difficulties and also can be much simpler to use than a CAD package because they can be tailored to a specific application. CAD-free methods are therefore considered to be more suitable for exploratory design studies and are used exclusively in this work.

Design shape changes cannot be analyzed without efficient and robust mesh movement algorithms to deform the aerodynamic and structural domains. In the context of aerostructural optimization, shape changes across disciplines must also be consistently parameterized to maintain the accuracy of the analysis \cite{Aerostructural_Accuracy}. The aerodynamic domain undergoes further deformations during aerostructural analysis due to structural deflections. This stems from the fact that the aerodynamic analysis typically uses an Eulerian formulation \cite{Eulerian_Formulation}. An efficient aerodynamic mesh movement algorithm that is capable of handling large geometry changes is therefore essential. In contrast, the structural analysis often uses...
a Lagrangian formulation, so that the structural mesh deformation is only executed once per design cycle for changes in the unstressed geometry. Nevertheless, the structural mesh deformation should minimize the introduction of any undesirable distortions in the structural members, such as ribs and spars, modelled in high-fidelity analysis.

The above challenges have been addressed in different ways in the existing literature on high-fidelity aerostructural optimization. Farhat et al. [24] proposed a three-field formulation to handle the motion of the flow grid due to structural deflections during transient aerostructural analysis. The flow grid was modelled explicitly alongside the flow and structural equations. This led to three coupled equations in the aeroelastic problem. Maute et al. [25] applied the three-field formulation to aerostructural optimization based on steady analysis involving the Euler equations and a linear structural analysis. The flow grid was modelled based on a spring analogy. During optimization, simple geometry changes to the outer mold line (OML) of a wing as well as the detailed finite-element model of the internal structure were parameterized using a number of Coons elements. The proposed methodology was applied to the optimization of an Aeroelastic Research Wing (ARW2). They used a direct method to evaluate the gradients. The same authors later described an alternative methodology using a coupled adjoint approach for gradient calculation [26]. Barcelos and Maute [27] expanded on the three-field methodology by modelling the flow with the Navier-Stokes equations and an algebraic turbulence model, and by using a nonlinear analysis of the structures.

Reuther et al. [11] and Martins et al. [28, 29] used an OML geometry database as an interface to the optimizer and between the disciplines. This allowed for the design of more general aircraft components. The flow grid was moved by an algebraic warping algorithm which did not appear explicitly in the equations of state, resulting in a two-field formulation. Martins et al. [29] further described the corresponding coupled adjoint approach for gradient calculation. In another paper [28], the design framework was applied to the optimization of a supersonic business jet. Kenway et al. [23] proposed a way to control the OML and the internal structure using an FFD technique. Deformation of the aerodynamic domain was achieved via a hybrid linear elasticity mesh movement. Using the same geometry parameterization and mesh movement methodology, Kenway et al. [30] addressed the limitations in the work of Martins et al. [29] by coupling a more advanced Euler solver with a fully parallel structural analysis package in a high-fidelity aerostructural optimization framework. Accuracy and efficiency of the gradient calculations were improved. The described framework was used in the optimization of a NASA Common Research Model (CRM) wing-body-tail configuration [31]. In a recent publication, Kenway et al. [32] conducted aerostructural optimization studies based on the Reynolds-Averaged Navier-Stokes equations with the CRM geometry.

None of the approaches summarized above use three-dimensional B-spline patches for geometry parameterization. This is also the case in a number of other methodologies not discussed above, including that of Abu-Zurayk and Brazillon [33] and Samareh [34]. Although B-splines have been widely used for shape parameterization in purely aerodynamic and structural optimization [18, 35, 36, 37], they are clearly lacking popularity in the field of fully coupled aerostructural optimization. As Samareh [20, 34] pointed out, there are a few significant challenges that have prevented the use of B-splines for aerostructural optimization despite the many advantages they provide for shape parameterization. For instance, it is difficult to generate grids for the flow and structures after geometry changes. Furthermore, the complex three-dimensional models are difficult to create outside of a CAD system. Isogeometric analysis using NURBS volumes has recently been applied to the study of fluid-structural interaction [35] and
structural shape optimization [39]. Isogeometric analysis is motivated by advantages similar to the use of B-spline geometry parameterization, such as those involving an exact geometry representation [40]. However, the same challenges related to the flow and structural mesh deformation in the presence of large shape changes must be addressed if isogeometric analysis is to be used for high-fidelity aerostructural optimization. The successful application of B-splines for aerodynamic shape optimization suggests that its potential for aerostructural optimization in the context of aircraft design should not be overlooked [41, 42, 43]. For this reason, this thesis proposes to use the integrated geometry parameterization and mesh movement of Hicken and Zingg [44] that was initially presented for purely aerodynamic shape optimization. It has been shown to produce high quality flow meshes even for very large geometric changes. This thus enables exploratory optimization with the possibility of dramatic shape changes, where other mesh deformation algorithms may often fail [44].

The integrated approach has two key components that distinguish it from existing methodologies for aerostructural optimization: an effective means for geometry parameterization and control using B-splines, and an efficient and robust mesh movement strategy. Both are essential in exploratory optimization based on high-fidelity aerostructural analysis. This approach parameterizes the grid for flow calculations by B-spline volumes. B-spline control points on the surface of the geometry simultaneously provide effective geometry parameterization and control. There is no clear winner when it comes to the best geometry parameterization, but there are a number of inherent advantages to using B-spline surfaces. B-spline curves of order $p$ are known to lie within the convex hull of $p$ neighboring control points, and the control points approximate the curves [45]. This allows for local control of shape changes and intuitive specification of geometric constraints. It also means that the B-spline parameterization has a high surface awareness which can be exploited for additional aerodynamic benefits during optimization [41, 46]. As a result, B-spline surface control points have worked well as geometric design variables [5, 10, 47], i.e. for both parameterization and control. The physical relationship between the surface control points and the underlying geometry has also allowed them to be used as part of a two-level FFD approach [19], where the geometry control is provided by FFD volumes. Furthermore, the approximation power of piecewise smooth B-spline surface patches allows complex geometries to be analytically represented and maintained throughout optimization. The initial and optimized geometries are therefore always independent of the mesh used to approximate them. The analytical geometry description may also be used for other important purposes such as rigorous mesh refinement studies, solution-adaptive gridding during the solution process, and high-order mesh generation [40, 48, 49]. Most importantly given the context of this work, the optimizer can in theory describe any arbitrary geometry with the same topology by manipulating the coordinates of the B-spline surface control points. Few assumptions are made about the final geometry, which leaves it entirely up to the optimizer what the optimal design shall be. This results in a high degree of flexibility that is crucial for exploratory optimization [20]. This amount of flexibility is at the same time achievable using a few hundred geometric design variables given by the number of surface control points, which can in turn be chosen based on the desired level of local control and does not scale with the grid resolution. Finally, Hicken and Zingg [44] are the first to tightly integrate the geometry parameterization via B-splines to a linear elasticity mesh movement of the B-spline control volumes. This offers a novel way to reduce the cost to the traditional linear-elasticity mesh movement of the actual computational grid. Nevertheless, it is sufficiently robust to preserve the grid quality in the presence of large shape changes.

For the above reasons, the integrated geometry parameterization and mesh movement algorithm is
particularly well-suited for optimization problems involving substantial geometry changes. This has been
demonstrated via its successful application to the aerodynamic shape optimization of a wide range of
aircraft configurations [5, 7, 19, 50]. The original integrated approach by Hicken and Zingg [44] addressed
the challenges associated with creating the B-spline model and the flow grid generation after each shape
change, as pointed out by Samareh [20, 34]. However, the algorithm must be extended to handle the
additional challenges associated with aerostructural optimization.

1.4 Robust and Efficient Solution to Coupled Problems

1.4.1 Partitioned Methods

The partitioned approach is used most extensively in past research on aerostructural analysis and opti-
mization. This includes a number of the aerostructural design methodologies summarized in Section 1.3,
namely the works of Reuther et al. [11], Maute et al. [25, 26], and Martins et al. [28, 29]. This is largely
due to the ease of implementing the partitioned methods, especially for aerostructural optimization
frameworks constructed over existing flow and structural modules.

Partitioned methods allow each of the flow, structures, and mesh equations in the coupled analysis
and adjoint problems to be solved using existing solution routines within the respective module. During
analysis, coupling between the flow and the structures is enforced by the transfer of loads and displace-
ments between each solution update, which occurs within each discipline. An outer iterative process
repeats this solve-then-transfer procedure until the coupled problem is converged. A block Gauss-Seidel
method is a partitioned method where the subproblems are solved sequentially, allowing the load or
displacement boundary conditions to be updated with each solution update. Alternatively, all equations
can be solved simultaneously, with loads and displacements transferred at the end of each coupled
iteration. This leads to a block Jacobi method. Similar procedures are applied to the partitioned coupled
adjoint solution, which is also referred to as a lagged-coupled adjoint method. The cross-coupling terms
in the coupled adjoint problem, which consist of adjoint variables from other disciplines, are lagged and
included on the right-hand side (RHS) or each adjoint equation. This allows the adjoint problem for
each discipline to be solved in a decoupled manner.

Aside from its low implementation cost, partitioned methods are also very effective in preserving the
modularity within the aerostructural methodology. This allows the coupled solution to take maximum
advantage of the software routines designed specifically for each subproblem [51]. With the use of under
relaxation, partitioned methods work sufficiently well for problems where the fluid-structure coupling is
weak. However, the performance of partitioned methods tends to deteriorate as the interaction between
the flow and the structures intensifies [30, 51, 52, 53, 54, 55]. Convergence becomes heavily dependent
on the choice of the relaxation parameter [30, 51], which is difficult to determine a priori. Even when
convergence can be achieved with a conservative relaxation parameter, many iterations are often neces-
sary, leading to infeasible computational times especially with the use of high-fidelity flow and structural
analysis. A number of authors have improved the efficiency of partitioned methods using information
computed on a coarse grid, as well as space or manifold mapping [56, 57, 58, 59]. Although these meth-
ods are promising, it is unclear how their performance compares with monolithic solution methods. The
superior efficiency and robustness of monolithic solution methods for solving coupled FSI problems has
been well established [30, 51, 53, 54, 55, 52, 60], as will be shown in the literature summarized next.
1.4. Robust and Efficient Solution to Coupled Problems

1.4.2 Monolithic Methods

It is important to first clarify the meaning of the term *monolithic* in the present context. In this thesis, monolithic solution methods refer to iterative methods that fully couple variables across all disciplines. In other words, they do not iterate between discipline solutions, unlike partitioned methods. They can be applied to the nonlinear analysis problem and the linear system in the discrete coupled adjoint formulation. Some literature on fluid-structure interaction (FSI) have also used the phrase *monolithic* to refer to the development of new methodologies specifically for the purpose of solving coupled FSI problems, starting from the governing partial differential equations [55, 61, 62, 63, 64, 65, 38]. This is in contrast to a *modular* approach which can be constructed from software modules that are independently developed for each discipline. Modularity can be easily achieved using partitioned solution techniques, but modularity does not imply the use of partitioned methods. Many fully coupled methods have been proposed which allow for the use of existing solvers, so that they are monolithic and modular [30, 53, 54, 55, 56, 57, 61, 62, 63, 64] at the same time. It is necessary to make this distinction because having modularity can be important due to the complexity of the physics and geometries involved, as well as the use of high-fidelity analysis tools.

Having made the clarification, monolithic solution techniques applied to aerostructural optimization are reviewed as well as popular methodologies in the general field of steady and unsteady FSI analysis. The literature review is organized into three sections. The first two sections include monolithic methods that operate on the full-size coupled problem involving all unknowns of interest. The last group of methods are based on reducing the size of the coupled problem to the set of unknowns at the aerostructural interface where the coupling occurs.

**Monolithic Methods Applied to the Unmodified Coupled Problem**

Steady-state analysis and unsteady simulations using an implicit time marching both require the solution to a system of nonlinear equations, which can be found using the Newton method. The resulting linear system is then solved by an iterative solution strategy that fully couples all unknowns at every iteration, such as a Krylov subspace method. This is the basis of the methodologies summarized below.

Heil [68] and Heil et al. [53] used a preconditioned Generalized Minimal Residual method (GMRES) to solve both steady and unsteady FSI problems. The authors emphasized the importance of an effective preconditioner. They proposed preconditioning techniques that take advantage of the block structure in the coupled linear system, which arises from partitioning the equations and variables according to the flow and structural subproblems. This in turn allows existing linear solvers specialized for each subproblem to be reused [53]. Heil [68] showed that a block Gauss-Seidel preconditioner, which uses the upper and lower block-triangular portions of the coupled Jacobian matrix, was more effective than a block Jacobi preconditioner, which uses the diagonal blocks only. The authors have also demonstrated improvements via scaling of the linear system. For both steady and unsteady problems, the proposed monolithic method was effective in strongly coupled FSI simulations when partitioned methods failed to converge. However, it remained competitive with partitioned methods even when the coupling was weak [53].

The simulations considered by Heil et al. [53, 68] were carried out in serial, while other authors have examined the application of monolithic methods in parallel. Barker and Cai [69] considered the use of a restricted additive Schwarz preconditioner for the Newton-Krylov solution algorithm. More specifically,
it was applied to two-dimensional unsteady calculations with an implicit time-marching method. Wu and Cai [55] later extended this methodology for three-dimensional problems. Their approach was nonmodular in that the flow, mesh, and structure equations were discretized simultaneously using the finite-element method. The domain decomposition was carried out independently of the type of variables at a given mesh node, so that a subdomain may contain elements from the fluid, structure, or mesh problem. The exact Jacobian at each Newton iteration was hand-differentiated. The local preconditioner on each processor was obtained by an incomplete point lower upper (LU) factorization. The proposed methodology was shown to be effective in parallel blood flow simulations with thousands of processors, including problems where partitioned methods were known to have convergence issues [55]. The authors later showed that the effectiveness of the Schwarz preconditioner may be improved by the use of a coarse grid correction [70]. The coarse grid calculation was further used to provide an appropriate initial guess for the Newton method.

Crosetto et al. [67] proposed a parallel monolithic solution strategy using the algebraic additive Schwarz preconditioner. In this approach, GMRES was preconditioned by a block Gauss-Seidel preconditioner analogous to what Heil et al. [53, 68] proposed. The calculation was parallelized by replacing the exact inverses of the fluid and structural blocks with an additive Schwarz preconditioner and local LU factorization on each processor. This was a modular approach as it allowed specialized solution techniques to be used for each of the fluid and structural blocks. The modular approach was compared to a nonmodular approach where the additive Schwarz preconditioner was applied to the Jacobian matrix in a way that disregarded the block structure arising from the distinct subproblems. The modular approach was shown to be more effective both in factorization time and the resulting number of GMRES iterations, as the number of processors was increased.

In the context of aerostructural optimization using mid-fidelity analysis, Elham et al. [71] used the Newton method for analysis, coupling a quasi-3D flow solver to a finite-element beam model that uses a quasi-analytical method for the structural sizing and weight estimation of the wing. Kennedy and Martins [72, 73] presented a parallel Newton-Krylov method for fully coupled aerostructural analysis. It was a modular approach where a high-fidelity linear structural analysis code was coupled to a medium-fidelity panel method for aerodynamic analysis. The coupled linear system at each Newton iteration was solved via a flexible variant of GMRES (FGMRES). It was preconditioned by a block Jacobi preconditioner that reused the distributed linear solvers in the flow and structural modules. Due to the linear nature of the panel method, the matrix-vector product evaluation via finite-differences rendered the monolithic analysis less competitive in comparison to partitioned methods. The authors instead assembled and stored in memory an approximate Jacobian matrix of the coupled problem, which was obtained by omitting the contribution of the follower forces in the structural Jacobian matrix. In addition, the shape derivative of the flow residual with respect to the flow grid was approximated and a lagged update was used. The authors further proposed using FGMRES to solve the coupled adjoint problem for gradient calculations. The coupled adjoint solution also used a block Jacobi preconditioner, but the transposed Jacobian matrix-vector product was computed exactly to ensure gradient accuracy. Both analysis and coupled adjoint solution demonstrated good parallel scalability up to 96 processors.

The monolithic solution strategy by Kennedy and Martins [72, 73] was applied to high-fidelity aerostructural optimization by Kenway et al. [30]. The same structural analysis was coupled with three-dimensional flow analysis based on the Euler equations. The matrix-vector product evaluation during nonlinear analysis was performed using finite-difference approximations. Due to the use of a two-field
formulation, each residual evaluation involves a flow grid deformation calculation. An appropriate initial guess for the Newton-Krylov procedure was found by applying a number of block Gauss-Seidel iterations with fixed relaxation. A block Jacobi preconditioner similar to that of Kennedy and Martins was used for the FGMRES solution of both the linearized analysis and the adjoint problem. Due to the size of the flow problem, the Krylov subspace size used to obtain the flow block preconditioner was constrained to reduce memory usage. However, the authors showed that the proposed monolithic solution method performed best when the Krylov subspace size for the flow problem was sufficient to ensure the effectiveness of the flow block preconditioner. For the analysis performed at a 1g and a 2.5g load condition, the Newton-Krylov approach was 9% and 25%, respectively, more efficient than the nonlinear block Gauss-Seidel method with Aitken acceleration. Savings of 19% and 29% were reported using the monolithic adjoint calculation for the 1g and 2.5g load conditions, respectively, relative to the linear block Gauss-Seidel method with the optimal relaxation parameter. The aforementioned studies were performed using a total of 72 processors.

Potential Improvements to Modular Block-based Preconditioners

The authors of numerous other papers have acknowledged the advantages of using modular preconditioning techniques based on the block structure of the coupled system. Although reusing specialized solution routines for the flow and structural problems during the preconditioning process is very effective in resolving the coupling within each subproblem, it is much less so in accounting for the fluid-structure coupling. As a result, many Krylov iterations may still be required when solving strongly coupled problems, in which case these methods are prone to the same drawback as partitioned methods. Therefore, performance of the monolithic methods may be improved by the use of more advanced preconditioning options.

To address these shortcomings of modular block-based preconditioners, Badia et al. proposed an alternative based on an inexact block LU factorization of the coupled Jacobian matrix, which required the inverse of one of the fluid blocks related to velocity. A cheaper approximation to the inverse was suggested, and it was used to compute the corresponding Schur complement matrix in the block lower factor. For the serial applications considered by the authors, GMRES with the proposed preconditioner was much more efficient than the partitioned preconditioner in solving strongly coupled problems. However, the use of such preconditioner requires the linear system solution to a new Schur complement problem, which in turn needs its own preconditioner. A good preconditioner for the Schur complement matrix becomes more crucial with the increase in problem size, but it remains nontrivial how it can be accomplished in a generalizable way. Another alternative by Badia et al. involved reformulating the block preconditioner based on the Robin transmission conditions. An appropriate coupling parameter needs to be specified for each of the flow and structural problems. The performance of the proposed method can be sensitive to the choice of coupling parameters, but it is unclear what the optimal values should be for the problems of interest to this thesis.

Gee et al. described a preconditioning technique based on algebraic multigrid (AMG). They first discussed the use of algebraic multigrid as the preconditioner of each field block within a block Gauss-Seidel preconditioner. The authors then proposed a new AMG procedure applicable to the full Jacobian matrix of the coupled problem, using the prolongation and restriction operators designed for each subproblem. The block Gauss-Seidel procedure was used as a smoother. The new AMG preconditioner also supports software modularity and can be applied in parallel. It has been shown to
outperform the block Gauss-Seidel preconditioner with local AMG in solving challenging FSI problems. However, the authors noted that the conclusion could be reversed if the subproblems differ substantially in size.

The improvements demonstrated by the above methodologies [54, 66, 74] have confirmed that preconditioners based on the block structure of the coupled Jacobian matrix certainly do have limitations despite their many advantages. However, it is important to point out that the applications considered by these authors are mostly in the area of unsteady haemodynamics, where the fluid-structure coupling can be much stronger than in an aerospace application. The limitations of block-based preconditioners may have less significance during steady-state aerostructural simulations in the context of aircraft design, such that a further increase in the complexity of the coupled solution strategy is not necessarily justified.

**Monolithic Methods Based on an Interface Formulation**

Another popular class of methods in the analysis of FSI problems is based on reformulating the nonlinear coupled problem into an interface problem. The interface problem has the following form [51]:

\[ R_{\Gamma}(x_{\Gamma}) = x - T(x_{\Gamma}), \]  

where \( x_{\Gamma} \) is an interface variable such as the interface position or pressure, and \( T(x_{\Gamma}) \) is a nonlinear mapping that is equivalent to the application of one block Gauss-Seidel iteration. For a three-field formulation, \( T(x_{\Gamma}) \) includes a mesh movement \((M)\), a fluid analysis \((F)\), and a structural analysis \((S)\). Each of the three subproblems can be treated as a function or mapping that operates on the solution to the preceding subproblem. Using the function composition notation, where \( f \circ g(x) \) is equivalent to \( f(g(x)) \) for some functions \( f(x) \) and \( g(x) \), \( T(x_{\Gamma}) \) can be defined as

\[ T(x_{\Gamma}) = S \circ F \circ M(x_{\Gamma}). \]  

By viewing a block Gauss-Seidel iteration as a composite function, it becomes evident the solution to \( R_{\Gamma} = 0 \) can be obtained using the Newton-Krylov algorithm. The nonlinear interface residual, \( R_{\Gamma} \), represents the changes in the interface variable between two successive block Gauss-Seidel iterations. When the coupled problem is converged, \( R_{\Gamma} \) identically vanishes. The interface formulation is much smaller in size than the original coupled nonlinear problem, which in turn reduces the memory requirement for the Krylov subspace method. However, the interface residual is more difficult to compute than the full aerostructural residual. An accurate evaluation of \( R_{\Gamma} \) further requires each subproblem to be converged to a tighter tolerance than in a typical block Gauss-Seidel iteration [51]. Hence the use of matrix-vector finite-difference during Krylov iterations can render the interface method more expensive than partitioned methods [75, 76, 77], despite the increased robustness. The Jacobian matrix-vector product with a given vector, \( z \), for the interface problem may be computed using the chain rule [75, 78]:

\[ \frac{\partial R_{\Gamma}}{\partial x_{\Gamma}} z = z - S' \left( \frac{\partial S}{\partial F \circ M(x_{\Gamma})} \right) \frac{\partial F}{\partial M(x_{\Gamma})} \frac{\partial M}{\partial x_{\Gamma}} z, \]  

where \( S' \), \( F' \), and \( M' \) represent the structural, fluid, and mesh tangent matrices respectively. Multiplication with each tangent matrix requires a linear system solution to the corresponding subproblem. Gerbeau and Vidrascu [75] proposed to replace the more expensive fluid tangent matrix with one from
a simplified fluid model to reduce the computational cost. The proposed method performed well comparing to a partitioned method with Aitken acceleration. In another paper, however, Fernandez and Moubachir [78] showed that the use of such approximate Jacobian may compromise the robustness of the solution method. They instead proposed using the exact Jacobian of the interface problem. Although more expensive than the approximate approach, it was shown to improve the nonlinear convergence and overall robustness.

Barcelos et al. [51] proposed a Schur-Newton-Krylov (SNK) method for steady-state aerostructural analysis and optimization. It is an interface approach similar to that of Fernandez and Moubachir [78] in that all derivatives are computed exactly. The authors remarked that the linearized interface problem is in fact a Schur complement of the original coupled problem; hence the interface Jacobian can be written in terms of the blocks in the Jacobian matrix of the original coupled problem. The linearized interface problem was solved by GMRES without preconditioning. Due to the use of a three-field formulation, each matrix-vector product evaluation with the interface Jacobian matrix requires a linear system solution for each of the flow, mesh, and structural subproblems. The SNK algorithm was also used to compute the gradient via a direct method. The efficiency and robustness of the SNK method was demonstrated via the analysis and sensitivity calculation on a stiff and a flexible wing configuration using 8 processors. For the stiff configuration, the SNK approach required almost twice as much processing time as the block Gauss-Seidel approach with Aitken acceleration, even though a smaller number of nonlinear iterations were required. The authors related the inefficiency to the expensive interface residual evaluation. On the other hand, the sensitivity calculation was substantially faster than the linear block Gauss-Seidel approach by more than 30%. For the flexible configuration, the SNK algorithm outperformed the partitioned algorithm for both analysis and sensitivity calculations, while the successful convergence of the partitioned approach became much more dependent on the specified relaxation parameter.

A number of methods have also been proposed based on approximating the interface Jacobian matrix using the interface residual vectors from successive iterations. An example of such methods includes the interface Newton-Krylov method of Michler et al. [79]. A linear approximation to the interface Jacobian matrix is made, where the matrix-vector products during GMRES are evaluated from the differences between interface residual vectors [80]. The quasi-Newton method of Degroote et al. [60, 81] involves collecting a set of vectors representing the changes in the interface positions, and another set of vectors for the corresponding changes in the interface residual. These are then used to predict the new interface position using a least-squares method. This procedure can also be likened to constructing an approximation to the interface Jacobian matrix [82]. The authors further illustrated the similarity between the least-squares quasi-Newton method and the popular Aitken acceleration technique, which improves the convergence of partitioned methods based on information from the two most recent iterates. The quasi-Newton method can therefore be interpreted as an augmented version of the Aitken acceleration that is able to retain additional information from earlier iterations. The interface Newton-Krylov method and the least-squares quasi-Newton method are modular and relatively easy to implement over existing block Gauss-Seidel solution procedures. Both methods improve upon the partitioned iterations with Aitken acceleration, while the quasi-Newton method was shown to have the best performance consistently [81]. However, the quasi-Newton method was still not competitive in comparison to a monolithic solution strategy [80]. In a recent paper, Bogaers et al. [82] suggested the use of Broyden’s method to construct a successively improved approximation to the interface Jacobian matrix.

In summary, the interface approach is advantageous in that it leads to a smaller problem size than
applying a monolithic solution strategy to the original coupled problem. It offers robustness improve-
ments over partitioned methods. Yet it can be relatively easy to implement on top of existing partitioned
solution routines via the use of finite-difference approximations or a quasi-Newton approach. However,
the use of approximations reduces the effectiveness of the interface approach relative to solving the un-
modified coupled problem monolithically. At the same time, the need to accurately evaluate the interface
residual $R_{\Gamma}$ can overshadow the increase in robustness with an increase in computational cost. Since
robustness and efficiency are equally important to this thesis in constructing an exploratory optimization
methodology, a monolithic solution algorithm similar to the first group of methods examined in this
section appears to be more appropriate.

1.5 Objectives and Thesis Outline

This work aims to provide researchers with a useful tool for the discovery of novel and efficient aircraft
concepts. Hence the overriding objective of this thesis involves the design of an effective optimization
methodology based on high-fidelity steady-state aerostructural analysis, that is equipped with the flex-
ibility, efficiency, and robustness to enable exploratory design. Within the context of this overall goal,
there are two specific objectives to be achieved in the present thesis.

The first objective involves the extension of the integrated geometry parameterization and mesh
movement of Hicken and Zingg [44] to fully coupled high-fidelity aerostructural optimization. The inte-
grated approach presents an effective way to enable large shape changes during exploratory optimization.
The ability to handle shape changes is not only useful during optimization, it is also necessary for large
structural deflections during aerostructural analysis. However, two additional challenges must be ad-
ressed in extending the integrated approach to aerostructural optimization. A new algorithm needs to
be introduced to deform the internal structures consistently with the B-spline geometry parameteriza-
tion. It must also be shown that the B-spline mesh movement presents a feasible alternative to existing
methodologies in accounting for shape changes due to structural deflections, in addition to those due to
optimization.

The second objective involves the development and investigation of a monolithic solution algorithm
for the coupled analysis and adjoint problems. Monolithic solution methods are potentially more efficient
and robust than their partitioned counterparts. They are hence advantageous for effective exploratory
optimization based on high-fidelity aerostructural calculations. In the context of the present thesis, a
modular approach will be adopted in developing the monolithic solution strategy, as the high-fidelity flow
and structural analysis capabilities are constructed over existing software modules. The effectiveness of
monolithic solution methods for general FSI problems has been demonstrated by a number of authors,
which focused largely on analysis only. However, a much smaller pool of studies are available for large-
scale exploratory and high-fidelity aerostructural applications which include sensitivity calculations.
There is likely still room for further improvements on top of what has been reported in these few studies.
Moreover, it is important to determine how different aspects of the monolithic solution algorithm should
be designed for the present methodology such that its efficiency is maximized. This is a subject of
investigations in this thesis.

In light of the above discussions, the main objectives of the present thesis are summarized below:

- Extend the integrated geometry parameterization and mesh movement technique to aerostructural
  optimization:
1.5. Objectives and Thesis Outline

– Devise a method to deform the mesh of the detailed structural model consistently with the B-spline parameterization of the aerodynamic geometry in response to large shape changes due to optimization.

– Couple the linear elasticity mesh movement algorithm in the integrated approach with the high-fidelity aerodynamic and structural analysis using a three-field formulation of the steady-state aerostructural analysis problem.

– Formulate an augmented three-field coupled adjoint problem for accurate gradient calculations.

• Design and develop effective monolithic solution strategies for the coupled analysis and adjoint problems:
  – Implement a Newton-Krylov solution method for the nonlinear analysis problem.
  – Investigate and identify the appropriate scaling, matrix-vector product evaluation, and preconditioning strategies for optimal performance of the solution algorithm.
  – Perform similar investigations for the development of a monolithic method for the coupled adjoint problem using a Krylov subspace iterative scheme.

This thesis involves many levels of intricacy. Although the various aspects of the flow, mesh, and structural modules are not direct contributions of the present work, they must still be discussed and well understood in order to ensure the accuracy and efficiency of the coupled methodology. Equally important are the details of how the coupled methodology itself is constructed. Therefore, this thesis report is organized by first introducing the individual building blocks of the aerostructural optimization framework, where all necessary details relevant to the specific components of the methodology are presented. This is the main purpose of Chapters 2, 3, and 4. Chapter 5 then focuses on the construction of the new aerostructural optimization methodology from software components described in the earlier chapters. Chapter 6 investigates the design of an efficient monolithic solution strategy for the analysis and the coupled adjoint problems. Finally, Chapter 7 includes some results and applications for demonstrating the effectiveness of the monolithic solution methods and the capability of the present aerostructural optimization methodology.
Chapter 2

Aerostructural Optimization

Problem Overview

Before presenting the details of the aerostructural optimization methodology, it is necessary to first define the optimization problem. An optimization problem minimizes an objective function, $J$, with respect to a set of design variables, $v$. The aerostructural optimization is subject to a partial differential equation (PDE) constraint given by the discrete steady aerostructural equations, $R_{AS}$. The optimization may also be subject to a number of equality constraints, $C_{eq}$, and inequality constraints, $C_{in}$, which may be linear or nonlinear. The optimization problem can be summarized as follows:

$$
\begin{align*}
\min_v & \quad J(v, [q, b_\Delta, u]^T), \\
\text{subject to:} & \quad R_{AS}(v, [q, b_\Delta, u]^T) = 0, \\
& \quad C_{eq,i}(v, [q, b_\Delta, u]^T) = 0, \quad i = 1, \ldots, n_{eq}, \\
& \quad l_j \leq C_{in,j}(v, [q, b_\Delta, u]^T) \leq u_j, \quad j = 1, \ldots, n_{in}
\end{align*}
$$

The lower and upper bounds for the $j^{th}$ inequality constraint are given by $l_j$ and $u_j$, respectively. The aerostructural state variables are given by $[q, b_\Delta, u]^T$, where $q$ is the flow state, $b_\Delta$ is the state of the deformed aerodynamic grid with structural deflections, and $u$ is the structural state. By solving $R_{AS}(v, [q, b_\Delta, u]^T) = 0$, the state variables become a function of $v$. Aerostructural analysis is discussed in Chapter 5. Details regarding the equations governing the flow, structure, and mesh subproblems are found in Chapters 3 and 5.

Aerostructural analysis allows multidisciplinary objectives, including both aerodynamic and structural functionals, to be evaluated and minimized. Examples of objective functions include range, fuel burn, or some linear combination of weight and drag for more specific tradeoff studies. A lift constraint specified in terms of the weight of the aircraft is often necessary. To avoid structural failure, a Kreisselmeier-Steinhauser (KS) aggregation function [83] is used to ensure that the von Mises stress values of all structural elements are below the yield stress of the material by a specified factor of safety. Constraints on the geometry may also be included. The design variables are categorized into geometric (i.e. $v_G$), aerodynamic (i.e. $v_A$), and structural (i.e. $v_S$) design variables, which specify the geometry, flow conditions, and structural properties, respectively, for the analysis. For this work, aerodynamic design variables consist of an angle of attack for each load condition. Structural design variables specify the
thickness of individual structural members. To capture the tradeoff between weight and drag, the main load bearing components of a wingbox are modeled, including spars, ribs, top skin, and bottom skin. Geometric design variables depend on the choice of geometry parameterization and control techniques. The present thesis proposes the use of B-spline surfaces for geometry parameterization and control, the relevant details of which will be discussed in Chapter 4.

The optimization problem in (2.1) is solved using SNOPT \[84\], which is a gradient-based sequential quadratic programming (SQP) algorithm that is well-suited for large-scale, nonlinear optimization problems. Solution iterates to the optimization are updated by solving a QP subproblem that minimizes the quadratic approximation to the Lagrangian function subject to linearized constraints. Linear constraints are specified in terms of a sparse constraint Jacobian matrix at the start of an optimization, and they are satisfied at each optimization or design iteration. SNOPT requires the values of the objective and all nonlinear constraints, as well as their gradient values with respect to the design variables at each design iteration. The PDE constraint, given by \( R_{\text{AS}} = 0 \), is not treated by SNOPT. Each design iteration hence includes an analysis that solves the discrete steady aerostructural equations, after which the appropriate functional and gradient values can be evaluated. The Hessian of the QP subproblem is updated using a quasi-Newton approximation. In theory, the optimization is considered converged upon satisfying the first-order Karush-Kuhn-Tucker optimality condition to within a user-specified tolerance. However, it is often difficult to satisfy this criterion for large-scale nonlinear design problems with many design variables, such as ones of interest to this thesis. Therefore, the optimization problems considered in this thesis are considered converged when a feasibility measure is reduced to below \( 10^{-6} \), which means all nonlinear constraints have been satisfied to this tolerance and that the changes in the augmented Lagrangian merit function are sufficiently small. Upon satisfaction of all nonlinear constraints, the value of the merit function approaches that of the objective function.

The software architecture for aerostructural optimization is programmed in Python, which allows existing flow and structural modules written in different programming languages to be used as building blocks. A third-party Python interface to SNOPT \[85\] is used for this work. However, modifications to the Python interface are necessary for the specification of linear and nonlinear constraints with a sparse Jacobian. These changes allow SNOPT to handle the geometric constraints for the B-spline geometry parameterization much more effectively.

Finally, unsteady aeroelastic phenomena such as flutter, and structural failure modes such as buckling, are not currently considered. The purpose of this work is to explore methodologies which can effectively capture the important tradeoffs between weight and drag, while enabling substantial geometric variation. Modelling static aerostructural coupling is sufficient for this purpose, and it will serve as an important step towards incorporating these additional constraints and unsteady effects in the future.
Chapter 3

Aerodynamics and Structural Modules

High-fidelity aerostructural analysis in the present framework is realized by coupling existing aerodynamics and structural modules, each equipped with its own gradient calculation capabilities for purely aerodynamic or structural optimization. Routines for linear and nonlinear system solutions, as well as gradient calculations will be reused in constructing the aerostructural optimization methodology. The relevant components in each software module are summarized in this chapter.

3.1 Aerodynamics Module

The aerodynamics module is capable of analysis based on both the three-dimensional Euler equations governing inviscid flow \[86\] and the Reynolds-Averaged Navier-Stokes (RANS) equations governing turbulent flow \[87\]. This section will focus on the inviscid part of the flow solver, because the present aerostructural optimization framework is primarily coded for calculations involving the Euler equations. RANS aerostructural analysis has been implemented for validation with experimental data, on which additional details will be provided later. An extensive description of the viscous and turbulent flow analysis capabilities of the flow solver can be found in Osusky and Zingg \[87\].

3.1.1 Transformed Three-Dimensional Euler Equations

The flow analysis module uses a finite-difference discretization applied to a body-fitted multi-block structured grid placed around the geometry of interest. To simplify the application of finite-difference operators, a generalized curvilinear coordinate transformation is applied to the governing PDE. This transformation maps the coordinates \((x, y, z)\) in physical space to parametric coordinates \((\xi, \eta, \zeta)\) in computational space, such that

\[
\begin{align*}
\xi &= \xi(x, y, z), \\
\eta &= \eta(x, y, z), \\
\zeta &= \zeta(x, y, z).
\end{align*}
\]
The grid in computational space is assumed to have unit spacing, or \( \Delta \xi = \Delta \eta = \Delta \zeta = 1 \). The transformed Euler equations have the following form [88]:

\[
\frac{\partial \hat{q}}{\partial t} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{G}}{\partial \zeta} = 0, \tag{3.2}
\]

where

\[
\hat{q} = \frac{1}{J} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \hat{E} = \frac{1}{J} \begin{bmatrix} \rho U \\ \rho u U + p \frac{\partial \xi}{\partial x} \\ \rho v U + p \frac{\partial \xi}{\partial y} \\ (e + p) U \end{bmatrix}, \quad \hat{F} = \frac{1}{J} \begin{bmatrix} \rho V \\ \rho u V + p \frac{\partial \eta}{\partial x} \\ \rho v V + p \frac{\partial \eta}{\partial y} \\ (e + p) V \end{bmatrix}, \quad \hat{G} = \frac{1}{J} \begin{bmatrix} \rho W \\ \rho u W + p \frac{\partial \zeta}{\partial x} \\ \rho v W + p \frac{\partial \zeta}{\partial y} \\ (e + p) W \end{bmatrix}. \tag{3.3}
\]

The overhead symbol "\(^\wedge\)" denotes the transformed conservative state variables and fluxes, where \( J \) is the Jacobian of the transformation. The state variables \( q \), which are simply \( J\hat{q} \), include the density, \( \rho \), the momentum per unit volume in each Cartesian coordinate direction, \( \rho u, \rho v, \) and \( \rho w \), as well as the energy per unit volume, \( e \). All state variables have been nondimensionalized as described in [88]. The pressure \( p \) is given by

\[
p = (\gamma - 1) \left[ e - \frac{1}{\rho} (u^2 + v^2 + w^2) \right], \tag{3.4}
\]

where \( \gamma = 1.4 \) is the specific heat ratio for air, and \( u, v, w \) are the velocity components in each Cartesian coordinate direction. The contravariant velocities are given by

\[
\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{3.5}
\]

The grid metrics in (3.3) and (3.5) are written in terms of the partial derivatives of the Cartesian coordinates with respect to the parameteric coordinates:

\[
\begin{align*}
\frac{\partial \xi}{\partial x} &= J \left( \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta} \right), \\
\frac{\partial \eta}{\partial x} &= J \left( \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \xi} \right), \\
\frac{\partial \zeta}{\partial x} &= J \left( \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} \right), \\
\frac{\partial \xi}{\partial y} &= J \left( \frac{\partial x}{\partial \eta} \frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial z}{\partial \eta} \right), \\
\frac{\partial \eta}{\partial y} &= J \left( \frac{\partial x}{\partial \xi} \frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial z}{\partial \xi} \right), \\
\frac{\partial \zeta}{\partial y} &= J \left( \frac{\partial x}{\partial \eta} \frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial z}{\partial \eta} \right), \\
\frac{\partial \xi}{\partial z} &= J \left( \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \eta} \right), \\
\frac{\partial \eta}{\partial z} &= J \left( \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \xi} \right), \\
\frac{\partial \zeta}{\partial z} &= J \left( \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \right).
\end{align*} \tag{3.6}
\]

Finally, the metric Jacobian is computed by

\[
J^{-1} = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \zeta} + \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta} + \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta} + \frac{\partial x}{\partial \zeta} \frac{\partial y}{\partial \zeta} \frac{\partial z}{\partial \eta}. \tag{3.7}
\]
3.1.2 Discretization and Solution to the Nonlinear Problem

The spatial derivatives in (3.2) and those used for the grid metric calculations are approximated using second-order summation-by-parts (SBP) finite-difference operators. The use of simultaneous approximation terms with the SBP operators simplifies interface solution coupling and boundary treatments, by maintaining solution accuracy and time stability without requiring grid continuity at block interfaces and boundaries [86]. Furthermore, the grid metric terms are computed using an averaging procedure of the finite-difference approximations in order to satisfy the metric invariants [89]. Applying the spatial discretization leads to the following semi-discrete form of (3.2):

\[
\frac{d\hat{q}}{dt} + R_A(q, G) = 0,
\]  

(3.8)

where \( \hat{q} \) and \( q \) are the solution vectors with and without the curvilinear transformation, respectively, that contain the discrete conservative solution state at all nodes in the flow domain. The flow residual vector, \( R_A \), holds a discrete approximation to the derivative of the inviscid flux vector at every node. The flow residual vector is not only a function of \( q \), but also a function of \( G \), which represents the coordinates of the flow grid nodes in physical space. This is to remind the reader of the presence of the metric terms in the flow equation. Although \( G \) is not part of the PDE solution in a pure aerodynamic analysis, it will become a variable during coupled aerostructural calculations. Artificial dissipation is added to \( R_A \) to prevent spurious oscillations in the numerical solution of nonlinear hyperbolic PDEs such as the Euler equations. A scalar dissipation model based on the work of Jameson [90] and Pulliam [91] is used, as detailed in [88]. Both \( q \) and \( R_A \) are of size \( n_A = 5 n_G \), where \( n_G \) is the number of flow grid nodes. The steady-state solution to (3.8) is sought by solving

\[
R_A(q, G) = 0.
\]  

(3.9)

The aerodynamics module is equipped with a sophisticated and efficient parallel implicit Newton-Krylov-Schur algorithm to solve (3.9) for \( q \), assuming \( G \) is given. This solution procedure is reused during aerostructural analysis. Therefore, key features of this algorithm are highlighted in the subsequent paragraphs.

Equation (3.9) is a nonlinear system of equations which can be solved by an inexact Newton method. However, convergence of the Newton method is only guaranteed if the initial iterate is sufficiently close to the solution of (3.9). This is addressed by the use of a continuation strategy, and numerous options are available in the flow solver [92]. The default pseudo-transient methodology proposed by Hicken and Zingg [86] is used, which introduced an approximate-Newton phase in the solution procedure, preceding the inexact-Newton phase. During the approximate Newton phase, the solution at the \( n^{th} \) iteration is updated by solving the following linear system for \( \Delta q^{(n)} \):

\[
(T^{[m/n/m]} + A_1^{[m/n/m]}) \Delta q^{(n)} = -R_A^{(n)}.
\]  

(3.10)

An expression similar to (3.10) can result from applying the implicit Euler time marching method to (3.8). However, time accuracy of the implicit Euler scheme is not required here, making it possible to accelerate the start-up process via a number of modifications. The full flow Jacobian, \( \partial R_A / \partial q \), is replaced in (3.10) by a first-order approximation denoted by \( A_1 \), which has a smaller stencil size than \( \partial R_A / \partial q \). The rate of convergence is further improved by the use of a spatially varying time-step, which
is a function of the metric Jacobian at each node and a reference time step that increases by a specified factor at successive iterations. The inverse of the local time step for each row of (3.10) is included in the diagonal matrix, $T$. Finally, (3.10) is solved using a Krylov subspace method, where the left-hand side (LHS) is factored for the preconditioning procedure. Both $A_1$ and $T$ are only recomputed once every $m$ iterations to reduce the matrix factorization cost. This is reflected in the superscript $(m \lfloor n/m \rfloor)$, where $\lfloor \cdot \rfloor$ is the floor operator that returns the largest integer value less than or equal to $n/m$.

Once the nonlinear residual norm, $\| R_A^{(n)} \|_2$, drops below a specified tolerance relative to its initial value, $\| R_A^{(0)} \|_2$, the algorithm switches into the inexact-Newton phase. The linear system solved during each inexact-Newton iteration has a similar form as (3.10):

$$
(T^{(n)} + A^{(n)}) \Delta q^{(n)} = - R_A^{(n)}.
$$

Instead of the first-order approximation $A_1$, a more accurate flow Jacobian, $A$, is used in (3.11). Only the matrix-vector product with $A$ is required during the iterative solution to (3.11). Therefore, it is evaluated using a first-order forward difference approximation without explicitly storing $A$. The preconditioner used for the linear system solution of (3.10) is based on $A_1$, which is assembled and factored at each nonlinear iteration. The diagonal matrix, $T$, of the inverse of the local time steps in (3.11) acts as a successive relaxation parameter that allows for a smooth transition from the approximate to the inexact-Newton phase. For this reason, the reference step used in calculating $T$ starts with its value at the end of the approximate-Newton phase, and tends to zero quadratically with $\| R_A \|_2$. As $T$ vanishes, the inexact Newton method in its conventional form is recovered from (3.11). More details regarding the nonlinear solution process as well as the selection of the solver parameters can be found in [89].

### 3.1.3 Distributed Solution to the Linearized Flow Problem

Equations (3.10) and (3.11) are solved in a distributed manner using FGMRES. The iterative solution is preconditioned using an approximate Schur complement of the linear system [86]. Not only is this distributed solution strategy an essential component of the Newton-Krylov-Schur algorithm for the solution to (3.9), it will also be part of the monolithic aerostructural analysis in Chapter 6. The flow adjoint problem, which will be discussed in Section 3.1.5, is preconditioned in a similar manner. Therefore, this section reviews the distributed data structure involved and the preconditioning procedure.

A distributed data structure is used in the aerodynamics module, such that each processor is responsible for storing and evaluating the variables and equations belonging to one or more blocks on the multi-block structured grid. The unknowns assigned to each processor are categorized into:

- Internal unknowns, which includes those only coupled to other unknowns on the same processor $i$;

- Internal interface unknowns, which include those stored on processor $i$ that are coupled to unknowns from a different processor $j \neq i$;

- External interface unknowns, which include those stored on a different processor $j \neq i$ that are coupled to local (interface) unknowns belonging to processor $i$.

By ordering the internal unknowns on processor $i$ before the interface unknowns, equations assigned to
3.1. Aerodynamics Module

Processor $i$ can be partitioned in the following way:

$$
\begin{bmatrix}
B_i & F_i \\
E_i & C_i
\end{bmatrix}
\begin{bmatrix}
y_{i,\text{local}} \\
y_{i,\text{interface}}
\end{bmatrix} +
\begin{bmatrix}
0 \\
\sum_{j=1}^{S} E_{ij} y_{j,\text{interface}}
\end{bmatrix} =
\begin{bmatrix}
g_{i,\text{local}} \\
g_{i,\text{interface}}
\end{bmatrix}.
\tag{3.12}
$$

The LHS of (3.12) corresponds to the local contribution to $A_1$ from processor $i$. The interior unknowns are denoted by $y_{i,\text{local}}$, the local interface unknowns by $y_{i,\text{interface}}$, and the external interface unknowns by $y_{j,\text{interface}}$. The total number of subdomains is given by $S$, and $E_{ij}$ is only nonzero if processor $j$ has one or more neighbours to the blocks on processor $i$. Communication with other processors is only required for the summation, $\sum_{j=1}^{S} E_{ij} y_{j,\text{interface}}$, involving the external interface unknowns.

The Schur complement of the local system of equations in (3.12) is found by first writing the internal unknowns in terms of the local interface unknowns:

$$
y_{i,\text{local}} = B_i^{-1} (g_{i,\text{local}} - F_i y_{i,\text{interface}}),
\tag{3.13}
$$

which is then substituted into the equations for the local interface unknowns. Doing so results in the following:

$$
S_i y_{i,\text{local}} + \sum_{j \in N_i} E_{ij} y_{j,\text{interface}} = g_{i,\text{interface}} - E_i B_i^{-1} g_{i,\text{local}} = g_i', y_{i,\text{interface}},
\tag{3.14}
$$

where $S_i = C_i - E_i B_i^{-1} F_i$ is the Schur complement of the local problem in (3.12). The local Schur complement problems on all processors are gathered into a global Schur complement problem of all interface variables in the linearized flow problem as follows:

$$
\begin{bmatrix}
S_1 & E_{12} & \cdots & E_{1S} \\
E_{21} & S_2 & \cdots & E_{2S} \\
\vdots & \ddots & \ddots & \vdots \\
E_{S1} & E_{S2} & \cdots & S_S
\end{bmatrix}
\begin{bmatrix}
y_{1,\text{interface}} \\
y_{2,\text{interface}} \\
\vdots \\
y_{S,\text{interface}}
\end{bmatrix} =
\begin{bmatrix}
g_{1,\text{interface}} \\
g_{2,\text{interface}} \\
\vdots \\
g_{S,\text{interface}}
\end{bmatrix}.
\tag{3.15}
$$

Once (3.15) is solved for the interface unknowns on all processors, they can be substituted back into (3.13) to evaluate the values for the local interior unknowns on each processor.

The aerodynamics module preconditions the linearized flow problems such as (3.10) and (3.11) by seeking an approximate solution to the global Schur complement problem in the form of (3.15), obtained iteratively to a loose tolerance. The matrix-vector product evaluation and the preconditioner, as required by the iterative solution to the Schur complement problem, make use of the incomplete LU (ILU) factorization of the distributed linear system on each processor [89]. The LHS in (3.12) can be factored as follows:

$$
L_i = \begin{bmatrix}
L_{B_i} & 0 \\
E_i & L_{B_i}^{-1} L_{S_i}
\end{bmatrix},
\quad
U_i = \begin{bmatrix}
U_{B_i} & L_{B_i}^{-1} F_i \\
0 & U_{S_i}
\end{bmatrix},
\tag{3.16}
$$

where it can be shown that

$$
B_i = L_{B_i} U_{B_i},
\quad
S_i = L_{S_i} U_{S_i}.
\tag{3.17}
$$

In other words, (3.16) contains the necessary information to approximate $B_i^{-1}$ in evaluating the matrix-vector product with $S_i$. The approximate factorization of $S_i^{-1}$ can further serve as a block Jacobi preconditioner for the global Schur complement problem in (3.15). The approximate Schur preconditioner has worked well in purely aerodynamic shape optimization problems [86] [44].
3.1.4 Inviscid Surface Traction Calculation

During aerostructural analysis, it is necessary to transfer the aerodynamic forces from the flow module to the structural module. Specifically, the force transfer algorithm used for this thesis, which will be introduced in Section 3.2.3, requires a vector of surface tractions, \( \mathbf{f}_A \), evaluated at all the nodes on the discrete solid boundary.

Assuming that all discrete surface grid nodes have been organized into a one-dimensional array of Cartesian coordinates in space, then the magnitude of the inviscid surface force per unit area at the node \( i \) is given by

\[
\mathbf{f}_{A,i} = q_\infty \left[ \frac{2}{\gamma M_\infty^2} (\gamma p_i - 1) \right].
\]

(3.18)

The nodal pressure, \( p_i \), is computed from the flow state at node \( i \), and \( M_\infty \) is the freestream Mach number. Due to the use of nondimensionalized variables in the flow module, \( p_i \) is in fact

\[
p_i = \frac{P_i}{\gamma P_\infty},
\]

(3.19)

where \( P_i \) is the dimensional pressure, \( a_\infty \) and \( P_\infty \) are the dimensional freestream sound speed and pressure respectively. The structural analysis requires forces that are dimensional, hence the dynamic pressure, \( q_\infty \), is required in (3.18). The direction of the surface traction is given by the unit normal to the surface at node \( i \), so that the nodal surface traction vector is

\[
\mathbf{f}_{A,i} = \mathbf{f}_{A,i} \frac{n_i}{n_i \cdot n_i},
\]

(3.20)

where \( n_i \) is given by the appropriate grid metrics at node \( i \).

To illustrate the connection between the surface normal and the grid metric at the node \( i \), let \( \xi \) and \( \eta \) be the two parametric coordinate directions tangential to the surface on which the forces are being calculated. Let \( \zeta \) be the remaining parametric coordinate direction. The Cartesian coordinates of the surface, \( s \), can be described in terms of \( \xi \) and \( \eta \) by

\[
s(\xi, \eta) = x(\xi, \eta) \mathbf{i} + y(\xi, \eta) \mathbf{j} + z(\xi, \eta) \mathbf{k}.
\]

(3.21)

The tangent vectors to the surface are then

\[
\frac{\partial s}{\partial \xi} = \frac{\partial x}{\partial \xi} \mathbf{i} + \frac{\partial y}{\partial \xi} \mathbf{j} + \frac{\partial z}{\partial \xi} \mathbf{k},
\]

\[
\frac{\partial s}{\partial \eta} = \frac{\partial x}{\partial \eta} \mathbf{i} + \frac{\partial y}{\partial \eta} \mathbf{j} + \frac{\partial z}{\partial \eta} \mathbf{k},
\]

(3.22)

The normal to \( s \) at node \( i \) can be found by taking the cross product between the tangent vectors evaluated at the same node:

\[
\mathbf{n}_i = \frac{\partial s}{\partial \xi} \times \frac{\partial s}{\partial \eta} = \left( \frac{\partial y}{\partial \eta} \frac{\partial z}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial z}{\partial \eta} \right) \mathbf{i} + \left( \frac{\partial x}{\partial \eta} \frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial z}{\partial \eta} \right) \mathbf{j} + \left( \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} \right) \mathbf{k}
\]

(3.23)

A comparison between the above expression for \( \mathbf{n}_i \) and the grid metric calculation in Section 3.1.1 reveals that

\[
\mathbf{n}_i = J^{-1} \left( \frac{\partial \zeta}{\partial x} \mathbf{i} + \frac{\partial \zeta}{\partial y} \mathbf{j} + \frac{\partial \zeta}{\partial z} \mathbf{k} \right),
\]

(3.24)
where the partial derivative terms on the RHS are the grid metric terms from the curvilinear coordinate transformation. The RHS may further be multiplied by -1 to ensure that the surface normal is pointing outward from the block.

For aerostructural optimization, it is necessary to differentiate the surface traction with respect to the flow state and the flow grid. Both are fairly straightforward to compute in the case of inviscid flow, because the surface traction at every node only depends on the flow state and the grid metric at the same node. The partial derivative of the surface traction, \( f_{A,i} \), at node \( i \) with respect to \( q \) is

\[
\frac{\partial f_{A,i}}{\partial q_i} = \frac{2}{M_\infty^2} \frac{n_i}{\|n_i\|_2} \frac{\partial p_i}{\partial q_i}.
\] (3.25)

The surface traction also needs to be differentiated with respect to the nodal surface normal as follows:

\[
\frac{\partial f_{A,i}}{\partial n_i} = f_{A,i} \frac{\partial (n_i/\|n_i\|_2)}{\partial n_i} = f_{A,i} \left[ \frac{I}{\|n_i\|_2} - \frac{n_i n_i^T}{\|n_i\|_2^3} \right],
\] (3.26)

where \( I \in \mathbb{R}^{3\times3} \) is an identity matrix. By writing the above expression in terms of the global vector of all grid metric terms, \( \nabla \xi \), the partial derivative of the vector of all surface tractions, \( f_A \), with respect to the flow grid, \( G \), is found by applying the chain rule as follows:

\[
\frac{\partial f_A}{\partial G} = \frac{\partial f_A}{\partial \nabla \xi} \frac{\partial \nabla \xi}{\partial G}.
\] (3.27)

The specific ways in which the above terms are incorporated into the gradient calculation for aerostructural optimization will be discussed in Section 5.2.

### 3.1.5 Aerodynamic Functionals and the Flow Adjoint Equation

Apart from the flow analysis, the flow module is used to evaluate aerodynamic functionals of interest. Aerodynamic functionals, including the lift and drag coefficients, are denoted by \( J_A \) in this thesis. For inviscid problems, they are obtained by numerically integrating the pressure at each node over the discrete representation of a solid boundary. The resulting lift and drag values are therefore functions of the flow state, \( q \), as well as the flow grid, \( G \). Routines for gradient calculations during purely aerodynamic shape optimization are also useful in aerostructural optimization. Discussion of the gradients of \( J_A \) with respect to the grid and geometric design variables will be reserved for Section 4.5. The gradients with respect to aerodynamic design variables, or \( v_A \), through the dependence of \( J_A \) on \( q \), are computed using a discrete adjoint approach.

The gradients of each objective or constraint dependent on \( q \) require a linear system solution to the following flow adjoint problem:

\[
\left( \frac{\partial R_A}{\partial q} \right)^T \Psi_A = - \left( \frac{\partial J_A}{\partial q} \right)^T.
\] (3.28)

The partial derivative of \( J_A \) with respect to \( q \) is available in the flow module via analytical differentiation. The coefficient matrix in (3.28) is the transposed Jacobian of the flow residual. It is formed explicitly using a combination of analytical and complex-step differentiation [89], and is subsequently stored in memory. Equation (3.28) is solved iteratively using GCROT\((m,k)\) [93], which is a flexible variant of GCROT [94] with a simplified truncation strategy. GCROT\((m,k)\) outperforms restarted FGMRES in terms of robustness when a large number of Krylov iterations is required for convergence and the size
of the Krylov subspace is constrained by the memory available [89, 93]. The use of GCROT\((m, k)\) for the adjoint problem is motivated by the need to solve (3.28) to a tight tolerance in order to ensure the gradient accuracy. When the flow adjoint solution routine is reused as part of a coupled aerostructural solution process, the linear solution tolerance is more relaxed. However, GCROT\((m, k)\) is still chosen due to the improved robustness for a given memory requirement. Finally, the use of a flexible iterative method allows (3.28) to be preconditioned using the transposed approximate-Schur preconditioner summarized in Section 3.1.3, but a higher fill level is often adopted for the incomplete LU factorization of the local block matrices to accelerate the convergence.

Using the flow adjoint variables, \(\Psi_A\), from (3.28), the gradient with respect to the angle of attack (AOA) as a design variable is computed in the flow module by

\[
\frac{dJ_A}{dv_A}^T = \left( \frac{\partial J_A}{\partial v_A} \right)^T_q + \left( \frac{\partial R_A}{\partial v_A} \right)^T_q \Psi_A,
\]

where the product \(\frac{\partial R_A}{\partial v_A}^T \Psi_A\) is computed using complex-step differentiation. The lift and drag functionals are also explicit functions of the AOA, which defines the direction of the lift and drag forces relative to the computational grid. This means \(\frac{\partial J_A}{\partial v_A}^T \neq 0\), and it is differentiated analytically. The above partial derivative terms are also needed to obtain the gradients in aerostructural optimization, as will be discussed in Section 5.2.

### 3.2 Structural Module

The Toolkit for the Analysis of Composite Structures (TACS), which was developed by Kennedy and Martins [73, 95], is used as the structural module for the present framework. It is a sophisticated parallel finite-element code designed to perform high-fidelity structural analysis, as well as gradient-based structural and aeroelastic optimization. TACS is capable of modelling isotropic and composite materials. Implementation of various element types is also available with both linear and geometrically nonlinear analysis. The present methodology models the thin-wall structures commonly found in aerospace vehicles using second-order mixed interpolation of tensorial components (MITC) shell elements [96]. MITC shell elements are effective in avoiding shear and membrane locking, as discussed in Kennedy and Martins [95].

A linear structural analysis is used in this work. For the purpose of developing the present methodology, it is assumed throughout this thesis that the structural material is based on the 7075 Aluminum with a Poisson’s ratio of 0.33, a Young’s modulus of 70GPa, and a yield stress of 434MPa, unless otherwise specified. Composite structures are expected to play an important role in the design of next-generation aircraft. The present methodology can be extended to allow for the use of composite materials via modifications internal to the structural module. Appropriate structural design variables will have to be introduced and the changes in the material properties need to be incorporated into the structural analysis [97, 98]. However, discussions related to the geometry parameterization, mesh deformation, coupled analysis, and gradient evaluations presented later in this thesis are still applicable.

#### 3.2.1 Shell Element Formulation

This section summarizes the finite-element formulation in TACS. The nodal variables in a MITC shell element consist of the displacement and rotation variables found in a classical shell element. The dis-
placement field within the \(i\)th shell element is a function of the mid-surface displacement, \(U_0 \in \mathbb{R}^{3 \times 1}\), and the small rotation angles in the global Cartesian frame, \(\Theta \in \mathbb{R}^{3 \times 1}\), as follows:

\[
U(\xi, \zeta) = U_0(\xi) + \zeta \omega(\Theta(\xi)).
\]

The parametric coordinate, \(\xi = [\xi_1, \xi_2]\), is a set of parameteric coordinates along the mid-surface of the element, while the through-thickness parameteric coordinate is given by \(\zeta \in [-t/2, t/2]\). The thickness, \(t\), of the shell element is used as a design variable for structural sizing during optimization. The through-thickness rate of change of the displacement, \(U\), is \(\omega\), which is a function of \(\Theta\). Furthermore, the variations of \(U_0\) and \(\Theta\) through space are obtained by interpolating the nodal degrees of freedom (DOF), \(u_e^i \in \mathbb{R}^{n_e}\), of the \(i\)th element using bi-Lagrange shape functions, \(N_i(\xi) \in \mathbb{R}^{6 \times n_e}\):

\[
\begin{bmatrix}
U_0(\xi) \\
\Theta(\xi)
\end{bmatrix} = N_i(\xi) u_e^i.
\]

Each MITC shell element of degree \(p\) consists of \(p^2\) nodes, and there are 6 DOF per node. The total number of DOF in an element is hence \(n_e = 6p^2\).

An element injection operator, \(P_i \in \mathbb{R}^{n_S \times p^2}\), is defined such that the DOF belonging to the \(i\)th element can be obtained from the vector of all structural unknowns, \(u \in \mathbb{R}^{n_S}\) as follows:

\[
u_e^i = (P_i^T \otimes I_6) u,
\]

where \(\otimes\) represents the Kronecker product, and \(I_6\) is a 6 \(\times\) 6 identity matrix \([95]\). Using a similar notation, the coordinates in space, \(x_e^i \in \mathbb{R}^3\), for all nodes in the element \(i\) are given by

\[
x_e^i = (P_i^T \otimes I_3) X_S,
\]

where \(X_S \in \mathbb{R}^{3n_S}\) is a vector containing the coordinates of all nodes on the structural mesh. For the force and displacement transfer calculations in Section 3.2.3 it is necessary to describe the geometry of the element as a function of \(\xi\). To this end, the location of the midsurface is interpolated using the same shape functions from (3.31), such that \(x_e^i(\xi) = N_i(\xi) x_e^i\).

By applying the principle of virtual work to each individual element, and subsequently summing the element residual over the structural domain, the resulting global structural residual, \(R_S\), can be written as

\[
R_S = K_S u - f_S,
\]

where \(K_S\) is a linear stiffness matrix, and \(f_S\) is a vector of external forces acting on the structure. Details of the derivation are omitted here, but is available in \([95]\). The stiffness matrix, \(K_S\), depends on the geometry of the structure, or the spatial coordinates of the structural mesh; it is also a function of the thickness of the shell elements. In the present framework, the force vector \(f_S\) represents the aerodynamic forces on the wing. Calculation of \(f_S\) is discussed in Section 3.2.3.

### 3.2.2 Parallel Finite-Element Analysis

Due to the poor conditioning in the stiffness matrix resulting from the use of thin shell elements, TACS solves the finite-element analysis problem via an efficient parallel direct matrix factorization \([95]\). The
unknowns on each processor are grouped by interior unknowns, which are only coupled to variables on the same processor, and interface unknowns which are coupled to variables on other processors. Using this partitioning, the local contribution to the global stiffness matrix on processor $i$ is written as follows [95]:

$$A_i = \begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix}. \quad (3.35)$$

Using an exact LU factorization of the matrix $B_i$ in (3.35), such that $B_i = L_{B_i}U_{B_i}$, a local Schur complement of $A_i$ can be found by

$$S_i = C_i - F_iU_{B_i}^{-1}L_{B_i}^{-1}E_i. \quad (3.36)$$

The above calculation can be performed in parallel. A global Schur complement matrix is then obtained by summing the local Schur complements over all of $S$ subdomains or processors:

$$S = \sum_i^S T_i^T S_i T_i, \quad (3.37)$$

which is then factored exactly as $S = LU$. The assembly of $S$ here differs from the assembly of the global Schur complement for the flow problem in Section [3.1.3] due to the difference in discretization. Furthermore, the permutation matrices, $T_i$, in (3.37), along with ordering of the unknowns in $A_i$, were chosen to reduce the fill-in during the LU factorization of $S$. To distribute the cost associated with storing and factorizing the global Schur complement, TACS uses a sparse two-dimensional (2D) block-cyclic matrix storage format [95]. Kennedy and Martins [95] provide the additional details on the parallel factorization procedure as well as various investigations on the performance of the parallel analysis capability of TACS. To ensure the convergence of the finite-element analysis, GMRES is used for the iterative refinement of the direct solution from matrix factorization, where the factored global stiffness matrix is applied as a preconditioner [99].

### 3.2.3 Force and Displacement Transfer

During an aerostructural analysis, the coupling between the flow and the structures is accomplished by transferring the forces and displacements through the aerostructural interface. This interface includes the flow grid nodes on the surface of the geometry and structural nodes adjacent to the aerodynamic surface. TACS is equipped with a force and displacement transfer algorithm which allows it to be integrated into an aerostructural analysis and optimization framework [73]. The force and displacement transfer scheme in TACS is based on the rigid link method by Brown [100]. It involves creating rigid link vectors which connect the aerodynamic and structural grids. These vectors then allow displacements and forces to be extrapolated between the two grids even though they may not necessarily overlap at the interface. This is particularly useful when modelling structures such as the wingbox, where there is a gap between the aerodynamic geometry and the structures at the leading and trailing edges, as well as at the wing tip. The relevant details of this approach are reviewed below.

The rigid link vectors are created at the start of an optimization. Each flow grid node is paired with the closest point on the structural model. The rigid link vector for the $j^{th}$ flow grid node, $r_j$, is given
by the following expression:
\[ r_j = x^e_k(\xi_j) - G_{sJ,j}, \] (3.38)
where \( G_{sJ,j} \) and \( x^e_k(\xi_j) \) are the coordinates of the flow grid node and the point on the structural model, respectively. The element index, \( k \), and the parametric coordinates, \( \xi_j \), within the element are determined such that the length of the rigid link vector, \( ||x^e_k(\xi_j) - G_{sJ,j}||_2 \), is minimized. As \( G_{sJ,j} \) and \( x^e_k(\xi_j) \) change throughout optimization due to changes in the design geometry, the length of the rigid link vector is not guaranteed to remain as the shortest distance between the flow grid node and the structural model. As a result, the consistency of shape changes between the structure and the OML during optimization is an important requirement for the structural mesh movement algorithm discussed in Section 4.4. This ensures that the lengths of the rigid link vectors, as well as the results of the displacement and force transfer, remain reasonable.

The objective of the displacement transfer is to compute the displacements of the surface flow grid nodes from the structural state variables. Using the rigid link vector, \( r_j \), defined above, the displacement of the corresponding flow grid node, \( u_{A_j} \), is given by \[ u_{A_j} = U_0(\xi_j) + \Theta(\xi_j) \times r_j. \] (3.39)

The displacement and rotation at the parametric location \( \xi_j \) in the \( k \)th element, \( U_0(\xi_j) \) and \( \Theta(\xi_j) \), are evaluated according to (3.31).

The inputs to the force transfer include the aerodynamic surface traction, \( f_A \), computed in Section 3.1.4 as well as a set of surface grid nodes describing the OML at its deflected state, \( S_A \). Using this information, the appropriate force vector \( f_S \) in (3.34) can be obtained for structural analysis. The force transfer calculation is derived by examining the virtual work performed by the pressure field through the displacements of the aerodynamic surface. More specifically, it should be equal to the work performed by \( f_S \) through the displacement at each structural DOF. This equality can be expressed as
\[ \delta W = \int_{S_A} f_A(\eta)^T \delta u_A(\eta) dS_A = f_S^T \delta u. \] (3.40)
The LHS to (3.40) integrates the aerodynamic surface traction distribution, \( f_A(\eta) \), over the deflected aerodynamic geometry, \( S_A \), where \( \eta \in \mathbb{R}^2 \) is the parametric coordinates on \( S_A \).

The integration over \( S_A \) can be performed numerically using Gauss quadrature of order chosen to match the order of the shell elements. Equation (3.40) then becomes
\[ f_S \delta u = \sum_{c=1}^{n_c} \sum_{j=1}^{n_{GQ}} w_j S_{A,j} f_A(\eta_j)^T \delta u_A(\eta_j), \] (3.41)
where \( n_c \) is the number of cells on the aerodynamic surface grid, and \( n_{GQ} \) is the number of quadrature points. Furthermore, \( w_j \), \( S_{A,j} \), and \( \eta_j \) represent the quadrature weight, surface area contribution, and parametric coordinates, respectively, of the \( j \)th quadrature point. The locations of the quadrature points in general do not coincide with the nodes on the discrete surface at which the values of \( f_A \) are known from the flow module. Therefore, both \( S_A \) and \( f_A(\eta) \) are reconstructed in space from the discrete set of nodal coordinates and surface tractions available.

Evaluating the displacements, \( \delta u_A(\eta_j) \), due to structural deflections at all quadrature points in (3.41)
requires a second set of rigid link vectors. The rigid link vectors for the force transfer are found by replacing $G_{s,j}$ in (3.38) with the coordinates of a quadrature point on $S_A$. After substituting (3.39) for displacement transfer into $\delta u_A(\eta_j)$, (3.41) becomes

$$f_T^S \delta u = \sum_{c=1}^{n_c} \sum_{j=1}^{n_{GQ}} w_j S_A,j \left[ f_A(\eta_j)^T \delta U_0(\xi_j) - (f_A(\eta_j) \times r_j)^T \delta \Theta(\xi_j) \right].$$  (3.42)

Finally, using the shape functions within the structural element according to (3.31) and (3.32), $\delta U_0(\xi_j)$ and $\delta \Theta(\xi_j)$ can be written in terms of the structural state, $\delta u$. This leads to the following expression for the structural force vector [73]:

$$f_S = \sum_{c=1}^{n_c} \sum_{j=1}^{n_{GQ}} w_j S_A,j (P_k \otimes I_6) N^T_k(\xi_j) \left[ f_A(\eta_j) - f_A(\eta_j) \times r_j \right].$$  (3.43)

The cross product of the surface traction with the rigid link vector (3.43) results in non-zero moments in $f_S$.

Kennedy and Martins [73] introduced a subcell refinement procedure in the force transfer calculations to improve the smoothness of the resulting von Mises stress distribution. Subcells are added to the original aerodynamic surface to match the discretization level between the flow and structural grids. In this case, the outer summation in (3.41) would be over the number of subcells instead of the number of cells on the original surface.

3.2.4 Structural Functionals

TACS is used for the calculation of structural functionals, denoted by $J_S$, including the mass and structural failure constraints. The structural mass is computed by summing the contributions from all structural elements. It has an explicit dependence on the geometric and structural thickness design variables but does not depend explicitly on the flow or structural state variables. The structural failure constraint requires the load factor, $\lambda_k$, in every element $k$, to satisfy $\lambda_k = F_s \times \left( \sigma_k / \sigma_{yield} \right) \leq 1$, where $\sigma_k$ is the von Mises stress in the element, and $\sigma_{yield}$ is the yield stress of the material [95, 101]. The factor of safety, $F_s$, is given a value of 2 for the results presented in this thesis. The von Mises stress is a function of the local strain values computed from the structural state, structural design variables controlling the thickness of the elements, and other material properties specified by the user [95]. It has an implicit dependence on the geometric design variables through the strain. The von Mises stress of an element is computed using the average strain value over all Gauss quadrature points in the element.

A constraint aggregation technique is adopted in TACS to reduce the total number of structural failure constraints required for an optimization, so that gradient calculation using the coupled adjoint method remains efficient. Specifically, a discrete Kreisselmeier-Steinhauser (KS) function [83, 102] is used in this thesis, which is given by:

$$KS = \lambda_{max} + \frac{1}{\rho_{KS}} \ln \left\{ \sum_{k=1}^{N_e} \exp \left[ \rho_{KS}(\lambda_k - \lambda_{max}) \right] \right\},$$  (3.44)

where $\lambda_{max} = \max_k \lambda_k$, $N_e$ is the number of elements, and $\rho_{KS}$ is a positive user-specified weighting parameter. The KS function overestimates the maximum $\lambda_k$ over the aggregated elements by at most
\[ \ln(N_e)/\rho_{KS} \] In the limit of \( \rho_{KS} \to \infty \), \( KS \to \lambda_{\text{max}} \). Larger \( \rho_{KS} \) values lead to a less conservative and more accurate representation of the failure envelope, but at the cost of reducing the smoothness of the optimization problem. The appropriate value of \( \rho_{KS} \) is typically between 30 and 50 [72, 103]. The primary focus of this work is not on the accurate weight assessment of a realistic aircraft. A \( \rho_{KS} \) value of 30 is therefore used for the results presented in this thesis, which leads to better conditioning of the optimization problem [95]. Furthermore, a separate KS function is typically used for each of the top skin, bottom skin, ribs and spars modelled in the present methodology, instead of aggregating the element-wise failure criteria over the entire structural domain. Grouping elements by the structural components they belong to can lead to constraints which are less nonlinear and therefore easier to satisfy during an optimization [95].

3.2.5 Calculation of Design Variable Sensitivities

Kennedy and Martins [95] describe in detail the gradient calculation for a purely structural optimization problem using the discrete adjoint approach. Many of the calculations are also necessary for aerostructural optimization. Hence they are summarized here. The structural adjoint problem is given by [95]:

\[ \frac{\partial R_S}{\partial u}^T \Psi_S = K_S \Psi_S = - \frac{\partial J_S}{\partial u}^T. \] (3.45)

Due to the symmetry of the structural stiffness matrix, \( K_S \), which is part of the structural residual in \[3.34\], the parallel direct factorization of \( K_S \) in Section 3.2.2 can be reused. The RHS of (3.45) is first computed within each element, then summed over the structural domain [95].

Upon solving (3.45), the following expression is evaluated for the gradient with respect to a set of design variables, \( v \):

\[ \frac{dJ_S}{dv}^T \Psi_S = \left( \frac{\partial J_S}{\partial v} \bigg|_u \right)^T + \left( \frac{\partial R_S}{\partial v} \bigg|_u \right)^T \Psi_S. \] (3.46)

Partial derivative terms on the RHS of (3.46) are calculated differently for structural design variables, \( v_S \), and geometric design variables, \( v_G \) [95]. For structural thickness variables, \( \partial J_S/\partial v_S \) and \( \left( \partial R_S/\partial v_S \right)^T \Psi_S \) are computed within each element and subsequently summed over the structural domain. The evaluation of \( \partial J_S/\partial v_G \) and \( \left( \partial R_S/\partial v_G \right)^T \Psi_S \) first requires differentiating \( J_S \) and \( R_S \) with respect to all nodal coordinates of the structural mesh, \( X_S \). The results are then contracted with \( \partial X_S/\partial v_G \), which is the partial derivative of the structural mesh with respect to geometric design variables [95]. This term can be computed outside of TACS, and it depends on the choice of geometry parameterization.

For the structural functionals considered in this thesis, \( \Psi_S = 0 \) for mass as it does not depend on the structural deflections, and \( \partial J_S/\partial v_G = 0 \) for the KS functionals because the von Mises stress only has an implicit dependence on the geometry. Furthermore, partial derivatives with respect to \( v_S \) can be used as is during an aerostructural optimization, while \( \partial R_S/\partial v_G \) requires some modifications. The specifics of such modifications will be discussed in the context of the aerostructural optimization methodology.

3.3 Chapter Summary

In describing the aerodynamics and structural modules, this chapter introduced the governing equations for the flow and structural analysis, as well as how they are discretized. The force and displacement
transfer procedures through which the flow and structural modules are coupled were discussed. Furthermore, this chapter summarized the sophisticated parallel solution routines in each module which enable the large-scale high-fidelity computations. Relevant details of the sensitivity analysis were also described. All of the above are important ingredients for the aerostructural optimization methodology presented in later chapters.
Chapter 4

Integrated Geometry Parameterization and Grid Movement

The integrated geometry parameterization and mesh movement algorithm of Hicken and Zingg [44] is advantageous for exploratory optimization for two reasons: a flexible geometry parameterization that gives the optimizer sufficient freedom to explore novel features while maintaining an analytical geometry representation throughout, as well as an efficient and robust mesh movement strategy to accommodate large shape changes. This chapter reviews some of the important aspects of the integrated approach as it was originally proposed for aerodynamic shape optimization. This is necessary to facilitate later discussions and is accomplished in Sections 4.1, 4.2, and 4.5. In addition, the original methodology has been extended in a number of ways to enable its application to aerostructural optimization. Section 4.4 introduces a new algorithm to deform the internal structure consistently with the B-spline geometry parameterization. Furthermore, efficiency of the flow grid movement is much more crucial in aerostructural optimization than in purely aerodynamic shape optimization, because it is used not only for shape changes due to optimization but is also coupled to the aerostructural analysis. Section 4.3 discusses the use of parallel mesh solution strategies to further accelerate the flow grid movement during aerostructural calculations.

4.1 Geometry and Flow Grid Parameterization using B-Spline Surfaces and Volumes

The integrated geometry parameterization and mesh movement technique uses a B-spline tensor-product volume which maps a point from parametric space, $D = \{ \xi = (\xi, \eta, \zeta) \in \mathbb{R}^3 | \xi, \eta, \zeta \in [0,1] \}$, to physical space, $P \subset \mathbb{R}^3$. The mapping is described by the following mathematical relationship [44]:

$$
x(\xi) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{k=1}^{N_k} B_{ijk} N_i(\xi) N_j(\eta) N_k(\zeta),
$$

(4.1)
where $B_{ijk}$ refers to the coordinates of individual de Boor control points, and $\hat{N}_i$, $\hat{N}_j$, and $\hat{N}_k$ are the number of control points in each parametric coordinate direction. The B-spline basis functions of order $\mathcal{p}$ in each direction are given by $N_i^{(p)}(\xi), N_j^{(p)}(\eta)$, and $N_k^{(p)}(\zeta)$. The grid of B-spline control points embeds and fully describes the computational grid for flow calculations given the parametric coordinates of all flow grid nodes in the control volume. The flow grid can be modified via changes to the B-spline control volume, and the number of control grid nodes is typically about two orders of magnitude fewer than the number of flow grid nodes. Describing the state of the flow grid in terms of B-spline control points also reduces the size of the analysis and adjoint problems.

Hicken and Zingg [44] used generalized basis functions that incorporate spatially-varying knots, so that they can be tailored to different edges of a geometry. Basis functions in the $\xi-$direction are given by

$$
N_i^{(1)}(\xi; \eta, \zeta) = \begin{cases} 
1 & \text{if } T_i(\eta, \zeta) \leq \xi < T_{i+1}(\eta, \zeta) \\
0 & \text{otherwise}
\end{cases}
$$

$$
N_i^{(p)} = \left(\frac{\xi - T_i(\eta, \zeta)}{T_{i+p-1}(\eta, \xi) - T_i}\right) N_i^{(p-1)}(\xi; \eta, \zeta) + \left(\frac{T_{i+p}(\eta, \zeta) - \xi}{T_{i+p}(\eta, \zeta) - T_{i+1}(\eta, \zeta)}\right) N_{i+1}^{(p-1)}(\xi; \eta, \zeta),
$$

where $T_i(\eta, \zeta)$ are the knot values. Open knot vectors are used where the first and last $p$ knots of $T_i(\eta, \zeta)$ are forced to be 0 and 1, respectively. The internal knots follow a bilinear distribution such that

$$
T_i(\eta, \zeta) = [(1 - \eta)(1 - \zeta)] T_{i,(0,0)} + [\eta(1 - \zeta)] T_{i,(1,0)} + [(1 - \eta)\zeta] T_{i,(0,1)} + [\eta \zeta] T_{i,(1,1)},
$$

where $T_{i,(0,0)}$, $T_{i,(1,0)}$, $T_{i,(0,1)}$, and $T_{i,(1,1)}$ are the $i^{th}$ knot values along the $\eta$ and $\zeta$ edges of the parameter space. The basis functions $N_j^{(p)}(\eta; \xi, \zeta)$ and $N_k^{(p)}(\zeta; \xi, \eta)$ are defined in a similar fashion.

Parameterization of the flow grid requires the parametric coordinates of each grid node, determined using a chord-length parameterization. The edge knot values are chosen to ensure an equal number of nodes within each knot interval, resulting in a chord-length-based knot distribution. The initial B-spline control point coordinates are found by a least-squares fitting of the initial flow grid for the optimization. The control volume has mesh spacing that resembles a coarsened flow grid due to the nature of the knot distribution. This characteristic is exploited in the mesh movement algorithm described later. For the multi-block structured flow grids used in this work, each block is described by a separate control grid with coincident control points at the block interfaces to ensure continuity [44].

Figure [4.1] provides an example of the control volumes and the corresponding flow grid. Control points at the surface of the geometry are represented by blue spheres. They define a set of B-spline surface patches which analytically describe the geometry of interest. In Figure [4.1] a planar wing geometry is manually deformed into a user-specified C-wing geometry using the integrated geometry parameterization and mesh movement methodology. It has been chosen to demonstrate the capability of the integrated approach in handling extreme shape changes. It shows that a wide range of geometries can be defined by moving the surface control points. This example further presents a challenging flow and structural mesh deformation problem. Therefore, it is used throughout the present and the next chapter to illustrate various aspects of the extended integrated methodology for aerostructural optimization.

In the present framework, the geometric design variables, $v_G$, directly manipulate the coordinates of the B-spline surface control points to specify the undeflected design, or jig shape. Planform variables like sweep and sectional twist are defined by a combination of linear constraints and control point
4.2 Flow Grid Movement based on Linear Elasticity

Changes in the B-spline surfaces are propagated to the interior of the control volume via a linear elasticity mesh movement algorithm. It was first developed by Truong et al. [105] for two-dimensional problems, and was later extended to three-dimensions by Hicken and Zingg [44]. Movement of the flow grid is heavily involved in the derivations and algorithms presented throughout this thesis. In order to provide 

\[1\] Used with permission from [104]
The mesh movement algorithm models the control grid as an elastic solid, subjected to displacement boundary conditions specified at the surface of the geometry, on the symmetry plane, and at the far field boundaries. It is assumed that the strains in the solid are sufficiently small such that the nonlinear terms in the strain-displacement relationship can be neglected \[106\], and that the stresses and strains are related linearly. A finite-element discretization is applied to the static equilibrium equation governing the internal deformation of the grid of control points. Trilinear elements, as illustrated in Figure 4.2, are used. The displacement at any point within the element is interpolated from the displacements of the eight control points at the corners of the element. Furthermore, the Young’s modulus of the element is defined such that elements that are smaller or more skewed are stiffer \[44\]. This leads to a spatially-varying Young’s modulus that ensures the quality of the deformed control grid by attenuating any deformations that may further distort the elements.

The finite-element discretization results in a linear system in the form of \[44, 105\]:

\[
K_M(\mathbf{b}^{(0)}) \Delta \mathbf{b} = f_M(\mathbf{b}^{(0)}, \mathbf{b}_s), \quad \text{where} \quad \Delta \mathbf{b} = \mathbf{b} - \mathbf{b}^{(0)}. \tag{4.4}
\]

Vectors of the initial and final control grid coordinates are given by \(\mathbf{b}^{(0)} \in \mathbb{R}^{nM}\) and \(\mathbf{b} \in \mathbb{R}^{nM}\), respectively, where \(n_M\) is the number of unique or non-overlapping DOF in the finite-element problem. The vector of unknowns, \(\Delta \mathbf{b}\), in (4.4) include DOF which are constrained by the displacement boundary conditions. They are referred to as the fixed DOF. The global stiffness matrix, \(K_M \in \mathbb{R}^{nM \times nM}\), is obtained by summing appropriate entries from the element stiffness matrices throughout the domain. It is a function of \(\mathbf{b}^{(0)}\) due to the spatially varying Young’s modulus. The implicit force vector, denoted by \(f_M \in \mathbb{R}^{nM}\), is determined from the displacement boundary conditions. It is a function of \(\mathbf{b}_s \in \mathbb{R}^{nM_s}\), which is a vector of B-spline surface control point coordinates describing the geometry to be achieved, and is the input that drives the mesh movement.

The implicit force vector, \(f_M\), needs to be differentiated for gradient calculations (Chapter 5) and the Newton-Krylov solution to the coupled analysis problem (Chapter 6). Hence it is worthwhile to provide additional details on how \(f_M\) is computed from the displacement boundary conditions imposed on different boundary nodes. For DOF on the surface of the geometry, the boundary displacements are constrained by \((\mathbf{b}_s - P_s \mathbf{b}^{(0)})\), where \(P_s \in \mathbb{R}^{nM_s \times nM}\) is a restriction operator that extracts from \(\mathbf{b}^{(0)}\) the entries corresponding to any surface DOF. The boundary displacements are zero for fixed DOF on a
far-field boundary or symmetry plane. To enforce the boundary conditions, the prescribed displacement at a fixed DOF is assigned to the corresponding entry in $f_M$, while the same row in $K_M$ is replaced by a 1 on the diagonal. For the remaining unknowns, any dependence on the fixed DOF are removed from $K_M$ and instead included in $f_M$. In other words, if $\Delta b_{BC} \in \mathbb{R}^{n_M}$ is a vector filled with zeros except for entries specifying the boundary displacements at the fixed DOF, then $f_M$ can be obtained by

$$f_M = -K_{Mf}(b^{(0)}) \Delta b_{BC} = -K_{Mf}(b^{(0)}) P_s^T [b_s - P_s b^{(0)}] ,$$ \hfill (4.5)

where $K_{Mf} \in \mathbb{R}^{n_M \times n_M}$ is assembled in a similar fashion as $K_M$ by summing appropriate stiffness entries from elements containing one or more fixed DOF. For this reason, $K_{Mf}$ and $f_M$ are both functions of $b^{(0)}$. It is also possible to determine $\Delta b_{BC}$ directly from the displacements at the surface of the geometry using $P_s^T$ as a prolongation operator.

Equation (4.4) assumes that the shape changes are achieved in one step starting from the initial geometry. However, mesh movements during exploratory optimization must respond to shape changes that are often large and highly nonlinear. In order to avoid violation of the small strain assumption, it can be necessary to move the flow grid in $m$ increments. Let $b_s^*$ be the final geometry, and let each increment account for $1/m$ of the total surface displacements given by $(b_s^* - P_s b^{(0)})$. The boundary displacement at the surface of the geometry then becomes

$$b_s^{(i)} - P_s b^{(i-1)} = \left[ \frac{i}{m} (b_s^* - P_s b^{(0)}) + P_s b^{(0)} \right] - P_s b^{(i-1)} , \quad i = 1, \ldots, m .$$ \hfill (4.6)

The vector of all control point coordinates at the $i$th increment are given by $b^{(i)}$. In aerodynamic shape optimization, (4.6) can be simplified to $(b_s^{(m)} - P_s b^{(0)})/m$ because the prescribed displacement boundary conditions are satisfied at every increment. The same assumption is not necessarily true when the flow grid movement is coupled to aerostructural analysis. It is therefore important to write (4.6) in the above form.

Moving the flow grid in $m$ increments involves solving the following equations for $i = 1, \ldots, m$ [44]:

$$R_M^{(i)} (b^{(i-1)}, b^{(i)}) = K_M^{(i)} (b^{(i-1)}) [b^{(i)} - b^{(i-1)}] - f_M^{(i)} (b_s^{(i)} - P_s b^{(i-1)}) = 0 ,$$ \hfill (4.7)

where $R_M^{(i)}$ is the mesh residual at the same increment. The stiffness matrices $K_M^{(i)}$ and $K_{Mf}^{(i)}$ for the calculation of $f_M^{(i)}$ need to be recomputed at every increment because they are functions of $b^{(i-1)}$. Otherwise, they are assembled in the same manner as for (4.4). Upon solving (4.7) for all $m$ increments, the new computational grid used in flow analysis is simply re-evaluated according to (4.4).

Linear elasticity mesh movement is very robust even in the presence of large shape changes. Furthermore, it is much cheaper to apply this method to the control grid as opposed to the flow grid. The quality of the flow grid is nonetheless preserved by the similarity in the mesh spacing between the control grid and the flow grid.

### 4.3 Parallelizing the Flow Grid Movement Calculations

The original algorithm by Hicken and Zingg [44] solves the mesh movement equations in (4.7) in serial. It is often sufficient in aerodynamic shape optimization because the flow grid movement only contributes to a fraction of the total cost per design iteration. In aerostructural optimization, however, movement of
the flow grid is part of the coupled analysis and adjoint calculations, so its efficiency becomes much more crucial. Some of the results presented in this thesis have been obtained using the parallel mesh movement implementation by a former colleague in the group, Dr. Hugo Gagnon. It involves solving the linear system at every mesh movement increment using the parallel preconditioned conjugate gradient (PCG) solver from the PETSc library [107]. The iterative solution uses an additive Schwarz preconditioner with local Jacobi or incomplete Cholesky preconditioning, all of which are available from PETSc.

Although the use of PETSc reduces the mesh movement time substantially by solving the mesh equations in parallel, the stiffness matrix at each increment is assembled in serial and stored on a single processor. The time required for matrix-assembly becomes a bottleneck in the overall efficiency of the mesh movement calculations. The associated memory cost also prevents the possibility of storing $K_M^{(i)}$ and its factorization for subsequent calculations. Neither of the above are significant issues during a nonlinear block Gauss-Seidel solution to the coupled analysis problem, which will be presented in Chapter 5. Nonetheless, recomputing $K_M^{(i)}$ during the coupled adjoint calculations even though it does not change between successive iterations results in unnecessary computational overhead. Chapter 6 will further show that the need to assemble $K_M^{(i)}$ repeatedly can significantly compromise the performance of the monolithic solution method for both analysis and coupled adjoint calculations. It is therefore necessary to store the mesh movement data and assemble the stiffness matrix in a distributed manner.

There are two ways to accomplish the above objective. PETSc provides a number of routines which can be used to assemble, store, and factor $K_M^{(i)}$ in parallel. Alternatively, a new parallel mesh solution capability can be developed based on the serial algorithm from Hicken and Zingg [44]. With the first option, any access to $K_M^{(i)}$ is through PETSc, which will affect the evaluation of the mesh residual and any matrix-vector products involving $K_M^{(i)}$. This leads to a mesh movement module that is more centred around the use of PETSc. The second option, on the other hand, allows for more direct control over different parts of the parallel mesh movement algorithm. This is important during the development of the monolithic solution method. As a result, the second option is chosen for this thesis. Details of the parallel mesh solution methodology are provided in the remainder of this section. Its performance is benchmarked against the existing parallel mesh solution using PETSc in Section 4.3.5.

4.3.1 Distributed Data Storage

The current implementation assumes that there is one grid block on each processor. A control grid node in each block can either be an interior or an interface node, depending on whether or not it lies on an interface with an adjacent block. Interior nodes belong to elements that are local to the current processor, while interface nodes may belong to elements from multiple processors. DOF associated with interface nodes are ordered last on each processor. For the mesh movement equation at the $i^{th}$ increment, the local stiffness matrix and unknowns on processor $j$ are partitioned as follows:

$$\tilde{K}_j^{(i)} = \begin{bmatrix} A_j^{(i)} & E_j^{(i)} \\ E_j^{(i)} T & \tilde{C}_j^{(i)} \end{bmatrix} \quad \text{and} \quad \Delta b_j^{(i)} = b^{(i)} - b^{(i-1)} = \begin{bmatrix} \Delta b_j^{(i), \text{local}} \\ \Delta b_j^{(i), \text{interface}} \end{bmatrix}. \quad (4.8)$$

There are a total of $n_j$ unknowns on processor $j$. Given that there are 3 DOF per node, $n_j$ is equal to 3 times the number of control points in the $j^{th}$ block of the control grid. This leads to a substantially smaller memory requirement than maintaining the full mesh movement problem on a single processor, making it feasible to store the mesh vectors and matrices for multiple increments. The local stiffness
matrix, $\tilde{K}_j^{(i)} \in \mathbb{R}^{n_j \times n_j}$, on each processor is assembled in the same way as the global stiffness matrix in the serial case. Unknowns which are interior to the block, $\Delta b_{j,\text{local}}^{(i)} \in \mathbb{R}^{n_j,\text{local}}$, are coupled by $A_j^{(i)}$. Interior unknowns are coupled to the interface unknowns on the same block or processor, $\Delta b_{j,\text{interface}}^{(i)} \in \mathbb{R}^{n_j,\text{interface}}$, by $E_j^{(i)}$ and $E_j^{(i)T}$. The matrix $C_j^{(i)}$ contains entries that couple the interface unknowns on the local processor, and is summed over all processors to obtain the coupling matrix for all interface unknowns in the global stiffness matrix $K_M^{(i)}$. All variables used throughout this section are specific to the $i^{th}$ increment, so the superscript $(i)$ will be dropped for convenience. Similarly, $n_{M,\text{interface}}$ is used to refer to the size of the mesh problem at each increment for the purpose of the present discussion. Let $s$ be the number of processors or blocks in the flow or control grid, and $R_j \in \mathbb{R}^{n_j,\text{interface} \times n_{M,\text{interface}}}$ be the restriction operator that extracts the local interface unknowns from the vector of all $n_{M,\text{interface}}$ interface unknowns over the domain. Then the global linear system in (4.4) can be written as follows:

$$
\begin{pmatrix}
A_1 & 0 & \cdots & 0 & E_1R_1 \\
0 & A_2 & \cdots & 0 & E_2R_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & A_s & E_sR_s \\
R_1^T E_{1}^T & R_2^T E_{2}^T & \cdots & R_s^T E_{s}^T & \sum_{i=1}^{s} R_i^T \tilde{C}_j R_j \\
\end{pmatrix}
\begin{pmatrix}
\Delta b_{1,\text{local}} \\
\Delta b_{2,\text{local}} \\
\vdots \\
\Delta b_{s,\text{local}} \\
\Delta b_{\text{interface}} \\
\end{pmatrix}
= 
\begin{pmatrix}
f_{1,\text{local}} \\
f_{2,\text{local}} \\
\vdots \\
f_{s,\text{local}} \\
f_{\text{interface}} \\
\end{pmatrix}.
$$

(4.9)

The interface unknowns, $\Delta b_{\text{interface}}$, for the global problem are also placed at the end of the vector of all unknowns. The linear system in (4.9) is solved using a parallel PCG algorithm, where the partitioning in (4.9) allows any matrix-vector products with $K_M$ to be computed in a distributed manner. A parallel PCG solution algorithm further requires an appropriate preconditioner. This will be discussed next.

### 4.3.2 Local Factorization and the Additive Schwarz Preconditioning

The additive Schwarz preconditioner is a relatively straightforward way to precondition the conjugate gradient solution to (4.9). It can be written as

$$
K_M^{-1} \approx M^{-1} = \sum_{j=1}^{s} P_j^T K_j^{-1} P_j, \quad \text{where} \quad K_j = P_j K_M P_j^T.
$$

(4.10)

The matrix $P_j$ is a restriction operator for all variables on processor $j$ such that

$$
P_j K_M P_j^T = \begin{bmatrix} A_j & E_j \\ E_j^T & R_j C R_j^T \end{bmatrix}, \quad \text{where} \quad C = \sum_{i=1}^{s} R_i^T \tilde{C}_j R_j.
$$

(4.11)

The restricted matrix $K_j$ has the same sparsity pattern as the local contribution $\tilde{K}_j$ to $K_M$, but it should be noted that $K_j \neq \tilde{K}_j$. It is sufficient to approximate $K_j^{-1}$ in (4.10) by an ILU factorization of $K_j$ obtained using the SPARSKIT library. An ILU factorization does not exploit the symmetry of $K_j$; hence an incomplete Cholesky (IC) factorization routine with zero fill has also been implemented. The performance of the additive Schwarz preconditioner with different local factorizations will be investigated in Section 4.3.5.
4.3.3 Approximate Schur Preconditioning

Despite the simplicity of the additive Schwarz preconditioner, an alternative based on the approximate Schur complement of $K_M$ has been shown to be more effective in reducing the number of Krylov iterations required [86, 108]. This can lead to better computational efficiency in comparison to the additive Schwarz preconditioner, especially with calculations involving many processors. Therefore, an approximate Schur preconditioner has also been considered. The Schur complement matrix, $S$, relating all interface unknowns in (4.9) is defined as follows:

$$S = C - E^T A^{-1} E,$$  
(4.12)

where

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_s \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} E_1 R_1 \\ E_2 R_2 \\ \vdots \\ E_s R_s \end{bmatrix}.$$  
(4.13)

The Schur complement matrix can be computed in parallel by taking advantage of the block structure in $K_M$ [109]:

$$S = \sum_{j=1}^s R_j^T \tilde{C}_j R_j - \sum_{j=1}^s R_j^T E_j^T A_j^{-1} E_j R_j$$  
(4.14)

$$= \sum_{j=1}^s R_j^T \left( \tilde{C}_j - E_j^T A_j^{-1} E_j \right) R_j,$$

where $S_j$ is the Schur complement of the locally assembled stiffness matrix, $\tilde{K}_j$.

The Schur complement preconditioner requires an approximate solution to

$$S \Delta b_{\text{interface}} = f_{\text{interface}} - E^T A^{-1} E f_{\text{local}},$$  
(4.15)

where $\Delta b_{\text{interface}}$, $f_{\text{interface}}$, and $f_{\text{local}}$ are as in (4.9). The Schur complement problem requires its own PCG solution. Matrix-vector products with $S$ during the iterative solution can take advantage of the following inexact factorization of $K_j$ on the $j$th processor [108]:

$$K_j = \begin{bmatrix} A_j & E_j \\ E_j^T & R_j C R_j^T \end{bmatrix} \approx \begin{bmatrix} L_{A_j} & 0 \\ E_j U_{A_j}^{-1} L_{S_j} \\ 0 & U_{S_j} \end{bmatrix} \begin{bmatrix} U_{A_j} & L_{A_j}^{-1} E_j^T \\ 0 & U_{S_j} \end{bmatrix},$$  
(4.16)

which allows the local Schur complement matrix to be approximated by

$$S_j \approx \tilde{C}_j - E_j^T U_{A_j}^{-1} L_{A_j}^{-1} E_j.$$  
(4.17)

Equation (4.17) can be combined with (4.14) to obtain an approximation to the global Schur complement matrix. The inexact factorization of $K_j$ likewise allows the Schur complement problem to be preconditioned as follows:

$$S^{-1} \approx M_s^{-1} = \sum_{j=1}^s R_j^T U_{S_j}^{-1} L_{S_j}^{-1} R_j.$$  
(4.18)
4.3 Parallelizing the Flow Grid Movement Calculations

Equation (4.18) is like an additive Schwarz preconditioner for (4.15), except it can be shown that \( L_S, U_S \) is not quite the approximate factorization of \( R_jS R_j^T \). Nevertheless, (4.18) appears to work reasonably well. It is found that solving (4.15) to a tolerance of 0.1 or a maximum of 5 iterations is sufficient to ensure the effectiveness of the Schur complement preconditioner. Upon solving the approximate Schur complement problem, the solution to (4.15) can be used to calculate the interior unknowns local to each processor.

4.3.4 Scaling

Experience with the parallel solution to (4.4) using the PETSc library has shown that a simple Jacobi preconditioner works effectively with PCG. This suggests that it is perhaps beneficial to scale the linear system in (4.9) as follows:

\[
D^{-1/2} K_M D^{-1/2} \begin{pmatrix} \frac{1}{2} \Delta b \\ \Delta b \end{pmatrix} = D^{-1/2} f_M, \tag{4.19}
\]

where \( D \in \mathbb{R}^{n_M \times n_M} \) is a diagonal matrix containing the diagonal entries of \( K_M \). The scaling procedure in (4.19) preserves the symmetry in the scaled problem. The effect of scaling will be examined in the results that follow.

4.3.5 Parallel Mesh Solution Performance

In the aerostructural optimization methodology to be introduced later, movement of the flow grid is an integral part of the coupled calculations. Therefore, it is necessary to investigate and report the efficiency of the mesh solution algorithm. This is especially important for the discussions in Chapter 6 on the monolithic solution method. The results presented in this section are intended to serve two main purposes: to benchmark the performance of the new parallel implementation of the PCG algorithm relative to the existing implementation using PETSc, and to document the solver parameters that lead to the best parallel mesh solution performance. The results presented in this section are intended to serve two main purposes: to benchmark the performance of the new parallel implementation of the PCG algorithm relative to the existing implementation using PETSc, and to document the solver parameters that lead to the best parallel mesh solution performance. The chosen test problem is the mesh movement required to produce the shape change shown in Figure 4.1, where a C-wing geometry is manually created from an initially planar wing. This is a challenging mesh movement problem due to the large shape change. Hence the efficiency and robustness of the mesh solution algorithm is particularly important. The flow grid consists of 193,536 nodes and 112 blocks. Each block is parameterized by \( 6 \times 6 \times 6 \) control points. The shape change is accomplished using 15 increments and 112 processors.

In Table 4.1, performance of the parallel PCG algorithm is benchmarked in reference to the PETSc library. Both methods use an additive Schwarz preconditioner with different local preconditioners. Columns 4 to 6 in Table 4.1 are obtained by averaging over the 15 increments, whereas the matrix assembly and factorization time in columns 7 and 8 are for the entire mesh movement calculation. For all results using the parallel PCG implementation, matrix assembly requires less than 1% of the total mesh movement time thanks to the distributed data storage. In contrast, the serial matrix assembly used with the current PETSc implementation contributes to over 30% of the total mesh movement time. Such differences will become more noticeable during coupled aerostructural calculations. This result hence substantiates the importance of assembling mesh stiffness matrix in parallel.

For the first two rows in Table 4.1, both PETSc and the parallel PCG algorithm have been set up to be identically preconditioned by a global Jacobi preconditioner involving simple inverse diagonal scaling. This allows the two methods to be compared purely from an implementation perspective. The parallel
Table 4.1: Benchmarking the performance of the parallel PCG algorithm discussed in this section with parallel mesh solution using the PETSc library. The calculations using both methods are distributed over 112 processors. The original serial implementation is also provided as a reference.

PCG requires a smaller number of iterations, but the average solution time per increment is higher than PETSc. The domain decomposition in the parallel PCG algorithm follows the blocking of the control grid, resulting in a total of 648 unknowns on each processor. The partitioning in PETSc involves evenly distributing the equations and unknowns among the processors, leading to approximately 411 unknowns per processor. The differences in the local problem size, together with the fact that the code in PETSc is likely optimized, are possible explanations for the difference in computational time per iteration.

The next three rows in Table 4.1 examine the use of local incomplete Cholesky or LU factorization for the additive Schwarz preconditioners. The parallel PCG algorithm again requires a lower number of iterations but higher computational time per iteration comparing to PETSc. In all cases, the average mesh solution time is reduced relative to the use of a Jacobi preconditioner as expected. For the parallel PCG algorithm, using an incomplete Cholesky factorization is more efficient than using ILU due to the shorter factorization time. The use of IC however requires a slightly higher number of iterations than ILU, which may be attributed to the round-off error introduced during the factorization process.

Putting the above discussions into perspective, the last row of Table 4.1 shows the mesh movement time using serial calculations. Although the number of iterations is much smaller, the total mesh movement time is longer by a factor of 10 comparing to either PETSc or the parallel PCG algorithm. Therefore, despite the small differences reported in the rest of Table 4.1, both parallel implementations provide a comparable speedup of the mesh movement calculations relative to the original serial implementation.

The first part of Table 4.2 studies the performance of the parallel PCG algorithm with different preconditioning options. Using an approximate Schur preconditioner reduces the number of iterations substantially, but the improvement has been overshadowed by the increased cost per iteration. Higher levels of fill for ILU have also been considered. Despite the increase in factorization time and time per iteration, it does not lead to a substantially lower number of iterations with either the additive Schwarz or the approximate Schur preconditioner. This is reaffirmed by the second part of Table 4.2 which examines the effectiveness of local IC factorization with higher levels of fill using PETSc. The results also show that the use of a lower level of fill is more efficient. Table 4.2 further suggests that for the parallel PCG algorithm, IC(0) should be used if matrix factorization is required during each mesh solution. Otherwise, ILU(0) may be slightly more advantageous because of the smaller number of PCG iterations.

All of the results presented so far for the parallel PCG algorithm, with the exception of those using the
4.4. Structural Mesh Deformation

The structural analysis module in the present framework uses a Lagrangian formulation, such that the structural mesh remains fixed during aerostructural analysis. The geometry of the structures, however, changes with the OML due to optimization. In fact, consistent shape changes across different disciplines is an important requirement for multidisciplinary optimization [20]. The objective of this section is to devise an appropriate strategy to deform the structural model consistently with the B-spline geometry parameterization.

It is insightful to first examine how shape changes are accomplished in the shape-optimal design of structures, which has been on its own an active area of research [113]. Numerous approaches have been proposed using B-splines or NURBS for shape parameterization. Some examples include structural shape optimization methodologies where the analysis is performed using boundary element methods [118, 114] or isogeometric analysis [40, 39, 115]. Among approaches applicable to any finite-element discretizations, Yao and Choi [116] proposed to parameterize the boundaries of 3D structures by Bezier surfaces for optimization. Deformation of the finite-element mesh was accomplished by modelling it as a linear

---

### Table 4.2: Preconditioning parameters for parallel mesh solutions.

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>Total Time (s)</th>
<th>Avg. # of Iter.</th>
<th>Time per Iter. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC(0)</td>
<td>0.13</td>
<td>10.17</td>
<td>15.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1023</td>
<td>234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00061</td>
<td>0.0042</td>
</tr>
<tr>
<td>ILU(0)</td>
<td>0.17</td>
<td>11.86</td>
<td>13.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>986</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00061</td>
<td>0.0042</td>
</tr>
<tr>
<td>ILU(1)</td>
<td>0.61</td>
<td>12.04</td>
<td>16.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>924</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0008</td>
<td>0.0051</td>
</tr>
<tr>
<td>ILU(2)</td>
<td>1.87</td>
<td>23.29</td>
<td>22.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>911</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0016</td>
<td>0.0066</td>
</tr>
</tbody>
</table>

### Table 4.3: The ratio between the largest and smallest 1-norm of the rows in the original ($K_M$) and the scaled ($\hat{K}_M$) stiffness matrices at each increment $i$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$K_M^{(i)}$</th>
<th>$\hat{K}_M^{(i)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7e+6</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>9e+6</td>
<td>7.5</td>
</tr>
<tr>
<td>3</td>
<td>1e+7</td>
<td>7.5</td>
</tr>
<tr>
<td>4</td>
<td>2e+7</td>
<td>7.5</td>
</tr>
<tr>
<td>5</td>
<td>9e+7</td>
<td>7.5</td>
</tr>
<tr>
<td>6</td>
<td>3e+9</td>
<td>7.5</td>
</tr>
<tr>
<td>7</td>
<td>6e+12</td>
<td>7.5</td>
</tr>
<tr>
<td>8</td>
<td>7e+11</td>
<td>7.5</td>
</tr>
<tr>
<td>9</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>10</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>11</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>12</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>13</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>14</td>
<td>2e+10</td>
<td>7.5</td>
</tr>
<tr>
<td>15</td>
<td>6e+10</td>
<td>7.5</td>
</tr>
</tbody>
</table>

Jacobi preconditioner, involve scaling the linear system at each increment according to (4.19). Table 4.3 shows the ratio between the maximum and minimum 1-norm of the rows in the stiffness matrix before and after scaling is applied. A well scaled matrix is one where all columns or rows have norms which are similar in magnitude [110, 111]. This is evidently not the case with the first row of Table 4.3 but the use of scaling improves the results in the second row substantially. The deformation in the control grid has led to large entries in the stiffness matrix, especially during later increments. From a linear solution perspective, it is desirable to have a well scaled matrix as it can reduce the condition number and improve the numerical behaviour of the iterative solution [111, 112]. The use of scaling did not reduce the number of PCG iterations for this particular problem. However, both the incomplete Cholesky factorization and the approximate Schur preconditioner have shown increased robustness with scaling, suggesting that scaling does improve the quality of the linear system. Due to the above observations, scaling is applied by default whenever the parallel PCG algorithm is used for the rest of this thesis.

### 4.4 Structural Mesh Deformation

The structural analysis module in the present framework uses a Lagrangian formulation, such that the structural mesh remains fixed during aerostructural analysis. The geometry of the structures, however, changes with the OML due to optimization. In fact, consistent shape changes across different disciplines is an important requirement for multidisciplinary optimization [20]. The objective of this section is to devise an appropriate strategy to deform the structural model consistently with the B-spline geometry parameterization.

It is insightful to first examine how shape changes are accomplished in the shape-optimal design of structures, which has been on its own an active area of research [113]. Numerous approaches have been proposed using B-splines or NURBS for shape parameterization. Some examples include structural shape optimization methodologies where the analysis is performed using boundary element methods [118, 114] or isogeometric analysis [40, 39, 115]. Among approaches applicable to any finite-element discretizations, Yao and Choi [116] proposed to parameterize the boundaries of 3D structures by Bezier surfaces for optimization. Deformation of the finite-element mesh was accomplished by modelling it as a linear
Chapter 4. Integrated Geometry Parameterization and Grid Movement

elastic solid subject to displacement boundary conditions. Kegl [117] used rational Bezier volumes as design elements, which enclose and parametrically describe the nodal coordinates on the finite-element mesh. The control points for the Bezier volumes are used as geometric design variables. All the above approaches have been applied to the optimization of 2D or 3D solid structures with relatively simple topologies.

A comparison with the above examples in the literature helps identify the challenges that are unique to the present problem. The first challenge is associated with the complexity of the topology. The detailed wingbox models used in this work consist of three-dimensional arrangements of surfaces in space. The topologies are therefore much more complicated than those considered by the optimization methodologies summarized earlier. It is possible to parameterize each spar or rib by a separate B-spline surface. This would then result in a collection of B-spline surfaces which must intersect in a very specific way, and it is difficult to maintain the structural geometry while moving the control points during optimization. The second difference between the present mesh deformation problem and one involved in a structural shape optimization is that the structural components cannot be parameterized independently of the OML. In other words, the B-spline surfaces or curves defining the geometric boundaries of the structures must be part of the B-spline surfaces that describe the OML. Otherwise, the B-spline representation of the OML and the structural members must be moved in such a way ensures consistent shape changes in the underlying geometries. This is also not a trivial task.

The first challenge associated with the structural topology can be effectively handled using an FFD method, which is geometric fidelity independent [23, 118]. In other words, it is not necessary to distinguish between nodes belonging to different structural components. FFD is also a volumetric parameterization technique that moves the finite-element mesh nodes in the interior of the structural model without needing a separate mesh deformation algorithm [34]. The FFD method has been used in a number of aerostructural optimization methodologies in the past [23, 34]. It is the most similar to a 3D design element approach for structural shape optimization [117, 119], but the parameterization in an FFD is achieved via B-spline or NURBS volumes. What remains now is the second challenge, where the deformation produced by the FFD must be matched with those described by the B-spline geometry parameterization. Two structural mesh deformation strategies based on FFD have been designed and developed specifically for this purpose. They are each described in the subsequent text. Other possible structural mesh deformation strategies are discussed at the end of this section.

4.4.1 Inverse FFD

The inverse FFD approach is based on direct manipulation methods [120], which have been designed for geometry modelling using FFD in the field of computer graphics. Direct manipulation methods allow the user to manipulate a selected number of pilot points on the geometry. Deformation of the FFD volume, which controls the rest of the geometry, is then determined such that the pilot points are moved exactly as the user specified. The same technique has been applied to parameterizing the geometry for aerodynamic shape optimization [121, 122].

Following the direct manipulation approach, the inverse FFD method works with an FFD volume enclosing both the discrete aerodynamic geometry and the detailed structural model. An example of such an FFD volume is provided in Figure 4.3. It uses cubic B-splines for the upper and lower surfaces, and linear B-splines in the z direction. Using the parametric coordinates of each node within the FFD volume, the coordinates of the discrete aerodynamic geometry in physical space, $P_A \in \mathbb{R}^{N_{surf} \times 3}$, and
4.4. Structural Mesh Deformation

Figure 4.3: An example of an FFD volume used in the inverse FFD approach. The lattice of FFD control points are shown in blue, the OML of the geometry in black, and the layout of the internal structure in red.

those of the structural model, \( P_S \in \mathbb{R}^{N_S \times 3} \), can be written in terms of the coordinates of the control points, \( V_{\text{FFD}} \in \mathbb{R}^{N_{\text{FFD}} \times 3} \), which form the FFD control lattice:

\[
P_A = N_A V_{\text{FFD}}, \quad \text{and} \quad P_A = N_S V_{\text{FFD}}.
\] (4.20)

In the above expression, \( N_A \in \mathbb{R}^{N_A \times N_{\text{FFD}}} \) and \( N_S \in \mathbb{R}^{N_S \times N_{\text{FFD}}} \) are the coefficient matrices for the aerodynamic and the structural geometries. The number of aerodynamic surface grid nodes, the number of structural mesh nodes, and the number of FFD control points are given by \( N_{A,\text{surf}} \), \( N_S \), and \( N_{\text{FFD}} \), respectively.

Deforming the finite-element mesh of the structural model using inverse FFD involves choosing \( P_A \) as the pilot points. At every optimization iteration, changes in \( P_A \) are given by the B-spline surface control points which describe the OML as a function of the geometric design variables. The corresponding changes in the FFD control points are then computed as follows:

\[
\Delta V_{\text{FFD}} = \left( N_A^T N_A \right)^{-1} N_A^T \Delta P_A.
\] (4.21)

Equation (4.21) solves for the deformation in the FFD volume such that the changes in the aerodynamic geometry are replicated as closely as possible in a least-squares sense. The coefficient matrix, \( N_A \), is constant. Hence its pseudoinverse, \( N_A^+ \), can be precomputed and stored in factored form during the start of an optimization. Deformation of the FFD volume also warps the structural model embedded within. The updated coordinates of the structure mesh, \( P_S^{(\text{new})} \), can be found by evaluating

\[
P_S^{(\text{new})} = P_S + N_S \Delta V_{\text{FFD}},
\] (4.22)

using \( \Delta V_{\text{FFD}} \) from (4.21). Sensitivities of the structural mesh points with respect to the geometric
design variables, $v_G$, can be obtained by differentiating (4.21) and (4.22):

$$
\frac{\partial P_S}{\partial v_G} = \frac{\partial P_S}{\partial \Delta V_{FFD}} \frac{\partial \Delta V_{FFD}}{\partial \Delta P_A} \frac{\partial \Delta P_A}{\partial b_{s_j}^*} \frac{\partial b_{s_j}^*}{\partial v_G}
$$

(4.23)

where $b_{s_j}^*$ is the vector of B-spline surface control points describing the jig shape.

The inverse FFD approach is relatively easy to implement. Forcing the deformation of the FFD volume to reproduce the changes in the OML appears to be a good way to enforce consistent shape changes across disciplines. However, the number of FFD control points is often smaller than the number of aerodynamic surface grid nodes. This means that the changes in $P_A$ are only approximately captured by the changes in the FFD volume, which translates to an error associated with the deformed structural model relative to the OML. While it is possible to reduce the error by increasing the number of FFD control points, a sufficient number of flow grid nodes is needed within each B-spline knot interval for a well-posed least-squares problem [123]. Otherwise, a unique solution cannot be found unless additional constraints are specified [122], but having to do so takes away from the simplicity of the algorithm. Furthermore, it is in general difficult to control the deformation everywhere in the FFD volume. Due to these potential sources of inaccuracy in the inverse FFD approach, an alternative has been developed which allows consistency in the shape changes to be enforced more directly.

### 4.4.2 Surface-based FFD

A surface-based deformation approach involves linking a 3D solid object with a parametric or discrete surface as a deformation tool [124]. The object can then be modified via changes to the deformation surface. The design of the surface-based FFD method is based on the same idea by linking a point on the structural model with the upper and lower surfaces of the OML. The surface-based FFD approach makes a number of assumptions with regard to the patch topology of the geometry. It is assumed that the upper and lower surfaces of the OML are made of distinct patches. In other words, there needs to be a stitch between two adjacent B-spline patches along the leading and trailing edges of the geometry. Furthermore, the number of spanwise patches on the upper and lower surfaces must be the same. These assumptions are reasonable for most geometries that are relevant to the static aerostructural design of an aircraft.

Given a point $P$ on the structural model, the surface-based FFD method begins by associating it with two points $U$ and $L$, on the upper and lower surfaces of the geometry, respectively. This is illustrated in Figure 4.4. The coordinates of $U$ and $L$ are given by

$$
U(\xi_1, \eta) = \sum_{j=1}^{N_j} \sum_{k=1}^{N_k} N_j(\xi_1)N_k(\eta)B_{jkN_m} \quad \text{and} \quad L(\xi_2, \eta) = \sum_{j=1}^{N_j} \sum_{k=1}^{N_k} N_j(\xi_2)N_k(\eta)B_{jk1},
$$

(4.24)

where $\xi_1$, $\xi_2$ and $\eta$ are the chordwise and spanwise parametric coordinates of $U$ and $L$ on the B-spline surfaces, and $B_{jkN_m}$ and $B_{jk1}$ are the corresponding surface control points. The points $P$, $U$, and $L$ are collinear, such that the coordinates of $P$ can be described by a parametric distance $\ell$ as follows:

$$
U(\xi_1, \eta) + \ell [L(\xi_2, \eta) - U(\xi_1, \eta)] - P = 0, \quad \text{where} \quad \ell = \frac{\|P - U(\xi_1, \eta)\|_2}{\|L(\xi_2, \eta) - U(\xi_1, \eta)\|_2}.
$$

(4.25)
The appropriate $U$ and $L$ are found for each $P$ at the start of an optimization. The values of $\xi_1$, $\xi_2$, $\eta$ and $\ell$ are then fixed for the remainder of the optimization, and the coordinates of $P$ become a function of the B-spline surface control points. Since $U$ and $L$ share the same $\eta$, $P$ will always remain in the same spanwise section traced by a constant $\eta$. In order to associate each $P$ with a unique pair of $(U, L)$, two constraints are defined based on vector dot products:

$$
\begin{align*}
U \cdot C &= 0 \\
L \cdot C &= 0.
\end{align*}
$$

Equation (4.26) ensures that the line $UPL$ (see Figure 4.4), along which $P$ is parameterized, is normal to the chord defined by the leading edge, $LE$, and trailing edge, $TE$, at the same value of $\eta$. For most structural layouts, these constraints ensure that $UPL$ is close to being tangent to the ribs and spars and is aligned with the direction in which sectional shape changes are defined. This can reduce the amount of unwanted distortions introduced to the structural components during shape changes.

The search for an appropriate pair of $U$ and $L$ for each grid node $P$ on the structural model needs to work with multiple surface patches. The implemented procedure is briefly outlined here. The B-spline representation of the upper and lower surfaces is divided into a number of regions, each encompassing a single spanwise patch. If the point $P$ is within the bounding box of a spanwise region, an iterative search is performed within this region for the points $U$ and $L$. The initial guess to the iterative search is obtained by computing the distance between $P$ and a total of $N_t \times N_t$ trial points on each patch. The pair of trial points with the shortest distance to $P$ are used to obtain the initial values of $\xi_1$, $\xi_2$, and $\zeta$.

If $P$ happens to be near the junction between two spanwise regions, the algorithm searches through both regions for the appropriate $U$ and $L$. However, a pair of $U$ and $L$ which form a straight line with $P$ should be unique, and they are only found in one spanwise region. To handle possible scenarios where the expression given by (4.25) is invalid, the iterative search for $U$ and $L$ takes the form of the following minimization problem

$$
\min_{\xi_1, \xi_2, \eta} \| U \times L \|_2^2.
$$

The above minimization problem is subject to the constraints given by (4.26), and it is solved via a sequential quadratic programming algorithm [125].

Posing the search as a minimization problem also allows the surface-based FFD algorithm to be extended to handle more complex geometries in the future. Figure 4.5 illustrates an example of a geometry where the leading and trailing edges, shown in red, bifurcate after a junction. The drawing on the right of Figure 4.5 shows that if $P$ is located inside the triangular prism space at the centre of
Chapter 4. Integrated Geometry Parameterization and Grid Movement

Figure 4.5: An example of a junction where a straight line through $U$, $P$, and $L$ cannot be found in any of the spanwise regions meeting at the junction.

Figure 4.6: The equivalent parametrization of surface-based FFD within the volume of space enclosed between the upper and lower surfaces of the OML is shown in black. The structural model, shown in red, and the B-spline surface control points which parameterize the OML, shown in blue, are also included.

the junction, a straight line satisfying (4.26) cannot be found in any of the spanwise regions meeting at the junction. However, if the three pairs of points: $(U_1, L_1)$, $(U_2, L_2)$, and $(U_3, L_3)$, are found by solving (4.27) in the respective region, and if the chord line vectors, $\overrightarrow{C}$, for the three regions are parallel at the junction, then by the constraints in (4.26), the three dashed lines and the point $P$ are coplanar. In this case, it is possible to parameterize the location of point $P$ using a Barycentric coordinate system. However, the treatment of this particular type of geometry has not yet been implemented into the present framework.

The surface-based FFD method effectively parametrizes the space enclosed between the B-spline surfaces and allows it to act as an FFD volume for deforming the structure. However, it does so without imposing additional restrictions on the number and distribution of control points used to describe the OML, other than the ones listed earlier. Figure 4.6 illustrates the equivalent parameterization of the volume inside the OML, shown in relation to the structural model and the B-spline surface control points. As with the FFD volume shown in Figure 4.3, the parameterization is piecewise cubic in the chordwise and spanwise directions, and linear in the vertical direction.

Associating all points on the structural model with the B-spline geometry definition is the most expensive part of this surface-based FFD method. However, it is performed once at the beginning of an optimization, and its cost is negligible in comparison to the aerostructural optimization problem. Re-evaluation of the new structural geometry involves simple algebraic expressions, so it is extremely
4.4. Structural Mesh Deformation

4.4.3 Comparison between Inverse and Surface-based FFD

While Figure 4.1 illustrates the flow grid movement in going from a planar wing to a C-wing, this section examines the structural mesh deformation corresponding to the same shape change using the inverse and surface-based FFD methods. The results are included in Figure 4.7, where Figure 4.7a shows the aerodynamic surface grid for the final geometry. Each of the upper and lower surfaces of the OML is

Figure 4.7: The surface grid and structural models after having gone through a deformation from a planar to a C-wing geometry.

efficient. Sensitivities of a structural mesh node with respect to the geometric design variables, $v_G$, can be obtained by applying the chain rule:

$$\frac{\partial P}{\partial v_G} = \left[ (1 - \ell) \frac{\partial P}{\partial U} \frac{\partial U}{\partial b_{ij}} + \ell \frac{\partial P}{\partial L} \frac{\partial L}{\partial b_{ij}} \right] \frac{\partial b_{ij}}{\partial v_G}. \quad (4.28)$$

$$\frac{\partial P}{\partial v_G} = \left[ (1 - \ell) \frac{\partial P}{\partial U} \frac{\partial U}{\partial b_{ij}} + \ell \frac{\partial P}{\partial L} \frac{\partial L}{\partial b_{ij}} \right] \frac{\partial b_{ij}}{\partial v_G}. \quad (4.28)$$
described by 5 B-spline patches in the spanwise direction and 2 in the chordwise direction. Each patch consists of $6 \times 6$ control points. The internal structure is modelled with 30,473 second-order MITC shell elements. The initial structural layout for the planar wing is shown in Figures 4.3 and 4.6 along with the deformation volume used in each structural mesh deformation method. The FFD volume in Figure 4.3 for the inverse FFD further consists of 26 and 10 control points in the spanwise and chordwise directions, respectively.

Both inverse and surface-based FFD require a few seconds to setup at the start of the simulation, but are very efficient in the subsequent evaluations of the new structure. For simple shape changes involving little rotation, the resulting structural models from inverse FFD and surface-based FFD are indistinguishable by eye. In the case of a much more dramatic shape change in going from a planar wing to a C-wing, it can be observed in Figure 4.7b that inverse FFD has produced noticeable waviness in the spars, near where the wing is bent to create the nonplanar feature. The structure also has a higher radius of curvature such that it is unable to follow the B-spline surfaces closely. This causes the structure to protrude from the wing surface in a number of locations. This is also reflected in the large root mean square (RMS) difference of 0.019 between the changes in the OML produced by the deformed FFD and the real changes described by the B-spline surfaces. In comparison, surface-based FFD does not have any of these issues. The deformed structural model is evidently of much better quality. This demonstrates the ability of the surface-based FFD approach to handle large shape changes. For this reason, the surface-based FFD approach is used for the remainder of this thesis.

4.4.4 Other Possible Alternatives

Despite the effectiveness of surface-based FFD for the purpose of this work, it makes a number of assumptions with regard to the patch topology of the geometry. For future applications where a more generalized approach is needed, some possible alternatives are discussed here.

The inverse and surface-based FFD methods allow the structural model to be created and meshed using another software package, such as ICEM. It is therefore easier to experiment with different structural topologies. The shape, distribution, and size of the elements in the structural model can also be controlled more directly. It is possible to instead create the wingbox model internally based on the B-spline parameterization of the OML, given the locations of the ribs and spars. The geometric boundaries to the structural components are then inherently part of the B-spline surfaces describing the OML, allowing the structural and aerodynamic geometries to be parameterized consistently without the introduction of an FFD volume. This nonetheless requires programming specialized routines for creating and meshing the structural model, which leads to a higher development cost.

An alternative structural mesh deformation strategy which allows the structural models to be created externally is inspired by the work of Yao and Choi [116]. This approach requires extracting the structural mesh nodes which lie on the OML. A reverse mapping procedure can be applied to identify the parametric coordinates of these boundary nodes on the B-spline surfaces describing the OML. This allows the locations of these boundary nodes to be specified by the B-spline geometry parameterization during optimization. Displacements of the remaining nodes in the structural model can then be obtained by solving a set of linear elasticity equations with the appropriate displacement boundary conditions [116]. The finite-element discretization for the structural analysis may be used for this purpose. This method involves more intrusive modifications to the structural module than the inverse and surface-based FFD. Solutions to the linear elasticity equations are also more expensive. However, it may provide a more
general way to ensure the quality of the resulting finite-element mesh.

4.5 Geometric Gradient Calculations for Aerodynamic Shape Optimization

Hicken and Zingg [44] proposed to compute the geometric gradients for the integrated geometry parameterization and mesh movement strategy through a set of mesh adjoint variables. Their approach inspired the augmented coupled adjoint formulation for aerostructural optimization, which will be introduced in the next chapter. Furthermore, some of the derivative terms computed for aerodynamic shape optimization are also needed during aerostructural optimization. For these two reasons, the geometric gradient calculations for aerodynamic shape optimization are summarized here.

In an aerodynamic shape optimization, flow grid movement is required once per design iteration in response to shape changes due to optimization. Gradients with respect to geometric design variables are obtained via changes in the flow grid, and they require the solution to the following adjoint equations:

\[
\frac{\partial R_A}{\partial q}^T \Psi_A = - \frac{\partial J_A}{\partial q}^T \quad (4.29)
\]

\[
\left( \frac{\partial R_M}{\partial b^{(m)}} \right)^T \Psi_M^{(m)} = \left[ \frac{\partial J_A}{\partial b^{(m)}} \right]^T - \frac{\partial R_A}{\partial b^{(m)}}^T \Psi_A \quad (4.30)
\]

\[
\left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i)} = \left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i+1)} - \frac{\partial R_M}{\partial b^{(i)}} \left( \Psi_M^{(i+1)} \right) \quad i = m - 1, \ldots, 1 \quad (4.31)
\]

Equation (4.29) is the flow adjoint equation from Section 3.1.5. Equations (4.30) and (4.31) are the mesh adjoint equations corresponding to the \( m \) increments used to move the flow grid. The RHS to (4.30) for the last increment includes the influence of the grid on the functional and flow residual evaluations. The two terms are more specifically computed as follows:

\[
- \frac{\partial J_A}{\partial b^{(m)}}^T = - \frac{\partial G}{\partial b^{(m)}}^T \left( \frac{\partial J_A}{\partial G} \right) \left[ \nabla \xi \right] \left( \frac{\partial \nabla \xi}{\partial G} \right) \left( \frac{\partial \Psi_A}{\partial G} \right) \quad (4.32)
\]

and

\[
- \frac{\partial R_A}{\partial b^{(m)}}^T \Psi_A = - \frac{\partial G}{\partial b^{(m)}}^T \frac{\partial \nabla \xi}{\partial G} \left( \frac{\partial R_A}{\partial \nabla \xi} \right) \left( \frac{\partial \Psi_A}{\partial \nabla \xi} \right) \quad (4.33)
\]

where \( G \) is the vector of flow grid coordinates, and \( \nabla \xi \) is the vector of grid metrics for the curvilinear coordinate transformation introduced in Section 3.1.1. The RHS to (4.31) accounts for the influence of the control point coordinates at the \( i \)th increment on the mesh residual at increment \( i + 1 \). Using (4.6) and (4.7), it can be shown that

\[
\left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i+1)} = \left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i+1)} + \left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i+1)} \quad (4.34)
\]

where

\[
\left( \frac{\partial R_M}{\partial b^{(i)}} \right)^T \Psi_M^{(i+1)} = \left[ -K_M^{(i+1)} + \frac{\partial K_M^{(i+1)}}{\partial b^{(i)}} (b^{(i+1)} - b^{(i)}) \right] \left( \Psi_M^{(i+1)} \right) \quad (4.35)
\]

\[
\left( \frac{\partial R_M}{\partial f} \right)^T \Psi_M^{(i+1)} = \left[ -K_M^{(i+1)} P_s P_s + \frac{\partial K_M^{(i+1)}}{\partial b^{(i)}} (P_s b^{(i+1)} - b^{(i)}) \right] \left( \Psi_M^{(i+1)} \right) \quad (4.36)
\]
The matrices used in the above matrix-vector products are not stored for aerodynamic shape optimization. The partial derivatives of $K_M$ and $K_{Mf}$ are evaluated using the complex-step method. Due to the symmetry of $K_M^{(i)}$, the LHS to the mesh adjoint equations are equal to the mesh stiffness matrix from the respective increment in the mesh movement. This allows the mesh adjoint equations to be solved using the solution routines described in Section 4.3.

Using the mesh adjoint variables, the gradients of the functional of interest with respect to the geometric design variables are obtained by

$$
\frac{dJ_A}{dV_G}^T = \left( \frac{\partial J_A}{\partial V_G^{(i)}} \right)^0 + \sum_{i=1}^{m} \left( \frac{\partial R_M^{(i)}}{\partial V_G} \right)^T \Psi_M^{(i)} + \left( \frac{\partial R_A}{\partial V_G^{(i)}} \right)^0 \Psi_A^T.
$$

(4.35)

An aerodynamic functional, such as lift and drag, and the flow residual have no explicit dependence on the geometric design variables. Hence the first and last terms in the above expression are zero. The partial derivative term $(\partial R_M^{(i)}/\partial V_G)^T$ accounts for changes in the prescribed boundary displacements at each increment due to changes in the geometric design variables. The matrix-vector product with this term is computed by

$$
\left( \frac{\partial R_M^{(i)}}{\partial V_G} \right)^T \Psi_M^{(i)} = \left( \frac{\partial b_*^{(i)}}{\partial V_G} \right)^T \left( \frac{\partial b_*^{(i)}}{\partial b_*^{(i)}} \right)^T \Psi_M^{(i)},
$$

(4.36)

where $I \in \mathbb{R}^{nMs \times nMs}$ is an identity matrix. Recall that $b_*^{(i)}$ describes the final geometry, and $\partial b_*^{(i)}/\partial V_G$ depends on the geometric design variable definition used for the optimization.

Most of the terms described in this section are also present in the Jacobian matrix of the discrete steady aerostructural residual, which will be introduced in the following chapter.

### 4.6 Chapter Summary

This chapter described in detail the integrated geometry parameterization and mesh movement technique, as well as a number of modifications to the original algorithm so that it is appropriate for use in aerostructural optimization. Important aspects of the linear elasticity mesh movement algorithm, which operates on the B-spline volumes describing the flow grid, were provided. This includes a parallel implementation of the mesh movement procedure introduced to accelerate the flow grid movement which will be part of the aerostructural analysis. The performance of the parallel mesh movement strategy was investigated. In addition, a structural mesh deformation methodology was proposed to move the internal structures consistently with the B-spline geometry parameterization. The requirements driving the design of the structural mesh deformation algorithm and a number of possible options were discussed. Suitability of the surface-based FFD method was demonstrated by successfully warping the structural model from a planar wing to a C-wing geometry without distorting the structural components. The calculation of geometric gradients for aerodynamic shape optimization was also reviewed to facilitate the discussions of sensitivity calculations in the context of aerostructural optimization.
Chapter 5

Aerostructural Analysis and Coupled Adjoint Formulation

This chapter proposes a new methodology for aerostructural optimization that features the use of the integrated geometry parameterization and mesh movement algorithm described in Chapter 4. The present methodology is constructed over existing flow and structural modules introduced in Chapter 3. It will be shown that the B-spline mesh movement presents an effective alternative to existing methodologies [26, 30] in accounting for shape changes due to structural deflections, in addition to those due to optimization. Sections 5.1 and 5.2 outline the aerostructural analysis and coupled adjoint formulation, respectively. In this chapter, the steady-state solution to the coupled analysis problem is obtained via a nonlinear block Gauss-Seidel method. Likewise, a linear block Gauss-Seidel method is used for the coupled adjoint solution. The partitioned method described here provides a baseline for comparing with the monolithic solution method in Chapter 6. Using the partitioned method, a number of studies have been conducted on the accuracy of the new methodology. More specifically, the suitability of the integrated mesh movement for the purpose of aerostructural analysis is thoroughly investigated in Sections 5.1.4 and 5.1.5. Accuracy of the gradient calculation is further verified in Section 5.2.3.

5.1 Steady-State Aerostructural Analysis

The present methodology adopts a three-field formulation of the discrete steady aerostructural residual, \( \mathbf{R}_{AS} \). The mesh equations, \( \mathbf{R}_{M\Delta} \), appear explicitly in \( \mathbf{R}_{AS} \), along with the aerodynamic residual, \( \mathbf{R}_A \), and the structural residual, \( \mathbf{R}_S \):

\[
\mathbf{R}_{AS} = \begin{bmatrix}
\mathbf{R}_A(q, b_\Delta) \\
\mathbf{R}_{M\Delta}(b_\Delta, u) \\
\mathbf{R}_S(q, b_\Delta, u)
\end{bmatrix} = 0. 
\] (5.1)

The aerodynamic equations govern the state of the flow, \( \mathbf{q} \). In a three-field formulation, \( \mathbf{R}_A \) does not explicitly depend on the structural state, \( \mathbf{u} \), but is instead a function of \( \mathbf{G} \), which in turn depends on the mesh state, \( b_\Delta \). The structural equations relate \( \mathbf{u} \) to the aerodynamic loading computed based on \( \mathbf{q} \) and \( b_\Delta \). Details on the flow and structural modules have been provided in Chapter 3. The mesh equations couple \( b_\Delta \), which describes the deformed control grid due to structural deflections, with \( \mathbf{u} \). A
Chapter 5. Aerostructural Analysis and Coupled Adjoint Formulation

Subscript \( \Delta \) has been added to the mesh equations and variables to distinguish them from those involved in moving the flow grid due to changes in the undeflected shape. An important distinction between the present methodology with others using a three-field formulation \[21, 26] is the use of B-spline control grid coordinates to describe the state of the deformed flow grid instead of the flow grid coordinates. This results in a much smaller mesh subproblem, which is another advantage to using the integrated approach for aerostructural optimization.

In the case where multiple increments are used to move the flow grid during analysis, \( R_{M\Delta} \) refers to the vector containing all the incremental mesh residuals, i.e. \( R_{M\Delta} = [R_{M\Delta}^{(1)}, R_{M\Delta}^{(2)}, \cdots, R_{M\Delta}^{(m\Delta)}]^T \). A similar notation is used for \( b_{\Delta} \) and \( b_{s\Delta} \), such that the control point coordinates for all increments are expressed as a single vector. The size of the mesh problem, \( n_{M\Delta} \), is therefore given by \( n_{M\Delta}^{(i)} \times m_{\Delta} \), where \( n_{M\Delta}^{(i)} \) is the size of the mesh movement problem at each increment. This new notation does more than simplify the discussions for the rest of this thesis. It also suggests a shift in perspective. Although the incremental mesh states have so far only been updated in a sequential manner, they can in fact be treated as parts of the same variable, in which case they can be modified simultaneously. This is an important concept to recognize when introducing the monolithic solution method in the next chapter.

A two-field formulation of \( R_{AS} \), which consists only of the aerodynamic and structural equations, has been used in a number of existing methodologies for aerostructural optimization \[29, 30, 126, 127\]. Appendix \[3\] shows how a two-field formulation can be recovered from (5.1). This exercise reveals a number of reasons why a three-field formulation is more appropriate for the present methodology. The two- and three-field formulations differ in that the former treats the flow grid as an intermediate variable. This leads to an explicit dependence of \( R_A \) on \( u \), or \( \partial R_A / \partial u \neq 0 \). A two-field formulation requires solving \( R_{M\Delta} = 0 \) for every change in \( u \). The evaluation of \( \partial R_A / \partial u \) or its transpose further requires the partial derivative of the flow grid, \( G \), with respect to \( u \). However, this term cannot be easily obtained for the linear elasticity mesh movement strategy, because the mesh nodes are implicitly coupled to the surface control points according to (4.7). Appendix \[3\] shows that in order to compute \( \partial G / \partial u \), one must invert \( \partial R_{M\Delta} / \partial b_{\Delta} \). Therefore, using a two-field formulation in the present methodology will have the following implications:

- The mesh problem needs to be solved to a tighter tolerance than necessary during each nonlinear block Gauss-Seidel iteration for the solution to (5.1).

- A number of terms in the Jacobian of \( R_{AS} \) and its transpose become more complex, as shown in (B.2). The need to invert \( \partial R_{M\Delta} / \partial b_{\Delta} \) further increases the cost to evaluate these terms.

- Each re-evaluation of \( R_{AS} \) requires a full mesh solution, which makes it more expensive to approximate the Jacobian matrix-vector products with \( R_{AS} \) using finite-differences.

The above can affect the complexity and efficiency of the partitioned and monolithic solution algorithms applied to the analysis and coupled adjoint problems. For these reasons, a three-field formulation is chosen for this work. The nonlinear block Gauss-Seidel solution procedure, as well as the sequence of operations that occur at each design iteration, are discussed in Section 5.1.1. Sections 5.1.2 and 5.1.3 then provide additional details on specific aspects of the present methodology. This includes a modified displacement transfer procedure and a strategy to move the flow grid more efficiently in response to shape changes due to both optimization and structural deflections.
5.1. Steady-State Aerostructural Analysis

5.1.1 Nonlinear Block Gauss-Seidel Iterations

Prior to an aerostructural analysis, the flow grid for the jig shape is obtained by solving the linear elasticity equations, \( R_{MJ} \), given \( b_{sJ}(v_G) \). The subscript \( J \) is added to \( R_M \) and \( b_s \) to differentiate from the mesh movement during analysis. Furthermore, the structural geometry is deformed using the surface-based FFD method from Section 4.4. The appropriate freestream conditions are assigned according to the aerodynamic design variables or \( v_A \). Structural stiffness is evaluated after updating the structural component thickness according to the structural design variable, \( v_S \), and the updated structural geometry. The flow, mesh, and structural state variables are then computed by solving (5.1) via a nonlinear block Gauss-Seidel method.

The solution procedure at the \( k \text{th} \) block Gauss-Seidel iteration is outlined below:

1. Transfer displacements:
   \[
   u^{(k-1)} \rightarrow u_A^{(k)} \rightarrow \Delta b_s^*(k),
   \]
   \[b_s^*(k) = b_{sJ} + \Delta b_s(k) \rightarrow b_{s\Delta};\]

2. Solve \( R_{M\Delta}(b_{s\Delta}(u), b_{\Delta}) = 0 \) to update \( b_{\Delta} \):
   \[b_{\Delta}^{(k)} \rightarrow b_{\Delta}^{(k)} \text{ s.t. } \|R_{M\Delta}(b_{s\Delta}^{(k)}, b_{\Delta}^{(k)})\|_2 \leq \epsilon_M\|R_{M\Delta}(b_{s\Delta}^{(k-1)}, b_{\Delta}^{(k-1)})\|_2;\]

3. Deform flow grid:
   \[b_{\Delta}^{(k)} \rightarrow G^{(k)};\]

4. Solve \( R_A(q, G(b_{\Delta})) = 0 \) to update \( q \):
   \[q^{(k-1)} \rightarrow q^{(k)} \text{ s.t. } \|R_A(q^{(k)}, G^{(k)})\|_2 \leq \epsilon_A\|R_A(q^{(k-1)}, G^{(k)})\|_2;\]

5. Transfer forces:
   \[q^{(k)}, G^{(k)} \rightarrow f_A^{(k)}, f_A^{(k)}, G^{(k)} \rightarrow f_S^{(k)};\]

6. Solve \( R_S(f_S(q, b_{\Delta}), u) = 0 \) to update \( u \):
   \[u^{(k-1)} \rightarrow \Delta u^{(k)} \text{ s.t. } \|R_S(f_S^{(k)}, u^{(k-1)} + \Delta u^{(k)})\|_2 \leq \epsilon_S\|R_S(f_S^{(k)}, u^{(k-1)})\|_2,\]
   \[u^{(k-1)} + \theta \Delta u^{(k)} \rightarrow u^{(k)}\]

The displacement transfer in step 1 above follows the rigid link method described in Section 3.2.3, which translates \( u \) into a vector of displacements, \( u_A \), for all nodes on the surface of the flow grid. The flow grid movement, however, requires the coordinates of the B-spline surface control points. Hence additional steps are required to obtain the changes in the B-spline surfaces, \( \Delta b_s^* \), from the jig shape, \( b_{sJ}^* \), to the deflected shape, \( b_{s\Delta}^* \). Recall that the superscript \( * \) is used to differentiate between the geometry definition via B-spline surface control points (\( b_{s\Delta}^{*} \)) and the boundary conditions for the mesh equations (\( b_{s\Delta} \)). The latter contains surface control point coordinates for all increments. The relationship between
\( \mathbf{b}_{\Delta} \) and \( \mathbf{b}_{\Delta}^* \) is given by (4.6). The modified displacement transfer procedure is discussed in Section 5.1.2. The force transfer in step 5 is unchanged from Section 3.2.3.

Steps 2, 4, and 6 involve subiterations within each of the mesh, flow, and structural subproblems using existing solution routines described previously. The under-relaxation factor, \( \theta \), in step 6 above is computed using Aitken acceleration [128, 129]. It is an adaptive under-relaxation algorithm which has been shown to improve the stability and convergence of the nonlinear block Gauss-Seidel method [30]. The parameters \( \epsilon_M, \epsilon_A \) and \( \epsilon_S \) determine how tightly each subproblem is solved, and are typically set to \( 10^{-6}, 10^{-1}, \) and \( 10^{-3} \), respectively. The stopping criterion for the nonlinear block Gauss-Seidel iterations examines the residual norm of individual equations to account for differences in scaling [25, 72]:

\[
\| \mathbf{R}_{\Delta}^{(k)}(\mathbf{u}^{(k-1)}, \mathbf{b}^{(k-1)}) \|_2 < \epsilon_{AS} \| \mathbf{R}_{\Delta}^{(0)}(\mathbf{u}^{(0)}, \mathbf{b}^{(0)}) \|_2, \quad (5.2)
\]

\[
\| \mathbf{R}_{A}(\mathbf{q}^{(k-1)}, \mathbf{b}^{(k)}) \|_2 < \epsilon_{AS} \| \mathbf{R}_{A}(\mathbf{q}^{(0)}, \mathbf{b}^{(0)}) \|_2, \quad (5.3)
\]

\[
\| \mathbf{R}_{S}(\mathbf{q}^{(k)}, \mathbf{u}^{(k-1)}, \mathbf{b}^{(k)}) \|_2 < \epsilon_{AS} \| \mathbf{R}_{S}(\mathbf{q}^{(0)}, \mathbf{u}^{(0)}, \mathbf{b}^{(0)}) \|_2. \quad (5.4)
\]

The relative tolerance for the aerostructural problem, \( \epsilon_{AS} \), is typically set to \( 10^{-8} \). Results obtained by aerostructural analysis are validated with experimental data from the HIRENASD project in Section 5.1.5.

The nonlinear block Gauss-Seidel method allows the flow, mesh, and structural modules to be integrated in a straightforward manner. Nonetheless, they may suffer from efficiency and robustness issues for strongly coupled problems [30, 51, 53, 54, 55]. The development of more advanced solution strategies for the present framework will be discussed in Chapter 6.

### 5.1.2 Modified Displacement Transfer

The displacement transfer in Section 3.2.3 results in a vector of displacements, \( \mathbf{u}_A \), for all nodes at the surface of the flow grid. This needs to be converted into the changes in the B-spline surfaces for the mesh movement algorithm described in Section 4.2. This is accomplished by fitting the discrete surface displacements using a linear least-squares method.

The fitting is performed independently on each B-spline surface patch. In order to ensure continuity at the boundaries of the B-spline patches, the corners of each patch are fitted first, followed by the edges, and finally the interior of the surface patch. This is equivalent to solving the following equations in the given order:

\[
\Delta \mathbf{b}_{sc}^* = \mathbf{u}_{Ac}, \quad (5.3)
\]

\[
N_i^T \mathbf{N}_{ie} \Delta \mathbf{b}_{ie}^* = N_i^T \mathbf{u}_{Ac} - N_i^T \mathbf{N}_{ie} \Delta \mathbf{b}_{sc}^*, \quad (5.4)
\]

\[
N_i^T \mathbf{N}_{ie} \Delta \mathbf{b}_{ie}^* = N_i^T \mathbf{u}_{Ai} - N_i^T \mathbf{N}_{ie} \Delta \mathbf{b}_{ec}^* - N_i^T \mathbf{N}_{ic} \Delta \mathbf{b}_{ic}^*, \quad (5.5)
\]

where \( \mathbf{u}_{Ac} \), \( \mathbf{u}_{Ac}^* \), and \( \mathbf{u}_{Ai} \) describe the discrete surface displacements at the corner, along the edge, and at the interior of the surface patch. Similarly, \( \Delta \mathbf{b}_{sc}^* \), \( \Delta \mathbf{b}_{se}^* \), and \( \Delta \mathbf{b}_{si}^* \) refer to the changes in the B-spline surface control points at the corners, edge, and interior of the surface patch. The corresponding coefficient matrices are given by \( \mathbf{N} \) with the appropriate subscripts. After the changes in the corner or edge control points are obtained, their effects are included in the RHS of the subsequent equations.

Equations (5.4) and (5.5) are the normal equations corresponding to two overdetermined problems. The above fitting procedure therefore computes the changes in the surface control point coordinates which best describe the structural deflections. Once \( \Delta \mathbf{b}_{sc}^* \) has been found for the entire surface patch, it
is added to the control point coordinates for the jig shape, i.e. \( b_{sJ}^{*} \) in step 1 of Section 5.1.1, to obtain the surface control point coordinates, \( b_{s\Delta}^{*} \), for the deflected geometry. However, due to the error in fitting, the surface grid coordinates described by \( b_{s\Delta}^{*} \) are not the same as those obtained by adding \( u_{A} \) to the jig shape. Some possible implications of this error on the accuracy of the analysis are investigated in Section 5.1.4. On the other hand, the grid smoothing introduced by the fitting has improved the convergence of the analysis in some cases, so it may in fact be desirable. Continued investigations are necessary in the future to draw a definite conclusion.

For the coupled adjoint calculations in Section 5.2, it is further necessary to compute

\[
\left( \frac{\partial b_{s\Delta}^{*}}{\partial u_{A}} \right)^{T} w = \left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{A}} \right)^{T} w, \tag{5.6}
\]

where \( w \in \mathbb{R}^{nMs} \) is a vector used for the matrix-vector product calculation. Upon splitting \( w \) for each patch into three vectors: \( w_{c} \), \( w_{e} \), and \( w_{i} \), the matrix-vector product can be rewritten as follows

\[
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{A}} \right)^{T} w = \begin{bmatrix}
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{i} + \left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{c} \\
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{e} \\
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{i}
\end{bmatrix} \tag{5.7}
\]

The above is computed analytically by differentiating (5.3) to (5.5). The derivative terms with respect to \( u_{Ac} \) are the most involved. Hence they are provided here for the purpose of illustration:

\[
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{e} = w_{e} \tag{5.8}
\]

\[
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{c} = -N_{sc}^{T}(N_{sc}^{T}N_{sc})^{-1}N_{sc}w_{c} \tag{5.9}
\]

\[
\left( \frac{\partial \Delta b_{s\Delta}^{*}}{\partial u_{Ac}} \right)^{T} w_{i} = -N_{sc}^{T}(N_{sc}^{T}N_{sc})^{-1}N_{sc}w_{i} + N_{sc}^{T}N_{sc}(N_{sc}^{T}N_{sc})^{-1}N_{sc}(N_{sc}^{T}N_{sc})^{-1}N_{sc}w_{i}. \tag{5.10}
\]

The above terms require the matrix-vector products with the inverse of a number of normal matrices, for which Cholesky factorization is used. The remaining derivative terms in (5.7) are computed in a similar fashion.

### 5.1.3 Flow Grid Movement for Aerostructural Optimization

It was pointed out in Section 5.1.1 that a set of mesh movement equations, \( R_{MJJ} \), is solved prior to an aerostructural analysis, while another set of mesh equations, \( R_{M\Delta} \), is coupled to the analysis. Aerostructural optimization is unlike aerodynamic shape optimization in that changes in the geometry are the combined result of changes due to optimization and structural deflections. Movement of the control grid in all design iterations must start with the initial control grid to ensure the smoothness of the gradient. As a result, shape changes due to optimization are always measured from the initial geometry, while those due to structural deflections are measured from the undeflected or jig shape at every design...
Equation (5.11) describes a mesh movement operation with \( m \) increments. The control point coordinates, before an analysis, are chosen by solving (4.7) rewritten in the following form:

\[
R^{(i)}_{\Delta J}(b^{(i)}_j - b^{(i-1)}_j) - f^{(i)}_{\Delta J}(b^{(i)}_j - P_s b^{(i-1)}_j) = 0, \quad i = 1, \cdots, m_J.
\]

Equation (5.11) describes a mesh movement operation with \( m_J \) increments starting from the initial geometry. The control point coordinates, \( b_j^{(0)} \), in (5.11) and in Figure 5.1 are the same as those in \( b_j^{(0)} \) for method 1. Another set of mesh movement equations,

\[
R^{(i)}_{\Delta J}(b^{(i-1)}_\Delta - b^{(i-1)}_\Delta) = K^{(i)}_{\Delta J}(b^{(i-1)}_\Delta - b^{(i-1)}_\Delta) - f^{(i)}_{\Delta J}(b^{(i)}_\Delta - P_s b^{(i-1)}_\Delta) = 0, \quad i = 1, \cdots, m_\Delta,
\]

is coupled to the aerostructural analysis using a second \( m_\Delta \) set of increments to obtain the grid for the deflected shape, starting from the grid for the jig shape. The two sets of mesh movement equations, \( R_{\Delta J} \) and \( R_{\Delta J} \), are related by setting \( b^{(0)}_\Delta = b^{(m_J)}_\Delta \) and \( b^{(0)}_\Delta = b^{(m_J)}_\Delta \). Using this strategy, mesh movement during analysis only needs to accommodate shape changes due to structural deflections. This allows for a relatively small value of \( m_\Delta \) that is independent of the value of \( m_J \) needed to accommodate the changes in the jig shape. In other words, larger values of \( m_J \) can be chosen to ensure the grid quality without incurring a significant cost penalty in the aerostructural calculations. This is particularly important in exploratory optimization where large shape changes are often present.

Figure 5.1: Two ways to accomplish shape changes due to optimization and structural deflections using the linear elasticity mesh movement algorithm. The control grid coordinates corresponding to each geometry are also included.

Figure 5.1 illustrates two methods for mesh movement. Method 1 is depicted in (a) and Method 2 in (b). Figure 5.1 is important in exploratory optimization where large shape changes are often present.
The advantage of method 2 is best illustrated by performing an aerostructural analysis on the C-wing geometry introduced in Figure 4.1 from the previous chapter. The analysis uses a Mach number of 0.785, an AOA of 0°, and assumes an altitude of 35,000 feet. All structural components in the wing have a thickness of 8mm. These parameters have been chosen to induce a realistic deflection at the tip of the wing, as shown in Figure 5.2 along with the computed distribution of the coefficient of pressure. Recall that the C-wing has been created from an initially planar wing by manually moving the B-spline surface control points. Due to the substantial amount of shape change, the flow grid for the jig shape is obtained using 15 increments to ensure the resulting grid quality. The use of method 1 hence requires 15 mesh solutions per nonlinear block Gauss-Seidel iteration, and 15 iterations were required to converge the coupled analysis problem to a relative tolerance of $5 \times 10^{-9}$. This translates to a total of 225 mesh solutions for one analysis. Using method 2 with an $m_\Delta$ of 1, the analysis converged in 14 block Gauss-Seidel iterations, requiring a total of 14 mesh solutions. The total mesh movement time is substantially reduced from being 45% of the total analysis time for method 1 to only 7.5% using method 2.

Figures 5.3 and 5.4 examine the quality of the initial grid, the grid of the C-wing, and the final grid at the end of analysis using methods 1 and 2. The fraction of grid cells with different cell aspect ratios and orthogonality measures are plotted and compared between the different grids. The cell aspect ratio here is the ratio between the longest and the shortest edges of the cell. The cell orthogonality measure is the smallest of the orthogonality values computed at the eight vertices of the cell. The grid aspect ratio distribution has not changed significantly from the initial grid in all cases. Although the orthogonality distributions for the undeflected and deflected C-wing geometries are shifted to the left compared to the initial geometry, there is not a large increase in the number of cells with low orthogonality values. Past experience has shown that the changes in grid quality have little impact on the conclusions of the optimization. Figures 5.3 and 5.4 therefore provide some evidence that the mesh movement algorithm offers sufficient robustness for the analysis of novel designs. The new mesh movement procedure introduced in method 2 additionally allows the aerostructural analysis to be performed efficiently despite substantial

\[ ^1 \text{Used with permission from [102]} \]
changes in the undeflected geometry.

Method 2 is advantageous when shape changes due to optimization are much larger and more complex than the shape changes due to structural deflections. However, method 1 may still be useful if the two types of shape changes are expected to be on par in their magnitudes. The remaining discussions in this thesis will assume that method 2 is used unless otherwise stated. Any discussions related to coupled aerostructural solution strategies are applicable to both methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of aerostructural solution strategies are applicable to both methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods. During gradient calculations, however, it will be shown in Section 5.2 that method 2 requires the solution to an additional set of methods.

For the gradient calculations presented in Section 5.2 it is also helpful to describe the partial derivative of $R_{M\Delta}$ with respect to $b_j^{(m)}$ for method 2. This term is needed in the form of a transposed matrix-vector product. Being equal to $b_j^{(0)}$, $b_j^{(m)}$ influences $R_{M\Delta}$ in the same way control points from one increment influence the mesh equation in the subsequent increment. This can be easily computed using existing routines for (4.34). Moreover, $b_j^{(m)}$ influences the implicit force vector for $R_{M\Delta}$ at every increment because the surface displacements are measured from the control point coordinates at the zeroth increment. In other words, the implicit force vector in $R_{M\Delta}$ is calculated as follows:

$$f_{M\Delta}^{(i)} = -K_{M\Delta}(i)^T P_s \left[ \frac{i}{m\Delta} b_{\Delta}^* + \left( 1 - \frac{i}{m\Delta} \right) P_s b_j^{(m)} - P_s b_j^{(i-1)} \right].$$

This leads to an extra term which must be included in $\partial R_{M\Delta}/\partial b_j^{(m)}$, as given below:

$$\left( \frac{\partial f_{M\Delta}^{(i)}}{\partial b_j^{(m)}} \bigg|_{b_j^{(i-1)}} \right)^T \Psi_{M\Delta}^{(i)} = \left( 1 - \frac{i}{m\Delta} \right) P_s^T P_s K_{M\Delta}(i)^T \Psi_{M\Delta}^{(i)}.$$

The mesh adjoint variable, $\Psi_{M\Delta}^{(i)}$, for the $i^{th}$ increment will be introduced in Section 5.2.

### 5.1.4 Force and Energy Conservation

The present analysis methodology introduces an additional fitting step to the displacement transfer, so that the B-spline surface control point coordinates for the deflected shape can be determined from
the structural state. The transfer calculations therefore do not guarantee that the work done at the aerostructural interface is conserved. Many authors in the literature believe that such a conservation property is important [100, 130, 131, 132]. This is achieved by making the force transfer operator the transpose of the displacement transfer operator [131, 133]. However, the use of such operators can introduce unphysical oscillations in the interface solution, which can in turn reduce the overall accuracy of the analysis [131, 132]. Although energy conservation is important, it is unclear whether it should be the sole criterion for assessing a transfer scheme. Fitting the deflected surface with B-spline control points smooths the oscillations on the deflected geometry which have been observed without fitting. This has improved the solution convergence in some cases. Nonetheless, it is still important to investigate whether the error due to fitting has any negative implications in the context of aerostructural optimization.

The applications of relevance to this thesis often involve wingbox geometries with physical gaps between the aerodynamic and structural domains. It is difficult to define an “exact” transfer of forces and displacements with reference to which a direct error measure can be obtained. To indirectly assess the effects due to fitting, this study compares the results using the present framework with those using an earlier framework described in Appendix A. This earlier framework has been constructed over the same aerodynamics and structural modules, as well as the same transfer scheme. However, an algebraic mesh movement is used to move the flow grid during aerostructural analysis, such that no fitting is required. The rest of this section will refer to this framework as the second framework. This investigation was done in collaboration with Shahriar Khosravi, who also created the figures presented here.

This study is designed with the consideration that a post-optimality analysis is often performed subsequent to an optimization, which involves analyzing the optimized design with increased flow grid resolution. The post-optimality analysis ensures that the primary conclusions from the optimization, which usually uses a flow grid of lower resolution, are independent of flow grid refinement. The fitting error is unacceptable if it leads to inconsistent conclusions on the coarse optimization and fine analysis grids. Therefore, the objective of the present investigation is to establish that the fitting errors introduced during the displacement transfer does not affect the grid convergence during post-optimality analysis. More specifically, values of important aerodynamic functionals computed using the present methodology should become independent of the grid resolution to within a small tolerance.

With this in mind, two optimized geometries are obtained using the present framework and the second framework, respectively. The optimization capability of the present framework has not yet been fully
described, but the remaining details are not relevant to the discussions here. Separate grid refinement studies are conducted for each framework on the corresponding optimized geometry, and the results will be compared. The planform of the initial geometry and the structural layout are illustrated in Figures 5.5 and 5.6. The initial planform is based on the Boeing 737-900 wing with RAE 2822 airfoil sections. It is parameterized with 5 patches in the spanwise direction and 2 in the chordwise direction. Each surface patch is controlled by $6 \times 6$ control points shown as red spheres in Figure 5.5. The optimizer is free to change the sectional shape and twist of the wing. The thickness distribution in the structural components is also optimized. The optimization minimizes the drag at a 1g load condition. The optimized design must also satisfy the stress constraints defined using the KS functions from Section 3.2.4, which are evaluated at a 2.5g load condition to size the structural components.

Results of the post-optimality grid convergence studies using the optimized designs from the two frameworks are shown in Figure 5.7. Figures 5.7a and 5.7b show the convergence of the lift coefficient and the drag coefficient in drag counts ($10^{-4}$) with the present framework, where $N$ on the $x$-axis refers to the total number of flow grid nodes. The number of control points per patch in the chordwise direction

---

2 Used with permission from [135]
5.1. Steady-State Aerostructural Analysis

is also varied for the present framework to examine the effects of improving the accuracy of fitting. The sequence of aerodynamic grids is obtained by doubling the total number of grid nodes at every level. This can be conveniently accomplished using the B-spline volume parameterization of the flow grid. Figures 5.7c and 5.7d show the convergence of lift and drag coefficients for the second framework. It can be observed that the results from the two frameworks exhibit the same convergence trends. In all cases, the values of both functionals change by less than 1% from the second finest to the finest grid level. This means that the fitting error introduced during the transfer process does not hinder the grid convergence of the lift and drag values.

To further support the above observations, the total force acting as well as the work done by the displacement field on either side of the aerostructural interface are compared for each data point from the functional convergence study. The differences in the forces and the work done at the interface are calculated at steady-state. These differences indicate the amount of inconsistency, or the error in conservation, introduced by the transfer procedure. Results from the two frameworks are again compared. Figure 5.8a plots the error in the force transfer for both frameworks. Figure 5.8b shows a similar plot for the error in the work done at the aerostructural interface. The error in the force transfer is not a consequence of fitting. It can be attributed to the differences between the force integration schemes used in the aerodynamic and structural solvers. Hence this error is present in both frameworks. It shows a second-order convergence rate consistent with that of the surface integration scheme. The error also appears to be independent of the number of surface control points. The error in Figure 5.8b is the combined result of the differences shown in Figure 5.8a and the error in the displacement transfer due to surface fitting. Although the error values in both figures are not zero, they are small and can be further reduced with refinement of the flow grid or the control grid. Most importantly, the fitting procedure in the present framework does not appear to contribute a substantial amount of error to the force and energy calculation in comparison to the second framework. It can then be concluded that the fitting process has a minor impact on the accuracy of the transfer scheme.

5.1.5 Validation Based on the HIRENASD Wing

The objective of this section is to validate the aerostructural analysis against the static experimental data from the HIgh REynolds Number Aero-Structural Dynamics (HIRENASD) project [136, 137, 138].
This is necessary to show that the correct physics are captured by the present methodology, especially given the presence of the fitting error. This study has also been conducted in collaboration with Shahriar Khosravi.

Aerostructural analysis with the RANS equations is needed to fully capture the flow physics from the experiment. The flow module has the capability to perform viscous and turbulent analysis, where the turbulence is modelled with the one-equation Spalart-Allmaras model \[87\]. The use of a nonlinear block Gauss-Seidel method allows the RANS aerostructural analysis to be performed using the same procedure described in Section 5.1.1. Viscous effects are accounted for simply by replacing the inviscid flow analysis in step 4 with the appropriate routines for RANS analysis. This is an advantage to using a partitioned solution strategy for the development of new analysis and optimization capabilities. Another necessary modification involves the calculation of the surface traction, \( f_A \), to include the viscous effects. The viscous surface traction is computed in the same manner as the viscous force vector acting on each node, details of which are given in \[139\]. The resulting viscous force vector at each node must be normalized by the surface area defined at the same node and scaled by \( q_\infty \) to obtain the surface tractions in dimensional units.

The Mach number for the chosen experiment is 0.80, the AOA is 1.5°. The Reynolds number, which is based on the mean aerodynamic chord, is \( 7.0 \times 10^6 \). The computational grid for the flow domain has a total of 3,548,095 nodes, with an average \( y^+ \) value of 0.24. The structural model provided by the HIRENASD project for the purpose of validation extends to the leading and trailing edges of the wing and consists of solid elements. Since the present framework primarily works with wingbox models using shell elements, the structural model has been recreated for the numerical analysis using 38,000 second-order MITC shell elements. The placement and thickness of the structural components have been determined based on careful measurements taken from the original structure of the HIRENASD wing to approximate it as closely as possible.

Figure 5.9 shows the pressure coefficient distribution computed using the present methodology. In addition, the chordwise variation in the pressure coefficient is included for two spanwise stations near the tip of the wing, superimposed on the corresponding experimental measurements. Numerical results from pure aerodynamic analysis assuming no structural deflections are also shown as a reference. There are noticeable deviations between results obtained using pure aerodynamic analysis and those computed using the coupled aerostructural analysis capability, and the latter consistently demonstrate better agreement with the experimental data. Furthermore, the computed tip deflection is 12.6mm, which is very close to the measured value of 12.5mm reported for the experiment \[140\]. The results of this study provide some evidence that the analysis methodology is able to make accurate predictions by capturing the correct physics.

### 5.2 Gradient Calculations by the Coupled Adjoint Method

Gradient-based optimization requires the derivatives of all functionals of interest with respect to the design variables. This chapter describes a discrete coupled adjoint approach for gradient calculations. For each functional that depends on the state variables, which may include the objective function or other nonlinear constraints, the gradients are computed by solving a coupled adjoint problem. The cost of gradient calculations using the adjoint method is nearly independent of the number of design variables. It is therefore appropriate for optimization problems where the number of design variables
5.2. Gradient Calculations by the Coupled Adjoint Method

is much greater than the number of functionals [13, 14]. This is often the case in the applications of interest to this thesis, due to the agglomeration of constraints. The discrete coupled adjoint problem and the total gradient expression are derived for the present methodology in Section 5.2.1. It is followed by Sections 5.2.2 where the calculation of the required partial derivative terms are discussed. The gradient calculation is verified with finite-difference approximations in Section 5.2.3.

5.2.1 The Discrete Coupled Adjoint Problem

The coupled adjoint formulation has been previously described by a number of authors [25, 29]. It is presented here for the proposed aerostructural optimization methodology using the method of Lagrange multipliers adopted by Hicken and Zingg [44] for aerodynamic optimization. Consider the optimization of a functional \( J \), subject to \( R_{AS} = 0 \) and \( R_{MJ} = 0 \). The Lagrangian function for this problem is

\[
\mathcal{L} = J(v, b_J, [q, b_\Delta, u]^T) + \Psi_{MJ}^T R_{MJ}(v, b_J) + \left[ \Psi_A^T \Psi_{M\Delta}^T \Psi_S^T \right] \begin{bmatrix} R_A(v, b_J, [q, b_\Delta]^T) \\ R_{M\Delta}(v, b_J, [b_\Delta, u]^T) \\ R_S(v, b_J, [q, b_\Delta, u]^T) \end{bmatrix}, \tag{5.15}
\]

where \( \Psi_{MJ}, \Psi_A, \Psi_{M\Delta} \) and \( \Psi_S \) are the Lagrange multipliers. The aerostructural equations and state variables are expanded into \( R_{AS} = [R_A, R_{M\Delta}, R_S]^T \) and \( [q, b_\Delta, u]^T \), respectively, so that individual derivative terms can be discussed. However, brackets are kept around the equations and variables to serve as a reminder that they are parts of the same term. The first-order optimality conditions require

\[\text{fig:adjoint}\]
that the partial derivatives of \( \mathcal{L} \) with respect to \( \mathbf{b}_j \) and \( [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T \) be zero:

\[
\frac{\partial \mathcal{L}}{\partial \mathbf{b}_j} = 0 \Rightarrow \begin{bmatrix} \Psi_{M\Delta}^T & \Psi_{S}^T \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \mathbf{b}_j} \mathbf{R}_A \\ \frac{\partial}{\partial \mathbf{b}_j} \mathbf{R}_{M\Delta} \\ \frac{\partial}{\partial \mathbf{b}_j} \mathbf{R}_S \end{bmatrix} = -\frac{\partial \mathcal{J}}{\partial \mathbf{b}_j},
\]

\[
\frac{\partial \mathcal{L}}{\partial [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T} = 0 \Rightarrow \begin{bmatrix} \Psi_A^T & \Psi_{M\Delta}^T & \Psi_S^T \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T} \mathbf{R}_A \\ \frac{\partial}{\partial [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T} \mathbf{R}_{M\Delta} \\ \frac{\partial}{\partial [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T} \mathbf{R}_S \end{bmatrix} = -\frac{\partial \mathcal{J}}{\partial [\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}]^T}.
\]

Taking the transpose of the above equations leads to the following linear system:

\[
\begin{bmatrix}
\frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_j} & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{q}} & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} & \frac{\partial \mathbf{R}_S}{\partial \mathbf{u}} \\
\frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{q}} & 0 & \frac{\partial \mathbf{R}_S}{\partial \mathbf{u}} & 0 \\
\frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} & \frac{\partial \mathbf{R}_S}{\partial \mathbf{u}} & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Psi_{M\Delta}^T \\
0 \\
\Psi_{M\Delta}^T \\
0
\end{bmatrix}
= \begin{bmatrix}
-\frac{\partial \mathcal{J}}{\partial \mathbf{b}_j}^T \\
0 \\
-\frac{\partial \mathcal{J}}{\partial \mathbf{q}}^T \\
0
\end{bmatrix}
\]

Some equations do not depend explicitly on all variables, resulting in the zero entries in (5.17). The derivation is completed by taking the partial derivative of \( \mathcal{L} \) with respect to \( \mathbf{v} \):

\[
\mathcal{G} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}} = \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{v}} \Psi_{M\Delta}^T + \frac{\partial \mathbf{R}_A}{\partial \mathbf{v}} \Psi_A^T + \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{v}} \Psi_{M\Delta}^T + \frac{\partial \mathbf{R}_S}{\partial \mathbf{v}} \Psi_S^T.
\]

Equation (5.18) is the expression for the gradient of \( \mathcal{J} \) with respect to the design variables. The Lagrange multipliers here are the adjoint variables. The coupled adjoint equations refer to the block 3 \( \times \) 3 system in (5.17), where \([\Psi_A, \Psi_{M\Delta}, \Psi_S]^T\) are coupled by the transposed Jacobian of \( \mathbf{R}_{AS} \) on the LHS. The coupled adjoint problem is augmented by the mesh adjoint equations for the jig shape in the first row of (5.17). The coupled adjoint approach here in fact has a similar form as the discrete adjoint approach described in Section 4.5 for aerodynamic shape optimization. This can be observed by replacing \( \mathbf{R}_A \) and \( \mathbf{q} \) in (4.29–4.35) by \( \mathbf{R}_{AS} \) and \([\mathbf{q}, \mathbf{b}_\Delta, \mathbf{u}] \). Using this approach, the gradients are calculated by solving (5.17) for all adjoint variables and subsequently evaluating (5.18).

The solution to the coupled adjoint problem in (5.17) can be obtained in a relatively straightforward manner using a linear block Gauss-Seidel method. Each block Gauss-Seidel iteration solves the following equations in sequence:

\[
\frac{\partial \mathbf{R}_A}{\partial \mathbf{q}} \Psi_A^{(k+1)} = -\frac{\partial \mathcal{J}}{\partial \mathbf{q}}^T - \frac{\partial \mathbf{R}_S}{\partial \mathbf{q}} \Psi_S^{(k)},
\]

\[
\frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} \Psi_{M\Delta}^{(k+1)} = -\frac{\partial \mathcal{J}}{\partial \mathbf{b}_\Delta}^T - \frac{\partial \mathbf{R}_A}{\partial \mathbf{b}_\Delta} \Psi_A^{(k+1)} - \frac{\partial \mathbf{R}_S}{\partial \mathbf{b}_\Delta} \Psi_S^{(k)},
\]

\[
\mathbf{K}_S \Psi_S^{(k+1)} = -\frac{\partial \mathcal{J}}{\partial \mathbf{u}}^T - \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{u}} \Psi_{M\Delta}^{(k+1)}
\]

where \( k \) is the iteration index. The iterations are repeated until the residual norms of all equations
5.2. Gradient Calculations by the Coupled Adjoint Method

Drop below a specified tolerance relative to their initial values. A relaxation factor is often necessary in the structural adjoint equation, (5.21), to ensure the robustness of the partitioned calculations. It will be shown in later chapters that this is a disadvantage of the block Gauss-Seidel method compared to the monolithic method. The coupled adjoint problem is typically solved to a tolerance of $10^{-9}$ during optimization.

The LHS to (5.19) and (5.21) are the same as the LHS to the adjoint equations for purely aerodynamic and purely structural optimization, respectively, which have been described previously in Chapter 3. The LHS to (5.20) is also part of the augmented adjoint system in Section 4.5 for the geometric gradient calculations during aerodynamic shape optimization. This means that each of the above adjoint equations can be solved using existing iterative solution procedures in the flow, mesh, and structural modules. However, it is only necessary to partially converge each adjoint equation during a block Gauss-Seidel iteration. Typically, the flow adjoint equation is solved to a relative tolerance of $10^{-3}$, although it is possible to relax the tolerance to $10^{-1}$ in some cases. The mesh and structural adjoint equations are solved to a relative tolerance of $10^{-6}$ and $10^{-3}$, respectively. The different tolerances for each adjoint equations are chosen to optimize the performance of the block Gauss-Seidel method. Tolerances which are too large could compromise the stability of the partitioned iterations, while tolerances which are too small increase the computational cost without necessarily improving the coupled adjoint convergence. Finally, the RHS to (5.19–5.21) includes cross-coupling terms from the off-diagonal blocks in the transposed coupled Jacobian matrix. The calculation of these derivative terms are discussed in the next section.

5.2.2 Calculation of the Partial Derivative Terms

Calculation of the partial derivative terms in (5.17) and (5.18) follows the work of Kenway et al. [30] for a two-field formulation. Appropriate modifications have been introduced for the present methodology due the use of an augmented three-field coupled adjoint approach. The relationship between the derivative terms described in this section and those in a two-field formulation is further illustrated in Appendix B.

Before proceeding with this discussion, it is helpful to first distinguish between aerodynamic functionals, denoted by $J_A$, and structural functionals, denoted by $J_S$. An aerodynamic functional, such as lift and drag, is computed in the flow module and it has no explicit dependence on the structural design variables. In a three-field formulation, it also has no explicit dependence on the structural state. Likewise, structural functionals, such as mass and the KS functions, are computed in the structural module and they have no explicit dependence on the aerodynamic state or design variables. Composite functionals, such as range, can be written in terms of pure aerodynamic and structural functionals, so they are not discussed separately.

The flow adjoint equation in (5.19) is the same between a two-field and a three-field formulation. The coupling term in the flow adjoint equation is calculated by applying the chain rule [30]:

$$\frac{\partial R_S}{\partial q}^T \Psi_S = \frac{\partial f_A}{\partial q}^T \frac{\partial f_S}{\partial f_A} \frac{\partial R_S}{\partial f_S}^T \Psi_S, \quad (5.22)$$

where $\mathbb{I} \in \mathbb{R}^{ns \times ns}$ is an identity matrix. TACS computes $(\partial f_S/\partial f_A)^T \Psi_S$ by analytical differentiation of the force transfer routine, where $f_A$ is the surface traction computed by the flow solver at the start of a force transfer. The matrix-vector product with $(\partial f_A/\partial q)^T$ is then calculated in the flow module, as discussed in Section 3.1.4. For an aerodynamic functional, $(\partial J_A/\partial q)^T$ on the RHS of (5.19) can use
existing routines in the flow module for (3.28). Structural functionals are not explicit functions of \(q\), hence \((\partial J_S/\partial q)^T = 0\).

For the mesh adjoint equation in (5.20), the flow grid does not contribute directly to the calculation of \(J_S\), so \((\partial J_S/\partial b_\Delta)^T = 0\). Furthermore, the flow grid, \(G\), which is involved in the calculation of \(R_A\) and the force calculations in \(R_S\), does not depend on the control grid coordinates in the intermediate mesh movement increments. As a result, all partial derivative terms on the RHS of (5.20) are only nonzero with respect to \(b_\Delta^{(m\Delta)}\). Equation (5.20) is solved as a system of \(m\Delta\) equations in a similar manner as in (4.30) and (4.31) for aerodynamic shape optimization:

\[
\begin{align*}
\frac{\partial R_{M\Delta}^{(m\Delta)}}{\partial b_\Delta^{(m\Delta)}} T \Psi_{(m\Delta)}^{(i\Delta)} & = - \frac{\partial J_A}{\partial b_\Delta^{(m\Delta)}} T \Psi_A - \frac{\partial R_S}{\partial b_\Delta^{(m\Delta)}} T \Psi_S, & i = m\Delta, \\
& = - \frac{\partial G}{\partial b_\Delta^{(m\Delta)}} T \left[ \frac{\partial J_A}{\partial G} T + \frac{\partial R_A}{\partial G} T \Psi_A + \frac{\partial R_S}{\partial G} T \Psi_S \right] T, \\
\frac{\partial R_{M\Delta}^{(i\Delta)}}{\partial b_\Delta^{(i\Delta)}} T \Psi_{(i\Delta)}^{(m\Delta)} & = - \left( \frac{\partial R_{M\Delta}^{(i\Delta)}}{\partial b_\Delta^{(m\Delta)}} \right)^T \Psi_{(i\Delta)}^{(m\Delta)}, & i = m\Delta - 1, \ldots, 1.
\end{align*}
\]

The above equations can reuse the same solution routines for (4.30) and (4.31), as well as the evaluation of \((\partial J_A/\partial G)^T\), \((\partial R_A/\partial G)^T \Psi_A\), \((\partial R_{M\Delta}^{(i\Delta)}/\partial b_\Delta^{(i\Delta)}) T \Psi_{(i\Delta)}^{(m\Delta)}\), and the matrix-vector product with \((\partial G/\partial b_\Delta^{(m\Delta)}) T\). The cross coupling term related to the structural residual on the RHS of (5.23) is evaluated as follows:

\[
\frac{\partial R_S}{\partial G} T \Psi_S = - \left[ \left( \frac{\partial f_S}{\partial G} \right)_{f_A} T \Psi_S + \frac{\partial f_A}{\partial G} T \frac{\partial f_A}{\partial f_A} T \Psi_S \right].
\]

The first term in the square bracket is due to the force integration in TACS over the discrete deflected surface geometry. The second term accounts for the contribution of \(G\) in the surface traction calculation via the grid metric. The partial derivative of the aerodynamic surface traction with respect to the flow grid, \(\partial f_A/\partial G\), is evaluated as discussed in Section 3.1.4. This term is then given to TACS, which completes the calculation for (5.25).

For the RHS in the structural adjoint equation, the partial derivative of \(J_S\) with respect to \(u\) is unchanged from purely structural optimization (see (3.45)). Due to the presence of an explicit mesh state in a three-field formulation, changes in \(u\) do not affect the aerodynamic functionals in an explicit way. Hence \((\partial J_A/\partial u)^T\) is zero in (5.21). Contributions of the following forces in the structural Jacobian, i.e. \((\partial f_S/\partial u)^T (\partial R_S/\partial f_S)^T \Psi_S^{(k)}\), are also zero for the same reason. The effects of \(u\) on the mesh movement are included in

\[
\frac{\partial R_{M\Delta}}{\partial u} T \Psi_{M\Delta} = \frac{\partial u_A}{\partial u} T \frac{\partial b_\Delta}{\partial u_A} T \frac{\partial R_{M\Delta}}{\partial b_\Delta} T \Psi_{M\Delta}.
\]

The term \((\partial R_{M\Delta}/\partial b_\Delta)^T \Psi_{M\Delta}\) can be calculated using existing routines in the flow module developed for (4.36). The matrix-vector product \((\partial b_\Delta/\partial u_A)^T\) is related to fitting the deflected geometry. The evaluation of this term can be found in Section 5.1.2. The left-most term, \((\partial u_A/\partial u)^T\), depends on the displacement transfer scheme and is computed analytically in TACS [73].

Once the solution to the coupled adjoint problem is found, \(\Psi_A\), \(\Psi_{M\Delta}\), and \(\Psi_S\) are substituted into
the mesh adjoint equation for the jig shape. This results in the following expression:

$$\frac{\partial \mathbf{R}_{M_J}}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_{M_J} = - \frac{\partial \mathcal{J}}{\partial \mathbf{b}_j}^T - \frac{\partial \mathbf{R}_A}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_A - \frac{\partial \mathbf{R}_{M_D}}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_{M_D} - \frac{\partial \mathbf{R}_S}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_S. \quad (5.27)$$

Recall that there are two ways to apply the flow grid movement algorithm, as illustrated in Figure 5.1. If method 1 is used, it can be deduced that $\boldsymbol{\Psi}_{M_J} = 0$ by recognizing that $\partial(\cdot)/\partial \mathbf{b}_j = \partial(\cdot)/\partial \mathbf{b}_\Delta$. Hence by solving (5.20), which is the mesh adjoint equation for the deflected shape, the RHS to (5.27) sums to zero. This also means that the use of method 2, which involves using two sets of increments to accomplish changes in the jig shape and the deflected shape, requires solving an additional set of mesh adjoint equations for the jig shape. However, the associated increase in computational time is usually much lower than the amount saved during coupled analysis and adjoint calculations from choosing method 2 instead of method 1.

In method 2, control grid coordinates for the jig shape, $\mathbf{b}_j$, only appear in $\mathbf{R}_{M_D}$ and do not contribute to any other parts of the analysis or functional evaluations. This observation reduces (5.27) to

$$\frac{\partial \mathbf{R}_{M_J}}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_{M_J} = - \frac{\partial \mathbf{R}_{M_D}}{\partial \mathbf{b}_j}^T \boldsymbol{\Psi}_{M_D}. \quad (5.28)$$

Furthermore, only control grid coordinates in the $m_J$th increment of $\mathbf{R}_{M_J}$ are involved in the calculation of the stiffness, $\mathbf{K}_{M_D}$, and the implicit force vector, $\mathbf{f}_{M_D}$, in $\mathbf{R}_{M_D}$. Equation (5.28) therefore translates to the following $m_J$ adjoint equations:

$$\left(\frac{\partial \mathbf{R}_{M_J}}{\partial \mathbf{b}_j^{(m_J)}}\right)^T \boldsymbol{\Psi}_{M_J}^{(m_J)} = - \left(\frac{\partial \mathbf{R}_{M_D}}{\partial \mathbf{b}_j^{(m_J)}} \mathbf{f}_{M_D}^{(1)}\right)^T \boldsymbol{\Psi}_{M_D}^{(1)} + \sum_{j=1}^{m_D} \left(\frac{\partial \mathbf{f}_{M_D}}{\partial \mathbf{b}_j^{(m_J)}}\right)^T \boldsymbol{\Psi}_{M_D}^{(j)}, \quad i = m_J, \quad (5.29)$$

$$\left(\frac{\partial \mathbf{R}_{M_J}}{\partial \mathbf{b}_j^{(i)}}\right)^T \boldsymbol{\Psi}_{M_J}^{(i)} = - \left(\frac{\partial \mathbf{R}_{M_J}}{\partial \mathbf{b}_j^{(i)}}\right)^T \boldsymbol{\Psi}_{M_J}^{(i+1)}, \quad i = m_J - 1, \ldots, 1,$$

which are solved in a similar fashion as (5.20). Additional details on the RHS to the adjoint equation corresponding to $\boldsymbol{\Psi}_{M_J}^{(m_J)}$ have been provided in Section 5.1.3.

Equation (5.18) for the gradient can be simplified for aerodynamic design variables, $\mathbf{v}_A$, and structural design variables, $\mathbf{v}_S$, respectively, as follows:

$$\mathcal{G}_A = \frac{d \mathcal{J}}{d \mathbf{v}_A} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}_A}^T \frac{\partial \mathbf{R}_A}{\partial \mathbf{v}_A}^T \boldsymbol{\Psi}_A, \quad (5.30)$$

$$\mathcal{G}_S = \frac{d \mathcal{J}}{d \mathbf{v}_S} = \frac{\partial \mathcal{J}}{\partial \mathbf{v}_S}^T \frac{\partial \mathbf{R}_S}{\partial \mathbf{v}_S}^T \boldsymbol{\Psi}_S,$$

where the partial derivatives $(\partial \mathcal{J}_A/\partial \mathbf{v}_S)^T$ and $(\partial \mathcal{J}_S/\partial \mathbf{v}_A)^T$ are zero. The two expressions in (5.30) are in fact evaluated in the same way as in purely aerodynamic and purely structural optimization once $\boldsymbol{\Psi}_A$ and $\boldsymbol{\Psi}_S$ are obtained from the coupled adjoint solution. The gradients with respect to geometric design variables, $\mathbf{v}_G$, which affect both the flow and structural subproblems, are more involved. Hence they will be the focus of the remaining discussions.

In terms of equations and variables that depend on the geometry of the flow domain, both $(\partial \mathcal{J}_A/\partial \mathbf{v}_G)^T$ and $(\partial \mathbf{R}_A/\partial \mathbf{v}_G)^T \boldsymbol{\Psi}_A$ are equal to zero. The grid dependence of $\mathbf{R}_A$ and $\mathcal{J}_A$ is instead expressed through
Chapter 5. Aerostructural Analysis and Coupled Adjoint Formulation

the two mesh adjoint terms in (5.18), where
\[
\frac{\partial R_{MJ}}{\partial v_G}^T \Psi_{MJ} = \frac{\partial b_{sJ}}{\partial v_G}^T \frac{\partial R_{MJ}}{\partial b_{sJ}}^T \Psi_{MJ},
\]
(5.31)
\[
\frac{\partial R_{MA}}{\partial v_G}^T \Psi_{MA} = \frac{\partial b_{sA}}{\partial v_G}^T \frac{\partial R_{MA}}{\partial b_{sA}}^T \Psi_{MA}.
\]
(5.32)

Equations (5.31) and (5.32) differ in that \(b_{sJ}\) is determined directly from \(v_G\), but \(b_{sA}\) depends on \(v_G\) via fitting the shape changes due to the structural deflections. Furthermore, as Kenway et al. [30] have pointed out, the partial derivative of \(b_{sA}\) with respect to \(v_G\) needs to account for changes in both the undeflected geometry and the surface deflection. The latter in turn depends on the changes in the rigid link vectors. This subtle relationship needs to be included to accurately evaluate \((\partial b_{sA}/\partial v_G)^T\) in (5.32), which is computed as follows:
\[
\left( \frac{\partial b_{sA}}{\partial v_G} \right)^T = \left( \frac{\partial b_{sA}}{\partial b_{sJ}} \bigg|_{u_A} \frac{\partial b_{sJ}}{\partial v_G} + \frac{\partial b_{sA}}{\partial u_A} \bigg|_{b_{sJ}} \frac{\partial r}{\partial v_G} \right)^T
\]
(5.33)

where
\[
\frac{\partial r}{\partial v_G} = \frac{\partial r}{\partial X_S} \frac{\partial b_{sJ}}{\partial v_G} + \frac{\partial r}{\partial G_{sJ}} \frac{\partial b_{sJ}}{\partial v_G}.
\]
(5.34)

The above relation arises from the fact that the rigid link vectors, defined by (3.38), are attached to the undeflected aerodynamic surface geometry, \(G_{sJ}\), and the structural geometry, which is interpolated from the nodal coordinates in the structural mesh, \(X_S\). The transposed matrix-vector product with \(\partial u_A/\partial v_G\) in (5.33) is computed analytically in TACS, given the sensitivity information of the discrete aerodynamic and structural geometries with respect to \(v_G\). Assuming surface-based FFD is used to warp the structural mesh, \((\partial X_S/\partial v_G)^T\) is evaluated according to (4.28). The transposed matrix-vector product with \(\partial b_{sA}/\partial u_A\) is evaluated in the same way as in (5.26).

The values of \(R_S\) and \(J_S\) are explicit functions of \(v_G\) due to changes in the structural mesh, \(X_S\), as a result of geometric changes. For structural functionals, \(\partial J_S/\partial v_G\) is unmodified from purely structural optimization. The partial derivative of the structural residual with respect to the geometric design variables is given by the following:
\[
\frac{\partial R_S}{\partial v_G}^T \Psi_S = \left[ \frac{\partial X_S}{\partial v_G}^T \frac{\partial K_S}{\partial X_S} \frac{\partial R_S}{\partial K_S}^T \frac{\partial r}{\partial v_G} \right] \left( \frac{\partial f_S}{\partial r} \bigg|_{G_S} \right)^T \frac{\partial R_S}{\partial f_S}^T \Psi_S.
\]
(5.35)

This differs from the expression given by Kennedy and Martins [73] due to the presence of an explicit mesh state. In a two-field formulation, the lengths of the individual rigid links affect the moment terms in the force vector (see (3.43)), as well as the surface geometry over which the integration is performed. Both of these change the results of the force transfer. In a three-field formulation, however, the surface grid, which is a part of the volumetric grid, \(G\), is a function of the mesh state. As a result, only the changes in the rigid link moments are included in (5.35).

### 5.2.3 Gradient Verification

This section compares the gradients computed by the coupled adjoint method with a second-order finite-difference approximation. The analysis and the gradient evaluation are performed on an intermediate
5.2. Gradient Calculations by the Coupled Adjoint Method

Figure 5.10: The initial and final wing geometries used for the gradient verification tests. The nonuniform thickness distribution of the structural components is also shown.

Figure 5.11: Normalized difference in functional gradients between the adjoint method and finite-difference approximations.

Design during an optimization. This helps generalize the conclusion of this study by ensuring that the flow and structural grids have both been changed prior to the analysis. Figure 5.10 shows the starting geometry for the optimization and the jig shape of the design on which the gradients are verified. The chosen design has a nonuniform thickness distribution which is also illustrated in Figure 5.10.

The flow grid has 458,752 nodes and is divided into 112 blocks. Each block is parameterized by $6 \times 6 \times 6$ B-spline control points. The finite-element mesh consists of 30,030 second-order MITC shell elements. The study is conducted at a Mach number of 0.785 at an altitude of 35,000 feet, with an AOA of 1.0°. The functionals considered include the drag and lift forces, as well as the KS functions defined in Section 3.2.4. Three KS functions are used to monitor the stresses in the 1) ribs and spars, 2) top skin, and 3) bottom skin. The gradients with respect to a thickness design variable for a lower skin panel near the crank of the wing, the AOA, and the dihedral at the wing tip are examined. This allows the gradients with respect to each type of design variables to be verified independently due to the differences in the calculations involved. The aerostructural analysis and coupled adjoint calculations are converged to relative tolerances of $10^{-9}$ and $10^{-10}$ respectively.

Figure 5.11 plots the normalized difference between the calculated gradient and the finite-difference approximations with different step sizes. The largest minimum normalized difference is on the order of $10^{-6}$, but the majority of the gradient values have a minimum normalized difference on the order of $10^{-8}$. This shows excellent agreement between the computed gradients and the finite-difference approximations.
5.3 Chapter Summary

This chapter proposed a new methodology for aerostructural analysis and optimization, where the integrated geometry parameterization and mesh movement is used for shape changes due to both optimization and structural deflections. The different software modules introduced in earlier chapters were integrated effectively. The present methodology uses a three-field formulation to simplify the introduction of the linear elasticity mesh movement of the B-spline volumes into the analysis and coupled adjoint problems. A fitting step was added to the displacement transfer to produce an appropriate set of control point displacements at the aerodynamic surface for the purpose of mesh movement during analysis. Efficient ways to move the flow grid during aerostructural optimization were also investigated. Details of the gradient calculations were further provided. Using the block Gauss-Seidel solution method for the analysis and the coupled adjoint problems, the new methodology was characterized via a number of results. It was shown that the fitting error during displacement transfer does not affect the post-optimality grid convergence of important functionals. In addition, the present methodology was shown to be able to reproduce the results from the HIRENASD experiment and compute highly accurate gradients for optimization.
Chapter 6

Monolithic Solution to Aerostructural Analysis and Coupled Adjoint Problems

Exploratory high-fidelity aerostructural optimization is not possible without efficient and robust solution methods for the coupled analysis and adjoint problems. This motivates the use of monolithic solution methods. In comparison to the block Gauss-Seidel method used in Chapter 5, monolithic solution methods require more developmental effort. Different aspects of the solution algorithm must be carefully designed and tested in order to maximize its efficiency. Although monolithic solution methods have been an active area of research in the general field of FSI, only a handful of authors have investigated their application to high-fidelity aerostructural optimization based on steady analysis. Some notable examples include the work of Barcelos et al. [51] and Kenway and Martins [30]. The reported computational savings from using monolithic solution methods have been between 10% to 30% for wings with realistic deflections. However, it is unclear whether these numbers represent the maximum possible efficiency improvements. More specifically for this work, it also remains to be demonstrated whether a monolithic solution technique is effective for the analysis and the coupled adjoint formulations introduced in Chapter 5 where the mesh subproblem could involve a number of increments. This chapter attempts to address these questions by investigating the design of an efficient monolithic solution method for the present aerostructural optimization methodology.

A modular approach to monolithic solution methods is adopted here, which differentiates between the flow, mesh, and structural equations although the calculations are fully coupled. In other words, the block structure in the coupled Jacobian matrix of $R_{AS}$,

$$
A = \begin{bmatrix}
\frac{\partial R_A}{\partial q} & \frac{\partial R_A}{\partial b_\Delta} & 0 \\
0 & \frac{\partial R_{M\Delta}}{\partial b_\Delta} & \frac{\partial R_{M\Delta}}{\partial u} \\
\frac{\partial R_S}{\partial q} & \frac{\partial R_S}{\partial b_\Delta} & \frac{\partial R_S}{\partial u}
\end{bmatrix},
$$

(6.1)

or its transpose in the adjoint problem, is preserved during the solution process. The Jacobian blocks...
Chapter 6. Monolithic Solution Method

in Section 6.1 group together all equations and unknowns related to each subproblem. A nonmodular approach that does not distinguish between the different field blocks may simplify the application of certain matrix operations such as row and column reordering. However, a modular approach allows for block operations during the monolithic solution process that reuse existing routines discussed in earlier chapters. This could include routines for matrix-vector product evaluations and the distributed iterative solutions to linear systems. Doing so is attractive partly because of the reduced implementation cost. More importantly, an effective monolithic solution strategy should take advantage of the specialized numerical treatment of each subproblem. The physics modelled by the three equations are inherently different. These in turn lead to differences in the appropriate choice of spatial discretizations and in the characteristics of the resulting systems of equations. It becomes more difficult to avoid these differences as the coupled problem becomes more complex and the fidelity of the simulation increases. This certainly holds true for the aerostructural optimization methodology developed for this thesis. The flow, mesh, and structural modules in the present framework are highly advanced and have been specialized to solve each subproblem very efficiently. Maintaining the block structure within the coupled system makes it easier to exploit these specialized features during the monolithic solution process.

The remainder of this chapter is divided into three sections. Section 6.1 applies a Newton-Krylov solution algorithm to the nonlinear analysis problem. The effects of scaling in the linearized coupled problem at each Newton iteration, matrix-vector product evaluation, and the preconditioner for the Krylov subspace method are thoroughly studied. A detailed cost comparison between the monolithic analysis and the nonlinear block Gauss-Seidel method is provided in Section 6.1.4. Similar investigations are conducted in Section 6.2 for the use of a Krylov subspace iterative method to solve the coupled adjoint problem. Section 6.3 verifies the present implementation of the monolithic method by ensuring that it does not alter the verification results obtained using the block Gauss-Seidel method.

### 6.1 Aerostructural Analysis using a Newton-Krylov Method

This section examines the use of a Newton-Krylov method to solve the discrete steady aerostructural equations,

\[
\begin{bmatrix}
R_A(q, b_{\Delta}) \\
R_M\Delta(b_{\Delta}, u) \\
R_S(q, b_{\Delta}, u)
\end{bmatrix} = 0, \tag{6.2}
\]

which are now treated as a single system of nonlinear equations. Convergence of the Newton method requires an initial guess that is sufficiently close to the steady-state solution of the coupled analysis problem. This initial guess is currently found by applying up to a given number of nonlinear block Gauss-Seidel iterations to reduce the norms of \(R_A\), \(R_M\Delta\), and \(R_S\) below a relative tolerance of 0.05. This is sufficient for the purpose of studying the application of the Newton method to the coupled nonlinear analysis. However, it could suffer from the shortcomings of a partitioned method and ultimately compromise the overall robustness and efficiency of the solution algorithm. This motivates future investigations into a more effective way to globalize the Newton method for the present problem, but this subject is beyond the scope of the present thesis.

Each iteration of the Newton method requires the solution to the following coupled linear system of equations:

\[
\frac{\partial R_{AS}^{(n)}}{\partial (q, b_{\Delta}, u)^{(n)}} \begin{bmatrix}
\Delta q \\
\Delta b_{\Delta} \\
\Delta u
\end{bmatrix}^{(n)} = \Lambda^{(n)} \delta^{(n)} = -R_{AS}^{(n)}. \tag{6.3}
\]


Using the solution to (6.3), the new iterate for the next Newton iteration is updated by

$$
\begin{bmatrix}
q \\
b_{\Delta} \\
u
\end{bmatrix}^{(n+1)} = \begin{bmatrix}
q \\
b_{\Delta} \\
u
\end{bmatrix}^{(n)} + \lambda^{(n)} \begin{bmatrix}
\Delta q \\
\Delta b_{\Delta} \\
\Delta u
\end{bmatrix}^{(n)},
$$

(6.4)

where $\lambda^{(n)} \in (0, 1]$ is a damping/relaxation parameter chosen to prevent unphysical flow or mesh updates, and it is set to 1 by default. The Newton method is popular due to its fast local convergence. If $R_{\text{AS}}$ is Lipschitz continuous near the steady-state solution, then the Newton method converges $q$-quadratically when the iterates are sufficiently close to the solution [125]. For large-scale engineering applications, a direct solution to (6.3) is impractical, and it is much more common to solve (6.3) iteratively to some tolerance such that

$$
\|A^{(n)} \delta^{(n)} + R_{\text{AS}}^{(n)}\| \leq \epsilon^{(n)}, \quad \epsilon \in [0, 1),
$$

(6.5)

where $\epsilon$ is called a forcing parameter. The use of an iterative method at each Newton step results in an inexact Newton method [141]. The inexact Newton method converges $q$-linearly with a sufficiently small $\epsilon^{(n)}$. It converges $q$-superlinearly if $\epsilon^{(n)} \to 0$, and it converges $q$-quadratically if $\epsilon^{(n)} = O(\|R_{\text{AS}}^{(n)}\|)$ [125]. However, $q$-quadratic convergence is not necessarily advantageous if it comes at the cost of an expensive linear system solution at every Newton step. Methods have been proposed for choosing a suitable $\epsilon^{(n)}$ for each nonlinear iteration to prevent oversolving the linear system. This was shown to be necessary towards the beginning of a nonlinear solution process using an inexact Newton method globalized by backtracking [142]. However, since it is assumed here that a good initial guess has been given, a fixed value of $\epsilon^{(n)} = 0.01$ is found to be more effective than an adaptive forcing parameter.

Krylov subspace methods form an important class of iterative methods for solving large linear systems such as (6.3). Given an initial residual, $r_0$, of the linear system of interest, a Krylov subspace method seeks an approximate linear system solution in the Krylov subspace given by [109]

$$
K_m(A, r_0) = \text{span}\{r_0, A r_0, A^2 r_0, \cdots, A^{m-1} r_0\},
$$

(6.6)

where $m$ is the number of Krylov iterations. The Newton iteration index $n$ is dropped for convenience. Among Krylov subspace methods for general nonsymmetric matrices, there are two subgroups of methods based on the long-recurrence Arnoldi orthogonalization procedure and the short-recurrence Lanczos bi-orthogonalization procedure [143]. The two differ in that the former generates bases of the Krylov subspace that are orthogonal, while the latter generates two sets of basis vectors that are bi-orthogonal. It is generally believed that no Krylov subspace method is clearly superior to others and different iterative methods are suitable for different problems [144, 145]. The Arnoldi-based Generalized Minimal Residual (GMRES) method has been found to be more robust than some of the popular Lanczos methods when the Jacobian matrix-vector product is approximated by finite-differences [146]. This is a convenient feature as it may not always be desirable to compute $A$ when solving (6.3). GMRES has also been successfully applied in the flow problem [86]. For these reasons, GMRES is chosen here for the monolithic solution of the aerostructural analysis problem.

Each GMRES iteration seeks an approximation to the linear system solution that minimizes the linear system residual over the Krylov subspace, thus producing monotonically decreasing residuals. The Arnoldi basis vector at each iteration needs to be orthonormalized against all basis vectors from the
previous iterations using a modified Gram-Schmidt procedure. This can result in heavier memory usage than Lanczos methods which do not require storing as many basis vectors [143, 144]. Although restarting can be used to limit the memory requirements, it often degrades the performance of GMRES [144]. As a result, it is important to use an effective preconditioner that allows GMRES to converge in as few iterations as possible. The preconditioner for the present methodology will make use of existing linear solution routines from the flow, mesh, and structural modules. This in turn requires a flexible variant of GMRES, or FGMRES. FGMRES is based on the right-preconditioned GMRES algorithm, which solves
\begin{equation}
AM^{-1}(M\delta) = -R_{AS}
\end{equation}
in place of the original linear system, where $M \approx A$ is the preconditioner matrix. FGMRES modifies the right-preconditioned GMRES such that the calculation of $M^{-1}z$ with a given vector $z$ can be replaced by another iterative solution procedure. FGMRES, however, does use more memory than GMRES, as it needs to store $m$ additional preconditioned Arnoldi basis vectors [109].

The remainder of this section provides the details on the linear system scaling, matrix-vector product calculations, and the preconditioning methods used for the Newton-Krylov solution algorithm. A test problem is necessary to demonstrate the effects of different solution parameters. A wing with planform based on the Boeing 737-900 wing and with the RAE 2822 airfoil section, as shown in Figure 6.1, is used for this purpose. It is operating at a Mach number of 0.798, an altitude of 12,000 feet, and an AOA of 2°. The flow grid has a total of 458,752 nodes and is partitioned into 112 blocks, where each block is controlled by $6 \times 6 \times 6$ control points. The wingbox model used for the aerostructural analysis is shown in Figure 6.2. It has a total of 171,960 degrees of freedom. A uniform component thickness of 10mm is assumed. This problem corresponds to a maneuver flight condition, where the use of a monolithic solution strategy is potentially beneficial because finding the coupled solution can be challenging with a partitioned method. The mesh equation uses three increments to move the flow grid due to structural deflections. For the results presented in the remainder of this section, aerostructural analysis of the test problem is terminated when the nonlinear residual convergence plateaus for all three equations in $R_{AS}$. Experience from using the partitioned method suggests that this is expected to occur at relative residual values on the order of $10^{-11}$, $10^{-14}$, and $10^{-9}$ for the flow, mesh, and structural subproblems, respectively.
6.1.1 Scaling

During a Newton-Krylov solution procedure, it is important to ensure that all equations and variables are roughly on the same scale. Discretization of a PDE does not automatically guarantee good scaling, especially when the equations and variables represent inherently different physical quantities. This is true, for example, in the RANS equations with the presence of a turbulence model \cite{147}. Poor scaling is also a by-product of the curvilinear transformation of the Euler equations discussed in Section 3.1.1. Due to the presence of the metric Jacobian, \(J\), which is proportional to the volume of the computational cell, variables and residual entries near the far field appear greater in magnitude to the linear solver. GMRES can be tricked into minimizing the linear system residual by reducing the magnitudes of the residual components near the far field simply due to incorrect scaling. This could have a detrimental effect on the convergence of the Newton iterations \cite{147}. There are other reasons for having a well-scaled linear system, although the effects may be less obvious. Scaling can improve the condition number of the linear system as well as the accuracy of the iterative solution \cite{111,112}. When a finite-difference approximation is used to compute the matrix-vector product, it is easier to choose an optimal steps size that maximizes the accuracy of the approximation when the equations and variables have similar magnitudes \cite{147}. Scaling is of particular relevance when solving the coupled aerostructural problems in (6.3), where the scaling between the flow, mesh, and structural subproblems differ by orders of magnitude. Scaling in the coupled problem can be improved by applying appropriate row and column scaling to the linear system at every Newton iteration. This needs to be incorporated into the matrix-vector product calculation and the preconditioning procedure.

Row and column scaling are applied to a general linear system \(Ax = b\) in the form of diagonal matrices \(D_t\) and \(D_c\), so that a scaled system of equations \(D_tAD_c(D_t^{-1}x) = D_tb\) is solved instead for the same solution \(x\). Choices of \(D_t\) and \(D_c\) which minimize the condition number of \(D_tAD_c\) based on the 1-, 2-, and \(\infty\)-norms have been the subject of numerous studies \cite{148,149,150,151}. Such optimal scaling of matrices is impractical in the context of engineering applications as the inverse of \(A\) is often needed. Furthermore, the exact minimization of the condition number is not necessary as it is only a heuristic measure of the solution accuracy \cite{112}. It is much more common to select \(D_t\) and \(D_c\) such that all rows and columns in \(D_tAD_c\) have norms that are at most an order of magnitude different \cite{112}. This still requires access to individual entries in the LHS matrix, which is not practical for the coupled aerostructural system due to the intensive memory requirements. It is finally important to note that a general scaling procedure may not be suitable for all problems \cite{112}. Hence it is best to scale the linear system with the nature of the underlying problem in mind.

In the present algorithm, the coupled linear system at every Newton iteration is scaled as follows:

\[
\begin{bmatrix}
    r_{scl,A} \left( \frac{\partial \hat{R}_A}{\partial q} \right) c_{scl,A} & r_{scl,A} \left( \frac{\partial \hat{R}_A}{\partial \delta} \right) c_{scl,M} & 0 \\
    0 & r_{scl,M} \left( \frac{\partial \hat{R}_{Ma}}{\partial \delta} \right) c_{scl,M} & r_{scl,M} \left( \frac{\partial \hat{R}_{Ma}}{\partial u} \right) c_{scl,S} \\
    r_{scl,S} \left( \frac{\partial \hat{R}_S}{\partial q} \right) c_{scl,A} & r_{scl,S} \left( \frac{\partial \hat{R}_S}{\partial \delta} \right) c_{scl,M} & r_{scl,S} \left( \frac{\partial \hat{R}_S}{\partial u} \right) c_{scl,S}
\end{bmatrix}
\begin{bmatrix}
c_{scl,A} \Delta q \\
c_{scl,M} \Delta b_{\delta} \\
c_{scl,S} \Delta u
\end{bmatrix} =
\begin{bmatrix}
-r_{scl,A} \hat{R}_A \\
-r_{scl,M} \hat{R}_{Ma} \\
-r_{scl,S} \hat{R}_S
\end{bmatrix}
\]

(6.8)

where \(\hat{\delta} = -\hat{R}_{AS}\) is the new linear system to be solved. Two types of scaling are applied in (6.8): scaling within the flow and mesh subproblems, where the scaled equations and variables are denoted
by the \( \hat{\cdot} \) symbol, and scaling of the 9 Jacobian blocks in \( A \) using scalars \( r_{\text{sc1},A} \) and \( c_{\text{sc1},A} \). Equations and variables in the flow and mesh subproblems are scaled using existing routines within the respective solver module for a number of reasons. The effect of \( J^{-1} \) in the flow equations is independent of the presence of the mesh and structural equations, so the appropriate row scaling is still applicable during coupled aerostructural calculations. Moreover, existing linear solution routines from each discipline are used as block preconditioners for the solution to the linearized coupled problem. It makes sense to also reuse the existing scaling routines, which have been optimized to work well with the individual linear solvers.

Rows corresponding to the flow equations are first scaled by a factor of \( J^{2/3} \) to alleviate the effect of the \( J^{-1} \) term from curvilinear transformation. In addition, differences in scaling between the mass, momentum, and energy equations are accounted for by dividing each row by the norm of the corresponding equation computed over the entire flow domain \( \| \hat{R}_A \| \). The scaled flow equations are denoted by \( \hat{\hat{R}}_A \). For the mesh equation, Section 4.3 showed that \( K^{(i)}_{M\Delta} \) at an increment \( i \) can also be poorly scaled due to the presence of small or skewed elements in the moved control grid. This issue can be effectively addressed by scaling \( K^{(i)}_{M\Delta} \) according to \( (4.19) \), where the scaled matrix \( \hat{K}^{(i)}_{M\Delta} \) is given by \( D_{(i)}^{-1/2} K^{(i)}_{M\Delta} D_{(i)}^{-1/2} \).

\[
D_M = \begin{bmatrix}
D_{(1)} & 0 & \cdots & 0 \\
0 & D_{(2)} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & D_{(n\Delta)}
\end{bmatrix}, \quad \text{where } D_{(i)} = \text{diag}(K^{(i)}_{M\Delta}),
\]

then the same scaling of \( K^{(i)}_{M\Delta} \) in \( (4.19) \) is accomplished by defining \( \hat{R}_{M\Delta} = D_M^{-1/2} R_{M\Delta} \) and \( \hat{b}_{\Delta}^{(i)} = D_M^{1/2} b_\Delta \). Since the original structural module does not apply any scaling to \( R_S \), only the Jacobian block is applied to the structural subproblem, as discussed below.

In order to choose the appropriate row and column scaling values for the Jacobian blocks in \( A \), it is instructive to examine the relative scaling between each discipline. To do so without having access to the individual entries in the coupled aerostructural Jacobian matrix, an estimate is obtained in the following manner:

\[
S = \begin{bmatrix}
\frac{1}{n_A} \left\| \frac{\partial R_A}{\partial \mathbf{q}} \right\|_{c_{\text{sc1},A}} & \frac{1}{n_M} \left\| \frac{\partial R_A}{\partial \mathbf{b}} \right\|_{c_{\text{sc1},M}} & 0 \\
0 & \frac{1}{n_M} \left\| \frac{\partial R_M}{\partial \mathbf{R}_{M\Delta}} \right\|_{c_{\text{sc1},M}} & 0 \\
\frac{1}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{q}} \right\|_{c_{\text{sc1},A}} & \frac{1}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{b}} \right\|_{c_{\text{sc1},M}} & \frac{1}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{u}} \right\|_{c_{\text{sc1},S}}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\frac{r_{\text{sc1},A}}{n_A} \left\| \frac{\partial R_A}{\partial \mathbf{q}} \right\|_{c_{\text{sc1},A}} & \frac{r_{\text{sc1},A}}{n_A} \left\| \frac{\partial R_A}{\partial \mathbf{b}} \right\|_{c_{\text{sc1},M}} & 0 \\
0 & \frac{r_{\text{sc1},M}}{n_M} \left\| \frac{\partial R_M}{\partial \mathbf{R}_{M\Delta}} \right\|_{c_{\text{sc1},M}} & 0 \\
\frac{r_{\text{sc1},S}}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{q}} \right\|_{c_{\text{sc1},A}} & \frac{r_{\text{sc1},S}}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{b}} \right\|_{c_{\text{sc1},M}} & \frac{r_{\text{sc1},S}}{n_S} \left\| \frac{\partial R_S}{\partial \mathbf{u}} \right\|_{c_{\text{sc1},S}}
\end{bmatrix}, \quad (6.10)
\]

where \( n_A, n_M, \) and \( n_S \) represent the number of equations or unknowns in the flow, mesh, and structural subproblem, respectively. Each entry in \( S \in \mathbb{R}^{1 \times 3} \) can be interpreted as a measure of the changes in each equation, normalized by the size of the subproblem, due to a unit perturbation in \( \mathbf{q}, \mathbf{b}_\Delta, \) or \( \mathbf{u}, \)
where $1_A \in \mathbb{R}^{n_A \times 1}$, $1_M \in \mathbb{R}^{n_M \times 1}$, and $1_S \in \mathbb{R}^{n_S \times 1}$ are vectors of ones. Some entries in $1_A$, $1_M$, and $1_S$ are given a negative sign to avoid producing rigid-body displacements in the perturbed mesh and structural state. Figure 6.3 shows the values of $S/S_{11}$ with and without any Jacobian block scaling for the test problem described on page 74. The normalization by $S_{11}$ assumes that the flow Jacobian block $\partial \tilde{R}_A/\mathbf{q}$ is well scaled due to the row scaling described previously and the nondimensionalization of all flow variables. Figure 6.3a indicates the relative scaling in $\tilde{A}$ when all $r_{scl,*}$ and $c_{scl,*}$ are set to unity. Evidently, the system is not very well scaled as $S_{33}$ for the structural Jacobian block is greater than the rest of the entries by orders of magnitude.

Figure 6.4 further plots the convergence of $\|R_A\|_2$, $\|R_{M\Delta}\|_2$, and $\|R_S\|_2$ with no Jacobian block scaling applied in $\tilde{A}$.

Two different ways to choose $r_{scl,*}$ and $c_{scl,*}$ have been considered, which are summarized in Table 6.1. Since the relative scaling of the Jacobian blocks is measured relative to $S_{11}$, both $r_{scl,A}$ and $c_{scl,A}$ are set to unity. The values of $r_{scl,M}$ and $r_{scl,S}$ in the norm-based scaling include the inverse of the initial mesh and structural residual norms, respectively, from the start of the analysis. The initial structural residual, $R_S^{(0)}$, is an initial estimate of the forces acting on the structures, the norm of which does not
both control point coordinates between successive increments, denoted by $\Delta c^{\ddagger} b$, as close to unity as possible in a least-squares sense, for all

$$\| R_A \|_2, \| R_{M\Delta} \|_2, \text{ and } \| R_S \|_2$$

with different Jacobian block scaling in $\hat{A}$.

Table 6.1: Different block row and column scaling factors considered for $\hat{A}$.

<table>
<thead>
<tr>
<th></th>
<th>$r_{sc1,A}$</th>
<th>$r_{sc1,M}$</th>
<th>$r_{sc1,S}$</th>
<th>$c_{sc1,A}$</th>
<th>$c_{sc1,M}$</th>
<th>$c_{sc1,S}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Norm-based Scaling</td>
<td>1.0</td>
<td>$n_M/|\hat{R}_{M\Delta}^{(0)}|_2$</td>
<td>1.0/|\hat{R}_{S}^{(0)}|_2$</td>
<td>1.0</td>
<td>$m_\Delta/|\Delta \hat{b}^{(n)}_\Delta|_2/n_M$</td>
<td>$|\hat{u}|_2/n_S$</td>
</tr>
<tr>
<td>Automatic Scaling</td>
<td>1.0</td>
<td>Automatic</td>
<td>1.0</td>
<td>Automatic</td>
<td>Automatic</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

![Figure 6.5: Convergence of $\| R_A \|_2, \| R_{M\Delta} \|_2, \text{ and } \| R_S \|_2$ with different Jacobian block scaling in $\hat{A}$.](image)

tend to scale with $n_S$. On the other hand, $\| R_{M\Delta}^{(0)} \|_2$ is not a physical quantity, so it is necessary to include a factor of $n_M$ in $r_{sc1,M}$. The mesh block column scaling, $c_{sc1,M}$, is based on the scaled change in the control point coordinates between successive increments, denoted by $\Delta \hat{b}$, and $n_M/m_\Delta$ is the size of the mesh problem at each increment. Similarly, the value of $c_{sc1,S}$ is chosen based on an averaged norm of the structural state. Both $c_{sc1,S}$ and $c_{sc1,S}$ are recomputed at every Newton iteration. The second scaling option determines the values of $r_{sc1,M}$, $r_{sc1,S}$, $c_{sc1,M}$, and $c_{sc1,S}$ automatically such that $\log_{10}(S_{ij}/S_{11})$ is as close to unity as possible in a least-squares sense, for all $S_{ij} \neq 0$ and $i \neq j \neq 1$. This involves solving an overdetermined system with 6 equations and 4 unknowns at each Newton iteration.

Figures 6.3b and 6.3c show the results after applying the two types of Jacobian block scaling from Table 6.1. The relative scaling between the Jacobian blocks in $\hat{A}$ has clearly been improved. This is also reflected in Figure 6.5, which plots the nonlinear residual convergence with the use of the norm-based and the automatic scaling methods. The expected convergence rate for an inexact Newton method has been recovered for all equations, including $R_A$ and $R_{M\Delta}$ whose convergence in Figure 6.4 appears to be adversely affected by poor scaling. A total of 7 Newton iterations are required to fully converge all equations, whereas 35 iterations are required in Figure 6.4. The two methods listed in Table 6.1 are generally similar in their effectiveness. This shows that the approximate scaling values in $\hat{S}$ provide a reasonably good indication of how the Jacobian blocks in $\hat{A}$ should be scaled. Although the automatic scaling has led to slightly faster convergence than the norm-based scaling method in this case, the difference is not significant. However, the automatic scaling method requires extra calculations at each Newton iteration, so the norm-based scaling is used by default for analysis.

### 6.1.2 Matrix-Vector Product

A routine to compute the matrix-vector product with the scaled matrix $\hat{A}$ in (6.8) is required by FGMRES. Using the block structure in $\hat{A}$, the Jacobian matrix-vector products with the individual field blocks can be considered separately, each of which may be evaluated analytically or by finite-difference
6.1. Aerostructural Analysis using a Newton-Krylov Method

approximations. The choice of an appropriate method for matrix-vector product calculation is influenced by two factors. It must be efficient, as it is needed many times during the iterative solution process. It should also be sufficiently accurate relative to the required linear system solution tolerance in order to ensure the convergence of the outer Newton iterations.

Evaluation of the exact Jacobian matrix-vector product can be accomplished by assembling the matrix explicitly and storing it in memory. It may also be desirable to evaluate the matrix-vector product using automatic differentiation to avoid the extra memory cost and simplify the development process. Not forming the matrix, however, also increases the time required per matrix-vector product evaluation which can render automatic differentiation less competitive, especially for equations which are expensive to differentiate. In contrast, approximating the Jacobian matrix-vector product using finite-differences is more straightforward and only requires the storage of the perturbed residual vectors. During an inexact Newton-Krylov solution procedure, an analytical Jacobian matrix-vector product can improve the robustness of the linear and nonlinear solution algorithms by avoiding the Jacobian-free breakdown [147]. Nevertheless, if an appropriate finite-difference step size can be found, the added accuracy from using an exact Jacobian is not necessary because the linear system solution is not converged deeply. Despite the above arguments in favour of finite-differences, it can be more efficient to use a matrix-explicit approach depending on the cost associated with forming the Jacobian matrix and subsequent matrix-vector multiplications relative to the cost of evaluating the residual. With the above considerations in mind, matrix-vector products with the Jacobian blocks in (6.8) are evaluated using a combination of finite-difference approximation and analytical differentiation.

Using \( \mathbf{z} = [\mathbf{z}_A, \mathbf{z}_M, \mathbf{z}_S]^T \) to represent the vector to be multiplied by \( \hat{\mathbf{A}} \), the block column scaling, \( c_{\text{scl},A}, c_{\text{scl},M} \), and \( c_{\text{scl},S} \), in \( \hat{\mathbf{A}} \) can be incorporated into \( \mathbf{z} \). It can also be recognized that

\[
\frac{\partial \hat{\mathbf{R}}_{AS}}{\partial \mathbf{b}_\Delta} (c_{\text{scl},M} \mathbf{z}_M) = \frac{\partial \hat{\mathbf{R}}_{AS}}{\partial \mathbf{b}_\Delta} (c_{\text{scl},M} \mathbf{z}_M) = \frac{\partial \hat{\mathbf{R}}_{AS}}{\partial \mathbf{b}_\Delta} D_M^{-1/2} (c_{\text{scl},M} \mathbf{z}_M),
\]

which means that by including the scaling of the mesh state variable in \( \mathbf{z}_M \), all derivatives with respect to the scaled mesh state, \( \hat{\mathbf{b}}_\Delta \), can be taken instead with respect to the unscaled mesh state, \( \mathbf{b}_\Delta \). For this reason, it is convenient to use the following scaled vector,

\[
\hat{\mathbf{z}} = \left[ c_{\text{scl},A} \mathbf{z}_A; c_{\text{scl},M} \mathbf{D}_M^{-1/2} \mathbf{z}_M; c_{\text{scl},S} \mathbf{z}_S \right]^T = [\mathbf{\hat{z}}_A, \mathbf{\hat{z}}_M, \mathbf{\hat{z}}_S]^T,
\]

to calculate the Jacobian matrix-vector products.

The matrix-vector products with the Jacobian blocks of the flow equation use a first-order finite-difference approximation given by the following expression:

\[
r_{\text{scl},A} \left( \frac{\partial \hat{\mathbf{R}}_{A}^{(n)}}{\partial [\mathbf{q}, \mathbf{b}_\Delta]^{(n)}} \right) \left[ \begin{array}{c} \hat{\mathbf{z}}_A \\ \hat{\mathbf{z}}_M \end{array} \right] \approx r_{\text{scl},A} \frac{\hat{\mathbf{R}}_{A}(\mathbf{q}^{(n)} + \delta_{\text{FD}} \hat{\mathbf{z}}_A, \mathbf{b}_\Delta^{(n)} + \epsilon \hat{\mathbf{z}}_M) - \hat{\mathbf{R}}_{A}(\mathbf{q}^{(n)}, \mathbf{b}_\Delta^{(n)})}{\delta_{\text{FD}}},
\]

where \( n \) is the Newton iteration index. It is crucial to choose an optimal finite-difference step size, \( \delta_{\text{FD}} \), to ensure the accuracy of (6.13). A smaller \( \delta_{\text{FD}} \) leads to a smaller truncation error, but the resulting approximation is more prone to subtractive cancellation error. Here the value of \( \delta_{\text{FD}} \) is determined by [80, 102]

\[
\delta_{\text{FD}} = \sqrt{\frac{(n_A + n_M)\epsilon_{\text{Machine}}}{\hat{\mathbf{z}}_A^T \hat{\mathbf{z}}_A + \hat{\mathbf{z}}_M^T \hat{\mathbf{z}}_M}} = \frac{\epsilon_{\text{Machine}}}{\sqrt{\text{RMS}([\hat{\mathbf{z}}_A; \hat{\mathbf{z}}_M]^T)}},
\]
where $\epsilon_{\text{Machine}} = 10^{-13}$ is roughly the expected accuracy from floating point operations. This choice of $\delta_{\text{FD}}$ leads to perturbations, $\delta_{\text{FD}} \hat{z}_A$ and $\delta_{\text{FD}} \hat{z}_M$, of the state variables that are roughly on the order of $\sqrt{\epsilon_{\text{Machine}}}$, where the RMS value of the entries in $\hat{z}_A$ and $\hat{z}_M$ provides an estimate to the average magnitude of $[\hat{z}_A, \hat{z}_M]$. Equation (6.13) only requires a single evaluation of $\hat{R}_A$ because $\hat{R}_A(q^{(n)}, b^{(m)})$ is part of the RHS in (6.8). In contrast, calculating the Jacobian terms in (6.13) exactly requires $\partial \hat{R}_A/\partial q \hat{z}_A$ and $(\partial \hat{R}_A/\partial b\Delta) \hat{z}_M$ need to be computed separately. Furthermore, it is expensive to differentiate $\hat{R}_A$. For the test problem considered for this section, it takes roughly 0.18s to form $\partial \hat{R}_A/\partial q$, whereas one evaluation of $\hat{R}_A$ requires only 3.7ms. The use of (6.14) does not eliminate the need to form a first-order approximation to the flow Jacobian for the flow block preconditioner, which will be introduced in Section 6.1.3. However, the time and memory requirement associated with assembling the flow Jacobian approximation is much smaller than that required for the exact Jacobian. Therefore, the finite-difference approximation is the more competitive option here if it can be shown that it is sufficiently accurate.

The Jacobian matrix-vector products related to the mesh equation, given by

$$r_{\text{scl}, M} \left( \frac{\partial \hat{R}_{M\Delta}^{(n)}}{\partial b_\Delta, u|^{(n)}} \left[ \hat{z}_M \right] \right) = r_{\text{scl}, M} \left( \frac{\partial \hat{R}_{M\Delta}^{(n)}}{\partial b_\Delta} \hat{z}_M + \frac{\partial \hat{R}_{M\Delta}^{(n)}}{\partial u} \hat{z}_S \right), \quad (6.15)$$

are evaluated analytically. Recall that the mesh residual vector $R_{M\Delta}$ consists of $m_\Delta$ linear elasticity mesh movement equations for the $m_\Delta$ increments used during aerostructural analysis. The first term on the RHS of (6.15) is expanded to

$$\frac{\partial \hat{R}_{M\Delta}}{\partial b_\Delta} \hat{z}_M = \begin{bmatrix} D_{(1,n)}^{-1/2} K_{M\Delta}^{(1,n)} & 0 & \cdots & 0 & 0 \\ D_{(2,n)}^{-1/2} K_{M\Delta}^{(2,n)} & K_{M\Delta}^{(2,n)} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & D_{(m_\Delta,n)}^{-1/2} K_{M\Delta}^{(m_\Delta,n)} & K_{M\Delta}^{(m_\Delta,n)} \end{bmatrix} \begin{bmatrix} \hat{z}_M^{(1)} \\ \hat{z}_M^{(2)} \\ \vdots \\ \hat{z}_M^{(m_\Delta)} \end{bmatrix}. \quad (6.16)$$

The expressions for the off-diagonal entries in (6.16), $\partial R_{M\Delta}^{(i,n)} / \partial b_\Delta^{(i-1,n)}$ for $i = 2, \ldots, m_\Delta$, are given in (4.34). Each evaluation of $R_{M\Delta}$ involves a matrix-vector product with the stiffness matrix $K_{M\Delta}^{(i)}$ per increment. In comparison, each evaluation of (6.16) requires an additional matrix-vector product with $\hat{z}_M^{(i-1)}$ for each of the $m_\Delta - 1$ increments. However, the entries in (6.16) can be precomputed and stored at the start of each Newton iteration, but each finite-difference approximation of (6.15) requires updating all stiffness matrices according to the perturbed mesh states. Using the distributed data storage in the parallel mesh movement algorithm described in Section 4.3, the memory cost associated with storing (6.16) is much more manageable. This puts analytical differentiation at an advantage because it is a lot faster to multiply (6.16) with $\hat{v}_M$ than it is to assemble $K_{M\Delta}^{(i)}$ repeatedly. Furthermore, storing the entries in (6.16) also accelerates the block preconditioning calculations associated with the mesh equation, more details of which will be discussed in Section 6.1.3.

The second term on the RHS of (6.15) is obtained in the following manner:

$$\frac{\partial \hat{R}_{M\Delta}^{(i,n)}}{\partial u|^{(n)}} \hat{z}_S = D_{(1,n)}^{-1/2} K_{M\Delta f}^{(1,n)} \mathbf{P}^T \left( i_{m_\Delta} \frac{\partial b_\Delta^{(m_\Delta,n)}}{\partial u|^{(n)}} \right) \frac{\partial u|^{(n)}}{\partial u|^{(n)}} \hat{z}_S. \quad (6.17)$$
The matrix, $K_{(i,n)}^{(m,n)}$, used for the implicit boundary force vector is computed and stored for the same reasons discussed for (6.16). The terms $\partial u_A / \partial u$ and $\partial b_{\Delta} / \partial u_A$ are partial derivatives related to the displacement transfer and the least-squares fitting of the deflected geometry, respectively. Both operations are linear, hence

$$\frac{\partial b_{\Delta}^{(m,n)}}{\partial u_A^{(n)}} \frac{\partial u_A^{(n)}}{\partial u} \tilde{z}_S = b_{\Delta}^{(m,n)}(u_A(u = \tilde{z}_S)).$$

(6.18)

In other words, the partial derivative term above can be calculated with analytical accuracy by performing the displacement transfer and surface fitting procedure with $z_S$ in place of the structural state $u$. The cost of evaluating (6.17) is equivalent to the cost of evaluating the implicit force vectors in $R_{\Delta}$. Among the Jacobian matrix-vector products with the structural equations, $r_{scl, S}(\partial R_S / \partial u)\tilde{z}_S$ is simply $r_{scl, S}K_S\tilde{z}_S$ in a three-field formulation, where the linear structural stiffness matrix $K_S$ is explicitly formed at the start of each nonlinear aerostructural analysis. The off-diagonal Jacobian blocks in $\hat{A}$ which correspond to $R_S$ are due to the structural forces. The matrix-vector products with these terms can be approximated as

$$- r_{scl, S} \frac{\partial f_S^{(n)}}{\partial \{q, b_{\Delta}\}^{(n)}} \tilde{z}_M \approx -r_{scl, S} \left[ f_S(q^{(n)} + \delta_{FD} \tilde{z}_A, b_{\Delta}^{(n)} + \delta_{FD} \tilde{z}_M) - f_S(q^{(n)}, b_{\Delta}^{(n)}) \right],$$

(6.19)

as the exact evaluation of these terms does not appear to be particularly advantageous. The same perturbation parameter, $\delta_{FD}$, from (6.13) can be reused in (6.19).

The test problem described on Page 74 is used to compare the described Jacobian matrix-vector product calculation with a number of alternatives. The resulting nonlinear residual convergence is included in Figure 6.6 and the computational time required in Table 6.2. The preconditioner used for
Figures 6.4 and 6.5 is also used here. The differences in the total Newton-Krylov (NK) solution time listed in the second column in Table 6.2 are due to the computational time differences in the matrix-vector product evaluation and in the mesh block preconditioning calculations, which are listed in the third and fourth columns in Table 6.2 respectively. The described approach is labelled as “Default”. The other options considered include a first- and a second-order finite-difference (FD) approximation of the full Jacobian matrix-vector product with $\hat{A}$, with the exception of $r_{scl, S}(\partial R_s/\partial u)\hat{z}_S$, which is computed using $K_S$. The last method is introduced to confirm that the results of (6.13) and (6.19) are sufficiently accurate with the present choice of $\delta_{FD}$. It is based on the default matrix-vector product calculation with a few modifications: the finite-difference approximation to $r_{scl, A}(\partial R_A/\partial q)\hat{z}_A$ is replaced by an exact Jacobian matrix-vector product; the remaining terms in (6.13) and (6.19) are approximated by second-order finite-differences, with $q$ and $b_\Delta$ perturbed independently using stepsizes that are optimal for $\hat{z}_A$ and $\hat{z}_M$, respectively. This method is labelled as “Default modified” in Figure 6.6 and Table 6.2.

Figure 6.6 shows that all methods for evaluating the Jacobian matrix-vector products result in similar nonlinear convergence. This means that the default matrix-vector product calculation is equally accurate as the other alternatives considered. For the results in Table 6.2, the differences between the default calculation and the finite-difference approximations reflect the extra matrix assembly and factorization time incurred by not storing $\partial R_M(\partial b)/\partial b$ in memory. The mesh block preconditioning calculations, as will be detailed in Section 6.1.3 involve iterative solutions to the mesh subproblem that also require $\hat{R}_M/\partial b$. Therefore, storing the mesh Jacobian matrix and its factored preconditioner can reduce significantly the computational time associated with evaluating the matrix-vector products and the mesh block preconditioner. This demonstrates the importance of storing $\partial R_M(\partial b)/\partial b$, especially when multiple increments are used for analysis. For the test problem, which uses $6 \times 6 \times 6$ control points per block, this translates to an extra 0.9 megabyte of data per processor for each increment after the first one. It includes the local zero fill factorization for the preconditioner and is not a substantial storage requirement in most cases. The differences between the default calculation and the modified calculation are partly due to the cost of differentiating $\partial R_A/\partial q$ as reported earlier, and partly due to the extra flow residual evaluations needed for $(\partial R_A/\partial b)\hat{z}_M$. These results therefore suggest that the proposed matrix-vector product calculation is the most efficient choice for a similar level of accuracy.

6.1.3 Preconditioner

An effective preconditioner for the linear system solution is the key to an efficient Newton-Krylov solution algorithm. However, the approximate Schur preconditioner, which has worked well for the flow, mesh, and structural solvers in the present framework, would not be suitable for the coupled linear system given by (6.8). Domain decomposition for each of the three equations involves treating the subproblem on each processor as a subdomain. It is unclear how a Schur complement preconditioner should be defined for the coupled aerostructural problem using the same form of domain decomposition. The finite-difference discretization of the flow problem leads to a vertex-based partitioning of all the unknowns on each processor, while an element-based partitioning is used for the mesh and structural problems [109]. Due to the difference in partitioning, the Schur complement matrix for the flow problem, as discussed in Section 3.1.3 is assembled differently from the Schur complement matrix for the mesh and structure problems, as discussed in Sections 4.3 and 3.2.2 respectively. Furthermore, the structural Schur complement matrix is part of a direct factorization procedure, but the flow and mesh movement modules use the approximate Schur complement as a preconditioner. Resolving these differences is not
6.1. Aerostructural Analysis using a Newton-Krylov Method

A trivial task without making a distinction between the three equations in $R_{AS}$. Compared to a Schur complement preconditioner, the Schwarz preconditioner is more tolerant of the differences between subdomains. It is possible to define a Schwarz preconditioner for the coupled problem using the existing domain decomposition in the flow, mesh, and structural subproblems. However, it will not be able to reuse the direct factorization of the structural stiffness matrix, which may compromise the convergence of the coupled problem. This again highlights the importance of modularity in designing a preconditioner for the present problem.

Modularity can be introduced into the preconditioning procedure by treating each of the flow, mesh, and structural subproblems as a subdomain, whereby the local preconditioner can be tailored to each subproblem. The subdomains have no overlap in this case. Hence an additive or multiplicative Schwarz preconditioner defined in this manner is equivalent to a block Jacobi or block Gauss-Seidel preconditioner, respectively, as adopted by a number of authors in the literature [30, 54, 67, 73]. The Schur-Newton-Krylov approach by Barcelos et al. [51] further suggests the possibility of a Schur complement preconditioner that preserves the modularity between the three subproblems. This is discussed in Appendix C along with reasons why it is not used in the present methodology. The objective of this section is to identify an effective preconditioner that maximizes the efficiency and robustness of the Newton-Krylov solution algorithm. The performance of the block Jacobi and block Gauss-Seidel preconditioners with different local preconditioning parameters will be investigated.

Block Jacobi Preconditioner

Given an input vector $z = [z_A, z_M, z_S]^T$, a block Jacobi preconditioner returns

$$w = \begin{bmatrix} w_A \\ w_M \\ w_S \end{bmatrix} = \begin{bmatrix} c_{scl,A}^{-1} M_A^{-1} r_{scl,A} z_A \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ c_{scl,M} M_M^{-1} r_{scl,M} z_M \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c_{scl,S} M_S^{-1} r_{scl,S} z_S \end{bmatrix}, \quad (6.20)$$

where

$$M_A^{-1} \approx \left( \frac{\partial R^{(n)}_A}{\partial q^{(n)}} \right)^{-1}, \quad M_M^{-1} \approx \left( \frac{\partial R^{(n)}_M}{\partial \delta^{(n)}} \right)^{-1}, \quad M_S^{-1} \approx \left( \frac{\partial R^{(n)}_S}{\partial u^{(n)}} \right)^{-1}, \quad (6.21)$$

are the preconditioner matrices for the diagonal blocks in the coupled aerostructural Jacobian matrix, $\hat{A}$. Equation (6.21) is written in the present form to highlight that the preconditioned vectors, $w_A, w_M,$ and $w_S$, can be computed without any communication between subproblems, so they can in theory be evaluated in parallel. This is not currently permitted in the present framework because the flow and structural modules share the same group of processors. However, the potential performance of the block Jacobi preconditioner with no such constraint will be quantified in later studies. For the subsequent discussions, it is also convenient to define

$$\hat{z}_A = r_{scl,A}^{-1} z_A, \quad \hat{z}_M = r_{scl,M}^{-1} z_M, \quad \hat{z}_S = r_{scl,S}^{-1} z_S, \quad (6.22)$$

and drop the Newton iteration index $n$ from (6.21).

Existing linear system solution routines serve as effective block preconditioners. The flow module has a sophisticated iterative solution algorithm for

$$\frac{\partial R_A}{\partial q} \Delta q = -\hat{R}_A \quad (6.23)$$

which was described in Section [3.1.3] as part of the nonlinear flow analysis capabilities. The same routine
can be used to compute $M_A^{-1} \hat{z}_A$ by replacing $\Delta \mathbf{q}$ and $-\hat{R}_A$ in the above equation with $c_{\text{cl},A} \mathbf{w}_A$ and $\hat{z}_A$, respectively. Likewise, $M_M^{-1} \hat{z}_M$ can be calculated using existing routines in the mesh movement module for the solution of

$$\hat{K}^{(i)}_{M\Delta} \Delta \mathbf{b}^{(i)} = \hat{r}^{(i)}_{M\Delta},$$

for increment $i = 1, \ldots, m_\Delta$, as discussed in Section 4.3. More specifically, the mesh Jacobian block on the diagonal of $\hat{A}$, which $M_M$ should approximate, expands to

$$\frac{\partial \hat{R}_{M\Delta}}{\partial \mathbf{b}_{\Delta}} = D_M^{-1/2} \frac{\partial R_{M\Delta}}{\partial \mathbf{b}_{\Delta}} D_M^{-1/2} = \begin{bmatrix} \hat{K}^{(1)}_{M\Delta} & 0 & \cdots & 0 & 0 \\ \frac{\partial \hat{R}^{(2)}_{M\Delta}}{\partial \mathbf{b}^{(2)}_{\Delta}} & \hat{K}^{(2)}_{M\Delta} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \hat{R}^{(m_\Delta)}_{M\Delta}}{\partial \mathbf{b}^{(m_\Delta-1)}_{\Delta}} & \hat{K}^{(m_\Delta)}_{M\Delta} \end{bmatrix}$$

for $m_\Delta > 1$. The subblocks in (6.25) are precomputed and stored for the matrix-vector product calculations discussed in Section 6.1.2. Given $\hat{z}$, the preconditioned vector, $w_M$, for the mesh subproblem can be obtained from (6.25) using block forward-substitution and the solution routines for (6.24). The direct factorization of the structural stiffness matrix, $K_S$, can be used to compute $M_S^{-1} \hat{z}_S$.

The preconditioning operations with $M_A$ and $M_M$ both invoke iterative solution routines in the flow and mesh modules. The respective tolerance used in each case can be a useful parameter. It is not necessary to evaluate $M_A^{-1} \hat{z}_A$ and $M_M^{-1} \hat{z}_M$ to a tighter tolerance than the linear solution tolerance for the coupled problem, which is 0.01, as given on page 73. A larger tolerance for $M_A^{-1} \hat{z}_A$ and $M_M^{-1} \hat{z}_M$ leads to a cheaper preconditioner, but it also increases the number of coupled Krylov iterations. The increase in iterations is expected to overshadow the reduction in the cost of the preconditioner at some point, especially given that the costs per evaluation are fixed for the Jacobian matrix-vector product and the structural block preconditioner. The optimal tolerances for $M_A^{-1} \hat{z}_A$ and $M_M^{-1} \hat{z}_M$ are investigated in the first 6 sets of results in Figure 6.7. The iterative solutions to the linearized flow problem in (6.23) and the linear elasticity mesh movement problem in (6.24) further require their own preconditioners. More specifically, the linearized flow problem uses an approximate Schur preconditioner with local ILU(2) factorization, and the mesh solution is preconditioned by an additive Schwarz preconditioner with local ILU(0) factorization. These preconditioning routines may also be used to compute $M_A^{-1} \hat{z}_A$ and $M_M^{-1} \hat{z}_M$. This could reduce the block preconditioning cost substantially. It may also save the memory required for the additional layer of nested iterative solvers. The use of such flow and mesh block preconditioners are examined in the last 3 sets of results in Figure 6.7. This investigation is conducted using the test problem introduced on page 74. Calculations for the flow and mesh block preconditioners are performed on 112 Intel Xeon E5540 processors, whereas those involving the structural subproblem are distributed over 8 processors. Each processor operates at 2.53GHz and has 2GB of memory available.

The bar graph in Figure 6.7 illustrates the breakdown of computational time required by various operations during the Newton-Krylov solution. It shows the time required by the block preconditioning calculations for each equation, the Jacobian matrix-vector product evaluation, and other operations such as file output. The table in Figure 6.7 lists the preconditioning parameters used for the corresponding bar graph. The total number of coupled FGMRES iterations over the course of the Newton-Krylov solution and the maximum number of FGMRES iterations required per Newton iteration are also included.
6.1. Aerostructural Analysis using a Newton-Krylov Method

Figure 6.7: Comparison of different flow ($\mathbf{M}_A^{-1}$) and mesh ($\mathbf{M}_M^{-1}$) block preconditioners in the block Jacobi preconditioner for the coupled problem. The first six sets of results from the left use iterative solvers for $\mathbf{M}_A^{-1}$ and $\mathbf{M}_M^{-1}$. The iterative solution tolerances are listed in the first two rows in the table. The last three sets of results use the Schur complement preconditioner in the flow solver as the flow block preconditioner. The corresponding entries in the first row indicate the local factorization used. The last two rows describe the total and the maximum number of coupled FGMRES iterations during the Newton-Krylov solution of the coupled problem. The dashed lines show an estimated ideal time if all block preconditioning operations can be performed in parallel.

When linearized flow and mesh solution routines are used as the flow and mesh block preconditioners, the coupled problem converges after 7 Newton iterations in all 6 cases. Figure 6.7 shows that a solution tolerance of 0.1 for $\mathbf{M}_A^{-1}\hat{\mathbf{z}}_A$ and 0.01 for $\mathbf{M}_M^{-1}\hat{\mathbf{z}}_M$ is the most efficient. A tolerance of 0.01 for both $\mathbf{M}_A^{-1}\hat{\mathbf{z}}_A$ and $\mathbf{M}_M^{-1}\hat{\mathbf{z}}_M$ leads to the smallest number of coupled Krylov iterations, but the overall solution time is increased due to the cost of the flow block preconditioner. Hence it may be desirable to relax the tolerance for $\mathbf{M}_A^{-1}\hat{\mathbf{z}}_A$ as much as possible when the size of the flow subproblem is much larger than the sizes of the mesh and structural subproblems. However, a flow tolerance greater than or equal to 0.5 is also not recommended, as it results in a large number of coupled FGMRES iterations regardless of the tolerance for $\mathbf{M}_M^{-1}\hat{\mathbf{z}}_M$. This increases the memory required by FGMRES, which scales with the number of iterations. Experience also suggests that a loose tolerance for $\mathbf{M}_A^{-1}\hat{\mathbf{z}}_A$ or $\mathbf{M}_M^{-1}\hat{\mathbf{z}}_M$ can lead to an unreasonably large number of coupled iterations when the size of the flow subproblem or the degree of aerostructural coupling increases, indicating that the robustness of the monolithic solution algorithm has been compromised.

Based on the above observations, it can be anticipated that the preconditioners for the iterative flow and mesh solvers are likely not effective preconditioners for the flow and mesh blocks in (6.21). This is indeed the case. No results are shown in Figure 6.7 using the preconditioners for the mesh solution to compute $\mathbf{M}_M^{-1}\hat{\mathbf{z}}_M$ because over 200 coupled FGMRES iterations are required per Newton
iteration, regardless of the choice of the mesh preconditioner. This has resulted in poor performance of the Newton-Krylov solution algorithm in terms of computational time. For the last 3 sets of results in Figure 6.7 the parallel PCG solver for (6.24) is used to evaluate $M_M^{-1} \hat{z}_M$, while the approximate Schur preconditioner for (6.23) is used for $M_A^{-1} \hat{z}_A$. These additional results show the same disadvantage as using a loose tolerance for the linearized flow solution routine in the first 6 set of results in Figure 6.7. However, 10 Newton iterations, instead of 7, are now necessary. Increasing the level of fill in the local ILU factorization for the approximate Schur preconditioner leads to minor improvements in the number of iterations but not in the computational time. From a memory usage perspective, FGMRES is used for the iterative solution to (6.23) and the coupled linear system. A maximum Krylov subspace size of 30 is required to compute $M_A^{-1} \hat{z}_A$ to a tolerance of 0.1. Doing so reduces the Krylov subspace size needed for the coupled problem from 82 to 17 relative to the case when the approximate Schur preconditioner alone is used to obtain $M_A^{-1} \hat{z}_A$. Using the iterative flow solver as the flow block preconditioner in fact leads to a smaller memory requirement. These results therefore suggest that it is advantageous to use an iterative solution procedure for both $M_A^{-1} \hat{z}_A$ and $M_M^{-1} \hat{z}_M$.

Figure 6.7 also includes an estimate of the Newton-Krylov solution time taking into account the advantage in parallelism as permitted by the block Jacobi preconditioner. This ideal solution time is indicated by dashed lines in the bar graph. It is obtained by adding the time required by matrix-vector product evaluations and other operations to the highest of the flow, mesh, or structural block preconditioning time. In this case, using a tolerance of 0.1 for $M_A^{-1} \hat{z}_A$ and a tolerance of 0.01 for $M_M^{-1} \hat{z}_M$ appears to be the most efficient for the chosen test problem. However, the ideal time does not factor in the use of additional processors dedicated to solving the mesh subproblem.

**Block Gauss-Seidel Preconditioner**

Multiplicative Schwarz and block Gauss-Seidel preconditioners are not often used in domain decomposition because the local preconditioner on each subdomain must be applied in serial. The parallelism in an additive Schwarz or a block Jacobi preconditioner, however, comes from not accounting for the coupling between subdomains. More of such coupling can be included in a multiplicative Schwarz preconditioner, often resulting in a much smaller number of iterations. Full parallelism is less crucial in the present approach where the Schwarz preconditioner is applied to three subproblems instead of a few hundred processors. It is also more expensive to evaluate the local preconditioners in this case. Hence it seems more logical to maximize the effectiveness of each block preconditioner evaluation by using a multiplicative Schwarz or block Gauss-Seidel preconditioner.

Application of the block Gauss-Seidel preconditioner involves solving the following equations in sequence [109]:

\[
\begin{align*}
   w_M &= c_{sc1,M}^{-1} M_M^{-1} \left( r_{sc1,M}^{-1} z_M \right), \\
   w_A &= c_{sc1,A}^{-1} M_A^{-1} \left( r_{sc1,A}^{-1} z_A - \frac{\partial R_A}{\partial \delta} \left( r_{sc1,M}^{-1} w_M \right) \right), \\
   w_S &= c_{sc1,S}^{-1} M_S^{-1} \left( r_{sc1,S}^{-1} z_S - \frac{\partial R_S}{\partial q} \left( r_{sc1,A}^{-1} w_A \right) - \frac{\partial R_S}{\partial \delta} \left( r_{sc1,M}^{-1} w_M \right) \right),
\end{align*}
\]

where $M_A^{-1} \hat{z}_A$, $M_M^{-1} \hat{z}_M$, and $M_S^{-1} \hat{z}_S$ are computed in the same manner as for the block Jacobi preconditioner. Evaluation of the extra matrix-vector product terms in $\hat{z}_A$, $\hat{z}_M$, and $\hat{z}_S$ is as discussed in
6.1. Aerostructural Analysis using a Newton-Krylov Method

Figure 6.8: Comparison of different flow ($M^{-1}_A$) and mesh ($M^{-1}_M$) block preconditioners in the block Gauss-Seidel preconditioner for the coupled problem. The first six sets of results from the left use iterative solvers for $M^{-1}_A$ and $M^{-1}_M$. The iterative solution tolerances are listed in the first two rows in the table. The last three sets of results use the Schur complement preconditioner in the flow solver as the flow block preconditioner. The corresponding entries in the first row indicate the local factorization used. The third and fourth rows describe the total and the maximum number of coupled FGMRES iterations during the Newton-Krylov solution of the coupled problem. The last row shows the total number of coupled FGMRES iteration using the block Gauss-Seidel preconditioner normalized by the total number of coupled FGMRES iterations using the block Jacobi preconditioner.

Section 6.1.2, and they will increase the time required per coupled FGMRES iteration. Figure 6.8 presents a similar investigation for the block Gauss-Seidel preconditioner as Figure 6.7 for the block Jacobi preconditioner. The time to calculate $\tilde{z}_A$, $\tilde{z}_M$, and $\tilde{z}_S$ is included in the time required by the corresponding block preconditioner in the bar graphs.

Although it is not shown in Figure 6.8, the use of a block Gauss-Seidel preconditioner does not change the number of Newton iterations required for the coupled analysis. In most cases, the number of coupled FGMRES iterations per Newton iteration is reduced. The last row in the table in Figure 6.8 shows the ratios between the total number of FGMRES iterations needed for the block Gauss-Seidel preconditioner and that required by the block Jacobi preconditioner. In particular, with a tolerance of 0.01 for both $M^{-1}_A\tilde{z}_A$ and $M^{-1}_M\tilde{z}_M$, the coupled linear system at each Newton step is solved in at most 4 Krylov iterations, which is less than half of the number of iterations needed using the block Jacobi preconditioner. This also leads to the fastest solution time among all results presented in Figure 6.8. Using a tolerance of 0.1 for $M^{-1}_A\tilde{z}_A$ in with a tolerance of 0.01 for $M^{-1}_M\tilde{z}_M$ also appears to be very efficient. It is found to be a more suitable combination of preconditioning parameters when the size of the flow subproblem increases relative to that of the mesh subproblem. The effectiveness of the block
Gauss-Seidel preconditioner in terms of the number of coupled FGMRES iterations decreases with the use of cheaper flow and mesh block preconditioners. When the approximate Schur preconditioner for the flow problem is used to evaluate $\mathbf{M}_A^{-1} \mathbf{z}_A$, the block Gauss-Seidel preconditioner needs roughly the same number of coupled FGMRES iterations as the block Jacobi variant. Given the extra cost in evaluating the block Gauss-Seidel preconditioner per coupled FGMRES iteration, it is no longer a competitive option in these cases. This suggests that the quality of the block preconditioners is more crucial in a block Gauss-Seidel preconditioner than in a block Jacobi preconditioner. Furthermore, the earlier conclusion that iterative solvers should be used as block preconditioners still holds.

Among the results from both Figures 6.7 and 6.8, the best Newton-Krylov solution time, which is 39.2s, is from using the ideal block Jacobi preconditioner. Recall that this ideal time estimate assumes that the flow, mesh, and structural block preconditioning can be performed in parallel. The second best Newton-Krylov solution time is 46.8s using the block Gauss-Seidel preconditioner. Despite this observation, there are a number of arguments favouring the use of the block Gauss-Seidel preconditioner. For the recommended range of tolerances between 0.1 and 0.01 for the flow and mesh block preconditioners, performance of the block Gauss-Seidel is more stable and less sensitive to the choice of block preconditioning tolerance. In two out of the first three sets of results in Figures 6.7 and 6.8 the block Gauss-Seidel preconditioner is more efficient than the block Jacobi preconditioner, even in the ideal scenario. The use of a block Gauss-Seidel preconditioner can improve the robustness of the solution algorithm by making the number of coupled FGMRES iterations less dependent on the size and complexity of the analysis. The reduced maximum Krylov subspace size is also advantageous from a memory standpoint. Finally, the estimated ideal block Jacobi time equivalently assumes that an additional 112 processors are allocated for the mesh calculations. It is unclear whether the ideal block Jacobi preconditioner would remain the most efficient if the total number of processors were to remain unchanged from the present implementation. For the above reasons, the block Gauss-Seidel preconditioner is a more effective way to precondition the Newton-Krylov solution algorithm.

### 6.1.4 Detailed Cost Comparison between Monolithic and Partitioned Analysis

Figure 6.9 plots the nonlinear residual convergence for the full analysis of the test problem using the monolithic method described in this chapter and the partitioned method from Chapter 5. The computational time required by each method is normalized by the cost of a flow analysis on the same grid assuming no structural deflections, which takes approximately 40s. Block Jacobi (bJac) and block Gauss-Seidel (bGS) preconditioners are considered for the monolithic analysis, both with a tolerance of 0.1 for $\mathbf{M}_A^{-1} \mathbf{z}_A$ and 0.01 for $\mathbf{M}_M^{-1} \mathbf{z}_M$. Each coupled nonlinear iteration is represented by a marker in Figure 6.9. The partitioned method converges in 25 iterations. The first 5 partitioned iterations overlap with the start-up iterations in the monolithic analysis before it switches to the Newton-Krylov solution algorithm, after which only 7 additional iterations are needed. Figure 6.9 shows that the monolithic solution method improves the efficiency of the coupled analysis by almost 50% relative to the partitioned method. However, an important question remains as to what factors contribute to the savings in computational time. Such insights are valuable to the development of more sophisticated coupled solution algorithms in the future. This motivates a detailed cost comparison between the monolithic and partitioned analysis. A higher level performance comparison between the monolithic and partitioned methods for a wider range of test conditions will be presented in the next chapter.
Aerostructural Analysis using a Newton-Krylov Method

Figure 6.9: Convergence of $\|R_A\|_2$, $\|R_M\|_2$, and $\|R_S\|_2$ using a partitioned method and a monolithic method where the Newton-Krylov solution uses a block Jacobi and a block Gauss-Seidel preconditioner. Computational time on the x-axis is normalized by the cost of a flow analysis assuming a rigid structure.

<table>
<thead>
<tr>
<th></th>
<th>bGS Precond.</th>
<th>bJac Precond.</th>
<th>Partitioned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time per coupled nonlinear iter. (s)</td>
<td>6.7</td>
<td>8.4</td>
<td>9.0</td>
</tr>
<tr>
<td>Average time for subproblem calculations (s)</td>
<td>Flow 3.68 Mesh 1.56</td>
<td>Flow 4.20 Mesh 2.62</td>
<td>Flow 6.16 Mesh 0.76</td>
</tr>
<tr>
<td>Average time for linear system solutions (s)</td>
<td>3.00</td>
<td>1.22</td>
<td>2.88</td>
</tr>
<tr>
<td>Average Number of linear system solutions</td>
<td>5.3</td>
<td>15.9</td>
<td>9.3</td>
</tr>
<tr>
<td>Krylov iter. per coupled nonlinear iter.</td>
<td>101</td>
<td>1935</td>
<td>129</td>
</tr>
<tr>
<td>Krylov iter. per linear system solution</td>
<td>19</td>
<td>122</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 6.3: The details of a Newton-Krylov iteration with a block Gauss-Seidel and a block Jacobi preconditioner, as well as the details of a partitioned iteration, in terms of computational time and linear system solutions in the flow and mesh subproblems.

A number of insights can be revealed by deconstructing each nonlinear iteration from the monolithic and partitioned methods in Table 6.3. It should first be clarified that all average values reported exclude the first 5 nonlinear block Gauss-Seidel iterations. Furthermore, only the flow and mesh subproblems are studied in detail because the costs associated with them are more heavily influenced by the decisions made in this chapter. However, some conclusions from the subsequent discussions are also applicable to the structural subproblem. It can first be pointed out from Table 6.3 that the average time per coupled nonlinear iteration does not differ substantially between the monolithic and partitioned analysis. Hence the efficiency benefits of using a monolithic method are mainly due to the much smaller number of coupled nonlinear iterations. The fourth row of Table 6.3 further shows that the monolithic solution method has reduced the average flow calculation time per coupled nonlinear iteration. However, it has more than doubled the average computational time related to the mesh subproblem. It is evident from the subsequent row in Table 6.3 that a large portion of the subproblem calculations involves the iterative solution to the respective linear systems. It is therefore instructive to examine the number of linear system solutions required by the flow and mesh subproblems, and the resulting number of Krylov iterations needed to obtain these solutions.

Each partitioned iteration involves a nonlinear flow analysis. The number of linear system solutions
needed by the flow module is then given by the number of nonlinear iterations in the flow analysis. In an inexact Newton iteration of the monolithic solution method, a linearized flow solution is needed to compute $M^{-1}A\hat{z}_A$ for each coupled FGMRES iteration. Table 6.3 shows that the number of linearized flow solutions is actually comparable between the block Jacobi preconditioner and the partitioned solution method. Nevertheless, a tolerance of 0.1 works sufficiently well as a flow block preconditioner in the present example, whereas the linearized flow problem must be solved to a tighter tolerance in the partitioned method. This reduces the total number of Krylov iterations needed within the flow module for the block Jacobi preconditioner. This is partly why the average flow calculation time is 4.20s for each Newton-Krylov iteration with a block Jacobi preconditioner, as shown in the fourth row of Table 6.3, while it is 6.16s for each partitioned iteration. The difference in flow calculation time can additionally be explained by the fact that each nonlinear flow iteration during the partitioned analysis requires forming and factoring a new preconditioner matrix for the linear system solution. During a monolithic iteration, the same preconditioner is used for all linearized flow solutions. For the block Gauss-Seidel preconditioner, the number of linearized flow solutions required is quite low compared to the block Jacobi preconditioner and the partitioned method. However, it is also found that evaluating $M^{-1}A\hat{z}_A$ for the block Gauss-Seidel preconditioner often requires more Krylov iterations in the flow module, even though the same tolerance is used as the block Jacobi preconditioner. It could be attributed to the extra matrix-vector product in $\hat{z}_A$, as shown in (6.20). As a result, the flow calculation time reported in the fourth row of Table 6.3 for the block Gauss-Seidel preconditioners is not as low as the number of coupled FGMRES iterations would suggest.

Table 6.3 shows that the reduction in flow calculation time via the use of a monolithic solution method is somewhat counterbalanced by the increase in mesh calculation time. Each nonlinear block Gauss-Seidel iteration solves 3 linear elasticity mesh movement equations to move the mesh in 3 increments. Each mesh equation is solved to a relative tolerance of $10^{-6}$ in order to ensure the grid quality for the subsequent flow analysis. In contrast, the number of linear mesh solutions per inexact Newton iteration during monolithic analysis is given by 3 times the total number of coupled FGMRES iterations. The tolerance can be much more relaxed when the mesh solution is only used as a block preconditioner during monolithic analysis. Therefore, the last row of Table 6.3 shows that the number of PCG iterations per mesh solution is lower for the monolithic methods than for the partitioned method. Even then, the monolithic methods still require more than twice the number of PCG iterations in total per coupled nonlinear iteration. This is due to the much higher number of linear system solutions needed compared to a partitioned iteration. The above observations help explain why the efficiency of the mesh solution is of crucial importance to the performance of the monolithic solution method, and why the matrix assembly time can become significant, as demonstrated in Table 6.2.

Summarizing the key points from this investigation, the cost per inexact Newton iteration relative to a partitioned iteration is largely related to the number of linear system solutions required by different subproblems using either method. In the case of the present framework, the structural analysis is linear, and the mesh movement is pseudo-nonlinear, neither of which require any nonlinear iterations during a partitioned analysis. Therefore, the costs associated with the mesh and structural subproblems are expected to increase in an inexact Newton iteration, because the number of mesh and structural linear system solutions now depends on the number of coupled Krylov iterations needed to solve (6.3). The flow analysis, on the other hand, requires nonlinear iterations during a partitioned iteration. A monolithic method can reduce the flow calculation time per coupled nonlinear iteration by using an
effective preconditioner to maintain a low number of coupled Krylov iterations. However, such reduction is offset by the increase in computational time required by the mesh and structural subproblems. This has a number of implications. First of all, the coupled analysis can potentially benefit more from using a monolithic solution strategy if the flow subproblem is much larger in size than the mesh and structural subproblems. Secondly, the use of monolithic analysis can be more advantageous with the use of nonlinear structural analysis or a nonlinear mesh movement strategy.

6.2 Monolithic Coupled Adjoint Solution

Recall from (5.17) in Section 5.2 that the coupled adjoint problem has the following form:

\[
\begin{bmatrix}
\frac{\partial R_A}{\partial q}^T & 0 & \frac{\partial R_S}{\partial q}^T \\
\frac{\partial R_A}{\partial b} & \frac{\partial R_{M\Delta}}{\partial b} & \frac{\partial R_S}{\partial b} \\
0 & \frac{\partial R_{M\Delta}}{\partial u} & \frac{\partial R_S}{\partial u}
\end{bmatrix}
\begin{bmatrix}
\Psi_A \\
\Psi_{M\Delta} \\
\Psi_S
\end{bmatrix}
= 
\begin{bmatrix}
-\frac{\partial J}{\partial q}^T \\
-\frac{\partial J}{\partial b}^T \\
-\frac{\partial J}{\partial u}^T
\end{bmatrix}.
\]

Equation (6.27) is a linear system of equations which can be solved monolithically using a Krylov subspace method. The LHS of (6.27) is the transpose of \(A\), which is the Jacobian matrix of \(R_{AS}\) on the LHS of the linear system solved at every inexact Newton iteration during analysis. Therefore, \(A^T\) is also a block \(3 \times 3\) matrix and (6.27) corresponds to a large, sparse, and nonsymmetric system of equations. Based on observations from the previous section, the use of existing iterative solvers as block preconditioners will likely work well for the coupled adjoint problem. This also means that the iterative solution to (6.27) must allow for a nonstationary preconditioner. Furthermore, (6.27) needs to be solved to a tolerance of at least \(10^{-8}\) in order to ensure gradient accuracy [44, 89], unlike the linear system solutions during an analysis. Although FGMRES could be used, this requirement could potentially translate into a very large Krylov subspace size. To avoid a further increase in the memory requirement by a monolithic solution method in comparison to a partitioned method, it is likely desirable to use a restarted FGMRES for the solution to (6.27). Nonetheless, restarting is known to compromise the robustness of the iterative solution in some cases [89].

In order to address these issues, the present methodology solves the coupled adjoint system using GCROT\((m, k)\) by Hicken and Zingg [93], which is a simplified and flexible variant of GCROT. GCROT stands for generalized conjugate residual with inner orthogonalization and outer truncation, while \(m\) and \(k\) are the inner and outer subspace sizes, respectively. FGMRES is used as the inner method which allows for the use of iterative methods as a preconditioner. The first outer iteration of GCROT\((m, k)\) is equivalent to FGMRES with an inner subspace size of \(m + k\). To maintain a fixed memory usage, the inner subspace size decreases incrementally with each outer iteration to make room for a growing set of outer vectors. Upon reaching the \(k\)th outer iteration, the oldest outer vector is discarded as a truncation strategy in subsequent iterations [93]. GCROT\((m, k)\) minimizes the residual over the inner and outer sets of vectors. It increases the robustness of the iterative solution relative to the restarted FGMRES\((m)\), which fully discards the Krylov subspace upon restarting [89]. Hicken and Zingg [93] showed that GCROT\((m, k)\) is more effective than the flexible variants of other truncated Krylov subspace methods.
including LGMRES and BiCGStab for flow adjoint calculations, especially for flows with a higher Mach number. It has also been successfully applied to aerodynamic shape optimization involving both inviscid and viscous turbulent calculations.

The iterative solution to the coupled adjoint problem using GCROT($m, k$) requires the matrix-vector products with $A^T$. The matrix-vector products with the individual transposed Jacobian blocks in $A^T$ are also part of (6.19–6.20). Therefore, the calculations can be reused from the coupled adjoint solution using a block Gauss-Seidel method. The remainder of this section will focus on the effects of scaling and the use of different preconditioners for the monolithic coupled adjoint solution. To simplify the discussions, it is temporarily assumed that the inner Krylov subspace size is large enough to converge the coupled adjoint problem to the desired tolerance. The values of $m$ and $k$ are otherwise left unspecified until the appropriate choice of these values is recommended at the end of this section. The test problem introduced in the previous section is used again here for demonstrations.

### 6.2.1 Scaling

The coupled adjoint problem can be scaled in a similar manner as (6.8) for analysis, which involves scaling within each subproblem and scaling of the Jacobian blocks. Due to the symmetry in the mesh stiffness matrices, the rows and columns associated with the mesh subproblem are again scaled such that $\hat{R}_{M\Delta} = D_M^{-1/2}R_{M\Delta}$ and $\hat{b}^{(i)}_{\Delta} = D_M^{1/2}b_{\Delta}$, where $D_M$ is defined in (6.9). The original flow and structural adjoint equations introduced in Chapter 3 do not include any scaling, so the related equations and variables are left unscaled within their respective subproblems. The conclusions from Section 6.1.1 on the relative scaling between the Jacobian blocks in $A$ are still applicable here. The norm-based scaling adopted for analysis is only meaningful in the context of a nonlinear solution process. Hence the differences in scaling between the Jacobian blocks in $A^T$ are adjusted instead by the automatic scaling procedure introduced in Section 6.1.1. An expression similar to (6.10) is defined for the transposed linear system:

$$
S = \begin{bmatrix}
\begin{array}{ccc}
\frac{1}{n_A} & \frac{1}{n_A} & 0 \\
\frac{1}{n_M} & \frac{1}{n_M} & 0 \\
0 & \frac{1}{n_S} & 0 \\
\end{array}
\begin{array}{ccc}
\frac{\partial R_A}{\partial q} & \frac{\partial R_A}{\partial b_{\Delta}} & \frac{\partial R_A}{\partial u} \\
\frac{\partial R_A}{\partial q} & \frac{\partial R_A}{\partial b_{\Delta}} & \frac{\partial R_A}{\partial u} \\
\frac{\partial R_A}{\partial q} & \frac{\partial R_A}{\partial b_{\Delta}} & \frac{\partial R_A}{\partial u} \\
\end{array}
\begin{array}{c}
1_A \\
1_M \\
1_S \\
\end{array}
\end{bmatrix}
\begin{bmatrix}
c_{scl,A} \\
c_{scl,M} \\
c_{scl,S} \\
\end{bmatrix}
\begin{bmatrix}
\hat{b}_{\Delta} \\
\hat{b}_{\Delta} \\
\hat{b}_{\Delta} \\
\end{bmatrix}
\begin{bmatrix}
\frac{\partial R_S}{\partial q} & \frac{\partial R_S}{\partial b_{\Delta}} & \frac{\partial R_S}{\partial u} \\
\frac{\partial R_S}{\partial q} & \frac{\partial R_S}{\partial b_{\Delta}} & \frac{\partial R_S}{\partial u} \\
\frac{\partial R_S}{\partial q} & \frac{\partial R_S}{\partial b_{\Delta}} & \frac{\partial R_S}{\partial u} \\
\end{bmatrix}
\begin{bmatrix}
1_S \\
1_M \\
c_{scl,S} \\
\end{bmatrix}
\right)
$$

(6.28)

The block row and column scaling values, $r_{scl,A}$ and $c_{scl,A}$, are chosen such that $\log_{S_{10}}(S_{ij})$ for all $S_{ij} \neq 0$ are as close to unity as possible in a least-squares sense. When computing the automatic scaling for analysis, $S_{11}$ is excluded from the calculations, and $r_{scl,A}$ and $c_{scl,A}$ are both set to unity. However, $S_{11}$, $r_{scl,A}$, and $c_{scl,A}$ are all part of the present calculation, which results in an overdetermined system with 7 equations and 6 unknowns.

Figure 6.10 shows the entries in $S$ for the test problem with and without the block Jacobian scaling. Figures 6.11 and 6.12 additionally compare the corresponding convergence of the relative linear system residual using GCROT($m, k$). The coupled adjoint convergence for an aerodynamic functional and a structural functional are each examined, where KS2 refers to the stress constraint aggregation for the bottom skin of the structures. Each GCROT($m, k$) iteration is represented by a square symbol. Recall
that the coupled adjoint RHS vector for an aerodynamic functional has the form of

$$\text{RHS} = \left[ -\frac{\partial J_A}{\partial q}, -\frac{\partial J_A}{\partial b_{\Delta}} 0 \right]^T, \quad (6.29)$$

while the RHS vector for a structural functional has the form of

$$\text{RHS} = \left[ 0 \quad 0 \quad -\frac{\partial J_S}{\partial \mathbf{u}} \right]^T. \quad (6.30)$$

It can first be observed that the differences in the RHS lead to noticeably different convergence behaviours between Figures 6.11 and 6.12. The disparity in scales between the Jacobian blocks in $A^T$ has a more noticeable effect on the convergence for the KS2 functional than for the lift functional. This is true for all structural functionals relevant to this thesis. The Jacobian block scaling deters the convergence of the coupled linear system by introducing an initially flat region to the convergence history in Figure 6.12. However, it will be shown below that the gradients resulting from the unscaled calculations are inaccurate even though the coupled adjoint system appears to converge in fewer iterations.

The coupled adjoint solution obtained using a partitioned block Gauss-Seidel method is unaffected by the differences in scaling between the three subproblems. The resulting gradients have further been verified in Section 6.2.3. Therefore, they are used as a reference in Table 6.4 to assess the accuracy of the gradients computed using the monolithic approach. The gradients with respect to the AOA, a uniform thickness variable for the entire structural model, and the $z$-component of a surface control point are examined. The partitioned coupled adjoint solution is converged to a relative tolerance of $10^{-10}$. For the lift functional, the gradient values calculated using the monolithic method agree well with the reference values with and without the Jacobian block scaling. For the KS2 functional, however, solving the coupled adjoint problem to a relative tolerance of $10^{-10}$ only produces the correct gradient values when the Jacobian block scaling is applied. Otherwise, a tighter relative tolerance of $10^{-13}$ is necessary. Examining the relative linear system residual for the three subproblems in the unscaled case has shown that both the mesh and flow adjoint equations are undersolved relative to the structural adjoint. Nevertheless, without the use of any block row and column scaling, this is not reflected in the linear system residual for the fully coupled problem, whose value has been dominated by the structural

---

Figure 6.10: Estimated scaling in $A^T$ before and after Jacobian block scaling is applied.

Figure 6.11: Coupled adjoint convergence for the lift functional.

Figure 6.12: Coupled adjoint convergence for the KS2 functional.
subproblem. This problem can be addressed by checking the linear system residual of the individual equations for convergence. However, scaling the Jacobian blocks in $A^T$ reduces the impact of round-off error [112]. For this reason, scaling is still useful for ensuring the accuracy of the monolithic coupled adjoint solution.

6.2.2 Preconditioner

As for the Newton-Krylov solution algorithm for analysis, the coupled adjoint solution using GCROT($m, k$) can be preconditioned using a block iterative strategy. In particular, a block Gauss-Seidel iteration, which involves solving (5.19–5.20), can be used as a preconditioner with the following modifications:

$$\frac{\partial R_A}{\partial q}^T (c_{\text{scl}, A} w_A) = r_{\text{scl}, A} z_A = \tilde{z}_A$$  \hspace{1cm} (6.31)

$$\frac{\partial R_M}{\partial b_\Delta}^T (c_{\text{scl}, M} w_M) = r_{\text{scl}, M} z_M - \frac{\partial R_A}{\partial b_\Delta}^T (c_{\text{scl}, A} w_A) = \tilde{z}_M$$  \hspace{1cm} (6.32)

$$M_S(c_{\text{scl}, S} w_S) = r_{\text{scl}, S} z_S - \frac{\partial R_M}{\partial u}^T (c_{\text{scl}, M} w_M) = \tilde{z}_S.$$  \hspace{1cm} (6.33)

Here $[w_A, w_M, w_S]$ is the preconditioned vector given $[z_A, z_M, z_S]$ as an input, and $M_S$ is the direct factorization of $K_S$. The approximate solutions to (6.31) and (6.32), which use existing iterative adjoint solvers in the flow and mesh modules, respectively, serve the same purpose as $M^{-1}_A \tilde{z}_A$ and $M^{-1}_M \tilde{z}_M$ during the monolithic analysis. The use of different tolerances for the evaluation of $M^{-1}_A \tilde{z}_A$ and $M^{-1}_M \tilde{z}_M$ is investigated in Figure 6.13 for the lift functional. The coupled adjoint problem is solved to a tolerance of $10^{-10}$. The bar graphs show a similar breakdown of the computational time as in Figures 6.7 and 6.8. The results for other functionals are very similar; hence they are not presented here.

A number of observations can be made from Figure 6.13. The flow block preconditioning calculation involving the solution to (6.31) is consistently much more expensive than the rest of the coupled adjoint calculations. This indicates that it is desirable to relax the tolerance for $M^{-1}_A \tilde{z}_A$ as much as possible while maintaining a small number of GCROT($m, k$) iterations. The results in Figure 6.13 show that lowering the flow block preconditioning tolerance below 0.1 while maintaining the strength of the mesh block preconditioner does very little to reduce the number of GCROT($m, k$) iterations. On the other hand, increasing the tolerance for $M^{-1}_M \tilde{z}_M$ from $10^{-4}$ to $10^{-2}$ increases the number of coupled iterations by a noticeable amount. Although not shown in Figure 6.13, using a flow block preconditioning tolerance of 0.5 increases the number of coupled Krylov iterations to 27, while reducing the tolerance for $M^{-1}_M \tilde{z}_M$ to $10^{-8}$ still results in 13 iterations. Therefore, using a tolerance of $10^{-1}$ for (6.31) with a tolerance between $10^{-4}$ and $10^{-6}$ for (6.32) seems to be optimal in that it leads to the cheapest block preconditioners while

<table>
<thead>
<tr>
<th>Lift Functional</th>
<th>KS2 Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaled</td>
<td>Tolerance</td>
</tr>
<tr>
<td>True</td>
<td>1E-10</td>
</tr>
<tr>
<td>False</td>
<td>1E-10</td>
</tr>
<tr>
<td>False</td>
<td>1E-13</td>
</tr>
<tr>
<td>True</td>
<td>1E-10</td>
</tr>
<tr>
<td>False</td>
<td>1E-10</td>
</tr>
<tr>
<td>False</td>
<td>1E-13</td>
</tr>
</tbody>
</table>

Table 6.4: Relative difference in the computed gradients using GCROT($m, k$) measured with respect to the gradients computed using a partitioned block Gauss-Seidel method.
Figure 6.13: Comparison of different solution tolerances for the flow block preconditioning calculation ($M_A^{-1}\hat{z}_A$) and the mesh block preconditioning calculation ($M_M^{-1}\hat{z}_M$). The coupled adjoint problem considered is for the lift functional. The monolithic solution time listed in the last two rows is normalized by the partitioned coupled adjoint solution time, where $\theta$ is the relaxation parameter used for the partitioned method.

Table 6.5 includes a more detailed computational cost comparison between the coupled adjoint solution for the lift and KS2 functionals using a monolithic and a partitioned method. The flow and mesh subproblems are distributed over a total of 112 processors, and a total of 8 processors are used for the structural subproblem. The total coupled adjoint solution time has been normalized with respect to the flow adjoint solution time of 31s on the same grid to indicate the relative cost of the aerostructural calculations. For all methods, each coupled adjoint iteration requires the solution to three equations in the form of (5.19–5.20). In a partitioned iteration, the flow adjoint equation is solved to a tolerance of $10^{-1}$, and the mesh adjoint equation is solved to a tolerance of $10^{-6}$. These are the same as the

still allowing the block Gauss-Seidel preconditioner to be effective.

The last rows in the table of Figure 6.13 show the monolithic adjoint solution time normalized by the time required using the partitioned method. When comparing the efficiency of the monolithic and partitioned adjoint solutions, it is necessary to consider different relaxation parameters ($\theta$), the choice of which influences both the efficiency and robustness of the partitioned calculations. The dependence of the partitioned method on $\theta$ will be illustrated more clearly in the next chapter. Using a tolerance of $10^{-6}$ for $M_A^{-1}\hat{z}_A$ and a tolerance of $10^{-1}$ for $M_M^{-1}\hat{z}_M$, the monolithic coupled adjoint solution leads to a 72% reduction in computational time relative to the partitioned method with $\theta = 0.5$. For the present test problem where the coupling is moderate, it is more efficient to use a higher value of $\theta$. A reduction of 61% in computational time is achieved when $\theta = 0.75$. Not all options considered in Figure 6.13 result in an efficiency improvement. This highlights the importance of choosing an effective preconditioner for the monolithic coupled adjoint solution.
Table 6.5: Comparison between the computational time required by the monolithic and the partitioned solution to the coupled adjoint problem. The coupled adjoint problems are solved to a tolerance of $10^{-10}$. The second column includes the total coupled adjoint solution time normalized by the time it takes to solve a flow adjoint problem to a relative tolerance of $10^{-12}$. The last five columns indicate the average time per coupled adjoint iteration.

<table>
<thead>
<tr>
<th></th>
<th>Lift functional</th>
<th>KS2 functional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Normalized Average time per iteration (s)</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>iterations time</td>
<td></td>
</tr>
<tr>
<td>Monolithic</td>
<td>13</td>
<td>1.58</td>
</tr>
<tr>
<td>Partitioned ($\theta = 0.5$)</td>
<td>48</td>
<td>5.46</td>
</tr>
<tr>
<td>Partitioned ($\theta = 0.75$)</td>
<td>29</td>
<td>4.00</td>
</tr>
</tbody>
</table>

optimal tolerances for the flow and mesh block preconditioners identified from Figure 6.13, which are also used for the results in Table 6.5. Hence the flow and mesh calculation time per coupled adjoint iteration is comparable between all methods. The time required by the structural module per adjoint iteration is much smaller for the monolithic method. This is due to using the factorization of $\mathbf{K}_S$ as the structural block preconditioner, while the structural adjoint equation is solved iteratively during a partitioned iteration. A coupled adjoint iteration in a monolithic method additionally needs to evaluate the matrix-vector product with $\mathbf{A}^T$ and perform other calculations required by the Krylov subspace method. The cost of the latter is insignificant and is not included in Table 6.5. Overall, the time required per coupled adjoint iteration is similar between the monolithic and the partitioned calculations using both relaxation parameters. Therefore, the efficiency gain from using a monolithic method is once again from reducing the number of coupled iterations. Based on this observation, the block Jacobi preconditioner is not considered for the present methodology because it generally leads to more iterations than a block Gauss-Seidel preconditioner. In terms of the coupled adjoint solutions for different functionals, more coupled Krylov iterations are usually needed for a structural functional than an aerodynamic functional, which is also reflected in Table 6.5. Meanwhile, the number of partitioned iterations seems to be less affected by whether the RHS of the coupled adjoint problem is for an aerodynamic or structural functional. The efficiency benefit of using the monolithic method is slightly less for structural functionals for this reason.

As a final remark, solving the coupled adjoint problem with a Krylov subspace method performs best when the number of coupled iterations is kept small via an effective preconditioner. For the monolithic solution strategy developed in this section, convergence is often achieved within 20 iterations. This is in contrast to a typical flow adjoint solution in aerodynamic shape optimization, which may require a substantially larger number of iterations. The potential memory requirement needed for storing a large Krylov subspace is therefore less of an issue during the coupled adjoint calculations. In reality, however,
the number of iterations required for the specified solution tolerance cannot be predicted in advance, and memory cannot be allocated for a Krylov subspace size that is arbitrarily large. When the number of iterations does exceed the maximum allowable Krylov subspace size, the ability to continue the iterative solution without significantly hindering the convergence is very important. This is especially the case given the high cost associated with each coupled adjoint iteration; thus the use of GCROT\((m,k)\) is still desirable. Using the information obtained in this section, GCROT\((m,k)\) with \(m = 20\) and \(k = 1\) should be suitable for the coupled adjoint solutions using the present methodology for aerostructural optimization.

6.3 Verification

The optimization methodology has been largely verified and validated using the block Gauss-Seidel method for analysis and coupled adjoint calculations. It is necessary to ensure that the verified results can be recovered with the use of the monolithic solution method. Therefore, the gradient verification study from Section 5.2.3 and the transonic sweep optimization study from Zhang et al. [104] are repeated in this section, and the results are summarized.

6.3.1 Gradient Verification

In repeating the gradient verification from Section 5.2.3, the Newton-Krylov solution algorithm from Section 6.1 is used for analysis, and the monolithic coupled adjoint solution strategy from Section 6.2 is used for gradient calculations. The aerostructural analysis is converged to a relative tolerance of \(10^{-9}\), while the coupled adjoint solutions are converged to a relative tolerance of \(10^{-10}\). The results are compared with those obtained using the partitioned method described in Chapter 5. Table 6.6 shows that the values of a number of important functionals are very similar between the monolithic and partitioned methods, agreeing to at least 13 significant digits. Figure 6.14 further shows that the monolithic and partitioned methods produce consistent gradient values. The normalized differences in the coupled adjoint gradient with respect to all 672 design variables, which consist of 150 structural design variables, 1 AOA design variable, and 521 geometric design variables, are examined. The normalized differences for most gradient values are on the order of \(10^{-10}\) or less, which is smaller than the minimum normalized differences measured with respect to finite-difference approximations in Section 5.2.3.

6.3.2 Inviscid Transonic Wing Sweep Optimization

The optimization methodology described in Chapter 5 has been validated through an inviscid transonic wing sweep optimization study by Shahriar Khosravi, the details of which can be found in Zhang et al. [104]. The objective of the present section differs from the original sweep optimization in that it is not intended to be a stand-alone study. Capabilities of the present methodology in conducting aerostructural optimization, including the use of efficient coupled solution strategies and the ability to capture the correct design trends, will be demonstrated in Chapter 7. The focus of the present investigation is to reproduce the results from an existing optimization case. Therefore, only the relevant details to the sweep optimization are described.

The optimization uses the same geometry and structural layout as the test problem used throughout this chapter, which are included in Figures 6.1 and 6.2. The objective function for the optimization is a
linear combination of drag, \( D \), and weight, \( W \), normalized by their respective initial values, \( D_0 \) and \( W_0 \):

\[
J = \beta \frac{D}{D_0} + (1 - \beta) \frac{W}{W_0}.
\]

(6.34)

Different emphasis may be placed on the drag and weight during an optimization by varying the parameter \( \beta \in [0, 1] \). Three values of \( \beta \) are considered: 0.5, 0.75, and 1.0. The optimization includes a cruise and a 2.5g maneuver condition. The cruise Mach number is 0.785, and the cruise altitude is 35,000 feet. The Mach number and altitude for the maneuver condition are 0.798 and 12,000 feet, respectively. Each load condition is subject to a lift constraint. The maneuver condition, which is used to size the structure, is additionally subject to three failure constraints given by KS functions. The three KS functions correspond to the failure criteria in the elements belonging to the ribs and spars, top skin, and bottom skin, respectively. There are two AOA design variables, one for each load condition. There are also 226 geometric design variables, allowing for changes in quarter-chord sweep angle, tip twist of the wing, and the section shape at 16 stations along the span. Finally, the thickness distribution in the structural components of the wing is controlled by 156 structural design variables.

The optimization studies are first performed on a coarse flow grid with 193,536 nodes until all nonlinear constraints have been satisfied. They are then restarted on a finer flow grid with 458,752 nodes. Figure 6.15 compares the optimization convergence history for \( \beta = 0.5 \) on the fine grid between the partitioned and monolithic methods. The convergence histories are very similar between the two methods at the beginning. However, the accumulation of the small differences in the functional and gradient values eventually causes the optimizer to take slightly different paths, as reflected in the deviations in the feasibility and optimality convergence. The convergence histories in Figure 6.15 are obtained over the course of 48 hours for both the partitioned and monolithic methods. The optimization using the
6.3. Verification

Figure 6.15: Optimization history for $\beta = 0.5$ on the finer flow grid using both the partitioned and monolithic methods.

![Convergence of feasibility and optimality](image)

(a) Convergence of feasibility and optimality

![Convergence of the merit function](image)

(b) Convergence of the merit function

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Sweep</th>
<th>Objective</th>
<th>Drag (N)</th>
<th>Weight (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>16°</td>
<td>1.60459</td>
<td>13946.3</td>
<td>2066.56</td>
</tr>
<tr>
<td>0.75</td>
<td>25°</td>
<td>1.71298</td>
<td>12401.7</td>
<td>2771.47</td>
</tr>
<tr>
<td>1.0</td>
<td>35°</td>
<td>1.65718</td>
<td>12035.3</td>
<td>4081.95</td>
</tr>
</tbody>
</table>

Partitioned

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Sweep</th>
<th>Objective</th>
<th>Drag (N)</th>
<th>Weight (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>16°</td>
<td>1.60470</td>
<td>13972.8</td>
<td>2061.07</td>
</tr>
<tr>
<td>0.75</td>
<td>24°</td>
<td>1.71847</td>
<td>12411.5</td>
<td>2769.83</td>
</tr>
<tr>
<td>1.0</td>
<td>35°</td>
<td>1.65643</td>
<td>12030.6</td>
<td>4145.20</td>
</tr>
</tbody>
</table>

Monolithic

![Optimized sweep angles](image)

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Sweep</th>
<th>Objective</th>
<th>Drag (N)</th>
<th>Weight (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>16°</td>
<td>1.60470</td>
<td>13972.8</td>
<td>2061.07</td>
</tr>
<tr>
<td>0.75</td>
<td>24°</td>
<td>1.71847</td>
<td>12411.5</td>
<td>2769.83</td>
</tr>
<tr>
<td>1.0</td>
<td>35°</td>
<td>1.65643</td>
<td>12030.6</td>
<td>4145.20</td>
</tr>
</tbody>
</table>

Table 6.7: Comparison of the optimized sweep angle, converged objective function, drag and weight values of the wing between the monolithic solution method and the partitioned algorithm.

The partitioned solution method.

The monolithic method completed 238 design iterations, but the optimization using the partitioned method was stopped after 97 design iterations. In other words, the efficiency benefit of using the monolithic solution method has allowed more than twice the number of design iterations to be performed in the same amount of time. The partitioned method required approximately 7 and 34 hours on a total of 240 processors to converge the merit function to within 1% and 0.1% of its final value, respectively. Achieving the same levels of convergence using the monolithic method needed 3 and 15 hours, respectively.

Figure 6.16, Table 6.7 and Figure 6.17 summarize the key optimization results when the monolithic solution method is used for both the coarse and fine grid optimization. Figure 6.16 superimposes the
two sets of optimized planforms, and it can be observed that they are very similar for all three $\beta$ values. This is confirmed by the optimized sweep angles listed in Table 6.7, where the original and the present investigations differ by less than 1° for all $\beta$. Furthermore, the functional values in Table 6.7 differ by less than 1% in most cases. Finally, Figure 6.17 reproduces the same optimized spanwise lift distributions from the original investigation in [104]. An important conclusion can be established that the monolithic solution method developed herein does not affect the results of the optimization. The expected tradeoff between drag and weight has been recovered. During the transonic sweep optimization study, the span of the wing is held fixed, which makes the coupled calculations less challenging for the partitioned method. To further demonstrate the advantage of using the monolithic solution method for optimization, a similar verification study with the wing span as a design variable will be included in the next chapter.

6.4 Three-Field Formulation and Monolithic Solution Method

The monolithic solution strategy proposed here uses a three-field formulation and iterates on the full set of aerostructural state variables. This approach has a number of advantages not demonstrated in other methodologies in the literature for high-fidelity aerostructural optimization. Section 5.1 has alluded briefly to some of these advantages, but it is worthwhile to revisit them in light of the details presented in this chapter.

Using a three-field formulation allows for more efficient coupled Jacobian matrix-vector product evaluation during the monolithic analysis and coupled adjoint solution. Contribution to the matrix-vector product from the mesh subproblem can be obtained via multiplications with the mesh Jacobian block. This eliminates the need to solve a linear system involving $\partial R_{M\Delta}/\partial b_{\Delta}$, which is necessary in a two-field formulation. Moreover, the block Jacobi and block Gauss-Seidel preconditioners can be more effective in a three-field formulation. The structural Jacobian block in $A$, $\partial R_{S}/\partial u$, in a two field formulation is a sum of the structural stiffness matrix, $K_S$, and another term that includes $\partial G/\partial u$ (see (B.5) in Appendix B). Existing routines in the structural module for computing $K_S^{-1}$ therefore provide a better structural block preconditioner in a three-field formulation. In addition, the block Gauss-Seidel preconditioner in the present methodology is able to capture the influence of the flow grid on both the flow and the structural subproblems relatively easily and efficiently. In contrast,
accomplishing the same goal in a two-field formulation requires the partial derivative term $\partial G/\partial \mathbf{u}$, which necessitates another mesh movement calculation. The effectiveness of the present methodology has been demonstrated via the performance improvements relative to the partitioned method reported throughout this chapter.

### 6.5 Chapter Summary

This section investigated the design of an efficient monolithic solution method for the high-fidelity aerostructural optimization methodology introduced in Chapter 5. A Newton-Krylov solution algorithm was developed for the coupled analysis problem and a Krylov subspace method, GCROT$(m,k)$, was used for the monolithic solution of the coupled adjoint problem. A modular approach was adopted where the subproblems are treated as blocks in the fully coupled linear system. The present monolithic methodology has a number of features that distinguish it from similar approaches in the literature. More specifically, the use of a three-field formulation has allowed the Jacobian matrix-vector product to be evaluated efficiently during each coupled Krylov iteration. For the monolithic analysis, the Jacobian matrix-vector product is obtained using a combination of analytical differentiation and finite-difference approximation. The use of a three-field formulation additionally leads to effective block Gauss-Seidel preconditioners, which allow the coupled linear system to converge in a minimal number of iterations. Existing iterative linear system solution routines in the flow, mesh, and structural modules were found to be effective block preconditioners. The optimal iterative solution tolerances for the flow and mesh block preconditioners were identified. The present methodology is also unique in the use of a pseudo-nonlinear mesh movement in the coupled problem. This chapter showed that a monolithic solution technique is effective for this type of problem, where the multiple increments present in the mesh states are updated simultaneously instead of sequentially. Finally, the importance of scaling was established in ensuring the nonlinear convergence during analysis and the accuracy of the gradients during the coupled adjoint calculations. For the test problem chosen for this chapter, which is at flow conditions typically specified for a 2.5g load condition, the use of the monolithic method improved the efficiency of the coupled analysis and adjoint calculations by 48% and 60%, respectively, in comparison to the partitioned method. Key factors contributing to the efficiency improvements were investigated through detailed cost comparisons between the monolithic and partitioned calculations. Results obtained using the monolithic solution strategies were verified against results obtained previously using the partitioned method.
Chapter 7

Results and Applications

This chapter demonstrates the effectiveness of the present methodology via a number of applications, highlighting the effectiveness of the monolithic solution capabilities developed in Chapter 6. Section 7.1 examines the performance of the monolithic and partitioned solution methods presented in this thesis with varying degrees of coupling in the problem. Section 7.2 applies the present methodology to a span optimization study with the goal to capture the tradeoff between weight and drag. Such optimization further involves large planform changes as well as flexible wing structures. It is therefore an ideal test case for illustrating the use of the integrated geometry parameterization and mesh movement for aerostructural optimization, and for showing the robustness and efficiency of the monolithic solution algorithm.

7.1 The Performance of Partitioned and Monolithic Solution Methods at Different Design Conditions

The previous chapter has compared the monolithic and partitioned methods from a computational and algorithm development perspective. This section furthers the investigation by comparing the performance of both types of solution method in applications with varying degrees of aerostructural coupling. It has been shown in the literature that monolithic solution methods are more advantageous than partitioned methods when the problem is more tightly coupled [30, 51, 53, 54, 55]. However, it remains unclear, in the context of steady aerostructural analysis of aircraft wings, if it is always beneficial to use monolithic solution strategies. Another important question is whether the benefits are sufficiently large for most problems of interest to justify the additional memory usage and implementation efforts.

The level of coupling in the aerostructural problem depends on many factors, including the conditions of the flow, material properties, geometry of the wing, thickness of the structural components, etc. The objective of this study is to examine the impact of various parameters chosen for an optimization study on the performance of the solution method. During an aerostructural optimization, the geometry and the thickness of the structural components are often manipulated by the optimizer to achieve the desired aeroelastic tailoring. The effects of the geometry and the structural thickness distribution are additionally difficult to generalize in terms of a small number of controllable parameters. For these reasons, these effects are excluded from this investigation by choosing a wing with fixed geometry and structural sizing. The span, which intuitively should have a more profound impact on the degree of coupling relative to
other geometric variables, will be examined in the next section. The angle of attack is also not included in this study because its value is usually determined such that the lift constraint is satisfied for a given geometric and structural design. The present investigation hence focuses on parameters that define the design conditions for the optimization. This includes the freestream velocity, \( v_\infty \), freestream density, \( p_\infty \), freestream Mach number, \( M_\infty \), and the Young’s modulus of the material, \( E \). Using dimensional analysis and the Buckingham Pi theorem \cite{153,154}, it can be shown that the set of independent parameters can be further reduced to two dimensionless variables: \( q_\infty/E \) and \( M_\infty \), where \( q_\infty = \frac{1}{2} \rho_\infty v_\infty^2 \) is the freestream dynamic pressure. The \( q_\infty/E \) ratio, Mach number, and the Reynolds number are also identified as relevant experimental parameters for the HIRENASD project \cite{136}. However, investigating the effect of the Reynolds number requires RANS aerostructural calculations, so it is deferred to future work.

Ten equally spaced \( q_\infty/E \) values are selected for this study, as summarized in Table 7.1. Both \( q_\infty \) and \( M_\infty \) depend on the \( p_\infty \) and \( v_\infty \), so the following expression of the freestream dynamic pressure is adopted to vary \( q_\infty \) and \( M_\infty \) independently:

\[
q_\infty = \frac{1}{2} \gamma p_\infty M_\infty^2, \quad \text{where} \quad M_\infty = v_\infty \sqrt{\frac{p_\infty}{\gamma p_\infty}}, \quad \gamma = 1.4.
\]  

Using the above expression, the \( q_\infty \) value for data point 3 in Table 7.1 can be computed for a 1g cruise flight condition, with a freestream Mach number of 0.785 and an altitude of 35,000 feet, or equivalently a freestream pressure of 23.843 kPa. Similarly, the \( q_\infty \) value for data point 7 is taken from a 2.5g maneuver flight condition, which has a freestream Mach number of 0.798 and is assumed to be at an altitude of 12,000 feet, translating to a freestream pressure of 64.442 kPa. The structures modeled in all applications in this thesis are made of Aluminum, which has a Young’s modulus of 73.1 GPa. This allows \( q_\infty/E \) for data points 3 and 7 to be determined, using which the rest of the data points are then extrapolated. This ensures that the typical flight conditions considered in an optimization are included in this study, along with additional scenarios where \( q_\infty/E \) is either between or beyond those corresponding to the cruise and maneuver conditions.

After the appropriate range of \( q_\infty/E \) has been determined, the cruise and maneuver load conditions become irrelevant to how \( q_\infty/E \) is varied. Each column in Table 7.1 has the same \( q_\infty/E \). In the third to fifth rows of Table 7.1 \( q_\infty/E \) is varied by changing the Young’s modulus of the structure while keeping \( p_\infty \) at 23.843 kPa and the Mach number at 0.785, 0.798, and 0.5, respectively. This will illustrate the effect of changing \( M_\infty \) independently of \( q_\infty/E \). For completeness, the last row in Table 7.1 considers varying \( q_\infty/E \) by changing \( p_\infty \) while keeping the Mach number and Young’s modulus at 0.785 and 73.1 GPa, respectively. This should have the same effect as changing \( q_\infty/E \) at the same Mach number.

<table>
<thead>
<tr>
<th>Data Point</th>
<th>1</th>
<th>2</th>
<th>3(^\dagger)</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7(^\dagger)</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_\infty/E \times 10^{-8} )</td>
<td>1.46</td>
<td>7.76</td>
<td>14.1</td>
<td>20.4</td>
<td>26.7</td>
<td>33.0</td>
<td>39.3</td>
<td>45.6</td>
<td>51.9</td>
<td>58.2</td>
</tr>
<tr>
<td>( E(\text{GPa}) \text{ at } M = 0.785 )</td>
<td>707</td>
<td>132</td>
<td>73.1</td>
<td>50.5</td>
<td>38.5</td>
<td>31.2</td>
<td>26.2</td>
<td>22.6</td>
<td>19.8</td>
<td>17.7</td>
</tr>
<tr>
<td>( E(\text{GPa}) \text{ at } M = 0.798 )</td>
<td>730</td>
<td>137</td>
<td>75.5</td>
<td>52.2</td>
<td>39.8</td>
<td>32.2</td>
<td>27.0</td>
<td>23.3</td>
<td>20.5</td>
<td>18.3</td>
</tr>
<tr>
<td>( E(\text{GPa}) \text{ at } M = 0.5 )</td>
<td>287</td>
<td>53.8</td>
<td>30.0</td>
<td>20.5</td>
<td>15.6</td>
<td>12.6</td>
<td>10.6</td>
<td>9.15</td>
<td>8.04</td>
<td>7.17</td>
</tr>
<tr>
<td>( p_\infty(\text{kPa}) \text{ at } M = 0.785 )</td>
<td>2.467</td>
<td>13.155</td>
<td>23.843</td>
<td>34.530</td>
<td>45.218</td>
<td>55.906</td>
<td>66.594</td>
<td>77.282</td>
<td>87.969</td>
<td>98.657</td>
</tr>
</tbody>
</table>

Table 7.1: A sample of ten \( q_\infty/E \) values used to analyze the performance of the monolithic and partitioned solution methods. The superscripts \( \dagger \) and \( \ddagger \) indicate that the corresponding \( q_\infty/E \) ratio is obtained from the cruise and maneuver flight conditions, respectively.
7.1. Comparing the Performance of Partitioned and Monolithic Methods

Figure 7.1: Surface patches on the geometry used for the analysis and adjoint calculations.

Figure 7.2: The component thickness distribution in the structural model.

Figure 7.3: Deflected wing shapes at the $q_{\infty}/E$ values corresponding to data points 1, 3, 5, and 10 at $M_{\infty} = 0.785$.

by varying $E$, which will confirm that the results obtained in this investigation do not depend on the combination of parameters leading to the same $q_{\infty}/E$ ratio.

This study is performed on a geometry with a planform based on the Boeing 737-900 wing, a half span of 16.13m, and the RAE 2822 airfoil section. An angle of attack of $2^\circ$ is used in all cases. Surface patches and control points describing the geometry of interest are shown in Figure 7.1. The flow grid has 458,752 nodes and is divided into 112 blocks. Each block is parameterized by $6 \times 6 \times 6$ B-spline control points. The internal structure is modelled with 30,473 MITC shell elements and is illustrated in Figure 7.2.

A thickness distribution from an optimization is not used here to avoid aeroelastic tailoring at specific wing loads. The component thickness values shown in Figure 7.2 have been manually specified such that they follow a linear variation along the span. Three mesh movement increments are used to obtain the deformed flow grid due to structural deflections. For aerostructural analysis, Aitken acceleration is used with the partitioned nonlinear block Gauss-Seidel method, while a block Gauss-Seidel preconditioner is used with the Newton-Krylov solution algorithm. All analysis in this study are converged to a tolerance of $10^{-8}$, and the coupled adjoint problems are converged to a tolerance of $10^{-9}$.

Figure 7.3 includes a front view of the deflected wing shapes for the $q_{\infty}/E$ values corresponding to data points 1, 3, 5, and 10 at $M_{\infty} = 0.785$. It can be observed that the increase in tip deflections slows down as $q_{\infty}/E$ increases due to the loss of lift. However, the increase in computational time with increasing $q_{\infty}/E$, as plotted in Figure 7.4, shows that the coupled analysis continues to become more challenging independently of the resulting tip deflections. For the smallest $q_{\infty}/E$ value considered, the partitioned method and the monolithic solution method require roughly the same amount of computing time. The solution time for the partitioned method increases rapidly with increasing $q_{\infty}/E$. While the monolithic solution time also increases with increasing $q_{\infty}/E$, it does so at a much lower rate. As a result, the benefit of using the monolithic strategy becomes more apparent as the problem becomes more tightly
coupled. At the last data point where $q_\infty / E$ is the highest, the monolithic solution method results in an overall saving of about 52% in computational time relative to the partitioned method. It is also worth pointing out that the start-up phase of the monolithic solution using partitioned iterations exhibits a more rapid increase in computational time with increasing $q_\infty / E$ than the Newton-Krylov portion of the algorithm. While this is not surprising, it serves as the motivation for more robust globalization strategy as part of future work.

Figure 7.5 shows a similar comparison between the partitioned adjoint solution using the linear block Gauss-Seidel method and the monolithic adjoint calculations using GCROT(20,1). Due to the observation from Section 6.2 that the monolithic coupled adjoint solutions require a higher number of coupled Krylov iterations for structural functionals than for aerodynamic functionals, calculations involving both functionals are considered in Figure 7.5 The difference in monolithic adjoint solution time between aerodynamic and structural functionals is small in comparison to the time difference between monolithic and partitioned calculations. As a result, the computational time plotted for each $q_\infty / E$ is simply averaged between the calculations involving the two types of functionals. Unlike the results for aerostructural analysis, the coupled adjoint solution using the monolithic method consistently outperforms the partitioned method for all values of $q_\infty / E$. For smaller $q_\infty / E$ where there are less interactions between the flow and the structures, the monolithic method is about 42% more efficient than the partitioned method with a relaxation factor of 0.75. However, it is evident that the use of such large relaxation factor becomes inefficient as $q_\infty / E$ increases. In fact, for the largest $q_\infty / E$ value, the partitioned method requires over 100 coupled iterations to converge to the specified tolerance with $\theta = 0.75$. This is undesirable during an optimization in terms of both robustness and efficiency. A more conservative choice of $\theta = 0.5$ is likely needed during an exploratory optimization where the level of aerostructural interaction is difficult to predict in advance. A relaxation factor of 0.5 is more robust than $\theta = 0.75$, but the convergence has also been reduced unnecessarily for less challenging cases. In comparison, the monolithic method offers the same level of robustness, if not more, as the partitioned method with $\theta = 0.5$, while being about 60–70% more efficient in terms of the computational time.
7.1. Comparing the Performance of Partitioned and Monolithic Methods

Figure 7.6: Tip deflections and computational savings from using the monolithic solution method for increasing $q\infty/E$ values. Each subfigure corresponds to a different row in Table 7.1.

(required.

The results from Figures 7.3, 7.4, and 7.5 are summarized in Figure 7.6a. The percentage reduction in computational time from using the monolithic solution method for analysis and coupled adjoint calculations, as well as the tip deflections, are plotted as functions of $q\infty/E$. Figure 7.6a, which corresponds to the third row of Table 7.1, can be compared with the remaining plots in Figure 7.6 which correspond to the fourth to sixth rows in Table 7.1. Figures 7.6a and 7.6b represent two different ways to vary $q\infty/E$ at the same Mach number. It is evident that the two figures look almost identical, confirming that the conclusions from Figures 7.3, 7.4, and 7.5 apply to all combinations of flow conditions and material properties with the same $q\infty/E$ ratios at $M\infty = 0.785$. Figures 7.6c and 7.6d show the computational advantage of using the monolithic method and the tip deflections with different $q\infty/E$ at two additional Mach numbers: $M\infty = 0.5$ and $M\infty = 0.798$. Figures 7.6c and 7.6a are very similar as they both involve transonic flows. The tip deflection at $M\infty = 0.5$ is 1 to 1.5 percentage half span lower than in the other three cases. The partitioned solution to the coupled adjoint problem with $\theta = 0.75$ at $M\infty = 0.5$ is...
also more well-behaved with increasing $q_\infty/E$ than at the higher Mach numbers. Even then, for the last three values of $q_\infty/E$, the monolithic method is still almost 50% more efficient for analysis and 50%-60% more efficient for coupled adjoint solutions. This shows that the $q_\infty/E$ ratio has a bigger impact on the difficulty of the aerostuctural calculations than the Mach number. For Mach numbers between from 0.5 and 0.8, the advantage of the monolithic solution method is quite clear, especially if the optimization includes load conditions where the $q_\infty/E$ is similar to or higher than the 2.5g load condition considered here. Finally, in all results considered in this investigation, the monolithic method is at least as competitive as the partitioned method.

### 7.2 Span Optimization Study

An optimization study is presented in this section to demonstrate the capability of the proposed methodology for high-fidelity aerostructural optimization. The optimization includes the span as a planform design variable, so that the tradeoff between weight and drag can be investigated. The induced drag of a wing is inversely proportional to the wing span squared, as given by the following expression \[D_i = \frac{L^2}{\pi q_\infty b^2 e},\] (7.2)

where $L$ is the lift produced by the wing, $b$ is the wing span, and $e$ is the span efficiency factor. Equation (7.2) shows that there is a substantial induced drag benefit in increasing the span of the wing. From a structural perspective, however, a higher span incurs a higher root-bending moment, necessitating a heavier structure to avoid structural failure. It is important to ensure that this tradeoff can be captured correctly by the present methodology. An optimization with span as a design variable is also more challenging from an aerostructural solution perspective, because wings with a light structure or high aspect ratio that occur during the optimization may both lead to more tightly coupled problems. This allows the robustness of the proposed monolithic solution algorithm to be demonstrated.

The geometry used for this study, along with the structural layout, is shown in Figure 7.7. The wing planform is based on the Boeing 737-900 wing, with an initial half span of 16.13m and the RAE 2822 airfoil section. Given the tradeoff discussed earlier, the optimal span for the wing of interest depends on the relative emphasis on weight and drag in the objective function. The objective function for practical wing design is derived from the design requirements for the aircraft, which implicitly determines the relative importance between weight and drag. However, as the primary interest of this study is simply to capture the tradeoff, an objective function given by the linear combination of the normalized weight and drag values is used. Specifically, it is given by the following expression:

$$J = \beta \frac{D}{D_0} + (1 - \beta) \frac{W}{W_0},$$ (7.3)

where $\beta$ is a parameter between zero and unity, $D$ is the inviscid drag produced by the wing of interest, and $W$ is the weight of the wingbox structural model. Both the drag and weight values are normalized by their respective initial values, $D_0$ and $W_0$, at the start of an optimization. A larger $\beta$ indicates that more emphasis is placed on the reduction in drag than on the reduction in weight. Three $\beta$ values of 0.25, 0.5, and 0.75 are considered.

The optimization for each $\beta$ value includes a 1g cruise condition and a 2.5g maneuver condition.
7.2. Span Optimization Study

The cruise condition assumes a Mach number of 0.785 and an altitude of 35,000 feet. The maneuver condition assumes a Mach number of 0.789 and an altitude of 12,000 feet. The aerodynamic performance, or the value of $D$ in (7.3), is evaluated at the cruise condition. The maneuver condition is added to capture the correct trends in sizing the structures, which determines the value of $W$ in (7.3). Each of the cruise and maneuver conditions is subject to a lift constraint, such that the lift produced by the wing is equal to the estimated weight of the aircraft scaled by a load factor of 1.0 for the cruise condition and 2.5 for the maneuver condition. The aircraft weight used in defining the lift constraint is obtained by adding the weight of the wing, as given by the finite-element structural model sized by the optimization, to a fixed weight value of 785,000N. The fixed weight value is estimated to be 93% of the maximum take-off weight of a Boeing 737-900 aircraft, the remaining 7% of which has been discounted as the weight of the wing. In addition, the predicted weight of the wing by the structural model is multiplied by a mark-up factor of 1.5 to account for changes in the weight of any wing components not included in the finite-element model [73].

Structural constraints are also imposed at the maneuver condition to ensure that the von Mises stresses in the structures do not exceed the yield stress of the material. More specifically, the KS function introduced in Section 3.2.4 is used to aggregate the failure constraints on the ribs and spars, the top skin, and the bottom skin of the wingbox, resulting in a total of three KS constraints. When optimizing with respect to the wing span with a fixed structural topology, buckling constraints also become more important in preventing the rib spacings from becoming impractical due to the increase in span. While buckling constraints are not included in this study, it has been ensured that the span increase from the optimization remains reasonable. The structural sizing in practical wing design also considers many additional load conditions, but doing so is computationally expensive. For the purpose of this study, it suffices to capture the correct trends in structural sizing, which can be accomplished by the use of a single 2.5g maneuver load condition with a safety factor of 2.

There are 2 AOA design variables, one for each load condition. The thickness of the structural components in Figure 7.7 is controlled by 156 structural design variables, which are initialized with a uniform thickness distribution of 10mm everywhere. Figure 7.8 illustrates the geometry parameterization of the wing using B-spline surfaces, where the surface control points are shown as red spheres.
are two span and two twist design variables, as indicated in Figure 7.8. The span and twist between
the locations of the planform variables are interpolated while maintaining a straight leading edge and
trailing edge. In addition, the optimizer can modify the airfoil sections along the span using 16 of the
26 spanwise stations of control points. Each spanwise station has 20 control points in the chordwise
direction, 14 of which are design variables. This leads to a total of 228 geometric design variables.
The remaining B-spline surface control points are used to ensure curvature continuity between adjacent
patches along the surface of the wing.

The flow grid used for the analysis at each design iteration has 112 blocks. Each block is parameterized
with $6 \times 6 \times 6$ control points, consistent with the geometry parameterization shown in Figure 7.8. The
optimization is first performed on a coarse flow grid with 193,536 nodes. Upon satisfying all nonlinear
constraints on the coarse grid, the optimization is continued on a flow grid with 458,752 nodes. Due
to the grid dependence of the flow solution, which has been illustrated in Figure 5.7, a post-optimality
analysis using higher flow grid resolution is usually needed to predict the lift and drag values with
sufficient accuracy. However, the main focus of this section is on performing the optimization and past
experience has suggested that the chosen flow grid resolutions are sufficient for this purpose [104] [153].
The structural mesh resolution is fixed during both stages of the optimization with 30,473 MITC shell
elements, or 174,204 degrees of freedom. Figure 7.9 includes the optimization convergence history for
$\beta = 0.5$ in terms of the feasibility, optimality, and merit function. The dotted line in both plots indicates
where the switch from the coarse flow grid to the fine grid occurs. It is evident that the optimization
has successfully reduced the feasibility to below $10^{-6}$ from a peak value that is on the order of $10^{-2}$.
At the same time, the merit function, which represents the value of the the objective function when all
nonlinear constraints have been satisfied, is changing by less than 0.003% by the end of the optimization.
The optimization convergence histories for $\beta = 0.25$ and $\beta = 0.75$ are similar and are hence not included.

Table 7.2 summarizes the results of the optimization in terms of the wing span, estimated weight
of the aircraft each wing must carry (half the total weight), and the inviscid drag of the wing, for the
7.2. Span Optimization Study

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>Half Span (m)</th>
<th>Weight (N) ( \times 10^5 )</th>
<th>Drag (N) ( \times 10^4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>17.3</td>
<td>4.15</td>
<td>1.57</td>
</tr>
<tr>
<td>0.50</td>
<td>21.4</td>
<td>4.28</td>
<td>1.09</td>
</tr>
<tr>
<td>0.75</td>
<td>24.8</td>
<td>4.48</td>
<td>8.54</td>
</tr>
</tbody>
</table>

Table 7.2: Optimized half span, weight of the aircraft (half), and drag (one wing) for the three \( \beta \) values.

It can be observed that when more emphasis is placed on drag by increasing \( \beta \), the optimized span has indeed increased at the cost of a higher weight. The resulting increase in weight can be explained by examining the optimized thickness distribution on the upper and lower skins in Figure 7.10 for the three \( \beta \) values. It is clear that the increase in the span has led to an increase in the thickness of the skin panels in order to maintain the structural integrity. The optimizer has also increased the thickness inboard for all \( \beta \) values, which is in agreement with the results reported by other authors for similar studies [98].

Figure 7.11 further shows the spanwise lift distribution at both load conditions for the three \( \beta \) values. The elliptical lift distributions are also included for reference. All lift distributions shown are normalized by the maximum value from the cruise elliptical lift distribution at \( \beta = 0.25 \). Two key observations can be made. First of all, for the majority of the span, the spanwise loading at a given span location is higher for larger \( \beta \) values. This in turn increases the root-bending moment with increasing \( \beta \), which is consistent with the increase in the structural component thickness shown in Figure 7.7. This means that for higher \( \beta \) values, the optimizer has chosen to increase the span at the cost of increasing weight to achieve a reduction in drag, which is as expected. Secondly, the optimizer has evidently reduced the tip loading for all \( \beta \) values considered, even though the cruise lift distribution becomes slightly less triangular as more emphasis is placed on drag in the objective. Recall from the expression for the induced drag in (7.2) that increasing the span is an effective mechanism for reducing the induced drag. Therefore, even for higher \( \beta \) values, the optimization is driven by increasing the span as much as possible without causing failure in the structures, resulting in designs which are very effective in reducing the tip loading.

The above results show that the expected tradeoff has been successfully captured.

The optimization results presented for this study have been obtained using the monolithic solution method described in Chapter 6. The optimized design from the last design iteration on the fine mesh has been repeated using the block Gauss-Seidel method. Table 7.4 shows that the use of the monolithic method for analysis has led to reductions of 40–70% in computational time at the maneuver conditions, and 17–40% at the cruise conditions. Efficiency improvements in the coupled adjoint solutions are also summarized in Table 7.4. The partitioned adjoint solution with \( \theta = 0.75 \) fails to converge at the maneuver condition for \( \beta = 0.50 \) and diverges for \( \beta = 0.75 \). Therefore, no comparison is available with respect to these calculations. The previous section has shown that a relaxation factor of 0.75 works reasonably well when the problem is loosely or moderately coupled; the result in Table 7.4 hence confirms that an optimization with the span as a design variable can be challenging. A more conservative relaxation parameter of 0.5 is necessary in this case, relative to which the monolithic adjoint solution is shown to be approximately 60–70% more efficient for both load conditions.

The monolithic and partitioned methods are compared in Table 7.5 in terms of the computational time required per design iteration for different \( \beta \) values. The calculations are performed using a total of 240 processors, and the results on the coarse and fine flow grids are both included. In all cases, the use of the monolithic solution method allows at least twice, and sometimes almost three times, as many
Chapter 7. Results and Applications

Figure 7.10: Optimized thickness distribution on the upper and lower skins for different $\beta$ values.

Figure 7.11: Spanwise lift distribution for different $\beta$ values.

<table>
<thead>
<tr>
<th>$\beta$ Value</th>
<th>Analysis</th>
<th>Adjoint ($\theta = 0.5$)</th>
<th>Adjoint ($\theta = 0.75$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = 0.25$ Cruise</td>
<td>39%</td>
<td>63%</td>
<td>45%</td>
</tr>
<tr>
<td>Maneuver</td>
<td>70%</td>
<td>59%</td>
<td>77%</td>
</tr>
<tr>
<td>$\beta = 0.5$ Cruise</td>
<td>26%</td>
<td>66%</td>
<td>44%</td>
</tr>
<tr>
<td>Maneuver</td>
<td>60%</td>
<td>58%</td>
<td>(Does not converge)</td>
</tr>
<tr>
<td>$\beta = 0.75$ Cruise</td>
<td>17%</td>
<td>66%</td>
<td>53%</td>
</tr>
<tr>
<td>Maneuver</td>
<td>38%</td>
<td>72%</td>
<td>(Diverges)</td>
</tr>
</tbody>
</table>

Table 7.4: Percentage saving in computational time from the use of the monolithic solution strategies for analysis and adjoint calculations at the optimal design for different $\beta$ values.
Table 7.5: The computational cost of one design iteration using the monolithic and partitioned methods at different $\beta$ values and flow grid resolutions. The relative cost between analysis and gradient calculations is also shown.

design iterations to be performed compared to the partitioned method with the same computational effort. The second column in Table 7.5 also shows the relative cost between an analysis and a coupled adjoint solution during a design iteration. For the monolithic method, the relative cost is between 3.6 and 4 on the coarse flow grid, and is between 1.7 and 3.5 on the fine grid. For the partitioned method, the relative cost is between 2 and 2.5 on the coarse grid, and is between 1.6 and 3 on the fine grid.

To illustrate more clearly the performance of the partitioned and monolithic methods over the course of the optimization, the initial part of the optimization on the coarse grid is repeated using the partitioned method. The partitioned adjoint calculations use a relaxation factor of 0.5. Figure 7.12 plots the percentage of design iterations where the analysis and coupled adjoint solutions require a certain number of iterations. The pair of bar graphs for each $\beta$ value represent the cruise and maneuver calculations, respectively. Figure 7.12 shows a clear increase in the number of analysis and coupled adjoint iterations using the partitioned method when the span increases with higher $\beta$. It is also evident that the maneuver calculations are more challenging than the cruise calculations for both the monolithic and partitioned methods. However, the monolithic solution method demonstrates a consistent performance in that all analysis and adjoint calculations converge well within 20 iterations, even in the most challenging scenarios. The partitioned method, on the other hand, requires more than 30 analysis iterations and over 50 coupled adjoint iterations for the maneuver calculations with $\beta = 0.75$. Considering the increasing popularity in the use of composite materials, aircraft design in the future is likely to encounter high aspect ratio wings with flexible materials. Extrapolating the results in Figure 7.12 even if the partitioned method can still be used for the design of such wings, the designer must be willing to work with very conservative relaxation factors and a potentially unpredictable amount of computational time. The above arguments thus favour strongly the use of the monolithic solution method for the design of highly flexible wings, for its advantage in efficiency and reliable performance.

7.3 Chapter Summary

This chapter presented two studies which examine the performance of the monolithic and partitioned solution methods on a range of different problems. Section 7.1 investigated the variations in the flow conditions and material properties in terms of $q_\infty/E$ and $M_\infty$, which ultimately led to varying degrees of coupling in the aerostructural calculations. For the range of design conditions commonly specified for
Chapter 7. Results and Applications

Figure 7.12: Percentage of design iterations requiring a given number of iterations for analysis and coupled adjoint calculation, during the initial stage of optimization on the coarse flow grid.

optimization studies of commercial flight, the advantage of the monolithic solution method was clearly demonstrated. For the highest \( q_{\infty}/E \) values at Mach numbers between 0.5 and 0.8, the monolithic method is 50%–60% more efficient than the partitioned method for analysis, and is 60%–70% more efficient for the adjoint calculations. At the lowest \( q_{\infty}/E \) values, the monolithic method is at least as competitive as the partitioned method. The monolithic solution method additionally has the advantage of not relying on the choice of an appropriate relaxation parameter. Section 7.2 included an optimization study using the wing span as a geometric design variable. This investigation has accomplished two goals. First of all, the capability of the present optimization methodology has been demonstrated. The expected tradeoff between weight and drag was recovered, where an increased emphasis on drag relative to weight in the objective function led to an increase in span. Secondly, the optimization substantiated the suitability of the monolithic solution method for optimization involving potentially challenging designs. The efficiency improvements from the use of the monolithic solution method allowed, on average, the same number of design iterations to be performed in less than half the time required by the partitioned method. Performance of the monolithic solution method was also shown to less sensitive to the increase in span with higher \( \beta \) values.
Chapter 8

Conclusions, Contributions, and Future Work

8.1 Conclusions and Contributions

An effective optimization methodology has been proposed which is suitable for exploratory design based on tightly integrated high-fidelity aerostructural analysis. The key features of the present methodology and main conclusions from the investigations conducted throughout this thesis are summarized here. The contributions of this work are then listed at the end of this section.

8.1.1 Aerostructural Optimization with an Integrated Geometry Parameterization and Mesh Movement Algorithm

Exploratory aerostructural optimization requires a flexible geometry parameterization, as well as a robust mesh movement algorithm to accommodate large shape changes due to optimization and structural deflections. This has motivated the use of an integrated geometry parameterization and mesh movement algorithm originally proposed for aerodynamic shape optimization [44]. The integrated approach has several advantages as a geometry parameterization and control technique. It analytically describes both the undeflected geometry and the flying shape of the design using B-spline surface control points. The geometry parameterization enables and is tightly integrated with an efficient and robust mesh movement algorithm that allows flow grids of sufficiently high quality to be obtained despite substantial geometry changes. Chapters 4 and 5 describe and characterize the extension of this technique to high-fidelity aerostructural optimization. The original algorithm has been modified in a number of ways to enable its application to aerostructural optimization. The following conclusions have been reached in developing the new methodology:

• Efficiency of the flow grid movement is crucial during aerostructural optimization, where the flow grid deformation is coupled to the flow and structural analysis. This has been accomplished effectively by:
  – Storing the mesh movement data in a distributed manner, which has substantially reduced the matrix assembly time. The smaller problem size on each processor also allows the matrices
and their factorization at multiple increments to be stored with a small memory requirement.

- Performing the mesh movement calculations in parallel using the distributed data storage. Performance of this new parallel mesh movement algorithm is comparable to that of PETSc, but it allows for more direct access to the mesh movement data and algorithm.

- The challenge associated with deforming the mesh of the detailed structural model consistently with the B-spline geometry parameterization is addressed using a surface-based FFD method. It has successfully moved the structures of an initially planar wing to a new C-wing geometry created by hand. The surface-based FFD approach introduces noticeably less distortion to the structural components compared to an alternative inverse FFD method.

- It has been demonstrated that the integrated geometry parameterization and mesh movement algorithm can effectively accommodate shape changes due to both optimization and structural deflections:
  
  - The use of a three-field formulation, where B-spline control point coordinates are treated as explicit state variables, simplifies the analysis and the gradient calculations. Derivation of the gradient terms is further simplified by including an additional mesh adjoint equation for the jig shape in an augmented three-field coupled adjoint approach.
  
  - A strategy has been proposed such that the flow grid is moved in two sets of increments for changes in the jig shape and for structural deflections during analysis, respectively. As demonstrated via an aerostructural analysis performed on a C-wing geometry obtained from an initially planar wing, this approach reduces the size of the mesh subproblem without compromising the resulting flow grid quality.

- Through a number of verification and validation studies, it has been shown that the integrated geometry parameterization and mesh movement algorithm is sufficiently accurate for practical aerostructural design applications:
  
  - The surface fitting introduced during the displacement transfer does not affect the grid convergence of functional values. The error associated with the force and displacement transfer is reduced with increasing flow grid and control resolutions.
  
  - The proposed methodology has been validated successfully against the static aerostructural experimental data from the HIRENASD project.
  
  - The gradient calculations have also been verified via comparison with finite-difference approximations using a range of stepsizes. The minimum normalized error is on the order of $10^{-8}$ for most of the gradient values examined, with the largest minimum normalized difference on the order of $10^{-6}$.

### 8.1.2 Monolithic Solution to Coupled Aerostructural Problems

Exploratory aerostructural optimization requires efficient and robust solutions to the coupled problems, which can be accomplished using a monolithic solution method. For the present methodology, a Newton-Krylov solution algorithm is used for aerostructural analysis. The inexact Newton method is globalized by a number of partitioned iterations. Solution to the coupled linear system at each Newton iteration is
computed using FGMRES. Solution to the coupled adjoint problem is obtained via \text{GCROT}(m, k) [3].

Different aspects of the monolithic analysis and coupled adjoint calculations are investigated in Chapter 6. It is found that the efficiency of the monolithic solution algorithm can be maximized by the following:

- It is important to take into account the different scaling between the flow, mesh, and structural subproblems. The use of Jacobian block scaling improves the convergence of the Newton iterations in the case of analysis and the accuracy of the resulting gradient values in the case of the coupled adjoint calculations.

- The Jacobian matrix-vector product required by FGMRES at each Newton iteration is evaluated most efficiently by a combination of first-order finite-difference approximation and analytical differentiation. Evaluating the matrix-vector products for the coupled adjoint problem can fully reuse existing routines from the partitioned solution procedure.

- Both FGMRES and \text{GCROT}(m, k) allow for the use of block iterative methods as part of a nonstationary preconditioner, allowing the modularity between different solver modules to be preserved. A block Gauss-Seidel preconditioner works well for both analysis and the coupled adjoint problem. Existing iterative solvers are used as block preconditioners for the flow and mesh subproblems. The structural block is preconditioned by the direct factorization of the structural stiffness matrix computed prior to each analysis. Optimal solution tolerances for the flow and mesh block preconditioners have been identified, achieving a good balance between the effectiveness and cost of the preconditioning calculations.

- The use of a three-field formulation can be inherently advantageous due to more efficient matrix-vector products evaluation and more effective block preconditioning during Krylov iterations.

- A detailed cost comparison between a monolithic and a partitioned analysis iteration has shown that the flow grid movement problem is solved many more times during monolithic analysis. This highlights the importance of an efficient mesh movement algorithm.

- For both analysis and the coupled adjoint problem, the computational time required per coupled nonlinear or linear iteration is similar between the monolithic and the partitioned solution procedures. However, the monolithic solution method requires a much smaller number of coupled iterations. For this reason, using the proposed monolithic solution strategy has reduced the total aerostructural analysis time by almost 50% and sped up the coupled adjoint calculations by up to 70% for the selected test problem.

- Finally, it has been shown that the proposed monolithic solution method is able to consistently replicate the gradient values, as well as the results from a sweep optimization study obtained previously using the partitioned solution method.

### 8.1.3 Results and Applications

Chapter 7 demonstrates the capability of the present aerostructural optimization methodology, in particular the monolithic solution method, via two studies. The first study examines the performance of the proposed monolithic solution method and the partitioned method under different design conditions specified in terms of the $q_{\infty}/E$ ratio and the freestream Mach number. The second investigation includes
three optimization problems with varying emphasis between weight and drag in the objective function and with span as one of the design variables. The span optimization leads to large planform changes and highly flexible wings, which allows the effectiveness of the integrated geometry parameterization and mesh movement as well as the coupled solution capabilities to be demonstrated. The main conclusions from the two studies are summarized as follows:

• The proposed monolithic solution method demonstrates a consistent performance by solving the analysis and coupled adjoint problems efficiently regardless of increasing $q_{\infty}/E$ ratios, freestream Mach numbers, and increasing span.

• On the other hand, performance of the partitioned method is sensitive to changes in $q_{\infty}/E$, which is evidenced by a drastic increase in computational time, especially at higher Mach numbers corresponding to transonic flows. Using the partitioned method for the span optimization study also shows an increase in the number of iterations required for analysis and adjoint calculations with increasing wing aspect ratio.

• Performance of the partitioned adjoint calculations further depends heavily on the choice of relaxation parameter. This is undesirable for exploratory optimization where an appropriate relaxation parameter can be difficult to determine in advance.

• Based on the above observations, the benefits of the proposed monolithic solution method have been clearly demonstrated:

  – For high $q_{\infty}/E$ values and Mach numbers between 0.5 and 0.8, the proposed monolithic solution method is 50–60% more efficient for analysis in comparison to the partitioned method. For coupled adjoint calculations, the monolithic solution method can be as much as 90% more efficient if the partitioned method uses a relaxation parameter of 0.75. For a relaxation parameter of 0.5, the monolithic adjoint solutions are 60–70% more efficient for all combinations of $q_{\infty}/E$ and $M_{\infty}$ considered. In cases where the partitioned method performs well, the monolithic solution method remains competitive in terms of computational time.

  – Using the proposed monolithic solution strategy for the span optimization reduces the average computational time per design iteration by more than 50% compared to the partitioned method. This strengthens the conclusion that using the monolithic solution method is advantageous for aerostructural optimization applications involving computationally challenging designs.

• The span optimization study additionally demonstrates the use of the present optimization methodology in solving aerostructural optimization problems with a realistic number of design variables. Results of the optimization have accurately captured the tradeoff between the induced drag benefits of increasing the wing span and the resulting increase in the structural weight. Expected trends in the structural sizing and the spanwise lift distributions have also been recovered.

8.2 Contributions

The goal of this thesis is to design and develop a flexible, efficient, and robust exploratory optimization methodology based on high-fidelity steady-state aerostructural analysis. This goal is accomplished by
achieving two specific objectives: the extension of the integrated geometry parameterization and mesh movement algorithm to aerostructural optimization, and the design of an efficient monolithic method for the solution to coupled problems. The specific contributions of this thesis as related to each of the two objectives are summarized below:

- The integrated geometry parameterization and mesh movement algorithm has been successfully extended to aerostructural optimization. This was in turn accomplished by addressing a number of challenges:
  - Geometry parameterization using B-spline surfaces has been made possible via the introduction of a surface-based FFD method which allows the structural model to be warped consistently with the OML while minimizing any unwanted distortions in the structural components.
  - By adopting a three-field formulation of the coupled problem, it has been shown that the integrated geometry parameterization and mesh movement algorithm is effective for shape changes due to both optimization and structural deflections. Suitability of the proposed methodology for accurate aerostructural analysis has been demonstrated.
  - An augmented three-field coupled adjoint approach has been proposed for gradient calculations, the results of which have been verified relative to second-order finite-difference approximations.
  - Efficiency of the flow grid movement has been improved for use in coupled calculations. This has been realized by solving the mesh movement problems in a distributed manner, and by introducing separate sets of grid movement increments for changes in the jig shape and for structural deflections to reduce the size of the mesh subproblem.

- An effective monolithic solution strategy has been proposed for the present methodology, which has led to substantial efficiency improvements relative to the use of the partitioned method. Multiple aspects in the design of the coupled solution algorithm have been investigated and their effects quantified:
  - The importance of scaling during monolithic analysis and coupled adjoint calculations has been established. Appropriate scaling strategies have been proposed.
  - An efficient and accurate way to evaluate the Jacobian matrix-vector products for analysis using a Newton-Krylov solution algorithm has been described.
  - Factors contributing to the effectiveness of the block Gauss-Seidel and block Jacobi preconditioners have been identified. A block Gauss-Seidel preconditioner has further been recommended for both analysis and coupled adjoint solution along with the optimal choices of preconditioning parameters.

- The effectiveness of the proposed monolithic solution method for aerostructural optimization has been demonstrated via comparison with the partitioned method in a number of applications.
8.3 Future Work

• The globalization of the inexact Newton method is currently achieved by applying a number of nonlinear block Gauss-Seidel iterations. Although the use of a Newton-Krylov solution algorithm has led to substantial efficiency improvements, the start-up phase is still prone to the same drawback as a partitioned method. This is illustrated in Figures 6.9 and 7.4, where it is evident that the start-up phase of the nonlinear analysis constitutes a large portion of the overall solution time. This suggests that a more effective globalization strategy is an important next step. A monolithic homotopy continuation algorithm proposed recently by Brown and Zingg [156] has been applied successfully to high-fidelity flow analysis and may present an effective option for aerostructural analysis.

• A natural extension of the present methodology is to include the RANS flow analysis into the coupled aerostructural optimization calculations. This will involve differentiating the relevant cross coupling terms for the gradient calculations and introducing appropriate modifications to the monolithic solution algorithms.

• The two-level FFD approach by Gagnon and Zingg [157] can be added to the aerostructural optimization framework to expand on the current geometry parameterization and control capabilities. The two-level FFD approach, which acts on the geometry via the B-spline surface control points, is compatible with the surface-based FFD algorithm. This allows a wider range of geometries to be studied and optimized.

• All applications considered for this thesis have assumed that the structures are made of Aluminum. In recognizing the increasingly important role played by composite materials in the design of next-generation aircraft, it will be useful to enable the analysis and design of composite structural components during aerostructural optimization.

• Although the present methodology allows for a large amount of geometric freedom for exploratory optimization, a significant portion of the design space either cannot be realized or is not feasible without modifications to the structural topology. Therefore, concurrent aerostructural and topology optimization will be another step towards an automated design process that is fully exploratory.

• The present methodology is able to capture the correct trends in structural sizing via the use of stress constraints. However, there are other modes of failure such as flutter, buckling, divergence, and buffeting, which are also relevant during an aerostructural optimization. Having the capabilities to evaluate these constraints allows the structural components to be sized more accurately. For a similar reason, the use of nonlinear structural analysis or unsteady aerostructural analysis are other possibilities for future research directions.
References


References


References


Appendices
Appendix A

A Two-Field Formulation with Algebraic Grid Movement

Another optimization framework has been developed near the beginning of this research project in collaboration with Dr. Timothy Leung and Dr. Xiaodong Wang, who were then post-doctoral fellows at the University of Toronto. This early framework is a first attempt at performing aerostructural analysis and optimization using the flow and structural modules described in Chapter 3. It is largely based on the framework developed by Kenway et al., who adopted a two-field formulation of the coupled aerostructural equations and an FFD technique to control the aerodynamic and structural geometries during optimization. Deformation of the flow grid due to optimization and structural deflection is accomplished using an algebraic mesh movement. Algebraic mesh movement allows for an easier integration of the flow and structural modules without significant modifications to the original multi-disciplinary architecture by Kenway et al. Algebraic mesh movement is very effective in performing detailed designs of the geometry, where shape changes are mostly small. However, it is not meant for exploratory optimization, where large shape changes can be expected. It is also more difficult to ensure the grid quality during the analysis of designs with highly flexible structures. These limitations have motivated the development of the methodology presented in Chapters 5 and 6, which uses an integrated geometry parameterization and mesh movement technique. A brief description of this early framework is provided here.

A.1 Aerostructural Analysis

In a two-field formulation, the nonlinear aerostructural analysis solves a set of coupled equations in the form of

\[ R_{AS}(q, u) = \begin{bmatrix} R_A(q, u) \\ R_S(q, u) \end{bmatrix} = 0, \]  

(A.1)

where \( R_A \) and \( R_S \) refer to the flow and structural residuals, respectively, and \( q \) and \( u \) refer to the flow and structural state variables, respectively. Equation (A.1) is solved by a nonlinear block Gauss-Seidel method. The flow residual and the forces acting on the structure, \( f_S \), are explicit functions of \( u \) via changes in the flow grid due to structural deflections. This implies an instantaneous change in the

\[^1\text{Dr. Leung completed the majority of the analysis capability prior to the start of my PhD project. We then collaborated with Dr. Wang on the development of the optimization capability during the subsequent year and a half.}\]
Appendix A. A Two-Field Formulation with Algebraic Grid Movement

flow grid with every change in \( u \), which can be accomplished relatively easily with an algebraic mesh movement.

The algebraic mesh movement used for this methodology evaluates the new location of a grid node, \( x_k \), algebraically according to the following expression [158]:

\[
x_k^{\text{new}} = x_k^{\text{old}} + \frac{\Delta x_1}{2} \left[ 1 + \cos(\pi S_k) \right],
\]

where \( x_1 \) is the coordinates of the surface node, and \( S_k \) is the arc-length parametric location of the node along the grid line. The algebraic mesh movement uses a discrete representation of the aerodynamic geometry. This allows it to be interfaced directly with the FFD method by Kenway et al. [30] for shape optimization, as well as the displacement transfer scheme from Section 3.2.3. Furthermore, the partial derivative of the deformed flow grid with respect to \( u \), as needed for gradient calculations later, does not require any matrix inversion as in the case of a linear elasticity mesh movement algorithm. However, (A.2) implies that all changes in the surface nodes are absorbed entirely within the blocks adjacent to the surface. This constrains the magnitude of the geometry changes which can be achieved using the algebraic mesh movement.

A.2 Optimization and Gradient Calculations

In the earlier framework, geometry control during optimization is achieved by embedding the discrete aerodynamic surface geometry and the mesh of the structural model within an FFD volume [23]. Changes to the aerodynamic and structural geometries can be specified by moving the FFD control points as geometric design variables. The flow grid is then moved by algebraic mesh movement based on changes in the discrete aerodynamic surface.

The gradients are calculated using the discrete coupled adjoint method as follows [30, 73]:

\[
G = \frac{\partial J}{\partial q}^T + \frac{\partial R_A}{\partial q}^T \Psi_A + \frac{\partial R_S}{\partial q}^T \Psi_S,
\]

where the adjoint variables \( \Psi_A \) and \( \Psi_S \) are the solution to the following coupled adjoint problem:

\[
\begin{bmatrix}
\frac{\partial R_A}{\partial q}^T & \frac{\partial R_S}{\partial q}^T \\
\frac{\partial R_A}{\partial u}^T & \frac{\partial R_S}{\partial u}^T
\end{bmatrix}
\begin{bmatrix}
\Psi_A \\
\Psi_S
\end{bmatrix}
= \begin{bmatrix}
-\frac{\partial J}{\partial q}^T \\
-\frac{\partial J}{\partial u}^T
\end{bmatrix}.
\]

Solution to a linear system in the form of (A.4) is required for each objective or constraint that depends on the state variables. Equation (A.4) is solved via a block Gauss-Seidel method. Every \( k \)th iteration consists of solving the following [30]:

\[
\frac{\partial R_A}{\partial q}^T \Psi_A^{(k)} = -\frac{\partial J}{\partial q}^T - \left( -\frac{\partial f_S}{\partial q}^T \Psi_S^{(k-1)} \right),
\]

\[
K_S \Psi_S^{(k)} = -\frac{\partial J}{\partial u}^T - \frac{\partial R_A}{\partial u}^T \Psi_A^{(k)} - \frac{\partial G_s}{\partial u}^T \left( -\frac{\partial f_S}{\partial G_s}^T \right) \Psi_S^{(k-1)}.
\]
Equations (A.5) and (A.6) correspond to the flow and structural adjoint equations, respectively. The RHS of (A.5) and (A.6) include the off-diagonal terms in the coupled Jacobian matrix in (A.4). The nonsymmetric contribution to $\partial R_S/\partial u$ due to the force vector $f_S$ is also included in the RHS of (A.6), using a lagged adjoint value from the previous iteration. This allows existing solution routines for (3.28) and (3.45) in the flow and structural modules to be reused, as in the case with the three-field formulation.

Some of the derivative terms required in the two-field coupled adjoint formulation and total gradient calculation are also present in the three-field formulation described in Section 5.2. These terms will not be discussed extensively here. The relationship between a two-field and a three-field formulation will be further illustrated in Appendix B.

The flow adjoint equation is the same as (5.19) in a three-field formulation. The coupling term in the structural adjoint equation is found by

$$\frac{\partial R_A}{\partial u}^T \Psi_A = \frac{\partial u_A}{\partial u}^T \partial G_{s\Delta} \frac{\partial G}{\partial G_{s\Delta}}^T \frac{\partial R_A}{\partial G} \frac{\partial R_A}{\partial \Psi_A},$$

where $G_{s\Delta} = G_{sJ} + u_A$ is the coordinates of the deflected aerodynamic geometry. Most terms above also appear in the three-field coupled adjoint calculation, with the exception of $\partial G/\partial G_{s\Delta}$. This term involves differentiating the algebraic mesh movement. It is needed for the partial derivative of the flow grid with respect to the structural deflections, or $\partial G/\partial u_A$, which is also present in the remaining terms on the RHS of (A.6) to be discussed next. The right-most term in (A.6) is due to the follower forces, $f_S$, in $R_S$, which depends on $u$ in a two-field formulation. It is computed as follows

$$\frac{\partial f_S}{\partial u}^T \Psi_S = \frac{\partial u_A}{\partial u}^T \frac{\partial G}{\partial u_A}^T \left[ -\left( \frac{\partial f_S}{\partial G_{s\Delta}^T} \Psi_S - \frac{\partial f_A}{\partial G}^T \frac{\partial f_S}{\partial G} \Psi_S \right) \right].$$

The terms in the square bracket are the same as those seen in (5.25). The partial derivative of an aerodynamic functional with respect to the structural state is also non-zero in (A.6). In a two-field formulation, any change in $u$ will affect the aerodynamic functional values evaluated on the flow grid. These effects are expressed via (30):

$$\frac{\partial J_A}{\partial u}^T = \frac{\partial u_A}{\partial u}^T \frac{\partial G}{\partial u_A}^T \frac{\partial J_A}{\partial G},$$

where $\partial J_A/\partial G$ is present in the RHS of (5.20) in a three-field formulation. The partial derivative of a structural functional with respect to $u$ is unchanged from (3.45).

Equation (A.4) for aerodynamic and structural design variables is the same as in a three-field formulation. The gradients with respect to geometric design variables, $v_G$, are different because the flow grid is not an explicit state variable in a two-field formulation. The partial derivatives of $J_A$ and $R_A$ with respect to $v_G$ are computed as follows:

$$\frac{\partial J_A}{\partial v_G}^T = \frac{\partial G_{s\Delta}}{\partial v_G}^T \frac{\partial G}{\partial G_{s\Delta}}^T \frac{\partial J_A}{\partial G},$$

$$\frac{\partial R_A}{\partial v}^T \Psi_A = \frac{\partial G_{s\Delta}}{\partial v}^T \frac{\partial G}{\partial G_{s\Delta}}^T \frac{\partial R_A}{\partial G} \frac{\partial R_A}{\partial \Psi_A}.$$
where
\[
\frac{\partial G_s\Delta}{\partial v_G}^T = \left( \frac{\partial G_s\Delta}{\partial G_{sJ}} \bigg|_r \frac{\partial G_{sJ}}{\partial u_A} \bigg|_{G_{sJ}}^{\partial r} \frac{\partial u_A}{\partial v_G} \right)^T.
\] (A.11)

It is interesting to note that the partial derivative of $G_s\Delta$ with respect to $v_G$ has the same form as (5.33). Both expressions describe the sensitivity of the deflected geometry with respect to the geometric design variables. However, (A.11) is in terms of the discrete surface nodes, while (5.33) is in terms of the B-spline surface control points. The partial derivative term, $\partial J_S/\partial v_G$, is unmodified from purely structural optimization and from the three-field formulation. The matrix-vector product with the partial derivative of the structural residual with respect to the geometric design variables is evaluated by the following expression [73]:
\[
\frac{\partial R_S}{\partial v_G}^T \Psi_S = \left[ \frac{\partial R_S}{\partial X_S} \bigg|_r \frac{\partial X_S}{\partial v_G} - \frac{\partial f_S}{\partial v_G} - \frac{\partial G_{s\Delta}}{\partial v_G} \frac{\partial G_{s\Delta}}{\partial v_G} \right]^T \Psi_S.
\] (A.12)

The first term on the RHS of (A.12) accounts for any changes in $K_S$ due to changes in the structural geometry. The second term accounts for the changes in the rigid moment due to changes in the rigid links. The last term includes the influence of the deflected geometry on the surface traction and force integration calculations. The partial derivative of $G_{s\Delta}$ with respect to $v_G$ is as given in (A.11).
Appendix B

Recovering the Two-Field Formulation

B.1 Coupled Adjoint Problem and the Jacobian Matrix of $\mathbf{R}_{AS}$

This section illustrates the relationship between a two-field and a three-field formulation in terms of the Jacobian matrix of $\mathbf{R}_{AS}$. The coupled adjoint problem from (5.17) is used for this calculation, but the conclusions are applicable to the coupled Jacobian matrix used for the Newton-Krylov solution algorithm for the analysis problem in Section 6.1.

Below is the coupled adjoint problem in a three-field formulation:

$$
\begin{bmatrix}
\frac{\partial \mathbf{R}_A}{\partial \mathbf{q}} & \frac{\partial \mathbf{R}_S}{\partial \mathbf{q}} & 0 \\
0 & \frac{\partial \mathbf{R}_S}{\partial \mathbf{u}} & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{u}} \\
\frac{\partial \mathbf{R}_A}{\partial \mathbf{b}_\Delta} & \frac{\partial \mathbf{R}_S}{\partial \mathbf{b}_\Delta} & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta}
\end{bmatrix}
\begin{bmatrix}
\Psi_A \\
\Psi_S \\
\Psi_{M\Delta}
\end{bmatrix}
= 
\begin{bmatrix}
\frac{\partial \mathcal{J}}{\partial \mathbf{q}} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{u}} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{b}_\Delta}
\end{bmatrix}
$$

(B.1)

The third row of (B.1) can be used to eliminate the column related to $\Psi_{M\Delta}$:

$$
\begin{bmatrix}
\frac{\partial \mathbf{R}_A}{\partial \mathbf{q}} \\
\frac{\partial \mathbf{R}_S}{\partial \mathbf{q}} \\
- \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{u}} \left( \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} \right)^{-1} \frac{\partial \mathbf{R}_A}{\partial \mathbf{b}_\Delta} - \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{u}} \left( \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} \right)^{-1} \frac{\partial \mathbf{R}_S}{\partial \mathbf{b}_\Delta}
\end{bmatrix}
\begin{bmatrix}
\Psi_A \\
\Psi_S
\end{bmatrix}
= 
\begin{bmatrix}
\frac{\partial \mathcal{J}}{\partial \mathbf{q}} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{u}} \\
- \frac{\partial \mathcal{J}}{\partial \mathbf{b}_\Delta} + \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{u}} \left( \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_\Delta} \right)^{-1} \frac{\partial \mathcal{J}}{\partial \mathbf{b}_\Delta}
\end{bmatrix}
$$

(B.2)
Note that in a two-field formulation, it is assumed that $R_{M\Delta} = 0$ for every change in $u$. Therefore,

$$\left( \frac{\partial R_{M\Delta}}{\partial u} \right)_q = 0 = \left( \frac{\partial R_{M\Delta}}{\partial u} \right)_{q,b_\Delta}^T + \left( \frac{\partial b_\Delta}{\partial u} \right)_q^T \left( \frac{\partial R_{M\Delta}}{\partial b_\Delta} \right)_q^T \left( \frac{\partial R_{M\Delta}}{\partial b_\Delta} \right)^{-1}$$

(B.3)

In addition, any derivative terms with respect to $b_\Delta$ in (B.2) can be expanded as follows:

$$\frac{\partial (\cdot)}{\partial b_\Delta} = \frac{\partial G}{\partial b_\Delta} \frac{\partial (\cdot)}{\partial G}$$

(B.4)

Then using [B.3] and [B.4], (B.2) can be simplified to

$$\begin{bmatrix}
\frac{\partial R_A}{\partial q}^T \\
\frac{\partial G}{\partial u} \frac{\partial R_A}{\partial G}^T \\
\frac{\partial R_S}{\partial q}^T \\
\frac{\partial R_S}{\partial u} \frac{\partial R_S}{\partial G}^T
\end{bmatrix}
\begin{bmatrix}
\Psi_A \\
\Psi_S
\end{bmatrix}
= - \frac{\partial J}{\partial u} \frac{\partial G}{\partial u} - \frac{\partial G}{\partial v} \frac{\partial J}{\partial v}$$

(B.5)

The shaded terms in (B.5) are the result of eliminating the mesh state from a three-field formulation. From left to right, they correspond to (A.7), (A.8), and (A.9) for the two-field formulation covered in Appendix A. It can also be pointed out that the two-field coupled adjoint formulation is simply a row-reduced form of the three-field coupled adjoint problem. Therefore, the two systems have the same solutions.

### B.2 Gradient Calculations

This section further recovers the total gradient expression in (A.3) for a two-field formulation from the equivalent expression, (5.18), from a three-field formulation. The focus will be on the geometric design variables, as the mesh state does not affect the gradient calculations with respect to the flow and structural design variables. All partial derivatives in this section assumes that $q$ and $u$ are held fixed.

Using the last row of (B.1), the contribution of $\Psi_{M\Delta}$ to the total gradient expression can be rewritten in terms of the solution to the two-field coupled adjoint problem:

$$\frac{\partial R_{M\Delta}}{\partial v_G}^T \Psi_{M\Delta} = \frac{\partial R_{M\Delta}}{\partial v_G}^T \left( \frac{\partial R_{M\Delta}}{\partial b_\Delta} \right)^{-1} \left[ - \frac{\partial J}{\partial b_\Delta}^T \frac{\partial R_A}{\partial b_\Delta} - \frac{\partial R_S}{\partial b_\Delta} \frac{\partial \Psi_A}{\partial \Delta} - \frac{\partial R_S}{\partial \Delta} \frac{\partial \Psi_S}{\partial \Delta} \right]$$

(B.6)

A two-field formulation implicitly assumes that the flow grid changes instantaneously with any changes in the geometric design variables. Although there is no change in $u$ and $b_J$, changes in $r$ will change $u_A$:

$$\left( \frac{\partial R_{M\Delta}}{\partial v_G} \right)^T_{b_J} = 0 = \left( \frac{\partial R_{M\Delta}}{\partial v_G} \right)^T_{b_J,b_\Delta} + \left( \frac{\partial b_\Delta}{\partial v_G} \right)^T_{b_J} \left( \frac{\partial R_{M\Delta}}{\partial b_\Delta} \right)^T_{b_J,v_G} \left( \frac{\partial R_{M\Delta}}{\partial b_\Delta} \right)^{-1}$$

(B.7)
Using (B.7), (B.8) can be simplified in a similar way as in (B.2):

\[
\frac{\partial R_{M\Delta}}{\partial v} \Psi_{M\Delta} = \left( \frac{\partial G}{\partial v} \right)_{b_j}^T \frac{\partial J}{\partial G}^T + \left( \frac{\partial G}{\partial v} \right)_{b_j} \frac{\partial R_A}{\partial G}^T \Psi_A + \left( \frac{\partial G}{\partial v} \right)_{b_j} \frac{\partial R_S}{\partial G}^T \Psi_S \quad (B.8)
\]

The same procedure can be repeated for the jig shape mesh adjoint equation:

\[
\frac{\partial R_{MJ}}{\partial b_j} \Psi_{MJ} = -\frac{\partial R_{M\Delta}}{\partial b_j} \left( \frac{\partial R_{M\Delta}}{\partial b}^T \right)^{-1} \left[ -\frac{\partial J}{\partial b}^T \frac{\partial R_A}{\partial b}^T \Psi_A - \frac{\partial R_S}{\partial b}^T \Psi_S \right] \\
= -\frac{\partial G}{\partial b_j}^T \left( \frac{\partial J}{\partial v} \right)_{vG}^T - \frac{\partial G}{\partial b_j} \frac{\partial R_A}{\partial G}^T \Psi_A + \frac{\partial G}{\partial b_j} \left( \frac{\partial R_S}{\partial G} \right)_{vG}^T \Psi_S \\
= \left( \frac{\partial G}{\partial v} \right)_{vG}^T \frac{\partial J}{\partial G}^T + \left( \frac{\partial G}{\partial v} \right)_{vG} \frac{\partial R_A}{\partial G}^T \Psi_A + \left( \frac{\partial G}{\partial v} \right)_{vG} \frac{\partial R_S}{\partial G}^T \Psi_S \quad (B.9)
\]

Equation (B.9) hence accounts for the changes in the final flow grid, which includes the structural deflections, due to changes in the jig shape grid. Using (B.9), the contribution from the jig shape mesh adjoint variable to the total gradients then becomes

\[
\frac{\partial R_{MJ}}{\partial v} \Psi_{MJ} = \frac{\partial R_{MJ}}{\partial v} \left( \frac{\partial R_{MJ}}{\partial b_j}^T \right)^{-1} \frac{\partial R_{MJ}}{\partial b_j} \left[ -\frac{\partial J}{\partial b}^T \frac{\partial R_A}{\partial b}^T \Psi_A - \frac{\partial R_S}{\partial b}^T \Psi_S \right] \\
= \left( \frac{\partial G}{\partial v} \right)_{vG}^T \frac{\partial J}{\partial G}^T + \left( \frac{\partial G}{\partial v} \right)_{vG} \frac{\partial R_A}{\partial G}^T \Psi_A + \left( \frac{\partial G}{\partial v} \right)_{vG} \frac{\partial R_S}{\partial G}^T \Psi_S \quad (B.10)
\]

The above expression includes the effects of changing the grid for the undeformed shape.

Equations (B.8) and (B.10) can be substituted into (5.18) for the total gradient. Collecting all the partial derivative terms related to \( J, R_A, \) and \( R_S \) results in the following:

\[
\mathcal{G} = \frac{\partial J}{\partial v}^T \frac{\partial v}{\partial v}^T \Psi_A + \frac{\partial R_A}{\partial v}^T \Psi_S + \frac{\partial R_S}{\partial v}^T \Psi_S \\
\text{where}
\]

\[
\frac{\partial J}{\partial v}^T = \frac{\partial J}{\partial v} \left| _G \right. + \frac{\partial G}{\partial v} \left| _r \right. \frac{\partial J}{\partial v} + \frac{\partial G}{\partial v} \left| _b \right. \frac{\partial J}{\partial v} \\
\frac{\partial R_A}{\partial v}^T \Psi_A = \left[ \frac{\partial R_A}{\partial v} \left| _G \right. + \frac{\partial G}{\partial v} \left| _r \right. \frac{\partial R_A}{\partial v} + \frac{\partial G}{\partial v} \left| _b \right. \frac{\partial R_A}{\partial v} \right] \Psi_A \\
\frac{\partial R_A}{\partial v}^T \Psi_A = \left[ \frac{\partial R_S}{\partial v} \left| _G \right. + \frac{\partial G}{\partial v} \left| _r \right. \frac{\partial R_S}{\partial v} + \frac{\partial G}{\partial v} \left| _b \right. \frac{\partial R_S}{\partial v} \right] \Psi_A \\
\frac{\partial R_S}{\partial v}^T \Psi_A = \left[ \frac{\partial R_S}{\partial v} \left| _G \right. + \frac{\partial G}{\partial v} \left| _r \right. \frac{\partial R_S}{\partial v} + \frac{\partial G}{\partial v} \left| _b \right. \frac{\partial R_S}{\partial v} \right] \Psi_A \\
\]

Each of (B.13), (B.14), and (B.15) consists of three terms. The first of the three terms is from the original total gradient expression using the augmented three-field coupled adjoint formulation, which assumes a fixed flow grid in taking the partial derivatives. The second and third terms are contributions from \( \Psi_{MJ} \) and \( \Psi_{M\Delta} \) to the total gradient values. They encompass the changes in the flow grid due to changes in the jig shape and changes in the structural deflections, respectively. It should be noted that the inverse of \( (\partial R_{M\Delta}/\partial b) \) and \( (\partial R_{MJ}/\partial b) \) are still necessary to evaluate the derivative of the flow grid with respect to the geometric design variables, which is now present in multiple term. This then shows that the augmented three-field formulation used in Chapter 5 has greatly simplified the derivative term calculations in comparison to a two-field formulation.
Appendix C

A Modular Schur Preconditioner

Section 6.1.3 in Chapter 6 discusses the importance of modularity in preconditioning the monolithic solution to coupled problems. It further draws the connection between the block Jacobi/Gauss-Seidel preconditioner and a Schwarz preconditioner that treats each of the subproblem as a subdomain. A similar analogy can be made between the Schur complement preconditioner from domain decomposition and the linearized Jacobian matrix for the interface problem, which is discussed in Section 1.4.2. This is accomplished by treating the flow and mesh subproblems together as a subdomain, and the structural subproblem as another. The Schur complement problem is then in terms of the unknowns at the aerostructural interface. This process recovers the Schur complement problem in the Schur-Newton-Krylov approach by Barcelos et al. \cite{barcelos2000}. A Schur preconditioner formulated this way preserves the modularity between subproblems during monolithic calculations.

To obtain such a Schur complement matrix in the context of the present methodology, an interface equation

\[
\begin{align*}
\mathbf{R}_{s\Delta} &= \mathbf{b}_{s\Delta} - \mathbf{D}(\mathbf{u}) 
\end{align*}
\]

in terms of the coordinates of the B-spline surface control points, or \( \mathbf{b}_{s\Delta} \), can be defined. The matrix, \( \mathbf{D} \), can be the displacement transfer operator that translates the structural deflections into an appropriate set of B-spline surface control points describing the deflected geometry. By adding \( \mathbf{R}_{s\Delta} \) to the three-field formulation of the coupled analysis problem and treating \( \mathbf{b}_{s\Delta} \) as another state variable, the coupled Jacobian matrix, \( \mathbf{A} \), in (6.1) becomes

\[
\mathbf{A}_{\text{mod}} = \begin{bmatrix}
\frac{\partial \mathbf{R}_{\Delta}}{\partial \mathbf{q}} & \frac{\partial \mathbf{R}_{\Delta}}{\partial \mathbf{b}_{\Delta}} & 0 & 0 \\
0 & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_{\Delta}} & 0 & \frac{\partial \mathbf{R}_{M\Delta}}{\partial \mathbf{b}_{s\Delta}} \\
\frac{\partial \mathbf{R}_{S}}{\partial \mathbf{q}} & \frac{\partial \mathbf{R}_{S}}{\partial \mathbf{b}_{\Delta}} & \frac{\partial \mathbf{R}_{S}}{\partial \mathbf{u}} & 0 \\
0 & 0 & -\frac{\partial \mathbf{D}}{\partial \mathbf{u}} & 1
\end{bmatrix}.
\]
Appendix C. A Modular Schur Preconditioner

The Schur complement matrix of (C.2) for the interface unknowns is as follows:

\[
S = I - \begin{bmatrix}
0 & 0 & -\frac{\partial D}{\partial u} \\
0 & \frac{\partial R_A}{\partial q} & \frac{\partial R_A}{\partial b_\Delta} \\
\frac{\partial R_S}{\partial q} & \frac{\partial R_S}{\partial b_\Delta} & \frac{\partial R_S}{\partial u}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial R_M}{\partial b_\Delta} \\
0 \\
0
\end{bmatrix}^{-1}.
\]

(C.3)

After recognizing that the matrix \( B \) is permuted from a block triangular matrix, it is not difficult to show that the matrix-vector product with the Schur complement matrix, \( S \), can be evaluated by

\[
Sz = z + \frac{\partial D}{\partial u} \left( \frac{\partial R_S}{\partial u} \right)^{-1} \left[ -\frac{\partial R_S}{\partial b_\Delta} + \frac{\partial R_S}{\partial q} \left( \frac{\partial R_A}{\partial q} \right)^{-1} \frac{\partial R_A}{\partial b_\Delta} \right] \left( \frac{\partial R_M}{\partial b_\Delta} \right)^{-1} \frac{\partial R_M}{\partial b_\Delta} z.
\]

(C.4)

Equation (C.4) has the same form as the Schur complement matrix described by Barcelos et al. However, the use of B-spline surfaces to describe the deflected geometry in the present methodology further reduces the size of (C.4) comparing to other interface approaches in the literature, which are often in terms of the nodal variables at the interface. If \( x \) is the solution to the coupled linear system \( A x = g \), then it can be shown that the modified linear system \( A_{\text{mod}} x_{\text{mod}} = g_{\text{mod}} \) has the solution \( x_{\text{mod}} = \begin{bmatrix} x & \star \end{bmatrix}^T \) by choosing \( g_{\text{mod}} = \begin{bmatrix} g & 0 \end{bmatrix}^T \). The \( \star \) in \( x_{\text{mod}} \) denotes interface variables that are not part of the solution vector to the original linear system. This means that the Schur complement matrix in (C.4) can be used to precondition the iterative solution to (6.3) without requiring modifications to the original problem.

The Schur complement matrix in (C.4) is similar to one from traditional domain decomposition in that it can be evaluated using the local preconditioners of the individual subdomains, or in this case, subproblems. Furthermore, (C.4) is modular in that the same block preconditioners from Chapter 6 can be used. A Schur complement in this form can likely reduce the number of coupled Krylov iterations even further than what is currently accomplished using a block Gauss-Seidel preconditioner.

While the cost to one evaluation of (C.4) is similar to one evaluation of the block Gauss-Seidel preconditioner, many matrix-vector product evaluations could be necessary without an effective preconditioner for the Schur complement problem. Unlike the Schur complement preconditioner for the flow problem, the Schur complement in (C.4) cannot be preconditioned by the incomplete LU factorization of the local problem on each processor. The search of an effective preconditioner for the Schur complement problem could be an interesting problem for future work. However, until such a preconditioner is found, the modular Schur preconditioner described here will likely not present a competitive option for the monolithic solution method developed for this work.