I, Robert Knapke, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Aerospace Engineering.

It is entitled:
High-Order Unsteady Heat Transfer with the Harmonic Balance Method

Student's name: Robert Knapke

This work and its defense approved by:

Committee chair: Mark Turner, Sc.D.
Committee member: John A. Benek, Ph.D.
Committee member: Shaaban Abdallah, Ph.D.
Committee member: Paul Orkwis, Ph.D.
High-Order Unsteady Heat Transfer with the Harmonic Balance Method

A dissertation submitted to the Graduate School of the University of Cincinnati in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the School of Aerospace Systems of the College of Engineering by

Robert David Knapke

B.S., University of Cincinnati, 2009
M.S., University of Cincinnati, 2011

Date 4/2/2015

Committee Chair: Mark G. Turner, Sc.D.
Abstract

Despite the significant advancements in computational fluid dynamics, modeling turbomachinery flows remains extremely difficult. The challenges include complex unsteady blade row interactions, large thermal gradients, and complex geometries. During the design process, simplifications and approximations are necessary to reduce the computational cost. Two common simplifications are the use of adiabatic boundary conditions and steady methods for resolving the flow field in multistage turbomachinery.

The Harmonic Balance (HB) method is an efficient way to simulate periodic unsteady phenomena. Compared to traditional time-marching methods, the HB method reduces the computational cost by considering only the dominant frequencies of the solution field. Using a Fourier series representation of the solution variables, an unsteady governing equation transforms into a series of steady-like equations. The cost is further reduced when considering multistage turbomachinery. Unlike a time-marching method, which requires periodic boundaries, the HB method models a single blade passage per blade row. A phase lag condition is applied instead of a periodic condition. At the junction between blade rows, an interface resolves the relative motion and any passage mismatch.

Assuming the heat transfer of wall boundary conditions is, by definition, non-physical. Assuming no heat transfer (an adiabatic wall) is certainly wrong for turbomachinery flows because of the large thermal gradients. An constant temperature wall can provide a better approximation, but for complex geometries, the temperature is not known a priori. The most accurate approach involves modeling both the fluid and solid domains, which is called Conjugate Heat Transfer (CHT). This can be performed in several ways, but the most stable method is one in which the fluid and solid are strongly coupled. This strong coupling is achieved by using the same discretization in both domains.

This dissertation details an approach for accurately and efficiently simulating the unsteady heat transfer in turbomachinery flows. The HB and CHT methods are developed within a framework that uses a
high-order Discontinuous Galerkin (DG) spatial discretization and a versatile Chimera overset scheme. The implemented HB method is fully linearized, allowing the use of an efficient Quasi-Newton solver. The HB equations are coupled within a linear system that includes the linearized HB pseudo-spectral operator. Phase lag and relative motion interfaces are included, hence the computational domain of multistage turbomachinery simulations is reduced to one passage per blade row. Modeling turbulent flows is achieved with the Spalart-Allmaras turbulence model. A strongly coupled CHT method is introduced. The solid and fluid domains are both discretized with the DG method. The fluid to solid interface enforces a consistent wall temperature and heat flux.

The Harmonic Balance method, the Conjugate Heat Transfer method, and the Spalart-Allmaras turbulence model are independently verified using a series of test cases. In addition, CHT on curved 3D geometries is performed for the first time. Lastly, unsteady heat transfer is simulated using the combination of the HB and CHT methods. These test cases demonstrate the fast convergence and accurate modeling of the implemented methods. This work provides the basis for the accurate and efficient simulation of turbomachinery flows.
Acknowledgments

Getting to this point has been a long, and at times difficult, process. I can safely say that I would not be here without the support of many people.

I am extremely grateful for the guidance given by my advisor Dr. Mark Turner. His passionate interest in the understanding of turbomachinery flows and CFD in general has been a driving force during my graduate studies. He is equally enthusiastic about teaching. Mark is always willing to set aside time to work through problems with all of his students. The many discussions we have had over the years have greatly benefited me.

I would also like to thank the rest of my committee members, both for their guidance and for the time they have taken to review my research and dissertation. The classes I have taken from Dr. Paul Orkwis have provided a foundation for my understanding of fluid dynamics and numerical methods. His strong knowledge in the subject has been an excellent resource for many years. My first few classes relating to turbomachinery flows were taught by Dr. Shaaban Abdallah. His many interesting ideas (most of which I overheard during my meetings with Mark) have inspired me to search for creatives solutions to engineering problems. Dr. John Benek has a knack for posing difficult, yet thought provoking questions about my research. His unique perspective and review has undoubtedly improved the quality of this dissertation, in particular my discussion of the Harmonic Balance method.

Graduate research can be hard, at times, which makes a light-hearted yet research focused environment vital. Over my many years at the Gas Turbine Simulation Laboratory, I have been blessed to work with so many great people. I owe a big debt of gratitude to Marshall Galbraith. Not only did he allow me to work with his code for this research, but he also took the time to explain his code and software development in general. Most of my C++ knowledge is derived from him. Also, his strong work ethic has been an invaluable source of motivation. I would also like to thank Kiran Siddappaji and Andy Schroder for their
support. Our discussions were a bright spot in what was sometimes a difficult process. Kiran helped me with blade geometry, but was also good at giving pep-talks. His positive attitude in the face of sometimes daunting problems was always welcome. I always enjoyed my conversations with Andy, whether they were related to engineering or to more philosophical ideas.

I owe many thanks to my family. My parents, Deb and Tony Knapke, have always supported me and provided a great example of the benefits of a hard-working attitude. I would also like to thank my older sister, Sarah, who was the one that suggested I look into the University of Cincinnati. She made those early years of college less stressful. In addition, I need to thank my older brother John. Little did I know that all of the fights we had as kids would have pushed to me achieve more.

Throughout the good and the bad, my loving wife Amy has always been by my side. She has taken excellent care of our children. I know that my research has meant that more of the parenting burden has been placed on her, especially when I have worked late into the night and through the weekends. Amy has always been there through one of the longest, most exhausting, sometimes disheartening, and yet most rewarding exercises in my life (this statement works for the dissertation, too). I am overly grateful for my parents (Grandpa and Grandma) and Amy’s parents (Mimi and Poppy) for helping with the kids (and our sanity). They have been a great foundation for Amy and me as we build our family. I would also like to thank my children, Lily and Luke. They are too young now to know how much sleep and energy they drain from me, but they also don’t know that the joy they give me is beyond words.

I would like to thank the Ohio Space Grant Consortium (OSGC) Fellowship Program for their financial support. I also received their support as an undergrad, which is greatly appreciated. They provide much needed financial assistance, but also show interest in all of the projects they support. I would also like to thank the Defense Advanced Research Projects Agency (DARPA) for the financial assistance I received in conjunction with the SBIR Contract numbered W911QX-13-C-0133.
# Contents

Abstract ii

Contents vii

List of Figures xi

List of Tables xv

Nomenclature xvi

1 Introduction 1

1.1 Motivation ......................................................... 1

1.2 Capturing Multistage Turbomachinery Flow Physics ................. 2

1.2.1 Time-Marching Methods ........................................ 3

1.2.2 Steady Methods .................................................. 4

1.2.3 Reduced Unsteady Methods ...................................... 5

1.2.4 Mixed Steady-Unsteady Methods ................................ 6

1.2.5 Frequency Domain Methods ..................................... 6

1.3 Capturing Fluid-Solid Heat Transfer ................................ 9

1.3.1 Coupling Fluid and Solid Simulations .......................... 10

1.4 Dissertation Objectives ............................................ 13

1.4.1 The DG-Chimera Framework ..................................... 14

1.4.2 Outline .......................................................... 15
# Methodology

2.1 Spatial Discretization ................................. 16
   2.1.1 Discontinuous Galerkin Method ...................... 17

2.2 Governing Equations ................................. 22
   2.2.1 Solid Conduction Equation ......................... 23
   2.2.2 Fluid Flow Equations .............................. 25

2.3 Harmonic Balance Method ............................. 39
   2.3.1 Newton’s Method .................................. 43
   2.3.2 Turbomachinery Flows .............................. 45

2.4 Conjugate Heat Transfer .............................. 60
   2.4.1 Fluid Domain Interface Conditions ................. 61
   2.4.2 Solid Domain Interface Conditions ................ 62
   2.4.3 Extension to Unsteady Fluid-Solid Heat Transfer with the Harmonic Balance Method 63

# Implementation

3.1 Implicit Chimera Communication ..................... 64
3.2 Turbulence Model ................................... 65
   3.2.1 Wall Distance Calculation ......................... 66
3.3 Steady Solution Advancement .......................... 70
3.4 Harmonic Balance Method ............................... 73
   3.4.1 Solution Advancement ............................... 75
   3.4.2 Linear System Preconditioning ..................... 76
   3.4.3 Phase Lag Interface ................................ 81
   3.4.4 Relative Motion Interface ......................... 82
   3.4.5 Harmonic Balance Interface Study ................. 86
3.5 Conjugate Heat Transfer ............................... 94
   3.5.1 Chimera Boundaries ............................... 94
   3.5.2 Gap Study ....................................... 95

# Verification

4.1 Steady ............................................. 100
7.1.1 Turbulence Model .................................................. 170
7.1.2 Non-Reflective Boundaries ....................................... 171
7.1.3 Computational Performance ...................................... 171
7.1.4 HB Phase Lag and Relative Motion Interfaces ............... 172
7.1.5 Chimera Connection at Conjugate Heat Transfer Interfaces .................................................. 172

Bibliography ................................................................. 173

A Fluid Flow Jacobian Matrices ........................................ 186
  A.1 Fluxes ........................................................................ 186
  A.2 Source Term ............................................................. 192
  A.3 Upwind Dissipation .................................................... 197
  A.4 Time Term ............................................................... 201
  A.5 Boundary Conditions ................................................ 202

B Spalart-Allmaras Conservative Form .................................. 210

C Iterative Method Preconditioning ...................................... 212
  C.1 Harmonic Balance Lower Upper Symetric Gauss-Seidel Preconditioner ................................................. 214
  C.2 Harmonic Balance Incomplete Lower Upper Decomposition Preconditioner ................................................. 215

D Moore-Penrose Generalized Matrix Inverse ......................... 221
List of Figures

2.1 Multi-domain problem. ......................................................... 17
2.2 Spatial discretization notation. .............................................. 18
2.3 Multistage turbomachinery domain orientation. ......................... 46
2.4 Comparison between periodic and aperiodic pitchwise perturbations. 49
2.5 Two blade passage domains. .................................................. 53
2.6 Conjugate heat transfer domains. ........................................... 61
2.7 Notation for fluid and solid sides of the interface. ....................... 62
3.1 Example positioning of a mesh and wall boundaries. ..................... 67
3.2 Example mesh lines and nodes (blue) and wall faces and nodes (green). 68
3.3 Contours of wall distance for several geometries. .......................... 69
3.4 Contours of wall distance for the flap domain. ............................ 70
3.5 Schematic for the CFL scheduling based on the number of iterations to solve the linear system. 73
3.6 Schematics of an existing solver and HB solver implementation. ........ 74
3.7 Harmonic Balance simulation process. .................................... 74
3.8 Number of iterations to achieve FGMRES convergence. ................. 78
3.9 Averaged number of FGMRES iterations. .................................. 79
3.10 Representation of the structure of the HB linearization matrix. .......... 80
3.11 Representation of the rearranged HB linear system matrix. ............. 80
3.12 Illustration of the passing of periodic faces. .............................. 81
3.13 Two domains moving relative to each other. .............................. 82
3.14 Relative motion receiver faces generated at several instances in time. 83
3.15 Two methods to fill orphaned nodes of relative motion faces. .......... 85
3.16 Blue domain receiver faces and copies at several time steps. .............................. 86
3.17 Two overlapping domains moving relative to each other. .................................... 86
3.18 Mesh of domains moving relative to each other. ................................................... 87
3.19 Convergence histories for $r_{sum} = 3$ and $K = 5$. ........................................ 88
3.20 Instantaneous contours lines of temperature for $K = 5$, and $q = 4$. ............... 89
3.21 Total temperature contours for $K = 5$, and $q = 4$ with relative velocity vectors. 89
3.22 Temperature along both sides of the interface at an instant in time. ....................... 90
3.23 Convergence histories for $K = 1$. ................................................................. 92
3.24 Solution for $K = 3$, and $q = 2$. Inflow bottom left, outflow top right. ............ 92
3.25 Temperature along both sides of the interface at an instant in time. ....................... 93
3.26 Mesh configurations with domains of different equation sets. ............................ 95
3.27 Four example meshes for the circular arc with $m = 4$. .................................. 96
3.28 Amount of overlap required for the connection process to pass for the circular arc case. .......................................................... 97
3.29 Four example meshes for the 3D polynomial shape with $m = 1$. ....................... 98
3.30 Amount of overlap required for the connection process to pass for the 3D polynomial shape case. .......................................................... 99

4.1 Mesh of the smooth bump domain with $12 \times 4$ cells. ..................................... 101
4.2 Smooth bump case convergence history for the $48 \times 16$ mesh. ........................ 101
4.3 Coefficient of pressure for $q = 0$ and $q = 4$ on the $48 \times 16$ mesh. .............. 102
4.4 Entropy error for a series of uniformly refined meshes at different solution polynomial orders. Black lines indicate the expected slope. .......................................................... 103
4.5 Mesh of the channel domain with $20 \times 4$ cells. .................................................. 104
4.6 Convergence history for channel flow on the $40 \times 8$ mesh. ............................... 105
4.7 Streamwise velocity contours for $q = 4$ on a $40 \times 8$ cell mesh. ...................... 105
4.8 Pressure gradient in the x-direction for $q = 4$ on a $40 \times 8$ cell mesh. .............. 105
4.9 Streamwise velocity profile at the exit of the viscous channel domain. .................. 106
4.10 Mesh of the flat plate domain with $18 \times 12$ cells. ............................................. 107
4.11 Convergence history for turbulent flat plate and the $36 \times 24$ mesh. ................. 108
4.12 Boundary layer eddy to freestream viscosity ratio contours. .............................. 109
4.13 Contours of the S-A working variable depicting Gibbs phenomena at the plate leading edge. 109
4.14 Profiles of streamwise velocity at \( x = 1.095 \). 110
4.15 Profiles of eddy to freestream viscosity ratio at \( x = 1.095 \). 111
4.16 S-A working variable profile comparison at \( x = 1.095 \). 111
4.17 Coefficient of friction along the plate. 112
4.18 Plate coefficient of drag for varying mesh sizes. 113
4.19 Coefficient of friction at \( x = 1.095 \) for varying mesh sizes. 113
4.20 Inner wall variable profiles at \( x = 1.095 \). 114
4.21 Chimera meshes for the \( 18 \times 12 \) mesh. 115
4.22 Comparison between single and chimera solutions for the \( 18 \times 12 \) mesh. 116
4.23 Solutions to the linear advection equation with a sinusoidally varying boundary. 119
4.24 Solutions to Poisson’s equation with a sinusoidally varying boundary. 121
4.25 Comparison of the HB solution and the TM solutions with different time steps at \( t = 10 \). 122
4.26 Cascade computational domain. 123
4.27 HB simulation convergence histories. 124
4.28 Instantaneous total temperature contours for HB \( K = 5 \) and TM with 160 time steps per period. 125
4.29 1\textsuperscript{st} mode pressure magnitude for HB \( K = 5 \) and TM with 160 time steps per period. 126
4.30 2\textsuperscript{nd} mode pressure magnitude for HB \( K = 5 \) and TM with 160 time steps per period. 126
4.31 3\textsuperscript{rd} mode pressure magnitude for HB \( K = 5 \) and TM with 160 time steps per period. 127
4.32 4\textsuperscript{th} mode pressure magnitude for HB \( K = 5 \) and TM with 160 time steps per period. 127
4.33 5\textsuperscript{th} mode pressure magnitude for HB \( K = 5 \) and TM with 160 time steps per period. 128
4.34 Comparison of 1\textsuperscript{st} mode pressure magnitude extracted along a line mid-passage. 129
4.35 Cylinder computational domain. 130
4.36 HB simulation convergence histories for the shedding cylinder. 131
4.37 Coefficients of lift and drag over one period for the TM and several HB simulations. 132
4.38 Comparison of vorticity contours in the cylinder wake. 132
4.39 Conduction CHT simulation setup. 133
4.40 Conduction simulation convergence histories. 134
4.41 Solutions for varying \( \frac{k_s}{k_f} \) with \( q = 1 \) and a mesh of 5x10 in each domain. 135
4.42 Source driven Poiseuille flow simulation setup. 135
4.43 Convergence history for the convection simulation on the 16x16 mesh. .......................... 137
4.44 Solutions with \( q = 3 \) and a mesh of 10x10 in each domain. ........................................ 138
4.45 Error metrics for a series of uniformly refined meshes with different solution polynomial 
orders. ......................................................................................................................... 139
5.1 Mesh of the curved duct domain. ................................................................................. 142
5.2 Curved duct iteration history for different thermal conductivity ratios. ................. 143
5.3 Solution for \( q = 3 \) and \( \frac{k}{k_f} = 1 \). ............................................................................ 144
5.4 Location of slices extracted for comparison. ............................................................. 145
5.5 Fluid domain streamwise vorticity and tangential velocity vectors for three different poly-
nomial orders. ............................................................................................................. 146
5.6 Contour lines of temperature and temperature gradients for three different polynomial orders. 147
5.7 Percent difference of the integrated heat transfer on both sides of CHT interface. .... 148
5.8 Contours of temperature and streamwise vorticity for varying thermal conductivity ratios. . 149
5.9 Contours of temperature gradients and streamwise velocity for varying thermal conductivity ratios. ........................................................................................................... 150
5.10 Comparison of tangential velocity vectors. Fluid domain only. ............................... 151
5.11 Meshes for the unsteady channel flow simulation. ..................................................... 152
5.12 Unsteady channel flow convergence histories. ......................................................... 154
5.13 Unsteady channel flow convergence histories for varying \( K \). .................................. 155
5.14 Contours of temperature and heat flux in the y-direction at \( t = 0, q = 2, K = 1, \frac{k}{k_f} = 1 \). .. 155
5.15 Percent difference of the integrated heat transfer on both sides of CHT interface. .... 156
5.16 Location of slice extracted for comparison. ............................................................. 156
5.17 Solution contours for varying thermal conductivity ratios. ....................................... 157
5.18 Comparison of tangential velocity vectors at \( t = 0 \). .................................................. 158
6.1 Execution time for several preconditioners, orders of accuracy, and number of HB frequencies. 161
# List of Tables

4.1 Entropy error order of accuracy slopes. .................................................. 103
4.2 Mesh sizes and first cell $y^+$ values at $x = 1.095$. ................................. 113
4.3 Error for the HB and TM methods at $t = 0$. ........................................ 118
4.4 $L^2$-norm of the difference between the HB solution and TM solutions at $t = 10$. 120
4.5 Sizes for the cascade meshes. Numbers correspond to Fig. 4.26a. .................. 122
4.6 Time averaged lift and drag coefficients. ............................................... 131
4.7 Temperature error order of accuracy slopes. ....................................... 139
4.8 Heat transfer error order of accuracy slopes. ...................................... 139

6.1 Speedup of the HB ILU(0) preconditioner relative to the other preconditioners for spatial high order accuracy. .................................................. 161
6.2 Execution time (in hours) for the TM and HB simulations of the cascade. .... 162
6.3 CPU hours and number of cores for the TM and HB simulations of the shedding cylinder. 163
6.4 Execution time (in minutes) for simulations of the relative motion interface in two dimensions. 164
6.5 CPU hours and number of cores for the simulations of the 3D curved duct. .... 164
6.6 CPU hours and number of cores for the simulations of the 3D unsteady channel flow. 165
Nomenclature

General

$\vec{F}$  \hspace{0.5cm} Flux

$\vec{F}^a$  \hspace{0.5cm} Advective Flux

$\vec{F}^d$  \hspace{0.5cm} Diffusive Flux

$\nabla$  \hspace{0.5cm} Gradient vector, $\left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$

$Q$  \hspace{0.5cm} Conservative Variable Vector

$S$  \hspace{0.5cm} Source Term

$t$  \hspace{0.5cm} Temporal Dimension

$W$  \hspace{0.5cm} Dependent Variable Vector

$x, y, z$  \hspace{0.5cm} Cartesian Coordinates

Discretization

$\eta_e$  \hspace{0.5cm} BR2 Stabilization Coefficient

$\Delta W^m$  \hspace{0.5cm} Newton Solver Update Vector

$\Delta t_e$  \hspace{0.5cm} Time Step of the Cell $e$

$\Gamma_a$  \hspace{0.5cm} Artificial Spatial Boundary

$\Gamma_i^b$  \hspace{0.5cm} Spatial Boundary of Domain $i$
\( \Gamma_e \) Spatial Boundary of the Cell \( e \) in Domain \( i \)

\( h \) Representative Cell Size \((1/DOF \text{ for 1D, } \sqrt{1/DOF} \text{ for 2D})\)

\( \vec{n} \) Spatial Cell Boundary Unit Normal Vector

\( \Omega_e \) Volume of the Cell \( e \) in Domain \( i \)

\( \phi \) Numerical Dissipation Vector

\( \psi \) Polynomial Basis/Test Function

\( \mathcal{R}(U) \) Sum of Spatial Integrals

\( \vec{\tau} \) Lifting Operator for BR2 Viscous Scheme

\( \vec{R} \) Sum of Lifting Operators \( \vec{\tau} \) for BR2 Viscous Scheme

\( \Sigma \) Interface Spatial Boundary

**Governing Equations**

\( \alpha \) Thermal Diffusivity

\( \beta_{ij} \) Viscous Work Terms

\( c \) Speed of Sound

\( C_S \) Sutherland Constant

\( E \) Total Energy

\( \gamma \) Ratio of Specific Heats

\( H \) Total Enthalpy

\( k \) Thermal Conductivity

\( \lambda \) Characteristic Speed

\( \mu \) Dynamic Viscosity
\( \vartheta \) Spalart-Allmaras Working Variable

\( p \) Static Pressure

\( \sigma_{e,j} \) Spalart-Allmaras Viscous Term

\( Pr \) Prandtl Number

\( Re \) Reynolds Number

\( \rho \) Density

\( \rho u, \rho v, \rho w \) Cartesian Momentum Components in x, y, and z Directions

\( s \) Entropy

\( S_{\vartheta} \) Source Term for the Spalart-Allmaras Turbulence Model

\( T \) Temperature

\( \tau_{e,x,j} \) Viscous Stress Tensor

\( U \) Primitive Variable Vector

\( u, v, w \) Cartesian Velocity Components in x, y, and z Directions

\( \vec{V} \) Velcoity Vector \[ u, v, w \]

**Harmonic Balance**

\( A_k, B_k \) Fourier Coefficients

\( A_k^\prime, B_k^\prime \) Phase Shifted Fourier Coefficients

\( B \) Blade Row Periodicity

\( D \) Pseudo-Spectral Operator

\( \mathcal{D}^* \) Pseudo-Spectral Residual

\( E \) Fourier Transform Matrix
$E^{-1}$  Inverse Fourier Transform Matrix

$F$  Filter Operator

$F^*$  Series of Fluxes

$K$  Number of Frequencies

$N$  Number of Time Levels

$n$  Frequency Contribution Matrix

$N'$  Nodal Diameter, renamed HB diameter for this work

$\omega$  List of Frequencies

$\vec{\omega}$  Rotation Speed Vector

$\omega_k$  $k^{th}$ Frequency

$P$  Phase Shift Operator

$\Phi$  Phase Shift Matrix

$\hat{Q}$  Conservative Variable Fourier Coefficient Vectors

$Q^*$  Series of Conservative Solution Vectors

$\hat{R}^*$  Series of Spatial Integral Sums

$\sigma$  Spatial Extent of a Perturbation

$S^*$  Series of Source Terms

$t$  List of Time Levels

$\Delta \theta$  Blade Pitch

$T_{i,j}$  Perturbation Period Induced by Blade Row $i$ on Blade Row $j$

$t_n$  $n^{th}$ Time Level
Subscript

∞   Reference Quantities  
F   Filtered Quantity  
Σ   Interface Quantity  
ϕ   Phase Shifted Quantity

Superscript

−, +   Cell Interior and Exterior Values

Acronyms

BEM   Boundary Element Method  
CFD   Computational Fluid Dynamics  
CFL   Courant-Freidrichs-Lewy Number  
CHT   Conjugate Heat Transfer  
DG    Discontinous Galerkin  
FD    Frequency Domain  
FEA   Finite Element Analysis  
FGMRES Flexible Generalized Minimum Residual Algorithm  
GQ    Gauss Quadrature  
HB    Harmonic Balance  
NLFD Non-Linear Frequency Domain  
NLH   Non-Linear Harmonic  
TM    Time-Marching  
TS    Time Spectral
Chapter 1

Introduction

1.1 Motivation

Turbomachines by their very nature are unsteady. A simple exercise shows that without unsteadiness, the turbomachine, paradoxically, neither extracts work from, nor adds work to, the system [1]. The interactions of spinning blades induce pressure fluctuations that relate directly to enthalpy change. In addition to the enthalpy change in the main flow path, the combustor produces extreme enthalpy variations. In most modern turbomachines, the high combustor exit temperatures are mitigated by complex internal air systems in an effort to protect the metal parts. The combination of large enthalpy gradients and unsteadiness gives rise to non-linear unsteady phenomena, such as hot streak migration [2, 3, 4, 5, 6, 7, 8]. The complex blade row interactions, non-linear unsteady phenomena, large thermal gradients, and intricate internal air systems make the simulation of turbomachinery flows a monumental challenge.

With advances in Computational Fluid Dynamics (CFD), these topics are being addressed more readily. However, the CFD practices used by engineers during the design process tend to progress at a slower pace. In the interest of saving time, some of the complexities of turbomachinery flows are avoided by using steady simulations with adiabatic wall boundary conditions [9]. In many cases, the fluid-solid heat transfer (and cooling flows) are modeled with separate (often lower fidelity) simulations or empirical relationships [10]. The loss of simulation accuracy can result in overly conservative designs, lower performance, and shorter part life. This deficiency in the design process is not a reflection on the engineers, but rather on the current CFD tools.
1.2 Capturing Multistage Turbomachinery Flow Physics

The early application of CFD to turbomachinery flows involved the use of separate two dimensional methods for solving an axisymmetric passage flow and the flow between the blades. Some review of these techniques (often called S1/S2) is given by Denton and Dawes [11], Horlock and Denton [9], and Dawes [12]. The S1/S2 approach remains an important part of the design process and an active area of research. Attempts to combine axisymmetric and blade passage techniques into a quasi-3D method showed some promise [13]. However, this approach was not sufficiently accurate and it has seen limited use. Ultimately, higher fidelity simulation methods are used when understanding of the 3D flow field is desired.

The movement towards 3D simulations immediately imposes a serious challenge: the simulation of unsteady blade row interactions. The most straightforward approach is to simply solve the unsteady equations by stepping in time on each blade row and communicating the unsteady information across a sliding interface. This method is often called time accurate, though it will be referred to as time-marching (TM) in this work. One costly side effect of the TM method is that a periodic sector of the annulus must be simulated. This requirement often incurs a significant computational expense when simulating several blade rows. Additionally, differing time scales in the main flow path and internal air systems require simultaneously a small time step for high frequency features and a long simulation time for low frequency features. Therefore, a large number of iterations are needed for convergence.

The high cost of the traditional TM approach can be reduced in two ways. Either the unsteady equations and solution procedure are modified or the extent of the computational domain is reduced. Several methods that attempt to achieve these computational savings are explained below. These methods have been grouped to facilitate the discussion. The groups are:

- Methods that march in time on a reduced computational domain
- Methods that solve steady equations
- Methods that solve a reduced form of the unsteady equations
- Methods that solve steady equations in certain blade rows and unsteady equations in others
- Methods that represent the solution as a Fourier series
1.2.1 Time-Marching Methods

The traditional TM methods require a periodic sector. That is, the annulus can only be divided by the greatest common denominator of the blade row periodicities. For instance, consider two blade rows with 35 and 20 blades. One fifth of the annulus would be required (7 and 4 blade passages, respectively). Suppose a third blade row with 29 blades is to be added to the simulation. A computational domain including the entire annulus is now necessary.

1.2.1.1 Domain Scaling

One way to reduce the computational domain needed to perform a time-marching simulation is to use domain scaling. Domain scaling involves modifying the blade geometries and the passage domain to reach blade row periodicities that allow for a smaller periodic sector. Using the above example, modifying blade rows two and three to blade counts 21 and 28 allows for a simulation of one seventh of the annulus. However, this method alters the unsteady features it attempts to capture, thus reducing the accuracy of the solution.

A numerical study of domain scaling for the one and a half stage Aachen axial flow turbine was conducted by Yao et al. [14]. Two domain scaling cases were considered, one with approximately 3\% scaling and the other with approximately 14\% scaling. The study determined that the time averaged and unsteady pressure fields were not significantly affected for the blade scaling of 3\%. However, the effect of scaling on the time averaged relative total pressure loss was found to be significant.

1.2.1.2 Modified Periodic and Inter-Blade Row Boundaries

The Phase Lag approach developed by Erdos et al. [15] allows the computational domain to be condensed into one blade passage per blade row. The solution along the periodic and interface boundaries is stored in time. As the simulation iterates, these boundaries pull the stored solution from the previous blade passing. In an effort to reduce the storage requirements, a method for storing the lagged solution as a Fourier Series was developed by He [16]. Despite the small computational domain, the Phase Lag method is only valid for two blade rows. In addition, the storage can become prohibitive and the lagged boundary conditions can slow the rate of convergence.

As an answer to the Phase Lag periodic assumption and storage requirements, the Time-Inclining method was developed, by Giles et al. [17, 18]. A coordinate transformation from the physical space to the com-
Computational space is applied wherein the computational time is inclined. This inclination is dependent on the neighboring blade row periodicity. The temporal transformation causes the lagged periodic boundary condition to be satisfied automatically. Along the interface, information is passed in the transformed computational domain. However, the amount of inclination must be kept sufficiently small such that the characteristics do not propagate in different directions in the physical and computation spaces [19]. This limits the ratio of blade counts between blade rows. Also, the scheme is only applicable for simulations with two blade rows or with three blade rows if the first and last rows have the same blade counts.

1.2.2 Steady Methods

With these methods, the steady governing equations are solved. The computational domain is reduced to one blade passage per row. Special treatment must be applied at the blade row interfaces to account for the flow variations caused by the motion of the domains relative to each other.

1.2.2.1 Mixing-Plane

The mixing-plane approach was first applied to the Euler equations for fixed domains by Denton and Singh [20]. At the interface between two blade rows, the flow variables are circumferentially averaged. This averaging is effectively an instantaneous mixing of the downstream running wakes and upstream running pressure waves. Mixing-plane interfaces between fixed and rotating passages were implemented by Arts [21] and Ni and Bogoian [22]. The mixing-plane approach was applied to the Navier-Stokes equations by Dawes [23]. Additionally, any blade row could be represented by an axisymmetric through flow simulation, rather than the 3D simulation. A more recent implementation of the mixing-plane by Holmes [24] achieves several improvements over the simple circumferential averaging. Namely, flux conservation, non-reflective properties, user supplied mixing losses and other features were added to improve accuracy and robustness. Obviously, the trade-off of the simplicity of the mixing-plane approach is that it entirely neglects the unsteadiness of the flow field.

1.2.2.2 Averaged Passage

The averaged passage equations, derived by Adamczyk [25], provide a means for introducing some effects of neighboring blade rows while solving steady equations. The Navier-Stokes equations are time averaged
and then averaged across the passage. Additional terms are produced, including body forces, energy sources, and mixing stresses. Closure of the average passage equations for inviscid flows is discussed by Adamczyk et al. [26]. A two step procedure is used. First, the forces, sources and mixing stresses are calculated and stored on an axisymmetric mesh. Then, the average passage equations are solved. An extension to viscous flows is explained by Adamczyk et al. [27] and a method for general blade row meshes is provided by Kirtley et al. [28]. The average passage method accounts for some influence of neighboring blade rows that the mixing-plane procedure ignores. However, true unsteady phenomena, such as radial migration of vortices, are not captured [29].

1.2.3 Reduced Unsteady Methods

1.2.3.1 Quasi-Steady

A quasi-steady set of governing equations can be obtained by linearizing the unsteady equations, as discussed by Hall and Crawley [30]. The solution variables are assumed to be composed of a mean flow and harmonic perturbations with small amplitudes. The unsteady motion of shock waves and wakes are captured. However, non-linear time averaged flow features, such as hot streak migration, are neglected. In addition, the unsteady features present in turbomachinery flows often violate the small perturbation assumption.

1.2.3.2 Deterministic Source Terms

The Lumped Deterministic Stress (LDS) method is a way to incorporate the non-linearities of the unsteady turbomachinery interactions (wakes and pressure waves) without a time-marching multistage simulation. The first application of the LDS method, by Sondak et al. [31], involved obtaining source terms of the deterministic fluid stresses by time averaging unsteady solutions of isolated blade rows. These deterministic source terms are then applied to neighboring blade rows in a steady multistage simulation. Busby et al. [32] performed a similar procedure, using low-fidelity inviscid unsteady simulations to obtain the source terms. A further simplification of the LDS method used the quasi-steady method for calculating the source terms. As shown by Orkwis et al. [7], this LDS method was able to capture hot streak migration (a non-linear unsteady phenomena), while solving the quasi-steady equations.
1.2.4 Mixed Steady-Unsteady Methods

1.2.4.1 Frozen Gust

The frozen gust approach involves freezing the unsteady interaction between blade rows. Steady simulations are completed on isolated blade rows or the multistage machine. The wakes and pressure waves are then applied to an unsteady simulation of a single blade row as unsteady boundary conditions. By simulating the unsteady flow in an isolated blade row, the computational complexity is greatly reduced. However, any unsteady interaction between the blade rows is lost. An example of a quasi-steady frozen gust method and its applicability is discussed by Kapetanovic et al. [33, 34].

1.2.4.2 Coupled Steady-Unsteady

Perhaps a natural extension of the frozen gust method is the coupled steady-unsteady simulation methodology presented by Montomoli et al. [35]. Selected blade rows are simulated as steady, while others are simulated with a TM approach. The steady-steady and unsteady-unsteady blade row interfaces are achieved with traditional mixing-plane and sliding mesh interfaces, respectively. At the interface between steady and unsteady blade rows, time and circumferentially averaged variables are passed to the steady side. Either a mixing plane or a Fourier based sliding mesh approach can be used to transfer information from the steady rows to the unsteady rows. Using this coupled strategy is less computationally expensive than traditional TM methods. Unlike the fully steady approach, unsteady behavior is only modeled in areas of interest. However, the unsteady portions still suffer from the periodic sector requirement and large time scale disparities of the traditional TM approach. Also, the blade rows with meaningful unsteady phenomena are not always known a priori.

1.2.5 Frequency Domain Methods

Frequency Domain (FD) methods employ a Fourier series representation of the solution. FD methods are most suitable (and more heavily developed) for problems that are periodic in time and are dominated by a select number of frequencies. This includes multistage turbomachinery flows, which can be simulated using a single blade passage per blade row. The unsteady equation is reduced to a series of steady-like equations. These equations are simpler to solve than the unsteady equations and allow for acceleration techniques, such as multi-grid. Only a short discussion of FD methods is presented here. For a more detailed discussion, the
reader is directed to a review by He [36].

1.2.5.1 Non-Linear Harmonic Method

The Non-Linear Harmonic (NLH) method was originally applied to aeromechanics problems, such as the study of oscillating blades by He and Ning [37, 38]. The unsteady governing equation is transformed into a series of steady-like equations solving for the mean flow and a number of perturbations. To simplify the equations, the perturbations are assumed to be small and quasi-linear (linear for a given mean flow). The equations are time averaged, resulting in extra terms when compared to the steady form of the equations. The additional terms are non-linearities that arise from products of perturbations. These terms are then added to the time mean equation.

The NLH method has been applied to turbomachinery flows with good results [39, 40]. The perturbation frequencies are determined from the neighboring blade row periodicities and rotation speeds. A phase lag boundary condition is applied at the periodic boundaries. Along the interface, the solution is transferred between the two domains using a temporal and spatial Fourier Transform and a matching of the discrete number of harmonics. The solution representation allows these turbomachinery boundaries to be satisfied with a single passage per blade row. The NLH method is an available feature in the FINE™/Turbo flow solver and has been demonstrated for a variety of turbomachinery test cases [41].

1.2.5.2 Time Spectral and Harmonic Balance Methods

The Time Spectral (TS) and Harmonic Balance (HB) methods solve for a series of time levels rather than Fourier coefficients. A steady-like governing equation is solved for each time level. The series of governing equations is coupled by a pseudo-spectral operator. This operator is derived from the time derivative term of the original unsteady equations using relations between the time levels and Fourier coefficients. Unlike the NLH method, the perturbations are not assumed to be small or quasi-linear. The TS method resolves one fundamental frequency and higher harmonics throughout the entire domain. The HB method resolves any chosen frequencies and allows for each spatial domain to have a different temporal domain.

Since the TS method has a single fundamental frequency, it is used mostly for aeromechanics problems, such as the pitching airfoil studies by Gopinath and Jameson [42]. The TS method can be applied to turbomachinery flows [43]; however, the wide range of blade passing frequencies in each blade row are not well represented by a single fundamental frequency. For this reason, a TS simulation of multistage turbomachin-
ery generally requires more time levels than an HB simulation of the same accuracy. With the HB method, different sets of frequencies are chosen for each blade row. Several studies have shown good comparison with time-marching methods for turbomachinery flows [44, 45, 46, 47]. For the HB method, a phase lag periodic boundary is used. Two different methods have been proposed to resolve the flow along the blade row interface planes. The technique first employed in the Stanford University solver SUmb is similar to the NLH interface condition [44, 45]. Fourier transforms in space and time are performed and the coefficients are matched across the boundary. Research at Duke University produced a method for over-sampling time levels across the interface and filtering the result [46, 48]. Using the phase lag and blade row interface conditions, the HB method requires only one meshed blade passage per blade row.

Often, an explicit coupling of the TS and HB methods is chosen for the ease of implementation. The early TS and HB methods use the explicit implementation [44, 43, 45, 46, 49]. However, the explicit coupling of the pseudo-spectral operator is unconditionally unstable. A detailed discussion of this issue and a stabilization procedure is provided by Custer [50]. In a later work, the implicit formulation of the pseudo-spectral operator was shown, by Thomas et al. [51], to provide an unconditionally stable scheme.

Some recent research has focused on implicit implementations of the HB pseudo-spectral operator. The HB method in Star-CCM+ uses a linearization of the spatial terms and an approximate factorization of the linearized HB terms [52, 47]. However, the updates of the blade row interfaces are performed explicitly by Star-CCM+. Another interesting application of an implicit HB method is shown by Chaurasia [53]. A direct solve was performed on the linearization matrix that included the HB operator, the hybridized Discontinuous Galerkin (DG) spatial scheme, and an implicit relative motion interface. The direct solve gives the HB Newton step in one iteration, though the high memory requirements limit the applicability to two dimensions.

1.2.5.3 Non-Linear Frequency Domain Method

A variation of the TS and HB methods is the Non-Linear Frequency Domain (NLFD) method developed by McMullen and Jameson [54, 55]. Instead of solving on the series of time levels, the governing equations are solved in the frequency domain. These studies solved the shedding cylinder and pitching airfoil problems using a gradient-based search algorithm to determine the shedding frequency. An adjoint implementation of the NLFD method was used to optimize a pitching airfoil, as described by Nadaraja et al. [56]. The gradients of the HB method compared well with those computed from a time-marching adjoint. An implicit
NLFD method with an adaptive algorithm was demonstrated by Mosahebi and Nadarajah [57]. The algorithm varied the number of simulated harmonics throughout the domain to increase the temporal resolution in areas with large unsteady variations. The NLFD method requires the additional computational cost of transforming between the frequency and time domain, whereas the HB and TS methods solve entirely in the time domain.

1.3 Capturing Fluid-Solid Heat Transfer

When performing simulations of the fluid domain, heat transfer at the solid walls is accomplished in one of three ways: an adiabatic boundary condition, a constant wall temperature (iso-thermal) boundary condition, or a coupled heat transfer interface. During the design process, solid wall boundaries are generally considered adiabatic. Investigations by Isomura et al. [58], Bruna and Turner [59] and Knapke and Turner [60] have shown that the adiabatic wall assumption is inaccurate for compressor simulations. Given that large temperature gradients are more prevalent in turbine flows, it is safe to assume that the adiabatic assumption is inaccurate for all turbomachinery flows.

Iso-thermal boundaries provide more physically accurate thermal boundary conditions. Often, the isothermal boundary condition is used to extract heat transfer coefficients that are applied to simulations of the solid domain. In practice, only a few regions of constant temperature are applied to solid wall boundaries, as evidenced by the studies by Heidmann et al. [61], Bruna and Turner [59] and Knapke and Turner [60]. This simplification of the solid wall temperature specification is the result of the uncertainty of the true temperature and the complexity of specifying varying temperatures. Given that thermal gradients are prevalent in turbomachinery flow, the iso-thermal approach is likely to incur significant inaccuracies.

Including analysis of the solid domain can be performed in several ways. The most basic design approach is to model the solid with empirical relations and correlations [10]. The next step in complexity is to use heat transfer coefficients derived from CFD and apply them as boundary conditions to a solid heat transfer solver. A clear weakness of this uncoupled technique is that the solid domain does not influence the fluid simulation.
1.3.1 Coupling Fluid and Solid Simulations

Coupling of the fluid and solid domains is necessary to improve the accuracy of the overall system. The coupling can be performed in two ways. First, two separate solvers for the two domains can be driven to convergence simultaneously. This technique will be referred to as weakly coupled heat transfer. On the other hand, performing both the solid and fluid simulations with the same discretization within one software tool will be considered strongly coupled or conjugate heat transfer (CHT). Using one code with two different discretizations will still be considered weakly coupled because it exhibits the properties of weak coupling (see Section 1.3.1.1).

There is no agreement on these definitions in the literature. The use of the term CHT to describe weak coupling procedures is prevalent. This confusion is likely the result of the word “conjugate” being synonymous with “coupled”. The terms used here relate to the coupling properties and they follow the seminal paper by Giles [62]. Here, “strongly coupled” and “conjugate” will be used interchangeably. Henceforth, use of the term conjugate heat transfer or the acronym CHT will refer only to the strongly coupled heat transfer method. This choice is made by considering the mathematical definition of “conjugate”, in which the word denotes two objects with similar properties. In the case of Conjugate Heat transfer, the “similar properties” of the “two objects” (fluid and solid domains) are their discretizations.

1.3.1.1 Weakly Coupled Heat Transfer

Because two discretizations are used, the solution at each iteration is extracted from one side of the fluid-solid interface and applied as a boundary condition to the other side. This process generally involves some form of interpolation from one domain to the other. Coupling in this way leads to stability constraints, as explained by Giles [62]. In addition, care must be taken by the developer so that information is appropriately exchanged between the two discretizations, as discussed by Dannenhoffer and Haimes [63].

Early application of the weakly coupled approach was performed by Heselhaus et al. [64]. A Finite Element Analysis (FEA) code was used to update the CFD boundary condition after a user-specified number of time steps. Other early methods by Li and Kassab [65, 66] and Rahaim and Cavalleri [67] perform the solid conduction modeling with the Boundary Element Method (BEM).

Of the two parts of the fluid-solid interaction, the CFD solver is decidedly more complex. With the rise in popularity of robust commercial CFD software, weakly coupling a solid conduction solver to an existing
commercial CFD solver has become an attractive option. Illingworth et al. [68] describe the coupling of the CFD solver Fluent and a solid conduction code. However, interacting with commercial software is likely to be more complex and less efficient than traditional weak coupling because the CFD solver cannot be modified to accommodate the solid conduction solver.

In the interest of solving the fluid-solid heat transfer problem on complex geometries, several meshing techniques have been employed. Early works favored structured multi-block topologies, most likely for the simplicity of implementation. A hybrid unstructured approach, practiced by Han et al [69], performs well for an internally cooled stator. An overset method for the fluid and solid domain coupling was developed by Henshaw and Chand [70]. Weakly coupled heat transfer was performed on implicitly defined geometries by Gadeschi et al. [71]. Rather than explicitly defining boundaries, a distance field is stored where negative values are the solid and positive values are the fluid. A level set method is then used to implicitly define the boundary at the zero distance iso-surface.

Recent research of a steady 3D turbine stator, by Chaquet et al. [72], has compared an uncoupled and a weakly coupled approach. The analysis showed that the weakly coupled heat transfer method achieved half of the peak metal temperature error when compared to the uncoupled approach.

1.3.1.2 Conjugate Heat Transfer

Unlike the weakly coupled approach, the CHT approach is unconditionally stable [62]. One of the earliest formulations of a CHT method was performed by Bohn et al. [73]. In the following years, many CHT studies were conducted by Bohn and others at Aachen University. An investigation of cooling flow showed the improved accuracy of CHT over an adiabatic boundary condition [74]. A stator with internal cooling, tested by Hylton et al [75], was simulated and the effect of thermal barrier coating on the heat transfer was determined [76].

The Hylton stator was simulated with another early application of the CHT method, by Kao and Liou [77]. Unstructured meshes were used in the solid, while Chimera structured meshes were used in the fluid. The Hylton stator was also successfully simulated with the commercial software Fluent, as shown by York and Leylek [78]. Another internally cooled stator was analyzed with Fluent by Takahashi et al. [79]. These cooled stator simulations provide a basis of proof that the CHT method improves the understanding of heat transfer.

In recent years, the CHT method has been applied to increasingly complex turbomachinery flows. A
steady CHT analysis of a turbocharger by Bohn et al. [80] compared well with experimental results. In this
case, the heat transfer from the hot turbine to the cold compressor significantly affects the performance of
the compressor. Simulation of an internally cooled stator, including film cooling and serpentine passages,
was used by Kusterer et al. [81] to improve a cooling configuration. A similarly complex CHT analysis of a
stator was completed by Ni et al. [82]. Excellent agreement was shown between the experimental data and
the CHT solution [83]. Monomoli et al. [84] conducted an uncertainty analysis on a CHT simulation of a
film cooled turbine stator. The positioning of the combustor core flow entering the stator was shown to be
the most significant influence on the metal temperature uncertainty.

Some interesting approaches to the solution of the fully coupled heat transfer problem have been devised.
Rather than solving different systems of equations in the fluid and solid domains, Rigby and Lepicovsky [85]
and Nordstrom and Berg [86] have solved the Navier-Stokes equations in both domains. In the solid, the
velocities are set to zero. Sun and Darmofal. [87, 88] have used a Discontinuous Galerkin (DG) discretiza-
tion and an adaptive cut-cell meshing approach. With cut-cells and adaptive mesh refinement, the definition
of the fluid-solid interface is separated from the definition of the mesh. Another CHT method using the DG
discretization, by Hao et al. [89], was used to investigate different turbulence models.

1.3.1.3 Unsteady Heat Transfer

An early unsteady CHT simulation was performed by Amon [90]. A spectral element Fourier method
discretization was used to study heat exchangers. The meshed domain consisted of two dimensions while
the third dimension was captured with a truncated Fourier expansion. Two unsteady turbomachinery heat
transfer analyses were completed by Sondak and Dorney [91, 92]. A weakly coupled approach was used
along with a TM method and domain scaling to reduce the size of the computational domain. A Chimera
overset meshing technique was used to handle geometric complexities. These studies by Sondak et al.
showed the ability of coupled methods to capture unsteady heat transfer, including the effects of hot streak
migration. Another unsteady turbomachinery simulation was performed by Rehman [93]. Fluent was used
to simulate a radial turbine and the results were compared to experiment.

Turbomachinery flows present a difficulty when simulating unsteady heat transfer. As discussed by He
and Oldfield [94], the disparity in the time scales between the fluid and solid domains becomes large. For
TM methods, a small time step is required to resolve the unsteady fluid. Simultaneously, long time periods
are necessary to allow for the transients to move through the solid domain. This leads to large numbers of
time steps to achieve convergence and temporal accuracy. Another difficulty is that the unsteady temperature variations only penetrate a small distance into the solid domain. He and Oldfield propose a solution to these challenges. They solve a hybrid system with the TM method in the fluid domain and a steady method in the solid domain. At the interface a semi-analytical harmonic condition is applied. The steady solid domain reduces cost while the solid side of the semi-analytical interface attempts to capture the small penetration distance of the unsteady temperature variations.

Besides the approach by He and Oldfield, CHT methods are largely relegated to steady turbomachinery flows. This absence is a result of the complexities mentioned here and in Section 1.2, which result in high computational costs. In the design process, unsteady multistage turbomachinery CHT is non-existent.

1.4 Dissertation Objectives

The objective of this dissertation is to develop a methodology for efficient simulation of complex multistage turbomachinery flows while accurately capturing the fluid-solid heat transfer. To that end, the Harmonic Balance and Conjugate Heat Transfer methods are employed to capture unsteady phenomena and fluid-solid heat transfer, respectively. Improvements in computational stability and performance are achieved with the implicit implementation of both methods. The ultimate purpose of combining these techniques is to provide a tool that is fast enough to be used during the design process and accurate enough to influence design decisions that improve performance.

The HB method was chosen over the other multistage turbomachinery simulation techniques for several reasons. First, only a single blade passage is modeled for every blade row. Unlike the steady and quasi-steady approaches, the non-linear unsteady phenomena are captured with the HB method. Another benefit of the HB method is that a series of steady-like equations are solved, rather than the unsteady equations. Thus, different time scales are captured within a steady-like simulation. Traditionally, large numbers of small time steps are needed (at great computational cost) to resolve the different scales.

All of the FD methods capture non-linear unsteadiness with different time scales and reduced computational domains for multi-stage turbomachinery. However, the HB approach maintains some additional advantages. By solving on a series of time levels, the HB method is relatively easy to implement within an existing solver when compared to the NLH and NLFD methods. All boundary and interface conditions are the same for each time level, while the NLH and NLFD methods must treat the mean flow and perturba-
tions separately. Also, the HB method does not make the small perturbation and quasi-linear assumptions of the NLH method. The final factor for choosing the HB method rather than the TS method is that the HB method allows for multiple time domains and the choice of any frequency. With these added features, the HB method is better suited for multistage turbomachinery simulations.

When choosing a method for capturing the fluid-solid heat transfer, the adiabatic and iso-thermal wall boundary conditions were immediately discarded. The adiabatic wall is non-physical and inaccurate. The iso-thermal wall is only accurate if the correct temperatures are chosen. However, the complex nature of modern turbomachinery makes choosing correct temperatures impossible. It is clear that simulation of the solid domain must be included in some way. Four paths are possible for modeling the solid domain; empirical models, uncoupled simulations, weakly coupled simulations, or CHT (strongly coupled, see Section 1.3.1) simulations. As a consequence of increasingly complex blade geometries and the inclusion of internal air systems, accurate empirical models are becoming unmanageable (if not already so). Uncoupled simulations may provide some insight, but they ignore (or lag) any interaction from the solid to the fluid. The options are now reduced to weakly coupled heat transfer and CHT. Ultimately, CHT was chosen for two reasons. First, the strongly coupled CHT method has no stability requirements. Second, maintaining one discretization is believed to require less effort and it eliminates the complexity of ensuring the appropriate communication of information across the fluid-solid interface.

1.4.1 The DG-Chimera Framework

The developments outlined in this dissertation are built upon the partial differential equation framework created by Marshall Galbraith [95, 96, 97, 98, 99, 100, 101]. This framework utilizes a Discontinuous Galerkin discretization and a Chimera overset scheme. This first of its kind DG-Chimera framework provides several benefits. The general structure of the code allows the developer to include (within a short time frame) any kind of partial differential equation in conservative form. Currently, the list includes Euler, Navier-Stokes, Reynold’s Averaged Navier-Stokes, linear advection, solid conduction, linear advection-diffusion, non-linear advection-diffusion and Burger’s equations. This generality reduces code maintenance and allows for modifications at higher levels of the code to affect all equation sets simultaneously. With the addition of a general HB method, all existing and future partial differential equations are immediately HB ready. Another benefit of the DG-Chimera method is the high order accuracy. This feature is particularly beneficial in the near wall regions of the fluid and solid domains. The implemented Chimera scheme pro-
vides a means for simulating complex geometries [100]. In addition, the DG-Chimera scheme can handle curved cells and abutting meshes with non-coincident points, which allows considerable meshing flexibility along fluid-solid interfaces. The final advantage of the DG-Chimera framework is that it is fully implicit, including the artificial boundaries [98]. The implicit formulation provides improved speed and stability for the HB and CHT methods.

Several contributions to the DG-Chimera framework have been achieved for this dissertation. First, a general implicit HB implementation was added such that it is independent of the spatial flux and governing equations. Second, the Chimera scheme was generalized in two ways. The communication between different equation sets is allowed (i.e. the fluid to solid communication needed for CHT) and HB operations are made possible along artificial boundaries. From these operations, phase lag and relative motion interfaces are developed such that efficient multistage turbomachinery simulations are possible. The implicit property of the artificial boundaries is maintained along HB and CHT interfaces. The third contribution is the addition of a turbulence model to allow for the simulation of high Reynold’s number turbomachinery flows.

1.4.2 Outline

The spatial discretization, HB method, and CHT method are discussed in Chapter 2. In the spatial discretization section, a short explanation of the DG discretization is given along with the fluid and solid governing equations. A full description of the HB method is provided along with details of the phase lag and relative motion interfaces. Chapter 3 describes the implementation of the HB and CHT methods. Verification of these methods and the spatial scheme is shown in Chapter 4 through a series of test cases. Two heat transfer simulations are completed and presented Chapter 5. These simulations are a 3D curved duct and a 3D unsteady flow in a channel. A discussion of the computational cost of the HB and CHT methods is presented in Chapter 6. Conclusions and future work are given in Chapter 7.
Chapter 2

Methodology

This chapter outlines the Discontinuous Galerkin (DG) spatial discretization, the Harmonic Balance (HB) method, and the Conjugate Heat Transfer (CHT) method. Governing equations for solid conduction and turbulent fluid flow are defined. An implicit formulation of the HB method, including the linearization of the pseudo-spectral operator, is demonstrated. Turbomachinery flows are considered, with an explanation of the HB phase lag and relative motion interfaces. Details of the CHT fluid to solid interface are also presented.

2.1 Spatial Discretization

Consider the domains $A$, $B$, and $C$ depicted in Fig. 2.1. The domain volumes are $\Omega^A$, $\Omega^B$, and $\Omega^C$. These domains have external boundaries ($\Gamma^A_{\text{b}}$, $\Gamma^B_{\text{b}}$, and $\Gamma^C_{\text{b}}$) and joining boundaries ($\Gamma_{A\cap B}$ and $\Gamma_{B\cap C}$). Suppose that domains $A$ and $B$ are governed by the same physical properties, while the domain $C$ properties differ from those in $A$ and $B$. The interaction along the boundary between $A$ and $B$ ($\Gamma_{A\cap B}$) is meant to resemble the physical processes within the volume. Therefore, $\Gamma_{A\cap B}$ is considered an artificial boundary (denoted as $\Gamma_{\text{a}}$). This type of boundary allows for multi-block meshes, parallelization, and the Chimera overset scheme. The boundary between the dissimilar domains ($\Gamma_{B\cap C}$) cannot resemble the interior volume because the governing physics of $B$ and $C$ are different. Additional procedures are required along $\Gamma_{B\cap C}$ to provide an appropriate interaction between $B$ and $C$. Boundaries of this type will be called interfaces and identified with the symbol $\Sigma$. This chapter describes two situations where interfaces are applied. First, the CHT method requires an interface between the solid and fluid domains to couple the governing equations (Section 2.4). Second, HB
simulations of turbomachinery flows are accomplished with interfaces that communicate between different
temporal domains and time levels (Section 2.3.2).

2.1.1 Discontinuous Galerkin Method

The effort in this dissertation builds upon the work of Galbraith [101]. For a full discussion of the DG
method and the particular implementation used, the reader is directed to his work. Descriptions of the DG
method here will be brief and are only meant to emphasize areas to be expanded upon in later sections.

The DG method implemented by Galbraith [101] solves 1st and 2nd order partial differential equations
of the following form

$$\frac{\partial Q(W)}{\partial t} + \nabla \cdot \vec{F}(W, \nabla W) + S(W, \nabla W) = 0$$

(2.1)

where $W$ is the dependent variable vector, $Q(W)$ is the conservative variable vector, $\nabla \cdot \vec{F}(W, \nabla W)$ is the
flux, and $S(W, \nabla W)$ is the source term. The form of Eq. (2.1) is more general than the one used in the
derivation by Galbraith [101], in which $W = Q$. The more general form is desired to allow for a discussion
the fluid flow equations using primitive variables (see Section 2.2.2).

The dependent variable vector is represented by polynomials within non-overlapping cells. The compu-
tational domain is discretized into quadrilaterals in two-dimensions and hexahedron in three dimensions.
A Chimera overset method allows the modeling of complex geometries. Figure 2.2 shows an example of
the DG discretization applied to the situation presented in Fig. 2.1. The volume of cell $e$ in the $i^{th}$ domain
is denoted as $\Omega^i_e$. An interior cell boundary is $\Gamma^i_e$, while the other boundaries ($\Gamma^b_e$, $\Gamma_a$, and $\Sigma$) are those
described in the preceding section. The internal and external cell dependent variable vectors are $W^-$ and
$W^+$, respectively. The dependent variable vector along $\Gamma_b$ is $W_B$, an idea that is expanded upon in Eq. (2.6). The outward facing normal vectors are $\vec{n}$.

The dependent variable in each cell is defined by a polynomial expansion. The polynomials are expressed in general as

$$W(\vec{x}) = \sum_{i=1}^{N_m} W_i \hat{\psi}_i(\vec{x})$$  \hspace{1cm} (2.2)

where $N_m$ is the number of modes, $\hat{\psi}_{i(j,k,l)}(\vec{x}) = \hat{\psi}_j(x) \hat{\psi}_k(y) \hat{\psi}_l(z)$ are the polynomial basis functions and $W_i$ are the modes (polynomial coefficients) of the dependent variable. The number of modes is $N_m = (q + 1)^d$ where $q$ is the order of the polynomial and $d$ is the number of dimensions. The modal representation is used to facilitate the analytical integration. The Legendre polynomials were chosen as the polynomial basis functions. The orthogonality of the Legendre polynomials reduces the computational cost of the analytical integration.

The weak form of the governing equations is obtained by multiplying Eq. (2.1) by a vector of test functions $\left( \hat{\psi} = [\hat{\psi}_i(\vec{x})]^T, \forall \in [0,N_m] \right)$, integrating over the cell volumes, and applying Gauss’s theorem.

$$\int_{\Omega_e} \hat{\psi} \frac{\partial Q(W)}{\partial t} d\Omega + \int_{\Gamma_e} \hat{\psi} \vec{F} (W, \nabla W) \cdot \vec{n} d\Gamma - \int_{\Omega_e} \nabla \hat{\psi} \cdot \vec{F} (W, \nabla W) \cdot \vec{n} d\Omega + \int_{\Omega_e} \hat{\psi} S(W, \nabla W) d\Omega = 0$$ \hspace{1cm} (2.3)

For this DG discretization, the advective and diffusive fluxes are treated differently, thus the flux is split

$$\vec{F} (W, \nabla W) = \vec{F}^a (W) + \vec{F}^d (W, \nabla W)$$ \hspace{1cm} (2.4)
where $\bar{F}^a(W)$ is the advective flux and $\bar{F}^d(W, \nabla W)$ is the diffusive flux.

Along the cell boundary, the advective flux integral is approximated with the average of the interior and exterior boundary fluxes and an upwinding flux, $\phi$. Several techniques are available for approximating the diffusive fluxes. The Bassy-Rebay (BR2) scheme was chosen for this implementation because it maintains a small stencil [102]. On the cell boundaries, the average of the diffusive fluxes is computed with the lifted gradients $\nabla W^- + \eta_e \bar{r}^-$ and $\nabla W^+ + \eta_e \bar{r}^+$. The BR2 lifting operator for the dependent variables is defined as

$$\int_{\Omega_e} \hat{\psi} \hat{r}_e d\Omega = \int_{\Gamma_e} \hat{\psi} \frac{1}{2} (W^+ - W^-) \vec{n} d\Gamma$$

(2.5)

where $\eta_e$ is a stabilization coefficient taken to be equal to the number of faces. The volume diffusive flux is calculated with the lifted gradient $\nabla W^- + \bar{R}$ where $\bar{R}$ is the sum of all of $\bar{r}_e$ for a given cell.

Flux integrals along the external domain boundaries, $\Gamma_b$ are approximated with the fluxes calculated from the boundary dependent vector, as

$$\int_{\Gamma_b} \hat{\psi} \hat{F} (W_B, \nabla W_B) \cdot \vec{n} d\Gamma \Rightarrow \int_{\Gamma_b} \hat{\psi} \hat{F}^a (W_B (W^-, W_b)) \cdot \vec{n} d\Gamma$$

$$+ \int_{\Gamma_b} \hat{\psi} \hat{F}^d (W_B (W^-, W_b), \nabla W_B (\nabla W^-, \nabla W_b) + \eta_e \bar{r}_b) \vec{n} d\Gamma$$

(2.6)

where $\bar{r}_b$ is the boundary BR2 lifting operator, defined as

$$\int_{\Omega_e} \hat{\psi} \hat{r}_b d\Omega = \begin{cases} \int_{\Gamma_b} \hat{\psi} \frac{1}{2} (W_B (W^-, W_b) - W^-) \cdot \vec{n} d\Gamma & \text{Dirichlet Condition} \\ 0 & \text{Neumann Condition} \end{cases}$$

(2.7)

and $W_b$ are the imposed boundary variables, and $\nabla W_b$ are the imposed boundary condition variable gradients. The imposed conditions and the calculation of $W_b$ and $\nabla W_b$ depend on the type of boundary condition. Details of these calculations are given in Section 2.2. Approximations of the flux integrals along the artificial boundaries and interfaces are achieved in the same way as interior cell boundaries. The difference for interface integrals being that the variables and gradients are determined from the external cell variables and the conditions of the interface, as discussed in Sections 2.3.2.2, 3.4.4, and 3.5.1.
All of the integrals are combined into the final DG equation, which is shown below

\[
\int_{\Omega_e} \hat{\psi} \frac{\partial Q(W)}{\partial t} d\Omega + \mathcal{R}(W) = \\
\int_{\Omega_e} \frac{\partial Q(W)}{\partial t} d\Omega + \\
\int_{\Gamma} \frac{1}{2} \left[ \left( \tilde{F}^a(W^+) + \tilde{F}^a(W^-) \right) \cdot \vec{n} - \phi \left( W^+, W^- \right) \cdot \vec{n} \right] d\Gamma - \int_{\Omega_e} \nabla \hat{\psi} \cdot \tilde{F}^a(W^-) d\Omega + \\
\int_{\Gamma} \frac{1}{2} \left[ \left( \tilde{F}^d(W^+, \nabla W^+ + \eta e \tilde{\rho}^+) + \tilde{F}^d(W^-, \nabla W^- + \eta e \tilde{\rho}^-) \right) \cdot \vec{n} \right] d\Gamma - \\
\int_{\Gamma} \nabla \hat{\psi} \cdot \tilde{F}^d(W^-, \nabla W^- + \tilde{\rho}) d\Omega + \\
\int_{\Gamma} \hat{\psi} \tilde{F}^d(W_B(W^-, W_b), \nabla W_B(\nabla W^-, \nabla W_b) + \eta e \tilde{\rho}_b) d\Omega + \\
\int_{\Gamma} \frac{1}{2} \left[ \left( \tilde{F}^a(W^+) + \tilde{F}^a(W^-) \right) \cdot \vec{n} - \phi \left( W^+, W^- \right) \cdot \vec{n} \right] d\Gamma + \\
\int_{\Gamma_a} \frac{1}{2} \left[ \left( \tilde{F}^d(W^+, \nabla W^+ + \eta e \tilde{\rho}^+) + \tilde{F}^d(W^-, \nabla W^- + \eta e \tilde{\rho}^-) \right) \cdot \vec{n} \right] d\Gamma + \\
\int_{\Sigma} \frac{1}{2} \left[ \left( \tilde{F}^a(W^+_\Sigma) + \tilde{F}^a(W^-_\Sigma) \right) \cdot \vec{n} - \phi \left( W^+_\Sigma, W^-_\Sigma \right) \cdot \vec{n} \right] d\Gamma + \\
\int_{\Sigma} \frac{1}{2} \left[ \left( \tilde{F}^d(W^+_\Sigma, \nabla W^+_\Sigma + \eta e \tilde{\rho}^+\Sigma) + \tilde{F}^d(W^-_\Sigma, \nabla W^-_\Sigma + \eta e \tilde{\rho}^-\Sigma) \right) \cdot \vec{n} \right] d\Gamma + \\
\int_{\Omega_e} \hat{\psi} S(W^-, \nabla W^- + \tilde{\rho}) d\Omega = 0 \quad (2.8)
\]

where \(\mathcal{R}(W)\) is the sum of the spatial integrals. This equation is valid for all of the governing equations used for this work. The choice of fluxes, upwind dissipation and source terms determines which equations are used, as described in Section 2.2. The interface variable and interface variable gradients are denoted as \(W^+_\Sigma, \nabla W^+_\Sigma, W^-_\Sigma, \text{ and } \nabla W^-_\Sigma\). The interface BR2 lifting operators, \(\tilde{\rho}^+_\Sigma\) and \(\tilde{\rho}^-_\Sigma\), are calculated with \(W^+_\Sigma\) and \(W^-_\Sigma\) in place of \(W^+\) and \(W^-\), as

\[
\int_{\Omega_e} \hat{\psi}(\tilde{\rho})_\Sigma = \int_{\Sigma} \hat{\psi} \frac{1}{2} (W^+_\Sigma - W^-_\Sigma) \quad (2.9)
\]

The calculation of these interface quantities depends on the type of interface. Sections 2.3.2.2, 2.3.2.3 and 2.4 describe the three interfaces developed for this work. The integrals over \(\Gamma_b, \Gamma_a, \text{ or } \Sigma\) are zero if the cell faces of cell \(e\) do not intersect with these boundaries.
2.1.1.1  Newton’s Method

An efficient numerical method for obtaining a solution of a system of non-linear equations is Newton’s method [103, 104, 105]. Consider the steady form \( \frac{\partial}{\partial t} = 0 \) of Eq. (2.8), which is simply

\[
\mathcal{R}(W) = 0 \tag{2.10}
\]

Applying a generalization of Newton’s formula to the above equation gives

\[
\frac{\partial \mathcal{R}(W^m)}{\partial W} \Delta W^m = -\mathcal{R}(W^m) \tag{2.11}
\]

where \( m \) is the current iteration and \( W^m \) is a vector of dependent variables for all cells in the computational domain. The update vector, \( \Delta W^m \), is calculated at each iteration by solving the system of linear equations. Afterward, the new solution is calculated as \( W^{m+1} = W^m + \Delta W^m \). A complete linearization is used to form the matrix, \( \frac{\partial \mathcal{R}(W^m)}{\partial W} \). For structured meshes, the matrix is a block tri-diagonal, block peta-diagonal, or block hepta-diagonal for one, two and three dimensions. The linearization of \( \mathcal{R}(W^m) \) for a given cell with respect to its own interior variables, \( W^- \), forms the main diagonal of \( \frac{\partial \mathcal{R}(W^m)}{\partial W} \), while the linearization with respect to the variables of the neighboring cells, \( W^+ \), forms the off diagonals of \( \frac{\partial \mathcal{R}(W^m)}{\partial W} \). Linearizations of the time term and the \( \Sigma \) flux terms from Eq. (2.8) are shown below:

- **Time Integral** (for time-marching methods and Quasi-Newton method in Section 3.3)

\[
\frac{\partial Q}{\partial W} \frac{\partial W}{\partial t} \Delta W^- \tag{2.12}
\]

- **Advection Interface Integral**

\[
\frac{1}{2} \left[ \left( \frac{\partial \tilde{F}_a(W^+_\Sigma)}{\partial W^+_\Sigma} \Delta W^+_\Sigma + \frac{\partial \tilde{F}_a(W^-_\Sigma)}{\partial W^-_\Sigma} \Delta W^-_\Sigma \right) \cdot \vec{n} - \left( \frac{\partial \phi(W^+_\Sigma, W^-_\Sigma)}{\partial W^+_\Sigma} \Delta U^+_\Sigma + \frac{\partial \phi(W^+_\Sigma, W^-_\Sigma)}{\partial W^-_\Sigma} \Delta W^-_\Sigma \right) \right] \tag{2.13}
\]
• Diffusion Interface Integral

\[
\frac{\partial \tilde{F}^d (W^+_{\Sigma}, \nabla W^+_{\Sigma} + \eta e \vec{r}^+_{\Sigma})}{\partial W^+_{\Sigma}} \Delta W^+_{\Sigma} + \frac{\partial \tilde{F}^d (W^-_{\Sigma}, \nabla W^-_{\Sigma} + \eta e \vec{r}^-_{\Sigma})}{\partial W^-_{\Sigma}} \Delta W^-_{\Sigma} + \frac{\partial \tilde{F}^d (W^+_{\Sigma}, \nabla W^+_{\Sigma} + \eta e \vec{r}^+_{\Sigma})}{\partial W^+_{\Sigma}} \Delta W^+_{\Sigma} + \frac{\partial \tilde{F}^d (W^-_{\Sigma}, \nabla W^-_{\Sigma} + \eta e \vec{r}^-_{\Sigma})}{\partial W^-_{\Sigma}} \Delta W^-_{\Sigma} \tag{2.14}
\]

Notice that the interface update vectors are

\[
\Delta W^\pm_{\Sigma} = \frac{\partial W^\pm_{\Sigma}}{\partial W^\pm} \Delta W^\pm
\tag{2.15}
\]

where the Jacobian of the interface variables with respect to the dependent variables, \( \frac{\partial W^\pm_{\Sigma}}{\partial W^\pm} \), is applied during matrix multiplications rather than storing directly. This choice was made to reuse the structure of the artificial boundaries, thus simplifying the interface implementation.

In addition to the linearized terms presented here, those given by Galbraith [101] are used to form a complete set of tools to construct \( \frac{\partial \Phi(W^m)}{\partial W^m} \). The linearization of the artificial boundary flux terms gives the same result as Galbraith’s boundary integral definitions. Besides the time integral linearization, Galbraith’s linearized terms can be made consistent with the notation used here by substituting the dependent variable vector, \( W \), for the conservative variable vector, \( Q \).

### 2.2 Governing Equations

To demonstrate the capabilities of the DG-Chimera framework, Galbraith implemented several types of governing equations [101]. These included the conservative variables form of the laminar Navier-Stokes equations and Poisson’s equation. Presented here are variations of these two equations. The first is the solid conduction equation, which is a slight modification of the Poisson equation. Second, the compressible Navier-Stokes equations using a set of primitive variables are presented. An extension to turbulent flows has been achieved with the addition of the Spalart-Allmaras turbulence model. The implications of these two governing equations, as related to CHT, are explored in Section 2.4.
2.2.1 Solid Conduction Equation

Conduction is the change in internal energy caused by the movement and interaction of microscopic particles. Begin with the 1\textsuperscript{st} law of thermodynamics

\[ dU = \delta Q + \delta W \]  (2.16)

where \( U \) is the internal energy, \( Q \) is the heat added to the system, and \( W \) is the addition of work to the system. The rate of change in time of Eq. (2.16) for a solid with a constant volume and no work addition is

\[ \rho_s C_{p,s} \frac{\partial T}{\partial t} + \nabla \cdot \vec{q} = 0 \]  (2.17)

where \( T \) is the temperature, \( \vec{q} \) is the heat flux, \( \rho_s \) is the density, and \( C_{p,s} \) is the specific heat at constant pressure. Applying Fourier’s law

\[ \vec{q} = -k \nabla T \]  (2.18)

to Eq. (2.17), gives

\[ \rho_s C_{p,s} \frac{\partial T}{\partial t} + \nabla \cdot (k \nabla T) = 0 \]  (2.19)

where the thermal conductivity is \( k = k(T) \). Dividing by \( \rho_s C_{p,s} \) results in

\[ \frac{\partial T}{\partial t} + \nabla \cdot (\alpha \nabla T) = 0 \]  (2.20)

where \( \alpha = \frac{k}{\rho_s C_{p,s}} \) is the thermal diffusivity. Writing Eq. (2.20) in the form of Eq. (2.1)

\[ \frac{\partial T}{\partial t} + \nabla \cdot \vec{F}^d (T, \nabla T) + S(T, \nabla T) = 0 \]  (2.21)

For this equation, the advective flux is zero. Noting that \( \alpha = \alpha(T) \), the diffusive flux is

\[ \vec{F}^d (T, \nabla T) = \alpha (T) \nabla T \]  (2.22)
and the diffusive flux Jacobians are

\[
\frac{\partial \vec{F}_d(T, \nabla T)}{\partial T} = - \frac{\partial \alpha(T)}{\partial T} \nabla T \\
\frac{\partial \vec{F}_d(T, \nabla T)}{\partial \nabla T} = - \alpha(T) I
\]  

(2.23)

2.2.1.1 Boundary Conditions

Dirichlet and Neumann boundary conditions are available for the solid conduction equations. Besides the differing notation, these boundary conditions are the same as those defined by Galbraith [101].

**Dirichlet**

For the Dirichlet boundary condition, the temperature is specified and the temperature gradient is extrapolated

\[
T_B(T^-, T_b) = T_b \\
\nabla T_B(\nabla T^-, \nabla T_b) = \nabla T^-
\]  

(2.24)

and the corresponding Jacobians are

\[
\frac{\partial T_B(T^-, T_b)}{\partial T^-} = 0 \\
\frac{\partial \nabla T_B(\nabla T^-, \nabla T_b)}{\partial \nabla T^-} = I
\]  

(2.25)

**Neumann**

Conversely, for the Neumann condition, the temperature is extrapolated and the temperature gradient is specified

\[
T_B(T^-, T_b) = T^- \\
\nabla T_B(\nabla T^-, \nabla T_b) = \nabla T_b
\]  

(2.26)
and the temperature and temperature gradient Jacobians are

\[
\frac{\partial T_B(T^-, T_b)}{\partial T^-} = 1
\]
\[
\frac{\partial \nabla T_B(\nabla T^-, \nabla T_b)}{\partial \nabla T^-} = 0
\]

(2.27)

### 2.2.2 Fluid Flow Equations

A set of equations are used to represent compressible, viscous, turbulent fluid flow. Conservation of mass is enforced with the continuity equation. Newton’s second law, relating momentum change to force, is used to form a momentum equation for each Cartesian direction. The 1st law of thermodynamics is used to derive an equation that enforces energy/work conservation. The fluid is assumed to exhibit a time mean component and a fluctuating component. Time averaging the five equations results in the Reynold’s Averaged Navier-Stokes (RANS) equations. Closure of the RANS equations is provided by the Spalart-Allmaras (S-A) turbulence model [106, 107].

A set of primitive variables is used in dimensional form. The ideal gas law equation is used to relate pressure, temperature, and density. The gas is assumed to be ideal and calorically perfect. A formulation of the Navier-Stokes equations for the non-dimensional conservative variables is given by Galbraith [101]. For the fluid flow equations, the dimensional primitive variables formulation is used for all simulations, unless otherwise specified.

A subset of the RANS equations is the laminar Navier-Stokes equations. These equations are obtained by removing the turbulence model equation and assuming a zero eddy viscosity. Assuming a laminar and inviscid flow (all viscosity is zero) produces the Euler equations. The laminar Navier-Stokes and Euler equations are used for some verification test cases.

#### 2.2.2.1 Reynold’s Averaged Navier-Stokes Equations

The Jacobians for the RANS equations and boundary conditions are derived in Appendix A. The governing equation in conservative form, is

\[
\frac{\partial Q(U)}{\partial t} + \nabla \cdot \vec{F}^a(U) + \nabla \cdot \vec{F}^d(U, \nabla U) + S(U, \nabla U) = 0
\]

(2.28)
where \( Q = \begin{bmatrix} \rho, \rho u, \rho v, \rho w, \rho E, \rho \tilde{v} \end{bmatrix} \) is the conservative variable vector and the dependent variable vector, \( W \), is replaced by the primitive variable vector \( U = \begin{bmatrix} u, v, w, T, p, \tilde{V} \end{bmatrix} \). The total internal energy is \( E = e(T, p) + \frac{1}{2} \tilde{V} \cdot \tilde{V} \) where \( e(T, p) \) is the internal energy and \( \tilde{V} = \begin{bmatrix} u, v, w \end{bmatrix} \) is the velocity vector. The Spalart-Allmaras working variable is \( \tilde{v} \). The advective flux is a tensor, shown in the equation below.

\[
\tilde{F}^a(U) = \begin{bmatrix} F_x^a & F_y^a & F_z^a \end{bmatrix} = \begin{bmatrix}
\rho u & \rho v & \rho w \\
\rho u^2 + p & \rho vu & \rho vw \\
\rho uv & \rho v^2 + p & \rho wv \\
\rho uw & \rho vw & \rho w^2 + p \\
\rho uH & \rho vH & \rho wH \\
\rho u\tilde{v} & \rho v\tilde{v} & \rho w\tilde{v}
\end{bmatrix}
\]

(2.29)

where the total enthalpy is \( H = h(T, p) + \frac{1}{2} \tilde{V} \cdot \tilde{V} \) and \( h(T, p) \) is the static enthalpy. Generally, the governing equations are non-dimensionlized, as is the case with the previously implemented conservative variable fluid flow equations. However, the choice was made to use a dimensional form of the equations with the dependent variables. This choice simplifies any interface between two sets of governing equations (i.e. CHT), where any difference in non-dimensional terms would require extra reference conditions. Also, the future implementation of the real gas fluid flow equations will not need additional reference quantities.

For all simulations in this effort, the fluid is assumed to be an ideal and calorically perfect gas. These assumptions give rise to the equation of state

\[
p = \rho R_{\text{gas}} T
\]

(2.30)

and the definitions of internal energy and enthalpy

\[
e = C_v T \\
h = C_p T
\]

(2.31)

where \( R_{\text{gas}} \) is the gas constant, \( C_v \) is the specific heat at constant volume, and \( C_p \) is the specific heat at
constant pressure; all of which are constant. The specific heats are defined as

\[ C_v = \frac{R_{\text{gas}}}{\gamma - 1} \]
\[ C_p = \frac{\gamma R_{\text{gas}}}{\gamma - 1} \]

where \( \gamma \) is the ratio of specific heats. The diffusive flux tensor is

\[ \vec{F}^d(U, \nabla U) = \begin{bmatrix} F_x^d & F_y^d & F_z^d \end{bmatrix} = \begin{pmatrix} 0 & \tau_{yx} & \tau_{zx} \\ \tau_{xx} & \tau_{yy} & \tau_{zy} \\ \tau_{xz} & \tau_{zy} & \tau_{zz} \end{pmatrix} \begin{pmatrix} \beta_x \\ \beta_y \\ \beta_z \end{pmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{pmatrix} \]  \hspace{1cm} (2.32)

where \( \tau_{\alpha \beta} \) are the stress tensor components, \( \beta_{\alpha i} \) are the shear work and heat transfer terms, and \( \sigma_{\alpha i} \) are the diffusive flux terms of the Spalart-Allmaras transport equation. These terms are

\[ \tau_{xx} = 2(\mu + \mu_t)u_x + (\lambda_1 + \lambda_{1t}) \nabla \cdot \vec{V} \]
\[ \tau_{yy} = 2(\mu + \mu_t)v_y + (\lambda_1 + \lambda_{1t}) \nabla \cdot \vec{V} \]
\[ \tau_{zz} = 2(\mu + \mu_t)w_z + (\lambda_1 + \lambda_{1t}) \nabla \cdot \vec{V} \]
\[ \tau_{xy} = \tau_{yx} = (\mu + \mu_t)(u_y + v_x) \]
\[ \tau_{xz} = \tau_{zx} = (\mu + \mu_t)(u_z + w_x) \]
\[ \tau_{yz} = \tau_{zy} = (\mu + \mu_t)(v_z + w_y) \]
\[ \beta_{\alpha i} = (k + k_t) \frac{\partial T}{\partial x_i} + u \tau_{x'i} + v \tau_{y'i} + w \tau_{z'i} \]
\[ \sigma_{\alpha i} = \frac{1}{\sigma}(\mu + f_{nl} \rho \bar{V}) \frac{\partial \bar{V}}{\partial x_i} \]  \hspace{1cm} (2.33)

where \( u, v, \) and \( w \) are the velocity gradients of the \( i \)th Cartesian coordinate. The eddy viscosity, \( \mu_t \), and the function \( f_{nl} \) are described in Section 2.2.2.2. The molecular viscosity is calculated with Sutherland’s law as

\[ \mu(T) = \mu_0 \left( \frac{T}{T_0} \right)^{\gamma/2} \left( \frac{T_0 + C_S}{T + C_S} \right) \]  \hspace{1cm} (2.34)

27
where $C_S$ is Sutherland’s constant, $T_0$ is the reference temperature, and $\mu_0$ is the reference viscosity. Applying Stoke’s hypothesis, the 2nd coefficient of laminar and eddy viscosity are $\lambda_1 = -\frac{2}{3}\mu$ and $\lambda_{1t} = -\frac{2}{3}\mu_t$, respectively. From Reynold’s analogy, the laminar and turbulent thermal conductivities are $k = C_p \frac{\mu}{Pr}$ and $k_t = C_p \frac{\mu_t}{Pr_t}$, respectively. The properties of air are assumed for all fluid domain calculations in this dissertation, unless stated otherwise. These properties are $R_{gas} = 287 \frac{J}{kg\cdot K}$, $\gamma = 1.4$, $C_S = 110.4 K$, $\mu_0 = 1.716 \times 10^{-5} Pa\cdot s$, $T_0 = 273.15 K$, and $Pr = 0.72$. In reality, the turbulent Prandtl number for air varies throughout the flow, with large variations in the boundary layer. As a simplification, a constant turbulent Prandtl number of 0.9 is assumed. This value was chosen based on experiments described by McEligot and Taylor [108], Kays [109], and Kays et al. [110] and the S-A model common practice [107].

The source term is

$$S(U, \nabla U) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ S_\nu(U, \nabla U) \end{bmatrix} \tag{2.35}$$

where $S_\nu$ is the source term from the S-A transport equation.

### 2.2.2.2 Turbulence Model

Closure of the RANS equations is accomplished with the Spalart-Allmaras turbulence model. The eddy viscosity is defined as

$$\mu_t = \begin{cases} \rho f_{v1} \tilde{v} & \tilde{v} \geq 0 \\ 0 & \tilde{v} < 0 \end{cases} \tag{2.36}$$

where $\tilde{v}$ is the S-A working variable. The definition of $\mu_t$ is completed with

$$f_{v1} = \frac{\chi^2}{\chi^2 + c_{v1}} \quad \text{and} \quad \chi = \frac{v}{\tilde{v}}$$

where $v$ is the kinematic viscosity. The kinematic viscosity is related to the dynamic viscosity as

$$\mu = \rho v \quad \text{and} \quad \mu_t = \rho v_t$$
The flux contributions of the eddy viscosity can be seen in Eq. (2.33).

The S-A source term is

\[ S_{\tilde{\nu}} (U, \nabla U) = -\rho (P - D) - \frac{c_{h2}}{\sigma} \rho \nabla \tilde{\nu} \cdot \nabla \tilde{\nu} + \frac{1}{\sigma} (\nu + f_{n1} \tilde{\nu}) \nabla \rho \cdot \nabla \tilde{\nu} \]  

(2.37)

where \( P \) and \( D \) are the production and destruction source terms, respectively. The second term is a diffusive term from the original S-A transport equation. The last term arises when the original S-A equation is transformed into the conservative form (see Appendix B). The function \( f_{n1} \), is defined as

\[ f_{n1} = \begin{cases} 1 & \tilde{\nu} \geq 0 \\ \frac{c_{n1} + \chi}{c_{n1} - \chi} & \tilde{\nu} < 0 \end{cases} \]  

(2.38)

which maintains a positive diffusion coefficient for all \( \tilde{\nu} \). The production, \( P \), and destruction, \( D \), source terms in Eq. (2.37) are

\[ P = \begin{cases} c_{b1} (1 - f_{t2}) \tilde{Z} \tilde{\nu} & \tilde{\nu} \geq 0 \\ c_{b1} (1 - c_{t3}) \tilde{Z} \tilde{\nu} & \tilde{\nu} < 0 \end{cases} \]  

(2.39)

\[ D = \begin{cases} (c_{w1} f_{w} - c_{w1} f_{22}) \left[ \frac{\tilde{\nu}}{\tilde{\nu}} \right]^2 & \tilde{\nu} \geq 0 \\ -c_{w1} \left[ \frac{\tilde{\nu}}{\tilde{\nu}} \right]^2 & \tilde{\nu} < 0 \end{cases} \]  

(2.40)

where \( \tilde{Z} \) is the modified vorticity, \( Z \) is the vorticity magnitude, and \( d \) is the distance to the nearest wall. The method for calculating \( d \) is discussed in Section 3.2.1. The modified vorticity is defined as

\[ \tilde{Z} = \begin{cases} Z + \tilde{Z} & \tilde{Z} \geq -c_{v2} Z \\ Z + \frac{z(c_{v2} Z + c_{v3} Z)}{c_{v2} Z + c_{v3} Z - Z} & \tilde{Z} < -c_{v2} Z \end{cases} \]  

(2.41)

which is an updated formulation introduced by Allmaras et al. [107] to ensure that \( \tilde{Z} \) remains positive. The equation for \( \tilde{Z} \) is

\[ \tilde{Z} = \frac{\tilde{Z}}{c_{v2} f_{22}} f_{v2} \quad \text{and} \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \]  

(2.42)
and the vorticity magnitude is

\[ Z = \sqrt{(w_y - v_z)^2 + (u_z - w_x)^2 + (v_x - u_y)^2} \]  

(2.43)

However, when the squared terms are near zero, \( Z \) is not \( C^1 \) continuous with respect to the velocity gradients. This is clear by considering a situation where \( u_z - w_x = 0 \) and \( v_x - u_y = 0 \), resulting in

\[ Z = \sqrt{(w_y - v_z)^2 + (0)^2 + (0)^2} = \sqrt{(w_y - v_z)^2} = |w_y - v_z| \]  

(2.44)

which is not \( C^1 \) continuous at \( w_y - v_z = 0 \). To remove this discontinuity Eq. (2.43) is replaced with

\[ Z = \begin{cases} 
\frac{1}{2} \left( \varepsilon Z + \frac{Z^2}{2} \right) & Z^2 < \varepsilon Z \\
\sqrt{Z^2} & \text{otherwise}
\end{cases} \]  

(2.45)

where \( \varepsilon Z = 1 \times 10^{-6} \). The laminar suppression function, \( f_{t2} \), is

\[ f_{t2} = c_{t3} \exp \left( -c_{t4} \chi^2 \right) \]  

(2.46)

which was conceived to prevent premature transition. Transition is not considered in this dissertation; however, \( f_{t2} \) is included to maintain \( C^1 \) continuity of \( P \) at \( \tilde{\nu} = 0 \) and to ensure energy stability for negative \( \tilde{\nu} \) [107]. The \( f_w \) function is

\[ f_w = g \left( 1 + c_{w3} \right)^{1/6} \]  

(2.47)

with

\[ g = r + c_{w2} (r^6 - r) \]  

(2.48)

where the equation for \( r \), proposed by Allmaras et. al. [107], is

\[ r = \min \left( r_{lim}, \tilde{r} \right) \]  

(2.49)

with

\[ \tilde{r} = \frac{\tilde{\nu}}{Z \kappa^2 d^2} \]  

(2.50)
However, the minimization function in Eq. (2.49) produces a discontinuity in the derivative calculations. Instead, an approximation of the minimization function, recommended by Chaurasia [53], is used

\[
r = r_{lim} - (r_{lim} - \bar{r}) \left( \frac{\arctan \left( b \left( r_{lim} - \bar{r} \right) \right)}{\pi} + \frac{1}{2} \right) + \frac{\arctan (b)}{\pi} - \frac{1}{2}
\]

(2.51)

where \( r_{lim} \) is the limit of \( r \), and \( b \) determines how quickly \( r \) transitions to \( r_{lim} \). The constants are \( c_{b1} = 0.1355, \sigma = \frac{2}{3}, c_{b2} = 0.622, \kappa = 0.41, c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{(1+c_{b2})}{\sigma}, c_{w2} = 0.3, c_{w3} = 2, c_{v1} = 7.1, c_{v2} = 0.7, c_{v3} = 0.9, c_{v4} = 1.2, c_{t4} = 0.5, c_{n1} = 16, Pr_t = 0.9, r_{lim} = 5, b = 100. \)

The source term that arises from the conservation form transformation requires the density gradients. These gradients are obtained by differentiating the ideal gas law equation, which results in the following equation

\[
\nabla \rho = \frac{T \nabla p - p \nabla T}{R_{\text{gas}} T^2}
\]

(2.52)

Several challenges come about when applying this turbulence model. The issues are related to the use of a complete linearization and a modal representation of the cell polynomials. The difficulties and their solutions are discussed in Section 3.2. Verification of the S-A model is performed in Section 4.1.3. Additional evaluation of this implementation of the S-A model has been conducted by Schrock et al. [111].

### 2.2.2.3 Upwind Dissipation

The upwind dissipation of the Navier-Stokes advective fluxes is achieved with the Roe approximate Riemann solver [112]. A compact form developed for the DG solver PROJECT-X[113] and detailed in the theory guide of the DG solver XFLOW[114] is used. In this form, the upwind dissipation is

\[
\phi (U^+, U^-) = \left| \bar{A} (U^+, U^-) \right| (Q(U^+)) - (Q(U^-)) = \begin{pmatrix}
|\lambda_3| \Delta \rho + C_1 \\
|\lambda_3| \Delta (\rho \bar{V}) + C_1 \bar{V}_R + C_2 \bar{n} \\
|\lambda_3| \Delta (\rho E) + C_1 H_R + C_2 \bar{V}_R \cdot \bar{n} \\
|\lambda_3| \Delta (\rho \bar{V}) + C_1 \bar{V}_R
\end{pmatrix}
\]

(2.53)
where $\Delta a = (a^+ - a^-)$ for some variable $a$, and $(\cdot)_R$ indicates a Roe averaged quantity. The Roe averaged variables are defined as

\[
\begin{align*}
\bar{V}_R &= \frac{\sqrt{\rho^- V^-} + \sqrt{\rho^+ V^+}}{\sqrt{\rho^-} + \sqrt{\rho^+}} \\
H_R &= \frac{\sqrt{\rho^- H^-} + \sqrt{\rho^+ H^+}}{\sqrt{\rho^-} + \sqrt{\rho^+}} \\
\bar{v}_R &= \frac{\sqrt{\rho^- v^-} + \sqrt{\rho^+ v^+}}{\sqrt{\rho^-} + \sqrt{\rho^+}}
\end{align*}
\]  

(2.54)

The $C_1$ and $C_2$ terms from Eq. (2.53) are

\[
\begin{align*}
C_1 &= \frac{G_1}{c_R^2} s_1 + \frac{G_2}{c_R} s_2 \\
C_2 &= \frac{G_1}{c_R} s_2 + \frac{G_2}{c_R} s_1
\end{align*}
\]

(2.55)

where the Roe averaged speed of sound is

\[
c_R^2 = \gamma R_{gas} T_R
\]

(2.56)

and the Roe averaged temperature is

\[
T_R = \frac{\sqrt{\rho^- T^-} + \sqrt{\rho^+ T^+}}{\sqrt{\rho^-} + \sqrt{\rho^+}}
\]

(2.57)

The $G_1$, $G_2$, $s_1$, and $s_2$ terms from Eq. (2.55) are defined as follows

\[
\begin{align*}
G_1 &= (\gamma - 1) \left( \frac{1}{2} \bar{V}_R^2 \Delta \rho - \bar{V}_R \cdot \Delta (\rho \bar{V}) + \Delta (\rho E) \right) \\
G_2 &= -\left( \bar{V}_R \cdot \hat{n} \right) \Delta \rho + \Delta (\rho \bar{V}) \cdot \hat{n} \\
s_1 &= \frac{1}{2} (|\lambda_1| + |\lambda_2|) - |\lambda_3| \\
s_2 &= \frac{1}{2} (|\lambda_1| - |\lambda_2|)
\end{align*}
\]
An entropy fix formulation is used for the absolute value of the eigenvalues [115]. This formulation is

\[
|\lambda_i| = \begin{cases} 
\frac{1}{2} \left( \varepsilon_{\lambda} c_R + \frac{\lambda_i^2}{\varepsilon_{\lambda} c_R} \right) & -\varepsilon_{\lambda} c_R < \lambda_i < \varepsilon_{\lambda} c_R \\
\sqrt{\lambda_i^2} & \text{otherwise}
\end{cases}
\]  

(2.58)

where \(\varepsilon_{\lambda} = 0.01\). The eigenvalues are

\[
\lambda_1 = \bar{V}_R \cdot \bar{n} + c_R \\
\lambda_2 = \bar{V}_R \cdot \bar{n} - c_R \\
\lambda_3 = \bar{V}_R \cdot \bar{n}
\]  

(2.59)

### 2.2.2.4 Boundary Conditions

The boundary conditions in a DG discretization are weakly imposed. Hence, the boundary is imposed with fluxes integrated over \(\Gamma_b\), as shown in Eq. (2.8). These boundary fluxes require the calculation of \(U_B(U^-, U_b)\) and \(\nabla U_B(\nabla U^-, \nabla U_b)\). The boundary variable vectors, \(U_B\) and \(\nabla U_B\), are detailed for each boundary condition.

**Spalart-Allmaras**

The conditions imposed on \(\bar{V}_B\), \(p_B\), and \(T_B\) are independent of the conditions imposed on the S-A working variable. Therefore, the three types of S-A boundary conditions are discussed here and are not included the discussion of the other boundary conditions. For all inlet boundaries, the S-A variable to laminar viscosity ratio is specified. From this condition, the boundary value and boundary gradient are calculated as

\[
\bar{V}_B = \left( \frac{\bar{V}}{\bar{V}} \right)_b v^- \\
\nabla \bar{V}_B(\nabla \bar{V}^-, \nabla \bar{V}_b) = \nabla \bar{v}^-
\]  

(2.60)

where \(v^- = \frac{\mu(T^-)}{\rho}\). At outflows and symmetry planes, the S-A variable is extrapolated, which imposes the following

\[
\bar{V}_B(\bar{v}^-, \bar{v}_b) = \bar{v}^-
\]
\[ \nabla \tilde{v}_B = 0 \]  
(2.61)

The final boundary condition is a zero Dirichlet boundary condition for no-slip walls.

\[ \tilde{v}_B = 0 \]

\[ \nabla \tilde{v}_B \left( \nabla \tilde{v}^-, \nabla \tilde{v}_b \right) = \nabla \tilde{v}^- \]  
(2.62)

### Walls

Three walls are considered here; the slip wall, the adiabatic no-slip wall, and the iso-thermal no-slip wall. The slip wall models inviscid flow over a solid body. The adiabatic and iso-thermal no-slip walls model viscous boundaries with a zero heat flux and constant wall temperature, respectively. The weakly imposed slip walls and no-slip walls carry the same condition that \( \tilde{v}_B = 0 \). For both an adiabatic wall and slip wall, the boundary variable vector is

\[
U_B \left( U^-, U_b \right) = \begin{pmatrix}
0 \\
0 \\
0 \\
p^- \\
T^-
\end{pmatrix}
\]  
(2.63)

The iso-thermal wall imposes a wall temperature as

\[
U_B \left( U^-, U_b \right) = \begin{pmatrix}
0 \\
0 \\
0 \\
p^- \\
T_b
\end{pmatrix}
\]  
(2.64)

For the iso-thermal wall and slip-wall, the boundary gradient vector is

\[ \nabla U_B \left( \nabla U^-, \nabla U_b \right) = \nabla U^- \]  
(2.65)
The adiabatic no-slip wall specifies $\nabla T_B = 0$, such that the boundary gradient vector is

$$
\nabla U_B(U^-, U_b) = \begin{pmatrix}
\nabla u^- \\
\nabla v^- \\
\nabla w^- \\
\nabla p^- \\
0
\end{pmatrix}
$$

(2.66)

Symmetry

The symmetry condition requires that the velocity gradient normal to the boundary is zero. This is the same requirement of the slip wall, but the symmetry condition is imposed in a different way. The interior velocities are mirrored about the boundary face, as shown in the boundary value vector below

$$
U_B(U^-, U_b) = \begin{pmatrix}
u^- - 2 \left( \nabla^- \cdot \vec{n} \right) n_x \\
v^- - 2 \left( \nabla^- \cdot \vec{n} \right) n_y \\
w^- - 2 \left( \nabla^- \cdot \vec{n} \right) n_z \\
p^- \\
T^-
\end{pmatrix}
$$

(2.67)

where $n_x$, $n_y$, and $n_z$ are the x-, y-, and z-components of $\vec{n}$, respectively. As with the slip wall, the boundary gradient vector is

$$
\nabla U_B(\nabla U^-, \nabla U_b) = \nabla U^-
$$

(2.68)

Total Conditions Inflow

The boundary total pressure, $p_T$, and total temperature, $T_T$, are specified along with a unit vector, $\vec{n}_b$, to define the flow direction. From the total enthalpy equation, the static temperature is calculated as

$$
T_B = T_T - \frac{\nabla^- \cdot \vec{V}^-}{2C_p}
$$

(2.69)
Using the static and total temperature, the static pressure is determined using the isotropic relationship

\[ p_B = p_T \left( \frac{T_B}{T_T} \right)^\gamma \]  

(2.70)

Finally, the boundary velocities are calculated from the velocity magnitude and the specified flow direction, as follows

\[ \vec{V}_B = \left| \vec{V}^- \right| \vec{n}_b \]  

(2.71)

The boundary gradient vector is set to the interior gradients

\[ \nabla U_B (\nabla U^-, \nabla U_b) = \nabla U^- \]  

(2.72)

**Temperature and Velocity Inflow**

With this boundary condition, the velocity components and the temperature are specified and the pressure is extrapolated. The boundary variable and gradient vectors are

\[ U_B (U^-, U_b) = \begin{pmatrix} u_b \\ v_b \\ w_b \\ p^- \\ T_b \end{pmatrix} \]

\[ \nabla U_B (\nabla U^-, \nabla U_b) = \nabla U^- \]  

(2.73)
Static Pressure Outflow

The static pressure outflow is opposite of the previous boundary condition in that the pressure is specified and the other primitive variables are extrapolated. The boundary vectors are

\[
U_B(U^-, U_b) = \begin{pmatrix}
  u^- \\
v^- \\
w^- \\
p_b \\
T^-
\end{pmatrix}
\]

\[
\nabla U_B(\nabla U^-, \nabla U_b) = \nabla U^-
\]  

(2.74)

Riemann Invariant

The Riemann invariant boundary condition, described by Jameson [116], attempts to eliminate the reflection of outgoing waves. The invariants are based on one dimensional flow normal to the boundary. All quantities are specified along the boundary. How these specified quantities are imposed depends on the cell interior normal velocity direction and magnitude. The four possible imposed boundaries and the corresponding conditions are given below

- **Supersonic Inflow**
  \[ |\hat{V}^- \cdot \hat{n}| \geq c^-, \quad |\hat{V}^- \cdot \hat{n}| \leq 0 \]

- **Supersonic Outflow**
  \[ |\hat{V}^- \cdot \hat{n}| \geq c^-, \quad |\hat{V}^- \cdot \hat{n}| > 0 \]

- **Subsonic Inflow**
  \[ |\hat{V}^- \cdot \hat{n}| < c^-, \quad |\hat{V}^- \cdot \hat{n}| \leq 0 \]

- **Subsonic Outflow**
  \[ |\hat{V}^- \cdot \hat{n}| < c^-, \quad |\hat{V}^- \cdot \hat{n}| > 0 \]

where \( c^- = \sqrt{\gamma R_{gas} T^-} \) is the interior speed of sound. More than one boundary type can be imposed along one cell, which is not feasible for modal polynomials. Therefore, nodal polynomials are utilized for calculating the boundary vector quantities, which are then converted to modal polynomials. For all of the boundary types, the boundary gradient vector is set to zero.
Supersonic Inflow  Along the portion of the boundary designated a supersonic inflow, all of the boundary variables are imposed from the specified quantities. The boundary vector is

\[
U_B (U^-, U_b) = \begin{pmatrix} u_b \\ v_b \\ w_b \\ p_b \\ T_b \end{pmatrix}
\] (2.75)

Supersonic Outflow  To impose a supersonic outflow, all of the boundary variables are extrapolated from the interior. The boundary vector is

\[
U_B (U^-, U_b) = \begin{pmatrix} u^- \\ v^- \\ w^- \\ p^- \\ T^- \end{pmatrix}
\] (2.76)

Subsonic  For subsonic flow, the fixed \( R_\infty \) and extrapolated \( R_e \) Riemann invariants are introduced. The invariants are

\[
R_\infty = \left( \vec{V}_b \cdot \vec{n} \right) - \frac{2c_b}{\gamma - 1} \\
R_e = \left( \vec{V}^- \cdot \vec{n} \right) + \frac{2c^-}{\gamma - 1}
\] (2.77)

where \( c_b = \sqrt{\gamma R_{\text{gas}} T_b} \) is the specified speed of sound. From these invariants, the boundary speed of sound and normal velocity are

\[
c_B = \frac{\gamma - 1}{4} (R_e - R_\infty) \\
\vec{V}_n = \frac{1}{2} (R_e + R_\infty) \vec{n}
\] (2.78)
From the boundary speed of sound, the boundary temperature is

\[ T_B = \frac{c_B^2}{\gamma R_{\text{gas}}} \]

The tangential velocity is specified for an inflow boundary and extrapolated from the interior for an outflow boundary

\[
\vec{V}_t = \begin{cases} 
\vec{V}_b - \left( \vec{V}_b \cdot \vec{n} \right) \vec{n} & \text{Inflow} \\
\vec{V}^- - \left( \vec{V}^- \cdot \vec{n} \right) \vec{n} & \text{Outflow}
\end{cases}
\]

Similarly, a boundary entropy term can be defined as

\[
S_B = \begin{cases} 
\frac{c_p}{(T_b)_{\text{ref}} R_{\text{gas}}} & \text{Inflow} \\
\frac{c_p}{(T^-)_{\text{ref}} R_{\text{gas}}} & \text{Outflow}
\end{cases}
\]

From these boundary quantities, the boundary variable vector is

\[
U_B(U^-, U_b) = \begin{pmatrix} 
u_n + u_t \\
v_n + v_t \\
w_n + w_t \\
\frac{c_p}{(T_b)_{\text{ref}} R_{\text{gas}}} S_B \\
T_B
\end{pmatrix}
\]

where the \( ()_n \) and \( ()_t \) indicate the velocity components of the boundary normal and tangential velocity vectors, respectively.

### 2.3 Harmonic Balance Method

The derivation here is similar to the one presented by Ekici and Hall [45]. The fundamental assumption of the Harmonic Balance method is that the conservative variables are the combination of a time mean flow and a sum of periodic perturbations. Let \( Q(W) \) be a solution vector of conservative variables for all cells, where \( W \) is a solution vector of dependent variables. Suppose that a set of \( N \) equations are produced of the
form shown in Eq. (2.1), resulting in

\[
\frac{\partial Q(W_1)}{\partial t} + \nabla \cdot \vec{F}(W_1, \nabla W_1) + S(W_1, \nabla W_1) = 0
\]

\[
\frac{\partial Q(W_2)}{\partial t} + \nabla \cdot \vec{F}(W_2, \nabla W_2) + S(W_2, \nabla W_2) = 0
\]

\[
\vdots \quad \vdots
\]

\[
\frac{\partial Q(W_N)}{\partial t} + \nabla \cdot \vec{F}(W_N, \nabla W_N) + S(W_N, \nabla W_N) = 0 \tag{2.82}
\]

where \( W_n \) is the dependent variable solution vector for the \( n^{th} \) governing equation. Collapsing Eq. (2.82) into a single vector equation gives

\[
\frac{\partial Q^*}{\partial t} + \nabla \cdot \vec{F}^* + S^* = 0 \tag{2.83}
\]

where \( Q^* \) is a series of solution vectors, \( F^* \) is a series of fluxes, and \( S^* \) is a series of source terms

\[
Q^* = \begin{bmatrix} Q(W_1), & Q(W_2), & \ldots, & Q(W_N) \end{bmatrix}^T \tag{2.84}
\]

\[
F^* = \begin{bmatrix} \vec{F}(W_1, \nabla W_1), & \vec{F}(W_2, \nabla W_2), & \ldots, & \vec{F}(W_N, \nabla W_N) \end{bmatrix}^T \tag{2.85}
\]

\[
S^* = \begin{bmatrix} S(W_1, \nabla W_1), & S(W_2, \nabla W_2), & \ldots, & S(W_N, \nabla W_N) \end{bmatrix}^T \tag{2.86}
\]

and \( \nabla W_n \) is the variable gradient of the \( n^{th} \) equation. At this point, the equations in (2.82) remain independent of each other.

Consider the conservative solution vector represented as a Fourier series in time

\[
Q(W) = A_0 + \sum_{k=1}^{\infty} [A_k \sin(\omega_k t) + B_k \cos(\omega_k t)] \tag{2.87}
\]

where \( A_0, A_k, \text{ and } B_k \) are vectors of Fourier Coefficients for each cell and \( \omega_k \) is the \( k^{th} \) frequency. Note that the Fourier coefficients are not functions of time. Assume that the Fourier series presented in Eq. (2.87) is truncated to include only the frequencies \( \omega = [\omega_1, \omega_2, \ldots, \omega_K] \), as shown below

\[
Q(W) = A_0 + \sum_{k=1}^{K} [A_k \sin(\omega_k t) + B_k \cos(\omega_k t)] \tag{2.88}
\]

with \( K \) being the number of frequencies. Let the dependent variable solution vectors shown in Eq. (2.82) be
associated with an instant in time

\[ Q(W_n) = A_0 + \sum_{k=1}^{K} [A_k \sin(\omega_k t_n) + B_k \cos(\omega_k t_n)] \]  

(2.89)

where the instants in time, \( t_n \), make up a set of time levels, \( t = [t_1, t_2, \ldots, t_N] \). A restriction on the choice of time levels is that they must span the period of the lowest frequency. The most straightforward choice is to evenly space the time levels such as

\[ t = \frac{2\pi}{\omega_{\text{min}}} \left[ \frac{1}{N+1}, \frac{2}{N+1}, \ldots, \frac{N}{N+1} \right] \]

where \( \omega_{\text{min}} = \min(\omega) \). However, a more careful choice of time levels can improve the stability of HB simulations, as shown by Guedeney et al. [117]. The resulting system of linearly independent equations relating the conservative variable solution vectors is shown below

\[
\begin{bmatrix}
Q(W_1) \\
Q(W_2) \\
\vdots \\
Q(W_N)
\end{bmatrix}
= 
\begin{bmatrix}
1 & \sin(\omega_1 t_1) & \cdots & \sin(\omega_K t_1) & \cos(\omega_1 t_1) & \cdots & \cos(\omega_K t_1) \\
1 & \sin(\omega_1 t_2) & \cdots & \sin(\omega_K t_2) & \cos(\omega_1 t_2) & \cdots & \cos(\omega_K t_2) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
1 & \sin(\omega_1 t_N) & \cdots & \sin(\omega_K t_N) & \cos(\omega_1 t_N) & \cdots & \cos(\omega_K t_N)
\end{bmatrix}
\begin{bmatrix}
A_o \\
A_1 \\
\vdots \\
A_K
\end{bmatrix}
\]

(2.90)

This system of equations relates the series of conservative solution vectors,

\[ Q^* = \begin{bmatrix} Q(W_1), Q(W_2), \ldots, Q(W_N) \end{bmatrix}^T \]  

(2.91)

to the Fourier coefficient vectors,

\[ \hat{Q} = \begin{bmatrix} A_o, A_1, \ldots, A_K, B_1, \ldots, B_K \end{bmatrix}^T \]  

(2.92)

through an inverse Fourier transform matrix, denoted as \( E^{-1} \). The elements of \( E^{-1} \) are known because \( t \) and \( \omega \) are given. By taking the inverse of \( E^{-1} \), the Fourier transform operation can be represented as \( \hat{Q} = EQ^* \).

Using these two operations and recalling that the Fourier coefficient vectors are not time dependent, the
unsteady term in Eq. (2.83) can be written as

\[ \frac{\partial}{\partial t} (Q^*) = \frac{\partial}{\partial t} (E^{-1} \hat{Q}) = \frac{\partial}{\partial t} (E^{-1}) \hat{Q} = \frac{\partial E^{-1}}{\partial t} E^* = DQ^* \]  

(2.93)

where the HB community calls $D$ the pseudo-spectral operator (not to be confused with a spatial discretization approach that is similarly named). This operator is simply a matrix multiplication of the series of solutions that enforces Eq. (2.89). In other words, $D$ couples $Q^*$ such that the solutions satisfy time-varying sinusoidal functions. Through the available transformations, $Q^*$ can be used to obtain $\hat{Q}$ or a solution at any time $t$.

Analytical differentiation can be performed on $E^{-1}$ to obtain the following

\[ \frac{\partial E^{-1}}{\partial t} = \begin{bmatrix} 0 & \omega_1 \cos (\omega_1 t_1) & \cdots & \omega_K \cos (\omega_K t_1) & -\omega_1 \sin (\omega_1 t_1) & \cdots & -\omega_K \sin (\omega_K t_1) \\ 0 & \omega_1 \cos (\omega_1 t_2) & \cdots & \omega_K \cos (\omega_K t_2) & -\omega_1 \sin (\omega_1 t_2) & \cdots & -\omega_K \sin (\omega_K t_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \omega_1 \cos (\omega_1 t_N) & \cdots & \omega_K \cos (\omega_K t_N) & -\omega_1 \sin (\omega_1 t_N) & \cdots & -\omega_K \sin (\omega_K t_N) \end{bmatrix} \]  

(2.94)

The pseudo-spectral operator is calculated using the analytical derivative of $E^{-1}$ and the numerical calculation of $E$. Substituting Eq. (2.93) into Eq. (2.83) results in the Harmonic Balance equation

\[ DQ^* + \nabla \cdot F^* + S^* = 0 \]  

(2.95)

This equation is a series of steady-like equations where the derivation of the pseudo-spectral term $(DQ^*)$ is independent of the spatial flux and source terms. Multiplying Eq. (2.95) by the polynomial test function, $\hat{\psi}$, and integrating over the cell volumes, $\Omega_e$, provides the weak form of the Harmonic Balance equations

\[ \int_{\Omega_e} \hat{\psi} DQ^* d\Omega_e + \int_{\Omega_e} \hat{\psi} \nabla \cdot F^* d\Omega_e + \int_{\Omega_e} \hat{\psi} S^* d\Omega_e = 0 \]  

(2.96)

and applying Guass’s theorem gives

\[ \int_{\Omega_e} \hat{\psi} DQ^* d\Omega + \int_{\Gamma_e} \hat{\psi} F^* \cdot \vec{n} d\Gamma - \int_{\Omega_e} \nabla \hat{\psi} \cdot F^* \cdot \vec{n} d\Omega + \int_{\Omega_e} \hat{\psi} S^* d\Omega = 0 \]  

(2.97)

Besides the use of a series of solutions, the only difference between Eq. (2.3) and Eq. (2.97) is the first term.
Therefore, the spatial discretization of each equation can be considered independently. In fact, the spatial fluxes and source terms corresponding to the \(n^{th}\) solution are carried out identically to the process described in Section 2.1.1. The volume integral of the pseudo-spectral term is fairly simple to construct, as described in the section below.

### 2.3.1 Newton’s Method

As in Section 2.1.1.1, Newton’s method is used to solve Eq. (2.97). Let \(\mathcal{R}(W^m_n)\) be the spatial integrals of the \(n^{th}\) time level and the \(m^{th}\) Newton iteration. Though the DG discretization of Eq. (2.8) is used for this work, any formulation of \(\mathcal{R}(W^m_n)\) could follow this same process. Now, let \(\mathcal{R}^* = \left[ \mathcal{R}(W^m_1), \mathcal{R}(W^m_2), \ldots, \mathcal{R}(W^m_N) \right]^T\) be the series of residuals corresponding to the series of solutions \((Q^*)\). Performing the spatial discretization on Eq. (2.97) results in the following equation

\[
\int_{\Omega_e} \psi DQ^* d\Omega_e + \mathcal{R}^* = 0 \tag{2.98}
\]

where

\[
\mathcal{D}^* = \int_{\Omega_e} \psi DQ^* d\Omega_e \tag{2.99}
\]

is the pseudo-spectral residual. To solve Eq. (2.98), a system of linear equations is formed following the generalized form of Newton’s equation

\[
\left( \frac{\partial \mathcal{D}^*}{\partial W^*} + \frac{\partial \mathcal{R}^*}{\partial W^*} \right) \Delta W^* = - \left( \mathcal{D}^* + \mathcal{R}^* \right) \tag{2.100}
\]

where \(\Delta W^* = \left[ \Delta W^*_1, \Delta W^*_2, \ldots, \Delta W^*_N \right]\) are the update vectors such that \(W^{m+1}_n = W^m_n + \Delta W^*_n\). Comparing Eq. (2.98) to Eq. (2.97), \(\mathcal{R}^*\) is formed from the integrals of the series of fluxes and sources \((F^* \text{ and } S^*)\). Given that the \(n^{th}\) flux and source depends only on the \(n^{th}\) solution (see Eqs. 2.85 and 2.86), the \(n^{th}\) residual also depends only on the \(n^{th}\) solution. Therefore, the residual linearization can be simplified to

\[
\frac{\partial \mathcal{R}^*}{\partial W^*} = \begin{bmatrix}
\frac{\partial \mathcal{R}(W^m_1)}{\partial W^*_1} & 0 & 0 & 0 \\
0 & \frac{\partial \mathcal{R}(W^m_2)}{\partial W^*_2} & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \frac{\partial \mathcal{R}(W^m_N)}{\partial W^*_N}
\end{bmatrix} \tag{2.101}
\]
where $\frac{\partial(\hat{R}(W^m))}{\partial W^m}$ is the linearization of the spatial fluxes for the $n^{th}$ time level and $m^{th}$ Newton iteration. The pseudo-spectral linearization can be simplified because $D$ depends only on $\omega$ and $t$, and not the dependent variables. Hence, Eq. (2.100) becomes

$$
\left( \int_{\Omega_e} \psi D \frac{\partial Q^*}{\partial W^*} d\Omega_e + \frac{\partial \hat{R}^*}{\partial W^*} \right) \Delta W^* = - (\hat{R}^* + \hat{R}^*)
$$

(2.102)

with

$$
\frac{\partial Q^*}{\partial W^*} = \begin{bmatrix}
\frac{\partial Q(W^m)}{\partial W_1} & 0 & 0 & 0 \\
0 & \frac{\partial Q(W^m)}{\partial W_2} & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \frac{\partial Q(W^m)}{\partial W_N}
\end{bmatrix}
$$

(2.103)

where $\frac{\partial Q(W^m)}{\partial W^m}$ is the linearization matrix of the conservative variables for the $n^{th}$ time level and $m^{th}$ Newton iteration. The pseudo-spectral matrix is not a function of the cell volume, therefore it can be removed from the integral in Eq. (2.102), so that

$$
\int_{\Omega_e} \psi D \frac{\partial Q^*}{\partial W^*} d\Omega_e = D \int_{\Omega_e} \psi \frac{\partial Q^*}{\partial W^*} d\Omega_e = D
$$

$$
\begin{bmatrix}
\int_{\Omega_e} \psi \frac{\partial Q(W^m)}{\partial W_1} d\Omega_e & 0 & 0 & 0 \\
0 & \int_{\Omega_e} \psi \frac{\partial Q(W^m)}{\partial W_2} d\Omega_e & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \int_{\Omega_e} \psi \frac{\partial Q(W^m)}{\partial W_N} d\Omega_e
\end{bmatrix}
$$

(2.104)

The diagonal of $D$ is zero, so the above can be written as

$$
D \int_{\Omega_e} \psi \frac{\partial Q^*}{\partial W^*} d\Omega_e =
$$

$$
\begin{bmatrix}
0 & H^m_{1,2} & \cdots & H^m_{1,N} \\
H^m_{2,1} & 0 & \ddots & \vdots \\
\vdots & \ddots & \ddots & H^m_{N-1,N} \\
H^m_{N,1} & \cdots & H^m_{N,N-1} & 0
\end{bmatrix}
$$

(2.105)

where

$$
H^m_{i,j} = d_{i,j} \int_{\Omega_e} \psi \frac{\partial Q(W^m)}{\partial W_j} d\Omega_e
$$

(2.106)
Recasting Eq. (2.101)

\[
\frac{\partial R}{\partial W^*} = \begin{bmatrix}
A_1^m & 0 & \cdots & 0 \\
0 & A_2^m & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & A_N^m
\end{bmatrix}
\]

and summing the linearized terms, gives the full HB linear system

\[
A^* = \begin{bmatrix}
A_1^m & H_{1,2}^m & \cdots & H_{1,N}^m \\
H_{2,1}^m & A_2^m & \ddots & \vdots \\
\vdots & \ddots & \ddots & H_{N-1,N}^m \\
H_{N,1}^m & \cdots & H_{N,N-1}^m & A_N^m
\end{bmatrix}
\]

(2.107)

Equation (2.102) written in matrix-vector notation is

\[
\begin{bmatrix}
A_1^m & H_{1,2}^m & \cdots & H_{1,N}^m \\
H_{2,1}^m & A_2^m & \ddots & \vdots \\
\vdots & \ddots & \ddots & H_{N-1,N}^m \\
H_{N,1}^m & \cdots & H_{N,N-1}^m & A_N^m
\end{bmatrix}
\begin{bmatrix}
\Delta W_1^m \\
\Delta W_2^m \\
\vdots \\
\Delta W_N^m
\end{bmatrix} = -\begin{bmatrix}
R(W_1^m) + \sum_{n=1}^{N} \int_{\Omega_e} \psi d_{x,n} Q(W_n^m) d\Omega_e \\
R(W_2^m) + \sum_{n=1}^{N} \int_{\Omega_e} \psi d_{2,n} Q(W_n^m) d\Omega_e \\
\vdots \\
R(W_N^m) + \sum_{n=1}^{N} \int_{\Omega_e} \psi d_{N,n} Q(W_n^m) d\Omega_e
\end{bmatrix}
\]

(2.108)

which is solved simultaneously to obtain \(\Delta W^*\).

It is important to note that the linearization of the spatial residual and conservative variable series (\(\frac{\partial R^*}{\partial W^*}\) and \(\frac{\partial Q^*}{\partial W^*}\)) result in block diagonal matrices. Therefore, the full linearization matrix, Eq. (2.107), can be formed without interaction between solutions (except in the case of some interfaces).

### 2.3.2 Turbomachinery Flows

A common scenario for the simulation of multistage turbomachinery flows is depicted in Fig. 2.3. The axial direction is \(x\) and the rotation (or pitchwise) direction is \(\theta\). Each block indicates a blade passage from the \(i^{th}\) blade row. Every blade row has a rotation speed, \(\vec{\omega}\), and a spatial extent in the pitchwise direction, denoted as \(\Delta \theta\) and alternatively called the blade “pitch”. Generally, \(\Delta \theta\) varies from blade row to blade row because the blade counts are different. This difference is emphasized by the unequal size of the domains in Fig. 2.3. An additional complexity arises as \(\vec{\omega}\) alternates between zero in stationary blade rows to some value in the rotating blade rows. Simulations utilizing traditional time-marching techniques require equal
Figure 2.3: Multistage turbomachinery domain orientation.

pitch across these rotating domains. Therefore, blade passages must be added to the computational domain until \( \Delta \theta_i = \Delta \theta_j \) \( \forall i, j \in [1, M] \). This routinely leads to the simulation of a significant portion of the annulus for multistage simulations.

The HB method provides an alternative modeling strategy that reduces the number of simulated passages to one in each blade row. At the faces normal to \( \theta \) in Fig. 2.3, a phase lag condition is applied. As the domains move, the wakes and pressure waves produced by the blades perturb the flow field of each other. This interaction induces perturbation frequencies in the flow that vary from blade row to blade row. Therefore, the temporal and spectral characteristics of the blade rows are allowed to vary. In other words, the HB frequencies, \( \omega \), and time levels, \( t \), can differ between blade rows. The difference in \( \omega \) and \( t \) between intersecting blade rows is resolved using the relative motion interface, described in Section 2.3.2.3.

2.3.2.1 Blade Row Properties

The choice of HB frequencies for a given blade row is dependent upon the blade passing frequencies of the other blade rows. The fundamental frequencies of blade row \( j \) are defined as

\[
\omega_i^j = \left[ \frac{2\pi}{T_{1,j}}, \ldots, \frac{2\pi}{T_{i,j}}, \ldots, \frac{2\pi}{T_{M,j}} \right]^T
\]  

where \( T_{i,j} \) is the period of the perturbation induced by blade \( i \) on blade \( j \). The period is

\[
T_{i,j} = \frac{\Delta \theta_i}{\bar{\omega}_j - \bar{\omega}_i}
\]  

where \( \Delta \theta_i \) is the pitch of the \( i^{th} \) blade passage. Assuming an annulus with \( B_i \) being the blade count of the \( i^{th} \) blade row, the passage pitch is

\[
\Delta \theta_i = \frac{2\pi}{B_i}
\]
Substituting Eqs. (2.110) and (2.111) into Eq. (2.109) results in

\[
\omega_j^F = \begin{bmatrix} B_1(\bar{\omega}_j - \bar{\omega}_1), \ldots, B_i(\bar{\omega}_j - \bar{\omega}_i), \ldots, B_M(\bar{\omega}_j - \bar{\omega}_M) \end{bmatrix}^T
\]  

(2.112)

For a linear cascade, the blades move with a reference frame velocity vector, \( \vec{V}_{rf} = [u_{rf}, v_{rf}, w_{rf}] \).

Thus, \( \Delta \theta_i \) is taken as the distance of the blade passage tangent to \( \vec{V}_{rf} \), and \( \bar{\omega}_j \) is replaced with \( \vec{V}_{rf} \). The period for the cascade is

\[
(T_{i,j}) = \frac{\Delta \theta_i}{\left| (\vec{V}_{rf})_j - (\vec{V}_{rf})_i \right|}
\]

and the resulting fundamental frequencies according to Eq. (2.109) are

\[
\left( \omega_j^F \right)_{Cascade} = \begin{bmatrix} 2\pi \left| (\vec{V}_{rf})_j - (\vec{V}_{rf})_1 \right| / \Delta \theta_i, \ldots, 2\pi \left| (\vec{V}_{rf})_j - (\vec{V}_{rf})_i \right| / \Delta \theta_i, \ldots, 2\pi \left| (\vec{V}_{rf})_j - (\vec{V}_{rf})_M \right| / \Delta \theta_M \end{bmatrix}^T
\]

(2.113)

Often harmonics of the fundamental frequency and linear combinations of several blade row perturbation frequencies are desired. Hence, the following equation, discussed by Gopinath et al. [46], provides a general form for the choice of HB frequencies

\[
\omega_k^j = \sum_{i=1}^M n_{k,i} B_i(\bar{\omega}_j - \bar{\omega}_k)
\]

(2.114)

with a similar equation for a linear cascade being

\[
\left( \omega_k^j \right)_{Cascade} = \sum_{i=1}^M n_{k,i} \frac{2\pi \left| (\vec{V}_{rf})_j - (\vec{V}_{rf})_i \right|}{\Delta \theta_i}
\]

(2.115)

Alternatively, Eqs. (2.114) and (2.115) can be rewritten in matrix-vector form as

\[
\omega^j = n^j \omega_j^F
\]

(2.116)

\[
\left( \omega^j \right)_{Cascade} = n^j \left( \omega_j^F \right)_{Cascade}
\]

(2.117)
where \( \omega^j \) and \( (\omega)^\text{Cascade} \) are the vectors of specified HB frequencies for blade row \( j \), and

\[
n^j = \begin{bmatrix}
n^j_{1,1} & \cdots & n^j_{1,i} & \cdots & n^j_{1,M} \\
\vdots & & \ddots & & \vdots \\
n^j_{k,1} & n^j_{k,i} & n^j_{k,M} \\
\vdots & & \ddots & & \vdots \\
n^j_{K,1} & \cdots & n^j_{K,i} & \cdots & n^j_{K,M}
\end{bmatrix}
\]

(2.118)

where \( n^j_{k,i} \) are integers that describe the contribution of the \( i^{th} \) blade row to the \( k^{th} \) HB frequency of the \( j^{th} \) blade row. These integers are chosen to capture the most significant unsteady features. Generally, \( n^j \) is sparse because the dominant frequencies in any blade row are derived from one or two fundamental frequencies (often the upstream and downstream blade rows). From Eq. (2.115), it is clear that blade rows with the same speed do not contribute to the HB frequencies of each other. As a result, \( n^j_{k,i} \) are inconsequential when \( \vec{\omega}^j = \vec{\omega}^i \) or \( (\vec{V}_{rf})^j = (\vec{V}_{rf})^i \).

### 2.3.2.2 Phase Lag Interface

In traditional time-marching implementations, the blade passages are repeated until the faces in the rotation direction are periodic. Reducing the number of passages to one leads to periodic faces where the solution is not necessarily periodic. Consider the periodic and aperiodic perturbation of \( Q \) at any time level, as shown in Fig 2.4. The period of the perturbation in the rotation direction is denoted as \( \sigma \). Because \( \sigma \) is equivalent to the domain pitch \( (\Delta \theta) \) in Fig 2.4a, the periodic condition is valid (i.e. \( Q(\theta_j) = Q(\theta_l) \)). However, when \( \Delta \theta \neq \sigma \), as presented in Fig. 2.4b, the solution at the periodic boundaries does not match. Instead, the solutions are phase shifted along the perturbation. This phase shift occurs on phase lag interfaces, which are along the periodic faces.

From Fig. 2.4, the phase shift of the \( k^{th} \) frequency is

\[
\varphi_k = 2\pi \frac{\Delta \theta}{\sigma_k}
\]

(2.119)

where \( \sigma_k \) is the period of the perturbation related to the \( k^{th} \) HB frequency. Adding the phase shift to
Eq. (2.88) results in the following equation
\[
Q(W) = A_0 + \sum_{k=1}^{K} [A_k \sin (\omega_k t + \varphi_k) + B_k \cos (\omega_k t + \varphi_k)]
\] (2.120)

The trigonometric relation
\[
\begin{align*}
\sin (\alpha \pm \beta) &= \sin \alpha \cos \beta \pm \cos \alpha \sin \beta \\
\cos (\alpha \pm \beta) &= \cos \alpha \cos \beta \mp \sin \alpha \sin \beta
\end{align*}
\] (2.121)

applied to Eq. (2.120) and expanded gives
\[
Q(W) = A_0 + \sum_{k=1}^{K} [A_k \sin (\omega_k t) \cos \varphi_k + A_k \cos (\omega_k t) \sin \varphi_k + B_k \cos (\omega_k t) \cos \varphi_k - B_k \sin (\omega_k t) \sin \varphi_k]
\] (2.122)

Combing the $\sin (\omega_k t)$ and $\cos (\omega_k t)$ terms
\[
Q(W) = A_0' + \sum_{k=1}^{K} [A_k' \sin (\omega_k t) + B_k' \cos (\omega_k t)]
\] (2.123)
where $A'_k$ and $B'_k$ are the phase shifted Fourier coefficients

$$
A'_k = A_k \cos(\varphi_k) - B_k \sin(\varphi_k)
$$

$$
B'_k = A_k \sin(\varphi_k) + B_k \cos(\varphi_k)
$$

(2.124)

and $A'_0 = A_0$. This Fourier coefficient phase shift can be described with a phase shift matrix

$$
\Phi = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cos(\varphi_1) & 0 & 0 & -\sin(\varphi_1) & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \cos(\varphi_K) & 0 & 0 & -\sin(\varphi_K) \\
0 & \sin(\varphi_1) & 0 & 0 & \cos(\varphi_1) & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \sin(\varphi_K) & 0 & 0 & \cos(\varphi_K)
\end{bmatrix}
$$

(2.125)

and

$$
\hat{Q}_\varphi = \Phi \hat{Q}
$$

(2.126)

where $\hat{Q}_\varphi = \begin{bmatrix} A'_0, & A'_1, & \cdots, & A'_K, & B'_1, & \cdots, & B'_K \end{bmatrix}^T$ are the phase shifted Fourier coefficients. The shifted solutions can be determined using $E$ and $E^{-1}$ as

$$
Q^*_\varphi = E^{-1} \hat{Q}_\varphi = E^{-1} \Phi \hat{Q} = E^{-1} \Phi E Q^*
$$

(2.127)

with the final form

$$
Q^*_\varphi = P Q^*
$$

(2.128)

where $P$ is the phase shift operator.

A description of the perturbation period, $\sigma_k$, is based on the multistage turbomachinery properties of Section 2.3.2.1. This description uses the so-called “nodal diameters”, originally defined by Gopinath et al. [46] as

$$
A'_k = \sum_{i=1}^{n_{\text{Rows}}} n_{k,i} B_i
$$

(2.129)
where
\[ \phi^j_k = \frac{1}{N^j_k} \]

To avoid confusion with DG polynomial representations (nodal and modal), these “nodal diameters” will be referred to as HB diameters in this work. Comparing Eqs. 2.115 and 2.129, it is clear that the HB diameters are the ratio of the HB frequencies to the relative rotation speed. For linear cascade simulations, the HB diameters are
\[
\left( N^{i,j}_k \right)_{\text{Cascade}} = 2 \pi \sum_{i=1}^{n_{\text{Rows}}} \frac{\eta^i_{k,l}}{\Delta \theta_j} \tag{2.130}
\]

By inspection, the periodic perturbation defined by \( C \sigma_k = \Delta \theta \), where \( C \) is some integer constant, results in \( A'_k = A_k \) and \( B'_k = B_k \). It follows that \( \Phi = I \), \( P = I \), and \( Q^* = Q^* \). From Eqs. (2.129) and (2.130), the periodic case occurs for the annular problem when \( B_j = B_i \) and for the linear cascade problem when \( \Delta \theta_j = \Delta \theta_i \). This is expected as it is the same condition that must be reached in simulations with the traditional time-marching technique.

**Interface Conditions**

To complete the phase lag interface description, \( W^+_\Sigma \), \( \nabla W^+_\Sigma \), \( W^-_\Sigma \), and \( \nabla W^-_\Sigma \) must be defined. The interior state at the phase lag faces are not modified by the interface, hence \( W^-_\Sigma = W^- \) and \( \nabla W^-_\Sigma = \nabla W^- \) for all \( N \) solutions. The exterior quantities are obtained from the phase shifted condition as \( W^+_\Sigma = W_{\phi} \) and \( \nabla W^+_\Sigma = \nabla W_{\phi} \) for all \( N \) solutions, where \( W_{\phi} \) and \( \nabla W_{\phi} \) are the phase shifted exterior variables and gradients.

**Interface Jacobian Matrices**

To complete the linear system described in Section 2.1.1.1, the interior and exterior Jacobians of the interface, \( \frac{\partial W^\pm}{\partial W^\pm} \), are required. For the phase lag interface, the interior Jacobians are the identity matrix \( \left( \frac{\partial W^-}{\partial W^-} = I \right) \). The exterior Jacobians depend on the choice of \( W \). Two situations are discussed; when the dependent variable vector is the conservative variable vector, \( W = Q \), and when it is not (such as the primitive variable fluid flow equations). The HB operations cannot be directly applied to the dependent variables; therefore, an approximation is made, when \( W \neq Q \), to reduce the complexity of forming the dependent variable Jacobians.

**Conservative Variable Case, \( W = Q \)** Consider that the dependent variable vector is the conservative variable vector. To obtain the interface update vector in Eq. (2.15), the interface to conservative variable...
Jacobians on the exterior, $\frac{\partial Q_\Sigma^+}{\partial \phi}$, are needed. The Jacobian is

$$\frac{\partial Q_\Sigma^+}{\partial Q^+} = \frac{\partial Q^+}{\partial Q^+} = P$$  \hspace{1cm} (2.131)

making the update vector

$$\Delta Q_\Sigma^+ = P\Delta Q^+$$  \hspace{1cm} (2.132)

which means the interface update vector of each solution requires exterior variables from $N$ solutions. In practice, this would require the communication of $O(N)$ to pass the update vectors from the donor to the receiver. Instead, a phase shifted update vector is calculated by the donor

$$\Delta Q_\phi^+ = P\Delta Q^+$$

$$\Delta Q_\Sigma^+ = \Delta Q_\phi^+$$  \hspace{1cm} (2.133)

and only a single update vector, $\Delta Q_\phi^+$, is passed to the receiver.

**Dependent Variable Case, $W \neq Q$** The HB pseudo-spectral operator acts on the conservative variables. Therefore, an operator derived from the pseudo-spectral operator is only valid for the conservative variables. In other words,

$$W^*_\phi \neq PW^*$$  \hspace{1cm} (2.134)

Instead, $Q^*$ must be calculated from $W^*$ prior to the phase lag operation. Then, $W^*_\phi$ can be retrieved as

$$W^*_\phi = \left[ W(Q_{\phi 1}), \ W(Q_{\phi 2}), \ \ldots, \ W(Q_{\phi N}) \right]^T$$  \hspace{1cm} (2.135)

However, this operation is not linear. This means that the exterior interface to update vector, $\Delta W_\Sigma^+$, does not equal the phase shifted update vector, $\Delta W_\phi^+$, directly. Instead, Jacobians that follow the conversion to and from the conservative variables would be required. This increases the cost and complexity of the operation. To avoid this situation, $P$ is assumed to be valid for the dependent variables as well, hence $\Delta W_\Sigma^+ = P\Delta W^+$. This is an approximation, whose implications are discussed in Section 3.4.5.
2.3.2.3 Relative Motion Interface

Two techniques have been developed for the HB method to resolve the interactions at the interface between blade rows. The method by Ekici and Hall [45] involves matching the Fourier coefficients from a spatial transform of $\hat{Q}$ across the interface. Gopinath et al. [46] proposed a technique to filter solutions reconstructed across the interface. The filter method was chosen because it allows for a simpler implementation. Details of this choice and the spatial matching of the receiver and donor domains are discussed in Section 3.4.4.

Consider the two blade passages shown in Fig. 2.5. In practice, each domain will be both a donor and receiver. But for now, let the blue domain be the donor, with HB properties

$$
\begin{align*}
\mathbf{\hat{t}} &= [\mathbf{\hat{t}}_1, \mathbf{\hat{t}}_2, \cdots, \mathbf{\hat{t}}_\mathcal{N}] \\
\mathbf{\hat{\omega}} &= [\mathbf{\hat{\omega}}_1, \mathbf{\hat{\omega}}_2, \cdots, \mathbf{\hat{\omega}}_\mathcal{K}]
\end{align*}
$$

(2.136)

where $\hat{}$ indicates a donor property. Let the green domain be the receiver, with HB properties

$$
\begin{align*}
\mathbf{\tilde{t}} &= [\mathbf{\tilde{t}}_1, \mathbf{\tilde{t}}_2, \cdots, \mathbf{\tilde{t}}_\mathcal{N}] \\
\mathbf{\tilde{\omega}} &= [\mathbf{\tilde{\omega}}_1, \mathbf{\tilde{\omega}}_2, \cdots, \mathbf{\tilde{\omega}}_\mathcal{K}]
\end{align*}
$$

(2.137)

where $\tilde{}$ indicates a receiver property. There is no requirement that $\mathcal{N} = \tilde{\mathcal{N}}$, $\mathcal{K} = \tilde{\mathcal{K}}$, or $\mathbf{\hat{\omega}} = \mathbf{\tilde{\omega}}$. The donor
constructs the solution for the receiver at time levels $\bar{t}$, giving

$$(Q_1(W))_{\text{Receiver}} = (A_0)_{\text{Donor}} + \sum_{k=1}^{K} [(A_k)_{\text{Donor}} \sin (\hat{\omega}_k \bar{t}_1) + (B_k)_{\text{Donor}} \cos (\hat{\omega}_k \bar{t}_1)]$$

$$(Q_2(W))_{\text{Receiver}} = (A_0)_{\text{Donor}} + \sum_{k=1}^{K} [(A_k)_{\text{Donor}} \sin (\hat{\omega}_k \bar{t}_2) + (B_k)_{\text{Donor}} \cos (\hat{\omega}_k \bar{t}_2)]$$

\[ \vdots \]

$$(Q_N(W))_{\text{Receiver}} = (A_0)_{\text{Donor}} + \sum_{k=1}^{K} [(A_k)_{\text{Donor}} \sin (\hat{\omega}_k \bar{t}_N) + (B_k)_{\text{Donor}} \cos (\hat{\omega}_k \bar{t}_N)]$$

(2.138)

or in matrix-vector form

$$\mathbf{Q}^*_{\text{Receiver}} = \tilde{E}^{-1}_{11} \hat{\mathbf{Q}}_{\text{Donor}}$$

(2.139)

where

$$\tilde{E}^{-1}_{11} = \begin{bmatrix}
1 & \sin (\hat{\omega}_1 \bar{t}_1) & \cdots & \sin (\hat{\omega}_K \bar{t}_1) & \cos (\hat{\omega}_1 \bar{t}_1) & \cdots & \cos (\hat{\omega}_K \bar{t}_1) \\
1 & \sin (\hat{\omega}_1 \bar{t}_2) & \cdots & \sin (\hat{\omega}_K \bar{t}_2) & \cos (\hat{\omega}_1 \bar{t}_2) & \cdots & \cos (\hat{\omega}_K \bar{t}_2) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
1 & \sin (\hat{\omega}_1 \bar{t}_N) & \cdots & \sin (\hat{\omega}_K \bar{t}_N) & \cos (\hat{\omega}_1 \bar{t}_N) & \cdots & \cos (\hat{\omega}_K \bar{t}_N)
\end{bmatrix}$$

(2.140)

and the use of $\tilde{\cdot}$ signifies an operator composed of both donor and receiver HB properties. To relate the receiver solution and the donor solution, the donor Fourier transform matrix can be used

$$\hat{\mathbf{Q}}_{\text{Donor}} = \hat{E} \mathbf{Q}^*_{\text{Donor}}$$

(2.141)

However, the donor cannot provide $\mathbf{Q}^*_{\text{Donor}}$ along the entire face of the receiver domain for all $\bar{t}$ (notice the mismatch at the interface in Fig. 2.5). In practice, the donor provides information to copies of the receiver face that are offset by $\Delta \theta_{\text{Donor}}$ (see Section 3.4.4). This requires a phase shift of the donor solution

$$\hat{\mathbf{Q}}_{\text{Donor}} = \Phi \hat{E} \mathbf{Q}^*_{\text{Donor}}$$

(2.142)

where $\Phi$ is the same phase shift matrix in Eq. (2.125). The phase shifts, $\varphi_k$, are calculated from the offset of the receiver copies. Substituting Eq. (2.142) into Eq. (2.139) gives

$$\mathbf{Q}^*_{\text{Receiver}} = \tilde{E}^{-1}_{11} \Phi \hat{E} \mathbf{Q}^*_{\text{Donor}}$$

(2.143)
which completes the transfer of information from the donor variables to the receiver variables. The receiver is effectively sampling from the donor solution. If the donor frequencies are higher than the receiver frequencies, aliasing can occur. This effect has been known to cause spurious oscillations that significantly reduce the stability of the simulation (see Gopinath et al. [46]).

To mitigate the effect of aliasing, the receiver solutions are constructed at an increased number of time levels

$$\tilde{t}_2 = \left[ \tilde{t}_1, \tilde{t}_{1.5}, \tilde{t}_2, \tilde{t}_{2.5}, \cdots, \tilde{t}_{N-1}, \tilde{t}_N, \tilde{t}_{N+0.5} \right]$$  \hspace{1cm} (2.144)

where these $2\hat{N}$ time levels are generated by halving the time levels in Eq. (2.137). This can be thought of as the receiver domain increasing the sampling rate. Using $\tilde{t}_2$ in place of $\tilde{t}_1$ and reconstruction Eq. (2.143) results in

$$Q^*_\text{Receiver} = \tilde{E}^{-1}_{21} \Phi \tilde{E} Q^*_\text{Donor}$$  \hspace{1cm} (2.145)

where $Q^*_\text{Receiver}$ now contains $2\hat{N}$ receiver solutions. The combined inverse Fourier Transform operator is

$$\tilde{E}^{-1}_{21} = \begin{bmatrix}
1 & \sin(\tilde{\omega}_1 \tilde{t}_1) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_1) & \cos(\tilde{\omega}_1 \tilde{t}_1) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_1) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{1.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{1.5}) & \cos(\tilde{\omega}_1 \tilde{t}_{1.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{1.5}) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_2) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_2) & \cos(\tilde{\omega}_1 \tilde{t}_2) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_2) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{2.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{2.5}) & \cos(\tilde{\omega}_1 \tilde{t}_{2.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{2.5}) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_N) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_N) & \cos(\tilde{\omega}_1 \tilde{t}_N) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_N) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{N+0.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{N+0.5}) & \cos(\tilde{\omega}_1 \tilde{t}_{N+0.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{N+0.5})
\end{bmatrix}$$  \hspace{1cm} (2.146)

and the $\tilde{E}^{-1}_{21}$ matrix is of size $2\hat{N} \times \hat{N}$.

Now that the receiver solutions are calculated, a filtering operation is performed. The relation between the receiver frequencies and the $2\hat{N}$ generated solutions is

$$\tilde{E}^{-1}_{21} \hat{Q}_\text{Receiver} = Q^*_\text{Receiver}$$  \hspace{1cm} (2.147)
where the $\tilde{E}_{21}^{-1}$ matrix is

$$
\tilde{E}_{21}^{-1} = 
\begin{bmatrix}
1 & \sin(\tilde{\omega}_1 \tilde{t}_1) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_1)
& \cos(\tilde{\omega}_1 \tilde{t}_1) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_1) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{1.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{1.5})
& \cos(\tilde{\omega}_1 \tilde{t}_{1.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{1.5}) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_2) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_2)
& \cos(\tilde{\omega}_1 \tilde{t}_2) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_2) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{2.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{2.5})
& \cos(\tilde{\omega}_1 \tilde{t}_{2.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{2.5}) \\
& \vdots & \vdots & \vdots & \vdots \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{N}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{N})
& \cos(\tilde{\omega}_1 \tilde{t}_{N}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{N}) \\
1 & \sin(\tilde{\omega}_1 \tilde{t}_{N+0.5}) & \cdots & \sin(\tilde{\omega}_K \tilde{t}_{N+0.5})
& \cos(\tilde{\omega}_1 \tilde{t}_{N+0.5}) & \cdots & \cos(\tilde{\omega}_K \tilde{t}_{N+0.5})
\end{bmatrix}
$$

The inverse of $\tilde{E}_{21}^{-1}$ must be taken. Given that this matrix is not square, a generalized Moore-Penrose pseudo-inverse is performed (see Appendix D). The Moore-Penrose inverse is

$$
(\tilde{E}_{21}^{-1})^+ = \left( (\tilde{E}_{21}^{-1})^T \tilde{E}_{21}^{-1} \right)^{-1} (\tilde{E}_{21}^{-1})^T
$$

Applying $(\tilde{E}_{21}^{-1})^+$ to Eq. (2.147) gives

$$
\hat{Q}_{\text{Receiver}} = (\tilde{E}_{21}^{-1})^+ Q_{2\text{Receiver}}^* (2.150)
$$

and multiplying both sides by the Fourier transform operator, $\tilde{E}_{11}$, gives

$$
Q_{\text{Receiver}}^* = F Q_{2\text{Receiver}}^* (2.151)
$$

where $F$ is the filter operator

$$
F = \tilde{E}_{11}^{-1} (\tilde{E}_{21}^{-1})^+
$$

and $\tilde{E}_{11}$ is $E$ from Section 2.3 using the receiver HB properties.

**Linear Cascade Interface Conditions**

The relative motion interface must account for the difference in rotation speeds. In this work, only linear cascade simulations have been completed with the relative motion interface. The change in reference frames is reflected in the calculation of $W^+_\Sigma$ and $\nabla W^+_\Sigma$. 

56
For the conservative variable fluid flow equations, the filtered variables are

\[ Q^+_F = \left[ (\rho^+_F), (\rho u^+_F), (\rho v^+_F), (\rho w^+_F), (\rho E^+_F), (\rho \tilde{\nu}^+_F) \right]^T \]

which are supplied by the donor. The receiver side interface variables, \( Q^+_\Sigma \), are to be calculated. The density and S-A variables are unaltered. The change in momentum of the reference frame is reflected in the interface momentum

\[ \left( \rho \vec{V} \right)^+_\Sigma = \left( \rho \vec{V} \right)^+_F + \rho^+_F \Delta \vec{V}_{rf} \]  

where

\[ \Delta \vec{V}_{rf} = \begin{bmatrix} \Delta u_{rf} & \Delta v_{rf} & \Delta w_{rf} \end{bmatrix} = \left( \vec{V}_{rf} \right)_{\text{Receiver}} - \left( \vec{V}_{rf} \right)_{\text{Donor}} \]

is the velocity difference between the two reference frames. Total energy is converted by subtracting the contribution of the donor velocity and adding the same term using the receiver velocity, resulting in

\[ (\rho E)^+_\Sigma = (\rho E)^+_F - \frac{1}{2} \rho^+_F \vec{V}^+_F \cdot \vec{V}^+_F + \frac{1}{2} \rho^+_F \vec{V}^+_\Sigma \cdot \vec{V}^+_\Sigma \]  

By substituting \( \left( \rho \vec{V} \right)^+_F = \left( \rho \vec{V} \right)^+_\Sigma - \rho^+_F \Delta \vec{V}_{rf} \) into Eq. (2.154) and simplifying, the interface total energy is

\[ (\rho E)^+_\Sigma = (\rho E)^+_F + \left( \rho \vec{V} \right)^+_\Sigma \cdot \Delta \vec{V}_{rf} - \frac{1}{2} \rho^+_F \Delta \vec{V}_{rf} \cdot \Delta \vec{V}_{rf} \]

The resulting interface variable vector is

\[ Q^+_\Sigma = \begin{bmatrix} \rho^+_F \\ (\rho u^+_F) + \rho^+_F \Delta u_{rf} \\ (\rho v^+_F) + \rho^+_F \Delta v_{rf} \\ (\rho w^+_F) + \rho^+_F \Delta w_{rf} \\ (\rho E)^+_F + \left( \rho \vec{V} \right)^+_F \cdot \Delta \vec{V}_{rf} + \frac{1}{2} \rho^+_F \Delta \vec{V}_{rf} \cdot \Delta \vec{V}_{rf} \\ (\rho \tilde{\nu}^+_F) \end{bmatrix} \]

(2.155)
with the gradient vector being similarly formed

\[
\nabla Q^+ = \begin{pmatrix}
\nabla \rho^+_F \\
(\nabla \rho u)^+_F + \nabla \rho^+_F \Delta u_f \\
(\nabla \rho v)^+_F + \nabla \rho^+_F \Delta v_f \\
(\nabla \rho w)^+_F + \nabla \rho^+_F \Delta w_f \\
(\nabla \rho E)^+_F + \left( \nabla \rho \tilde{V} \right)^+_F \Delta \tilde{V}_f + \frac{1}{2} \nabla \rho^+_F \Delta \tilde{V}_f \cdot \Delta \tilde{V}_f \\
\n\nabla (\rho \tilde{V})^+_F
\end{pmatrix}
\]

(2.156)

The interior interface conditions are \( Q^- = Q^+ \), \( \nabla Q^- = \nabla Q^+ \).

For the primitive variable fluid flow equations, the change in reference frames only affects the velocities. The filtered variables are

\[
U^+_F = \begin{bmatrix}
u^+_F, \quad \nu^+_F, \quad \nu^+_F, \quad T^+_F, \quad \nu^+_F \end{bmatrix}^T
\]

and the interface variables and gradients are

\[
U^- = \begin{bmatrix}
u^+_F + \Delta u_f \\
\nu^+_F + \Delta v_f \\
\nu^+_F + \Delta w_f \\
p^+_F \\
T^+_F \\
\tilde{V}^+_F
\end{bmatrix}
\]

(2.157)

\[
\nabla U^+_F = \begin{bmatrix}
\nabla u^+_F \\
\nabla \nu^+_F \\
\nabla \nu^+_F \\
\nabla p^+_F \\
\nabla T^+_F \\
\nabla \tilde{V}^+_F
\end{bmatrix}
\]

(2.158)

The interior interface conditions are \( U^- = U^+ \), and \( \nabla U^- = \nabla U^+ \).
Interface Jacobian Matrices

As with the phase lag interface, the relative motion interior interface Jacobians are the identity matrix \( \left( \frac{\partial w}{\partial w} = I \right) \). Calculating the exterior interface Jacobians, \( \frac{\partial w}{\partial w} \), depends on the choice of the dependent variable vector. An argument similar to the one in Section 2.3.2.2 can be made here for the inaccuracy of directly applying the relative motion filter and phase shift operations to the dependent variables. Instead, transformations to and from the conservative variables are necessary. Performing these transformations imposes a significant obstacle when attempting to calculate the interface Jacobians.

Conservative Variable Case, \( W = Q \) Let the dependent variable vector be the conservative variable vector. Using the chain rule, the exterior Jacobian is

\[
\frac{\partial Q^+}{\partial Q^+} = \frac{\partial Q^+}{\partial Q^+} \frac{\partial Q^+}{\partial Q^+} \frac{\partial Q^+}{\partial Q^+} = \frac{\partial Q^+}{\partial Q^+} F \Phi
\]

(2.159)

where \( Q^+ \) are the filtered variables. The interface Jacobian with respect to the filtered variables is

\[
\frac{\partial Q^+}{\partial Q^+} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
\Delta u_{rf} & 1 & 0 & 0 & 0 \\
\Delta v_{rf} & 0 & 1 & 0 & 0 \\
\Delta w_{rf} & 0 & 0 & 1 & 0 \\
\frac{1}{2} \Delta \vec{V}_{rf} \cdot \Delta \vec{V}_{rf} & \Delta u_{rf} & \Delta v_{rf} & \Delta w_{rf} & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

(2.160)

Substituting Eq. (2.159) into Eq. (2.15), results in

\[
\Delta Q^+ = F \Delta Q^+ \\
\Delta Q^+ = \frac{\partial Q^+}{\partial Q^+} \Delta Q^+
\]

(2.161)

In practice, the filter operation must be performed by the receiver because it is possible that multiple domains are providing \( Q^*_{Donor} \) (see Section 3.4.4). Therefore, the reduction in the communication achieved for the phase lag interface is not possible here.
Dependent Variable Case, \( W \neq Q \). Additional work is required to incorporate the \( W \rightarrow Q \) and \( Q \rightarrow W \) transformations into the formulation of the exterior Jacobian. This task is exacerbated by the complex communication process that is necessary to provide the receiver with the appropriate information. Because any number of donors can provide information for the receiver at any given point in space and time, the Jacobian information from each donor (including the phase shift) would need to be collected to form the entire Jacobian matrix. The communication costs could be as steep as \( \mathcal{O}(2\bar{N}\bar{N}) \) if there are \( \bar{N} \) donor domains each supplying \( 2\bar{N} \) update vectors to the receiver. To avoid this cost, it is assumed that the application of the relative motion operations to the dependent variables is an acceptable approximation. In this case, the exterior interface Jacobian with respect to the filtered variables is \( \frac{\partial w_c}{\partial W} = I \). A discussion of the results of this assumption are made in Section 3.4.5.

2.4 Conjugate Heat Transfer

Modeling the fluid-solid heat transfer with the CHT method requires the simulation of the fluid and solid domains. Two domains with a CHT interface are depicted in Fig. 2.6. The fluid and solid governing equations are the Navier-Stokes equations and the solid conduction equation, respectively (Section 2.2). Only the primitive variable fluid flow equations are considered here. Along the interface, \( \Sigma \), the following two conditions are enforced on both domains

\[
T_{\text{fluid}} = T_{\text{solid}} \\
(k\nabla T \cdot \vec{n})_{\text{fluid}} = (k\nabla T \cdot \vec{n})_{\text{solid}}
\]

The first imposes consistent temperatures, while the second requires equal heat flux normal to the interface boundary, \( \Sigma \). In addition, a no-slip condition is applied to the fluid side of the interface. As indicated in Eq. (2.8), the interface conditions are weakly imposed.

Because the CHT interface is an extension of the artificial boundaries, it shares the same properties related to conservation. Namely, the CHT interface is only conservative when the cells are coincident. Otherwise, some errors in the fluxes will occur across the CHT interface. These errors decrease with increased polynomial order and mesh resolution. A detailed discussion of the conservation properties of the artificial boundaries is given by Galbraith [101].

60
2.4.1 Fluid Domain Interface Conditions

For the fluid side of the interface, the notation is shown in Fig. 2.7a, where the \((\cdot)^-\) conditions are from the fluid side of the interface and the \((\cdot)^+\) conditions are from the solid side of the interface. Enforcing Eq. (2.162) and the no-slip condition gives

\[
W^+\Sigma = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -p^- \end{pmatrix}, \quad W^-\Sigma = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -p^- \end{pmatrix}, \quad T^+ = T^- 
\]

(2.164)

Because the interface acts as a wall for the fluid domain, the upwind term, \(\phi\), of the advective flux approximation is set to zero. From Eq. (2.163) the variable gradient vector is

\[
\nabla W^+\Sigma = \begin{pmatrix} \nabla u^- \\ \nabla v^- \\ \nabla w^- \\ \nabla p^- \\ -k_{solid}(T^-) \nabla T^- \\ \nabla \nu^- \end{pmatrix}. \quad (2.165)
\]

where \(k_s\) and \(k_f\) are the solid and fluid thermal conductivities, respectively. The internal variable gradient vector is \(\nabla W^-\Sigma = \nabla W^-\).
The fluid domain interface Jacobians are

\[
\begin{align*}
\frac{\partial W^+}{\partial W^+} &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}, & \quad \frac{\partial W^-}{\partial W^-} &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix} 
\end{align*}
\] (2.166)

\[\begin{align*}
\Omega^{\text{Solid}} \\
\Omega^{\text{Fluid}} \\
\end{align*}\]

![Diagram](image)

Figure 2.7: Notation for fluid and solid sides of the interface.

### 2.4.2 Solid Domain Interface Conditions

The notation for the solid domain is shown in Fig. 2.7b, where the \((\cdot)^-\) conditions are from the solid side of the interface and the \((\cdot)^+\) conditions are from the fluid side of the interface. From Eq. (2.162), the external interface variable vector is

\[
W^+ = \begin{pmatrix} T^+ \end{pmatrix}
\] (2.167)

By applying Eq. (2.163), the variable gradient vector is determined as

\[
\nabla W^+ = \begin{pmatrix} \frac{k_{\text{solid}}(T^-)}{k_{\text{fluid}}(T^-)} \nabla T^+ \end{pmatrix}
\] (2.168)
The internal interface conditions are simply $W^-_x = W^-$ and $\nabla W^-_x = \nabla W^-$ and the interface Jacobians are

$$\frac{\partial W^+_x}{\partial W^+} = [1], \quad \frac{\partial W^-_x}{\partial W^-} = [1]$$

(2.169)

### 2.4.3 Extension to Unsteady Fluid-Solid Heat Transfer with the Harmonic Balance Method

Simulating unsteady solid wall heat transfer when modeling turbomachinery is complicated by the vastly different time scales in the fluid and solid domains. This complexity is easily resolved by using a frequency domain method, like the HB method. An added advantage of the HB method is that the spatial discretization does not affect the treatment of the time derivative term. The interface conditions just described are immediately available for an unsteady simulation with the HB method. Each HB time level applies the CHT interface conditions as if they were separate steady simulations.
Chapter 3

Implementation

This section includes details of the implementation of the primitive variable RANS equations, the Harmonic Balance (HB) method, and the Conjugate Heat Transfer (CHT) method. A short summary of the Chimera communication, developed and described by Galbraith [101], is provided. The implementation of the S-A turbulence model with a modal representation is discussed. A description of the HB method is presented, including the formulation of the implicit HB linear system of equations and an investigation into the effectiveness of various preconditioning matrices. The communication strategy of the HB relative motion and phase lag interfaces is shown, along with an analysis of these interfaces. Lastly, a discussion of the CHT interface is presented, including a study of the “gap problem”.

3.1 Implicit Chimera Communication

The Chimera overset method was originally conceived by Benek et al. [118] to provide flexibility when generating structured meshes on complex geometries. A drawback of the traditional Chimera overset method is fringe points. These are parts of overlapping meshes that cannot be supplied with the appropriate information (because of insufficient overlap for a given stencil). In addition, the piecewise-linear representation of curved surfaces (for finite-volume and finite difference discretizations) can result in inconsistent representations along curved boundaries. These two issues were explored and solved with the development of the DG-Chimera scheme by Galbraith [101]. The small stencil of DG eliminates the need for fringe points and the high-order mesh can accurately represent curved boundaries in regions of mesh overlap.

A short summary of the DG-Chimera communication process is provided here. First, the overlapping
faces are seeded with Gauss Quadrature (GQ) nodes. Variables and variable gradients are supplied at these GQ nodes by donor domains. A donor is a computational domain that includes the GQ node within its extents. The necessary information is interpolated from a donor cell’s polynomial representation to any node within them. From the supplied nodal quantities, the receiver calculates the fluxes. The DG-Chimera approach allows for any amount of overlap, including abutting meshes. The only invalid mesh configuration is one in which gaps occur between the adjoining meshes. Galbraith has shown that the DG-Chimera scheme maintains two important properties. First, it is numerically consistent as mesh refinement increases. Second, a simulation utilizing DG-Chimera overlap exhibits the same order of accuracy as a comparable simulation without overlap.

For this work, the DG-Chimera capabilities have been expanded to include the S-A turbulence model (Section 3.2), the CHT interface (Section 3.5.1), the HB phase lag interface (Section 3.4.3), and the HB relative motion interface (Section 3.4.4).

### 3.2 Turbulence Model

The original S-A model is not generally robust when used with DG discretizations. Undershoots in regions of high \( \tilde{\nu} \) gradients, such as the edge of boundary layers and wakes, can result in negative \( \tilde{\nu} \). In addition, the modified vorticity, \( \tilde{Z} \), can become negative. In both cases, the stability of the model, when applied to a DG discretization, is significantly affected. The modified S-A model is proposed by Allmaras et al. [107] to allow negative \( \tilde{\nu} \) and \( \tilde{Z} \) while maintaining the original character of the S-A model for positive \( \tilde{\nu} \) and \( \tilde{Z} \). However, other modifications are necessary. When calculating the linearization of the fluxes, \( C^1 \) continuity is vital. With the S-A model, the derivatives of \( r \) and \( Z \) must be made continuous. This is accomplished with Eqs. (2.51) and (2.45).

Another complication arises when using a modal polynomial formulation. The modal formulation relies on an analytical quadrature-free integration to perform operations between polynomials, such as polynomial multiplication. Unfortunately, this integration is untenable for non-polynomial operations, (i.e. square root, min/max, trigonometric functions) [101]. Instead, the traditional numerical Gaussian quadrature integration is used for these operations. For the S-A model, many of the equations are piece-wise functions that rely on checking the value of \( \tilde{\nu} \) or \( \tilde{Z} \). Much like the non-polynomial operations, these piece-wise functions cannot be easily evaluated using modal polynomials. Therefore, the calculations for the S-A model are conducted
using a nodal representation. First, the dependent variables and gradients are evaluated at the quadrature nodes. Then, the S-A calculations are performed on each node independently. Once the fluxes and source terms are obtained at the quadrature nodes, they are converted back to the modal representation. Finally, the fluxes and source terms are integrated analytically. The conversion to and from the nodal representation is explained by Galbraith [101]. The errors caused by the conversions are on the order of round off error. Hence, the use of the nodal representation for the S-A model is not expected to negatively impact the simulation accuracy.

### 3.2.1 Wall Distance Calculation

For the S-A model, a calculation of the minimum wall distance is required. For complex geometries this is not a trivial exercise. Efficient and robust methods for finding the wall distance are an active area of research.

The simplest and most accurate approach is to calculate the distance from each point in the domain to every wall, and then find the minimum distance. However, this brute force method incurs a significant computational cost. To reduce the cost, inaccurate measures are sometimes used. For instance, calculating distances along mesh lines, considering only the wall mesh nodes rather than the entire wall surface, and considering only the walls in the domain being searched.

More elegant and efficient methods have been derived from partial differential equations (PDEs) that govern the wall distance solution. One such PDE is the Hamilton-Jacobi equation. Osher and Sethian [119] applied this equation to solve propagating fronts whose speed is related to the wall curvature. Later, Sethian used the Eikonal equation (as subset of Hamilton-Jacobi) to develop the fast marching level set method for calculating the signed distance function [120]. The wall distance can also be calculated from the gradient of a Poisson equation solution, as described by Tucker [121]. These PDE methods are significantly faster, though are more complex to implement and less accurate than the search procedure. The distance errors are generally on the order of 1% of the search method. However, the largest errors are further from the wall where the effect of the error on the solution is diminished.

A sweeping method for solving the Eikonal equation with a local DG solver has been developed by Li et al. [122]. However, a global DG discretization of the aforementioned PDE equations has not been developed. Additionally, any discontinuities in the wall distance would produce errors when modeling the turbulence.
Ultimately, a search procedure was chosen to reduce the complexity of the implementation. Though the cost of searching is high, the wall distance calculation is only performed once per mesh. Also, the cost of the search is reduced in two ways. First, only the nearest faces are chosen for comparison with each domain. Second, the minimum distance search procedure is only performed on the wall faces adjacent to the face endpoint with the minimum distance.

Prior to the search, the walls nearest to the domain in question are gathered. An example scenario is depicted in Fig. 3.1. The searching mesh is designated by the blue curves. Several wall faces are shown as green and red curves. The dashed black boxes indicate the maximum and minimum extent of the mesh and each face. The maximum distance from the mesh bounding box to the nearest face bounding box is indicated with an orange arrow. This distance provides a bound (marked as an orange circle) for including and excluding faces for the search procedure of the blue domain. Any face whose bounding box is within or intersected by the orange circle is considered valid (green curves). The red faces that lie outside of the orange circle are ignored. This process is performed for all domains independently.

Next, consider the search process for the blue domain. The nodes of the this mesh are shown in Fig. 3.2, with the current node having an orange center. These nodes are the same mesh nodes that are agglomerated to form the mesh cells (see Galbraith [101]). The searching node first calculates the distance to each green wall face endpoint (green circles). The orange arrow points to the face endpoint that is closest to the current mesh node. The wall faces adjacent to this endpoint are then searched with the Bisection method to find the
minimum distance. The Bisection method was chosen because it is robust and bounded. This search relies on the function being smooth and having only one minimum. Though this is not guaranteed, generally the discretization of any surface to be modeled satisfies these conditions. Otherwise, the simulation itself would suffer from the non-smooth surface behavior.

The wall distance is calculated with this method on several geometries, as shown in Fig 3.3. For the shapes case, the walls are designated as the lower pink boundary, the lower left black boundary, the lower right red boundary and the interior boundary of the cylindrical domain. The surface of the airfoil and cylinders are assigned as walls. The wall distance fields are derived from the modal polynomial representation of the wall distance. As mentioned previously, the mesh nodes are agglomerated to form the polynomial. Therefore, the modal field is used by the turbulence model. From the wall distance contours, it is evident that algorithm successfully calculates the wall distance. Both the airfoil and cylinder cases also utilize the hole cutting developed by Galbraith [101]. One concern is that a negative distance will be calculated for nodes near walls that are used to cut holes in other domains. This could be caused by the agglomeration process for relative large cells near a cutting surface. Negative values could cause the modal representation to cross zero within the simulated cells. The S-A utilizes the inverse of the wall distance, therefore this occurrence could be disastrous.

To explore the possibility of a negative wall distance, the airfoil distance field was further interrogated and negative values were found. As expected, they occur near a hole cutting surface. Figure 3.4 shows distance contours of the flap domain near the airfoil trailing edge. The red regions near the airfoil surface are areas of negative wall distance. The light blue region bounded by the bold lines indicates the area of
Figure 3.3: Contours of wall distance for several geometries.
Figure 3.4: Contours of wall distance for the flap domain. Magnification of the region near the trailing edge of the airfoil. Regions of negative distance are shown in red. The bounded light blue region is blanked by the hole cutting process.

excluded cells. In other words, all of the negative values are within the hole created by the airfoil cutting process. And, these regions appear to be produced by large cells near the cutting surface. This example is by no means a rigorous proof that negative values will not occur within the simulated regions of the domain. However, the conditions that would cause negative distance (i.e. large cells near a wall) are not expected to exist outside of a cut region.

Although the search procedure is slow, it is effective at accurately and robustly calculating the wall distance. Several steps could be taken to reduce the cost. In the future, the process could be made parallel. After gathering faces, the procedure is embarrassingly parallel and would see significant performance gains. As it stands, the search procedure is adequate as a pre-processing step. However, if it were required more often (with moving meshes, for example), then a more efficient PDE approach would be necessary. Additional work is needed to extend the search to three dimensions. The difficulty is finding a robust and accurate minimization method for the distance between a mesh node and a two-dimensional surface. It is likely that gradient-based methods would not be robust enough. Another complication is the need to bound the search. One potential candidate is the Nelder-Mead method [123], though more research is required.

3.3 Steady Solution Advancement

For most complex problems, simulations using Newton’s method (as described in Section 2.1.1.1) diverge. To improve stability, a Quasi-Newton method is employed, which involves the addition of a pseudo-time
term to the linear system. The drawback of this method is the loss of quadratic convergence.

Consider the steady form (i.e. $\frac{\partial}{\partial t} = 0$) of Eq. (2.1), which is

$$\nabla \cdot \bar{F}(W, \nabla W) + S(W, \nabla W) = 0$$  \hspace{1cm} (3.1)

a pseudo-time term is then added

$$\frac{\partial Q(W)}{\partial \tau} + \nabla \cdot \bar{F}(W, \nabla W) + S(W, \nabla W) = 0$$  \hspace{1cm} (3.2)

where $\tau$ is pseudo-time. A weak form of Eq. (3.2) is discretized as described in Section 2.1.1, producing

$$\int_{\Omega_e} \psi \frac{\partial Q(W)}{\partial \tau} d\Omega + \mathcal{R}(W) = 0$$  \hspace{1cm} (3.3)

The Quasi-Newton method is obtained by applying a generalized Newton’s formula to Eq. (3.3), resulting in

$$\int_{\Omega_e} \psi \frac{\Delta Q^m}{\Delta t} d\Omega + \frac{\partial \mathcal{R}(W^m)}{\partial W} \Delta W^m = -\mathcal{R}(W^m)$$

and substituting $\Delta Q^m = \frac{\partial Q(W^m)}{\partial W} \Delta W^m$ gives

$$\left[ \int_{\Omega_e} \psi \frac{1}{\Delta t} \frac{\partial Q(W^m)}{\partial W} d\Omega + \frac{\partial \mathcal{R}(W^m)}{\partial W} \right] \Delta W^m = -\mathcal{R}(W^m)$$  \hspace{1cm} (3.4)

The pseudo-time term limits the local time step as

$$\Delta t^m = \frac{d}{\lambda_e} \sqrt{\frac{\Omega_e}{\Delta t_e}} CFL^m$$  \hspace{1cm} (3.5)

where $d$ is the dimension, $\lambda_e$ is the cell mean characteristic speed, and $d \sqrt{\Omega_e}$ is representative of the cell spacing. Applying this time step limit improves the matrix conditioning by increasing the magnitude diagonal elements. Control over the size of the time step is achieved by modifying the CFL number. In this work, two methods have been used. The first simply increases $CFL$ by a constant factor, $f$, after each Quasi-Newton iteration

$$CFL^{m+1} = f \times CFL^m$$  \hspace{1cm} (3.6)
The second method, by Orkwis and McRae [103], calculates $CFL$ using a ratio of the initial to current residual, as given by the following

$$CFL^{m+1} = CFL^0 \frac{\mathcal{R}(W^0)}{\mathcal{R}(W^m)}$$

(3.7)

$CFL^0$ is the initial $CFL$ value specified by the user, which is set to 1 for most of the simulations in this dissertation. Using the method in Eq. 3.7, $CFL \to \infty$ as the solution converges to a steady state. As this occurs, the pseudo-time term approaches zero, Eq. (3.4) approaches Newton’s method, and quadratic convergence is recovered.

The linear system is solved for each Quasi-Newton iteration using a flexible version of the General Minimum Residual algorithm, FGMRES[124]. For the verifications in Section 4.1, the system is preconditioned with an Incomplete Lower Upper decomposition with one level of fill in (ILU(1))[124]. The solution update is $W^{m+1} = W^m + \Delta W^m$. Derivations of the ILU(0), ILU(1), and the Lower Upper Symmetric Gauss-Seidel (LU-SGS) preconditioners, as implemented in the DG-Chimera framework, are provided by Galbraith [101].

For any simulation, the convergence of the FGMRES algorithm is guaranteed only if the number of iterations is equal to the number of degrees of freedom of the system. The FGMRES iterations are limited to a number well below the degrees of freedom to make the computational cost manageable. Therefore, convergence is not guaranteed. In practice, simulations with stiff linear systems will not converge if $CFL$ increases too quickly. This can occur when increasing $CFL$ with Eq. (3.6) or (3.7). If the linear solve does not converge to a low residual, then the solution update vector is not accurate. This can cause the simulation to move more slowly towards, or even away from, the correct solution. To avoid this problem, another method for advancing $CFL$ is made available. The $CFL$ number is increased or decreased depending on the number of FGMRES iterations required to solve the linear system.

Consider the schematic in Fig. 3.5, where $n^m$ is the number of FGMRES iterations to solve Eq. (3.4), within a given tolerance. Suppose that the number of iterations should be kept near $n_d$. Rather than simply limiting the FGMRES algorithm to $n_d$ iterations (regardless of the state of convergence), the $CFL$ number is modified such that $n^{m+1}$ is driven towards $n_d$. This is accomplished with the following equation

$$CFL^{m+1} = (f_n)^{\text{sign}(n_d-n^m)} \ast CFL^m$$

(3.8)

such that $CFL$ is decreased by a factor of $f_n$ if $n^m > n_{mid}$ and increased by a factor of $f_n$ if $n^m < n_{mid}$. To
prevent large changes in $CFL$, the magnitude of $f_n$ is limited, as shown in the equation below

\[
f_n = 1 + (f - 1) \cdot \min\left( \frac{|n_d - n^m|}{\Delta n_{\text{limit}}}, 1 \right)
\]

(3.9)

where $f$ is some constant factor and $\Delta n_{\text{limit}}$ sets the location past which $f_n = f$. Notice that $f_n$ varies linearly from $f_n = 1$ at $n_d$ to $f_n = f$ at $|n_d - n^m| = \Delta n_{\text{limit}}$. This linear variation mitigates the oscillatory behavior of $CFL$ when $n^m$ is near $n_d$.

### 3.4 Harmonic Balance Method

One advantage of the HB method is the ease of adding it to an existing solver, as compared to other FD methods. When implementing the NLH and NLFD methods, the mean and perturbation equations must be defined separately. Conversely, the HB method handles each equation in the same way. To illustrate this point, schematics of an existing and HB solver framework are shown in Fig. 3.6. There are two modifications of the existing solver that are needed to implement the HB method. First, the pseudo-spectral operator is added to the temporal integration step. Second, $N$ replications of the spatial operator and solution are needed, one for each time level. Although this description is an over-simplification, it emphasizes that the solver procedures are largely unchanged.

The DG-Chimera framework proves to be ideally suited to adding the HB method. A modular design allows for the addition of the HB temporal integration without affecting the existing procedures. In addition, the spatial operator and solution hierarchy is easily replicated. The modular design also allows each individual spatial operator, of the HB solver, to apply the fluxes and boundary conditions independently. Though not part of the schematic, the linear system solver does not distinguish between the system in Eq. (2.11) versus the one in Eq. (2.100).

The process for using the HB method is depicted in Fig. 3.7. First, the interior fluxes, boundary con-
Conditions and source terms are calculated at all \( N \) time levels. During this part of the process, no interaction is needed between the time levels. Next, communication between domains and time levels is performed as needed. At artificial boundaries, a given time level only “talks” to the corresponding time level in an adjoining domain. On the other hand, the phase lag and relative motion interfaces require interaction between the time levels. Details for these two interfaces are provided in Sections 3.4.3 and 3.4.4. The integration of the pseudo-spectral term and the pseudo-time term are added. The last step is the solve of the linear system (see Section 3.4.1). A loop over this process is carried out until the simulation converges.

In this section, details of the HB temporal integration are given, including an examination of the linearization of the pseudo-spectral terms. Verification tests of the HB method are presented. Also, implementation of the HB phase lag and relative motion interfaces is discussed and an accompanying study of these interfaces is provided.
3.4.1 Solution Advancement

Although the HB simulation produces an unsteady result, the equations are advanced like a steady-state simulation. As with the steady-state solution advancement (Section 3.3), a pseudo-time term is added to the HB governing equation, which is common practice for HB methods [43, 44, 45, 46, 47, 48, 125]. With the addition of the pseudo-time term, Eq. (2.95) becomes

\[
\frac{\partial Q^*}{\partial \tau} + DQ^* + \nabla \cdot F^* + S^* = 0
\]  

(3.10)

which is transformed into weak integral form and discretized, resulting in

\[
\int_{\Omega_e} \psi \frac{\partial Q^*}{\partial \tau} d\Omega + \int_{\Omega_e} \psi DQ^* d\Omega_e + R^* = 0
\]  

(3.11)

As in Section 3.3, Newton’s generalized formula is applied to Eq. (3.11), which produces the following equation

\[
\int_{\Omega_e} \psi \frac{\Delta Q^*}{\Delta t_e} d\Omega + \left( \int_{\Omega_e} \psi D \frac{\partial Q^*}{\partial W^*} d\Omega_e + \frac{\partial R^*}{\partial W^*} \right) \Delta W^* = -(D^* + R^*) = 0
\]  

(3.12)

where \( \Delta Q^* = \begin{bmatrix} \Delta Q_{1m}^* & \Delta Q_{2m}^* & \ldots & \Delta Q_{Nm}^* \end{bmatrix} \). Substituting \( \Delta Q^* = \frac{\partial Q^*}{\partial W^*} \Delta W^* \) into Eq. (3.12) gives

\[
\left( \int_{\Omega_e} \psi D \frac{\partial Q^*}{\partial W^*} d\Omega_e + \left[ \int_{\Omega_e} \psi \frac{1}{\Delta t_e} \frac{\partial Q^*}{\partial W^*} d\Omega_e + \frac{\partial R^*}{\partial W^*} \right] \right) \Delta W^* = -(D^* + R^*)
\]  

(3.13)

The spatial linearization and pseudo-time terms are

\[
\int_{\Omega_e} \psi \frac{1}{\Delta t_e} \frac{\partial Q^*}{\partial W^*} d\Omega_e + \frac{\partial R^*}{\partial W^*} = \begin{bmatrix} A_{1m}^* & 0 & \cdots & 0 \\ 0 & A_{2m}^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_{Nm}^* \end{bmatrix}
\]  

(3.14)

where \( A_{nm}^* \) is equivalent to the steady linearization (see the left-hand side of Eq.(3.4)) for the \( n^{th} \) solution and \( \Delta t_e \) is given in Eq. (3.5). The CFL number can be advanced in the ways described in Section 3.3.

As with steady simulations, the HB linear system is solved during each Quasi-Newton iteration using
a flexible version of the General Minimum Residual algorithm, FGMRES[124]. For the verifications in this Section 4.2, the system is preconditioned with either an HB version of Incomplete Lower Upper decomposition with zero fill in (HB ILU(0), Section C.2) or an HB version of the Lower Upper Symmetric Gauss-Seidel (HB LU-SGS, Section C.1). An analysis of the different preconditioning methods for the HB linear system is given in Section 3.4.2.

### 3.4.1.1 Matrix Storage

From Eq. (3.14), it is clear that the storage of the spatial flux and pseudo-time linearization is $O(N)$, where $N$ is the number of time levels. On the other hand, the storage of the pseudo-spectral linearization is $O(N^2)$ because the $D$ matrix is dense. This steep storage cost can be avoided by realizing that the pseudo-spectral matrix is independent of the mesh. The pseudo-spectral matrix can be removed from the volume integral

$$
\int_{\Omega_e} \psi D \frac{\partial Q^*}{\partial W} d\Omega_e = D \int_{\Omega_e} \psi \frac{\partial Q^*}{\partial W} d\Omega_e
$$

(3.15)

where $\frac{\partial Q^*}{\partial W}$ is a diagonal matrix, as shown in Eq. (2.103). Only the volume integrals of the $\frac{\partial Q^*}{\partial W}$ linearization are saved, reducing the storage to $O(N)$, which is in line with the spatial linearization cost. During the linear solve, the pseudo-spectral contributions are calculated for each FGMRES matrix-vector operation by multiplying by the appropriate element of $D$.

### 3.4.2 Linear System Preconditioning

Preconditioning the HB linear system poses a challenge because it has a different topology than the system described in Section 3.3. Namely, the HB linear system has multiple spatial flux linearizations and a pseudo-spectral linearization. Derivations of the preconditioning matrices for the HB linear system of equations are given in Appendix C. Two main paths are discussed. The less complex approach is to apply an existing preconditioning algorithm to each spatial linearization, as shown in Eq. (C.6). These will be referred to as the “spatial-only” preconditioners. This type of preconditioner has been implemented using the Lower Upper Symmetric Gaussian-Seidel (LU-SGS)[126] preconditioner and the Incomplete Lower Upper decomposition preconditioner with zero and one level of fill in (ILU(0) and ILU(1), respectively). The other approach is to include the pseudo-spectral linearization when forming the preconditioning matrix, as discussed in Sections C.1 and C.2. These are the HB LU-SGS and HB ILU(0) preconditioners.
A study of the effectiveness of the aforementioned preconditioners is performed using a simple domain with a wake passing through it, similar to the cases described in Section 3.4.5. A constant $CFL$ of 100 is used and the FGMRES residuals are converged below a tolerance of $1.0 \times 10^{-10}$. The wake is simulated with polynomials of order $q = 0, 1, 2, 3, 4$ and with HB frequencies of $K = 1, 2, 3, 4$. Because this study only concerns the effect of the preconditioner on the linear solve, the simulations are performed for 10 Quasi-Newton iterations each, and are not converged. The number of FGMRES iterations required to converge the linear solve are plotted for each preconditioner in Fig. 3.8.

As expected, the HB ILU(0) and HB LU-SGS preconditioners perform better than the spatial-only preconditioners. With one frequency, HB ILU(0) calculation requires roughly half the number of FGMRES iterations. As the number of frequencies increases, the gap between HB ILU(0) and the other preconditioners widens. For $K = 4$ and $q = 4$, FGMRES preconditioned with HB ILU(0) converges in an order of magnitude fewer iterations than FGMRES preconditioned with the spatial-only preconditioners. In Fig 3.9, the number of iterations has been averaged for each $q$ and $K$ and is plotted for each preconditioner. The averaged FGMRES iteration count increases more slowly for HB ILU(0), with respect to $K$, when compared to the other preconditioners. The HB ILU(0) trend is nearly linear with $K$, while the trend of the other preconditioners appears to be of higher order than linear. An odd result occurs for the spatial-only preconditioners. Besides $K = 1$, the less complex LU-SGS performs as well as, or better than, the two ILU preconditioners. Generally, ILU has better performance for non-HB fluid flow simulations. This suggests that the relative performance between spatial-only preconditioners for non-HB linear systems is not indicative of their performance when applied to the HB linear system.

Unlike the cases presented in Fig. 3.8, in practice, the number of FGMRES iterations is limited. This limitation is necessary because dot products are computed with the solutions of all previous FGMRES iterations. This operation causes each FGMRES iteration to take longer as the iteration count grows. In addition, storing the solution vectors for every iteration increases memory usage. Limiting the number of iterations means that the matrix solve is not guaranteed to converge to the desired tolerance. Therefore, the solve of the linear system could be less accurate than desired. For more complex cases, less accurate solves can degrade the solution and cause the simulation to diverge. The only remedy available, other than improving the preconditioner, is to decrease the $CFL$, thereby increasing the diagonal dominance of the matrix and allowing for faster FGMRES convergence. Hence, the maximum $CFL$ might be limited by the solve of the linear system, rather than the stability of the discretization.
Figure 3.8: Number of iterations to achieve FGMRES convergence.
Even with the preconditioning improvements provided by HB ILU(0), the FGMRES solve is still an issue. As will be discussed in Sections 4.2 and 5.2, simulating more complex cases and higher numbers of HB frequencies reduces the speed of the linear system solve. Figure 3.10 shows a graphical representation of the matrix defined by Eq. (2.108). The squares represent block matrices. The blue and green blocks depict the dense spatial flux linearizations with respect to $W^-$ and $W^+$, respectively. The red blocks are the pseudo-spectral linearizations, which are diagonal matrices. For $K = 1$, the spatial linearization blocks are labeled by time level and the HB blocks are labeled by the index of the pseudo-spectral matrix element that they represent. From $K = 1$ to $K = 2$ the number of off diagonals increases, which decreases the diagonal dominance. This reduction can be mitigated by decreasing $CFL$ (which increases the magnitude of the pseudo-time term added to the diagonal). However, lower $CFL$ increases the number of Quasi-Newton iterations required for the simulation to converge.

One possible method for improving the speed of the linear system solver involves rearranging the matrix. An inverse of the blue diagonal matrices is part of the formulation of all of the preconditioners mentioned previously. Instead of using increasingly complex algorithms (for instance, formulating HB ILU(1)), the matrix could be rearranged as shown in Fig. 3.11. Re-indexing the elements of the matrix from $i,j,k,n$...
Figure 3.10: Representation of the structure of the HB linearization matrix.

(a) $K = 1$

(b) $K = 2$

Figure 3.11: Representation of the rearranged HB linear system matrix.

(a) $K = 1$

(b) $K = 2$
to $n, i, j, k$ moves the time levels of each cell next to each other. Now, the combined red and blue blocks form a larger block diagonal (the blocks outlined with dashed lines). The transition from $K = 1$ to higher frequencies no longer adds off diagonal bands, though admittedly the size of the diagonal increases by $N^2$. In effect, the rearranged matrix mimics the topology of Eq. (C.18) in that off diagonals are only created for higher spatial dimensions (tri-, penta-, and hepta-diagonal for one, two, and three dimensions). By inverting the new diagonal, a more effective preconditioner could be formed with the same algorithms used for the steady linear system. The drawbacks are the complexity of the matrix rearrangement and the cost of inverting the larger diagonal blocks.

It is clear from this study that the use of more sophisticated techniques for solving the HB linear systems shows promise. Although the re-indexing of the matrix can be stated simply, it would require significant restructuring of the linear system storage. For this reason, the implementation and study of the rearranged HB system of equations is deferred to future development.

### 3.4.3 Phase Lag Interface

Within the DG-Chimera framework, the simulation of periodic boundaries is accomplished by offsetting the face by the distance of the periodicity [101]. This scenario is depicted in Fig. 3.12. Both the red and blue faces are copied and offset by the periodic distance. These faces then receive information from the cells that they overlap or adjoin. The phase lag interface is achieved in the same way. With the user specified HB diameters and the distance the face is offset, the entire phase lag operation is defined (see Section 2.3.2.2). The negative of the receiver offset distance must be used when forming the phase shift operator because it is the donor solution that is phase shifted. In other words, the donor domain supplies the variables as if it were offset to the position of the original face (which is opposite the direction of $\Delta \vec{d}$ in Fig. 3.12).

Either the donor domain or receiver face could perform the phase lag operation. Ultimately, the donor
3.4.4 Relative Motion Interface

As mentioned in Section 2.3.2.3, the two methods that have been used along the HB relative motion interface involve either the exchange of Fourier coefficients or the use of a filter. For the Fourier method, a spatial Fourier Transform is required along both side of the interfaces. The resulting coefficients are then exchanged and non-reflection conditions are imposed. With the filter method, more time levels than represented by the receiver domain are passed across the interface. Then, a filter operation is performed to remove higher frequencies and prevent aliasing. Ultimately, the filter method was chosen because it can be performed on each face independently. The filter operation does not need information from all of the cells along the donor side of the interface. This lends itself well to the DG-Chimera approach, whereby the donor cells send information to each face independently. With the Fourier method, all of the cells on the donor side would need to communicate, which would require an additional mechanism for communication not currently available in the DG-Chimera framework. The Fourier Transform is much like an all-gather operation, which is computationally costly. One final consideration is that the filter method can calculate the spatial Jacobians independently, whereas the linearization of the Fourier method would be more difficult to compute.

3.4.4.1 Connection Procedure

Prior to any simulation, a procedure is necessary to determine which domains will supply information to which receiver faces. This process is described in detail by Galbraith [101]. For the relative motion interface,
Figure 3.14: Green receiver faces generated at several instances in time. Blue filled circles indicate nodes that receive information from the blue donor domain. Green filled nodes have a donor. Red filled circles indicate orphan nodes.

an extension of this process is required to capture the motion of computational domains. First, Consider the two domains depicted in Fig. 3.13. The blue domain is stationary and the green domain is moving with speed $\vec{\omega}$. These domains are periodic in the $\theta$ direction. For now, let the blue domain be the donor and green domain be the receiver. The receiver requires information from the donor at a number of times steps as it moves past the donor. The receiver face is generated at each receiver time level, as depicted in Fig. 3.14. Because the green face is moving relative to the blue domain, the faces are offset in the pitch direction by an amount equal to $\vec{\omega} \Delta t$. A total distance of $\Delta \theta$ is traversed over one period of the motion. In Fig. 3.14, only the right side of the donor domain (which abuts the receiver face) is depicted for $t_2, \cdots, t_5$. On each face, a number of Gauss Quadrature (GQ) nodes exist. The number of GQ nodes depends on the number of cells and the solution order of accuracy. Each GQ node must be supplied with information from the donor, as are the green filled circles in Fig. 3.14. Because of the motion of the green domain, some nodes do not intersect the donor domain. These red nodes are orphans, i.e. they are not provided information from any donor.

However, the periodic nature of the domains allows the orphans to be supplied by the neighbors of the blue domain. By reconstructing blue domains offset in $\theta$ from the first, as seen in Fig. 3.15a, the previously orphaned GQ nodes are now intersecting a domain. This reconstruction process can be achieved by using the HB phase shift operator. Unfortunately, reconstructing portions of the donor domain can be expensive computationally. Alternatively, a less costly process, already available for copying periodic faces, can be used. As shown in Fig. 3.15b, the receiver face is copied and offset by the donor $\Delta \theta$. Individually, the
original and copied faces have orphan nodes, but added together, they form a face without orphans.

A similar process is carried out when the blue domain is the receiver and the green domain is the donor, as shown in Fig. 3.16. Again, the receiver faces are copied at each instant time. Note that they offset downwards as the motion of the blue domain relative to the green domain is $-\vec{\omega}$. In this case, more than one copy of the blue receiver face is required for some instances in time because $\Delta\theta_{green} > \Delta\theta_{blue}$. In practice, the location of the domains relative to each other is not known. Therefore, extra copies of the receiver face are made for robustness. The number of copies made in each direction is $floor\left(\frac{\Delta\theta_{rec}}{\Delta\theta_{donor}}\right) + 2$. This overhead of additional faces is only incurred during the initialization of a simulation. After the connection procedure is complete, any excess faces are discarded. If the domains are very far apart in the $\theta$ direction, such that orphans still occur, the user can specify an initial offset of the relative motion faces.

### 3.4.4.2 Phase Shift and Filter

At this point, the receiver faces are provided with donors. During the simulation, the phase shift and filtering processes (see Section 2.3.2.3) are applied to the donated variables. First, the donor cell generates the solution at the given receiver face. If the donation is occurring with a copy, the donor cell applies the phase shift operation. As with the phase lag interface, the negative of the receiver offset must be used when forming the phase shift operator. This shift is performed by the donor because the receiver nodes are at different locations in space at each time. The receiver then gathers the faces at each time level and merges them into one face. Then, the filter operation is performed on the merged mesh faces. The relative motion equations (such as the linear cascade equations in Section 2.3.2.3) are applied to the filtered variables. In accordance with all artificial boundaries, the interface Jacobians are not stored directly in the matrix, but rather are applied to the linear solve as matrix vector multiplications.

### 3.4.4.3 Current Limitation

Although the Chimera scheme allows overlap, in the case of the relative motion interface, it is not valid for the current implementation within the DG-Chimera framework. During initialization, all faces that require information are sent to all domains, at which point the domains decide if they can donate to any of the faces. This includes all periodic and relative motion faces. Suppose the blue and green domains are overlapped, as shown in Fig. 3.17. The region of upper periodic face of the green domain (in red) overlaps the blue domain. Because this face is sent to all domains, the blue domain would donate to the red region of the periodic face,
Figure 3.15: Two methods to fill orphaned nodes of relative motion faces. Copies of the receiver face are indicated by dashed lines.
Figure 3.16: Blue receiver faces and copies at several time steps. More than one copy is needed per time step to find a donor for all nodes.

Figure 3.17: Two overlapping domains moving relative to each other.

even though this is not the expectation of the user. The user intends for the blue domain to donate only to the relative motion face. Instead, the periodic face would receive incorrect information, corrupting the solution. Therefore, simulations with overlapping domains moving relative to each other are discouraged. With further development, this limitation could be lifted.

3.4.5 Harmonic Balance Interface Study

A simple test case is carried out to study two aspects of the phase lag and relative motion interfaces. First, the accuracy of the solution transferred is examined, including a comparison of solutions with different polynomial orders and numbers of frequencies. This aspect is investigated in two and three dimensions. The second goal is to understand the effect of the primitive variable approximation discussed in Sections 2.3.2.2
A description of the two dimensional test case is depicted in Fig. 3.18. The test consists of two domains. The first is stationary and the second moves past the first in the y-direction. The relative motion interface is applied between the two domains. The phase lag interface is imposed on the top and bottom of each domain. The extent of the two domains in the y-direction are $0.4L_{ref}$ and $0.29L_{ref}$, respectively. A Reynolds number of 500 is used to calculate $L_{ref}$. The stationary and moving domains have lengths of $L_{ref}$ and $2L_{ref}$, respectively. The velocity of the moving domain is $-0.39u_{ref}$, where $u_{ref}$ is calculated from the speed of sound and a Mach number of 0.1.

Freestream pressure of $p_{\infty} = 101.35\text{kPa}$ and temperature of $T_{\infty} = 293\text{K}$ are assumed. Total conditions are imposed on the left boundary of the stationary domain with an inlet angle of 10 degrees. A constant $P_T$ is calculated from the freestream pressure and the mach number. A total temperature variation is imposed as

$$T_T(y) = T_{T\infty} \left(1 + 0.1 \sum_k \sin \left(2\pi k \frac{y}{0.4L_{ref}} \right) \right)$$

where $n_{sum} = 3$ so that higher frequency content is present. Notice that the variation is steady with respect to the stationary domain; however, the variation imposes unsteady conditions on the second domain because of the prescribed motion, $V_{ref}$. Static pressure is specified at the right boundary of the second domain. The first and second domains are discretized uniformly with 12x12 and 24x9 cells, respectively.

This test case is simulated with both the conservative and primitive variables as the dependent variables. The laminar Navier-Stokes equations are used for these simulations. A series of cases are completed with $q = 0, 1, 2, 3, 4$ and $K = 1, 2, 3, 4$. The HB frequencies are the fundamental passing frequency, Eq. (2.115),
and higher harmonics. The HB diameters are these frequencies divided by the relative domain speed. An initial $CFL$ of 1 is specified and then ramped by the residual ratio (Eq. (3.7)). The convergence histories are shown in Fig. 3.19. The conservative simulations achieve convergence in approximately one third the time. By investigating Fig. 3.19c, the difference is clear. The primitive variables convergence is slower initially and then the slope flattens at about 60 iterations. On the other hand, the conservative variables approach achieves Newton convergence as $CFL \to \infty$. The root cause of the slow convergence is the primitive variable approximation of the HB operators, discussed in Sections 2.3.2.2 and 2.3.2.3.

The next step is to determine the effect of this approximation on the solution. Instantaneous contour
lines of the temperature field are shown in Fig. 3.20. Several of the stationary and moving domains are reconstructed for this instant in time to improve the visualization of the solution. Looking at the contour line comparison, it is clear that the primitive variables simulation produces the same solution as the conservative variables simulation. Although the primitive variable approximation degrades the convergence, it does not appear to decrease the solution accuracy. Overlaying the relative vectors on the solution, as in Fig. 3.21, shows that the transition from stationary to moving frames of reference is achieved.

Lastly, the effect of different orders of accuracy and simulated frequencies is studied. Figure 3.22 presents the temperature solution at an instant in time on the stationary and moving sides of the interface.
Figure 3.22: Temperature along both sides of the interface at an instant in time. Several combinations of solution polynomial order and numbers of frequencies are shown. Primitive variables solutions.
for different $q$ and $K$. Given that these are at the same location the curves should be identical. The $K = 1$ comparison exhibits a phase shift of the temperature signal. For the $q = 1, K = 5$ solution, the position of the temperature curve is captured, but the spatial resolution is not sufficient to accurately capture the curve. The curves for the highest spatial and spectral accuracy achieve the desired result of an indistinguishable solution on either side of the interface. This provides confidence that the interface accurately passes the information between the blade rows given a sufficient spatial and spectral resolution.

3.4.5.2 3D

A similar domain is used for the test case in three dimensions. The domain is extruded from $-0.25L_{ref}$ to $0.25L_{ref}$ in the $z$-direction. For this case, $M_{inlet} = 0.3$ and $Re = 5 \times 10^4$ is used to calculate $L_{ref}$. The larger Reynolds number is applied to reduce the diffusion of the temperature variation. Adiabatic walls are imposed on the maximum and minimum boundaries in the $z$-direction. The same boundary conditions are imposed on the boundaries that existed for the two-dimensional case. The only change is the way in which the total temperature variation is imposed. A wrapped normal distribution is used instead of the sum of sinusoidal terms. Also, a variation is applied in the $z$-direction to ensure the interface can accurately capture three-dimensional variations. The final form of the total temperature equation is as follows

$$T_T(y, z) = T_{T\infty} \left( 1 + 0.1 \sum_{k=-n}^{n} \exp \left[ -\pi^2 \left( \frac{y}{0.4L_{ref}} - 0.5 + k + \cos \left( \frac{\pi}{0.25L_{ref}} \right) \right)^2 \right] \right)$$ (3.17)

where $n = 2$ provides sufficient terms to make the distribution periodic for the given domain.

As with the 2D case, the laminar Navier-Stokes equations are used. Only simulations with the primitive variables are computed. Simulations are conducted for several polynomial orders and HB frequencies. A plot of the convergence history for $K = 1$ is given in Fig. 3.23. As seen previously, the convergence is slowed because of the primitive variable approximation of the HB operators (see Sections 2.3.2.2 and 2.3.2.3). Despite the slow convergence, high order simulations are achieved in 350 iterations. Contours and an iso-surface of the temperature field are depicted in Fig. 3.24. The iso-surface passes through the interface without significant changes in amplitudes or phase of the variation.

A closer evaluation of the transfer of information across the interface can be made using the contour line comparison in Fig. 3.25. As with the 2D case, low spectral resolution results in a phase shift of the solution across the interface. The low spatial order reduces the resolution of the features. Comparing the
Figure 3.23: Convergence histories for $K = 1$.

Figure 3.24: Solution for $K = 3$, and $q = 2$. Inflow bottom left, outflow top right.
Figure 3.25: Temperature contour lines on the stationary and moving sides of the interface at an instant in time. Several combinations of solution polynomial order and numbers of frequencies are depicted. Primitive variables solutions. Green lines indicate the edge of the stationary domain. Two moving domains shown.
\( q = 2 \) solutions, fewer frequencies not only causes a phase shift, but also changes the shape of the features in the central temperature band. Differences near the upper and lower \( z \)-direction boundary can be attributed to the singularity of the wall boundary condition at the interface. In other words, the flow changes from a stationary to a moving wall at the interface. Besides this expected inconsistency, the the interface is shown to accurately resolve the temperature variation as it passes from one reference frame to the other.

### 3.5 Conjugate Heat Transfer

#### 3.5.1 Chimera Boundaries

The Chimera communication described in Section 3.1 allows significant flexibility for mesh generation. Any amount of overlap is acceptable, including abutting meshes. The only restriction is that there cannot be gaps between meshes. When formulating an interface between domains with different equations sets, other restrictions arise. Consider the blue and green domains depicted in Fig. 3.26. Suppose the blue domains employ the fluid flow equations and the green domains use the solid conduction equation. The configurations in Figs. 3.26a and 3.26b are valid configurations. However, the domains in Fig. 3.26c cause a conflict. The left face of the cell marked with a red \( X \) is next to both the fluid and solid domain. Therefore, that face cannot appropriately apply the CHT interface because only half of it communicates with a solid domain. For this reason, any face that has donor cells of more than one equation set is considered an orphan.

Another interesting case arises when overlapping the solid and fluid domains, as seen in Fig. 3.26d. Although the connection is valid, the physical meaning is dubious. The solid domain would be receiving temperature information from somewhere above the fluid wall. Also, the fluid domain would receive temperature information from inside the solid. Unfortunately, attempting to denote nodes of the overlapped face as orphans poses other problems. The Chimera connection process would require additional complexity to determine whether the face is abutting. This would be possible, but as with most complex geometries, making exactly abutting meshes would be a challenge. Instead, some small allowable overlap would likely be required. However, defining what is “small” is always a challenge and redefining a tolerance based on the case being modeled causes additional problems. Ultimately, the overlapping CHT faces are considered valid from a connection stand point. A similar line of thought about complex geometries leads to the investigation of the “gap problem”, detailed in Section 3.5.2.

Given a valid amount of overlap, the interface conditions are applied. Generality is added to the Chimera
communication to allow for donors and receivers with different numbers of variables (as in the case of CHT). In addition, modular transformation classes are created to apply the appropriate conditions for a given donor and receiver equation type. This generality allows future interfaces to be added readily. In the case of CHT, described in Section 2.4, the interface conditions are applied by the receiver. The linearizations are not stored as a matrix, but rather applied as a matrix vector multiplication during the linear solve. For HB simulations, each domain applies the CHT interface conditions to each time level independently.

### 3.5.2 Gap Study

As mentioned previously, overlapping the solid and fluid domains for a CHT simulation is valid. However, this overset topology will certainly introduce errors. The goal of this study is to gain understanding of the amount of overlap required to satisfy the DG-Chimera connection algorithm. The errors introduced by the overlap will not be discussed as they could vary greatly depending on the problem.

Two cases are investigated. The first is a two-dimensional circular arc. This case will provide insight into the overlap necessary for non-polynomial shapes. The second case is a three-dimensional polynomial shape. This case will validate the connection process when the geometry and mesh polynomial orders are equivalent. Also, it will give insight into the overlap required when the mesh polynomial under-resolves the
polynomial shape. In both cases, a series of meshes are tested where the number of cells are varied on the solid and fluid domains.

### 3.5.2.1 2D Circular Arc

The circular arc is a two-dimensional test case and is representative of a duct flow with a 90 degree turn. The solid domain is placed along the inner radius of the fluid domain. Figure 3.27 illustrates four examples of the domain topology. The number of cells in the fluid (blue) and solid (green) domains range from 2 to 10. Quartic polynomials \((m = 4)\) are used to represent the mesh. Because the purpose of this exercise is to test the connection process, the simulation is conducted for one iteration. If the simulation completes, then the mesh passes. If orphans are found, then the mesh fails. When a mesh fails, the two domains are scaled radially to overlap by a small amount. This process is repeated and the overlap is increased until the mesh passes. For each failed check, the overlap is increased by an order of magnitude.

The required overlap is displayed by the contour plot shown in Fig. 3.28. The values of overlap are based
on the height of the fluid domain \((H_{\text{fluid}})\). Therefore, a \(\log_{10}(\text{overlap})\) of \(-8\) means that the domains overlap by \(H_{\text{fluid}} \times 10^{-8}\). As a guide, the four meshes in Fig. 3.27 represent the meshes at the four corners of the contour plot. As expected, the coincident domain topologies do not require any overlap. This is because the polynomial mesh represents the non-polynomial shape in the same way. It is also not surprising that the needed overlap increases as the disparity in cell counts increases. In general, lower resolution in the outer mesh (fluid) necessitates more overlap than when the inner mesh has fewer cells. This is indicated by higher values of overlap above the diagonal than below. A significant conclusion is that overlap is needed along non-polynomial boundaries unless the cells on either side are coincident. When the resolutions are similar, the needed overlap is very small, though even this small overlap might cause unacceptable errors.

3.5.2.2 3D Polynomial Shape

A similar exercise is carried out on a three-dimensional polynomial shape. The interface surface of the two domains is modeled by the following equation

\[
z(x, y) = 0.1 \left( 10x^3 + 4y^3 + 6x^2y - 5xy^2 + 3x^2 - x - 2y^2 + y + xy - 1 \right)
\]

A third order polynomial was chosen because a common method for generating complex geometries is the use of cubic splines. Four example meshes are depicted in Fig. 3.29. The number of cells in the
directions normal to the interface are varied from 2 to 10 in both blue and green domains. The meshes are constructed of linear ($m = 1$), quadratic ($m = 2$) and cubic polynomials ($m = 3$). As with the previous case, the simulations are stopped after the connection step is completed. When orphans are found, the overlap is increased until the orphans are eliminated.

The resulting contour plots of required overlap are given in Fig. 3.30. The $m = 3$ plot is notably absent. This is because all of the cases pass without overlap (making the entire plot white). For the other two polynomial representations, the overlap is relative to the average height of the blue and green domains ($H_{avg}$). Therefore, a $\log_{10}(\text{overlap})$ of $-3$ means that the domains overlap by $H_{avg} \times 10^{-3}$. In either case, the under resolved mesh surfaces do not match well. Besides the coincident cases, the required overlap is between one tenth and one thousandth of the domain height. This amount of overlap is certain to produce
Figure 3.30: Amount of overlap required for the connection process to pass for the 3D polynomial shape case. White regions indicate no overlap. The $m = 3$ plot is not shown because all cases pass without overlap.

large errors. Clearly, approximating the mesh with a polynomial of lower order than the geometry is likely to require either topologies with large errors or coincident cells along the interface. On the other hand, using polynomials of equal or higher order can represent the geometry exactly even with large cell disparities on either side of the interface. This is significant because the most common spline for geometry construction can be meshed using $m = 3$ polynomials.

3.5.2.3 Conclusion

Polynomial based geometry can be made abutting along the interface, even with different numbers of cells on either side. This is not the case for non-polynomial shapes. The circular arc, as tested above, is a commonly used shape for constructing more complex geometries. Simulating any DG-Chimera interface along these geometries requires either coincident cells or overlapped domains. In general, overset topologies are an effective tool for modeling complex geometries. Unfortunately, the physical intent of the CHT interface is negatively affected by the overlap. Ultimately, two choices are available. Either a small enough overlap is used to avoid “significant” errors (where “significant” depends on the problem and the overlap), or the cells are made coincident. The second option appears to be a restriction that removes the benefit of the DG-Chimera scheme. However, a single layer of coincident cells could be used on the interface with an overset mesh used on the interior of the domain. Continued research is needed to determine the best way to handle this “gap problem”.
Chapter 4

Verification

The purpose of verification is to show that the governing equations have been implemented correctly. Any of three different techniques are used to verify the solutions, depending on the type of simulation. First, the order of accuracy can be calculated by determining the solution error for sequentially refined meshes. For a solution with polynomials of order $q$, the order of accuracy should be $q + 1$. The second verification method involves comparing the solution to an analytical solution. The third method compares simulations to benchmark solutions.

4.1 Steady

Numerous verification simulations have been performed by Galbraith [101], including verification of the fluid flow equations with the conservative variables. Verification of the primitive variable version of the fluid flow equations is provided in this section.

4.1.1 Inviscid Flow Over a Smooth Bump

Verification of the advective terms of the primitive variable formulation of the Navier-Stokes equations is conducted with a channel flow over a smooth bump [127]. The flow is isentropic, allowing for the solutions to be verified by calculating an entropy error. The computational domain and an example mesh are shown in Fig. 4.1. The bottom domain boundary is defined as a smooth exponential function (indicated on the figure). Simulations for a range of solution polynomial orders is completed on a series of nested meshes. These meshes range from $6 \times 2$ to $192 \times 62$ with the dimensions of each successive mesh being multiplied by a
factor of 2. Quartic mesh polynomials ($m = 4$) are used for all simulations.

This case is simulated with the Euler equations. A uniform inflow with zero inlet angle is imposed along the left boundary. Total pressure and temperature are applied with a static pressure and temperature of $p = 101.35\text{kPa}$ and $T = 293\text{K}$ and an inlet Mach number of $M_\infty = 0.5$. The same static pressure is imposed at the right boundary. A slip wall condition is specified along the upper and lower boundaries.

Convergence histories of the 48x16 mesh are shown in Fig. 4.2. Each simulation is initialized from constant values. The CFL is ramped by a factor of 10 after each iteration. The $L_2$ norm of the residual drops by 13 orders after about 10 iterations. Contours of the coefficient of pressure are depicted in Fig. 4.3. Regions of high pressure form at the leading and trailing edge of the bump and a low pressure region forms over the bump. Although the $q = 0$ solution depicts these features, the size and symmetry of the high and low pressure regions are not accurately captured.

To test the order of accuracy of the simulations, an appropriate metric is required. For a constant volume inviscid flow without heat addition or work added, the solution should be isentropic. Therefore, an integral
of the entropy error over the domain is an appropriate metric for this case. Using the first and second laws of thermodynamics and the perfect and ideal gas assumptions, the change in entropy is

$$ds = C_p \frac{dT}{T} - R_{\text{gas}} \frac{dp}{p} \quad (4.1)$$

Integrating both sides, combining the constants of integration, and using the properties of natural log gives

$$e^{(s + C_1)} = \frac{T^\gamma}{p} \quad (4.2)$$

and because entropy is constant, the term on the right hand side of Eq. (4.2) should be constant throughout the domain. By comparing the entropy throughout the domain with the reference entropy, an entropy error can be calculated as

$$S_{\text{Error}} = \frac{(T)^{\gamma} - \frac{(T_\infty)^{\gamma}}{p_\infty}}{p - \frac{(T_\infty)^{\gamma}}{p_\infty}} \quad (4.4)$$

$$\text{Entropy Error} = \sqrt{\sum_{e=1}^{N_{cell}} \left[ \frac{\int (S_{\text{Error}})^2 \, d\Omega_e}{\int d\Omega_e} \right]} \quad (4.3)$$

Figure 4.3: Coefficient of pressure for $q = 0$ and $q = 4$ on the $48 \times 16$ mesh.
Figure 4.4: Entropy error for a series of uniformly refined meshes at different solution polynomial orders. Black lines indicate the expected slope.

Table 4.1: Entropy error order of accuracy slopes.

<table>
<thead>
<tr>
<th>q</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.10</td>
<td>1.27</td>
<td>2.50</td>
<td>3.78</td>
<td>5.96</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td>2.05</td>
<td>2.81</td>
<td>3.84</td>
<td>4.14</td>
<td></td>
</tr>
<tr>
<td>0.55</td>
<td>2.34</td>
<td>3.00</td>
<td>4.01</td>
<td>5.30</td>
<td></td>
</tr>
<tr>
<td>0.81</td>
<td>2.41</td>
<td>3.19</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

This error is plotted in Fig. 4.4 for the uniformly refined meshes for \( q = 0 \) through \( q = 4 \). The spacing parameter is \( h = \sqrt{\frac{1}{DOF}} \), where \( DOF = N_{\text{cells}} (q + 1)^2 \) is the number of degrees of freedom. The order of accuracy is the slope of the entropy error curves. These slopes are shown in Table 4.1. As the meshes are refined, the solutions achieve the expected \( q + 1 \) order of accuracy. The \( q = 0 \) solution is a little under the expected value, but from Fig. 4.4 the curve can be seen trending towards a higher value.

4.1.2 Viscous Channel Flow

Verification of the primitive variable formulation of the laminar Navier-Stokes equations is conducted with a viscous channel flow. The simulated streamwise velocity profile will be compared with the analytical solution of incompressible Poiseuille flow. The effects of compressibility on the simulated velocity profiles will be discussed.

The computational domain and an example mesh are shown in Fig. 4.5. The length to height ratio of the channel is 80 to allow for the flow to become fully developed. The height of the channel is based on
$Re_H = 1000$. Simulations for a range of solution polynomial orders is completed on a series of nested meshes. The mesh is represented by linear polynomials ($m = 1$).

Static pressure of $p = 101.35$ kPa and temperature of $T_\infty = 293K$ are assumed and the inlet mach number is $M_\infty = 0.05$. This low Mach number is chosen to reduce the compressibility effects. Total pressure and temperature and $\alpha = 0$ are imposed along the left boundary. A static pressure outflow condition is imposed upon the right boundary. The top and bottom boundaries are specified as no-slip adiabatic walls. Convergence histories of the 40x8 mesh are shown in Fig. 4.2. Each simulation is initialized from constant values. The CFL is ramped by a factor of 10 after each iteration. The $L_2$ norm of the residual drops by 8 orders of magnitude after approximately 15 iterations.

Contours of streamwise velocity are shown in Fig. 4.3. The velocity profile transitions from constant at the inlet, to fully developed at around 60 heights downstream. For Poiseuille flow between two plates, the incompressible theory results in an equation for the fully developed streamwise velocity profile shown below

$$u(y) = \left( -\frac{dp}{dx} \right) \frac{(yH - y^2)}{2\mu}$$

where the pressure gradient in the x-direction, $\frac{dp}{dx}$, is a constant. Despite the low Mach number, the flow simulated here is compressible. This compressibility has two consequences. First, the x-direction pressure gradient is not constant in the fully developed region (between 60 and 80 heights), as shown in Fig 4.8. With a non-constant pressure gradient, Eq. (4.5) is also a function of $x$ and the velocity profile is never truly fully developed. The second consequence of compressibility is that the density varies with span. The velocity profiles are shown in Fig 4.9. For each solution, the black curves depict the analytical velocity profile that is calculated using Eq. (4.5) with $\frac{dp}{dx}$ specified as the exit pressure gradient (see Fig 4.8). In each case, the
Figure 4.6: Convergence history for channel flow on the $40 \times 8$ mesh.

Figure 4.7: Streamwise velocity contours for $q = 4$ on a $40 \times 8$ cell mesh.

Figure 4.8: Pressure gradient in the x-direction for $q = 4$ on a $40 \times 8$ cell mesh.
Figure 4.9: Streamwise velocity profile at the exit of the domain for several polynomial orders and a $40 \times 8$ mesh. The exact solutions (in black) are calculated from Eq. (4.5) using the pressure gradient at the exit.
calculated solution under-predicts the exact solution because of compressibility effects. The $q = 1$ solution representation is less accurate because the expected solution is quadratic. All of the other orders achieve similar accuracies.

### 4.1.3 Turbulent Flat Plate

A two dimensional flat plate flow is used to verify the RANS equations with the S-A turbulence model. Benchmark solutions, computed with CFL3D and FUN3D (obtained from the NASA Turbulence Modeling Resource website [128]), will be used for comparison.

The computational domain and an example mesh are shown in Fig. 4.10. A reference length, $L_{\text{ref}}$, is derived from a Reynolds number of $5 \times 10^6$. The domain has a height equal to $L_{\text{ref}}$ and the plate extends from $x = -0.5L_{\text{ref}}$ to $x = 2L_{\text{ref}}$. Simulations for varying solution polynomial orders are completed on a series of nested meshes, ranging in size from $9 \times 6$ to $144 \times 96$. The meshes are obtained by starting with the finest mesh and delimiting every other cell for each successive mesh. The meshes are clustered at the leading edge and towards the lower boundary to capture the large velocity gradients in those regions. This clustering is similar to the clustering applied to the meshes on the NASA Turbulence Modeling Resource website. Linear polynomials are used to represent the mesh.

Freestream pressure of $p_\infty = 101.35kPa$ and temperature of $T_\infty = 300K$ are assumed and the inlet Mach number is $M_\infty = 0.2$. At the left inlet boundary, zero inlet angle ($\alpha = 0$) and total pressure and temperature are imposed. Also, the freestream flow is assumed to be fully turbulent, which is achieved by imposing $\tilde{\nu}/\nu_\infty = 3$ at the inlet. A static pressure outflow condition is imposed upon the right boundary. A symmetry condition is applied along the top boundary and the bottom boundary for $x < 0$. The remainder of the bottom
boundary is specified as a no-slip adiabatic wall. Convergence histories of the $36 \times 24$ mesh are shown in Fig. 4.11. For each mesh, the $q = 1$ simulation is initialized from constant values and higher order cases are initialized from the previous solutions. The $CFL$ is ramped by the ratio of initial to current $L_2$. The decrease of the residual for the $q = 1$ simulation slows as $\tilde{\nu}$ is produced and the boundary layer grows. Once the final extent of the boundary layer is reached, convergence is rapidly achieved. This effect is less pronounced for the simulations with higher polynomial orders because the lower order solutions are used for initialization.

Contours of eddy to freestream viscosity ratio in the boundary layer are shown in Fig. 4.12. The agreement between the current and CFL3D solutions is good. A small discrepancy occurs downstream, which is likely caused by Gibbs phenomena produced at the leading edge singularity. Figure 4.13 shows the overshoot of the S-A working variable near the leading edge region. This overshoot results in a small amount of eddy viscosity in the cell upstream of the leading edge. Similar over- and under-shoots are present in the other primitive variables as well.

Despite the existence of the Gibbs phenomena, the current simulation results compare exceptionally well with those of the CFL3D simulation at $x = 1.095$. The streamwise velocity in the boundary layer, shown in Fig. 4.14, compares particularly well. The difference in the boundary layer profile for different polynomial orders is also small. Details in the inner wall region are discussed in Sec. 4.1.3.1.

Profiles of the eddy to freestream viscosity ratio for the $q = 4$ case compare well with the CFL3D results, as depicted in Fig. 4.15. The largest difference occurs at the outer edge of the boundary layer. At the edge of the boundary layer, a quick transition between zero and non-zero eddy viscosity occurs. The
Figure 4.12: Boundary layer eddy to freestream viscosity ratio contours. Contour lines comparison: red is $q = 4$ on a 72x48 mesh, blue is CFL3D on a 545x385 mesh.

Figure 4.13: Contours of the S-A working variable depicting Gibbs phenomena at the plate leading edge. $q = 4$ solution on the $72 \times 48$ mesh.
CFL3D transition is rounded, suggesting the presence of numerical dissipation. On the other hand, the $q = 4$ solution transitions sharply, with a small over-shoot above the transition. Larger over- and under-shoots exist in the lower order solutions, as presented in Fig. 4.15b. The $q = 1$ solution, in particular, performs poorly at the boundary layer edge. The over- and under-shoots of eddy viscosity are a result of under-resolution of the high $\tilde{\nu}$ gradients at the boundary layer edge. Without adequate resolution, negative $\tilde{\nu}$ persists in the DG solutions, as illustrated in Fig. 4.16. Unfortunately, the position of the boundary layer edge cannot be known a priori. Possible solutions to this problem are adjoint error adaptation of the mesh, artificial viscosity applied to $\tilde{\nu}$, or a finite-volume treatment of the S-A equations (these are explored by Oliver [129] and Burgess [130]).

A comparison of the coefficient of friction along the plate is shown in Fig. 4.17. The $q = 4$ and CFL3D solutions compare well. Small differences exist along the plate for the lower polynomial order solutions. One clear difference between the current and CFL3D solutions is the significant oscillations present near the leading edge. As mentioned previously, a discontinuity exists at the transition from a symmetry boundary to a no-slip wall boundary. This discontinuity produces Gibbs phenomena in the DG solutions. One possible remedy is the use of artificial viscosity, though this only attempts to smooth over a singularity imposed by the test case. Another option is to use highly refined meshes around the leading edge by exploiting the Chimera overset capability of the solver.

The coefficient of drag and the coefficient of friction at $x = 1.095$ are shown for a series of meshes.
Figure 4.15: Profiles of eddy to freestream viscosity ratio at $x = 1.095$. Current simulation on a $72 \times 48$ mesh, CFL3D on a $545 \times 385$ mesh. The current simulation cell edges are indicated by the symbols.

Figure 4.16: S-A working variable profile comparison at $x = 1.095$. Several polynomial order solutions on a $72x48$ mesh. The cell edges are indicated by the symbols.
Figure 4.17: Coefficient of friction along the plate. Current simulation on a $36 \times 24$ mesh, CFL3D on a $545 \times 385$ mesh.

in Figs. 4.18 and 4.19. The curves of the current simulation appear to be approaching final values that differ slightly from CFL3D and FUN3D. This is not unexpected, given that small variations caused by the leading edge discontinuity. Unfortunately, the Gibbs phenomena at the leading edge makes a formal order of accuracy study impossible because increasing the mesh or polynomial resolution is not guaranteed to reduce that source of error.

4.1.3.1 Inner Layer Resolution

Resolving the inner layer region of a boundary layer is vital for the accurate prediction of turbulent flows. The relation between the inner layer velocity, $u^+$, and distance, $y^+$, is linear close to the wall and transitions to a logarithmic relationship. A common metric that is used to gauge the resolution of the inner layer is the $y^+$ value at the edge of the first cell off the wall. The $y^+$ values of the first cell for several of the meshes used here are listed in Table 4.2. For structured finite volume meshes, a value below one is desired. Often, a wall function is utilized to model the logarithmic region when the inner layer is under-resolved. When the solution is represented by polynomials, these rules-of-thumb do not apply.

Profiles of the inner wall variables are shown in Fig. 4.20 for several meshes and polynomial orders. Comparison of the $q = 4$ solutions and the CFL3D solution are shown in Fig. 4.20a. All of the meshes compare well with the CFL3D solution in the linear and log-law regions. The coarsest mesh has a slight inaccuracy in the transition region. However, the solution compares surprisingly well for a mesh with a
Figure 4.18: Plate coefficient of drag for a series of meshes. $h = 1/\sqrt{\text{DOF}}$ is measure of the mesh resolution.

Figure 4.19: Coefficient of friction at $x = 1.095$. $h = 1/\sqrt{\text{DOF}}$ is measure of the mesh resolution.

Table 4.2: Mesh sizes and first cell $y^+$ values at $x = 1.095$.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>$y^+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 × 6</td>
<td>10.4</td>
</tr>
<tr>
<td>18 × 12</td>
<td>3.5</td>
</tr>
<tr>
<td>36 × 24</td>
<td>1.5</td>
</tr>
<tr>
<td>545 × 385</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Figure 4.20: Inner wall variable profiles at $x = 1.095$. Current simulation mesh sizes denoted in the figure caption or key. CFL3D solution on a 545x385 mesh. Symbols indicate the location of cell boundaries for the current simulation.
y+ of 10.4. All of the current solutions compare well in the linear region, even the q = 1 solution on the coarsest mesh. However, at the coarsest resolution, the transition and log-law regions are not captured well for the lower polynomial orders. Clearly, the required inner layer resolution is polynomial dependent as well as mesh dependent. Also, for higher polynomial orders, y+ between 3.5 and 10.4 are adequate for this case. This is not meant to be recommendation for the first cell distance, but rather as a comparison with requirements of a finite volume simulation. The common recommendation of y+ < 1 is difficult to determine prior to the simulation. Hence, the accuracy of the solution with relaxed y+ limitations is a clear advantage of the DG approach. Further studies of more complex cases are necessary to determine the best practice for real-world applications.

4.1.3.2 Chimera Mesh

A study of the effectiveness of the DG-Chimera approach for the S-A turbulence model is presented here. The mesh shown in Fig. 4.10 is split into two meshes, which are overlapped by 50% of the upstream cell width, as shown in Fig. 4.21. A comparison of the single and chimera mesh solutions is presented in Fig. 4.22 for several polynomial orders. Also listed are the inlet to exit mass flow errors. As expected, the mass flow errors for the single mesh are machine zero. The use of the Chimera overset scheme introduces mass flow errors. This result is expected and discussed at length by Galbraith [101]. As the order of accuracy is increased, the mass flow errors decrease. Refining the mesh spacing also decreases the mass flow errors. The coefficient of friction plots compare well for all polynomial orders, with slightly larger discrepancies for the linear solution. A similar conclusion can be made for the contour plots of the eddy to inlet viscosity ratio. Some differences in the contour lines are present in the q = 1 boundary layer. In addition, small differences are present for all orders in the region above the boundary layer.
\( q = 1 \)

<table>
<thead>
<tr>
<th>( \Delta \dot{m}_{\text{error}} )</th>
<th>Single</th>
<th>Chimera</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2.83 \times 10^{-14} % )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5.34 \times 10^{-5} % )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( q = 2 \)

<table>
<thead>
<tr>
<th>( \Delta \dot{m}_{\text{error}} )</th>
<th>Single</th>
<th>Chimera</th>
</tr>
</thead>
<tbody>
<tr>
<td>( -7.67 \times 10^{-14} % )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -2.81 \times 10^{-5} % )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( q = 3 \)

<table>
<thead>
<tr>
<th>( \Delta \dot{m}_{\text{error}} )</th>
<th>Single</th>
<th>Chimera</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 7.44 \times 10^{-15} % )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 3.85 \times 10^{-6} % )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( q = 4 \)

<table>
<thead>
<tr>
<th>( \Delta \dot{m}_{\text{error}} )</th>
<th>Single</th>
<th>Chimera</th>
</tr>
</thead>
<tbody>
<tr>
<td>( -1.05 \times 10^{-14} % )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -8.45 \times 10^{-7} % )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.22: Comparison between single and chimera solutions for the 18 × 12 mesh.
4.2 Harmonic Balance

The ability of the Harmonic Balance (HB) method to accurately capture periodic unsteady flows is shown in this section. Comparisons will be made with a 4\textsuperscript{th} order explicit Runge-Kutta (RK4) method and a 3\textsuperscript{rd}-order accurate Diagonally Implicit Runge-Kutta (DIRK)[131] scheme. Both RK4 and DIRK have been implemented into the DG-Chimera framework and verified by Galbraith [101]. These two methods are in the category of time-marching (TM) methods in that they advance the solution by marching in time.

4.2.1 Linear Advection Equation

The first verification case for the HB method is a simulation of the one-dimensional linear advection equation. The governing equation in one dimension is

\[ u_t - cu_x = 0 \]  

(4.6)

where the advection speed is \( c = 1 \) for these simulations. The domain extends from \( x = 0 \) to \( x = 1 \) and is discretized with 200 cells uniformly spaced along the domain. Polynomials of order 3 are used (i.e. \( q = 3 \)). At the left boundary, a sinusoidal variation in time is prescribed, as

\[ u(0, t) = \sin(2\pi t) \]  

(4.7)

and \( u \) is extrapolated at the right boundary.

Verification is achieved through a comparison with the RK4 method. A time step of \( 6.25 \times 10^{-4} \) is used for the RK4 simulations. Two full periods of the oscillation are completed for the TM simulation, the first to allow the initial transient to pass and the second for solution comparison with the HB method. At each instant in time during the TM simulation, Eq. (4.7) is imposed. Similarly, the boundary condition is applied to the HB time levels as

\[ u_i(0) = \sin(2\pi t_i) \]  

(4.8)

where \( t_i \) is the time at each time level. A single HB frequency of \( 2\pi \) is used.

Because of the complete linearization and the linear nature of the governing equation, the HB simulation converges in one iteration. The HB solutions are shown in Fig. 4.23a. Notice that the value at the left
Table 4.3: Error for the HB and TM methods at $t = 0$.

<table>
<thead>
<tr>
<th></th>
<th>$L^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HB</td>
<td>$4.44 \times 10^{-11}$</td>
</tr>
<tr>
<td>TM</td>
<td>$1.03 \times 10^{-10}$</td>
</tr>
</tbody>
</table>

boundary for each time level is the same as the value imposed by Eq. (4.8) at the uniformly chosen times $t_1 = 0$, $t_2 = \frac{1}{3}$, and $t_3 = \frac{2}{3}$. Using the HB Fourier transform operator from Section 2.3, the coefficients are calculated and displayed in Fig. 4.23b. Finally, the truncated Fourier series, Eq. (2.88), can be used to reconstruct the solution at any time instance. The reconstructed HB solutions are compared with the TM solutions in Fig. 4.23c. For both the HB and TM methods, a comparison of the $L^2$-norm of the simulated solution, $u(x,t)$, to the analytical solution, $u_a(x,t)$, can be made with the following equation

$$L^2 = \sqrt{\int_{\Omega} (u - u_a)^2 \, d\Omega}$$

(4.9)

where

$$u_a(x,t) = -\sin(2\pi(x-t))$$

(4.10)

The low error shown in Table 4.3 indicates that both the TM and HB methods are accurately capturing the unsteadiness.

4.2.2 Poisson Equation

A similar exercise is performed for the Poisson equation. The governing equation in one dimension is

$$u_t - \alpha u_{xx} = 0$$

(4.11)

where $\alpha = 0.1$ for these simulations. The domain extends from $x = 0$ to $x = 1$ and is discretized with 200 cells uniformly spaced along the domain. Polynomials of order 3 are used (i.e. $q = 3$). At the left boundary, a sinusoidal variation in time is prescribed, as described by Eq. (4.7). At the right boundary, a zero Dirichlet boundary is imposed.

The DIRK method with a $5.0 \times 10^{-4}$ time step is used to calculate solutions for comparison with the HB method. A simulation of ten seconds is conducted to converge to a periodic solution. At each step of the TM simulation, Eq. (4.7) is imposed. Similarly, the boundary condition is applied to the HB simulations.
Figure 4.23: Solutions to the linear advection equation with a sinusoidally varying boundary. The time instances plot includes TM solutions and solutions reconstructed from the HB solutions at the same time instances.
Table 4.4: $L^2$-norm of the difference between the HB solution and TM solutions at $t = 10$.

<table>
<thead>
<tr>
<th>Time Step</th>
<th>$L^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5.0 \times 10^{-2}$</td>
<td>$8.85 \times 10^{-4}$</td>
</tr>
<tr>
<td>$5.0 \times 10^{-3}$</td>
<td>$7.61 \times 10^{-3}$</td>
</tr>
<tr>
<td>$5.0 \times 10^{-4}$</td>
<td>$7.93 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

given by Eq. (4.8). A single HB frequency of $2\pi$ is used.

Because of the complete linearization and the linear nature of the governing equation, the HB simulation converges in one iteration. The HB solutions are shown in Fig. 4.24a. Notice that the value at the left boundary for each time level is the same as the value imposed by Eq. (4.8) at the uniformly chosen times $t_1 = 0$, $t_2 = \frac{1}{3}$, and $t_3 = \frac{2}{3}$. Using the HB Fourier transform operator from Section 2.3, the coefficients are calculated and displayed in Fig. 4.24b. The truncated Fourier series, Eq. (2.88), is used to reconstruct the solution at any time instance. The reconstructed HB solutions are compared with the TM solutions in Fig. 4.24c. Visually, the agreement between the two methods is good.

Upon further inspection, some discrepancies are apparent, most notably at the left boundary. A comparison of the HB solution and TM solutions with varying time step is shown in Fig 4.25. As the time step is decreased, the TM solution more closely matches the imposed boundary condition (and the HB solution). For the TM solutions, an $L^2$-norm of the TM solutions is calculated, as shown in the equation below

$$L^2 = \sqrt{\int_{\Omega} (u_{TM} - u_{HB})^2 d\Omega} \quad (4.12)$$

where $u_{TM}$ and $u_{HB}$ are the TM and HB solutions, respectively. The values for several TM simulations for $t = 0$ are given in Table 4.4. These results emphasize the importance of choosing a time step that is small enough to accurately resolve the unsteady features. Another interesting conclusion is that the HB method more accurately represents the imposed boundary condition, $u = 0$ at $t = 10$, (see Fig. 4.25b), than the TM method (even for the smallest time step shown here).
Figure 4.24: Solutions to Poisson’s equation with a sinusoidally varying boundary. The time instances plot includes TM solutions and solutions reconstructed from the HB solutions at the same time instances.
4.2.3 Euler Cascade

A 2D linear cascade of the mid-span section of the rotor from the low speed Aachen turbine[132] is simulated using the Euler equations. The computational domain is shown in Figure 4.26a. An O-mesh (1) is placed around the blade and surrounded with 4 H-meshes (2-5). Additional H-meshes (6, 7) extend from the inlet and exit to the near blade meshes. The mesh sizes are listed in Table 4.5. A quartic polynomial representation ($m = 4$) is used for all meshes.

The freestream static and total temperature are $T_\infty = 295.2$K and $T_{T\infty} = 297.1$K, respectively. From these values, the freestream Mach number is approximately $M_\infty = 0.1794$. A freestream total pressure of $p_{T\infty} = 151.5$kPa is assumed, resulting in a freestream static pressure of roughly $p_{T\infty} = 148.1$kPa. Total conditions with a sinusoidally varying total temperature are applied at the inlet. Static pressure is applied at the exit. These inlet and exit boundary conditions are known to be reflective. The equation for the inlet total temperature is

$$T_T(y) = T_{T\infty} \left( 1 + 0.05 \sin \left( 2\pi \left( \frac{t}{\Delta \text{period}} + \frac{y}{H} \right) \right) \right)$$  \hspace{1cm} (4.13)
Figure 4.26: Cascade computational domain. Every other cell is delimited in (a). The red mesh in (b) is a copy of the black mesh that is offset by the passage height \(H\).

where \(\Delta t_{\text{period}} = \frac{H}{V_{\text{rotor}}}\) is the perturbation period and \(H\) is the height of the passage. The height is

\[
H = \frac{2\pi}{B_{\text{rotor}}} r_{\text{mid}}
\]

with \(r_{\text{mid}} = 0.2725\text{m}\) being the mid-span radius and \(B_{\text{rotor}} = 41\) being the blade count of the Aachen rotor. This choice of perturbation ensures that the solution is periodic across a single passage. The rotating speed is

\[
V_{\text{rotor}} = \frac{2\pi}{60} r_{\text{mid}} \omega_{\text{rotation}}
\]

where \(\omega_{\text{rotation}} = 3500\text{rpm}\) is the Aachen rotor rotational speed. A slip condition is enforced along the blade boundary. The upper boundaries of meshes 5, 6, 7 and the lower boundaries of meshes 4, 6 and 7 are periodic. These periodic boundaries are overset, as shown by Fig. 4.26b, by offsetting them by a distance \(H\) (see Section 3.4.3). Because the height of the perturbation is equal to the height of the passage, no phase shift will occur at the periodic boundaries.
The HB simulations include the frequency of the temperature variation and higher harmonics. The Quasi-Newton method is used to drive the HB residuals below a tolerance of $1.0 \times 10^{-9}$. The linear system is solved using FGMRES with HB ILU(0) preconditioning. Simulations with $K = 1, 2, 5$ frequencies have been computed. The $K = 5$ solutions are presented in this section, unless stated otherwise. Convergence histories of these simulations are shown in Fig. 4.27. Both the $K = 2$ and $K = 5$ simulations are initialized with the $K = 1$ solution. All of the HB simulations suffer from a slow convergence of the linear system. To maintain the simulation stability, the CFL is limited to 100 for $K = 1$ and 50 for all other $K$. Despite this limit, the simulations converge in a few hundred thousand.

![Figure 4.27: HB simulation convergence histories.](image)

Simulations using the DIRK scheme are completed for comparison. This scheme consists of 3 stages per time step. At each stage, the residual is driven below a tolerance of $1.0 \times 10^{-12}$. Completing each step to this tolerance is necessary to accurately capture the temporal variations. A time step resolution study has been conducted with 40, 80, and 160 time steps per period of the total temperature variation. Each simulation is conducted for 20 periods to ensure a periodic solution. Unless otherwise noted, the results of the 160 time steps per period simulation are shown in this section.

Both the HB and TM simulations utilize a 4th order accurate spatial representation ($q = 3$). Comparison between the unsteady total temperature contours of the two approaches is good, as shown in Figure 4.28. Recall that the imposed perturbation, in Eq. (4.13), consists of only one frequency. However, additional frequency content arises because of the non-linearity of the governing equations. In acoustics, this phe-
nomenon is termed frequency scattering. To understand the effect of frequency scattering, the magnitude of the first 5 modes of pressure are calculated for the HB and TM methods. These five modes are shown in Figs. 4.29-4.33. The HB contours compare very well for all 5 modes. At the higher modes (most notably the fifth mode), the HB method magnitudes are slightly higher than the TM magnitudes. This difference is the result of dispersion errors in the TM method and aliasing of higher frequency content into the lower modes for the HB method.

The reflective nature of the inlet and outlet boundary conditions produce spurious pressure waves that appear in the pressure magnitude plots. The features parallel to the inlet and exit boundaries, such as downstream in Figs 4.30 and 4.31, are the result of the boundary condition reflections. However, several features of these contours indicate that much of the frequency scatter content is physical. Most notable are the features within the blade passage in Fig. 4.29. The shape of these features and their higher magnitude relative to boundary condition reflections (and the rest of the flow field) are indicative of frequency scatter. Additional physical features are evident along the suction side near the trailing edge in Fig. 4.30, and along both the suction and pressure side of Fig. 4.33.

Figure 4.28: Instantaneous total temperature contours for HB $K = 5$ and TM with 160 time steps per period.
Figure 4.29: 1st mode pressure magnitude comparison for HB $K = 5$ and TM with 160 time steps per period.

Figure 4.30: 2nd mode pressure magnitude comparison for HB $K = 5$ and TM with 160 time steps per period.
Figure 4.31: 3\textsuperscript{rd} mode pressure magnitude comparison for HB $K = 5$ and TM with 160 time steps per period.

Figure 4.32: 4\textsuperscript{th} mode pressure magnitude comparison for HB $K = 5$ and TM with 160 time steps per period.
4.2.3.1 Which Method is Right?

A common theme with the research of HB and other frequency domain methods is to verify the approach by comparing to time-marching methods. Often, the researcher assumes the time-marching method captures the right answer, which the frequency domain methods can only hope to emulate. However, the accuracy of the TM solution is dependent upon both the order of accuracy of the scheme and the size of the time step. In addition, both the TM and HB simulations depend on the mesh resolution (and not necessarily in the same way). Instead of assuming one or the other is correct, the resolution of both schemes should be increased until the desired features are resolved. To that end, Fig. 4.34 displays the HB and TM solutions at several spectral and temporal resolutions, respectively. The magnitude of the first pressure mode is extracted along a line (shown in Fig. 4.34a) to more easily compare the solutions. A location between the blades is chosen for the extraction because the largest unsteady features are present in that region. Figure 4.34b shows the extracted pressure mode values with an inset figure showing a zoomed in view near the peak. Notice that as the resolution of the TM and HB solutions increases (smaller TM time step and more HB frequencies), the curves move towards each other. Namely, the 160 time step per period TM simulation and the $K = 5$ HB simulation curves agree well. In addition, the 80 time step per period TM simulation and the $K = 2$ HB simulation predict a similar curve for the pressure mode. The TM curves are monotonically trending upwards as the resolution increases. This behavior is expected as lower temporal resolution introduces
numerical dissipation that dampens unsteadiness (and therefore decreases the pressure magnitude). On the other hand, the HB trend is downward from $K = 1$ to $K = 2$ and then upwards from $K = 2$ to $K = 5$. Evidently, the errors are not reduced in a predictable way as more frequencies are added.

This discussion is not meant to answer the question posed by the section title. Ultimately, comparison with experiment would be necessary to better understand the accuracy of the HB and TM methods. This type of comparison is outside of the scope of a verification effort. The important conclusion to take away from this exercise is that both methods are capable of capturing similar unsteadiness given sufficient resolution.

### 4.2.4 Shedding Cylinder

A simulation of the Navier-Stokes equations is conducted for a cylinder with $Re = 100$ and $M_\infty = 0.1$. The computational domain is depicted in Figure 4.35. The mesh dimensions are 40 cells along the cylinder and 50 cells normal to the cylinder. Each cell is represented with 3rd order mesh polynomials ($m = 2$). An adiabatic no-slip boundary condition is applied at the cylinder surface and a Riemann Invariant condition is applied along the far field boundary to impose the freestream conditions. All simulations are completed using cubic solution polynomials ($q = 3$).
At this Reynolds number, the cylinder sheds naturally. For simulations with natural shedding, the dominant frequency is not known a priori. Experiments provide fundamental frequencies for simple cases (such as this case), but these values have some small uncertainty. For HB simulations, choosing an inaccurate frequency (even with small errors) will result in insufficient convergence. After reaching this insufficient convergence, the phase of the HB solutions changes at a constant rate [125]. This problem can be resolved with gradient-based methods for finding the fundamental frequency [55, 48]. Using the gradient-based methods, the HB frequencies are modified during the simulation until the phase rate of change of the HB solution reaches zero. A zero phase rate of change indicates that the shedding frequency has been found.

A gradient-based search procedure is not currently implemented in the implicit Harmonic Balance method. Instead, the natural shedding frequency is obtained from the TM simulation. The TM simulation utilizes the DIRK scheme with a time step of \(1.0 \times 10^{-3}\) seconds. Several HB simulations are presented ranging from \(K = 1\) to \(K = 8\). The frequencies include the natural shedding frequency and higher harmonics. The HB linear system is solved using FGMRES with HB ILU(0) preconditioning. The shedding for both the TM and HB simulations is started using a perturbation of the initial flow field. The convergence history of the HB simulations is shown in Fig. 4.36. Each case is initialized with the previous \(K\) solution except \(K = 5, 6\) and \(K = 7, 8\), which are initialized by \(K = 4\) and \(K = 6\), respectively. The HB simulations converge rapidly for a few hundred iterations and then remain nearly constant. In this case, the convergence is not limited by the linear solve. Instead, convergence is limited because the TM shedding frequency is used, rather than determining the frequency with a gradient-based method.
Figure 4.36: HB simulation convergence histories for the shedding cylinder.

Table 4.6: Time averaged lift and drag coefficients.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Max $C_L$</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>0.32375</td>
<td>1.3478</td>
</tr>
<tr>
<td>HB $K = 1$</td>
<td>0.30362</td>
<td>1.3342</td>
</tr>
<tr>
<td>HB $K = 2$</td>
<td>0.30735</td>
<td>1.3440</td>
</tr>
<tr>
<td>HB $K = 3$</td>
<td>0.32272</td>
<td>1.3463</td>
</tr>
<tr>
<td>HB $K = 4$</td>
<td>0.32380</td>
<td>1.3479</td>
</tr>
<tr>
<td>HB $K = 6$</td>
<td>0.32374</td>
<td>1.3479</td>
</tr>
<tr>
<td>HB $K = 8$</td>
<td>0.32376</td>
<td>1.3479</td>
</tr>
</tbody>
</table>

A comparison of the maximum coefficient of lift ($C_{L,max}$) and time averaged coefficient of drag ($C_D$) is shown in Table 4.6. As the number of frequencies is increased, the HB $C_{L,max}$ and time averaged $C_D$ approach the TM values. The HB cases with $K = 4$ and higher are within one drag count of the time averaged drag coefficient. The HB $C_{L,max}$ is within 0.5% of the TM value for $K = 3$ and within 0.02% for $K = 4$.

The HB method is able to accurately capture the unsteady lift and drag as depicted in Fig. 4.37. With just two frequencies, the HB computed $C_L$ compares well to the TM solution and with four frequencies the HB and TM curves are indistinguishable. The $C_D$ for the HB $K = 2$ simulation is not as accurate; however, the HB $K = 4$ and $K = 8$ simulations compare well with the TM simulation. Figure 4.38 provides a comparison of the downstream vorticity contour lines of the TM and HB $K = 1, 2, 4, 8$ solutions. The vortical structures of the HB $K = 1$ and $K = 2$ solutions have different shapes than the TM solutions. As the number of frequencies increases, the HB solution better matches the TM solution.
Figure 4.37: Coefficients of lift and drag over one period for the TM and several HB simulations.

Figure 4.38: Comparison of vorticity contours in the cylinder wake.
4.3 Conjugate Heat Transfer

Verification of the CHT method is made with comparisons to analytical solutions for two one-dimensional cases. The first case involves only conduction heat transfer, whereas the second includes convection.

4.3.1 Conduction

A simple case is constructed to ensure the CHT method accurately captures conduction heat transfer. The case is depicted in Fig. 4.39. The simulation is in two dimensions, but the solution is one-dimensional. Quiescent flow is imposed upon the blue fluid domain with a temperature and velocity inflow along the left boundary. Static pressure is imposed along the right boundary. An iso-thermal condition with temperature \( T_{fb} \) is applied to the top fluid boundary. For the green solid domain, adiabatic conditions are imposed on the side boundaries and a constant temperature, \( T_{sb} \), is given for the bottom boundary.

The laminar Navier–Stokes equations are solved in the fluid with a constant viscosity. This results in a constant thermal conductivity because \( C_p \) and \( Pr \) are constant. The solid conduction equation is applied to the solid domain with a constant thermal conductivity. Several solid to fluid thermal conductivity ratios, \( \frac{k_s}{k_f} \), are simulated. All simulations are initialized to the average of the fluid and solid boundary temperatures. Given the boundary conditions and the constant thermal conductivities, the temperature in the fluid and solid
domains depend only on \( y \), as described by the following equations

\[
T_{\text{Fluid}}(y) = (T_{fb} - T_{sb}) \frac{k_f y + k_f}{k_s + k_f} + T_{sb}
\]
\[
T_{\text{Solid}}(y) = (T_{fb} - T_{sb}) \frac{k_f y + k_s}{k_s + k_f} + T_{sb}
\]

where \( k_f \) and \( k_s \) are the fluid and solid thermal conductivities respectively.

Details of the simulation convergence for the different cases are shown in Fig. 4.40. All of the simulations converge in less than 30 iterations. A constant rate of convergence is seen. The expected convergence trends appear for the other CHT simulations in this work. Given that those simulations all include convecting flow, the only apparent explanation for the slow convergence is the imposition of quiescent flow in the fluid domain. It is possible that the simulation is ill-posed because the left and right fluid boundary conditions are designed for flow entering and leaving the domain, respectively. With the constant convergence rate, the cases with higher thermal conductivity ratios require more iterations for convergence because the initial solution is further from the converged solution. The solutions are displayed in Fig. 4.41. The results match the analytical solutions exactly, therefore the analytical values are not shown. The conductivity ratios of 1,000 and 10,000 are on the order of steel to air and aluminum to air conductivity ratios, respectively. At these high values, most of the temperature variation occurs in the fluid domain.
Figure 4.41: Solutions for varying $\frac{k_s}{k_f}$ with $q = 1$ and a mesh of 5x10 in each domain.

Figure 4.42: Source driven Poiseuille flow domain details. The fluid and solid domains shown in blue and green, respectively.

### 4.3.2 Convection

A common choice for simple convecting flow that varies only in one dimension is a Poiseuille channel flow (similar to Section 4.1.2). However, a true Poiseuille flow cannot provide a one-dimensional solution for CHT simulations because a pressure change (and thus a temperature change) is required in the $x$ direction. An appropriate fully developed flow for CHT simulations was proposed by Ojeda [133], whereby a source term drives the fluid flow, allowing temperature to be independent of the streamwise direction. An example of the domain topology is shown in Fig. 4.42. The solid domain is constructed in the same way as with the conduction case. Riemann invariant boundary conditions are imposed along the fluid inflow and outflow boundaries. The channel is completed with a symmetry boundary imposed along the top fluid boundary.

As with the conduction case, the fluid domain is simulated with the laminar Navier-Stokes equations
and the solid domain is simulated with the solid conduction equation. The fluid viscosity and solid thermal conductivity are made constant. For Poiseuille flow, the flow does not vary in the streamwise direction. Using similar assumptions, the governing equations for the source driven Poiseuille flow are

\[-\frac{\partial}{\partial y} \left( \mu \frac{\partial u}{\partial y} \right) = a \]

\[-\frac{\partial}{\partial y} \left( \mu u \frac{\partial u}{\partial y} + k_f \frac{\partial T}{\partial y} \right) = au \]  \( (4.16) \)

where \( k_f \) is the fluid thermal conductivity and \( a \) is the source term value. Note the similarity to traditional Poiseuille flow [134]. The difference is the source term, \( a \), in place of a pressure gradient in the momentum equation and the inclusion of the source term in the energy equation. The solid similarly does not vary in the streamwise direction, giving the governing equation as

\[-\frac{\partial}{\partial y} \left( k_s \frac{\partial T}{\partial y} \right) = 0 \]  \( (4.17) \)

where \( k_s \) is the solid thermal conductivity. These three equations can be solved analytically using integration, which requires 6 conditions to solve for the constants of integration. These conditions are

- \( \frac{\partial u}{\partial y} = 0 \) at \( y = \frac{H_f}{2} \)
- \( \frac{\partial T}{\partial y} = 0 \) at \( y = \frac{H_f}{2} \)
- \( u = 0 \) at \( y = 0 \)
- \( T_{\text{fluid}} = T_{\text{solid}} \) at \( y = 0 \)
- \( \left( k \frac{\partial T}{\partial y} \right)_{\text{fluid}} = \left( k \frac{\partial T}{\partial y} \right)_{\text{fluid}} \) at \( y = 0 \)
- \( T = T_w \) at \( y = H_s \)

where \( H_f \) is the channel height, \( \frac{H_f}{2} \) is the location of the symmetry boundary, and \( H_s \) is the height of the solid domain. These conditions are simply the application of the properties of the symmetry and solid wall
boundaries and the CHT interface. The resulting analytical solution is

\[
\begin{align*}
    u(y) &= \frac{a}{2\mu} \left( H_f y - y^2 \right) \\
    T_{\text{Fluid}}(y) &= \frac{a^2}{24k_f\mu} \left( 2H_f y^4 - 4H_f y^3 + 3H_f^2 y^2 - H_f^3 y \right) + T_i \\
    T_{\text{Solid}}(y) &= \frac{a^2 H_f^3}{24k_s\mu} y + T_i \\
    T_i &= \frac{a^2 H_f^3 H_s}{24k_s\mu} + T_w
\end{align*}
\]

(4.18)

where \( T_i \) is the interface temperature. For all cases presented here, the simulation parameters are \( H_f = 2, H_s = 1, C_p = 1000, Pr = 1, \mu = 1e^{-5}, a = 120\mu, \frac{k_s}{k_f} = 4, T_w = 300 \). A domain length of \( 5H_f \) is used to ensure variation in the streamwise direction is zero.

Simulations for a range of solution polynomial orders are completed on a series of nested meshes. An example convergence history is depicted in Fig. 4.43. The \( q = 1 \) simulations start with constant initial values. Subsequent higher polynomial orders initialize with the previous solution. Using this simulation procedure, solutions for all polynomial orders are obtained in 35 iterations. A comparison of the calculated and analytical solutions is shown in Fig. 4.44. The match is excellent and the error is dictated by the accuracy of the polynomial representation. In addition, the normal velocity, pressure, and streamwise velocity variations in the \( x \) direction are negligible.

The last step for verifying the CHT method is to perform an order of accuracy analysis. The order of
accuracy for a polynomial of order $q$ is expected to be $q + 1$. The order of accuracy can be obtained by calculating the slope of the logarithm of the mesh refinement versus the logarithm of the error. In this case, two errors are calculated for comparison purposes. The first is a volume integral of the temperature error, calculated as

$$\text{Temperature Error} = \sqrt{\sum_{e=1}^{N_{\text{cells}}} \left( T_{\text{analytical}} - T_{\text{computed}} \right)^2 d\Omega_e}$$  \hspace{1cm} (4.19)$$

where the volume integrals are computed using Gauss quadrature. The second is the error of the heat transfer along the interface, described by the following equation

$$\text{Heat Transfer Error} = \sum_{f=1}^{N_{\text{faces}}} \left| \int_{\Sigma} (q_{\text{Fluid}}) d\Gamma + \int_{\Sigma} (q_{\text{Solid}}) d\Gamma \right|$$  \hspace{1cm} (4.20)$$

where $q_{\text{Fluid}}$ and $q_{\text{Solid}}$ are the fluid and solid heat transfers, respectively. Plots of the accuracy versus mesh refinement for these two error metrics are shown in Fig. 4.45. The quantity $\log_{10} \left( \frac{1}{\sqrt{\text{DOF}}} \right)$ is a measure of the mesh resolution, where $\text{DOF}$ is the number of degrees of freedom (computed as $N_{\text{cells}} (q + 1)^2$). The most notable trend is for the $q = 4$ solutions where the error starts low and increases as the meshes are refined. This is because the analytical solution for temperature is a $4^{th}$ order polynomial. Therefore, the $q = 4$ solution is at the lowest error, which is around the tolerance for convergence of the solutions. As the
Figure 4.45: Error metrics for a series of uniformly refined meshes with different solution polynomial orders.

Table 4.7: Temperature error order of accuracy slopes.

<table>
<thead>
<tr>
<th>$q$ = 1</th>
<th>$q$ = 2</th>
<th>$q$ = 3</th>
<th>$q$ = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.18</td>
<td>2.23</td>
<td>4.00</td>
<td>–</td>
</tr>
<tr>
<td>1.63</td>
<td>2.35</td>
<td>4.16</td>
<td>–</td>
</tr>
<tr>
<td>1.89</td>
<td>2.65</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>1.97</td>
<td>3.04</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

number of cells or faces increases, the $q = 4$ error increases because that lowest error is added $N_{\text{cells}}$ and $N_{\text{faces}}$ times (see Eqs. 4.19 and 4.20). The errors for the $q = 3$ solutions drop to the minimum value by the 4th mesh. The slopes are obtained from Fig. 4.45 and are tabulated in Tables 4.7 and 4.8. The $q + 1$ order of accuracy is achieved for the temperature error. For the heat transfer error, the order of accuracy is $q$ rather than $q + 1$. This result is expected because heat transfer is computed from temperature gradients.

Table 4.8: Heat transfer error order of accuracy slopes.

<table>
<thead>
<tr>
<th>$q$ = 1</th>
<th>$q$ = 2</th>
<th>$q$ = 3</th>
<th>$q$ = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.67</td>
<td>1.56</td>
<td>3.12</td>
<td>–</td>
</tr>
<tr>
<td>0.91</td>
<td>1.67</td>
<td>3.09</td>
<td>–</td>
</tr>
<tr>
<td>1.01</td>
<td>1.82</td>
<td>3.61</td>
<td>–</td>
</tr>
<tr>
<td>1.02</td>
<td>1.95</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Chapter 5

Heat Transfer Simulations

The purpose of this chapter is to demonstrate the ability of the Conjugate Heat Transfer (CHT) and Harmonic Balance (HB) methods to capture flow features that are representative of turbomachinery flows. Though the cases presented here are greatly simplified, they can still provide valuable insight into the benefit of the CHT and HB methods.

The first case is a curved duct with conjugate heat transfer along the top surface. This simulation will demonstrate the conjugate heat transfer capability on curved geometries in three dimensions. In addition, the secondary flows created are similar to those that occur in turbomachinery simulations. These features allow for an examination of the effect on the flow field of an accurate thermal boundary representation. Also, simulations without conjugate heat transfer demonstrate the errors associated with adiabatic boundary conditions.

The second section examines the unsteady conjugate heat transfer capability. An unsteady temperature variation imposed on the fluid flow is convected across a flat plate. This type of flow feature is extremely common in turbomachinery flows. Varying thermal conductivity ratios are studied. An adiabatic wall case provides an understanding of the differences in the flow field associated with using conjugate heat transfer.

5.1 3D Curved Duct

A square duct with a 90 degree turn is modeled to demonstrate the CHT capability along curved geometries. The mesh and boundary conditions are depicted in Fig. 5.1. The fluid and solid domains are blue and green, respectively. The square cross section of the fluid has dimensions of $L_{ref} \times L_{ref}$ and the solid cross section
is \( L_{\text{ref}} \times \frac{L_{\text{ref}}}{3} \). The reference length is based upon a Reynolds number of 500. The fluid mesh cross section has \( 6 \times 6 \) cells and the length of the duct is discretized into 60 cells. The solid mesh has \( 6 \times 3 \) cells along the cross section and 40 cells along the length of the duct. The domains are coincident at the fluid-solid interface, with 20 cells along the curved portion of the duct and 20 cells along the straight portions. Both meshes are represented by cubic polynomials \((m = 3)\).

Total conditions are imposed at the maximum \( x \) boundary of the fluid domain. The static pressure and temperature are \( p = 101.35 \text{kPa} \) and \( T = 1000 \text{K} \) and the inlet Mach number is \( M_\infty = 0.1 \). At the maximum \( y \) boundary, a static pressure exit is set to the freestream pressure. Other than the interface boundary, the adiabatic no-slip conditions are applied to the walls of the fluid domain. The top solid domain boundary is set to a specified temperature of \( T_w = 293 \text{K} \). All other solid walls are defined as adiabatic, which is a specification of zero temperature gradient.

The fluid domain is simulated with the laminar Navier-Stokes equations and the solid domain is simulated with the solid conduction equation. Southerland’s formula is used to define the fluid molecular viscosity. The solid thermal conductivity is calculated from a ratio of thermal conductivities as

\[
k_s = \frac{k_s}{k_f} (k_{\text{ref}})^{\text{fluid}}
\]

where the reference fluid conductivity is calculated from the inlet conditions. Simulations with thermal conductivity ratios of \( \frac{k_s}{k_f} = 1, 2, 10, 100 \) are made. In addition, a simulation is performed with the solid domain replaced by an adiabatic boundary condition. Cases with linear, quadratic, and quartic solution polynomials \((q)\) are completed. The CFL is increased each iteration using the residual ramping method (see Section 3.3). The FGMRES tolerance is set to \( 1.0 \times 10^{-12} \) and the number of iterations is limited to 600 for \( q = 1, 2 \) and 800 for \( q = 3 \). The linear system is preconditioned with ILU(1).

Convergence histories for the three polynomial orders and the differing thermal conductivity ratios are shown in Fig. 5.2. The highest order solutions are obtained in less than 60 iterations for all cases. In general, the number of iterations increases for higher thermal conductivity ratios, with the adiabatic case converging fastest. In particular, the \( \frac{k_s}{k_f} = 100, q = 1 \) convergence becomes linear after 10 iterations, while the other cases converge quadratically. This poor convergence is a result of linear solve stopping before convergence is achieved, as indicated in Fig. 5.2b. The \( \frac{k_s}{k_f} = 100 \) curves reach the maximum number of iterations for the majority of the Quasi-Newton sub-iterations. Therefore, the resulting update vectors are not converged to the
desired tolerance (see Fig. 5.2c). Because the update vectors are inaccurate, the theoretical convergence rates (as $CFL \to \infty$) are not guaranteed. Although the $q = 2, 3$ convergence rates are not significantly affected, the $q = 1$ convergence is hampered by the less accurate solution updates.

Contours of temperature, velocities and streamwise vorticity are shown in Fig. 5.3 for the $q = 3, k = 1$ solution. Tangential velocity vectors are shown on the cut surfaces. Two counter-rotating vortices are produced in the turn, as shown by the tangential velocity vectors and the streamwise vorticity. These secondary flows are formed from the boundary layer vorticity along minimum and maximum $z$ walls. The boundary layer vortex lines (initially oriented along the x-axis) tip into the streamwise direction as they pass through the turn (see Grietzer et al. [1]). Because of these vortices, the cold flow near the solid domain is pulled into the center of the passage. The secondary flows dissipate as they move towards the exit of the duct. The turn also causes higher streamwise velocity near the bottom of the duct.

To compare the various solutions, slices are extracted just downstream of the turn (slice 1) and near the exit (slice 2), as depicted in Fig. 5.4. The streamwise vorticity for the three polynomial orders is shown in Fig. 5.5. The vortices are less coherent for the $q = 1$ solution and the velocity magnitudes are smaller. The shape of the vorticity contours are similar for the quadratic and cubic polynomial solutions, though the increased accuracy of the $q = 3$ solution provides a smoother shape. A comparison of temperature and temperature gradient contours is given in Fig. 5.6. The lowest contour line of the $q = 1$ solution does not
Figure 5.2: Iteration history comparing the different thermal conductivity ratios. Simulations using $q = 1, 2, 3$ displayed in succession.
Figure 5.3: Solution for $q = 3$ and $\frac{k_f}{k_f^*} = 1$. 
extend as far into the flow field as the other two solutions. The \( q = 1 \) polynomials cannot adequately resolve the gradients, particularly for \( \frac{\partial T}{\partial z} \). The \( q = 2 \) and \( q = 3 \) solutions capture similar features.

Another indication of the importance of high order accuracy is shown by the heat transfer along the CHT boundary. The CHT interface with coincident cells is conservative, hence the fluxes along both sides of the CHT interface (of a converged simulation) sum to zero. However, the heat transfer that is calculated from the solution does not sum to zero. This non-zero heat transfer is caused by an under-resolved fluid boundary layer. To show this outcome, integrals of the heat transfer along both sides of the interface are calculated. The values from both sides should be equal from a physics standpoint. Percent differences are calculated from the two quantities, as

\[
\dot{q}_{\text{diff}} = 100 \frac{\left| \dot{q}_{\text{fluid}} - \dot{q}_{\text{solid}} \right|}{\dot{q}_{\text{avg}}} \tag{5.2}
\]

where \( \dot{q}_{\text{avg}} \) is the average of the fluid and solid integrated heat transfer. Plots comparing the the computed values for the various solution polynomial orders and thermal conductivity ratios are presented in Fig. 5.7. For the higher order polynomials, the difference between the two sides is reduced as the solution is better resolved. The difference for \( q = 1 \) is very large because of the coarseness of the mesh. As the thermal conductivity ratio increases, the differences also increase. This is caused by the larger thermal gradients in the fluid boundary layer. More cells, higher order polynomials, or clustering the mesh at the wall are necessary to better resolve the large gradients and decrease \( \dot{q}_{\text{diff}} \).
Figure 5.5: Fluid domain streamwise vorticity and tangential velocity vectors for three different polynomial orders, located at slice 1, $\frac{\delta_k}{\kappa_f} = 1$. 
Figure 5.6: Contour lines of temperature and temperature gradients for three different polynomial orders, located at slice 1, $\frac{\kappa_s}{\kappa_f} = 1$. 
Figure 5.7: Percent difference of the integrated heat transfer on both sides of CHT interface.

The adiabatic solution and the CHT solutions with different thermal conductivity ratios are shown in Fig. 5.8. At slice 1, the higher the conductivity ratio, the lower the temperature of the near-solid flow. This is because the heat conducts faster in the solid, causing the interface temperature to be lower. As expected, the adiabatic temperature is constant. Near the exit (Fig. 5.8b), the temperature is lower towards the lower and side boundaries. At the same time, the cold core is hotter. This is caused by the heat transferring from the hotter boundaries into the middle of the flow. Also, the strength of the secondary flows has decreased (compare Fig. 5.8c and 5.8d), which reduces the amount of heat convected from the fluid-solid interface into the main flow path. Temperature gradient and streamwise velocity comparisons are shown in Fig. 5.9. As expected based on the temperature comparisons, the temperature gradients are larger for the higher thermal conductivity ratios. An interesting result is that streamwise velocity varies slightly as the thermal conductivity ratio increases.

The velocity of the fluid flow is affected by heat transferred from the solid domain. To investigate this further, comparison of the adiabatic case and \( \frac{k_f}{k_s} = 100 \) tangential velocity vectors are plotted in Fig. 5.10. Overall, the magnitude of the velocity vectors is greater for the CHT case as compared to the adiabatic case. The difference is most notable near the CHT interface. At slice 4, the differences are not as clear because of the dissipation of the secondary flows. Clearly, the accurate resolution of the thermal boundaries changes both the temperature distribution and the velocity gradients.
Figure 5.8: Contours of temperature and streamwise vorticity for varying thermal conductivity ratios, $q = 3$
Figure 5.9: Contours of temperature gradients and streamwise velocity for varying thermal conductivity ratios, slice 1, \( q = 3 \).
5.2 3D Unsteady Channel Flow

Unsteady heat transfer using the Harmonic Balance method is demonstrated for a channel flow. The mesh and simulation details are shown in Fig. 5.11. The fluid and solid domains are blue and green, respectively. A Reynolds number of 500 is used to define the reference length \( L_{\text{ref}} \). The channel height is \( L_{\text{ref}} \), though only half is simulated by using a symmetry plane. The width of the fluid and solid domains is \( L_{\text{ref}} \) and the length is \( 2L_{\text{ref}} \). The height of the solid domain is \( \frac{L_{\text{ref}}}{4} \). Both domains are discretized into \( 10 \times 10 \times 10 \) cells. At the CHT interface, the two domains are coincident. Linear polynomials \((m = 1)\) are used to represent the mesh.

The freestream pressure and temperature are \( p_\infty = 101.35\text{kPa} \) and \( T_\infty = 1000\text{K} \) and the inlet Mach number is \( M_\infty = 0.1 \). Total conditions are imposed at the inflow boundary, with the flow direction being along the channel axis. The total pressure is obtained from \( p_\infty \) and \( M_\infty \) using the isentropic relations. A time and space varying total temperature profile is imposed, as described by the following equation

\[
T_T(y,z,t) = T_{T\infty} \left( 1 + 0.1 \sum_{n=1}^{2} \exp \left[ -\pi^2 \left( \frac{z}{W} + \frac{t}{\Delta t_{\text{period}}} + k + 2\pi \cos \left( \frac{\pi}{2} - \frac{\pi y}{2H} \right) \right)^2 \right] \right)
\]

where \( T_{T\infty} \) is the total temperature calculated from the isentropic equation, \( W \) is the width of the fluid.
domain, $H$ is the height of the fluid domain, and $\Delta t_{\text{period}}$ is the perturbation passing period, defined as

$$\Delta t_{\text{period}} = t_{\text{ref}} = \frac{L_{\text{ref}}}{u_{\text{ref}}}$$

and $u_{\text{ref}} = M_{\infty} \sqrt{\gamma R_{\text{gas}} T_{\infty}}$. The variation in the $y$ direction was designed such that the profile is symmetric across the channel. The $z$ term provides a variation in the lateral direction. At the exit of the fluid domain, a static pressure boundary condition is imposed. As mentioned previously, a symmetric boundary is specified to allow the simulation of only half of the channel. The constant $z$ boundaries of both the fluid and solid domains are specified as HB phase lag interfaces. A constant temperature is applied to the lower solid boundary, with $T_w = 250$K. The constant $x$ boundaries of the solid are specified as adiabatic.

In the fluid domain, the laminar Navier-Stokes equations are used. In the solid domain, the solid conduction equation is used. The fluid molecular viscosity is defined by Southerland’s formula. As with the curved duct simulations, the solid thermal conductivity is calculated from a ratio of thermal conductivities, shown in Eq. 5.1. Thermal conductivity ratios of $\frac{k_s}{k_f} = 1, 2, 10, 100$ are simulated. In addition, a simulation is performed with the solid domain replaced by an adiabatic boundary condition. Solution using linear and quadratic polynomials ($q$) are completed. Simulations with $K = 1, 2, 3$ are obtained for $\frac{k_s}{k_f} = 1$, while only $K = 1$ is used for the other case. The CFL number is modified each iteration based upon the iterative linear solve method (see Section 3.3). This method is used because the other methods increased the CFL too rapidly, at which point the FGMRES does not converge and the simulations diverge. The FGMRES tolerance is set to $1.0 \times 10^{-10}$ and the number of iterations is limited to 600. The linear system is preconditioned with HB LU-SGS.

The residual convergence histories and CFL values are shown in Fig. 5.12. As with the curved duct
simulations, the lower the thermal conductivity ratio, the faster the convergence, with the adiabatic case
converging the fastest. This behavior is directly related to the stiffness of the linear system. As the ther-
mal conductivity ratio increases, the heat flux into the fluid and the flow gradients increase, making
the linear system stiffer (more difficult to solve). As the stiffness increases, the number of FGMRES iterations
increases. At a certain point, the $CFL$ is limited by Eq. (3.8). In turn, the lower $CFL$ slows the rate of
convergence. The oscillatory behavior of the convergence is a result of the adjustment of $CFL$ up and down
as the number of FGMRES iterations moves above and below $n_d$ (see Section 3.3). These changes in $CFL$
change the iteration count, which changes $CFL$, and so on. The behavior is more pronounced at higher
thermal conductivity ratios because the linear solve is more sensitive to changes in $CFL$. The convergence
history for higher HB frequencies is shown in Fig. 5.13. From initial solution to $K = 3$ requires less than
150 iterations. Much of the frequency content of the perturbation is captured by the $K = 1$ simulation, so
the higher HB frequency cases converge quickly.

Contours of temperature and y-direction heat flux at an instant in time are shown in Fig. 5.14. The con-
tours are plotted on the exit plane, left side of the domain, and the CHT interface location. Near the inlet,
the higher temperature perturbation is clearly visible. As it moves downstream, it dissipates but remains
coherent. The interface temperature near the inlet is strongly influenced by the hot streak. Further down-
stream, the boundary layer becomes thicker and little temperature variation is seen at the CHT interface.
The hot flow perturbation at the inlet causes heat transfer into the colder solid (blue region on the upstream
interface). After the perturbation passes, some heat transfers from the solid back into the flow.

An analysis of the difference in heat transfer on both sides of the CHT interface has been performed.
Although the CHT interface fluxes are conserved (they sum to zero) for coincident cells, the heat transfer
calculated from the solution is non-zero (a product of an under-resolved boundary layer). Equation 5.2 is
used to calculate the percent different of the integrated heat transfer. Values for $\dot{q}_{diff}$ are calculated at all
time levels and averaged. The results for varying thermal conductivity ratios are shown in Fig. 5.15. The
difference generally increases as the thermal conductivity ratio increases. This is a result of the larger tem-
perature gradients in the fluid boundary layer for larger $\frac{k_s}{k_f}$. Better resolution using more cells or polynomials
of higher order is needed to more accurately capture the gradients and reduce the integrated heat transfer
differences.

In order to compare cases with different thermal conductivity ratios, a slice is extracted across the chan-
nel at mid-length of the domain, as shown in Fig. 5.16. From this data, contour plots of several parameters
Figure 5.12: Convergence history for several thermal conductivity ratios with $K = 1$. 

(a) Residual, $\frac{k_s}{k_f} = 1, 2, 10$

(b) Residual, $\frac{k_s}{k_f} = 100$

(c) $CFL$, $\frac{k_s}{k_f} = 1, 2, 10$

(d) $CFL$, $\frac{k_s}{k_f} = 100$
Figure 5.13: Convergence history for $\frac{k_i}{k_f} = 1$ simulations.

Figure 5.14: Contours of temperature and heat flux in the y-direction at $t = 0$, $q = 2$, $K = 1$, $\frac{k_i}{k_f} = 1$. 
Figure 5.15: Percent difference of the integrated heat transfer on both sides of CHT interface. Averaged over all time levels. \( q = 2, K = 1 \).

![Graph showing heat transfer percent difference vs. k/k_s]

Figure 5.16: Location of slice (orange) extracted for comparison.

are made for each case (see Fig. 5.17). The adiabatic case hot streak extends down to the channel wall. As the thermal conductivity ratio increases, the solid domain temperature variation decreases. This causes the fluid to be colder near the wall. Also, large temperature gradients are present at the wall for high \( \frac{k_s}{k_f} \). As expected, the effect of the CHT interaction is not only felt by the temperature, but also by the fluid velocity. In Fig. 5.17c, the velocity along the channel increases for higher thermal conductivity ratios. Also, the boundary layer thickness is reduced. The different conductivity ratios also affect the velocity tangential to the plane, as shown by Fig. 5.17d.

A more detailed comparison of the tangential velocity is made by comparing the extreme cases, adiabatic and \( \frac{k_s}{k_f} = 100 \). This is accomplished at an instant in time, as shown in Fig. 5.18. Note that the solid domain is
Figure 5.17: Solution contours for varying thermal conductivity ratios at an $t = 0$, $q = 2$, $K = 1$
Figure 5.18: Comparison of tangential velocity vectors at $t = 0$. Full channel height. Fluid domain only.

not shown and that the entire height of the channel is displayed. The difference is the most apparent near the middle of the channel. The adiabatic velocity magnitudes are lower than the CHT case. Near the left side of the domain, the adiabatic flow moves towards the symmetry line, whereas the vectors for the $\frac{k_z}{k_f} = 100$ simulation point towards the walls. As was shown with the curved duct, the effect of accurately capturing the thermal boundary conditions on the flow field is very important.
Chapter 6

Computational Cost

The motivation for this work is the improvement of unsteady turbomachinery simulations. The Harmonic Balance (HB) and Conjugate Heat Transfer (CHT) methods have been demonstrated as viable options for achieving this aim. Up to this point, the focus has been on the accuracy of HB and CHT methods, rather than their efficiency. The implementation of these methods has not been optimized for performance, and there are many likely areas for improvement (Chapter 7). Nevertheless, analyzing the computational cost of the HB and CHT methods is necessary to assess the potential for increased efficiency. To that end, the execution times of several cases are presented here along with a general discussion of the memory usage of the DG spatial discretization and the HB method.

6.1 Computational Resources

All of the simulations presented here were conducted on one of two platforms. The first platform is a single computer equipped with two 6-core Intel Xeon X5660 2.8GHz processors and 72Gb of RAM. The second platform is a beowulf cluster consisting of 10 compute nodes. Each node has a single 4-core Intel Core i7-950 3.07GHz processor with 12Gb of RAM. For each test case, the resources used are identified.

The DG-Chimera framework can perform parallel processing in two ways. First, the Boost thread library is used for shared-memory computations. The second approach uses the Message Passing Interface (MPI) standard for distributed memory calculations.
6.2 Execution Time

The execution time is a key indicator of the performance of a CFD solver. The time taken for several verification cases and the heat transfer simulations is presented here. For the HB verification cases, the savings of the HB method as compared to the TM method is examined. For the relative motion study, the extra cost incurred because of the primitive variable approximation is discussed. Finally, the effect of the CHT interface and the ratio of solid to fluid thermal conductivities on the execution time is explored.

For each case, all of the simulations are completed on the same platform. However, in some instances, the simulations are completed with varying numbers of cores. When this occurs, the execution time is better compared using CPU hours. The definition of CPU hours is

\[
\text{CPU hours} = \text{wall clock time} \times \text{number of cores}
\]  

(6.1)

Although, CPU hours is a better metric, it is not independent of the number of cores. Namely, the parallel speedup of the DG-Chimera framework (and most parallel processing) is sub-linear [101]. Both the communication overhead and the amount of serial code contribute to the sub-linear trend. In addition, the preconditioning matrix applies only to the local linear system (it does not account for communication terms), resulting in a higher cost of the linear solve operation as the number of divisions of the computational domain increases. A further reduction in speedup can occur when the computational domain is not partitioned equally. For these reasons, changing the number of cores can affect the CPU hours required for a given simulation.

6.2.1 HB Linear System Preconditioning

A single core of the shared-memory machine was used for each test of the HB preconditioner. The execution times are shown in Fig. 6.1. The HB ILU(0) preconditioner is faster for all orders of accuracy and numbers of frequencies. The performance of HB ILU(0) relative to the other preconditioners improves as \( K \) increases, as shown in Table 6.1. With \( K = 4 \), the HB ILU(0) preconditioner is an order of magnitude faster than the other preconditioners. The only drawback of using HB ILU(0), as discussed in Section 6.3.2, is that it has the highest memory requirement. Interestingly, LU-SGS performs better than ILU(0), which in turn performs better than ILU(1). This trend is the opposite of what is seen with steady simulations.
Figure 6.1: Execution time for several preconditioners, orders of accuracy, and number of HB frequencies.

Table 6.1: Speedup of the HB ILU(0) preconditioner relative to the other preconditioners for spatial high order accuracy.

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>K = 1</th>
<th></th>
<th></th>
<th>K = 2</th>
<th></th>
<th></th>
<th>K = 3</th>
<th></th>
<th></th>
<th>K = 4</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>q = 2</td>
<td>q = 3</td>
<td>q = 4</td>
<td>q = 2</td>
<td>q = 3</td>
<td>q = 4</td>
<td>q = 2</td>
<td>q = 3</td>
<td>q = 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HB LU-SGS</td>
<td>1.7</td>
<td>1.4</td>
<td>1.2</td>
<td>1.7</td>
<td>1.8</td>
<td>1.4</td>
<td>4.2</td>
<td>3.7</td>
<td>3.1</td>
<td>7.3</td>
<td>7.1</td>
<td>7.2</td>
</tr>
<tr>
<td>ILU(1)</td>
<td>2.6</td>
<td>2.1</td>
<td>1.7</td>
<td>5.4</td>
<td>5.2</td>
<td>4.2</td>
<td>13.8</td>
<td>10.7</td>
<td>8.6</td>
<td>31.1</td>
<td>23.5</td>
<td>19.2</td>
</tr>
<tr>
<td>ILU(0)</td>
<td>1.9</td>
<td>1.5</td>
<td>1.4</td>
<td>3.6</td>
<td>3.3</td>
<td>2.7</td>
<td>9.9</td>
<td>7.1</td>
<td>5.4</td>
<td>25.1</td>
<td>17.2</td>
<td>13.6</td>
</tr>
<tr>
<td>LU-SGS</td>
<td>2.6</td>
<td>2.2</td>
<td>1.7</td>
<td>3.7</td>
<td>3.3</td>
<td>2.7</td>
<td>8.1</td>
<td>6.1</td>
<td>4.6</td>
<td>19.6</td>
<td>13.7</td>
<td>11.5</td>
</tr>
</tbody>
</table>
Table 6.2: Execution time (in hours) for the TM and HB simulations of the cascade. TSPP is the number of time steps per period.

<table>
<thead>
<tr>
<th>Time Marching</th>
<th>Harmonic Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSPP</td>
<td>Execution Time (hrs)</td>
</tr>
<tr>
<td>40</td>
<td>8.59</td>
</tr>
<tr>
<td>80</td>
<td>11.33</td>
</tr>
<tr>
<td>160</td>
<td>15.93</td>
</tr>
</tbody>
</table>

6.2.2 Euler Cascade

The cascade simulations were completed on the beowulf cluster. Each simulation utilized all 40 available cores. The execution times are presented in Table 6.2. Section 4.2.3 shows that the TM 80 and 160 time steps per passage solutions have similar temporal resolutions to the HB $K = 2$ and HB $K = 5$ solutions, respectively. For a similar accuracy, the $K = 2$ simulation is approximately 4 times faster than the 80 time step simulation. On the other hand, the $K = 5$ simulation is 4 times slower than the 160 time step simulation. The CFL of all of the HB simulations was limited to maintain the stability of the simulation. Figure 4.27 shows that this limit significantly affects the HB $K = 5$ convergence, which leads to the long simulation time. Improvements to the solve of the HB linear system could alleviate the CFL restriction and reduce the execution time.

The efficiency of the HB simulation would be better highlighted with a multistage turbomachinery simulation. For this cascade case, the perturbation is periodic across one blade passage. A multistage turbomachinery simulation generally requires multiple blade passages per blade row for the TM method, while the HB method needs only a single passage per blade row. Other researchers have shown that the HB method can achieve as much as two orders of magnitude reduction in execution time when compared to the TM method [43, 46, 52].

6.2.3 Shedding Cylinder

The beowulf cluster was used for simulations of the shedding cylinder. Details of the execution time are shown in Table 6.3. Most of the simulations were computed on two nodes, while the HB $K = 6, 7, 8$ simulations used 4 compute nodes. Recall that the HB simulations are initialized from previous HB simulations. Therefore, the HB $K = 2$ simulation was obtained in 97 CPU Hours, which is approximately the same as the TM simulation. The $K = 4$ solution, which is within a drag count of the TM time averaged $C_D$ and 0.02%
Table 6.3: CPU hours and number of cores for the TM and HB simulations of the shedding cylinder.

<table>
<thead>
<tr>
<th>Simulation</th>
<th># of Cores</th>
<th>CPU Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>8</td>
<td>96</td>
</tr>
<tr>
<td>HB ( K = 1 )</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>HB ( K = 2 )</td>
<td>8</td>
<td>87</td>
</tr>
<tr>
<td>HB ( K = 3 )</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>HB ( K = 4 )</td>
<td>8</td>
<td>109</td>
</tr>
<tr>
<td>HB ( K = 5 )</td>
<td>8</td>
<td>208</td>
</tr>
<tr>
<td>HB ( K = 6 )</td>
<td>16</td>
<td>342</td>
</tr>
<tr>
<td>HB ( K = 7 )</td>
<td>16</td>
<td>106</td>
</tr>
<tr>
<td>HB ( K = 8 )</td>
<td>16</td>
<td>141</td>
</tr>
</tbody>
</table>

of the TM \( C_{L,\text{max}} \), requires 266 CPU Hours or roughly 3 times the TM cost. The performance of the HB method would undoubtedly benefit from a gradient based method for finding the fundamental frequency. As discussed in Section 4.2.4, the convergence rate is limited by the use of an inaccurate fundamental frequency.

By comparing the HB \( K = 5 \) and \( K = 6 \) timings, it is clear that a linear speedup is not achieved when increasing the number of cores. Both simulations are initialized from the same solution and converge in roughly the same number of iterations (see Fig. 4.36). However, the \( K = 6 \) simulation (using 16 cores) requires 75% more CPU Hours than the \( K = 5 \) simulation (using 8 cores). A large portion of the added cost is likely related to the preconditioner only being applied to the local linear system.

### 6.2.4 Harmonic Balance Interface Study

All of the HB interface study simulations were completed on two cores of the shared-memory computer. Table 6.4 shows the wall clock time of the conservative variable and primitive variable forms of the fluid flow equations for different solution polynomial orders \( (q) \) and numbers of HB frequencies \( (K) \). Also listed are the primitive to conservative variable ratios for each \( q \) and \( K \). Given that the primitive variable simulations require more iterations to converge (shown in Fig. 3.19), it is not surprising that the execution time is also higher. This higher cost is a result of the approximation of the HB relative motion and phase lag operators for the primitive variables (see Sections 3.4.3 and 3.4.4). The ratio increases as \( q \) increases and as \( K \) decreases. For \( q = 1 \) and \( K = 4, 5 \), the primitive variable version is slightly faster than the conservative variable version. Even at high \( K \), the cost of the approximation is a factor of 3 increase in execution time. An exact form of the HB operators is needed to remove this steep penalty from the primitive variable form of the phase lag and relative motion interfaces.
Table 6.4: Execution time (in minutes) for simulations of the relative motion interface in two dimensions. Simulations for both the conservative and primitive variable forms of the fluid flow equations are shown along with the ratio of the primitive variable to conservative variable execution time.

<table>
<thead>
<tr>
<th></th>
<th>Conservative Variables</th>
<th>Primitive Variables</th>
<th>Primitive/Conservative Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$q = 1$</td>
<td>$q = 2$</td>
<td>$q = 3$</td>
</tr>
<tr>
<td>$K = 1$</td>
<td>0.5</td>
<td>1.8</td>
<td>5.1</td>
</tr>
<tr>
<td>$K = 2$</td>
<td>1.1</td>
<td>3.4</td>
<td>10.4</td>
</tr>
<tr>
<td>$K = 3$</td>
<td>3.5</td>
<td>8.7</td>
<td>25.9</td>
</tr>
<tr>
<td>$K = 4$</td>
<td>9.0</td>
<td>18.0</td>
<td>54.6</td>
</tr>
<tr>
<td>$K = 5$</td>
<td>23.9</td>
<td>69.0</td>
<td>135.6</td>
</tr>
</tbody>
</table>

Table 6.5: CPU hours and number of cores for the simulations of the 3D curved duct.

<table>
<thead>
<tr>
<th>$k_s/k_f$</th>
<th># of Cores</th>
<th>CPU Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q = 1$</td>
<td>6 6 6</td>
<td>0.4 4.8 31.7</td>
</tr>
<tr>
<td>$q = 2$</td>
<td>10 10 10</td>
<td>1.2 4.4 32.6</td>
</tr>
<tr>
<td>$q = 3$</td>
<td>10 10 10</td>
<td>1.4 7.3 36.7</td>
</tr>
<tr>
<td>Adiabatic</td>
<td>10 10 10</td>
<td>7.8 14.8 74.0</td>
</tr>
</tbody>
</table>

6.2.5 3D Curved Duct

Simulations of the curved duct were performed on the shared-memory computer. The execution times for simulations at varying orders of accuracy and thermal conductivity ratios are shown in Table 6.5. It should be noted that the computational domain of the adiabatic case has 2160 cells, whereas the CHT computation domain has 3120 cells. Despite having more cells, the CHT simulations with low $k_s/k_f$ have similar CPU hours for $q = 3$. The cost of the $k_s/k_f = 100$ simulation is nearly three times the adiabatic simulation. At lower $q$, the difference is even greater. The increased cost is related to the increase in the number of Quasi-Newton sub-iterations needed for convergence and the number of FGRMES iterations necessary to obtain the Quasi-Newton update vector (see Fig. 5.2).

6.2.6 3D Unsteady Channel Flow

The simulations for the unsteady channel flow were calculated on the shared-memory machine. Times for these simulations are listed in Table 6.6. The CHT and adiabatic computational domains consist of 2000 and 1000 cells, respectively. The adiabatic simulation is notably faster than the CHT simulations. The $k_s/k_f = 100$ case is particularly slow, being more than costing two orders of magnitude more than the adiabatic case. As
with the curved duct simulations, the long execution times are caused by the decrease in effectiveness of the linear system solve as the thermal conductivity ratio increases. The $CFL$ at each Quasi-Newton sub-iteration was increased or decreased based on the number of FGMRES iterations required to solve the linear system, as specified by Eq. (3.8). For the adiabatic case, the $CFL$ increases rapidly, as shown in Fig. 5.12. On the other hand, the $CFL$ increased much more slowly for the CHT simulations. In particular, the $k_s/k_f = 100$, $q = 2$ simulation $CFL$ remained below 4. Improvements to the linear system solver would allow for higher $CFL$ and shorter simulation times.

### 6.3 Memory Requirements

Two aspects of the HB formulation presented in this work result in high memory usage. First, the complete linearization of the Discontinuous Galerking (DG) spatial discretization requires more memory than implicit finite volume schemes. Second, the implicit HB method needs the full DG linearization of the spatial fluxes for each time level. For both cases, there is a trade-off. The DG discretization trades memory usage for high-order accuracy and fewer sub-iterations. The HB method trades memory usage for faster execution times when compared to TM methods. In addition, the preconditioning matrices that better condition the linear system are generally more memory intensive. A discussion of the memory needs of the algorithms used are presented here and suggestions for reducing the memory requirements are detailed in Section 7.1.
6.3.1 Discontinuous Galerkin

The linearization of the DG fluxes results in a sparse block matrix. Each of these blocks is dense. The number of floating point numbers (denoted as FPN) that must be stored in memory for each block is

\[(\text{FPN})_{\text{block}} = \left(N_{eq}(q+1)^d\right)^2\] (6.2)

where \(d\) is the dimension and \(N_{eq}\) is the number of equations (i.e. 5 for the laminar Navier-Stokes equations in three dimensions). Because only structured meshes are used, the matrix is in a block diagonal form, with each diagonal requiring a storage of

\[(\text{FPN})_{\text{diag}} = N_{cell}(\text{FPN})_{\text{block}}\] (6.3)

where \(N_{cell}\) is the number of cells in the domain. In one, two, and three dimensions the matrix is a block tri-, penta-, and hepta-diagonal, respectively. The off-diagonals have fewer than \(N_{cells}\) blocks, but a conservative estimate for the matrix storage can be given as

\[(\text{FPN})_A = (2d + 1)(\text{FPN})_{\text{diag}}\] (6.4)

In addition to the matrix, the FGMRES iterative solver uses a number solutions vectors, requiring the storage of

\[(\text{FPN})_x = N_{\text{FGMRES}}N_{cell}N_{eq}(q+1)^d\] (6.5)

where \(N_{\text{FGMRES}}\) is the number of FGMRES iterations. In general, \((\text{FPN})_A\) is larger than \((\text{FPN})_x\) because it increases quadratically with respect to \(N_{eq}(q+1)^d\), whereas \((\text{FPN})_x\) increases linearly. Only for simulations with a low order of accuracy and many FGMRES iterations will the vector storage be greater than the matrix storage.

The last significant memory requirement comes from the preconditioner. The number of floating point numbers stored for the LU-SGS and ILU(m) preconditioners is

\[\begin{align*}
(\text{FPN})_{\text{LU-SGS}} & = (\text{FPN})_{\text{diag}} \\
(\text{FPN})_{\text{ILU}(m)} & = (2d + 1)(\text{FPN})_{\text{diag}} + 2m(\text{FPN})_{\text{diag}}
\end{align*}\] (6.6)
where \( m \) is the number of diagonals of fill-in for the incomplete LU preconditioning matrix. While the LU-SGS storage is only one more diagonal, the ILU\((m)\) storage is equal to (for \( m = 0 \)) or greater than the matrix storage. As with Eq. (6.4), Eq. (6.6) is a conservative estimate because the fill-in diagonals are smaller than the main diagonal (i.e. less than \( N_{\text{cell}} \) blocks of storage). Nonetheless, the trade-off between lower memory usage, with LU-SGS, and greater effectiveness, with ILU\((m)\), is apparent.

### 6.3.2 Harmonic Balance

The price of a more efficient HB method for periodic unsteadiness is an increase in the memory usage. The memory usage is related to the number of HB times levels (denoted as \( N_{HB} \) here, and \( N \) in other sections), which is in turn related to the number of HB frequencies as \( N_{HB} = 2K + 1 \). Therefore, the accuracy of the HB simulation is increased (with increased \( K \)) at the expense of larger storage costs. Storing the entire HB linearization matrix (Eq. (2.107)) requires

\[
(FPN)_{A^*} = N_{HB} (FPN)_A + N_{HB} (N_{HB} - 1) (FPN)_{\text{diagonal}}
\]

The storage increases quadratically with the number of times levels. Instead, the entire pseudo-spectral linearization is not stored directly (see Section 3.4.1.1), reducing the cost to

\[
(FPN)_{A^*} = N_{HB} (FPN)_A + (FPN)_{\text{diagonal}}
\]

which scales linearly with \( N_{HB} \). The storage for the FGMRES update vectors is simply

\[
(FPN)_{x^*} = N_{HB} (FPN)_x
\]

which also scales linearly with \( N_{HB} \).

As discussed in Section 3.4.2, the effectiveness of traditional preconditioners noticeably decreases when applied to the HB linear system of equations. The cost of the “spatial-only” preconditioners, which precondition the spatial linearization matrix of each time level independently, scales linearly with the number of
time levels, as shown below

\[
(FPN)_{LU-SGS,HB} = N_{HB} (FPN)_{LU-SGS} \tag{6.10}
\]

\[
(FPN)_{ILU(m),HB} = N_{HB} (FPN)_{ILU(m)} \tag{6.11}
\]

For the HB LU-SGS preconditioner, the storage cost scaling is also linear

\[
(FPN)_{HBLU-SGS} = N_{HB} (FPN)_{LU-SGS} \tag{6.12}
\]

because only the inverse of the diagonal is stored for each spatial linearization. On the other hand, the memory usage of the HB ILU(0) precondition is

\[
(FPN)_{HBLU(0)} = N_{HB} (FPN)_{ILU(0)} + N_{HB} (N_{HB} - 1) (FPN)_{diagonal} \tag{6.13}
\]

which scales quadratically with $N_{HB}$. In three dimensions, the quadratic term is less than the linear term for $K < 4$ and less than twice the linear term for $K < 8$. The former inequality is often true and the later is almost always true for multistage turbomachinery simulations. Therefore, the significant reduction in time when using HB ILU(0), shown in Section 6.2.1, is almost certainly worth the additional memory usage.
Chapter 7

Conclusion

Implementation and simulation of the Harmonic Balance and Conjugate Heat Transfer methods are presented in this dissertation. The HB method is an effective way to perform periodic unsteady simulations, and, the fully coupled heat transfer approach provides a stable and accurate representation of fluid-solid heat transfer. These methods are added to a Chimera capable high-order accurate Discontinuous Galerkin framework, which allows modeling complex geometries with high spatial accuracy. The combination of HB and CHT provides a computational tool that can accurately capture unsteady heat transfer. These methods have applications to turbomachinery flows where unsteadiness is known to increase the heat transfer, but the effects are often not studied in detail because the tools are not readily available. The simulation and understanding of other unsteady fluids problems, such as heat exchangers, could also benefit from this computational framework.

A first of its kind, fully implicit, high-order Harmonic Balance method has been implemented. The form and solution strategy for the coupled linear system of equations has been studied. A new HB preconditioner improves the speed of the linear solve and further improvements are suggested. Demonstration of the HB method on several verification cases proves that it accurately captures unsteady flow physics. This implicit HB implementation is shown to provide high-order unsteady solutions in under 400 iterations (Section 4.2.3). Comparable simulations with other CFD software will take thousands of iterations to converge within larger residual tolerances. In addition, natural unsteady phenomena, such as a shedding cylinder, can be accurately captured. The HB capabilities are further enhanced with the phase lag and relative motion boundaries. This implementation is not the first high-order fully implicit relative motion interface; however, it is the first to be high-order, fully implicit, and three dimensional, while also allowing for adjacent domains.
with non-periodic pitch. A five frequency, fifth order accurate relative motion solution can be obtained in less than 200 iterations. In addition, the HB method can obtain unsteady solutions in a fraction of the time of comparable solution by a TM method (Section 6.2.2). With the improvements suggested in Section 7.1.3, the computational efficiency of the HB method can be further increased.

The fully coupled heat transfer approach is verified using conduction and convection test cases. Simulations with thermal conductivity ratios as high as 10,000 are computed without a loss of stability. These simulations converge in less than 30 iterations using the Quasi-Newton approach. The appropriate orders of accuracy are achieved for convection flows with the CHT interface. The ability of the chimera interface to connect the fluid and solid domains is studied for polynomial and non-polynomial shapes. To the author’s knowledge, the 3D square duct simulation is the first demonstration of a curved cell capable CHT method in three dimensions. A solution is obtained in less than 60 iterations. Comparisons with an adiabatic simulation illustrate the importance of an accurate thermal boundary. Convergence plots show that the stiffness of the linear system is directly related to the solid to fluid thermal conductivity ratio. Further investigation of methods for improving the linear solve would benefit CHT simulations.

Finally, the combination of the HB and CHT methods is shown in three dimensions. A three dimensional, unsteady total temperature variation is imposed at the inlet of a channel. The high-order solution is obtained in approximately 100 iterations with the Quasi-Newton method. As with the duct simulation, the deficiencies of the adiabatic boundary condition are apparent.

7.1 Future Work

The demonstrations of the HB and CHT methods are encouraging. Several areas in which further development of the computational framework would be beneficial are discussed in this Section.

7.1.1 Turbulence Model

The implemented Spalart-Allmaras model is a good starting point for capturing turbulent flows. However, other turbulence models have proven to be more accurate at capturing turbulent heat transfer. The DG discretization of other turbulence models is an active area of research. Utilizing newly arriving DG formulations of turbulence models could allow for more accurate heat transfer simulations. The S-A model and many other turbulence models rely on the calculation of a minimum wall distance. The current method
for calculating this distance is robust, but slow. Utilizing better methods for calculating the wall distance
(possibly solving the Eikonal equation) should be investigated.

7.1.2 Non-Reflective Boundaries

A known deficiency of the unsteady simulations presented in this work is the reflective nature of the boundary conditions. This deficiency not only causes spurious reflections, but it can also degrade the convergence rate and reduce the stability of unsteady simulations. For these reasons, the addition of non-reflective boundaries would improve the potential of the HB implementation.

7.1.3 Computational Performance

The largest amount of simulation time is spent solving the linear system. Even modest improvements to the linear solve could greatly benefit the computational performance. Several test cases in this dissertation have shown that in certain situations, the solution of the linear system with the current methods can be a challenge. Within this work, only the FGMRES algorithm and the ILU and LU-SGS preconditioning matrices have been used. The effectiveness of other algorithms should be explored. In addition, the artificial boundary and interface contributions to the linear system are not considered when formulating the preconditioner. Including these terms would likely increase the linear solve efficiency for simulations with CHT and HB interfaces. Another possible avenue for speeding up the linear solve is the use of graphics processing units (GPUs) to compute the matrix multiplication and inversion operations.

As discussed in Section 3.4.2, the traditional preconditioners are ineffective at solving the coupled HB linear system, particularly for high numbers of frequencies. The implemented HB ILU(0) preconditioner is a significant step in improving the speed of the linear system solve, but more advances are possible. As suggested, a rearrangement of the matrix could simultaneously increase the effectiveness and reduce the complexity of the preconditioner. For many of the HB test cases, the CFL number is reduced to allow the linear solve to converge to the desired tolerance. By increasing the speed of the linear solve, the CFL number can be increased, thereby increasing the rate of convergence. Application of multi-grid methods to the steady-like HB equations is another way to increase the convergence rate.

A challenge posed by the fully implicit DG linear system is the high memory cost. This cost is further increased by the HB method, which trades increased memory usage for increased accuracy (more frequen-
cies). The memory required by the HB method cannot be easily reduced. However, advancements in the area of DG methods (namely the Hybridized DG (HDG) and Embedded DG (EDG) methods) provide a mechanism for decreasing the size of the linear system.

7.1.4 HB Phase Lag and Relative Motion Interfaces

Several limitations exist with the current implementation of the HB phase lag and relative motion interfaces. The approximation used for the primitive variables version of the interface limits the convergence rate. By using an exact formulation, the convergence rate of the conservative variables version could be reached. Another possible improvement is to implement an annular version of the relative motion interface. Also, restructuring the communication setup procedure would allow domains with relative motion interfaces to overlap.

7.1.5 Chimera Connection at Conjugate Heat Transfer Interfaces

Section 3.5.2 describes the potential for gaps along the CHT interface. This problem occurs along geometry derived from non-polynomial functions. The two methods to remove the gap are to make the cells coincident on either side, or to overlap the domains. The first strictly limits the ability to mesh complex geometries, while the second is non-physical. Investigations should be made into methods for meshing the geometry in a way that does not produce gaps. Alternatively, studies of the effect of overlapping the fluid and solid domains could be used to determine how much, if any, overlap is acceptable.
Bibliography


Appendix A

Fluid Flow Jacobian Matrices

The linearization of the governing equations necessary for Newton’s method (Section 2.1.1.1), which requires the calculation of Jacobian matrices. These Jacobian matrices are shown here for the fluxes, sources, and boundary conditions of the primitive variable fluid flow equations. The Jacobian matrices are calculated analytically. Consistent use of the chain rule is used to reduce the complexity of the derivations.

A.1 Fluxes

A.1.1 Advective Flux

The Jacobian of the advective flux (Eq. 2.29) is

$$
\frac{\partial \tilde{F}_a^u (U)}{\partial U} = \begin{bmatrix}
\frac{\partial F_a^u}{\partial U} & \frac{\partial F_a^v}{\partial U} & \frac{\partial F_a^w}{\partial U} \\
\frac{\partial \tilde{F}_a^u}{\partial U} & \frac{\partial \tilde{F}_a^v}{\partial U} & \frac{\partial \tilde{F}_a^w}{\partial U}
\end{bmatrix}
$$

where the individual components are

$$
\frac{\partial F_a^u}{\partial U} = \begin{bmatrix}
\rho & 0 & 0 & \frac{\partial \rho}{\partial u} u & \frac{\partial \rho}{\partial T} u & 0 \\
2\rho u & 0 & 0 & \frac{\partial \rho}{\partial u} u^2 + 1 & \frac{\partial \rho}{\partial T} u^2 & 0 \\
\rho v & \rho u & 0 & \frac{\partial \rho}{\partial u} uv & \frac{\partial \rho}{\partial T} uv & 0 \\
\rho w & 0 & \rho u & \frac{\partial \rho}{\partial u} uw & \frac{\partial \rho}{\partial T} uw & 0 \\
\rho (H + u \frac{\partial H}{\partial u}) & \rho u \frac{\partial H}{\partial v} & \rho u \frac{\partial H}{\partial w} & \frac{\partial \rho}{\partial u} uH & u \left( \frac{\partial \rho}{\partial T} H + \rho \frac{\partial H}{\partial T} \right) & 0 \\
\rho \tilde{v} & 0 & 0 & \frac{\partial \rho}{\partial u} \tilde{v} & \frac{\partial \rho}{\partial T} \tilde{v} & \rho u
\end{bmatrix}
$$

(A.2)
\[ \frac{\partial F_y}{\partial U} = \begin{bmatrix}
0 & \rho & 0 & \frac{\partial \rho}{\partial p} v & \frac{\partial \rho}{\partial T} v & 0 \\
\rho v & \rho u & 0 & \frac{\partial \rho}{\partial p} u & \frac{\partial \rho}{\partial T} u & 0 \\
0 & 2\rho v & 0 & \frac{\partial \rho}{\partial p} v^2 + 1 & \frac{\partial \rho}{\partial T} v^2 & 0 \\
0 & \rho w & \rho v & \frac{\partial \rho}{\partial p} w v & \frac{\partial \rho}{\partial T} w v & 0 \\
\rho v \frac{\partial H}{\partial u} & \rho \left(H + v \frac{\partial H}{\partial v}\right) & \rho v \frac{\partial H}{\partial w} & \frac{\partial \rho}{\partial p} v H & \rho \left(\frac{\partial \rho}{\partial T} H + \rho \frac{\partial H}{\partial T}\right) & 0 \\
0 & \rho \tilde{v} & 0 & \frac{\partial \rho}{\partial p} v \tilde{v} & \frac{\partial \rho}{\partial T} v \tilde{v} & \rho v \\
\end{bmatrix} \] (A.3)

\[ \frac{\partial F_z}{\partial U} = \begin{bmatrix}
0 & 0 & \rho & \frac{\partial \rho}{\partial p} w & \frac{\partial \rho}{\partial T} w & 0 \\
\rho w & 0 & \rho u & \frac{\partial \rho}{\partial p} w u & \frac{\partial \rho}{\partial T} w u & 0 \\
0 & \rho w & \rho v & \frac{\partial \rho}{\partial p} w v & \frac{\partial \rho}{\partial T} w v & 0 \\
0 & 0 & 2\rho w & \frac{\partial \rho}{\partial p} w^2 + 1 & \frac{\partial \rho}{\partial T} w^2 & 0 \\
\rho w \frac{\partial H}{\partial u} & \rho w \frac{\partial H}{\partial v} & \rho \left(H + w \frac{\partial H}{\partial w}\right) & \frac{\partial \rho}{\partial p} w H & \rho \left(\frac{\partial \rho}{\partial T} H + \rho \frac{\partial H}{\partial T}\right) & 0 \\
0 & 0 & \rho \tilde{v} & \frac{\partial \rho}{\partial p} w \tilde{v} & \frac{\partial \rho}{\partial T} w \tilde{v} & \rho w \\
\end{bmatrix} \] (A.4)

and the derivatives of \( \rho \) and \( H \) with respect to the primitive variables are

\[
\frac{\partial \rho}{\partial p} = \frac{1}{R_{\text{gas}} T} \\
\frac{\partial \rho}{\partial T} = -\frac{p}{R_{\text{gas}} T^2} \quad (A.5)
\]

and

\[
\frac{\partial H}{\partial \tilde{v}} = \tilde{v} \\
\frac{\partial H}{\partial T} = C_p \quad (A.6)
\]

### A.1.2 Diffusive Flux

The linearization of the diffusive fluxes requires both Jacobian matrices with respect to the primitive variables and their gradients. The diffusive flux derivatives (derived from Eq. 2.32) with respect to the velocity
vector is
\[
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\tau_{x'x'} \\
0
\end{bmatrix}
\]
(A.7)

where \( \vec{F}_d^i \) is the \( i \)th component of the diffusive flux vector and \( \vec{V}_j \) is the \( j \)th component of the velocity vector.

Using chain rule, the derivatives with respect to \( p, T, \) and \( \vartheta \) have the same form. Letting
\[
A = \begin{bmatrix} T & p & \vec{v} \end{bmatrix}^T,
\]
these derivatives are
\[
\frac{\partial \vec{F}_d}{\partial A} = \begin{bmatrix}
0 & 0 & 0 \\
\frac{\partial \tau_{xx}}{\partial A} & \frac{\partial \tau_{xy}}{\partial A} & \frac{\partial \tau_{xz}}{\partial A} \\
\frac{\partial \tau_{yx}}{\partial A} & \frac{\partial \tau_{yy}}{\partial A} & \frac{\partial \tau_{yz}}{\partial A} \\
\frac{\partial \tau_{zx}}{\partial A} & \frac{\partial \tau_{zy}}{\partial A} & \frac{\partial \tau_{zz}}{\partial A} \\
\frac{\partial \beta_x}{\partial A} & \frac{\partial \beta_y}{\partial A} & \frac{\partial \beta_z}{\partial A}
\end{bmatrix}
\]
(A.8)

and the derivative of the stress terms and the work and heat transfer terms are
\[
\frac{\partial \tau_{xx}}{\partial A} = 2 \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) t_x + (\frac{\partial \lambda_1}{\partial A} + \frac{\partial \lambda_{1r}}{\partial A}) V \cdot \vec{V}
\]
\[
\frac{\partial \tau_{xy}}{\partial A} = 2 \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) t_y + (\frac{\partial \lambda_1}{\partial A} + \frac{\partial \lambda_{1r}}{\partial A}) V \cdot \vec{V}
\]
\[
\frac{\partial \tau_{xz}}{\partial A} = 2 \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) t_z + (\frac{\partial \lambda_1}{\partial A} + \frac{\partial \lambda_{1r}}{\partial A}) V \cdot \vec{V}
\]
\[
\frac{\partial \tau_{yy}}{\partial A} = \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) (u_y + v_x)
\]
\[
\frac{\partial \tau_{yx}}{\partial A} = \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) (u_x + v_y)
\]
\[
\frac{\partial \tau_{zx}}{\partial A} = \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) (u_z + w_x)
\]
\[
\frac{\partial \tau_{zy}}{\partial A} = \left( \frac{\partial \mu}{\partial A} + \frac{\partial \mu_r}{\partial A} \right) (v_z + w_y)
\]
\[
\frac{\partial \beta_x}{\partial A} = \left( \frac{\partial k}{\partial A} + \frac{\partial k_r}{\partial A} \right) \frac{\partial T}{\partial x^i} u_i + \frac{\partial \tau_{xx}}{\partial A} + \frac{\partial \tau_{xy}}{\partial A} v + \frac{\partial \tau_{xz}}{\partial A} w \tag{A.9}
\]
with the derivatives of the SA flux being

\[
\begin{align*}
\frac{\partial \sigma_{\chi}}{\partial p} &= \frac{1}{\sigma} \left( \frac{\partial f_n}{\partial p} \rho \frac{\partial \chi}{\partial x^i} + f_n \frac{\partial \rho}{\partial p} \frac{\partial \frac{\partial \chi}{\partial x^i}}{\partial \chi} \right) \\
\frac{\partial \sigma_{\chi}}{\partial T} &= \frac{1}{\sigma} \left( \frac{\partial f_n}{\partial T} \rho \frac{\partial \chi}{\partial x^i} + f_n \frac{\partial \rho}{\partial T} \frac{\partial \frac{\partial \chi}{\partial x^i}}{\partial \chi} + \frac{\partial \mu}{\partial T} \frac{\partial \frac{\partial \chi}{\partial x^i}}{\partial \chi} \right) \\
\frac{\partial \sigma_{\chi}}{\partial \chi} &= \frac{1}{\sigma} \left( \frac{\partial f_n}{\partial \chi} \rho \frac{\partial \chi}{\partial x^i} + f_n \frac{\partial \rho}{\partial \chi} \frac{\partial \frac{\partial \chi}{\partial x^i}}{\partial \chi} \right)
\end{align*}
\]  

(A.10)

To complete the Jacobian matrix, derivatives of the laminar and turbulent viscosities and thermal conductivities are needed. The derivatives of the laminar quantities with respect to \( p, T, \) and \( \chi \) are

\[
\begin{align*}
\frac{\partial k}{\partial \Delta} &= C_p \frac{\partial \mu}{\partial \Delta} \\
\frac{\partial \lambda_1}{\partial \Delta} &= \frac{2}{3} \frac{\partial \mu}{\partial \Delta} \\
\frac{\partial \mu}{\partial p} &= \frac{\partial \mu}{\partial \chi} = 0 \\
\frac{\partial \mu}{\partial T} &= \mu_0 \left[ \frac{3 T^{1/2} (T_0 + C_s)}{2 T_0^{3/2} (T + C_s)} - \left( \frac{T}{T_0} \right)^{3/2} \frac{(T_0 + C_s)^2}{(T + C_s)^2} \right]
\end{align*}
\]  

(A.11)

and the turbulent viscosities and conductivities are

\[
\begin{align*}
\frac{\partial k_t}{\partial \Delta} &= C_p \frac{\partial \mu_t}{\partial \Delta} \\
\frac{\partial \lambda_{1t}}{\partial \Delta} &= \frac{2}{3} \frac{\partial \mu_t}{\partial \Delta} \\
\frac{\partial \mu_t}{\partial p} &= \frac{\partial \rho}{\partial p} \frac{\partial v_i}{\partial p} + \rho \frac{\partial v_i}{\partial p} \\
\frac{\partial \mu_t}{\partial T} &= \frac{\partial \rho}{\partial T} \frac{\partial v_i}{\partial T} + \rho \frac{\partial v_i}{\partial T} \\
\frac{\partial \mu_t}{\partial \chi} &= \rho \frac{\partial v_i}{\partial \chi}
\end{align*}
\]  

(A.12)

From the description of the SA model in Section 2.2.2.2, the turbulent kinematic viscosity derivatives are
determined to be

\[
\frac{\partial \nu_t}{\partial p} = \begin{cases} 
\frac{\partial f_{n1}}{\partial \chi} \frac{\partial \chi}{\partial p} & \tilde{\nu} \geq 0 \\
0 & \tilde{\nu} < 0 
\end{cases}
\]

\[
\frac{\partial \nu_t}{\partial T} = \begin{cases} 
\frac{\partial f_{n1}}{\partial \chi} \frac{\partial \chi}{\partial T} & \tilde{\nu} \geq 0 \\
0 & \tilde{\nu} < 0 
\end{cases}
\]

\[
\frac{\partial \nu_t}{\partial \tilde{\nu}} = \begin{cases} 
\frac{\partial f_{n1}}{\partial \chi} \frac{\partial \chi}{\partial \tilde{\nu}} + f_{n1} & \tilde{\nu} \geq 0 \\
0 & \tilde{\nu} < 0 
\end{cases}
\]  \quad (A.13)

with

\[
\frac{\partial f_{n1}}{\partial \chi} = \left[ \frac{3c_{n1}^3 \chi^2}{(\chi^3 + c_{n1}^3)^2} \right] 
\]  \quad (A.14)

and

\[
\frac{\partial \chi}{\partial p} = \frac{1}{\mu} \frac{\partial \rho}{\partial p} \\
\frac{\partial \chi}{\partial T} = \frac{1}{\mu} \frac{\partial \rho}{\partial T} - \frac{\partial \mu}{\partial T} \frac{\rho}{\mu^2} \\
\frac{\partial \chi}{\partial \tilde{\nu}} = \frac{\rho}{\mu} 
\]  \quad (A.15)

and finally the derivatives of \( f_{n1} \) are

\[
\frac{\partial f_{n1}}{\partial A} = \begin{cases} 
0 & \tilde{\nu} \geq 0 \\
\frac{\partial \chi}{\partial A} \left[ \frac{6c_{n1}^2 \chi^2}{(c_{n1} - \chi)^2} \right] & \tilde{\nu} < 0 
\end{cases}
\]  \quad (A.16)
The gradient Jacobian of the diffusive flux is

$$\frac{\partial \vec{F}^d}{\partial \nabla \vec{U}} = \begin{bmatrix} 0 \\ \frac{\partial \tau_{ij}}{\partial \nabla \vec{U}} \\ \frac{\partial \tau_{ij}}{\partial \nabla \vec{U}} \\ \frac{\partial \tau_{ij}}{\partial \nabla \vec{U}} \\ \frac{\partial \beta_{ij}}{\partial \nabla \vec{U}} \\ \frac{\partial \sigma_{ij}}{\partial \nabla \vec{U}} \end{bmatrix} \tag{A.17}$$

Beginning with the derivatives with respect to the velocity gradients, the stress term derivatives are

$$\frac{\partial \tau_{ij}}{\partial \nabla \vec{V}_j} = \begin{cases} \frac{4}{3} (\mu + \mu_t) & i = j \\ (\lambda_1 + \lambda_1 t) & i \neq j \end{cases}$$

$$\frac{\partial \tau_{ij}}{\partial \nabla \vec{V}_i} = \frac{\partial \tau_{ij}}{\partial \nabla \vec{V}_j} = (\mu + \mu_t) \text{ for } i \neq j \tag{A.18}$$

where $\nabla \vec{V}_j = \frac{\partial \vec{V}_j}{\partial \tau}$. The work and heat transfer term velocity gradient derivatives are

$$\frac{\partial \beta_{ij}}{\partial \nabla \vec{V}} = u \frac{\partial \tau_{ij}}{\partial \nabla \vec{V}} + v \frac{\partial \tau_{ij}}{\partial \nabla \vec{V}} + w \frac{\partial \tau_{ij}}{\partial \nabla \vec{V}} \tag{A.19}$$

which is completed by using the derivatives in Eq. (A.18). The pressure gradient derivatives are zero. The temperature gradient derivatives are zero for all terms except the work and heat transfer term, as shown below

$$\frac{\partial \beta_{ij}}{\partial \nabla T} = (k + k_t) \tag{A.20}$$

Similarly, the diffusive flux derivatives with respect to the gradient of the SA variable are zero except for the SA diffusive flux, the derivatives of which are

$$\frac{\partial \sigma_{ij}}{\partial \nabla \vec{V}} = \frac{1}{\sigma} (\mu + f_{n1} \rho \vec{V}) \tag{A.21}$$
A.2 Source Term

The source term included in the governing fluid flow equations is defined in Eq. (2.37), which only includes the source applied to the SA equation. The Jacobian matrix and gradient Jacobian matrix are

\[
\frac{\partial S(U, \nabla U)}{\partial U} = \begin{bmatrix}
0 \\
0 \\
0 \\
\frac{\partial S_{\nu}}{\partial U}
\end{bmatrix} \tag{A.22}
\]

and

\[
\frac{\partial S(U, \nabla U)}{\partial \nabla U} = \begin{bmatrix}
0 \\
0 \\
0 \\
\frac{\partial S_{\nu}}{\partial \nabla U}
\end{bmatrix} \tag{A.23}
\]

To simplify the Jacobian derivation, Eq. (2.37) will be split into three equations such that

\[
S_{\nu} = S_I + S_{II} + S_{III}
\]

\[
S_I = -\rho (P - D)
\]

\[
S_{II} = -\frac{c_{\nu2}}{\sigma} \rho \nabla \nu \cdot \nabla \nu
\]

\[
S_{III} = \frac{1}{\sigma} (v + f_{n1} \nu) \nabla \rho \cdot \nabla \nu \tag{A.24}
\]

Because differentiation is a linear operation, the following is true

\[
\frac{\partial S_{\nu}}{\partial U} = \frac{\partial S_I}{\partial U} + \frac{\partial S_{II}}{\partial U} + \frac{\partial S_{III}}{\partial U}
\]

\[
\frac{\partial S_{\nu}}{\partial \nabla U} = \frac{\partial S_I}{\partial \nabla U} + \frac{\partial S_{II}}{\partial \nabla U} + \frac{\partial S_{III}}{\partial \nabla U}
\]
A.2.1 Jacobian

The Jacobian of $S_I$ is

$$\frac{\partial S_I}{\partial U} = \begin{cases} \frac{\partial S_I}{\partial \nu} = 0 \\ \frac{\partial S_I}{\partial p} = \frac{\partial p}{\partial p} (D - P) + \rho \left( \frac{\partial D}{\partial p} - \frac{\partial P}{\partial p} \right) \\ \frac{\partial S_I}{\partial T} = \frac{\partial p}{\partial T} (D - P) + \rho \left( \frac{\partial D}{\partial T} - \frac{\partial P}{\partial T} \right) \\ \frac{\partial S_I}{\partial \nu} = \rho \left( \frac{\partial D}{\partial \nu} - \frac{\partial P}{\partial \nu} \right) \end{cases}$$  \hspace{1cm} (A.25)

where the production derivatives are

$$\frac{\partial P}{\partial p} = \begin{cases} c_{b1} \left[ (1 - \frac{\partial f_{t2}}{\partial \chi} \frac{\partial \chi}{\partial p}) \tilde{Z} \nu + (1 - f_{t2}) \frac{\partial \tilde{Z}}{\partial p} \tilde{\nu} \right] & \tilde{\nu} \geq 0 \\ \begin{array}{c} 0 \\ \tilde{\nu} < 0 \end{array} \end{cases}$$

$$\frac{\partial P}{\partial T} = \begin{cases} c_{b1} \left[ (1 - \frac{\partial f_{t2}}{\partial \chi} \frac{\partial \chi}{\partial T}) \tilde{Z} \nu + (1 - f_{t2}) \frac{\partial \tilde{Z}}{\partial T} \tilde{\nu} \right] & \tilde{\nu} \geq 0 \\ \begin{array}{c} 0 \\ \tilde{\nu} < 0 \end{array} \end{cases}$$

$$\frac{\partial P}{\partial \nu} = \begin{cases} c_{b1} \left[ (1 - \frac{\partial f_{t2}}{\partial \nu} \frac{\partial \nu}{\partial \nu}) \tilde{Z} \nu + (1 - f_{t2}) \left( \frac{\partial \tilde{Z}}{\partial \nu} \tilde{\nu} + \tilde{Z} \right) \right] & \tilde{\nu} \geq 0 \\ c_{b1} (1 - c_{f3}) \tilde{Z} & \tilde{\nu} < 0 \end{cases}$$  \hspace{1cm} (A.26)

and the destruction derivatives are

$$\frac{\partial D}{\partial p} = \begin{cases} \left( c_{w1} \frac{\partial f_{w}}{\partial p} - c_{w3} \frac{\partial f_{w}}{\partial \chi} \frac{\partial \chi}{\partial p} \right) \left[ \frac{\nu}{\tilde{\nu}} \right]^2 & \tilde{\nu} \geq 0 \\ \begin{array}{c} 0 \\ \tilde{\nu} < 0 \end{array} \end{cases}$$

$$\frac{\partial D}{\partial T} = \begin{cases} \left( c_{w1} \frac{\partial f_{w}}{\partial T} - c_{w3} \frac{\partial f_{w}}{\partial \chi} \frac{\partial \chi}{\partial T} \right) \left[ \frac{\nu}{\tilde{\nu}} \right]^2 & \tilde{\nu} \geq 0 \\ \begin{array}{c} 0 \\ \tilde{\nu} < 0 \end{array} \end{cases}$$

$$\frac{\partial D}{\partial \nu} = \begin{cases} \left( c_{w1} \frac{\partial f_{w}}{\partial \nu} - c_{w3} \frac{\partial f_{w}}{\partial \chi} \frac{\partial \chi}{\partial \nu} \right) \left[ \frac{\nu}{\tilde{\nu}} \right]^2 + 2 \left( c_{w1} f_{w} - c_{w3} f_{w} \right) \frac{\tilde{\nu}}{\tilde{\nu}} & \tilde{\nu} \geq 0 \\ -2c_{w1} \frac{\tilde{\nu}}{\tilde{\nu}} & \tilde{\nu} < 0 \end{cases}$$  \hspace{1cm} (A.27)

where the derivatives of $\chi$ are given in Eq. (A.15) and

$$\frac{\partial f_{t2}}{\partial \chi} = -2c_{f3} c_{f4} \chi \exp \left( -c_{f4} \chi^2 \right)$$
Using chain rule, the derivatives of \( f_w \) are

\[
\begin{align*}
\frac{\partial f_w}{\partial p} &= \frac{\partial f_w}{\partial g} \frac{\partial g}{\partial r} \frac{\partial \bar{r}}{\partial \bar{Z}} \frac{\partial \bar{r}}{\partial \bar{r}} \frac{\partial f_{v2}}{\partial \chi} \\
\frac{\partial f_w}{\partial T} &= \frac{\partial f_w}{\partial g} \frac{\partial g}{\partial r} \frac{\partial \bar{r}}{\partial \bar{Z}} \frac{\partial \bar{r}}{\partial \bar{r}} \frac{\partial f_{v2}}{\partial \chi} \\
\frac{\partial f_w}{\partial \tilde{v}} &= \frac{\partial f_w}{\partial g} \frac{\partial g}{\partial r} \frac{\partial \bar{r}}{\partial \tilde{z}} \frac{\partial \bar{r}}{\partial \tilde{z}} \frac{\partial f_{v2}}{\partial \chi} \\
\end{align*}
\]

(A.28)

where

\[
\begin{align*}
\frac{\partial f_w}{\partial g} &= \frac{c_{w3}^6}{g^6 + c_{w3}^6} \left( \frac{1 + c_{w2}^6}{g^6 + c_{w3}^6} \right)^{1/6} \\
\frac{\partial g}{\partial r} &= 1 + c_{w2} (6r^5 - 1) \\
\frac{\partial r}{\partial \bar{r}} &= \tan^{-1} \left( \frac{b (r_{lim} - \bar{r})}{\pi} \right) - \frac{\bar{r}}{\pi} \frac{1}{1 + b^2 (r_{lim} - \bar{r})^2} + \frac{1}{2} \\
\frac{\partial \bar{r}}{\partial \bar{Z}} &= - \frac{\tilde{v}}{\left( \kappa d \bar{Z} \right)^2} \\
\frac{\partial \bar{Z}}{\partial \bar{Z}} &= \begin{cases} 
1 & \bar{Z} \geq -c_{v2} Z \\
\frac{(c_{v2} - c_{v1})Z}{(c_{v1} - 2c_{v2})Z - Z^2} & \bar{Z} < -c_{v2} Z 
\end{cases} \\
\frac{\partial \bar{Z}}{\partial f_{v2}} &= \frac{\bar{v}}{\kappa^2 d^2} \\
\frac{\partial f_{v2}}{\partial \chi} &= - \frac{1 + \chi^2 \frac{\partial f_{v1}}{\partial \chi}}{(1 + \chi f_{v1})^2} \\
\end{align*}
\]

(A.29)

and the \( \tilde{v} \) derivative can be closed with the following equations

\[
\begin{align*}
\frac{\partial \bar{r}}{\partial \tilde{v}} &= \frac{1}{\kappa^2 d^2 \bar{Z}} \frac{\partial \bar{r}}{\partial \bar{Z}} \frac{\partial \bar{Z}}{\partial \tilde{v}} \\
\frac{\partial \bar{Z}}{\partial \tilde{v}} &= \frac{f_{v2}}{\kappa^2 d^2} + \frac{\partial \bar{Z}}{\partial \tilde{v}} \frac{f_{v2}}{\partial \chi} \frac{\partial \chi}{\partial \tilde{v}} \\
\end{align*}
\]

(A.30)
The Jacobian of $S_{II}$ is

$$\frac{\partial S_{II}}{\partial U} = \begin{cases} 
\frac{\partial S_I}{\partial \vec{V}} = 0 \\
\frac{\partial S_I}{\partial p} = -\frac{ca_2}{\sigma} (\nabla \tilde{V} \cdot \nabla \tilde{V}) \frac{\partial p}{\partial p} \\
\frac{\partial S_I}{\partial T} = -\frac{ca_2}{\sigma} (\nabla \tilde{V} \cdot \nabla \tilde{V}) \frac{\partial p}{\partial T} \\
\frac{\partial S_I}{\partial \tilde{\nu}} = 0
\end{cases}$$

(A.31)

and lastly, the Jacobian of $S_{III}$ is

$$\frac{\partial S_{III}}{\partial U} = \begin{cases} 
\frac{\partial S_I}{\partial \vec{V}} = 0 \\
\frac{\partial S_I}{\partial p} = \frac{1}{\sigma} \left( \left( \frac{\partial f_n}{\partial p} \tilde{\nu} - \mu \frac{\partial \rho}{\partial \rho} \right) \nabla \rho \cdot \nabla \tilde{V} + (\nu + f_n \tilde{\nu}) \frac{\partial \rho}{\partial \rho} \cdot \nabla \tilde{V} \right) \\
\frac{\partial S_I}{\partial T} = \frac{1}{\sigma} \left( \left( \frac{\partial f_n}{\partial T} \tilde{\nu} - \mu \frac{\partial \rho}{\partial T} \right) \nabla \rho \cdot \nabla \tilde{V} + (\nu + f_n \tilde{\nu}) \frac{\partial \rho}{\partial \rho} \cdot \nabla \tilde{V} \right) \\
\frac{\partial S_I}{\partial \tilde{\nu}} = \frac{1}{\sigma} \left( f_n + \frac{\partial f_n}{\partial \tilde{\nu}} \tilde{\nu} \right) \nabla \rho \cdot \nabla \tilde{\nu}
\end{cases}$$

(A.32)

where the pressure and temperature derivatives of the density gradient are

$$\begin{align*}
\frac{\partial \nabla \rho}{\partial p} &= -\frac{\nabla T}{R_{\text{gas}} T^2} \\
\frac{\partial \nabla \rho}{\partial T} &= -\frac{\nabla p}{R_{\text{gas}} T^2}
\end{align*}$$

(A.33)

### A.2.2 Gradient Jacobian

The gradient Jacobian of $S_I$ is

$$\frac{\partial S_I}{\partial \nabla U} = \begin{cases} 
\frac{\partial S_I}{\partial \nabla \vec{V}} = \rho \left( \frac{\partial D}{\partial Z} - \frac{\partial \rho}{\partial Z} \right) \frac{\partial Z}{\partial \nabla \vec{V}} \\
\frac{\partial S_I}{\partial \nabla p} = 0 \\
\frac{\partial S_I}{\partial \nabla T} = 0 \\
\frac{\partial S_I}{\partial \nabla \tilde{\nu}} = 0
\end{cases}$$

(A.34)
where the production and destruction derivatives with respect to the vorticity magnitude are

$$\frac{\partial P}{\partial Z} = \begin{cases} \begin{align*} c_b (1 - f \tau) \frac{\partial \tilde{Z}}{\partial Z} \tilde{\nu} & \quad \tilde{\nu} \geq 0 \\ c_b (1 - f \tau) \tilde{\nu} & \quad \tilde{\nu} < 0 \end{align*} \end{cases}$$

$$\frac{\partial D}{\partial Z} = \begin{cases} \begin{align*} c_w [s] - \frac{2}{\bar{Z}} \frac{\partial \tilde{\nu} \sigma}{\partial \tilde{\nu} \sigma} \frac{\partial \tilde{\nu} \sigma}{\partial \tilde{\nu} \sigma} & \quad \tilde{\nu} \geq 0 \\ 0 & \quad \tilde{\nu} < 0 \end{align*} \end{cases}$$

(A.35)

with the intermediate derivatives in the destruction derivative being supplied by Eq. (A.29). The modified vorticity derivative is

$$\frac{\partial \tilde{Z}}{\partial \tilde{Z}} = \begin{cases} \begin{align*} 1 \quad \tilde{Z} \geq -cv_2 Z \\ 1 + \frac{c_2 (c_3 - 2c_2) Z^2 - 2c_2 \tilde{Z} \bar{Z} - cv_3 \tilde{Z}}{(c_3 - 2c_2) \bar{Z} - \tilde{Z}} & \quad \tilde{Z} < -cv_2 Z \end{align*} \end{cases}$$

(A.36)

The derivative of the vorticity magnitude with respect to the velocity gradients is

$$\frac{\partial \tilde{Z}}{\partial \tilde{\nu} \tilde{\nu}} = \frac{1}{\tilde{Z}} \begin{bmatrix} 0 & (u_y - v_x) & (u_z - w_x) \\ (v_y - u_x) & 0 & (v_z - w_y) \\ (w_y - u_x) & (w_z - v_x) & 0 \end{bmatrix}$$

(A.37)

The gradient Jacobian of $S_{II}$ is

$$\frac{\partial S_{II}}{\partial \tilde{\nu} \tilde{\nu}} = \begin{cases} \begin{align*} \frac{\partial S_{II}}{\partial \tilde{\nu} \tilde{\nu}} &= 0 \\ \frac{\partial S_{II}}{\partial \tilde{\nu} \tilde{\nu} \tilde{\nu}} &= 0 \\ \frac{\partial S_{II}}{\partial \tilde{\nu} \tilde{\nu} \tilde{\nu}} &= 0 \\ \frac{\partial S_{II}}{\partial \tilde{\nu} \tilde{\nu} \tilde{\nu} \tilde{\nu}} &= -2 \frac{g_0^2 \rho \tilde{\nu}}{\alpha} \end{align*} \end{cases}$$

(A.38)
and lastly, the gradient Jacobian of $S_{III}$ is

$$\frac{\partial S_{III}}{\partial \nabla U} = \begin{cases} \frac{\partial S_I}{\partial \nabla \tilde{\nu}} = 0 \\ \frac{\partial S_I}{\partial \nabla \rho} = \frac{1}{\sigma} (\mathbf{v} + f_{n1} \mathbf{v}) \nabla \mathbf{v} \cdot \frac{\partial \rho}{\partial \nabla \rho} \\ \frac{\partial S_I}{\partial \nabla T} = \frac{1}{\sigma} (\mathbf{v} + f_{n1} \mathbf{v}) \nabla \mathbf{T} \cdot \frac{\partial \rho}{\partial \nabla \rho} \\ \frac{\partial S_I}{\partial \nabla \tilde{\nu}} = \frac{1}{\sigma} (\mathbf{v} + f_{n1} \mathbf{v}) \nabla \tilde{\nu} \end{cases}$$  \hspace{1cm} (A.39)$$

where the density gradient derivatives are

$$\begin{align*}
\frac{\partial \nabla \rho}{\partial \nabla \rho} &= \frac{1}{R_{gas} T} \\
\frac{\partial \nabla \rho}{\partial \nabla T} &= -\frac{p}{R_{gas} T^2}
\end{align*}$$  \hspace{1cm} (A.40)$$

### A.3 Upwind Dissipation

The upwind dissipation by Roe is given in Eq. (2.53). The Jacobian matrix of the upwind dissipation is obtained by differentiating with respect to $U^+$ and $U^-$, as seen below

$$\frac{\partial}{\partial U^\pm} \phi(U^+, U^-) = \frac{\partial |\lambda_3|}{\partial U^\pm} \Delta Q + |\lambda_3| \frac{\partial \Delta Q}{\partial U^\pm} \begin{pmatrix} \frac{\partial C_1}{\partial U^\pm} \tilde{v}_R + C_1 \frac{\partial \tilde{v}_R}{\partial U^\pm} + \frac{2}{\rho} \frac{\partial C_1}{\partial U^\pm} \tilde{n} \\ \frac{\partial C_1}{\partial U^\pm} H_R + C_1 \frac{\partial H_R}{\partial U^\pm} + \frac{2}{\rho} \frac{\partial C_1}{\partial U^\pm} \tilde{n} + C_2 \frac{\partial \tilde{v}_R}{\partial U^\pm} \end{pmatrix}$$  \hspace{1cm} (A.41)$$

where the conservative jump vector is

$$\Delta Q = \begin{bmatrix} \Delta \rho & \Delta (\rho \mathbf{v}) & \Delta (\rho \mathbf{E}) & \Delta (\rho \mathbf{v}) \end{bmatrix}^T$$  \hspace{1cm} (A.42)$$
which gives the Jacobian matrices

$$\frac{\partial \Delta Q}{\partial U^\pm} = \pm \begin{bmatrix} 0 & 0 & 0 & \frac{\partial p^\pm}{\partial p^\pm} & \frac{\partial p^\pm}{\partial T^\pm} & 0 \\ \rho^\pm & 0 & 0 & \frac{\partial p^\pm}{\partial p^\pm} u^\pm & \frac{\partial p^\pm}{\partial T^\pm} u^\pm & 0 \\ 0 & \rho^\pm & 0 & \frac{\partial p^\pm}{\partial p^\pm} v^\pm & \frac{\partial p^\pm}{\partial T^\pm} v^\pm & 0 \\ 0 & 0 & \rho^\pm & \frac{\partial p^\pm}{\partial p^\pm} w^\pm & \frac{\partial p^\pm}{\partial T^\pm} w^\pm & 0 \\ (\rho u)^\pm & (\rho v)^\pm & (\rho w)^\pm & \frac{\partial p^\pm}{\partial p^\pm} E^\pm & \frac{\partial p^\pm}{\partial T^\pm} E^\pm + \rho^\pm C_v & 0 \\ 0 & 0 & 0 & \frac{\partial p^\pm}{\partial p^\pm} V^\pm & \frac{\partial p^\pm}{\partial T^\pm} V^\pm & \rho^\pm \end{bmatrix}$$

(A.43)

and from the perfect gas relations

$$\begin{align*}
\frac{\partial \rho^\pm}{\partial p^\pm} &= \frac{1}{R_{\text{gas}} T^\pm} \\
\frac{\partial \rho^\pm}{\partial T^\pm} &= -\frac{p^\pm}{R_{\text{gas}} (T^\pm)^2}
\end{align*}$$

(A.44)

To simplify the derivation, the Roe averaged equations are rearranged as shown below

$$\left(\right)_R = \rho^-_{\text{avg}} \left(\right)^- + \rho^+_{\text{avg}} \left(\right)^+$$

(A.45)

where $$\left(\right)_R$$ is the Roe averaged quantity, $$\left(\right)^+$$ and $$\left(\right)^-$$ are the quantity on either side of the jump, and

$$\rho^\pm_{\text{avg}} = \frac{\sqrt{\rho^\pm}}{\sqrt{\rho^-} + \sqrt{\rho^+}}$$

(A.46)

such that the Roe averaged Jacobians are

$$\begin{align*}
\frac{\partial V_R}{\partial U^\pm} &= \hat{V}^\pm \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} + \hat{V}^\pm \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} + \rho^\pm_{\text{avg}} \frac{\partial \hat{V}^\pm}{\partial U^\pm} \\
\frac{\partial H_R}{\partial U^\pm} &= \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} H^\pm + \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} H^\pm + \rho^\pm_{\text{avg}} \frac{\partial H^\pm}{\partial U^\pm} \\
\frac{\partial \hat{V}_R}{\partial U^\pm} &= \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} \hat{V}^\pm + \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} \hat{V}^\pm + \rho^\pm_{\text{avg}} \frac{\partial \hat{V}^\pm}{\partial U^\pm} \\
\frac{\partial T_R}{\partial U^\pm} &= \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} T^\pm + \frac{\partial \rho^\pm_{\text{avg}}}{\partial U^\pm} T^\pm + \rho^\pm_{\text{avg}} \frac{\partial T^\pm}{\partial U^\pm}
\end{align*}$$

(A.47)
noting that $\frac{\partial \mathbf{V}}{\partial U^\pm} = 0$. The $\rho_{avg}$ Jacobians are

$$
\begin{align*}
\frac{\partial \rho^\pm_{avg}}{\partial U^\pm} & = \frac{\partial \rho^\pm_{avg}}{\partial \rho^\pm} \frac{\partial \rho^\pm}{\partial U^\pm} \\
\frac{\partial \rho^+_{avg}}{\partial U^\pm} & = \frac{\partial \rho^+_{avg}}{\partial \rho^+} \frac{\partial \rho^+}{\partial U^\pm} \\
\frac{\partial \rho^-_{avg}}{\partial U^\pm} & = \frac{\partial \rho^-_{avg}}{\partial \rho^-} \frac{\partial \rho^-}{\partial U^\pm} \\
\frac{\partial \rho^-_{avg}}{\partial \rho^+} & = \frac{1}{2 \sqrt{\rho^+}} \\
\frac{\partial \rho^+_{avg}}{\partial \rho^-} & = -\frac{\partial \rho^+_{avg}}{\partial \rho^-} \\
\end{align*}
$$

and finally the Roe averaged speed of sound is

$$
\begin{align*}
\frac{\partial c_R}{\partial U^\pm} & = \frac{\partial c_R}{\partial T_R} \frac{\partial T_R}{\partial U^\pm} \\
\frac{\partial c_R}{\partial T_R} & = \frac{1}{2} \sqrt{\frac{\gamma R_{sys}}{T_R}} \\
\end{align*}
$$

(A.49)

To complete Eqs. (A.47) and (A.48), Jacobians of the left and right side quantities are needed. These Jacobians are

$$
\begin{align*}
\frac{\partial \tilde{V}^\pm}{\partial U^\pm} & = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{bmatrix} \\
\frac{\partial H^\pm}{\partial U^\pm} & = \begin{bmatrix} u^\pm & v^\pm & w^\pm & 0 & C_p \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \\
\frac{\partial \tilde{V}^\pm}{\partial U^\pm} & = \begin{bmatrix} 0 & 0 & \frac{\partial \rho^\pm}{\partial \rho^+} & \frac{\partial \rho^\pm}{\partial \rho^-} & 0 \\
\end{bmatrix} \\
\frac{\partial \rho^\pm}{\partial U^\pm} & = \begin{bmatrix} 0 & 0 & \frac{\partial \rho^\pm}{\partial \rho^+} & \frac{\partial \rho^\pm}{\partial \rho^-} & 0 \\
\end{bmatrix} \\
\end{align*}
$$

(A.50)

The Jacobians of the eigenvalues with the entropy fix are obtained readily from the chain rule

$$
\frac{\partial |\lambda_i|}{\partial U^\pm} = \frac{\partial |\lambda_i|}{\partial \lambda_i} \frac{\partial \lambda_i}{\partial U^\pm}
$$

(A.51)
with
\[
\frac{\partial |\lambda_i|}{\partial \lambda_i} = \begin{cases} 
\frac{\lambda_i}{\varepsilon_i c_R} & -\varepsilon_i c_R < \lambda_i < \varepsilon_i c_R \\
1 & \lambda_i \geq \varepsilon_i c_R \\
-1 & \lambda_i \leq -\varepsilon_i c_R
\end{cases}
\]  
(A.52)

and the eigenvalue Jacobians
\[
\frac{\partial \lambda_i}{\partial U^\pm} = \frac{\partial \vec{V}_R \cdot \vec{n}}{\partial U^\pm} + \frac{\partial c_R}{\partial U^\pm}
\]
\[
\frac{\partial \lambda_i}{\partial U^\pm} = \frac{\partial \vec{V}_R \cdot \vec{n}}{\partial U^\pm} - \frac{\partial c_R}{\partial U^\pm}
\]
\[
\frac{\partial \lambda_i}{\partial U^\pm} = \frac{\partial \vec{V}_R \cdot \vec{n}}{\partial U^\pm}
\]  
(A.53)

where
\[
\frac{\partial \vec{V}_R \cdot \vec{n}}{\partial U^\pm} = \left( u^\pm n_x + v^\pm n_y + w^\pm n_z \right) \frac{\partial \rho_{\text{avg}}^\pm}{\partial U^\pm} + \left( u^\mp n_x + v^\mp n_y + w^\mp n_z \right) \frac{\partial \rho_{\text{avg}}^\mp}{\partial U^\pm} + \rho_{\text{avg}} \left[ n_x, n_y, n_z, 0 \right] (A.54)
\]

The last terms to determine are the Jacobians of $C_1$ and $C_2$. Using the chain rule again, these Jacobians are
\[
\frac{\partial C_1}{\partial U^\pm} = \frac{\partial}{\partial U^\pm} \left( \frac{G_1}{c_R} \right) s_1 + \frac{G_1}{c_R} \frac{\partial s_1}{\partial U^\pm} + \frac{\partial}{\partial U^\pm} \left( \frac{G_2}{c_R} \right) s_2 + \frac{G_2}{c_R} \frac{\partial s_2}{\partial U^\pm}
\]
\[
\frac{\partial C_2}{\partial U^\pm} = \frac{\partial}{\partial U^\pm} \left( \frac{G_1}{c_R} \right) s_2 + \frac{G_1}{c_R} \frac{\partial s_2}{\partial U^\pm} + \frac{\partial}{\partial U^\pm} \left( \frac{G_2}{c_R} \right) s_1 + \frac{G_2}{c_R} \frac{\partial s_1}{\partial U^\pm}
\]  
(A.55)

where the Jacobians of the $G_1$ and $G_2$ ratios are
\[
\frac{\partial}{\partial U^\pm} \left( \frac{G_1}{c_R} \right) = \frac{1}{c_R} \frac{\partial G_1}{\partial U^\pm} - \frac{G_1}{c_R^2} \frac{\partial c_R}{\partial U^\pm}
\]
\[
\frac{\partial}{\partial U^\pm} \left( \frac{G_2}{c_R} \right) = \frac{1}{c_R} \frac{\partial G_2}{\partial U^\pm} - \frac{G_2}{c_R^2} \frac{\partial c_R}{\partial U^\pm}
\]
\[
\frac{\partial}{\partial U^\pm} \left( \frac{G_1}{c_R} \right) = \frac{1}{c_R^2} \frac{\partial G_1}{\partial U^\pm} - \frac{2}{c_R^3} \frac{\partial c_R}{\partial U^\pm}
\]  
(A.56)
and the $G_1$ and $G_2$ Jacobians are

$$
\frac{\partial G_1}{\partial U^\pm} = (\gamma - 1) \left( \dot{V}_R \Delta \rho \frac{\partial \dot{V}_R}{\partial U^\pm} + \frac{1}{2} \dot{V}_R^2 \frac{\partial \Delta \rho}{\partial U^\pm} - \dot{V}_R \left[ \frac{\partial \Delta (\rho \dot{V})}{\partial U^\pm} - \frac{\partial \Delta (\rho E)}{\partial U^\pm} \right] \right)
$$

$$
\frac{\partial G_2}{\partial U^\pm} = -\frac{\partial \dot{V}_R \cdot \bar{n}}{\partial U^\pm} \Delta \rho - \dot{V}_R \left[ \frac{\partial \Delta \rho}{\partial U^\pm} + \frac{\partial \Delta (\rho \dot{V})}{\partial U^\pm} \right]
$$

(A.57)

where

$$
\frac{\partial \Delta (\rho \dot{V}) \cdot \bar{n}}{\partial U^\pm} = \begin{bmatrix}
\rho^+ n_x \\
\rho^+ n_y \\
\rho^+ n_z \\
\dot{\rho}^+ v^+ (u^+ n_x + v^+ n_y + w^+ n_z) \\
\dot{\rho}^+ p^+ (u^+ n_x + v^+ n_y + w^+ n_z) \\
0
\end{bmatrix}
\begin{bmatrix}
\partial \rho^+/\partial \rho \\
\partial \rho^+/\partial \rho \\
\partial \rho^+/\partial \rho \\
\partial p^+/\partial \rho \\
\partial p^+/\partial \rho \\
\partial p^+/\partial \rho
\end{bmatrix}^{T}
\quad (A.58)
$$

The derivation is completed with the Jacobians of $s_1$ and $s_2$

$$
\frac{\partial s_1}{\partial U^\pm} = \frac{1}{2} \left( \frac{\partial |\lambda_1|}{\partial U^\pm} + \frac{\partial |\lambda_2|}{\partial U^\pm} \right) - \frac{\partial |\lambda_3|}{\partial U^\pm}
$$

$$
\frac{\partial s_2}{\partial U^\pm} = \frac{1}{2} \left( \frac{\partial |\lambda_1|}{\partial U^\pm} - \frac{\partial |\lambda_2|}{\partial U^\pm} \right)
\quad (A.59)
$$

where the entropy fixed eigenvalue Jacobians were derived earlier in Eq. (A.51).

### A.4 Time Term

The time linearization of the primitive variables version of the governing equation requires the Jacobian matrix of the conservative variable vector with respect to the primitive variables. This Jacobian matrix is
similar to Eq. (A.43), as shown below

\[
\frac{\partial Q}{\partial U} = \pm \begin{bmatrix}
0 & 0 & 0 & \frac{\partial \rho}{\partial \rho} & \frac{\partial \rho}{\partial T} & 0 \\
\rho & 0 & 0 & \frac{\partial \rho}{\partial \rho} u & \frac{\partial \rho}{\partial T} u & 0 \\
0 & \rho & 0 & \frac{\partial \rho}{\partial \rho} v & \frac{\partial \rho}{\partial T} v & 0 \\
0 & 0 & \rho & \frac{\partial \rho}{\partial \rho} w & \frac{\partial \rho}{\partial T} w & 0 \\
\rho u & \rho v & \rho w & \frac{\partial \rho}{\partial \rho} E & \frac{\partial \rho}{\partial T} E + \rho C_v & 0 \\
0 & 0 & 0 & \frac{\partial \rho}{\partial \rho} \bar{\nu} & \frac{\partial \rho}{\partial T} \bar{\nu} & \rho \\
\end{bmatrix}
\]  

(A.60)

where the density derivatives with respect to pressure and temperature are given by Eq. (A.5).

---

### A.5 Boundary Conditions

For the linearization of the boundary conditions, the Jacobian matrices of the fluxes with respect to the interior variables and gradients are needed. Consider the advective flux boundary condition Jacobian expanded using the chain rule

\[
\frac{\partial \bar{F}^a(U_B(U^-,U_b))}{\partial U^-} = \frac{\partial \bar{F}^a(U_B(U^-,U_b))}{\partial U_B} \frac{\partial U_B(U^-,U_b)}{\partial U^-}
\]  

(A.61)

where the first term of the expansion is equivalent to Eqs. (A.2), (A.3), and (A.4) with \( U \) replaced with \( U_B \).

Therefore, the Jacobian matrix needed for the boundary conditions is

\[
\frac{\partial U_B(U^-,U_b)}{\partial U^-} = \begin{bmatrix}
\frac{\partial u_B}{\partial u^-} & \frac{\partial u_B}{\partial v^-} & \frac{\partial u_B}{\partial w^-} & \frac{\partial u_B}{\partial \rho^-} & \frac{\partial u_B}{\partial T^-} \\
\frac{\partial v_B}{\partial u^-} & \frac{\partial v_B}{\partial v^-} & \frac{\partial v_B}{\partial w^-} & \frac{\partial v_B}{\partial \rho^-} & \frac{\partial v_B}{\partial T^-} \\
\frac{\partial w_B}{\partial u^-} & \frac{\partial w_B}{\partial v^-} & \frac{\partial w_B}{\partial w^-} & \frac{\partial w_B}{\partial \rho^-} & \frac{\partial w_B}{\partial T^-} \\
\frac{\partial \rho_B}{\partial u^-} & \frac{\partial \rho_B}{\partial v^-} & \frac{\partial \rho_B}{\partial w^-} & \frac{\partial \rho_B}{\partial \rho^-} & \frac{\partial \rho_B}{\partial T^-} \\
\frac{\partial T_B}{\partial u^-} & \frac{\partial T_B}{\partial v^-} & \frac{\partial T_B}{\partial w^-} & \frac{\partial T_B}{\partial \rho^-} & \frac{\partial T_B}{\partial T^-} \\
\frac{\partial \bar{\nu}_B}{\partial u^-} & \frac{\partial \bar{\nu}_B}{\partial v^-} & \frac{\partial \bar{\nu}_B}{\partial w^-} & \frac{\partial \bar{\nu}_B}{\partial \rho^-} & \frac{\partial \bar{\nu}_B}{\partial T^-} \\
\end{bmatrix}
\]  

(A.62)

This is further simplified by considering that the Spalart-Allmaras working variable is decoupled from the other variables for all boundary conditions used in this work (see Section 2.2.2.4). This reduces Eq. (A.62)
$$\frac{\partial U_B(U^-, U_b)}{\partial U^-} = \begin{bmatrix} \frac{\partial u_B}{\partial u} & \frac{\partial u_B}{\partial v} & \frac{\partial u_B}{\partial w} & \frac{\partial u_B}{\partial p} & \frac{\partial u_B}{\partial T} & 0 \\ \frac{\partial v_B}{\partial u} & \frac{\partial v_B}{\partial v} & \frac{\partial v_B}{\partial w} & \frac{\partial v_B}{\partial p} & \frac{\partial v_B}{\partial T} & 0 \\ \frac{\partial w_B}{\partial u} & \frac{\partial w_B}{\partial v} & \frac{\partial w_B}{\partial w} & \frac{\partial w_B}{\partial p} & \frac{\partial w_B}{\partial T} & 0 \\ \frac{\partial p_B}{\partial u} & \frac{\partial p_B}{\partial v} & \frac{\partial p_B}{\partial w} & \frac{\partial p_B}{\partial p} & \frac{\partial p_B}{\partial T} & 0 \\ \frac{\partial T_B}{\partial u} & \frac{\partial T_B}{\partial v} & \frac{\partial T_B}{\partial w} & \frac{\partial T_B}{\partial p} & \frac{\partial T_B}{\partial T} & 0 \\ \frac{\partial \tilde{\nu}_B}{\partial u} & \frac{\partial \tilde{\nu}_B}{\partial v} & \frac{\partial \tilde{\nu}_B}{\partial w} & \frac{\partial \tilde{\nu}_B}{\partial p} & \frac{\partial \tilde{\nu}_B}{\partial \tilde{\nu}} & \frac{\partial \tilde{\nu}_B}{\partial \tilde{\nu}} \end{bmatrix}$$  \hspace{1cm} (A.63)

To avoid repetition, the row for the $\tilde{\nu}_B$ derivatives are only provided in Section (A.5.1).

An argument similar to Eq. (A.61) can be made for the gradient Jacobian matrix. In addition, the boundary gradients used in this work are either the interior gradient for Dirichlet boundary conditions or zero for Neumann boundary conditions. This reduces the gradient Jacobians to diagonal matrices with elements equal to either zero or one. For example, the adiabatic wall gradient vector is

$$\nabla U_B(\nabla U^-, \nabla U_B) = \begin{pmatrix} \nabla u^- \\ \nabla v^- \\ \nabla w^- \\ \nabla p^- \\ 0 \\ \nabla \tilde{\nu}^- \end{pmatrix}$$ \hspace{1cm} (A.64)

which gives the Jacobian matrices

$$\frac{\partial \nabla U_B(\nabla U^-, \nabla U_B)}{\partial \nabla U^-} = (G, G, G)$$

\[ G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hspace{1cm} (A.65) \]
The majority of the boundary conditions are of the Dirichlet type. Therefore, the derivatives

\[ \text{diag}(G) = \begin{pmatrix}
\frac{\partial v_B}{\partial u} & \frac{\partial v_B}{\partial v} & \frac{\partial v_B}{\partial w} & \frac{\partial v_B}{\partial p} & \frac{\partial v_B}{\partial T} & \frac{\partial v_B}{\partial \bar{\nu}} \\
\end{pmatrix} \] (A.66)

are equal to one unless otherwise specified.

A.5.1 Spalart-Allmaras

Two conditions are needed for the S-A working variable. These are Dirichlet for the inflow and no-slip wall boundary conditions and extrapolation for slip wall, symmetry and outflow boundary conditions. The row for the S-A variable will be given in this section and will not appear in the Jacobian matrices of following sections.

A.5.1.1 Dirichlet

For inflow boundaries, the Dirichlet condition is imposed with the ratio of S-A to laminar viscosity. The resulting derivatives are

\[
\begin{align*}
\frac{\partial \bar{\nu}_B}{\partial \bar{v}} &= 0 \\
\frac{\partial \bar{v}_B}{\partial p} &= \left( \frac{\partial p}{\partial p^-} \right) \left( \frac{\bar{v}}{\bar{v}_b} \right) \\
\frac{\partial \bar{v}_B}{\partial T} &= \left( \frac{\partial p}{\partial T^-} \right) \left( \frac{\bar{v}}{\bar{v}_b} \right) + \rho \frac{\partial \mu^-}{\partial T^-} \left( \frac{\bar{v}}{\bar{v}_b} \right) \\
\frac{\partial \bar{v}_B}{\partial \bar{\nu}} &= 0
\end{align*}
\] (A.67)

where the density and viscosity derivatives are given in Eqs. (A.5) and (A.11). The no-slip wall condition is a zero value of \( \bar{v} \), which results in zeros for all of the gradients.
A.5.1.2 Extrapolation

For extrapolation, the interior S-A variable is specified on the boundary and the gradient is set to zero. The derivatives are simply

\[
\frac{\partial \tilde{\nu}_B}{\partial \vec{V}^-} = \frac{\partial \tilde{\nu}_B}{\partial p^-} = \frac{\partial \tilde{\nu}_B}{\partial T^-} = 0 \\
\frac{\partial \tilde{\nu}_B}{\partial \vec{V}^-} = 1 \\
\frac{\partial \nabla \tilde{\nu}_B}{\partial \vec{V}^-} = 0
\]

(A.68)

A.5.2 Walls

The conditions on the variables are the same for the adiabatic no-slip wall and slip wall. The derivatives for these boundary conditions are zero for all except

\[
\frac{\partial p_B}{\partial p^-} = \frac{\partial T_B}{\partial T^-} = 1
\]

(A.69)

The iso-thermal wall derivatives are zeros except

\[
\frac{\partial p_B}{\partial p^-} = 1
\]

(A.70)

The gradient derivatives for the adiabatic no-slip wall are given in Eq. (A.65). For the other two walls, \(G\) is the identity matrix.

A.5.3 Symmetry

For the symmetry boundary condition, the Jacobian matrix is

\[
\frac{\partial U_B (U^-, U_b)}{\partial U^-} = \begin{bmatrix}
1 - 2n_x^2 & -2n_xn_y & -2n_xn_z & 0 & 0 \\
-2n_xn_y & 1 - 2n_y^2 & -2n_yn_z & 0 & 0 \\
-2n_xn_z & -2n_yn_z & 1 - 2n_z^2 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

(A.71)
A.5.4 Total Conditions Inflow

Using the chain rule, the Jacobian matrix for this inflow are

\[ \frac{\partial U_B (U^-, U_b)}{\partial U^-} = \begin{bmatrix}
\frac{\partial |\vec{V}|}{\partial u} n_{x,b} & \frac{\partial |\vec{V}|}{\partial v} n_{y,b} & \frac{\partial |\vec{V}|}{\partial w} n_{z,b} & 0 & 0 \\
\frac{\partial |\vec{V}|}{\partial u} n_{x,b} & \frac{\partial |\vec{V}|}{\partial v} n_{y,b} & \frac{\partial |\vec{V}|}{\partial w} n_{z,b} & 0 & 0 \\
\frac{\partial |\vec{V}|}{\partial u} n_{x,b} & \frac{\partial |\vec{V}|}{\partial v} n_{y,b} & \frac{\partial |\vec{V}|}{\partial w} n_{z,b} & 0 & 0 \\
\frac{\partial T_B}{\partial u} & \frac{\partial T_B}{\partial v} & \frac{\partial T_B}{\partial w} & 0 & 0 \\
\frac{\partial T_B}{\partial u} & \frac{\partial T_B}{\partial v} & \frac{\partial T_B}{\partial w} & 0 & 0 \\
\frac{\partial T_B}{\partial u} & \frac{\partial T_B}{\partial v} & \frac{\partial T_B}{\partial w} & 0 & 0 \\
\end{bmatrix} \]  

(A.72)

where \( \vec{n}_b = [n_{x,b}, n_{y,b}, n_{z,b}] \) the derivatives of the velocity magnitude are

\[ \frac{\partial |\vec{V}|}{\partial \vec{V}^-} = \frac{\vec{V}^-}{|\vec{V}^-|} \]  

(A.73)

The derivative of the boundary pressure with respect to the boundary temperature is

\[ \frac{\partial p_B}{\partial T_B} = \frac{P_T}{T_B^{\gamma/(\gamma-1)}} \left( \frac{\gamma}{\gamma-1} \right) T_B^{1/(\gamma-1)} \]  

(A.74)

and the velocity derivatives of the boundary temperature are

\[ \frac{\partial T_B}{\partial \vec{V}^-} = -\frac{\vec{V}^-}{C_p} \]  

(A.75)

A.5.5 Temperature and Velocity Inflow

All of the variables are specified, except for pressure, which gives the Jacobian matrix

\[ \frac{\partial U_B (U^-, U_b)}{\partial U^-} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]  

(A.76)
A.5.6 Static Pressure Outflow

The pressure outflow boundary is the reverse of the temperature velocity inflow, resulting in the following

\[
\frac{\partial U_B(U^-, U_b)}{\partial U^-} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]  
(A.77)

A.5.7 Riemann Invariant

As discussed in Section (2.2.2.4), the Riemann Invariant boundary condition interrogates the interior variables and determines which of the following four conditions are present

- **Supersonic Inflow**  
  \[ |\vec{V}^- \cdot \vec{n}| \geq c^-, \quad |\vec{V}^- \cdot \vec{n}| \leq 0 \]

- **Supersonic Outflow**  
  \[ |\vec{V}^- \cdot \vec{n}| \geq c^-, \quad |\vec{V}^- \cdot \vec{n}| > 0 \]

- **Subsonic Inflow**  
  \[ |\vec{V}^- \cdot \vec{n}| < c^-, \quad |\vec{V}^- \cdot \vec{n}| \leq 0 \]

- **Subsonic Outflow**  
  \[ |\vec{V}^- \cdot \vec{n}| < c^-, \quad |\vec{V}^- \cdot \vec{n}| > 0 \]

These four cases produce different Jacobian matrices with respect to the variables. However, all four have the same gradient Jacobian with

\[
\begin{align*}
\frac{\partial \nabla u_B}{\partial \nabla u^-} &= \frac{\partial \nabla v_B}{\partial \nabla v^-} = \frac{\partial \nabla w_B}{\partial \nabla w^-} &= \frac{\partial \nabla p_B}{\partial \nabla p^-} = \frac{\partial \nabla T_B}{\partial \nabla T^-} = 1
\end{align*}
\]  
(A.78)

A.5.7.1 Supersonic Inflow

All of the variables are set to the specified variables, \((U^-)\), for portions of the boundary designated as a supersonic inflow, which gives

\[
\frac{\partial U_B(U^-, U_b)}{\partial U^-} = 0
\]  
(A.79)
A.5.7.2 Supersonic Outflow

Being the opposite of the supersonic inflow, in that all of the variables are taken from the interior, the Jacobian matrix is

$$\frac{\partial U_B(U^-, U_b)}{\partial U^-} = I \quad (A.80)$$

A.5.7.3 Subsonic

For the subsonic inflow and outflow, the boundary variables are constructed using the Riemann Invariants and the boundary entropy. The Jacobian of the boundary velocity vector is

$$\frac{\partial \vec{V}_B}{\partial \vec{V}^-} = \begin{cases} \frac{\partial \vec{V}_n}{\partial \vec{R}_e} = \frac{\vec{n}}{2} \\ \frac{\partial \vec{V}_e}{\partial \vec{R}_e} = \vec{n} \\ \frac{\partial \vec{V}_e}{\partial T^-} = \frac{1}{(\gamma - 1)} \sqrt{\frac{\gamma R_{gas}}{T^-}} \end{cases} \quad (A.81)$$

where the extrapolated invariant derivatives are

$$\frac{\partial \vec{V}_n}{\partial \vec{R}_e} = \frac{\vec{n}}{2}$$
$$\frac{\partial \vec{V}_e}{\partial \vec{R}_e} = \vec{n}$$
$$\frac{\partial \vec{V}_e}{\partial T^-} = \frac{1}{(\gamma - 1)} \sqrt{\frac{\gamma R_{gas}}{T^-}} \quad (A.82)$$

and the tangential velocity derivative is

$$\frac{\partial \vec{V}_t}{\partial \vec{V}^-} = \begin{cases} 0 & \text{Inflow} \\ \begin{bmatrix} 1 - n_x^2 & -n_xn_y & -n_xn_z \\ -n_xn_y & 1 - n_y^2 & -n_yn_z \\ -n_xn_z & -n_yn_z & 1 - n_z^2 \end{bmatrix} & \text{Outflow} \end{cases} \quad (A.83)$$
Next, we consider the Jacobian of the boundary pressure

\[
\frac{\partial p_B}{\partial U^-} = \begin{cases} 
\frac{\partial p_B}{\partial V^-} = \frac{C_p}{\gamma \rho} \frac{\partial T_B}{\partial V^-} \frac{p_B}{t_B} \\
\frac{\partial p_B}{\partial p^-} = -\frac{\partial S_B}{\partial p^-} \frac{p_B}{S_B} \\
\frac{\partial p_B}{\partial T^-} = \frac{C_p}{\gamma \rho} \frac{\partial T_B}{\partial T^-} \frac{p_B}{S_B} - \frac{\partial S_B}{\partial p^-} \frac{p_B}{S_B}
\end{cases}
\]  

(A.84)

where the entropy derivatives are

\[
\frac{\partial S_B}{\partial p^-} = \begin{cases} 
0 & \text{Inflow} \\
-\frac{S_B}{p} & \text{Outflow}
\end{cases}
\]  

\[
\frac{\partial S_B}{\partial T^-} = \begin{cases} 
0 & \text{Inflow} \\
\frac{C_p}{\gamma \rho} \frac{(T^-)^{\gamma -1}}{T^0} & \text{Outflow}
\end{cases}
\]  

(A.85)

and the Jacobian of the boundary temperature is

\[
\frac{\partial T_B}{\partial U^-} = \begin{cases} 
\frac{\partial T_B}{\partial V^-} = \frac{2C_p}{\gamma \rho} \frac{\partial c_B}{\partial V^-} \frac{\partial c}{\partial V^-} \\
\frac{\partial T_B}{\partial p^-} = 0 \\
\frac{\partial T_B}{\partial T^-} = \frac{2C_p}{\gamma \rho} \frac{\partial c_B}{\partial T^-} \frac{\partial c}{\partial T^-}
\end{cases}
\]  

(A.86)

Lastly, the derivative of the boundary speed of sound with respect to the extrapolated invariant is

\[
\frac{\partial c_B}{\partial R_c} = \frac{\gamma - 1}{4}
\]  

(A.87)
Appendix B

Spalart-Allmaras Conservative Form

The conservative form of the Spalart-Allmaras transport equation can be obtained as mentioned by Allmaras et al. [107]. This process is expanded upon here to demonstrate how the process produces another term in the S-A equation. Accidental exclusion of this term results in an inaccurate and potentially unstable governing equation.

The conservative form is achieved by summing the S-A equation multiplied by density and the mass conservation equation multiplied by the S-A variable, \( \tilde{\nu} \). Conservation of mass in conservative form is

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{B.1}
\]

and the S-A transport equation from (Allmaras et al. [107]) is

\[
\frac{D\tilde{\nu}}{Dt} = P - D + T + \frac{1}{\sigma} \left[ \nabla \cdot ((\nu + \tilde{\nu}) \nabla \tilde{\nu}) + c_{b2} (\nabla \tilde{\nu} \cdot \nabla \tilde{\nu}) \right] \tag{B.2}
\]

Note that transition is not considered in this work, therefore \( T = 0 \).

Multiplying Eq. (B.1) by \( \tilde{\nu} \) gives

\[
\tilde{\nu} \frac{\partial \rho}{\partial t} + \tilde{\nu} \nabla \cdot (\rho \vec{V}) = 0 \tag{B.3}
\]

and multiplying Eq. (B.2) by \( \rho \) gives

\[
\rho \frac{D\tilde{\nu}}{Dt} = \rho (P - D + T) + \frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{\nu}) \nabla \tilde{\nu}) + \frac{c_{b2} \rho}{\sigma} (\nabla \tilde{\nu} \cdot \nabla \tilde{\nu}) \tag{B.4}
\]
By expanding the substantial derivative, $\frac{D\tilde{v}}{Dt}$, on the left-hand side of Eq. (B.4) becomes

$$
\rho \frac{D\tilde{v}}{Dt} = \rho \frac{\partial \tilde{v}}{\partial t} + \rho \tilde{V} \cdot \nabla \tilde{v}
$$

(A.5)

A simplification is possible using product rule for the divergence of a scalar and vector, which is

$$
\nabla \cdot (a\vec{b}) = a\nabla \cdot \vec{b} + \vec{b} \cdot \nabla a
$$

(A.6)

where $a$ is a scalar and $\vec{b}$ is a vector. Using the chain rule and Eq. (B.6), the sum of the left-hand side of Eqs. (B.3) and (B.4) is

$$
\frac{\partial \rho \tilde{v}}{\partial t} + \nabla \cdot \left( \rho \tilde{V} \tilde{v} \right)
$$

(A.7)

On the right-hand side of Eq. (B.4), the second term must be converted to conservative form using Eq. (B.6), resulting in

$$
\frac{\rho}{\sigma} \nabla \cdot \left( (\nu + \tilde{\nu}) \nabla \tilde{v} \right) = \frac{1}{\sigma} \nabla \cdot \left( (\mu + \rho \tilde{\nu}) \nabla \tilde{v} \right) - \frac{1}{\sigma} (\nu + \tilde{\nu}) \nabla \tilde{v} \cdot \nabla \rho
$$

(A.8)

Summing Eqs. (B.3) and (B.4) and using the modifications of Eqs. (B.7) and (B.8) provides the final equation

$$
\frac{\partial \rho \tilde{v}}{\partial t} + \nabla \cdot \left( \rho \tilde{V} \tilde{v} \right) = \rho (P - D + T) + \frac{1}{\sigma} \nabla \cdot \left( (\mu + \rho \tilde{\nu}) \nabla \tilde{v} \right) + \frac{\varepsilon_{b2} P}{\sigma} (\nabla \tilde{v} \cdot \nabla \tilde{v}) - \frac{1}{\sigma} (\nu + \tilde{\nu}) \nabla \tilde{v} \cdot \nabla \rho
$$

where the last term is a product of the transformation into conservative form.
Appendix C

Iterative Method Preconditioning

The general form of a system of linear equations is

\[ Ax = b \]  \hspace{1cm} (C.1)

where the inverse of \( A \) is necessary to obtain the solution vector, \( x \). A direct solution to Eq. (C.1) can be obtained in matrix form by performing a Lower Upper (LU) decomposition, such that \( LU = A \). If \( A \) is a sparse matrix (it has mostly zero entries), the memory requirements of the LU decomposition can be significant. This cost is caused by the filling of many (or all) of the zero entries in \( A \) during the construction of \( L \) and \( U \). For this reason, it is often advantageous to employ an iterative method to solve Eq. (C.1), which requires the storage of only non-zero elements.

In general, iterative methods are slower and less robust than their direct counterparts. The speed and reliability of iterative methods can be improved by preconditioning \( A \). This is accomplished with a preconditioning matrix (often called a preconditioner), \( M \). Applying right preconditioning modifies Eq. (C.1), giving

\[ AM^{-1}y = b \]
\[ Mx = y \]  \hspace{1cm} (C.2)

where \( M \) is some approximation of \( A \). The benefits of applying \( M \) increase as \( M \to A \); however, the computational cost increases as well. Therefore, a trade-off is needed between effectiveness and cost.
The iterative method chosen for this work is the Flexible General Minimum Residual Method (FGMRES)\cite{124}. During the iterative process, this method requires the calculation of

$$MW = v$$ \hspace{1cm} (C.3)

where $w$ and $v$ are intermediate vectors. The Lower Upper Symmetric Guass-Seidel (LU-SGS)\cite{126} and the Incomplete Lower Upper decomposition (ILU)\cite{124} preconditioners were previously implemented by Galbraith \cite{101} within the DG-Chimera framework. For the implicit Harmonic Balance (HB) method, the system of equations becomes

$$A^*x^* = b^*$$ \hspace{1cm} (C.4)

and

$$A^* = \begin{bmatrix} A_1 & H_{1,2} & \cdots & H_{1,N} \\ H_{2,1} & A_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & H_{N-1,N} \\ H_{N,1} & \cdots & H_{N,N-1} & A_N \end{bmatrix}, \quad x^* = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad b^* = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$ \hspace{1cm} (C.5)

where $A^*$, $x^*$, and $b^*$ are derived in Section 3.4.1. In summary, $A_i$ are the linearized spatial fluxes, $H_{i,j}$ come from the linearization of the pseudo-spectral term, $x_i$ are the update vectors, $b_i$ are the residuals, and $N$ is the number of HB time levels. A simple extension of the aforementioned preconditioners to Eq. (C.4) would be to form them for each spatial linearization, $A_i$. This gives the preconditioning matrix

$$M^* = \begin{bmatrix} M_1 & 0 & \cdots & 0 \\ 0 & M_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & M_N \end{bmatrix}$$ \hspace{1cm} (C.6)

and extension of Eq. (C.3) is

$$M^*w^* = v^*$$ \hspace{1cm} (C.7)
where \( w^* \) and \( v^* \) are each a series of intermediate vectors, which gives

\[
M_i w_i = v_i \quad \forall i \in [1,N]
\]  

(C.8)

In words, this means that each time level is preconditioned independent of the other time levels. Unfortunately, the decreased effectiveness of this approach (as compared to the preconditioning of \( A \)) becomes readily apparent. This loss of performance is not surprising, given that the aforementioned \( M^* \) only approximates \( A_i \) and ignores the pseudo-spectral terms. The sections below provide the derivation of improvements to the LU-SGS and ILU preconditioning matrices for solving the implicit HB system of equations. Details of the enhanced performance of these HB preconditioners are given in Section 3.4.2.

### C.1 Harmonic Balance Lower Upper Symmetric Gauss-Seidel Preconditioner

The LU-SGS preconditioning matrix is implemented as described by Galbraith [101], with a short summary given here. Consider the \( A \) matrix split into several component matrices

\[
A = L + D + U \quad \text{(C.9)}
\]

where \( L \) is the strictly lower triangular matrix, \( U \) is the strictly upper triangular matrix, and \( D \) is the diagonal of the matrix. The preconditioning matrix is

\[
M = (L + D) D^{-1} (D + U) \quad \text{(C.10)}
\]

From this equation, the solution to Eq. (C.3) can be obtained with a forward and backward sweep, given by

\[
\begin{align*}
   w' & = D^{-1} (v - Lw') \\ 
   w & = w' - D^{-1} Uw
\end{align*}
\]

(C.11)

where \( w' \) is a temporary vector.

A similar process is used to precondition the HB system, where the HB linearization matrix is divided
as

\[ A^* = L^* + D^* + U^* \]  \hspace{1cm} \text{(C.12)}

and these component matrices are

\[
L^* = \begin{bmatrix}
L_1 & \cdots & \cdots & 0 \\
H_{2,1} & L_2 & \vdots & \\
\vdots & \ddots & \ddots & \ddots \\
H_{N,1} & \cdots & H_{N,N-1} & L_N
\end{bmatrix},
\]

\[
D^* = \begin{bmatrix}
D_1 & 0 & \cdots & 0 \\
0 & D_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & D_N
\end{bmatrix},
\]

\[
U^* = \begin{bmatrix}
H_1 & H_{1,2} & \cdots & H_{1,N} \\
\vdots & H_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & H_{N-1,N}
\end{bmatrix}
\]  \hspace{1cm} \text{(C.13)}

where \( L_i, D_i, \) and \( U_i \) are the component matrices of \( A_i \). Now, Eq. (C.7) can be solved as

\[
w'^* = (D^*)^{-1}(v^* - L^*w^*)
\]

\[
w^* = w'^* - (D^*)^{-1}U^*w^*
\]  \hspace{1cm} \text{(C.14)}

where \( w'^* \) is a series of temporary vectors. The \( (D^*)^{-1} \) matrix is stored, for a given \( A^* \), rather than recomputing it for each sweep.

### C.2 Harmonic Balance Incomplete Lower Upper Decomposition Preconditioner

As suggested by the name, the Incomplete Lower Upper preconditioner emulates LU decomposition, such that

\[
M = LU \approx A
\]  \hspace{1cm} \text{(C.15)}

where the amount that \( L \) and \( U \) are filled in is chosen. Filling all of \( L \) and \( U \) gives the full decomposition. Galbraith implemented versions of the preconditioner with zero fill in, ILU(0), and with one diagonal filled in per existing off-diagonal in \( A \), ILU(1) [101]. Solving Eq. (C.3) gives

\[
LUw = v
\]  \hspace{1cm} \text{(C.16)}
which is solved with a forward and backward substitution

\[ Lw' = v \]
\[ Uw = w' \]  \hspace{1cm} (C.17)

The \( L \) and \( U \) matrices are obtained by equating elements of the product \( LU \) with \( A \) only where \( A_{i,j} \) is non-zero. For a domain of the size \( 3 \times 2 \times 2 \), the linearization matrix takes the following form

\[
A = \begin{bmatrix}
D_{0,0,0} & \xi_{\max} & 0 & \eta_{\max} & 0 & 0 & \xi_{\max} & 0 & 0 & 0 & 0 & 0 \\
\xi_{\min} & D_{1,0,0} & \xi_{\max} & \eta_{\max} & 0 & 0 & \xi_{\max} & 0 & 0 & 0 & 0 & 0 \\
0 & \xi_{\min} & D_{1,0,0} & 0 & 0 & \eta_{\max} & 0 & 0 & 0 & \xi_{\max} & 0 & 0 \\
\eta_{0,1,0} & 0 & 0 & D_{0,1,0} & \xi_{\max} & 0 & 0 & 0 & 0 & 0 & \xi_{\max} & 0 \\
0 & \eta_{1,1,0} & 0 & \xi_{\min} & D_{1,1,0} & \xi_{\max} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \eta_{2,1,0} & 0 & \xi_{\min} & D_{1,1,0} & 0 & 0 & 0 & 0 & 0 & 0 \\
\xi_{\min} & 0 & 0 & 0 & 0 & \xi_{\max} & 0 & 0 & 0 & \eta_{0,0,1} & 0 & 0 \\
0 & \xi_{\min} & 0 & 0 & 0 & 0 & \xi_{\max} & 0 & \eta_{1,0,1} & 0 & 0 & 0 \\
0 & 0 & \xi_{\min} & 0 & 0 & 0 & 0 & \eta_{0,1,1} & 0 & 0 & \eta_{1,0,1} & 0 \\
0 & 0 & 0 & \xi_{\min} & 0 & 0 & \eta_{0,1,1} & 0 & 0 & \eta_{1,0,1} & 0 & \xi_{\max} \\
0 & 0 & 0 & 0 & \xi_{\min} & 0 & \eta_{0,1,1} & 0 & \xi_{\max} & 0 & \eta_{2,0,1} & 0 \\
0 & 0 & 0 & 0 & 0 & \xi_{\min} & 0 & \eta_{0,1,1} & 0 & \xi_{\max} & 0 & \xi_{1,1,1} \\
0 & 0 & 0 & 0 & 0 & \xi_{\min} & 0 & 0 & \eta_{0,1,1} & 0 & \xi_{\max} & 0 & \xi_{1,1,1} \\
\end{bmatrix}_{(C.18)}
\]

with

\[
x = \begin{bmatrix}
x_{0,0,0} & x_{1,0,0} & x_{2,0,0} & x_{0,1,0} & x_{1,1,0} & x_{2,1,0} & x_{0,0,1} & x_{1,0,1} & x_{2,0,1} & x_{0,1,1} & x_{1,1,1} & x_{2,1,1}
\end{bmatrix}^T
\]
\[
b = \begin{bmatrix}
b_{0,0,0} & b_{1,0,0} & b_{2,0,0} & b_{0,1,0} & b_{1,1,0} & b_{2,1,0} & b_{0,0,1} & b_{1,0,1} & b_{2,0,1} & b_{0,1,1} & b_{1,1,1} & b_{2,1,1}
\end{bmatrix}^T \hspace{1cm} (C.19)
\]
and with zero fill in, $L$ and $U$ are

$$L = \begin{bmatrix}
Ld_{0,0,0} & L\xi_{1,0,0} & L\eta_{0,1,0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & L\xi_{2,0,0} & L\eta_{1,1,0} & 0 & 0 & 0 & Ld_{0,1,0} & Ld_{1,1,0} & 0 & 0 \\
0 & 0 & L\eta_{2,1,0} & 0 & L\xi_{2,1,0} & Ld_{0,1,1,0} & 0 & 0 & 0 & 0 \\
L\xi_{0,0,1} & 0 & 0 & 0 & 0 & 0 & Ld_{0,0,1} & Ld_{1,0,1} & 0 & 0 \\
0 & L\xi_{1,0,1} & 0 & 0 & 0 & 0 & L\xi_{1,0,1} & Ld_{1,0,1} & 0 & 0 \\
0 & 0 & L\xi_{2,0,1} & 0 & 0 & 0 & L\xi_{2,0,1} & Ld_{1,0,1} & 0 & 0 \\
0 & 0 & 0 & L\xi_{0,1,1} & 0 & 0 & L\eta_{0,1,1} & 0 & L\xi_{1,1,1} & Ld_{1,1,1} \\
0 & 0 & 0 & 0 & L\xi_{2,1,1} & 0 & 0 & L\eta_{2,1,1} & 0 & Ld_{1,1,1} \\
0 & 0 & 0 & 0 & L\eta_{2,1,1} & 0 & 0 & L\xi_{2,1,1} & Ld_{1,1,1} & 0 \\
0 & 0 & 0 & 0 & 0 & L\eta_{2,1,1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & L\xi_{2,1,1} & Ld_{1,1,1} & 0 & 0
\end{bmatrix}
$$

(C.20)

$$U = \begin{bmatrix}
1 & L\xi_{0,0,0}^\text{max} & 0 & L\eta_{0,0,0}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{1,0,0}^\text{max} & 0 & L\eta_{1,0,0}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{2,0,0}^\text{max} & 0 & L\eta_{2,0,0}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{0,1,0}^\text{max} & 0 & 0 & 0 & 0 & L\xi_{0,1,0}^\text{max} & 0 & 0 & 0 \\
1 & L\xi_{1,1,0}^\text{max} & 0 & 0 & 0 & 0 & L\xi_{1,1,0}^\text{max} & 0 & 0 & 0 \\
1 & L\xi_{2,1,0}^\text{max} & 0 & 0 & 0 & 0 & L\xi_{2,1,0}^\text{max} & 0 & 0 & 0 \\
1 & L\xi_{0,0,1}^\text{max} & 0 & L\eta_{0,0,1}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{1,0,1}^\text{max} & 0 & L\eta_{1,0,1}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{2,0,1}^\text{max} & 0 & 0 & 0 & 0 & L\eta_{2,0,1}^\text{max} & 0 & 0 & 0 \\
1 & L\xi_{1,1,1}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{1,1,1}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & L\xi_{1,1,1}^\text{max} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
$$

(C.21)

Considering now the implicit Harmonic Balance system of equations, the ILU preconditioner is

$$M^* = L^*U^* \approx A^*$$

(C.22)
where \( A^* \) is given in Eq. (C.5). The lower and upper matrices are

\[
\begin{bmatrix}
L_1 & 0 & \cdots & 0 \\
LH_{2,1} & L_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
LH_{N,1} & \cdots & LH_{N,N-1} & L_N \\
\end{bmatrix}, \quad \begin{bmatrix}
U_1 & UH_{1,2} & \cdots & UH_{1,N} \\
0 & U_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & UH_{N-1,N} \\
0 & \cdots & 0 & U_N \\
\end{bmatrix}
\]

where \( L_i \) and \( U_i \) are the matrices in Eq. (C.20) and (C.21), respectively. For zero fill in, the \( LH_{i,j} \) and \( UH_{i,j} \) are diagonal matrices. The solution to Eq. (C.7) is now

\[
L^*w^* = v^* \\
U^*w^* = w'^*
\]

\[
(C.24)
\]

**C.2.1 HB ILU(0) Derivation**

Continuing with the matrices defined in Eq (C.23), the zero fill in Harmonic Balance Incomplete Lower Upper decomposition preconditioner will be derived. Let \( r \) and \( s \) be the integers indicating the \( r^{th} \) and \( s^{th} \) HB time level. The matrix-matrix operation \( L^*U^* \) is performed and the resulting elements are equated to the non-zero elements in \( A^* \).

The following process is defined for increasing \( r \). For \( s \), where \( 0 \leq s < r \), the equation for the lower \( H \) matrices is

\[
\sum_{p=0}^{s-1} LH_{r,p,i,j,k} UH_{p,s,i,j,k} + LH_{r,s,i,j,k} = H_{r,s,i,j,k}
\]

\[
(C.25)
\]

and the solution gives \( LH_{r,s,i,j,k} \) as

\[
LH_{r,s,i,j,k} = H_{r,s,i,j,k} - \sum_{p=0}^{s-1} LH_{r,p,i,j,k} UH_{p,s,i,j,k}
\]

\[
(C.26)
\]
For \( A_r \), a set of equations is formed

\[
L \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\min}
\]
\[
L \eta_{r,i,j,k} = \eta_{r,i,j,k}^{\min}
\]
\[
L \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\min}
\]
\[
Ld_{r,i,j,k} + L \xi_{r,i,j,k} U \xi_{r,i,j,k} - L \eta_{r,i,j,k} U \eta_{r,i,j,k} - L \xi_{r,i,j,k} U \xi_{r,i,j,k} -1 \\
+ \sum_{p=0}^{r-1} [LH_{r,p,i,j,k} U H_{p,r,i,j,k}] = D_{r,i,j,k}
\]
\[
Ld_{r,i,j,k} U \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\max}
\]
\[
Ld_{r,i,j,k} U \eta_{r,i,j,k} = \eta_{r,i,j,k}^{\max}
\]
\[
Ld_{r,i,j,k} U \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\max}
\]

(C.27)

this equation can be solved to give the elements of \( L_r \) and \( U_r \)

\[
L \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\min}
\]
\[
L \eta_{r,i,j,k} = \eta_{r,i,j,k}^{\min}
\]
\[
L \xi_{r,i,j,k} = \xi_{r,i,j,k}^{\min}
\]
\[
Ld_{r,i,j,k} = D_{r,i,j,k} - L \xi_{r,i,j,k} U \xi_{r,i,j,k} - L \eta_{r,i,j,k} U \eta_{r,i,j,k} - L \xi_{r,i,j,k} U \xi_{r,i,j,k} -1 \\
- \sum_{p=0}^{r-1} [LH_{r,p,i,j,k} U H_{p,r,i,j,k}]
\]
\[
U \xi_{r,i,j,k} = Ld_{r,i,j,k}^{-1} \xi_{r,i,j,k}^{\max}
\]
\[
U \eta_{r,i,j,k} = Ld_{r,i,j,k}^{-1} \eta_{r,i,j,k}^{\max}
\]
\[
U \xi_{r,i,j,k} = Ld_{r,i,j,k}^{-1} \xi_{r,i,j,k}^{\max}
\]

(C.28)

for \( i, j, \) and \( k \) increasing. Lastly, equating the upper \( H \) matrices for \( s \) in \( r < s \leq N \) gives

\[
\sum_{p=0}^{r-1} LH_{r,p,i,j,k} U H_{p,s,i,j,k} + Ld_{r,i,j,k} U H_{r,s,i,j,k} = H_{r,s,i,j,k}
\]

(C.29)
and its solution

\[ UH_{r,s,i,j,k} = Ld_{r,i,j,k}^{-1} \left( H_{r,s,i,j,k} - \sum_{p=0}^{r-1} LH_{r,p,i,j,k} UH_{p,s,i,j,k} \right) \]  

(C.30)

This concludes the process for obtaining the \( L^* \) and \( U^* \) matrices.

Setting up the first line in Eq. (C.24) gives

\[ Ld_{r,i,j,k}w_{r,i,j,k} + L\xi_{r,i,j,k}w_{r,i-1,j,k} + L\eta_{r,i,j,k}w_{r,i,j-1,k} + L\zeta_{r,i,j,k}w_{r,i,j,k-1} + \sum_{p=0}^{r-1} LH_{r,p,i,j,k}w_{p,i,j,k} = v_{r,i,j,k} \]

which can be solved for \( w' \)

\[ w'_{r,i,j,k} = Ld_{r,i,j,k}^{-1} \left( v_{r,i,j,k} - \sum_{p=0}^{r-1} LH_{r,p,i,j,k}w'_{p,i,j,k} \right) + Ld_{r,i,j,k}^{-1} \left( -L\xi_{r,i,j,k}w'_{r,i-1,j,k} - L\eta_{r,i,j,k}w'_{r,i,j-1,k} - L\zeta_{r,i,j,k}w'_{r,i,j,k-1} \right) \]  

(C.31)

for \( r, i, j, \) and \( k \) increasing. Finally, the backward step is

\[ w_{r,i,j,k} + U\xi_{r,i,j,k}w_{r,i+1,j,k} + U\eta_{r,i,j,k}w_{r,i,j+1,k} + U\zeta_{r,i,j,k}w_{r,i,j,k+1} + \sum_{p=r+1}^{N} UC_{r,p,i,j,k}w_{p,i,j,k} = w'_{r,i,j,k} \]  

(C.32)

which has the solution

\[ w_{r,i,j,k} = w'_{r,i,j,k} - \sum_{p=r+1}^{N} UC_{r,p,i,j,k}w_{p,i,j,k} \]  

(C.33)

\[ - U\xi_{r,i,j,k}w_{r,i+1,j,k} - U\eta_{r,i,j,k}w_{r,i,j+1,k} - U\zeta_{r,i,j,k}w_{r,i,j,k+1} \]  

(C.34)

for \( r, i, j, \) and \( k \) decreasing.
Appendix D

Moore-Penrose Generalized Matrix Inverse

Consider the system of equations

$$Bz = c$$  \hspace{1cm} (D.1)

where $B$ is not square. This is the case for the filter operator in Section 2.3.2.3. The methods discussed in Chapter C require a square matrix and therefore cannot solve this system of equations. Instead, a Moore-Penrose matrix inversion is performed. First, both sides are multiplied by the conjugate transpose of $B$, giving

$$B^H Bz = B^H c$$  \hspace{1cm} (D.2)

where the conjugate transpose, $B^H$, is equivalent to the transpose, $B^T$, if all of the elements of $B$ are real. If the inverse of $B^H B$ exists, then

$$z = B^+ c$$  \hspace{1cm} (D.3)

where

$$B^+ = (B^H B)^{-1} B^H$$  \hspace{1cm} (D.4)

is the Moore-Penrose generalized inverse. The term $(B^H B)^{-1}$ is square and therefore can be solved with direct or iterative methods discussed in Chapter C.