CFD PREDICTIONS OF THE ROTORDYNAMIC COEFFICIENTS OF A LABYRINTH SEAL USING A NOVEL DOMAIN-DECOMPOSITION PRECONDITIONER

A Dissertation

by

NEIL ROY MATULA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Paul G. A. Cizmas
Committee Members,	Adolfo Delgado
	Thomas Strganac
	Raytcho Lazarov
Head of Department,	Rodney D. W. Bowersox

May 2020

Major Subject: Aerospace Engineering

ABSTRACT

This dissertation presents methods for the prediction of the rotordynamic coefficients of annular gas seals using computational fluid dynamics (CFD). Improvements to an in-house Navier–Stokes solver for the purpose of investigating seal flows are discussed in this work. These improvements include an implicit method using an iterative linear solver. Additionally, a novel domain-decomposition preconditioner is developed in this work.

The solver improvements are demonstrated on small test problems. Simulations are then presented for the prediction of seal rotordynamic coefficients. The results are compared with experimental data from the literature.

To my family

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Paul Cizmas, for the knowledge he has shared with me, for finding funding for my work, and for the remarkable patience he has shown me over the years. I would also like to thank my committee members, Dr. Adolfo Delgado, Dr. Thomas Strganac, and Dr. Raytcho Lazarov, for their time and feedback. I further thank Dr. Delgado for agreeing to serve on my committee on short notice, and for his help in securing funding for some of this work. I also thank Michael Golla for the teaching position he awarded me.

I would like to thank my friends and fellow graduate students, Robert Brown, Forrest Carpenter, Raymond Fontenot, Brian Freno, Nathan Jones, Reza Karimi, Elizabeth Krath, David Liliedahl, Michael Polewski, and Allen Ream. Several deserve special thanks. Forrest has been my office mate, collaborator, and close friend for much of my journey. Reza helped me find a teaching job during a difficult time in my life. Finally, Brian pushed me to succeed when the road seemed endless, and helped me to find employment after graduate school. Thank you all.

I thank the Texas A&M High Performance Research Computing Center for the use of their systems, and the Turbomachinery Research Consortium for funding much of this work, as well as the other funding sources named below.

Finally, I thank my parents, John and Sharon, my sister, Shalane, and my grandmother, Dorothy. You all helped me during trying times, and provided me with hope and a reason to succeed. You are the reason I am here.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Paul Cizmas [advisor] and Professors Adolfo Delgado of the Mechanical Engineering Department, Thomas Strganac of the Aerospace Engineering Department, and Raytcho Lazarov of the Math Department.

The swirl vane grid generator in Chapter III was developed in collaboration with Professor Forrest Carpenter. All other work conducted for this dissertation was completed independently by the student.

Funding Sources

This work was made possible in part by contributions from:

1. Vestas Wind Systems,
2. Blue Energy,
3. Turbomachinery Research Consortium,
4. Slipstream Wind, and
5. the Graduate Teaching Fellowship.

Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the above parties.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGEMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES x

CHAPTER I INTRODUCTION 1

I.1. Statement of the Problem 1

I.2. Original Contributions 7
I.3. Dissertation Outline 8

CHAPTER II PHYSICAL MODEL 9

II.1. Navier–Stokes Equations 9
II.2. Favre– and Reynolds–Averaged Navier–Stokes Equations 11
II.3. Turbulence Modeling 14
II.4. Equation Nondimensionalization 17

CHAPTER III NUMERICAL METHODS 18

III.1. Grid Generation 18
III.1.1. Plain and Labyrinth Seal Grid Generation . . . 18
III.1.2. Swirl Brake Grid Generation 21
III.2. Flow Solver 25
III.2.1. Spatial Discretization 25

PAGE

III.2.1.1. Semi-Discrete Form 27
III.2.1.2. Nonlinear Solution Strategies 28
III.2.1.3. Implicit Scheme 31
III.2.1.4. Linear Solution Strategies 33
III.2.1.5. Preconditioning Strategies 40
III.2.1.6. A Novel Preconditioner 45

CHAPTER IV VERIFICATION AND VALIDATION RESULTS 52

IV.1. Verification of Implicit Solver 53
IV.1.1. Verification of Superlinear Convergence 53
IV.1.2. Evaluation of Preconditioner Effectiveness . . . 54
IV.1.2.1. Additive-Schwarz Preconditioner 54
IV.1.2.2. New Preconditioner 55
IV.2. Validation of Solver for Annular Gas Seal Flows 60
IV.2.1. Nelson Seal 60
IV.3. Case Definition 61
IV.4. Grid Generation 63
IV.5. Results 65
IV.6. Wright Seal 67
IV.6.1. Case Definition 68
IV.6.2. Grid Generation 70
IV.6.3. Results 74

CHAPTER V CONCLUSIONS 76

CHAPTER VI FUTURE WORK 78

REFERENCES 79

LIST OF FIGURES

FIGURE		Page
III.1	Grid block configuration for labyrinth seals.	19
III.2	Example multiblock grid around a labyrinth seal tooth.	20
III.3	Basic structured grid topologies: a) H-grid, b) C-grid, and c) O-grid.	21
III.4	O4H grid topology. Block I is an O-grid, and Blocks II-V are H-grids.	23
III.5	Illustration of the Additive-Schwarz decomposition.	41
IV.1	Computational grid for laminar channel flow.	55
IV.2	Convergence rate of the laminar channel flow test case.	56
IV.3	Wallclock time comparison for the laminar channel flow test case.	57
IV.4	Convergence of the GMRES algorithm for the 101^{st} iteration of the laminar channel problem.	58
IV.5	Improvement in the required number of search directions required fr the laminar channel problem.	60
IV.6	Improvement in the required number of search directions required for the Wright seal problem.	61
IV.7	Nelson seal geometry.	62
IV.8	Overview of the coarse computational grid for the Nelson seal.	64
IV.9	Cutaway view of the coarse computational grid for the Nelson seal.	64
IV.10	Grid independence study for the Nelson seal.	65
IV.11	Direct stiffness versus pressure ratio for the Nelson seal.	66

FIGURE Page

IV.12 Direct damping versus pressure ratio for the Nelson seal. 67

IV.13 Geometry of the Wright seal. 69

IV.14 Grid block configuration for labyrinth seals. 71

IV.15 Overview of the computational grid for the Wright seal. 72

IV.16 Cross-section view of the computational grid for the Wright seal. . 73

IV.17 Grid independence study for the Wright seal. 73

IV.18 Effective radial stiffness versus pressure difference for the Wright seal. 75

IV.19 Excitation constant versus pressure difference for the Wright seal. 75

LIST OF TABLES

TABLE Page

IV.1 Nelson seal operating conditions. 62

IV.2 Grid dimensions for the Nelson seal. 63

IV.3 Wright seal operating conditions. 70

IV.4 Grid dimensions for the grid independence study for the Wright seal. 72

CHAPTER I

INTRODUCTION

I.1. Statement of the Problem

In high-speed turbomahinery, clearances must necessarily exist between the rotating and stationary components to mitigate friction and machine wear. A true seal is therefore impossible, and the designer can, at best, hope to limit the leakage flow using non-contacting seals. If a perturbation causes the shaft to become offset from the centerline, the leakage flow creates a non-uniform pressure distribution around the seal, which results in lateral forces on the shaft. These lateral forces may act to amplify the initial perturbation, leading to expensive maintenance and, in the most extreme cases, machine failure. Historical examples of problems caused by these forces may be found in [Childs, 2013]. The turbomachine designer, therefore, requires methods to predict the stability characteristics of a rotor-seal system in order to ensure reliable, safe, and efficient operation of the machine under design conditions; the study and development of these methods is part of the field of rotordynamics.

The most popular method of quantifying the stability characteristics of a seal-rotor system is to model the forces on the shaft using a two-dimensional mass-spring-damper system with displacement- and velocity-independent (but possibly frequency-dependent) coefficients, although the mass coefficients are often neglected. Since this model is linear with respect to the displacements, it inherently assumes that the

displacements will be small. If the seal is assumed to be axisymmetric and the shaft is nominally centered, this model leads to four coefficients (neglecting mass) that can quantify the stability of the seal-rotor system; these are the direct stiffness, cross-coupled stiffness, direct damping, and cross-coupled damping. The direct stiffness and cross-coupled damping have little effect on the stability of the system, and instead influence the natural frequencies and critical speeds of the system. The cross-coupled stiffness and direct damping, on the other hand, determine whether the perturbations grow or decay. These two coefficients are often combined into one, called the effective damping coefficient, that functions as a quantititative measure of the stability of the system. If everything else is held equal, the best seal from the perspective of reducing subsynchronous whirl is the one that has the largest (positive) effective damping [John Vance, 2010].

The most common type of annular gas seal is the labyrinth seal, which uses multiple teeth, or "blades", to create a winding path for the fluid to follow, thereby reducing leakage. They are popular because of their simplicity and ease of manufacture; however, they are also known to have poor rotordynamic properties [John Vance, 2010]. Experiments conducted in the 1970's [Benckert & Wachter, 1980] established a link between the tangential velocity of the fluid entering the seal, known as swirl, and the cross coupled stiffness coefficients. In particular, positive swirl (that is, in the same direction as the shaft rotation) tends to result in large positive cross-coupled stiffness coefficients, which degrades the stability of the system. Three common options exist for addressing this issue. First, the designer may choose a more sophisticated type of seal, such as a honeycomb, hole pattern, or pocket damper seal, that

is designed to prevent the large swirl velocities within the cavities of the seal, thereby mitigating the follower force. This will solve the vibration issue at the expense of higher manufacturing costs. Second, an auxiliary flow may be injected opposite the direction of shaft rotation, thereby imbuing the fluid entering the seal with negative swirl. This method is known as shunt injection. This will result in the desired negative cross-coupled stiffness at the expense of additional design complexity. Finally, small vanes, known as preswirl brakes, may be mounted upstream of the seal to create negative swirl. This will improve the stability of the system with a lower manufacturing cost than that of the more sophisticated seal types, and with less design complexity than that of shunt injection. However, the flow around a swirl brake is necessarily separated and difficult to predict, and guidelines for choosing the dimensions of a swirl brake to match a particular application do not currently exist.

Bulk flow models are routinely used for predicting the rotordynamic coefficients for annular gas seals. However, they rely on certain problem-specific parameters that must be tuned using experimental data from similar geometries and conditions, and therefore are not entirely predictive in nature. They also do not give insight into the features of the flow within the seal. Computational fluid dynamics (CFD) simulations are becoming more common in the academic rotordynamics community, but are widely recognized as too computationally expensive for routine calculations [Childs, 2013]. The most common simulations take place in a reference frame attached to the whirling rotor. These are sometimes referred to as "quasi-steady" simulations, because in this reference frame the geometry does not change with respect to time;

however, the fluid motion may still be inherently unsteady even within this reference frame. Quasi-steady simulations necessitate that the fluid domain remain static. If the whirling shaft is assumed to move in a circular orbit with a constant period, the problem may be expressed in a coordinate system attached to the whirling rotor, thus creating a static fluid domain. However, this technique requires an axisymmetric stator geometry, and therefore does not apply to the more advanced types of seals mentioned above, nor to seals equipped with a swirl brake. It also precludes the use of shaft orbits that are not circular. For those cases the analyst must resort to fully unsteady time-domain simulations of the full Navier-Stokes equations with an appropriate turbulence model. Simulations for predicting the rotordynamic coefficients of a seal equipped with a swirl brake are noticeably lacking from the literature, likely due to the enormous computational burden involved and low likelihood of matching experimental data. As a result, most CFD simulations of swirl brake flows have fixated on predicting the swirl at the entrance to the seal. This is more computationally tractable due to the assumption of spatial periodicity, which is not valid for the computation of rotordynamic coefficients because it requires that the shaft be centered. However, these simulations that only predict the seal-entrance swirl are not experimentally verifiable, as the clearances involved are too small to permit direct measurement of the swirl velocity generated by the swirl brake. Hence, the state of the art is split between two extremes: relatively cheap but unverifiable predictions of the swirl generated by a preswirl brake, and computationally intractable (but verifiable) predictions of the full rotordynamic coefficients of a seal equipped with a swirl brake. To the author's knowledge, the second type of simulation has

never been attempted.

As a step toward this kind of simulation, this work is focused on the prediction of the full set of rotordynamic coefficients for seal flows using in-house CFD software. Given the computational burden of these simulations, several smaller cases are used to gauge the effectiveness of the software before the largest simulations are attempted, and all results will be compared to experimental data. A novel domain-decomposition preconditioner is developed that, in combination with an implicit time-integration method, will be leveraged to make these simulations possible.

Background

The foundational experiments of Benckert & Wachter [1980] established a strong link between the swirl velocity of the fluid entering a labyrinth seal and the associated cross-coupled stiffness coefficients. In particular, positive swirl (in the same direction as shaft rotation) leads to positive cross-coupled stiffness coefficients, which have a destabilizing effect. They also demonstrated that swirl brakes mounted upstream of the seal can dramatically improve the situation by reducing the swirl velocity entering the seal. In extreme cases, the swirl direction can be completely reversed along with the sign of the stiffness coefficients, leading to improved effective damping. Since then, swirl brakes have been used extensively to help solve stability problems with various kinds of turbomachinery, although cases in which instabilities were completely eliminated by the introduction of a swirl brake appear to be rare [Childs, 2013].

Iwatsubo [1980] and Kurohashi [1980] published the earliest useful one-control-volume models for labyrinth seals [Childs, 2013]. Childs & Scharrer [1986] expanded on these methods, and Wyssmann et al. [1984] and Scharrer [1988] developed two-control-volume methods. These models are very computationally cheap, but lack flow detail and require parametric tuning from experimental data for a similar seal under similar conditions.

CFD techniques are slowly becoming accepted in the rotordynamics community. Most authors use a "quasi-steady" framework, in which the reference frame is attached to the whirling rotor. Within this reference frame, the geometry does not change with respect to time, and therefore moving meshes are unnecessary. Rhode et al. [1992] performed a quasi-steady analysis of a tooth-on-rotor labyrinth seal with three teeth, showing excellent agreement with experiment; however, only one flow condition was evaluated. Moore [2003] examined a tooth-on-stator labyrinth seal. His CFD predictions showed only modest agreement with the Pelletti's experiments [Pelletti, 1990]; however, the two bulk flow methods that he considered fared no better. Hirano et al. [2003] examined several tooth-on-stator labyrinth seals, with only moderate agreement with bulk flow models.

For non-axisymmetric geometry, the only recourse is a full 3D transient analysis. Chochua & Soulas [2007] performed a transient analysis of a hole pattern seal and showed good agreement with experiment. Yan et al. [2011] also examined a hole-pattern seal with moderate agreement. Li et al. [2013] analyzed several kinds of seal in this manner; while the predicted coefficients agreed only modestly with the experiment for the labyrinth seal, the effective damping agreed very well at high

frequencies.

For seals equipped with a swirl brake, the CFD analyses to date have focused on predicting the swirl at the entrance to the seal, rather than the rotordynamic coefficients of the combined seal and swirl brake. This has the additional benefit of permitting the assumption of spatial periodicity, thus reducing the cost of the simulation by approximately two orders of magnitude. Soghe et al. [2013]), Baldassarre et al. [2014], Matula & Cizmas [2017], and Nielsen et al. [2001]) considered the effect of various geometric design parameters on seal entrance swirl.

I.2. Original Contributions

The goal of this dissertation is to develop novel computational methods to improve the performance of a given flow solver, and use the improved solver to facilitate the prediction of the rotordynamic coefficients of seal flows. This dissertation builds on the work of previous students, including Kim [2003], Gargoloff [2007], Brown [2016], and Carpenter [2016].

A grid generator for turbomachinery flows was developed in collaboration with Carpenter [2014]. It uses a five-block structured topology to generate the blade-to-blade grid around typical turbomachinery blades and vanes, and includes an additional two structured blocks to capture the clearance region. The five blocks are generated in tandem in order to provide smooth transitions across the block boundaries. In addition, several smaller grid generation codes were written to handle the various seal geometries encountered in this work.

A novel preconditioner was developed for the linear system arising from a lin-

earized backward Euler discretization of the governing equations. The preconditioner is based on a non-overlapping Additive-Schwarz method with an auxiliary coarse space to couple the subdomains. The novel aspect of the preconditioner lies in its use of a Quasi-Newton method to solve the coarse-grid problem without requiring the generation or storage of said coarse-grid problem, and without requiring any additional function evaluations beyond those already needed by the primary iterative method.

Two sets of experimental results for seal flows are considered for demonstrating the ability of the flow solver to adequately predict rotordynamic forces in annular gas seals. The first case is a straight (smooth seal), and the second is a labyrinth seal.

I.3. Dissertation Outline

Chapter I discusses the governing equations of the physical model. Chapter II discusses the grid generator, and the discretization and solution methods used in the flow solver. This chapter ends with a discussion of the new preconditioner. Chapter III presents results of several small cases that demonstrate the effectiveness and correct implementation of the implicit solver and new preconditioner. This chapter then discusses the results of the three seal validation cases. Finally, Chapter IV presents conclusions that may be drawn from the results of Chapter III, and Chapter V provides recommendations for future work.

CHAPTER II

PHYSICAL MODEL

This chapter begins with a statement the governing equations for the motion of compressible Newtonian fluid. The equations are then spatially averaged to facilitate the computation of turbulent flows. Finally, an appropriate turbulence model is discussed.

II.1. Navier–Stokes Equations

The flow of a compressible fluid is governed by the conservation of mass, momentum, and energy. Under the continuum assumption, these may be written as a system of partial differential equations (PDEs). When written for a Newtonian fluid, some authors refer to the momentum conservation equation as the Navier–Stokes equation, while some use that name to refer to the complete set of equations; we take the latter approach here. The full set of equations in three spatial dimensions may be written as

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) &= \rho \mathbf{f} - \nabla p + \nabla \cdot \boldsymbol{\tau} \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho H \mathbf{u}) &= \nabla \cdot (k \nabla T + \boldsymbol{\tau} \cdot \mathbf{u}) + \dot{q}, \end{aligned} \tag{2.1}$$

where ρ is the density, $\mathbf{u} = (u, v, w)^\intercal$ is the velocity expressed in an inertial reference frame, $\mathbf{f}$ is the vector of mass-specific volume forces, p is the pressure, $\boldsymbol{\tau}$ is the viscous

stress tensor, E is the mass-specific total energy, H is the mass-specific stagnation enthalpy, T is the thermodynamic temperature, k is the thermal conductivity coefficient, and $\dot{q}$ includes the work done by the volume forces and the energy addition due to any source other than conduction. The total energy and stagnation enthalpy are

$$E = e + \frac{\mathbf{u}^2}{2}, \qquad\qquad H = h + \frac{\mathbf{u}^2}{2} \tag{2.2}$$

where e and $h = e/\rho$ are the internal energy and static enthalpy, respectively. For a Newtonian fluid, under the assumption of Stokes' hypothesis, the stress tensor is

$$\boldsymbol{\tau} = \mu \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^{\intercal} - \frac{2}{3} (\nabla \cdot \mathbf{u}) \mathbf{I} \right], \tag{2.3}$$

where μ is the dynamic viscosity coefficient.

For a calorically perfect gas the equation of state is the ideal gas law,

$$p = \rho R T \,. \tag{2.4}$$

where R is the specific gas constant. For dry air we use the value R =287.16$\frac{\mathrm{J}}{\mathrm{kg \cdot K}}$. The mass-specific static enthalpy and internal energy are proportional to temperature for a calorically perfect gas:

$$h = c_p T, \qquad\qquad e = c_v T \tag{2.5}$$

where c_p is the specific heat capacity at constant pressure, and c_v is the specific heat capacity at constant volume.

At this point, we have 12 unknowns: $\rho, u, v, w, p, T, e, E, h, H, k, \mu$. The five governing equations, together with Eqs. 2.2-2.5, form a total of 10 equations. In

order to close the system, we require two more equations. The dynamic viscosity may be related to temperature using Sutherland's law, which may be written as

$$\mu(T) = \frac{CT^{3/2}}{T + S}, \tag{2.6}$$

where C and S are constants. For air, the values used were $C = 1.458 \times 10^{-6}$ and $S = 110.4$ K. The thermal conductivity coefficient may be related to the dynamic viscosity using

$$k = c_p \frac{\mu}{Pr}, \tag{2.7}$$

where Pr is the Prandtl number, which is assumed to have a constant value of 0.72 for air.

II.2. Favre– and Reynolds–Averaged Navier–Stokes Equations

The flow is turbulent in most cases of practical interest. The grid resolution and solution time requirements for the direct solution of the Navier–Stokes equations scale with $Re^{9/4}$ and Re^3, respectively [Blazek, 2005, p. 227], which is prohibitive for most engineering problems. Hence, direct solution of realistic engineering configurations remains out of reach for the foreseeable future. An averaging procedure is typically used to produce an approximation of the governing equations with more modest resolution requirements.

Reynolds averaging is used to decompose the flow variables into mean and fluctuating parts. Several different interpretations exist for the "mean" [Blazek, 2005,

p.231], but here, the mean is defined with respect to time:

$$\overline{w} = \frac{1}{T}\int_{t}^{t+T} w dt,$$

where T is as small as possible while still being large relative to the charactistic time scales of the turbulent fluctuations, w is a generic scalar variable, and $\overline{w}$ is the Reynolds average of w. The decomposition is then

$$w = \overline{w} + w',$$

where w' indicates the fluctuating part of w. In the event that density also fluctuates, an additional averaging procedure, called Favre averaging, is often used to simply the form of the averaged governing equations. The Favre average is

$$\tilde{w} = \frac{1}{\overline{\rho} T}\int_{t}^{t+T} \rho w dt,$$

and the associated decomposition is

$$w = \tilde{w} + w'',$$

where $\tilde{w}$ indicates the Favre average of w, and w'' indicates the fluctuating part. Most often, Reynolds averaging is used for the density and pressure, and all other variables are treated with Favre averaging [Blazek, 2005, p.232]. Several rules may

be derived for the averaging procedures:

$$\overline{w'} = 0,$$

$$\tilde{w''} = 0,$$

$$\overline{v'w'} \neq 0,$$

$$\tilde{v''w''} \neq 0,$$

$$\tilde{\rho w} = \overline{\rho}\tilde{w},$$

$$\overline{\rho w''} = 0,$$

$$\overline{w''} \neq 0.$$

Using Reynolds averaging for the pressure and density, and Favre averaging to all other variables, the Navier–Stokes equations become [Blazek, 2005, p. 234]

$$\begin{aligned}
\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho}\tilde{\mathbf{u}}) &= 0 \\
\frac{\partial (\bar{\rho}\tilde{\mathbf{u}})}{\partial t} + \nabla \cdot (\bar{\rho}\tilde{\mathbf{u}} \otimes \tilde{\mathbf{u}}) &= \nabla \bar{p} + \nabla \cdot (\tilde{\boldsymbol{\tau}} + \boldsymbol{\tau}_R) \\
\frac{\partial \left(\bar{\rho}\tilde{E}\right)}{\partial t} + \nabla \cdot \left(\bar{\rho}\tilde{H}\tilde{\mathbf{u}}\right) &= \nabla \cdot \left(k\nabla\tilde{T} + (\tilde{\boldsymbol{\tau}} + \boldsymbol{\tau}_R) \cdot \tilde{\mathbf{u}} - f_R\right),
\end{aligned} \tag{2.8}$$

Together these are known as the Favre- and Reynolds-Averaged Navier–Stokes equations. Note that the key difference between these and the original equations is the presence of additional stresses, $\boldsymbol{\tau}_R$, and an additional component of the heat flux, f_R. The specification of these additional terms is the responsibility of the turbulence model.

The Boussinesq hypothesis is used to relate the Reynolds stress to the mean

velocity gradient, as:

$$\boldsymbol{\tau}_R = \mu_T \left[\nabla \tilde{\mathbf{u}} + (\nabla \tilde{\mathbf{u}})^{\mathsf{T}}\right] - \frac{2}{3}\left(\nabla \cdot \tilde{\mathbf{u}}\right)\mathbf{I}, \tag{2.9}$$

where μ_T is the eddy viscosity. The turbulent heat flux is likewise related to the mean temperature gradient, as:

$$f_R = -k_T \nabla \tilde{T} \tag{2.10}$$

where k_T is the turbulent thermal conductivity coefficient, which is calculated as:

$$k_T = c_p \frac{\mu_T}{Pr_T} \tag{2.11}$$

where Pr_T is the turbulent Prandtl number. The turbulent Prandtl number is assumed to maintain a constant value of 0.9 throughout the flow. The only remaining difficulty is the prescription of the eddy viscosity coefficient, which is the responsibility of the turbulence model.

II.3. Turbulence Modeling

The Boussinesq hypothesis used in the previous section introduces a new variable into the governing equations (the eddy viscosity, μ_T), but did not introduce any further information, leading to an underdetermined system. Supplemental equations must be used to close the system. Two equation models provide the simplest complete models of turbulence [Wilcox, 2010, pg. 122], in the sense that the models are free from flow-dependent specifications [Pope, 2000, pg. 338]. The two most common two-equation turbulence models are the $\kappa - \epsilon$ and $\kappa - \omega$ models. The $\kappa - \omega$ is known to perform well in the boundary layer, but is sensitive to farfield conditions.

The $\kappa - \epsilon$ model is much less sensitive to farfield conditions, but does not perform as well in the boundary layer. The reader may consult [Menter, 1993] for a more complete comparison of the models. In that same work, Menter proposed a blending of $\kappa - \epsilon$ and $\kappa - \omega$ models that takes advantage of the strengths of each. This model is known as the Shear Stress Transport (SST) turbulence model. The model has undergone several revisions over its lifetime, and the updates due to Menter et al. [2003] and Hellsten [1998] are used in this work.

The model equations for the SST turbulence model are [Carpenter, 2016]

$$\begin{aligned}
\frac{\partial \rho\kappa}{\partial t} + \nabla \cdot (\rho \mathbf{u} \kappa) = &\quad P_\kappa - \beta^* \rho\omega\kappa + \nabla \cdot \left[\left(\mu + \sigma_\kappa \mu_T \right) \nabla \kappa \right] \\
\frac{\partial \rho\omega}{\partial t} + \nabla \cdot (\rho \mathbf{u} \omega) = &\frac{\rho\alpha}{\mu_T} P_\kappa - \beta \rho \omega^2 \quad + \nabla \cdot \left[\left(\mu + \sigma_\omega \mu_T \right) \nabla \omega \right] \\
&+ \left[2\rho \left(1 - F_1 \right) \sigma_{\omega 2} \right] \frac{\nabla \kappa \cdot \nabla \omega}{\omega},
\end{aligned} \tag{2.12}$$

where κ is the turbulent kinetic energy, ω is the turbulence dissipation rate, P_κ is the production of κ, and α, β, β^*, σ_κ, σ_ω, and $\sigma_{\omega 2}$ are model constants. The function F_1 will be defined shortly. The production term is given by

$$P_\kappa = \boldsymbol{\tau}_R : \nabla \mathbf{u}$$

where : represents a double contraction operation. (Note: two definitions are commonly used for the double contraction. The one used here is $\mathbf{A} : \mathbf{B} = \mathrm{tr}\left(\mathbf{A}\mathbf{B}^T\right)$.) The constants of the model are weighted averages of the $\kappa - \omega$ and $\kappa - \epsilon$ model constants:

$$\phi = (F_1)\phi_1 + (1 - F_1)\phi_2,$$

where ϕ represents an arbitrary model constant, and the subscripts 1 and 2 indicate the $\kappa - \omega$ and the $\kappa - \epsilon$ models, respectively. The model constants for the $\kappa - \omega$

model are given by [Menter et al., 2003]

$$\sigma_{\kappa 1} = 0.850, \qquad \sigma_{\omega 1} = 0.500,$$

$$\beta_1 = 0.0750, \qquad \alpha_1 = 0.55\bar{5},$$

and those for the $\kappa - \epsilon$ model are given by

$$\sigma_{\kappa 2} = 1.000, \qquad \sigma_{\omega 2} = 0.856,$$

$$\beta_2 = 0.0828, \qquad \alpha_2 = 0.440.$$

The blending function F_1 is evaluated at each point in space as follows:

$$F_1 = \tanh\left(arg_1^4\right) \tag{2.13}$$

$$arg_1 = \min\left[\max\left(\frac{\sqrt{\kappa}}{0.09\omega d}, \frac{500\mu}{\rho\omega d^2}\right), \frac{4\rho\sigma_{\omega 2}K}{CD_{\kappa\omega}d^2}\right], \tag{2.14}$$

where d is the distance to the nearest wall. The cross-diffusion term in Eq. 2.14 is the positive portion of the last term in Eq. 2.12, and is given by

$$CD_{\kappa\omega} = \max\left(2\frac{\rho\sigma_{\omega 2}}{\omega}\nabla\kappa \cdot \nabla\omega, 10^{-20}\right). \tag{2.15}$$

The value of the limiter (10^{-20}) in Eq. 2.15 varies in the literature. The current value is from [Menter et al., 2003]. The eddy viscosity is given by:

$$\mu_T = \frac{a_1\rho\kappa}{\max\left(a_1\omega, |\mathbf{S}|F_2\right)}, \tag{2.16}$$

where $a_1 = 0.31$ and

$$\mathbf{S} = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal\right)$$

is the mean strain rate tensor. The blending function F_2 is given by

$$F_2 = \tanh\left(arg_2\right)$$

$$arg_2 = \max\left(\frac{2\sqrt{\kappa}}{0.09\omega d}, \frac{500\mu}{\rho\omega d^2}\right).$$

II.4. Equation Nondimensionalization

The differential equations (2.8) and (2.12) are written in terms of dimensional variables. If these equations are implemented directly in software form, there will be two consequences: 1) the software will have an assumed system of units, and 2) the flow variables and fluxes will have wildly different orders of magnitude, thereby placing limits on the level of accuracy that can be obtained, and making interpretation of the convergence level difficult for the user. The nondimensionalization procedure used here follows that of Carpenter [2016]. The nondimensionalization of each variable is of the form $\hat{\phi} = \phi/\phi_{ref}$, where ϕ is the variable to be nondimensionalized, the subscript "ref" indicates a reference value, and a hat indicates the nondimensional quantity. The values of the reference quantities for each dimensional variable are given by

$$
\begin{aligned}
\hat{\mathbf{x}} &= \frac{\mathbf{x}}{l}, & \hat{t} &= \frac{t}{l/c_\infty}, & \hat{\mathbf{u}} &= \frac{\mathbf{u}}{c_\infty} \\
\hat{\rho} &= \frac{\rho}{\rho_\infty}, & \hat{p} &= \frac{p}{\rho_\infty c_\infty^2}, & \hat{E} &= \frac{E}{c_\infty^2} \\
\hat{H} &= \frac{H}{c_\infty^2}, & \hat{T} &= \frac{T}{T_\infty}, & \hat{\mu} &= \frac{\mu}{\mu_\infty} \\
\hat{\mu}_T &= \frac{\mu_T}{\mu_\infty}, & \hat{\kappa} &= \frac{\kappa}{c_\infty^2}, & \hat{\omega} &= \frac{\omega}{c_\infty/l}.
\end{aligned}
$$

CHAPTER III

NUMERICAL METHODS

This chapter begins by describing the turbomachinery grid generators that were developed for turbomachinery flows. Next, the discretization of the governing equations used by the flow solver is discussed. Following this, the modifications to the flow solver are discussed, including the implicit solver and novel preconditioning strategy.

III.1. Grid Generation

A number of commercial software packages exist that are capable of generating CFD grids for turbomachinery applications; examples include Pointwise and Autogrid. However, writing custom software for grid generation has three distinct advantages: 1) the user has full control over all aspects of the grid, 2) numerous grids with the same topology but slightly different geometrical features can be "mass-produced" without much user intervention, and 3) since the user is also the developer, all software-related difficulties may be directly handled without need for technical support. For these reasons, all of the grid generation for this work was handled with in-house FORTRAN programs. The following sections discuss the various grid generators that were developed for annular gas seal flows.

III.1.1. Plain and Labyrinth Seal Grid Generation

The procedure for the generation of plain and labyrinth seals using a multiblock structured grid is straightforward, as the individual blocks may be generated using

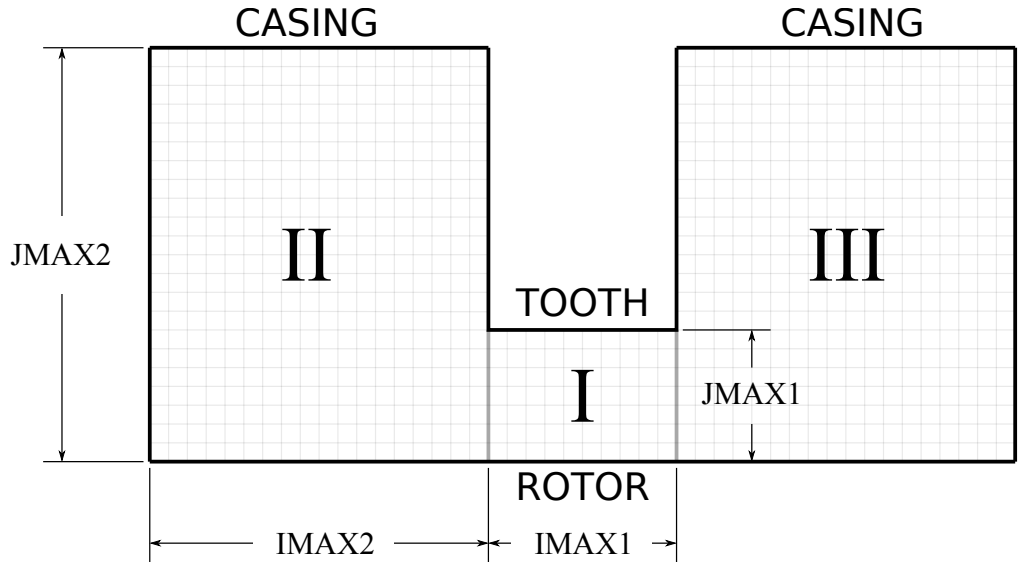

Figure III.1: Grid block configuration for labyrinth seals.

algebraic interpolation; that is, no PDE- or optimization-based grid-generation systems are necessary. The only non-trivial portion of the grid generation process is the selection of clustering functions. Some authors, such as Thompson et al. [1985], prefer to use algebraic functions for the clustering process. However, the ratio of adjacent element sizes is a critical parameter for grid point clustering, and these simple functions do not allow complete control over this ratio. A more direct approach would be to choose the clustering such that the ratio of adjacent element sizes is constant throughout the stretched region. This leads to a geometric series for the point distribution, which may be solved through Newton-Rhapson iteration for the ratio between adjacent element sizes. A similar procedure is used for clustering at both end points.

With the clustering algorithm in hand, all that remains is the designation of

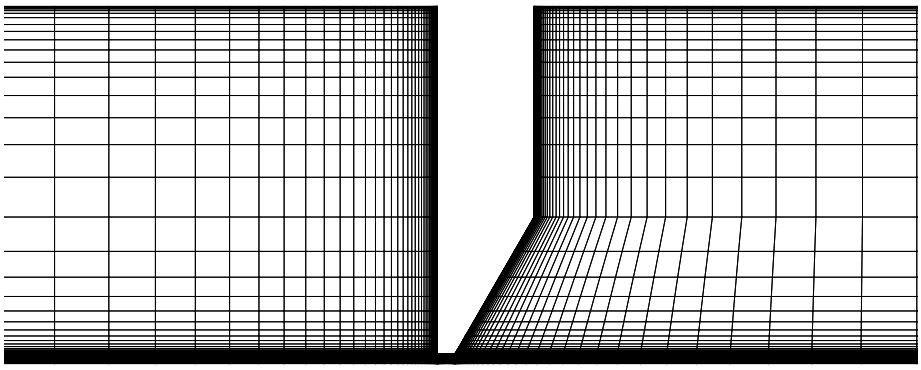

Figure III.2: Example multiblock grid around a labyrinth seal tooth.

block boundaries for the multiblock grid. For the types of seals encountered in this work, a straightforward choice of block boundaries that results in nearly perfect orthogonality measures is shown in Fig. III.1. The user provides the grid dimensions IMAX1, IMAX2, JMAX1, and JMAX2, with the only restriction being that JMAX2 must be greater than JMAX1. Blocks II and III share the same dimensions. These three blocks are copied for the total number of teeth required by the seal. An example of a seal tooth that is meshed in this fashion is shown in Fig. III.2. This two-dimensional grid is then revolved to create the full annulus. In order to move the numerically troublesome truncation boundaries (inlet and outlet) further away from the area of interest, additional structured padding grids are placed upstream of the first tooth and downstream of the last tooth in a straightforward manner.

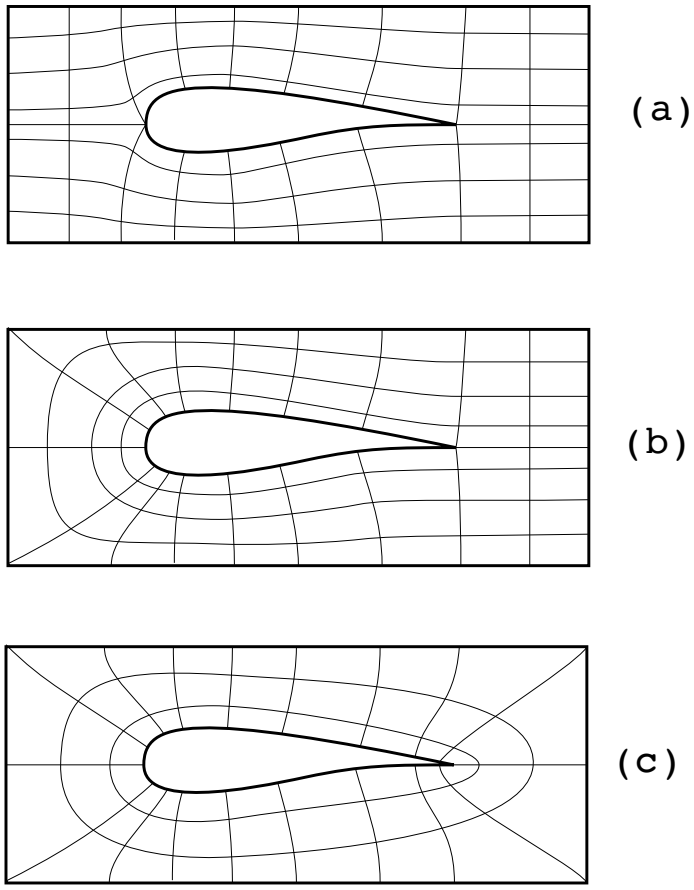

Figure III.3: Basic structured grid topologies: a) H-grid, b) C-grid, and c) O-grid.

III.1.2. Swirl Brake Grid Generation

The procedure used for the generation of grids for swirl vane geometries is more sophisticated. The three-dimensional grid is formed from topologically identical $x - \theta$ layers that are stacked in the radial direction. The "cap" grid that covers the clearance of the vane is then added separately. The radial distribution of the layers is based on the clustering procedure discussed above.

Several different topology configurations are popular in turbomachinery grid

generation. The most common ones are depicted in Fig. III.3 . These standard configurations all resemble letters of the alphabet, and so are given the names *H-grid*, *C-grid*, and *O-grid* respectively. The defining feature of each of these grids is the location of slope discontinuities on the boundaries. For the H-grid, these may appear at the leading and trailing edges of the airfoil. If the leading or trailing edge is "cusped", that is, if the suction and pressure sides terminate with the same slope, then no slope discontinuity must occur. However, if the edge has a finite termination angle, a slope discontinuity equal to half of that angle results, and if the edge is rounded, then a slope discontinuity of 90 degrees results. Any discontinuities in the boundary tend to propagate into the interior of the grid, thereby damaging the overall grid quality. For this reason, H-grids are ideal for airfoils with sharp leading and trailing edges, but less so for those with rounded edges. The O-grid has the entire airfoil as its inner boundary, and hence has the opposite problem: it is ideal for airfoils with rounded edges. However, the O-grid also carries some problems for the outer boundary. If the flow is external, then the outer boundary may be left rounded, and no issues result. However, for a turbomachine blade or vane passage, the understanding is that the grid must be *periodic* with respect to the θ-direction; that is, the points on the upper and lower boundaries in Fig. III.3 must match when the grid is rotated about the axial direction by an angle equal to the pitch. For this reason, the grid must have right angles on its outer boundaries, so as to interface properly with the next copy. These right angles create corners, and therefore slope discontinuities, in the outer boundary, thereby degrading the quality of the grid in the same manner as the H-grid. The C-grid is a compromise between the H- and

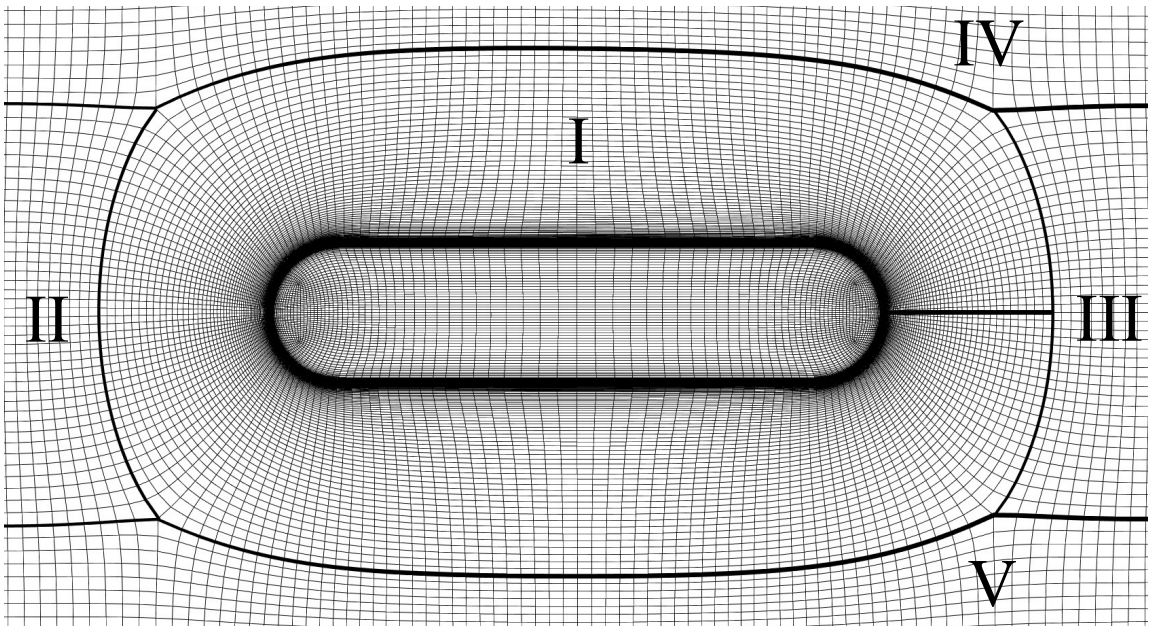

Figure III.4: O4H grid topology. Block I is an O-grid, and Blocks II-V are H-grids.

O-grids, in that the grid about the leading edge resembles an O-grid, and the grid around the trailing edge resembles an H-grid. It is therefore ideal for airfoils that have rounded leading edges and sharp trailing edges, as most airfoils nominally do. However, it retains the same problems with the outer boundary that the O-grid has.

For many turbomachinery applications, manufacturing concerns dictate that the edges of the airfoils must have some minimum radius, which is not negligable compared to the length scales of the fluid flow and hence cannot be idealized as a sharp edge. Thus the best grid topology would seem to be the O-grid. However, as discussed above, the right angles in the outermost grid lines tend to propagate far into the interior of the grid, and would likely interfere with the accuracy of the prediction of the blade-to-blade vortex that is known to appear in swirl brake flows [Nielsen

et al., 2001]. It follows that no simple grid topology is perfectly suited for these flows. A more modern approach is to combine several structured grid blocks in such a way that the burden of the non-ideal boundaries is shared between the blocks. This necessitates singularities in the mapping, where the grid quality is necessarily low, but results in higher overall grid quality. One possible choice is the so-called *O4H* grid topology, which is employed in this work. Figure III.4 shows a representative grid around a swirl brake. The grid for one vane is composed of four structured blocks. Block I is an O-grid which surrounds the vane. This grid is surrounded by four H-grid blocks, labeled II-V, which serve to provide a buffer between the O-grid and the harsh outer boundary. The singularities in the mapping occur at the "corners", of the O-grid block, where five edges meet at a single point. They are referred to as singularities because no one-to-one mapping from the $\xi - \eta$ domain to the $x - \theta$ domain exists, and a point surrounded by five neighbors does not permit a traditional finite difference representation of the derivatives. Hence, they must be either treated as boundary points, or different schemes must be used for the singular and non-singular points.

Two possibilities exist for the generation of such a multi-block grid. The first is to prescribe the block boundaries, perhaps using Bezier curves or some sort of spline, and generate the interior of the blocks using standard approaches. This approach, while simple and cheap, will likely result in awkward transitions at the block boundaries. Alternatively, one may attempt to smooth the entire grid at once, thereby eliminating the transition problems at the expense of complexity and computational cost. This is the approach used here. Each block is attacked with

a standard iterative smoothing algorithm. The overall grid is then smoothed in a manner analogous to Gauss-Seidel; that is, one block is smoothed while the others are held constant, and then that block is fixed while the next block is smoothed. The blocks are cycled in this manner until convergence is achieved. The result is a completely smooth grid with practically invisible block boundaries, as evident in Fig. III.4 .

III.2. Flow Solver

The RANS equations are solved using in-house CFD software called UNS3D, which stands for unsteady, unstructured Navier–Stokes in three dimensions. This software has been previously used to simulate turbomachinery flows [Carpenter, 2016; Kim, 2003; Brenner et al., 2013], cavity flows [Liliedahl et al., 2011], aeroelasticity problems [Gargoloff, 2007], and hypersonic flows [Brown, 2016].

III.2.1. Spatial Discretization

The spatial part of the system of partial differential equations must be discretized, resulting in a system of ordinary differential equations that govern the evolution of the flow variables through time. The standard methods of spatial discretization of PDE's include the finite element, finite volume and finite difference methods; in this work, the finite volume method is used. This requires that an integral formulation of the governing equations is used. The domain of interest is broken up into non-overlapping control volumes using a cell-vertex, median-dual scheme, which places the unknowns at the grid points of the original mesh. For each control

volume, the flux integral is approximated by a sum single-point quadratures for each face of the control volume, which is known to be sufficient for up to second-order accuracy [Blazek, 2005]. The viscous fluxes are evaluated using a central scheme. The inviscid fluxes are evaluated using Godunov's method [Godunov, 1959]. The resulting Riemann problem at each quadrature point is evaluated using Roe's scheme [Roe, 1986] with Harten's entropy fix [Harten, 1983].

The gradients of the flow variables at a grid point are evaluated with a least-squares fit using the immediate neighbors of that grid point. Since least-squares problems are known to be poorly conditioned, a QR decomposition is used for its stability properties [Haselbacher & Blazek, 2000]. A piecewise linear reconstruction over each control volume is used to obtain second-order accuracy [Barth & Jesperson, 1989]. Because of Godunov's accuracy barrier [Blazek, 2005], it is known that linear schemes of second and higher order produce nonphysical oscillations in regions of high gradients. Hence, nonlinear schemes must be used which locally reduce the order of accuracy in these regions, either by the addition of artificial dissipation, or by the reduction of the gradient estimates using limiters. A special modified version of Venkatakrishnan's limiter [Venkatakrishnan, 1995a] developed by Carpenter [2016] is used in this work.

The spatial discretization is parallelized using domain decomposition and the Message Passing Interface (MPI). Each processor receives a portion of the complete grid, as well as a thin buffer layer that overlaps with the subdomains allocated to neighboring processors. The original implementation due to Kim [2003] was restricted to grids composed of topologically identical layers. A refined implementation

due to Brown [2016] allowed arbitrary subdomain boundaries. The reader is referred to the dissertation by Brown [2016] for details regarding the parallelization algorithm.

This section is intended to introduce, at a surface level, the methods that make up the core of the flow solver. A more complete description may be found in the dissertations by Kim [2003] and Gargoloff [2007].

III.2.1.1. Semi-Discrete Form

The integral form of the governing equations may be written for a single control volume as

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}d\Omega + \oint_{\partial\Omega}\left(\mathbf{F}_{\mathrm{c}} - \mathbf{F}_{\mathrm{v}}\right)dS = \int_{\Omega}\mathbf{G}d\Omega.$$

For a constant or linear reconstruction, we have

$$\int_{\Omega}\mathbf{U}d\Omega = \mathbf{U}_{\mathrm{cen}(\Omega)}\Omega,$$

where $\mathbf{U}_{\mathrm{cen}(\Omega)}$ is the value of U at the centroid of the control volume. The grid point associated with a control volume will not, in general, lie at the center of that control volume in a median-dual scheme, resulting in a discrepancy between the storage location of the unknowns and the centroid. For steady flows, accurate time integration is not important, and this discrepancy may be neglected without any ill effects [Blazek, 2005]. For unsteady flows, a mass matrix is generally used to address the discrepancy. However, for the grids encountered in this work, the skewness is very low. Hence, the discrepancy between the storage location of the unknowns and the cell centroid is very small, and the mass matrix may be neglected without issue.

The flux through each face of the control volume is evaluated using a single

point quadrature. The total surface flux is then

$$\oint_{\partial\Omega} (\mathbf{F}_\mathrm{c} - \mathbf{F}_\mathrm{v})\, dS = \sum_{j=1}^{N_\mathrm{F}} (\mathbf{F}_\mathrm{c} - \mathbf{F}_\mathrm{v})_j\, S_j,$$

where N_F is the number of faces forming the boundary of the current control volume. This, taken with the assumption of the previous paragraph, leads to the following:

$$\frac{\partial \mathbf{U}}{\partial t} = -\sum_{j=1}^{N_\mathrm{F}} (\mathbf{F}_\mathrm{c} - \mathbf{F}_\mathrm{v})_j\, S_j + \mathbf{G} \equiv \mathbf{R}, \tag{3.1}$$

where $\mathbf{R}$ is the residual associated with the current control volume. This is known as the semi-discrete form, because the spatial operators have been discretized, but not the temporal operator. Taken for all the control volumes together, this represents a system of N ordinary differential equations, where $N = N_\mathrm{node} \times 5$ for inviscid or laminar flows, or $N_\mathrm{node} \times 7$ for turbulent flows.

III.2.1.2. Nonlinear Solution Strategies

For steady flows, the time derivative in Eq. 3.1 vanishes, and we arrive at a system of N nonlinear algebraic equations for the steady flow state. This system may be attacked directly using many standard methods, including Newton-type methods, Quasi-Newton methods, and multigrid methods. However, the extreme nonlinearity of these equations means that these fast methods are unlikely to succeed without an excellent starting guess, which is usually unavailable in the context of CFD simulations. In the case of the Newton and Quasi-Newton methods, this difficulty may be approached using standard globalization techniques, such as line searches and trust region approaches, but convergence from a bad starting guess is still unlikely.

The modern tool for dealing with the lack of a satisfactory starting guess is the class of methods called *continuation* methods. Broadly speaking, a continuation

methods forms a relationship between the problem at hand, which is presumably very difficult, and a simpler but related problem for which either an exact answer is known, or a satisfactory solution algorithm is available. The relationship between the two problems typically takes the form of an interpolation using a continuation parameter. For the simple problem, the continuation parameter is assigned a value (say, for example, 0), and the difficult problem is also assigned a value (say, 1). Some mechanism is then used to advance the continuation parameter from the simple problem to the difficult one. This description is intentionally vague, and there are numerous methods that fall under this umbrella term. Indeed, many tricks frequently employed by CFD practitioners may be interpreted as continuation methods. For example, one might start with a low-order scheme and advance toward a high-order scheme. Another example is starting with additional artificial dissipation, or starting with a coarse grid, before interpolating to subsequently finer grids (sometimes known as full multigrid). These techniques, though conceptually simple, have been by many CFD practitioners to successfully solve problems for which a direct attack using the aforementioned fast algorithms has failed.

One continuation method stands above the others in the context of fluid flow simulations, and that is the so-called *pseudo-transient continuation* method. In this case, the continuation parameter is time, and the solution of the "simple problem" is the solution at $t = 0$, which is the initial guess. The semi-discrete equations 3.1 are then integrated through time to a steady state, using any of a number of practical methods for ordinary differential equations. This approach is one of the oldest and most commonly used in CFD; indeed, practically every commercial solver for the

Navier–Stokes equations uses this approach as its core solution method. Its strength lies in its robustness; in the author's experience, if the solution procedure fails, it is almost always due to improper problem specification, and almost never due to the time-integration method, so long as one operates within the stability boundaries of the time-stepping method. The cost of this robustness, unfortunately, is an extended solution time. A reasonable approach, then, is to use the time stepping approach to obtain a good starting guess for the endgame solver, which is usually one of the aforementioned fast solvers.

Time stepping methods for Eq. 3.1 broadly fall into two categories: explicit and implicit. The explicit methods use a time discretization that may be trivially solved for the solution at the next time step. Hence, the computational work and storage needed in order to advance the solution to the next step are minimized. The drawback is that all explicit methods have stability problems that place restrictions on the maximum allowable time step, thereby driving up the total cost of the solution. The alternative is to use an implicit method, which removes the stability restriction. However, implicit methods require the solution of a system of linear or nonlinear equations at each time step. Hence, the cost per iteration is much larger than that of explicit methods, but the number of steps needed to obtain a solution is dramatically reduced. This tradeoff may situationally favor either class of methods, and the ideal solver should have access to methods from both classes.

The flow solver, UNS3D, uses a traditional four-stage Runge–Kutta scheme to integrate Eq. 3.1 to a steady state. Since the precise evolution of the solution through time is of no importance compared to the steady state result, convergence acceler-

ation techniques may be used to reduce the number of steps needed to arrive at an answer. Local time stepping [Jameson et al., 1981] and implicit residual smoothing [Jameson et al., 1986] are used in this work. The reader may consult the dissertation by Kim [2003] for a complete discussion of the explicit time stepping method.

As mentioned above, the stability restrictions of explicit methods often make them prohibitively slow for problems of practical interest. The next section introduces the basic implicit scheme that is intended to overcome this slow convergence.

III.2.1.3. Implicit Scheme

The time derivative in Eq. 3.1 may be evaluated using a backward Euler discretization, resulting in

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} = \mathbf{R}\left(\mathbf{U}^{n+1}\right), \tag{3.2}$$

where the superscript indicates the time step number. Since the solution at the next time level appears embedded in the right-hand side, this is an implicit method. The backward Euler method is known to have no stability restrictions. However, Eq. 3.2 represents a system of *nonlinear* algebraic equations that must be solved for the solution at the next time step. This system contains all of the same difficulties as the original steady state system, so it seems that we have gained little so far, although a good initial guess is available for Eq. 3.2 by applying an explicit method. In order to obtain a tractable problem, the right-hand side of Eq. 3.2 is expanded out to the first order term:

$$\mathbf{R}\left(\mathbf{U}^{n+1}\right) \approx \mathbf{R}\left(\mathbf{U}^n\right) + \mathbf{J}\left(\mathbf{U}^n\right)\left(\mathbf{U}^{n+1} - \mathbf{U}^n\right), \tag{3.3}$$

where $\mathbf{J}(\mathbf{U}^n)$ is the *Jacobian matrix* of the residual evaluated at time level n. Defining $\Delta\mathbf{U} = \mathbf{U}^{n+1} - \mathbf{U}^n$, the first-order term is moved over to the left-hand side of Eq. 3.2, resulting in

$$\left[\mathbf{J}^n - \frac{\mathbf{I}}{\Delta t}\right]\Delta\mathbf{U} = -\mathbf{R}^n. \tag{3.4}$$

This is the basic implicit method that is used in this work. The quantity in brackets is the *system matrix*, which is a sparse matrix of size N, which in this work will number up to, at most, a few hundred million. It is important to note that the accuracy of the final result is determined only by the accuracy of the right-hand side, while the convergence of the method is governed by the quality of the left-hand side. Hence, any number of approximations may be employed to make the system matrix cheaper to construct or solve.

Some insight may be gained by examining the form of the system matrix. In the limit of large time steps, the diagonal matrix vanishes, and we are left with none other than Newton's method. In the limit of small time steps, the diagonal matrix dominates, and we recover a forward Euler discretization. Hence, the present method represents a compromise between Newton's method and the forward Euler method, with the time step acting as a tuning parameter to control this compromise. The presence of the diagonal term has two effects. First, it reinforces the diagonal of the system matrix, thereby allowing easier solution with iterative methods. Second, by blending with a slower, simpler scheme, the likelihood of convergence is improved. This compromise resembles the trust-region globalization method for optimization, wherein the Newton step is blended with gradient descent [Dennis & Schnabel, 1996], to similar effect.

This implicit scheme (3.4) is quite standard in practical CFD computations, see for example [Blazek, 2005]. What distinguishes the various flow solvers is *how* this system is solved, which is discussed in the next section.

III.2.1.4. Linear Solution Strategies

The simplest approach for the solution of Eq. 3.4 is the use of direct methods, which almost always have their roots in Gaussian elimination. In this context, an LU decomposition would likely be employed. For dense matrices, an LU decomposition has $O(N^3)$ complexity, which would be entirely prohibitive in this context. For sparse matrices, the best-case complexity is $O(N_{\mathrm{NZ}})$, where N_{NZ} is the number of nonzero entries in the matrix. For a PDE discretization with a fixed stencil size, $N_{\mathrm{NZ}} \propto N$, so the cost of the LU decomposition is $O(N)$. However, the proportionality constant is large, and computational experience has shown that this approach is not feasible for three-dimensional problems [Vanden & Whitefield, 1995]. Furthermore, the author has some experience with standard packages such as *SuperLU* and *MUMPS*, and this experience indicates most direct methods for sparse linear systems do not parallelize well. Finally, all direct methods require the matrix to be built and stored in memory, which itself is likely prohibitive for three-dimensional problems with second-order or higher discretizations. Hence, iterative methods are practically mandatory for large-scale implicit CFD codes [Blazek, 2005, p. 193].

The modern tool for the iterative solution of large, sparse linear systems is the class of methods known as Krylov subspace methods [Saad, 2003]. These methods are generally faster than older stationary iterative methods, such as the Richardson,

Jacobi, Gauss–Seidel, and Successive-Over-Relaxation methods. In addition, they are generally more robust and carry theoretical guarantees that the older methods lacked, including upper bounds on the total number of steps. An final advantage of Krylov subspace methods for Newton-style iterations is that they only require matrix-vector products, and therefore may be performed *matrix-free*. For general sparse systems, the most popular methods are the Generalized Minimum Residual (GMRES) [Saad & Schultz, 1986] and the Stabilized Biconjugate-Gradient (BiCG-Stab) [Van der Vorst, 1992] methods. Because GMRES minimizes the linear residual over the Krylov subspace at each iteration, it guarantees a decrease in the residual norm with each iteration. The cost of GMRES scales quadratically, and the storage linearly, with the number of iterations performed, and thus a *restarted* version of the algorithm is often employed. However, the restarted algorithm does not carry the same convergence guarantees as the full algorithm [Saad & Schultz, 1986]. The BiCG-Stab algorithm alleviates the storage requirements of GMRES, but the convergence is often erratic, and thus one cannot guarantee that additional iterations will yield a better solution. It is for this reason that GMRES is usually preferred, and thus an *unrestarted* version of GMRES is used in this work, following the implementation in [Saad & Schultz, 1986].

A practical difficulty with iterative methods is that they usually need to be preconditioned to be effective. Indeed, it is often said that the choice of preconditioner is more important than the choice of Krylov subspace method [Saad, 2003]. Preconditioning refers to the transformation of a linear system to one with the same solution, but which is easier to solve. Preconditioning may be performed on the left,

i.e.

$$\mathbf{MAx} = \mathbf{Mb}$$

or on the right, i.e.

$$\left(\mathbf{AMM}^{-1}\right)\mathbf{x} = \mathbf{b} \tag{3.5}$$

where $\mathbf{M}$ is the preconditioning matrix. In the case of right preconditioning, the problem (3.5) is usually broken into two sub-problems

$$(\mathbf{AM})\,\mathbf{y} = \mathbf{b}$$

$$\mathbf{x} = \mathbf{My},$$

so that the inverse of M is never explicitly needed. It is also possible to employ "mixed" preconditioning, which is a combination of left and right preconditioning. While left and right preconditioning typically yield similar benefits [Saad, 2003], there exists a particular variant of GMRES which provides an estimate of the residual norm at each iteration. This variant of GMRES, which is used in this work, provides an estimate of the unpreconditioned norm of the residual for right preconditioning, but an estimate of the preconditioned residual norm in the case of left preconditioning. Since the preconditioner may have a dramatic effect on the residual norm, the preconditioned residual norm does not provide a fair way to compare the effectiveness of several preconditioning options. Hence, right preconditioning is often preferred for use with GMRES, and is used here for this reason. Right preconditioning also allows use of a so-called "flexible" algorithm [Saad, 1993] that permits the preconditioner to vary from one GMRES search direction to the next, and also saves

one preconditioner application, at the cost of doubling the storage requirements of the algorithm. Because the preconditioner did not vary from one GMRES search direction to the next, and because the preconditioners considered later in this work were memory intensive, the standard "inflexible" version of GMRES was used here.

As mentioned in the previous section, the system matrix need not be evaluated accurately to obtain a consistent method, because only the right-hand side determines the problem being solved. Therefore, so long as the right-hand side is evaluated accurately, and the method converges, any approximation of the system matrix may be used. Two common options are to use a low-order approximation of the system matrix, and to use a crude approximation of the viscous part of the spatial operator. Both of these approximations reduce the stencil size of the scheme, and therefore reduce the computational expense of building and solving the linear system. Indeed, a widely held opinion is that the construction of a second-order Jacobian is prohibitive, due to the excessive bandwidth. The penalty for using any of these approximations is that quadratic convergence can no longer be achieved. In most practical cases, quadratic convergence cannot be achieved regardless, so these reduced-order techniques can be used to create a tractable problem with some cost in convergence speed.

In the case of Newton-Krylov methods, where the Newton system is solved using a Krylov subspace technique, the possibility exists to obtain the benefits of the full second-order Jacobian without the cost of its construction. We speak here of *matrix-free* techniques. This possibility exists because Krylov subspace methods require only the action of the Jacobian on a vector, not the Jacobian itself. Since

the Jacobian is itself a derivative, its action on a vector may be interpreted as a directional derivative, and therefore approximated with a single finite difference:

$$\mathbf{J}\mathbf{v} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\mathbf{v} = \frac{\partial \mathbf{R}}{\partial v} = \lim_{\epsilon \to 0} \frac{\mathbf{R}(\mathbf{U} + \epsilon \mathbf{v}) - \mathbf{R}(\mathbf{U})}{\epsilon} \approx \frac{\mathbf{R}(\mathbf{U} + h\mathbf{v}) - \mathbf{R}(\mathbf{U})}{h},$$

where v is the distance traveled along the vector $\mathbf{v}$, and h is the finite difference step size. Hence, assuming the residual $\mathbf{R}(\mathbf{U})$ is already known, the action of the Jacobian on a vector may be approximated using a single residual evaluation. The choice of the differencing parameter h is not entirely trivial. If one has access to exact arithmetic, one would naturally set h as small as possible to reduce the truncation error of the finite difference approximation. However, in practice one must necessarily use finite-precision arithmetic, and roundoff error can easily spoil the solution. Hence, the differencing parameter must be carefully chosen to balance these two sources of error. A common recipe [Dennis & Schnabel, 1983] is to use the square root of machine zero, and this choice is used here. In the event that this prescription provides inadequate accuracy, another possibility is to use a central difference:

$$\mathbf{J}\mathbf{v} \approx \frac{\mathbf{R}(\mathbf{U} + h\mathbf{v}) - \mathbf{R}(\mathbf{U} - h\mathbf{v})}{2h}.$$

This will almost certainly alleviate the accuracy issue, at the expense of a second residual evaluation for each matrix-vector multiply, thereby doubling the cost of the algorithm. This should therefore only be used as a last resort. In the software implementation used here, both options are available, although the first-order difference was found sufficient for all cases in this work.

Using a matrix-free implementation, the cost of building and storing the Jacobian can be completely avoided, and hence the prohibitive bandwidth of the full

second-order Jacobian is no longer relevant. Quadratic convergence may therefore be pursued, although in many cases a matrix-free approach using first-order Jacobian has proven beneficial, and both options are available in the software. However, even when a matrix-free implementation is used, a preconditioner is still required. While matrix-free preconditioning strategies exist [Saad, 1993] that offer enormous savings in memory, they generally lag behind matrix-ready preconditioners in terms of wall-clock time. Therefore, matrix ready preconditioners are used in this work. Several options are available in terms of the fidelity used in the primary Jacobian and its preconditioner. Here we use the same terminology as [Cai et al., 1995]. The first possibility is to use a high fidelity Jacobian evaluation together with a high fidelity preconditioner:

$$\mathbf{M}_{\text{high}}\mathbf{J}_{\text{high}}\boldsymbol{\Delta}\mathbf{U} = \mathbf{M}_{\text{high}}\mathbf{U}, \tag{3.6}$$

where the subscripts "high" and "low" indicate the use of a high- or low-fidelity approximation. Left preconditioning is used in Eq. 3.6 for simplicity of presentation, but the same concepts apply to right preconditioning. This "high-high" approach has tremendous potential to offer quadratic convergence for realistic problems, but is generally considered infeasible for matrix-ready preconditioners because, as mentioned before, a full-order Jacobian matrix is prohibitively expensive to evaluate for three-dimensional problems. The second approach is to use a low-fidelity preconditioner together with a high-fidelity Jacobian:

$$\mathbf{M}_{\text{low}}\mathbf{J}_{\text{high}}\boldsymbol{\Delta}\mathbf{U} = \mathbf{M}_{\text{low}}\mathbf{U}.$$

This approach allows the pursuit of quadratic convergence with modest cost, and is used in most of this work. The third possibility is to use a low-fidelity preconditioner

together with a high-fidelity Jacobian:

$$\mathbf{M}_{\text{low}}\mathbf{J}_{\text{low}}\boldsymbol{\Delta}\mathbf{U} = \mathbf{M}_{\text{low}}\mathbf{U}.$$

This approach allows for the lowest possible cost. The issue that arises here is whether the low-fidelity Jacobian is "close enough" to the high-fidelity Jacobian that the computed step direction sufficiently approximates the true Newton step. For simple flow problems, the low-fidelity Jacobian is sufficient. For harder problems, the critical factor is the level of mesh resolution. As the mesh is refined, the first-order and second-order discretizations become closer, and one can expect a low-fidelity system to better approximate the true Newton step. However, in practical CFD the problem is often under-resolved, and hence it is possible that the low-fidelity Newton step will not provide a descent direction for the second-order residual. On the other hand, because the mismatch between the fidelity level of the preconditioner and Jacobian is eliminated, the preconditioner is more effective for this approach than for the "low-high" approach above. In this work, the "low-low" approach is used whenever possible, but when in doubt, the "low-high" approach is used. Of course, one could consider a fourth option, which is to use a high-fidelity preconditioner with a low-fidelity Jacobian:

$$\mathbf{M}_{\text{high}}\mathbf{J}_{\text{low}}\boldsymbol{\Delta}\mathbf{U} = \mathbf{M}_{\text{high}}\mathbf{U}.$$

This approach, however, is nonsensical, as the preconditioner will be both harder to construct and less effective at preconditioning the low-fidelity Jacobian than the low-fidelity preconditioner. Hence, this "high-low" approach is never used.

To summarize, in this work a low-fidelity preconditioner is always used, which

is build from a first-order discretization of the nonlinear residual, while the Jacobian sometimes uses a first-order discretization, and sometimes uses a second-order discretization. The viscous terms are never neglected, either in the Jacobian or the preconditioner, because it was found that they are essential for both convergence and effective preconditioning. For small nonlinear step sizes, the first-order representation of the Jacobian is usually tried first, and if convergence stalls, a second-order representation is used. For large nonlinear step sizes, the first-order representation is unlikely to work, and the second-order representation is used. The next section discusses the details of the preconditioning algorithm built from the first-order discretization.

III.2.1.5. Preconditioning Strategies

The most successful general purpose preconditioner for unsymmetric matrices is undoubtedly the Incomplete Lower-Upper (ILU) factorization, in which some or all of the *fill-in* encountered during an LU factorization is discarded. In the most extreme case, all fill-in is discarded, resulting in the ILU(0) algorithm. This preconditioner has the benefit of requiring $O(N)$ operations to build and apply, and requires no storage beyond that required for the matrix itself. However, while this approach has found enormous success, there are few theoretical guarantees of its success, and it sometimes fails spectacularly. For this reason, ILU factorizations using larger amounts of fill-in may be employed to better approximate the true factorization; however, the storage and computational cost increases as well.

A central problem with ILU factorizations is that the solution of sparse trian-

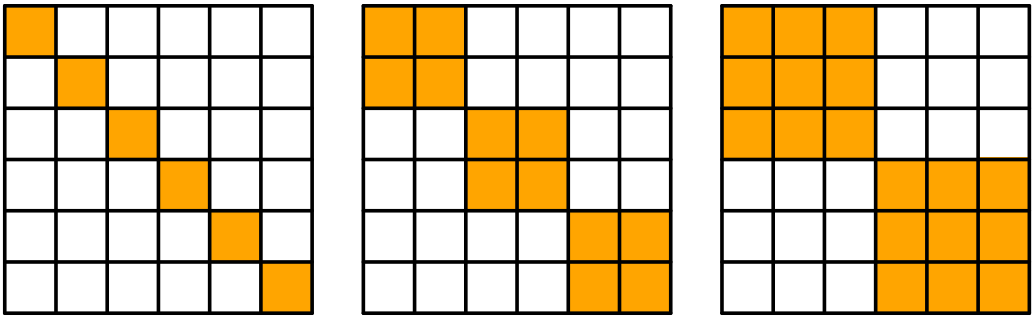

Figure III.5: Illustration of the Additive-Schwarz decomposition.

gular systems through forward- and back-substitution has an inherently *sequential* component; that is, one processor must complete its work before the next processor can begin. While approaches have been developed to mitigate this effect, the sequential bottleneck created by sparse triangular solves remains a problem in the application of ILU preconditioning.

One possibility to alleviate this sequential bottleneck is the use of *Additive-Schwarz* (AS) preconditioning [Dryja & Widlund, 1989]. Broadly, a Schwarz-type method breaks the problem into many sub-problems which, when solved one-by-one, form an approximation of the solution of the full problem. The Additive-Schwarz methods, in particular, allow the sub-problems to be solved completely independently. This differs from the Multiplicative-Schwarz methods, which require the sub-problems to be solved sequentially. In this sense, the additive methods are analogous to the Jacobi method, and the multiplicative methods are analogous to the Gauss-Seidel method. For this reason, AS methods are sometimes referred to as "Block-Jacobi" methods. This is illustrated in Fig. III.5. The white 6×6 box represents a matrix, while the yellow portion represents the portion of that matrix used to

construct the preconditioner. In the left image, the preconditioner is constructed using only the diagonal of the matrix. While a diagonal matrix is very cheap to invert, it represents a very small portion of the true matrix, and no coupling is captured in the preconditioner. Moving to the middle image, 2×2 diagonal blocks are chosen. This is still cheap to invert, since 2×2 matrices have simple explicit solutions, but more coupling is captured, and the preconditioner represents a larger portion of the original matrix. One can continue this approach with larger and larger blocks, leading to better and better performance, until one reaches a size where inversion of the block is no longer practical. While AS methods have very limited use as solvers, they are known to perform excellently as preconditioners. This is particularly true in a parallel computing environment, where the sub-problems can be formed from the sub-domains allocated to each processor. In this case, the preconditioner can be built and applied without any communication between the processors, resulting in a perfectly parallel preconditioner. The linear systems that arise on each processor are also sparse, and thus may be attacked with a sparse direct solver. In the event that this is too costly, an ILU factorization may be performed on each processor, thereby circumventing the aforementioned sequential bottleneck that plagues global ILU preconditioners.

The most glaring difficulty associated with parallel AS preconditioners is that as the processor count rises, the diagonal blocks constituting the preconditioner shrink relative to the size of the overall problem, and the effectiveness of the preconditioner is diminished. For a self-adjoint elliptic problem, an AS preconditioner (with exact subdomain solves) is known to reduce the condition number of the global problem

from $O(h^{-2})$ to $O(h^{-1}H^{-1})$, where H is the subdomain size [Keyes, 1995]. However, as the number of processors used for a given problem increases (in the sense of *strong* scaling), H approaches h, and the condition number approaches $O(h^{-2})$. Two methods exist to overcome this difficulty. The first is to include overlap the subdomain problems by small number of grid layers. With generous overlap, the condition number may be reduced to $O(H^{-2})$ [Keyes, 1995]. However, in practice, as one approaches larger and larger problems, one increases the number of processors used in proportion (in the sense of *weak* scaling). Therefore, $H \propto h$, and the condition number effectively still scales as $O(h^{-2})$. The other approach to overcoming this difficulty is the addition of a coarse-level component to the preconditioner, in the same spirit as multigrid methods. This, together with generous overlap, reduces the condition number to $O(1)$. Typically only one coarse grid is used, and experience indicates that approximately one coarse-grid point for each processor is optimal [Keyes, 1995]. While a coarse grid of that size is far too coarse to provide any benefit with a traditional multigrid method, here it serves not to eliminate the low-frequency components of the error, but merely to provide some global coupling between the sub-problems.

In this work, a non-overlapping AS method is supplemented by a coarse problem formed using the Additive Correction Multigrid (ACM) method [Hutchinson & Raithby, 1986]. This method is essentially a traditional linear multigrid method using piecewise constant restriction and prolongation operators [Gjesdal, 1996]. One immediate concern is whether such crude operators are sufficient to provide any benefit. Indeed, a known result in the multigrid literature is the so-called order rule [Gjesdal,

1996], which states that the sum of the orders of the restriction and prolongation operators must exceed the order of the differential equation if true multigrid performance is to be obtained. Since the Navier–Stokes equations are second-order PDE's, it follows that at least one of the interpolation operators must be piecewise linear in order to obtain textbook multigrid efficiency (TME); piecewise-constant operators should be sufficient for the Euler equations. However, it must be stressed here that the goal is *not* good multigrid performance, but simply a cheap way to add coupling between the subdomain problems. Hence, it reasonable to hope that while piecewise-constant interpolation operators may be too crude for a useful multigrid method for the Navier–Stokes equations, they may be sufficient to overcome the non-optimal scaling of the AS preconditioner.

One core difficulty with this approach is that in an AS preconditioning context, the coarse grid problem is extremely inconvenient to construct. In generating the diagonal blocks for the AS preconditioner, information about neighboring subdomains is intentionally avoided, because doing so results in additional communication among the processing elements as well as coding complexity, and the information is not needed in a pure AS context. Hence, in the event that a developer has already invested considerable time and money into a well-optimized and verified AS preconditioner, it is unlikely they will be willing to redo some of that work to create a coarse grid component. Further, this approach has been tried in a parallel, unstructured grid environment by Venkatakrishnan [1995b], who found that the gains obtained by using a coarse-grid component based on ACM were completely defeated by the additional cost of generating and applying the coarse-grid component, and therefore

did not justify the added complexity.

III.2.1.6. A Novel Preconditioner

A possible way to circumvent the painful generation of the coarse-grid problem is to attack it with a class of methods known as *Quasi-Newton* methods. These methods seek to obtain approximations for the Jacobian (or its inverse) by using readily available function evaluations. The oldest and most successful of these methods is Broyden's method [Broyden, 1969]. The basic idea is to update an estimate for the Jacobian by requiring that it satisfies a secant condition

$$\mathbf{As} = \mathbf{y},$$

where $\mathbf{s} = \Delta x$ is some incremental change in the unknowns, and $\mathbf{y} = \Delta f$ is some incremental change in the function f to be zeroed (in our case, this would be the residual vector). In the method Broyden originally envisioned, the incremental changes in x and f would come from the difference between the current and previous iterate; however, in practice, any function evaluations that are available may be used. In one dimension, the secant condition is enough to uniquely determine the new estimate for the Jacobian, but in multiple dimensions, it effectively provides one column of information about the Jacobian, and therefore the update is not unique. The various Quasi-Newton methods are distinguished by how they uniquely determine an update that satisfies the secant condition. Broyden himself conceived two methods, which history has given the unfortunate names of the "good Broyden" and "bad Broyden" methods. The "good" Broyden update is determined by requiring the least possible

change in the estimate of the Jacobian, and may be written as

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \frac{\mathbf{y}_k - \mathbf{A}_k \mathbf{s}_k}{||\mathbf{s}_k||^2} \mathbf{s}_k^T.$$

However, since the update is a rank-one perturbation, the Sherman-Morrison formula may be used to directly update the estimate of the inverse:

$$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1} + \frac{\mathbf{s}_k - \mathbf{A}_k^{-1} \mathbf{y}_k}{\mathbf{s}_k^T \mathbf{A}_k^{-1} \mathbf{y}_k} \mathbf{s}_k^T \mathbf{A}_k^{-1}.$$

This will circumvent the need to factor the matrix at every iteration. The "bad" Broyden method instead seeks the least possible change in the estimate of the *inverse* of the Jacobian, and may be written as

$$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1} + \frac{\mathbf{s}_k - \mathbf{A}_k^{-1} \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{y}_k} \mathbf{y}_k^T.$$

The "bad" method was given this name because Broyden himself found it did not converge well [Broyden, 1969]. However, later researchers have found that this may have been due to a mistake, and the "bad" method performs about as well as the good method on average [Kvaalen, 1991]. Which method is superior seems to be problem dependent, and in fact, methods have been devised that switch between the "good" and "bad" methods during the computation, using a simple test to indicate which will likely perform better.

In this work the Broyden update is applied to the coarse-grid problem, using any fine grid function evaluations that are available. In this way, the coarse-grid problem never has to be explicitly formed. Further, the fine-grid function evaluations are taken directly from the GMRES iterations from previous time steps, so no further function evaluations are necessary to estimate the inverse of the coarse grid Jacobian.

To firmly describe the new preconditioner, let us consider the following problem:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.7}$$

where $\mathbf{A}$ is a real $N \times N$ matrix, and $\mathbf{x}$ and $\mathbf{b}$ are real vectors of size N. Consider further that the vector $\mathbf{b}$ is a function to be zeroed, and $\mathbf{A}$ is its Jacobian, possibly with a modification such as a diagonal component due to an implicit time-stepping method. Equation 3.7 is then an approximate Newton method. The coarse grid problem is formed using a Galerkin approach. The *prolongation* operator transforms a vector represented on the coarse grid to the same vector represented on the fine grid:

$$\mathbf{P}\tilde{\mathbf{v}} = \mathbf{v}, \tag{3.8}$$

where $\mathbf{P}$ is the prolongation operator, $\mathbf{v}$ is a vector of size N represented on the fine grid, and $\tilde{\mathbf{v}}$ is a vector of size $\tilde{N}$ that represents $\mathbf{v}$ on the coarse grid. The prolongation operator used here is linear, and therefore may be interpreted as an $N \times \tilde{N}$ matrix. Substituting Eq. 3.8 into Eq. 3.7, we arrive at

$$\mathbf{A}\mathbf{P}\tilde{\mathbf{x}} = \mathbf{b}. \tag{3.9}$$

The matrix $\mathbf{AP}$ is an $N \times \tilde{N}$ matrix, and hence the problem (3.9) is overdetermined. In order to arrive at a uniquely solvable problem, the problem (3.9) is restricted onto the coarse grid, resulting in

$$\mathbf{R}\mathbf{A}\mathbf{P}\tilde{\mathbf{x}} = \mathbf{R}\mathbf{b}, \tag{3.10}$$

where the $\tilde{N} \times N$ matrix $\mathbf{R}$ represents the restriction operator. The $\tilde{N} \times \tilde{N}$ matrix $\tilde{\mathbf{A}} \equiv \mathbf{RAP}$ now represents the left-hand side of the coarse-grid problem, which is uniquely solvable so long as the matrix $\tilde{\mathbf{A}}$ is nonsingular.

Assuming the problem (3.10) is solvable, the coarse-grid solution may be analytically represented as

$$\tilde{\mathbf{x}} = (\mathbf{RAP})^{-1}\,\mathbf{Rb},$$

and this result is prolongated back to the fine grid:

$$\mathbf{x} = \mathbf{P}\tilde{\mathbf{x}} = \mathbf{P}\,(\mathbf{RAP})^{-1}\,\mathbf{Rb}. \tag{3.11}$$

It follows that the matrix

$$\mathbf{M}_{\mathrm{ACM}} = \mathbf{P}\,(\mathbf{RAP})^{-1}\,\mathbf{R} \tag{3.12}$$

represents an approximate inverse of the system matrix $\mathbf{A}$. This is combined with the AS component of the preconditioner to form the overall preconditioner:

$$\mathbf{M} = \mathbf{M}_{\mathrm{AS}} + \alpha\mathbf{M}_{\mathrm{ACM}}, \tag{3.13}$$

where α is a weighting factor. For this work, it was found that values of α between 0.1 and 1 worked well. Of course, in practice, one would not typically invert the the matrix $\mathbf{RAP}$, but perform an LU factorization. However, in this work, the inverse of this matrix will be directly approximated using a Quasi-Newton method.

In the ACM method, the restriction operator typically taken as a summation over all the control volumes that are to form a single coarse-grid control volume, and the prolongation operator is taken to be injection. In this manner, the restriction matrix is the transpose of the prolongation matrix. However, this leads to the awkward situation that restricted matrix entries are much larger than the fine-grid matrix entries. This is problematic because the final matrix is a weighted average of

the ACM component and the AS component, and using summation for the restriction operator will result in the ACM component overwhelming the AS component. This is completely contrary to the spirit of the coarse-grid operator, as it is only intended to supplement the AS preconditioner. Hence, the restriction operator is instead chosen to be an *average*, resulting in coarse-grid matrix entries of the same order of magnitude as the fine-grid matrix.

As mentioned above, the matrix $\mathbf{M}_{\text{ACM}}$ may be inconvenient to generate in a parallel context, as it requires not only an update of the AS component, but also significantly more communication than the AS component normally would. The generation of $\mathbf{M}_{\text{ACM}}$ may be circumvented by the use of a Quasi-Newton method. At every step of the GMRES solver, a matrix-vector multiply is performed, giving us new information about the matrix $\mathbf{A}$:

$$\mathbf{A}\mathbf{s} = \mathbf{y}. \tag{3.14}$$

The coarse-grid vector $\tilde{\mathbf{s}}$ that, when prolongated, forms the vector $\mathbf{s}$ is given implicitly by

$$\mathbf{P}\tilde{\mathbf{s}} = \mathbf{s}. \tag{3.15}$$

Taking the restriction of Eq. 3.15, we find:

$$\mathbf{R}\mathbf{P}\tilde{\mathbf{s}} = \mathbf{R}\mathbf{s}. \tag{3.16}$$

With the special choice of the restriction operator as an average, the matrix $\mathbf{RP}$ becomes the identity matrix, and the vector $\tilde{\mathbf{s}}$ is isolated:

$$\tilde{\mathbf{s}} = \mathbf{R}\mathbf{s}. \tag{3.17}$$

The reader should note two points. First, if the restriction had been instead chosen to be a summation, as in a typical ACM method, Eq. 3.17 would have had an additional diagonal matrix on the right-hand side. Second, while the matrix **RP** becomes the identity matrix with this special choice for the restriction, the product **PR** is **not** an identity matrix. Hence, it is somewhat fortunate that Eq. 3.17 called for the restriction of the prolongation, and not the prolongation of the restriction!

Now that we have isolated the vector $\tilde{\mathbf{s}}$, we can substitute its definition into the secant condition (3.14):

$$\mathbf{AP}\tilde{\mathbf{s}} = \mathbf{y}. \tag{3.18}$$

Taking the restriction, we find

$$\mathbf{RAP}\tilde{\mathbf{s}} = \mathbf{Ry} \equiv \tilde{\mathbf{y}}, \tag{3.19}$$

which is exactly a secant condition for the coarse-grid matrix **RAP**. We have now proven that, given a pair of vectors $\mathbf{s}$ and $\mathbf{y}$ for which the secant condition holds (3.14), these vectors may be restricted to obtain a pair of vectors $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{y}}$ which satisfy a secant condition for the coarse-grid matrix. While this is a straightforward and intuitive result, it bears repeating that for a typical ACM restriction operator, the math would have been slightly less clean.

The entire preconditioner may now be fully understood by the reader. An AS preconditioner is first formed by sparse finite differences on each processor. This component of the preconditioner is frozen for a number of iterations specified by the user. At each nonlinear iteration, the GMRES algorithm is used in conjunction with the AS preconditioner. At each iteration of the GMRES algorithm, a matrix-vector

product is formed in a matrix-free fashion using an nonlinear residual evaluation. The vector and its Jacobian-vector product are saved, and after the GMRES algorithm terminates, this pair of vectors is used to enrich the current approximation of the inverse of the coarse-grid matrix. After a number of nonlinear iterations specified by the user, the ACM component is added to the AS component of the preconditioner, thereby improving the global coupling of the AS preconditioner for all nonlinear iterations thereafter. The ACM component of the preconditioner is not frozen, but is continually enriched at each nonlinear iteration until the simulation terminates. This new preconditioner is tentatively referred to as the "AS+BROY" preconditioner.

A few small notes are in order. First, since the coarse-grid component of the preconditioner is no longer tied to the AS component, as would be the case if the coarse-grid problem were directly formed in the usual Galerkin sense, the coarse-grid component is not lost when the AS component is refreshed, and may be improved without interruption. Second, it was found experimentally that the "bad" Broyden method consistently outperforms its "good" counterpart when used in this context. The author can provide no rigorous explanation for this, but intuitively, it seems reasonable that a least-change update to the inverse of the Jacobian might prove better than a least-change update to the Jacobian itself in a preconditioning context, because the preconditioner is intended to approximate the inverse of the Jacobian.

CHAPTER IV

VERIFICATION AND VALIDATION RESULTS

This chapter presents the results of several test cases intended to show that the flow solver is able to capture the relevant physics of annular gas seal flows. The baseline solver has been previously verified and validated extensively for both simple canonical flows and flows of practical interest; see the dissertations by Carpenter [2016], Kim [2003] and [Brown, 2016] for an assortment. Thus, we focus here on verifying the present modifications to the solver and the validation of the ability of the solver to predict rotordynamic coefficients for annular gas seals. The modifications to the basic solver are as follows: 1) the option was added to solve the discretized governing equations and boundary conditions using GMRES with an Additive-Schwarz preconditioner, and 2) the wall rotation rate and frame rotation rate were decoupled to facilitate whirling seal flows. For item 1, it is sufficient to show that the new implicit solver produces the same result as the original solver, and does so in less time. Item 2 is trivial in it's implementation. While UNS3D has been shown to work properly for internal flows, the ability of the baseline solver to correctly predict rotordynamic forces has not been adequately documented. For this it is necessary to compare the computed solution against experimental results for several seal flows of practical interest.

Section 1 shows a comparison of the implicit and explicit options for several small canonical cases. Section 2 presents validation results for multiple seal flows.

IV.1. Verification of Implicit Solver

IV.1.1. Verification of Superlinear Convergence

One way to show that an implicit solver is working properly is to demonstrate that, in the limit of infinite time step, the solver behaves as a Newton solver and quadratic convergence is achieved. In this case, true quadratic convergence is unlikely to be achieved for two reasons. First, only first-order differences are used in computing Jacobian-vector products. Second, double-precision arithmetic is unlikely to be sufficient, as by the time that the algorithm has settled into quadratic convergence, machine error has already been reached. Thus, in this case, we will declare victory if 1) superlinear convergence is demonstrated, and 2) a decrease in the 2-norm of the residual of at least five orders of magnitude can be achieved in a handful of iterations. Superlinear convergence may be defined as

$$\lim_{k\to\infty} \frac{\epsilon_{k+1}}{\epsilon_k} = 0,$$

where ϵ is some measure of the error, and k is the iteration count. On a plot of the logarithm of the ϵ versus iteration count, this manifests as a downward bend in the curve. If the problem is sufficiently well behaved, we can expect to see the same behavior in the residual, and we can thus use the residual as a proxy for the error.

It is worth reiterating that superlinear convergence is not expected in practice, as we will usually use a modest CFL number to take advantage of the continuation properties of the implicit algorithm. Superlinear convergence is only worth checking on very small cases, for which good initial guesses are available and which may be well-solved by the iterative linear solver (GMRES).

IV.1.2. Evaluation of Preconditioner Effectiveness

IV.1.2.1. Additive-Schwarz Preconditioner

The baseline AS preconditioner is evaluated on a laminar channel flow at a Mach number of 0.2 and atmospheric conditions. The flow domain is pictured in Fig. IV.1. The flow enters through the $-x$ face and exits through the $+x$ face. The sides of the domain are all solid walls, so four boundary layers are present in the flow. The grid dimensions were 20 points in the streamwise direction, and 100 points in each of the transverse directions, which led to a total of 200,000 grid points. The initial spacing off the wall was chosen to correspond to a y^+ number of approximately unity. The domain was decomposed among 28 processors. To initialize the flow, the residual was driven down to a tolerance of 10^{-6} using the standard four-stage Runge–Kutta algorithm. The solution was then driven to a tolerance of 10^{-11} using both the Runge–Kutta scheme and the implicit solver. The maximum number of search directions used by the implicit solver was 20, and a reduction of one order of magnitude was requested at each time step. The CFL number used by the explicit solver was 2.0, which appeared to be approximately the stability limit for this problem. The CFL number used by the implicit solver was set to be practically infinite, so that the implicit scheme approximates a Newton solver.

The results of the two simulations are plotted in Figs. IV.2 and IV.3. The invisible implicit curve in Fig. IV.2 is not a mistake; the simulation terminates in approximately 250 iterations, and thus is within the border of the plot. Of course, this is not truly a fair comparison, because the implicit solver will take far longer per iteration than the explicit solver. A more fair comparison is in terms of wallclock

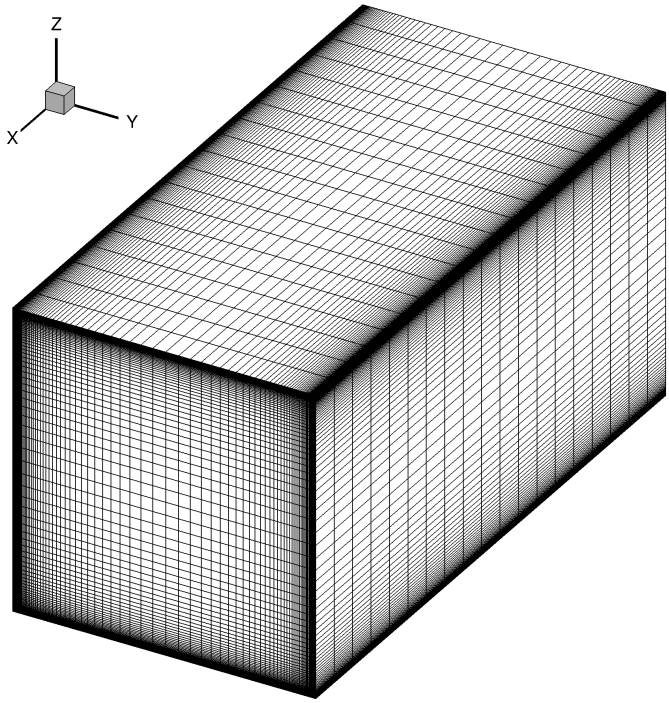

Figure IV.1: Computational grid for the laminar channel flow test case.

time, as shown in Fig. IV.3. Here, we see an improvement of approximately a factor of 40 in total simulation time.

IV.1.2.2. New Preconditioner

The new preconditioner is evaluated on the same laminar channel flow as the previous section, but an extremely coarse grid (approximately 500 grid points) is used to facilitate rapid testing. To determine the effectiveness of the new preconditioner in reducing the number of GMRES search directions, the solution is advanced from a constant initial guess for 100 nonlinear iterations using the basic AS preconditioner. During this time, the Broyden component of the preconditioner is simply "collecting

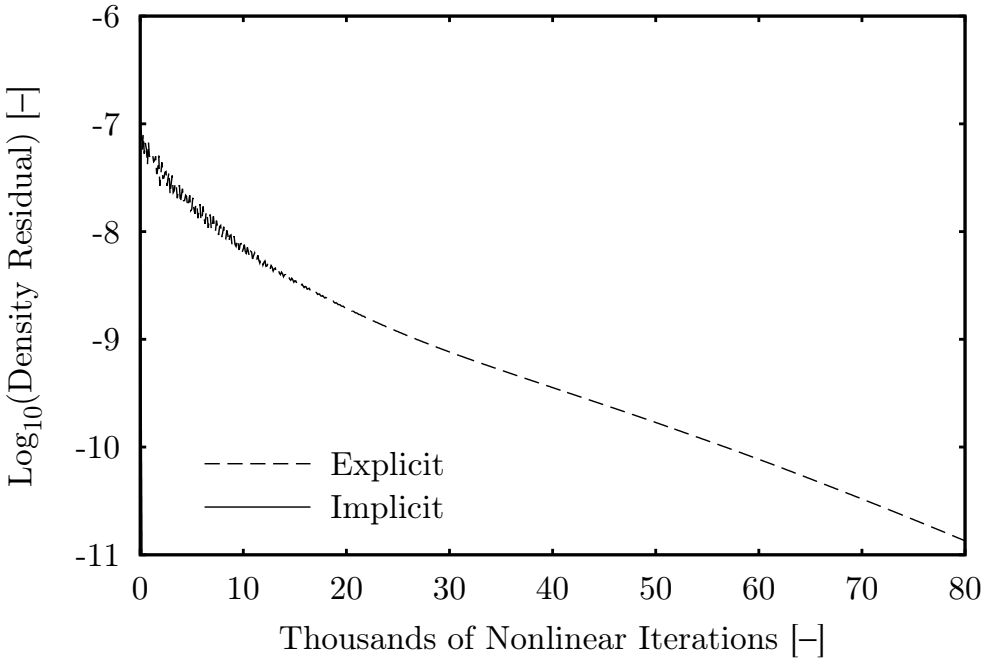

Figure IV.2: Convergence rate of the laminar channel flow test case.

data"; that is, the Broyden component is being enriched by every residual evaluation collected during the GMRES process, but is not contributing to the effectiveness of the preconditioner. This is done to ensure that the Broyden component of the preconditioner is sufficiently developed from its starting guess (the identity matrix) so as to not have a detrimental effect on the convergence of the linear solver. After 100 nonlinear iterations, the Broyden component is added to the preconditioner, and the solution is driven to a nonlinear residual tolerance of 10^{-11}. The CFL number was set to 100 for both simulations, and the maximum allowed number of search directions was set to 50, although neither solver hit this limit. A reduction in the linear residual norm of two orders of magnitude was required before advancing to the next step. The weighting factor assigned to the Broyden component was 0.2. The problem

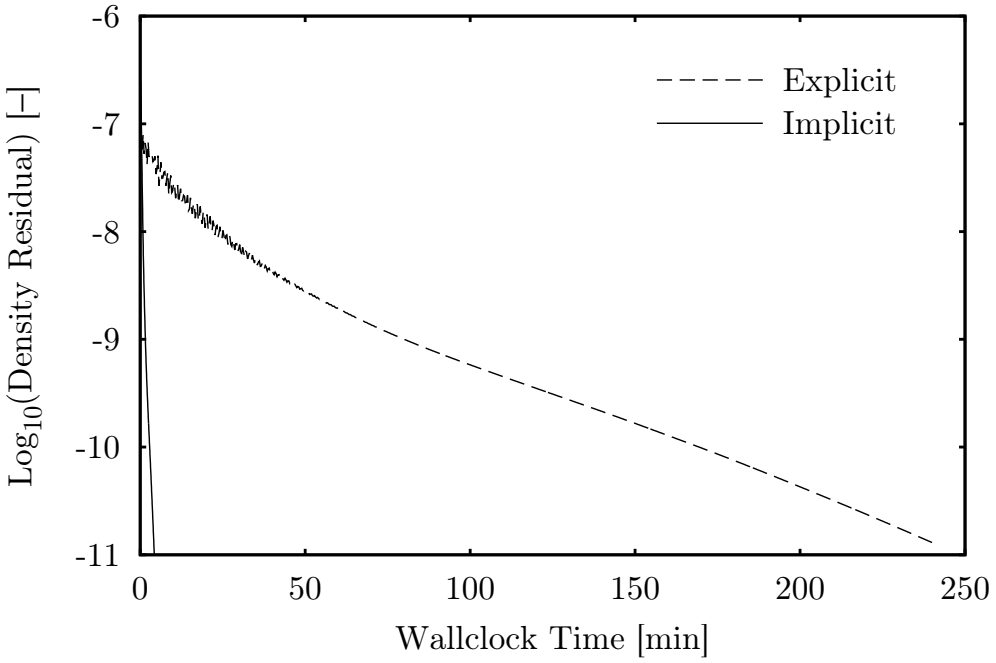

Figure IV.3: Wallclock time comparison for the laminar channel flow test case.

was decomposed among 28 processors, which is a much larger number than would normally be used for such a small problem. However, this is precisely the situation for which the coarse-grid component is needed; only when the number of processors is so large that the effectiveness of the basic AS preconditioner is diminished is the coarse-grid component necessary.

The convergence of the GMRES algorithm for both preconditioners is shown in Fig. IV.4. The basic AS preconditioner achieves a reduction in the linear residual of two orders of magnitude in 40 iterations, while the new preconditioner requires only 28, an improvement of approximately 41%. It is interesting to note that the difference in the two curves appears to widen as the iterative solver advances, indicating that the best benefits from the new preconditioner may be obtained when very deep

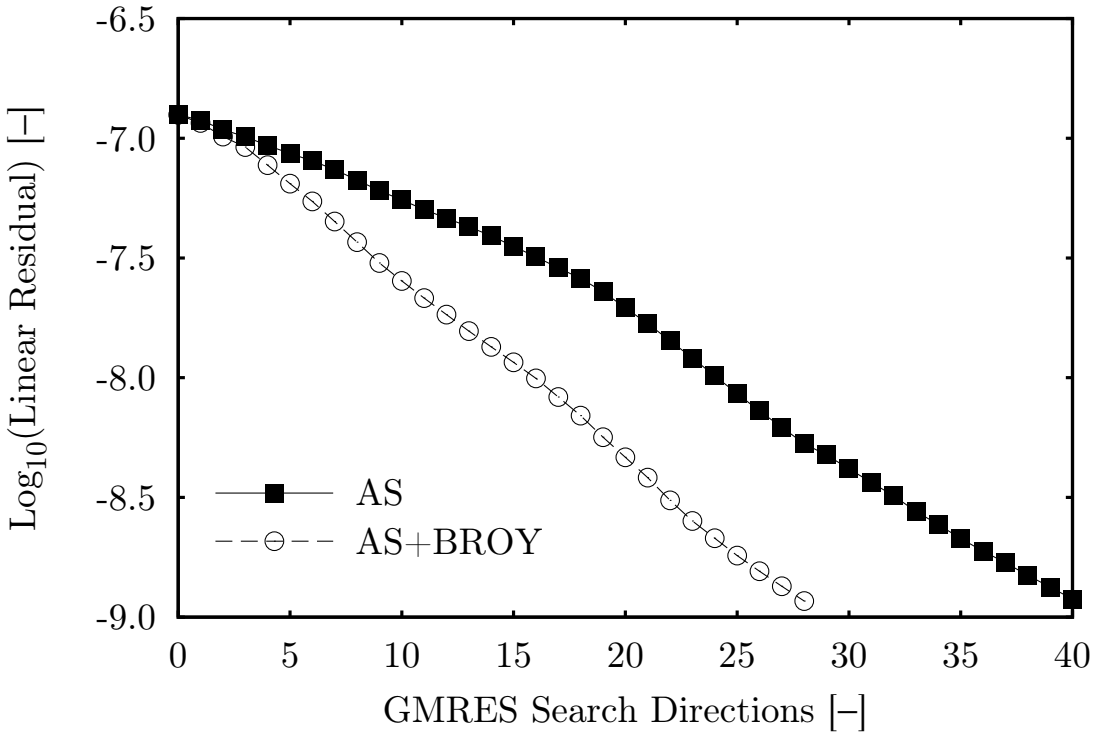

Figure IV.4: Convergence of the GMRES algorithm for the 101^{st} iteration of the laminar channel problem.

convergence is required from the linear solver. This would be the case when the the outer nonlinear problem is very stiff.

The number of GMRES search directions required at each nonlinear iteration of the laminar channel problem is plotted in Fig. IV.5. The "stairstep" pattern is expected, as the number of iterations is necessarily quantized to integer values. The reader will notice a sharp initial increase in the number of search directions required. This is due to the fact that the AS component of the preconditioner is "frozen" for the duration of this simulation; hence, the effectiveness of this component will be

very strong shortly after it is computed, but will decay rapidly thereafter. For longer simulations, the AS component of the preconditioner is updated every few hundred iterations. For the first 100 iterations, no difference exists between the performance of the two preconditioners, because the coarse grid component is only being enriched, and is not contributing to the linear system solution. After 100 iterations, the basic AS preconditioner requires a flat 40 search directions to achieve a reduction in the linear residual of two orders of magnitude, while the new preconditioner requires first 28 for approximately 35 iterations, and 27 thereafter. The reduction from 28 to 27 could be completely coincidental, but is in line with the expected behavior of the new preconditioner, because the Quasi-Newton method should be gradually improving the coarse-grid component of the preconditioner throughout the simulation. It is interesting that the new preconditioner seems to approach the effectiveness of the "fresh" AS preconditioner shortly after its generation.

Next, we test the new preconditioner on the Wright seal, using a very coarse grid (approximately 200,000 nodes.) The geometry and flow conditions of this problem will be discussed in detail later in this section. Here, the problem is restricted to laminar flow. Similar to the above flat plate test case, the flow is converged using a first order explicit method, then driven down 1 order of magnitude using a second order explicit method. From there, 100 iterations are performed using the implicit method, and on the 101st iteration, the preconditoner is switched on. Shown in Fig. IV.6 is the number of linear iterations required for the 101st step. We again see a substantial improvement in the number of linear steps required to obtain a given tolerance.

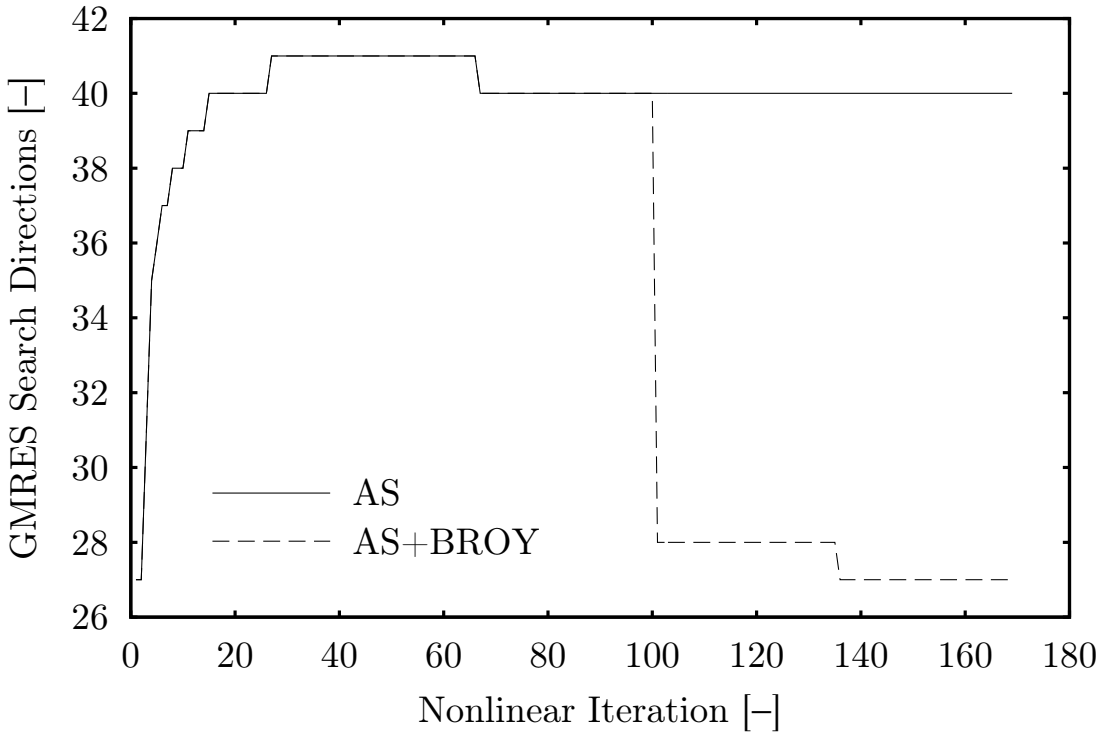

Figure IV.5: Improvement in the required number of search directions required fr the laminar channel problem.

IV.2. Validation of Solver for Annular Gas Seal Flows

IV.2.1. Nelson Seal

The simplest possible geometry for an annular gas seal is a smooth seal, similar in concept to journal bearings. In this case, no teeth, grooves, or pockets are present, and the sealing effect is entirely due to the viscous effects in the seal clearance. The geometry and flow conditions come from tests conducted at Texas A&M university [Nelson et al., 1986]. For this case, bulk flow results are available from the reference for comparison.

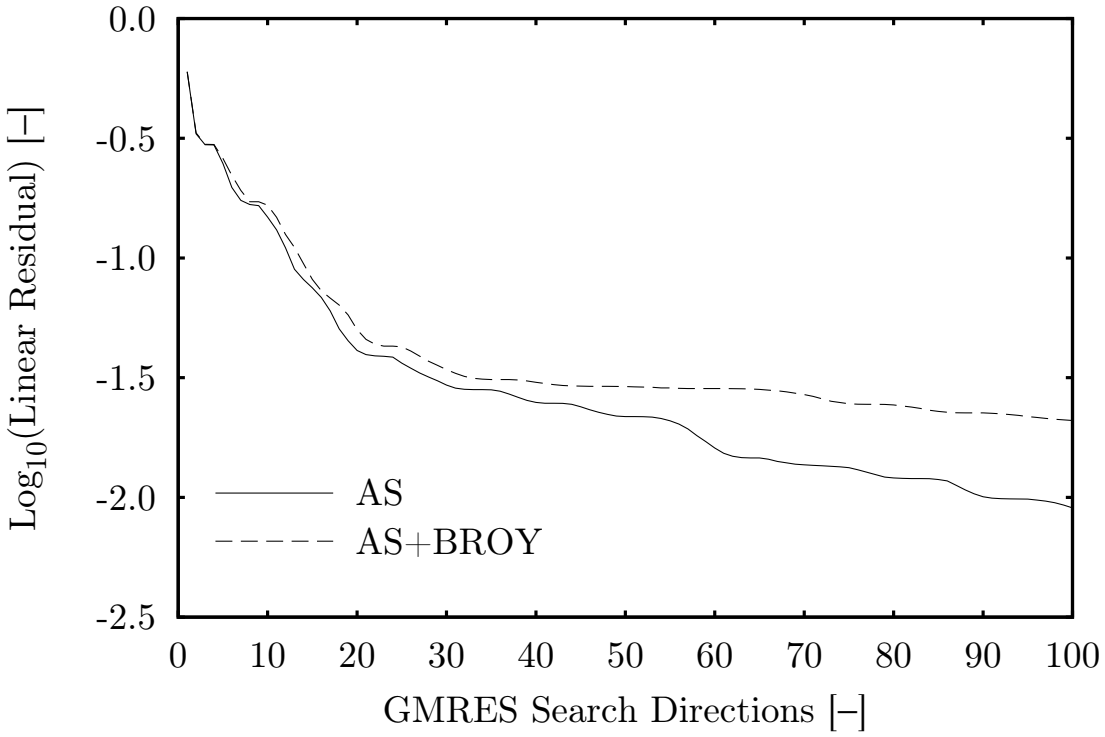

Figure IV.6: Improvement in the required number of search directions required for the Wright problem.

IV.3. Case Definition

The geometry is sketched in Fig. IV.7. The flow enters from the reservoir on the left, passes through the seal clearance, and exits to the sump on the right. The rotor is 75.679 mm in radius, the radial clearance of the seal is 1.114 mm, and the seal is 50.8 mm in length. The stagnation temperature was 305 K, the sump pressure was 100,000 Pa, and the shaft rotation rate was 52.36 rad/s. The reservoir pressure and mass flow are linked, and were found by digitizing the data in the reference. The pairs of stagnation pressure and mass flow rate are shown in Table IV.1. While

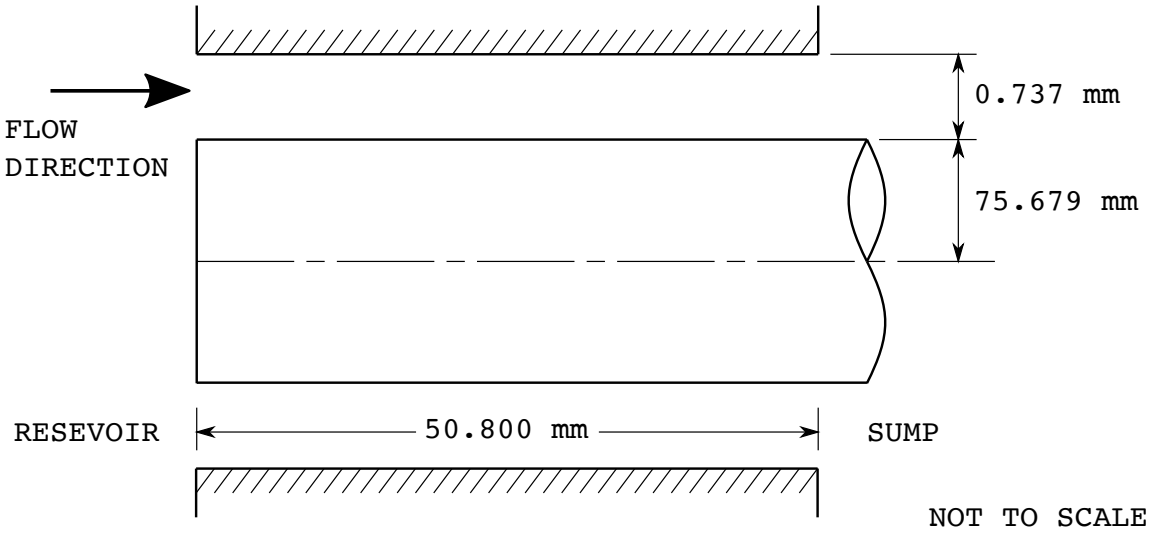

Figure IV.7: Nelson seal geometry.

cases with preswirl were available from the reference, the range given of $\zeta \pm 81$ did not seem plausible, nor was it specific enough to be useful. For this reason, it was decided that only the zero preswirl cases were worthwhile for comparison.

Table IV.1: Nelson seal operating conditions. [Nelson et al., 1986]

$P_{\text{res}}/P_{\text{sump}}$	P_{res} [Pa]	$\dot{m}$ [kg/s]
1.818	181,800	0.119
2.905	290,500	0.204
4.000	400,000	0.300
4.916	491,600	0.398
5.643	564,300	0.495

IV.4. Grid Generation

The computational grids for this case were composed of a single structured block which was algebraically generated, requiring only the clustering function from Chapter III. Table IV.2 gives the grid dimensions for the grid independence study, where IMAX is the number of points in the axial direction, JMAX is the number of points in the radial direction, KMAX is the number of points around the circumference, and Δs is the radial spacing at the wall, which was chosen to give a y^+ number of approximately unity. In a traditional grid independence study, the number of points in each direction is refined by a factor of two, resulting in the total number of grid points increasing by a factor of eight (and the total runtime increasing by a factor of 16). However, here it was deemed that refining in this manner would produce either a coarse grid that would be too coarse, or a fine grid that is excessively fine, so instead a factor of $\sqrt{2}$ was chosen, thereby resulting in an overall factor of eight increase in the number of nodes from the coarse to the fine grid.

Table IV.2: Grid dimensions for the Nelson seal.

Grid	IMAX	JMAX	KMAX	Δs [m]	Total Points
Coarse	50	100	200	8.0×10^{-7}	1,000,000
Medium	70	140	280	5.7×10^{-7}	2,744,000
Fine	100	200	400	4.0×10^{-7}	8,000,000

The results of the grid independence study is shown in Fig. IV.10. Since no more

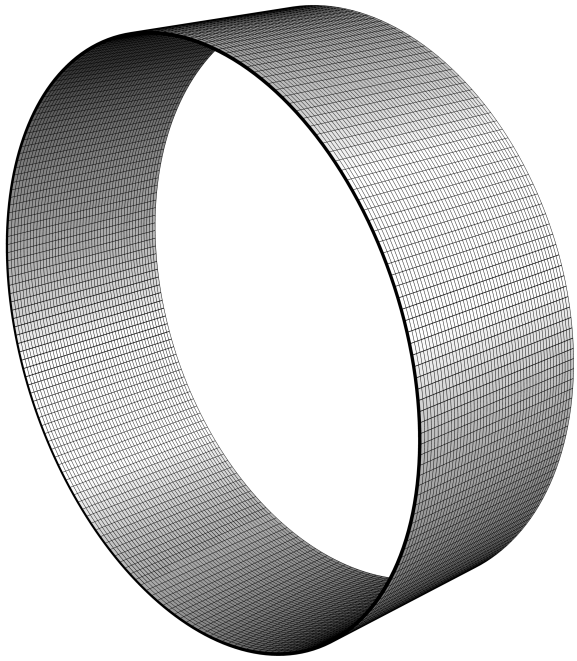

Figure IV.8: Overview of the coarse computational grid for the Nelson seal.

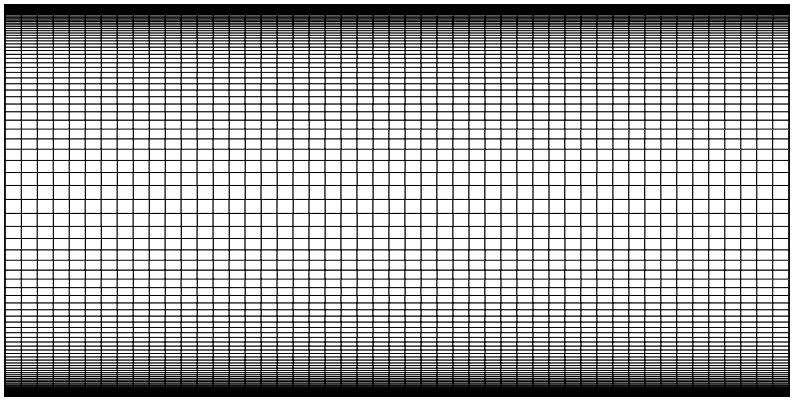

Figure IV.9: Cutaway view of the coarse computational grid for the Nelson seal.

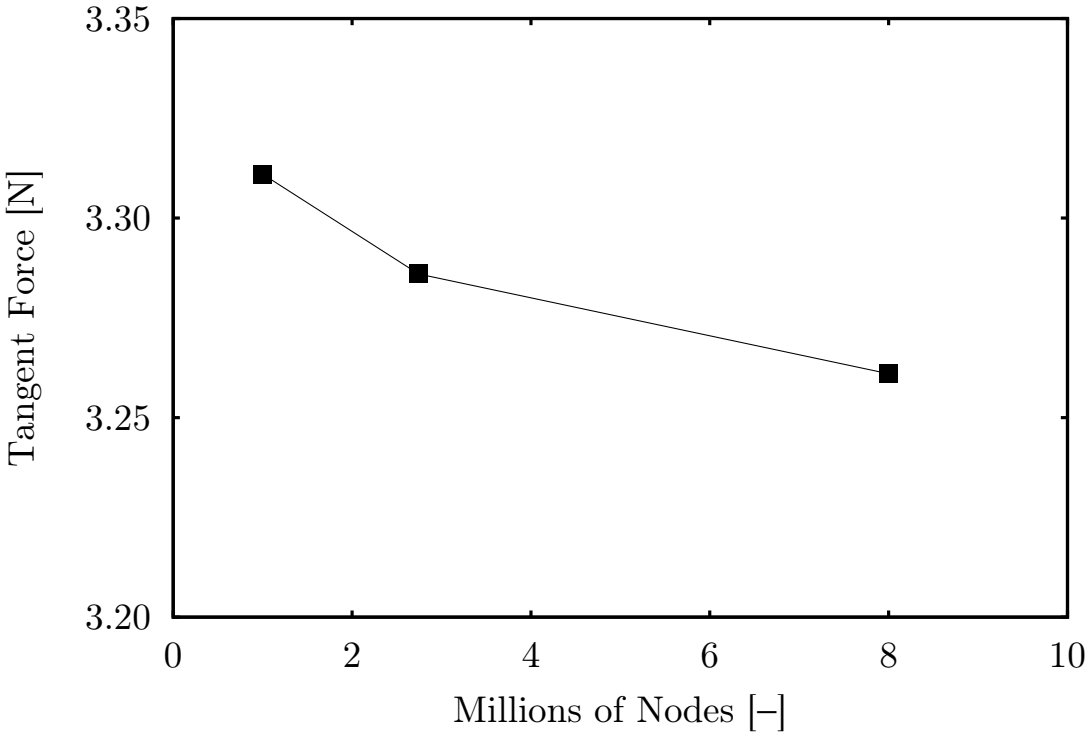

Figure IV.10: Grid independence study for the Nelson seal.

than a 2% difference was observed in the tangent force between the tested grids, it was determined that any of the tested grids would be sufficient. Hence, the medium grid was chosen to generate all the remaining results.

IV.5. Results

The direct stiffness is plotted in Fig. IV.11 with the experimental data and the accompanying bulk flow results. The reader may notice a small kink in the curves at the second data point; this occurs because only the first point is at an unchoked flow condition. The agreement between the present simulations and the experiment

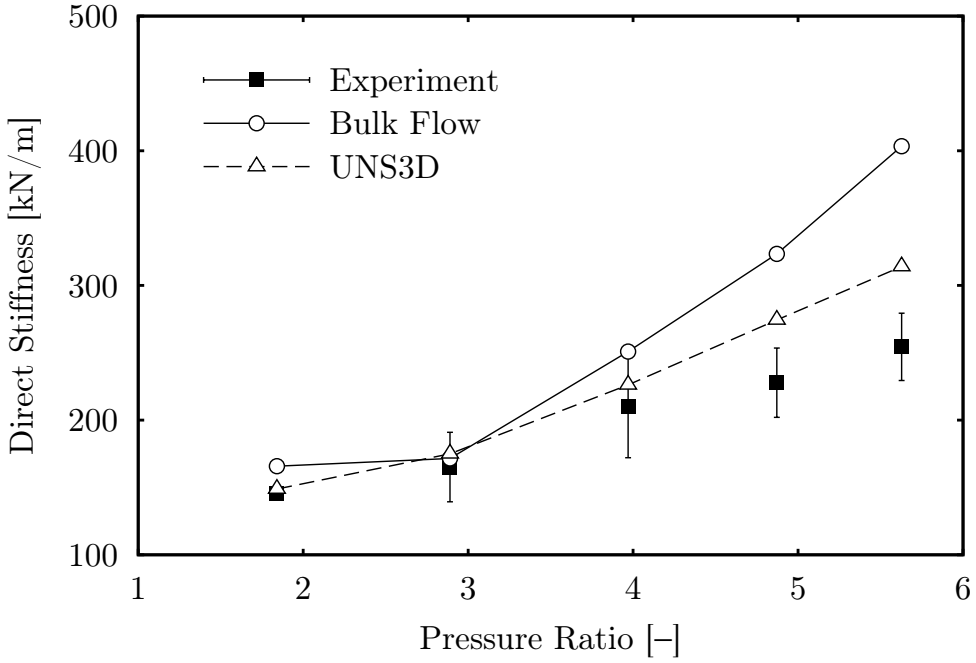

Figure IV.11: Direct stiffness versus pressure ratio for the Nelson seal. Experimental results from [Nelson et al., 1986].

is fair ($< 20\%$ error), and is within the experimental error bounds for all but the last two points. Even better, the error is about half that of the bulk flow model on average, with the exception of the second data point. It is unusual for an untuned CFD prediction to outperform a bulk flow model. However, it is worth noting that the direct stiffness is of little consequence for rotordynamic stability.

More important is the direct damping, shown in Fig. IV.12. Here, both models do an excellent job of matching the experimental data; the CFD simulation performs better at the low pressure ratios, while the bulk flow model is superior at the high pressure ratios. The second data point is of note, as it is outside of the general trend of the data and neither model manages to capture the deviation. The author has no

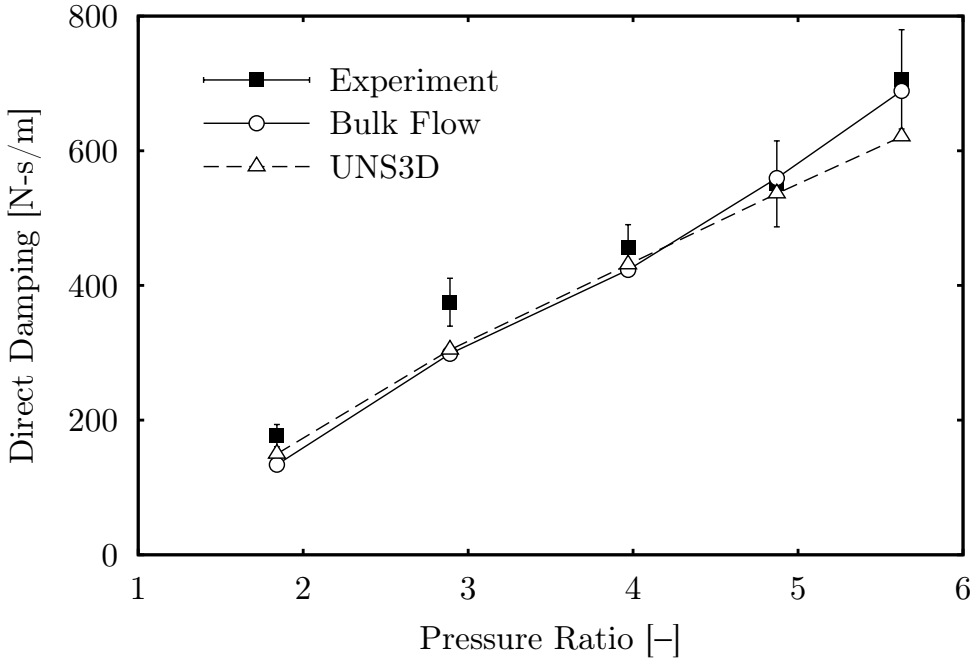

Figure IV.12: Direct damping versus pressure ratio for the Nelson seal. Experimental results from [Nelson et al., 1986].

convincing explanation for the deviation, but proposes that since the flow there is right on the verge of being choked, the experimental and simulation errors may both be magnified.

IV.6. Wright Seal

The next step up in complexity is a labyrinth seal with the smallest possible number of teeth. While such short seals are ideal for CFD validation, rotordynamic coefficient data for them is rare due to experimental difficulties with maintaining adequate pressure ratios and measurable forces. However, tests conducted at West-

inghouse in the late 1970's [Wright, 1978] found rotordynamic coefficients for a two-tooth tooth-on-stator labyrinth seal. This dataset is interesting for a few reasons. First, it was published around the same time as the foundational work by Benckert & Wachter [1980], so the field was in its infancy when this work was taking place. Second, while most authors obtain all four rotordynamic coefficients, Wright obtained only "radial stiffness" and an "excitation constant" (what later authors might call the effective direct and cross-coupled stiffness coefficients). These represent combinations of the four standard coefficients, and since only one whirling frequency was tested for each set of flow conditions, there is no way to convert them to the standard coefficients. While this could be viewed as a weakness of the data, it is in fact advantageous from the perspective of CFD validation, as we essentially have direct access to the forces measured during the experiment, rather than the curve-fit coefficients from many measurements at different frequencies.

IV.6.1. Case Definition

The geometry of the Wright seal is shown in Fig. IV.13. The flow enters from the reservoir on the left, passes through the clearances of the two teeth of the seal, and exits to the sump on the right. The rotor is 101.6 mm in radius, the radial clearance of the seal is 0.1585 mm, and the tooth height is 5.0305 mm. The teeth have a 30° chamfer on the back face and a 1.397 mm total thickness, and are spaced 12.7 mm apart. According to the discussion in [Gamal & Vance, 2008], past experiments and analytical investigations have suggested that seals with tapered or beveled teeth have superior leakage, and this was likely the working assumption around the time

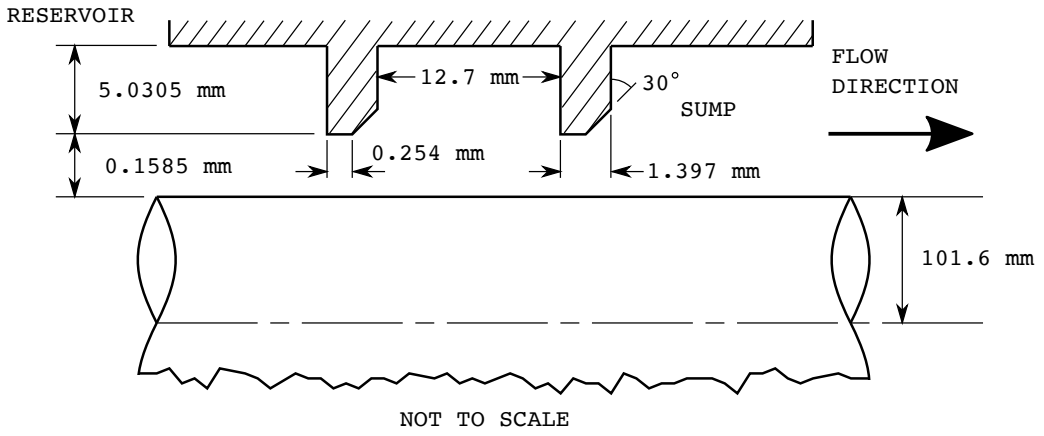

Figure IV.13: Geometry of the Wright seal.

of Wright's experiments. Later experiments provided conflicting evidence; the matter does not appear to be conclusively settled, and the optimal tooth profile likely depends on the operating conditions and overall thickness of the tooth.

Table IV.3 gives the reservoir pressures, whirling rates, mass flow rates, radial stiffness coefficients, and excitation constants used in both the experiments and the simulations; they are taken from Table 23 in [Wright, 1978]. The exit pressure was fixed at 103,421 Pa, and the running speed was 30 RPS. The temperature was not provided in the reference; however, in comparing with similar experiments, a value of 295 K was selected. The experiments were performed with an amplitude of oscillation equal to 0.019 mm; however, other references suggest a value of 10% of the clearance, and so a nearby value of 0.015 mm was selected for the simulations. No preswirl readings were available from the experiment. While Wright suggests that, in general, the preswirl velocity is probably around half of the rotor speed, this estimate assumes that the flow is fully developed. In reality, given the amount of

time the flow had to develop between the inlet pipe and the seal, it is more likely that the flow had approximately zero preswirl, and this value was used instead.

Table IV.3: Wright seal operating conditions. [Wright, 1978]

P_{res} [Pa]	Ω [RPS]	$\dot{m}$ [kg/s]	K_{s} [N/m]	E [N/m]
117277	13.75	0.00884	8318.5	-3677.6
131076	14.48	0.01387	17004.8	-5043.6
144876	15.20	0.01716	26671.8	-6217.0
158674	15.80	0.01988	35095.4	-7442.8
172473	16.24	0.02242	41592.6	-8090.8

IV.6.2. Grid Generation

The computational grids for this case were composed of three structured blocks per tooth, as shown in Fig. IV.14. The values for IMAX1, JMAX1, IMAX2, JMAX2, and KMAX can be specified independently, with the sole restriction that JMAX2 is greater than JMAX1. The blocks are all generated algebraically. Clustering is added to each solid wall using the algorithm presented in chapter III. In addition, padding blocks are placed upstream and downstream of the seal in an effort to keep the truncation boundaries as far from the area of interest as possible. The lengths of the upstream and downstream padding blocks were 20 and 30 mm, respectively, and the numbers of axial points used to discretize these blocks were IMAXIN and IMAXOUT, respectively.

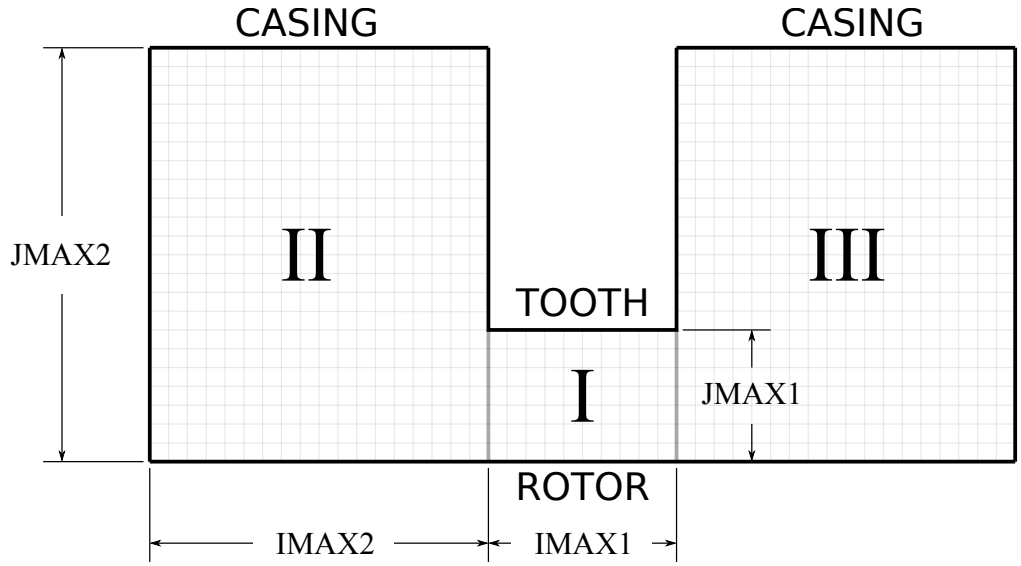

Figure IV.14: Grid block configuration for labyrinth seals.

As usual, a grid independence study was undertaken to determine a good compromise between accuracy and computational cost. The dimensions of the grids used in the study are listed in Table IV.4. As with the Nelson seal, refining each direction by a factor of 2 was deemed excessive. In this case, a factor of $\sqrt[3]{2}$ was used, so that the total number of grid points increased by a factor of two between subsequent grids.

The results of the grid independence study are shown in Fig. IV.17. This study is performed at the conditions corresponding to the lowest reservoir pressure in Table IV.3. The convergence is oscillatory, but the error is clearly decaying with the number of grid points, and there is no more than a 5% difference between the fine and superfine grids, so the fine grid was chosen for subsequent computations.

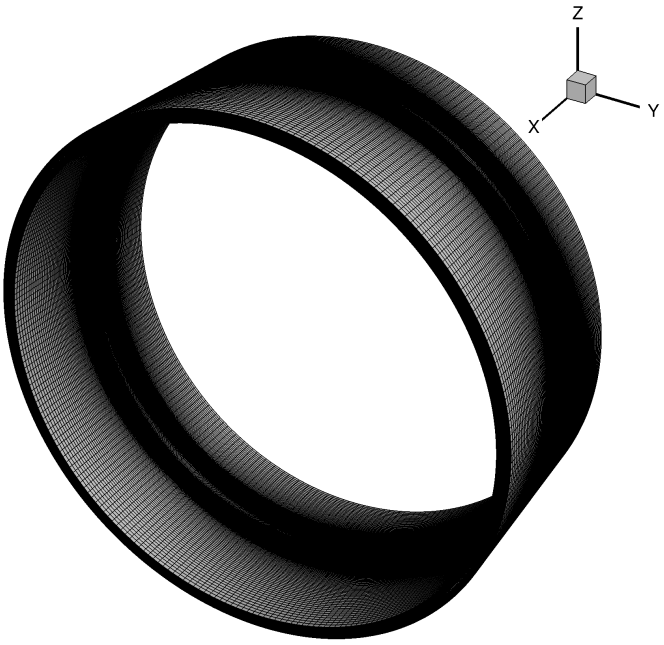

Figure IV.15: Overview of the computational grid for the Wright seal.

Table IV.4: Grid dimensions for the grid independence study for the Wright seal.

Grid	I_1	J_1	I_2	J_2	I_{IN}	I_{OUT}	K	Δs [m]	Total Points
Coarse	26	22	41	61	14	20	126	2.38×10^{-6}	1,618,875
Medium	33	28	52	77	17	25	159	1.89×10^{-6}	3,279,290
Fine	41	35	65	97	21	31	201	1.50×10^{-6}	6,540,600
Superfine	52	44	82	122	27	39	253	1.19×10^{-6}	13,129,704

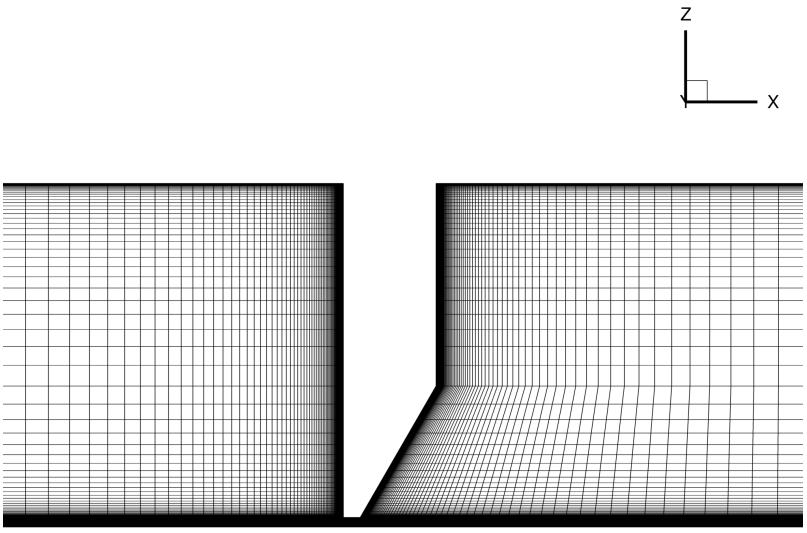

Figure IV.16: Cross-section view of the computational grid for the Wright seal.

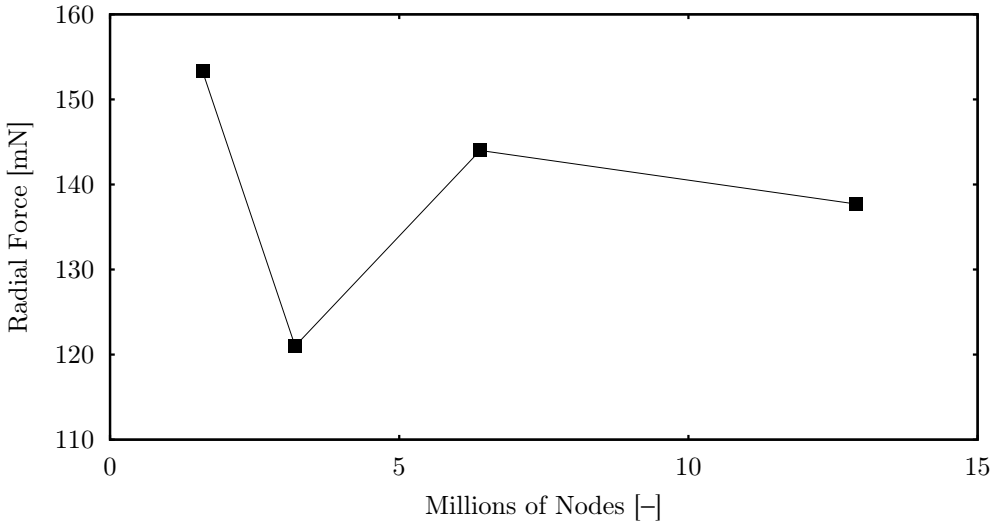

Figure IV.17: Grid independence study for the Wright seal.

IV.6.3. Results

The predicted effective radial stiffness for each pressure difference from Table IV.3 is plotted against the experimental results in Fig. IV.18. The agreement is quite good: the simulation overpredicts the stiffness, but only by 12-20%, and the slope matches well. However, once again, the radial stiffness is of little consequence to the stability of the rotor. More important is the excitation constant shown in Fig. IV.19, where the simulation underpredicts the experimental value by nearly half. However, it is worth reiterating that bulk flow predictions often show errors of over 100%, and so the discrepancy shown here is within reason. Putting aside the sources of error certainly present in the experiment itself, the discrepancy in Fig. IV.19 can likely be attributed to the inability of a two-equation turbulence model to adequately capture fully separated flows.

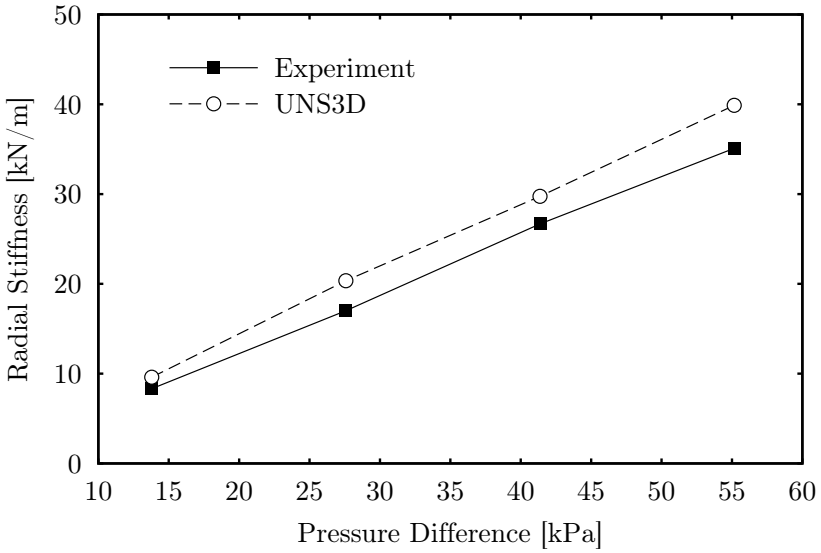

Figure IV.18: Effective radial stiffness versus pressure difference for the Wright seal. Experimental results taken from [Wright, 1978].

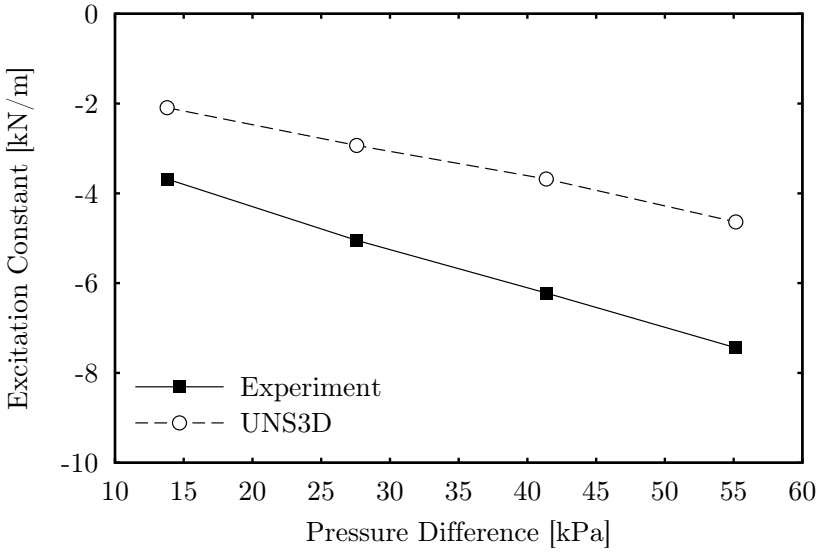

Figure IV.19: Excitation constant versus pressure difference for the Wright seal. Experimental results taken from [Wright, 1978].

CHAPTER V

CONCLUSIONS

This chapter presents conclusions that may be drawn from the previous chapters. In this dissertation, solver improvements to an existing Navier–Stokes code are described, and a new preconditioner was developed that combined a Quasi-Newton method with the Additive Correction Multigrid method to improve the subdomain coupling of an Additive-Schwarz preconditioner. The new preconditioner was shown to improve the number of linear system iterations required at each nonlinear step of the Navier–Stokes code, and the overall runtime, for viscous problems. The improved solver was used to predict the rotordynamic coefficients of several realistic seal flows from the literature, and the results are compared against the associated experiments. The following conclusions may be drawn from this work:

1. The implicit time-integration method was shown to provide superlinear convergence for small problems, even in conjunction with the turbulence model. This should not be expected to carry over to typical simulations, partly because the models are segregated, but also because modest timesteps are used in practice.

2. The implicit method, together with the AS preconditioner, was found to offer up to 40 times improvement in wallclock time over the existing explicit solver for real problems. While this is probably the upper bound of the improvement, one can expect improvements of five to ten times for harder problems.

3. The new preconditioner improves the coupling of the subdomains in the AS

preconditioner, and was found to offer substantial improvements in the number of GMRES iterations necessary for convergence within each global nonlinear iteration. Smaller, but substantial, benefits were seen in overall runtime.

4. The predicted rotordynamic coefficients showed excellent agreement with the experimental results for a smooth seal. This is likely because the flow is completely attached. The predictions for the simple labyrinth seal showed good agreement for the radial forces, but poor agreement for the tangential forces. This is likely due to the separated nature of the flow.

CHAPTER VI

FUTURE WORK

The goal of this work was to improve the runtimes of a Navier–Stokes code using advanced solvers with new preconditioners. The following work is recommended to further this goal:

1. The preconditioner has not been optimized in the sense of minimizing communication. Much can likely be gained by seeking opportunities to reduce the frequency of MPI calls.

2. Methods should be investigated to estimate the optimal weighting factor for the combined preconditioner.

3. The flows studied herein are likely outside the realm of applicability of the SST turbulence model. The effect of the turbulence model choice on labyrinth seal flows should be investigated.

REFERENCES

Baldassarre, L., Bernocchi, A., Fontana, M., Guglielmo, A., & Masi, G. (2014). Optimization of swirl brake design and assessment of its stabilizing effect on compressor rotordynamic performance. In *43rd Turbomachinery & 30th Pump Users Symposia*. Houston, Texas.

Barth, T. J., & Jesperson, D. C. (1989). The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366.

Benckert, H., & Wachter, J. (1980). Flow induced spring coefficients of labyrinth seals for application in rotor dynamics. *NASA. Lewis Res. Center Rotodyn. Instability Probl. in High-Performance Turbomachinery p 189-212*, .

Blazek, J. (2005). *Computational Fluid Dynamics: Principles and Applications*. (2nd ed.). Elsevier.

Brenner, T. A., Carpenter, F. L., Freno, B. A., & Cizmas, P. G. A. (2013). A reduced-order model for turbomachinery flows using proper orthogonal decomposition. In *ASME Turbo Expo 2013: Turbine Technical Conference and Exposition* GT2013-94914. ASME. doi:10.1115/gt2013-94914.

Brown, R. L. (2016). *Numerical Simulations of Hypersonic Aerothermoelastic Phenomena*. Dissertation Texas A&M University.

Broyden, C. (1969). A new method of solving nonlinear simultaneous equations. *The Computer Journal*, *12*, 94–99.

Cai, X.-C., Keyes, D. E., & Venkatakrishnan, V. (1995). Newton-krylov-schwarz: an implicit solver for cfd, .

Carpenter, F. L. (2014). Private Communication.

Carpenter, F. L. (2016). *Practical Aspects of Computational Fluid Dynamics for Turbomachinery*. Dissertation Texas A&M University.

Childs, D. W. (2013). *Turbomachinery Rotordynamics with Case Studies*. (3rd ed.). Minter Spring.

Childs, D. W., & Scharrer, J. K. (1986). An iwatsubo-based solution for labyrinth seals: Comparison to experimental results. *Journal of Engineering for Gas Turbines and Power*, *108*, 325. URL: https://doi.org/10.1115%2F1.3239907. doi:10.1115/1.3239907.

Chochua, G., & Soulas, T. A. (2007). Numerical modeling of rotordynamic coefficients for deliberately roughened stator gas annular seals. *Journal of Tribology*, *129*, 424. URL: https://doi.org/10.1115%2F1.2647531. doi:10.1115/1.2647531.

Dennis, J. E., & Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall.

Dennis, J. E., & Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations* volume 16. Siam.

Dryja, M., & Widlund, O. B. (1989). *Towards a unified theory of domain decomposition algorithms for elliptic problems*. New York University, Courant Institute of Mathematical Sciences, Division of Computer Science.

Gamal, A. J., & Vance, J. M. (2008). Labyrinth seal leakage tests: tooth profile, tooth thickness, and eccentricity effects. *Journal of Engineering for Gas Turbines and Power*, *130*, 012510.

Gargoloff, J. I. (2007). *A Numerical Method for Fully Nonlinear Aeroelastic Analysis*. Ph.D. thesis Texas A&M University College Station, Texas.

Gjesdal, T. (1996). A note on the additive correction multigrid method. *International communications in heat and mass transfer*, *23*, 293–298.

Godunov, S. K. (1959). A difference scheme for numerical computation discontinuous solution of hydrodynamic equations. *Math. Sbornik*, *47*, 271–306. Translated US Joint Publications Research Service, JPRS 7226, 1969.

Harten, A. (1983). Self adjusting grid methods for one dimensional hyperbolic conservation laws. *Journal of Computational Physics*, *50*, 235–269.

Haselbacher, A., & Blazek, J. (2000). Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids. *AIAA Journal*, *38*, 2094–2102.

Hellsten, A. (1998). Some improvements in menter's $k - \omega$ SST turbulence model. AIAA-98-2554. AIAA. doi:10.2514/6.1998-2554.

Hirano, T., Guo, Z., & Kirk, R. G. (2003). Application of CFD analysis for rotating machinery: Part 2 — labyrinth seal analysis. In *Volume 4: Turbo Expo 2003*. ASME. URL: https://doi.org/10.1115%2Fgt2003-38984. doi:10.1115/gt2003-38984.

Hutchinson, B., & Raithby, G. (1986). A multigrid method based on the additive correction strategy. *Numerical Heat Transfer, Part A: Applications*, *9*, 511–537.

Iwatsubo, T. (1980). Evaluation of instability forces of labyrinth seals in turbines or compressors, .

Jameson, A., Baker, T. J., & Weatherill, N. P. (1986). Calculation of inviscid transonic flow over a complete aircraft. In *AIAA 24th Aerospace Sciences Meeting* AIAA-86-0103. AIAA. doi:10.2514/6.1986-103.

Jameson, A., Schmidt, W., & Turkel, E. (1981). Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. AIAA Paper 81-1259.

John Vance, B. M., Fouad Zeidan (2010). *Machinery Vibration and Rotordynamics*. (1st ed.). John Wiley and Sons, Inc.

Keyes, D. E. (1995). Aerodynamic applications of newton-krylov-schwarz solvers. In *Fourteenth International Conference on Numerical Methods in Fluid Dynamics* (pp. 1–20). Springer.

Kim, K. S. (2003). *Three-Dimensional Hybrid Grid Generator and Unstructured Flow Solver for Compressors and Turbines*. Ph.D. thesis Texas A&M University College Station, Texas.

Kurohashi, M. (1980). Spring and damping coefficients of the labyrith seals. In *Proceedings IMechE-Second International Conference on Vibrations in Rotating Machinery, Cambridge, England*. volume 215.

Kvaalen, E. (1991). A faster broyden method. *BIT Numerical Mathematics*, *31*, 369–372.

Li, Z., Li, J., & Yan, X. (2013). Multiple frequencies elliptical whirling orbit model and transient RANS solution approach to rotordynamic coefficients of annual gas seals prediction. *Journal of Vibration and Acoustics*, *135*, 031005. URL: https://doi.org/10.1115%2F1.4023143. doi:10.1115/1.4023143.

Liliedahl, D. N., Carpenter, F. L., & Cizmas, P. G. A. (2011). Prediction of aeroacoustic resonance in cavities of hole-pattern stator seals. *Journal of Engineering for Gas Turbines and Power*, *133*.

Matula, N. R., & Cizmas, P. G. (2017). Numerical investigation of the effect of geometric design parameters on swirl brake performance. In *55th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics. URL: https://doi.org/10.2514%2F6.2017-0034. doi:10.2514/6.2017-0034.

Menter, F. R. (1993). Zonal two equation $k-\omega$ turbulence models for aerodynamic flows. In *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference* AIAA-93-2906. AIAA. doi:10.2514/6.1993-2906.

Menter, F. R., Kuntz, M., & Langtry, R. (2003). Ten years of industrial experience with the sst turbulence model. In K. Hanjalic, Y. Nagano, & M. Tummers (Eds.), *Turbulence, Heat and Mass Transfer 4* (pp. 625–632). Begell House, Inc.

Moore, J. J. (2003). Three-dimensional CFD rotordynamic analysis of gas labyrinth seals. *Journal of Vibration and Acoustics*, *125*, 427. URL: https://doi.org/10.1115%2F1.1615248. doi:10.1115/1.1615248.

Nelson, C., Childs, D., Nicks, C., & Elrod, D. (1986). Theory versus experiment for the rotordynamic coefficients of annular gas seals: Part 2—constant-clearance and convergent-tapered geometry. *Journal of Tribology*, *108*, 433–437.

Nielsen, K. K., Childs, D., & Myllerup, C. (2001). Experimental and theoretical comparison of two swirl brake designs. *Journal of turbomachinery*, *123*, 353–358.

Pelletti, J. M. (1990). *A Comparison of Experimental Results and Theoretical Predictions for the Rotordynamic Coefficients of Short (L/D=1/6) Labyrinth Seals*. Master's thesis Texas A&M University College Station, Texas.

Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press.

Rhode, D. L., Hensel, S. J., & Guidry, M. J. (1992). Labyrinth seal rotordynamic forces using a three-dimensional navier-stokes code. *Journal of Tribology*, *114*, 683. URL: https://doi.org/10.1115%2F1.2920936. doi:10.1115/1.2920936.

Roe, P. L. (1986). Characteristics Based Schemes for the Euler Equations. *Annual Review of Fluid Mechanics*, *18*, 337–365.

Saad, Y. (1993). A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, *14*, 461–469.

Saad, Y. (2003). *Iterative methods for sparse linear systems* volume 82. siam.

Saad, Y., & Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, *7*, 856–869.

Scharrer, J. K. (1988). Theory versus experiment for the rotordynamic coefficients of labyrinth gas seals: Part i—a two control volume model. *Journal of Vibration Acoustics Stress and Reliability in Design*, *110*, 270. URL: https://doi.org/10.1115%2F1.3269513. doi:10.1115/1.3269513.

Soghe, R. D., Micio, M., Andreini, A., Facchini, B., Innocenti, L., & Ceccherini, A. (2013). Numerical characterization of swirl brakes for high pressure centrifugal compressors. In *Volume 6C: Turbomachinery*. ASME. URL: https://doi.org/10.1115%2Fgt2013-94075. doi:10.1115/gt2013-94075.

Thompson, J. F., Warsi, Z. U., & Mastin, C. W. (1985). *Numerical grid generation: foundations and applications* volume 45. North-holland Amsterdam.

Vanden, K. J., & Whitefield, D. (1995). Direct and iterative algorithms for the three-dimensional euler equations. *AIAA journal*, *33*, 851–858.

Venkatakrishnan, V. (1995a). Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, *118*, 120–130.

Venkatakrishnan, V. (1995b). *Implicit schemes and parallel computing in unstructured grid CFD.*. Technical Report INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING HAMPTON VA.

Van der Vorst, H. A. (1992). Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, *13*, 631–644.

Wilcox, D. C. (2010). *Turbulence Modeling for CFD*. (3rd ed.). DCW Industries.

Wright, D. V. (1978). Air model tests of labyrinth seal forces on a whirling rotor. *Journal of Engineering for Power*, *100*, 533–543.

Wyssmann, H. R., Pham, T. C., & Jenny, R. J. (1984). Prediction of stiffness and damping coefficients for centrifugal compressor labyrinth seals. *Journal of Engineering for Gas Turbines and Power*, *106*, 920. URL: `https://doi.org/10.1115%2F1.3239659`. doi:`10.1115/1.3239659`.

Yan, X., Li, J., & Feng, Z. (2011). Investigations on the rotordynamic characteristics of a hole-pattern seal using transient CFD and periodic circular orbit model. *Journal of Vibration and Acoustics*, *133*, 041007. URL: `https://doi.org/10.1115%2F1.4003403`. doi:`10.1115/1.4003403`.