A Thesis

entitled

# Dynamic Grid Motion in a High-Order Computational Aeroacoustic Solver

by

Michael Alan Heminger

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Mechanical Engineering

Dr. Ray Hixon, Committee Chair

Dr. Abdollah Afjheh, Committee Member

Dr. Chunhua Sheng, Committee Member

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo
August 2010

An Abstract of

Dynamic Grid Motion in a High-Order
Computational Aeroacoustic Solver

by

Michael Alan Heminger

Submitted to the Graduate Faculty as partial fulfillment of the requirements
for the Master of Science Degree in Mechanical Engineering

The University of Toledo
August 2010

In this work, moving meshes will be employed to solve unsteady computational problems, while maintaining high-order, and high-accuracy. The main problem of interest is that of a plunging piston. The plunging piston problem, first presented in the First Workshop for Computational Aeroacoustics.

Typically, computational aeroacoustics is seperate from aeroelasticity, a field where moving surfaces is integral. This project will join the two field, attempting to resolve propagating waves from a moving boundary. While this particiular problem has been attempted before, it was done using boundary conditions.

This project's main goal is to bridge the gap between computation fluid dynamic disciplines, creating a general standalone mesh morpher, enabling a new breed of acoustic problems to be solved.

To do this, a highly efficient method of moving the mesh will need to be developed. Since the code uses high-order schemes to resolve the small sound waves, the mesh mover must be methods which keep the grid metrics continuous and smooth.

# Acknowledgments

I would first like to greatly acknowledge and thank Dr. Ray Hixon. Through classroom instruction and personal mentoring I have acquired a background; knowledge that will fuel my career. His patience, rather endurance to stay the path and believe that one day this nearly decade long process would complete is legendary.

I would also like to acknowledge Reid Melville, the original creator of GridWarp and the mastermind behind the process. Without his tireless efforts creating such a efficient code, this project would not exist.

Lastly, I would like to thank the NASA Engineering Safety Center (NESC) who funded this project and Dan Sutliff, the technical monitor.

# Contents

**Abstract** iii

**Acknowledgments** iv

**Contents** v

**List of Tables** viii

**List of Figures** ix

**List of Abbreviations** xiii

**List of Symbols** xiv

**1 Introduction** 1

1.1 Background . . . . . 1

1.1.1 Aeroelasticity . . . . . 2

1.1.2 Aeroacoustic Review . . . . . 3

1.1.3 Aeroacoustic Examples . . . . . 4

1.1.4 Coupling the fields . . . . . 8

**2 Grid Motion Background** 9

2.1 Grid Motion Methods . . . . . 9

2.1.1 Algebraic Methods . . . . . 9

2.1.2 Transfinite Interpolation . . . . . 12
2.1.3 Laplacian Method . . . . . 14
2.1.4 Vertex Springs . . . . . 14
2.2 Moving Wall Boundary Conditions . . . . . 17

**3 Grid Motion** **20**
3.1 Surface Projection Point Construction . . . . . 20
3.2 Distribution of Surface Attention . . . . . 22
3.3 Deformation Response . . . . . 23
3.4 Improving on Grid Motion Efficiency . . . . . 24

**4 Initial Grid Tests** **27**

**5 Vibrating Membrane** **35**

**6 Plunging Piston** **38**
6.1 Introduction . . . . . 38
6.2 Procedure . . . . . 42
6.3 Mesh Generation . . . . . 43
6.4 Grid Performance . . . . . 43
6.5 Results . . . . . 49
6.5.1 Problems . . . . . 52
6.5.1.1 Far-Field Definition . . . . . 52
6.5.1.2 Piston Definition . . . . . 52
6.5.2 An Effective Radius . . . . . 61
6.5.3 Problem Resolution: High-Frequency Comparison . . . . . 68
6.5.4 Low-Frequency Revisited: Part I . . . . . 73
6.5.5 Medium Frequency . . . . . 75
6.5.6 Very High Frequency . . . . . 75

6.5.7 Redefinition of Piston Edge . . . . . 80

**7 Conclusions** **85**

7.1 Piston Test Case Lessons Learned . . . . . 85

7.2 Summary . . . . . 86

7.3 Future Work . . . . . 88

**8 Looking Forward** **89**

8.1 The Black Box . . . . . 89

8.2 Rigid Surfaces . . . . . 90

8.2.1 Extension to CFD . . . . . 92

**References** **95**

**A Derivation of Blending Volume** **102**

**B Transfinite Interpolation (TFI) Further Reading** **105**

B.1 Recusion Formulation . . . . . 107

**C Thomas Algorithm** **108**

# List of Tables

6.1 Leading error terms for various points per wavelength . . . . . 58

6.2 Comparison of Various Parameters for Different Blending Regions, T=25 68

6.3 Comparison of Various Parameters for Different Blending Regions, T=12 73

# List of Figures


1-1 Grumman X-29 in flight . . . . . 2

1-2 Unsteady Shock Cell Structure inside Jet Plume[51] . . . . . 5

2-1 Blending Function $f(x) = 3x^2 - 2x^3$ . . . . . 10

2-2 Example one-dimensional vertex spring domain . . . . . 16

3-1 Target Point Projection onto the Surface . . . . . 21

3-2 Surface Attention Function Schematic . . . . . 23

4-1 Initial AFRL GridWarp test, $D_{far} = 4.0$ . . . . . 28

4-2 Initial AFRL GridWarp test $D_{far} = 6.0$ . . . . . 28

4-3 Initial AFRL GridWarp test $D_{far} = 8.0$ . . . . . 29

4-4 Initial AFRL GridWarp test $D_{far} = 10.0$ . . . . . 29

4-5 Multiple Surface GridWarp test $D_{far} = 2.0$ . . . . . 30

4-6 Multiple Surface GridWarp test $D_{far} = 5.0$ . . . . . 30

4-7 Multiple Surface GridWarp test $D_{far} = 6.0$ . . . . . 31

4-8 Multiple Surface GridWarp test $D_{far} = 8.0$ . . . . . 31

4-9 Multiple Surface GridWarp test $D_{far} = 10.0$ . . . . . 32

4-10 Multiple Surface GridWarp test, Rotation Matrix Off . . . . . 33

4-11 Multiple Surface GridWarp test, Different Surface Attentions . . . . . 34

5-1 Vibrating Membrane Test at 0.8 seconds . . . . . 36

5-2 Vibrating Membrane Test at 2.0 seconds . . . . . 37
5-3 Vibrating Membrane Test at 6.0 seconds . . . . . 37

6-1 Analytic Bessel Term . . . . . 40
6-2 YZ (+X) Slice Farfield Grid . . . . . 44
6-3 YZ (+X) Slice Nearfield Grid . . . . . 44
6-4 XY (+Z) Slice Showing Piston Edge Grid Clustering . . . . . 45
6-5 Isometric View of Grid Showing Topology . . . . . 45
6-6 Deformed Grid Over a Complete Cycle . . . . . 46
6-7 Center of Piston at the Start of a Run . . . . . 47
6-8 Piston blend length of 5 at various times over cycle . . . . . 48
6-9 Piston blend length of 1 at various times over cycle . . . . . 49
6-10 Initial Period 25 Piston Test, Polar Plot . . . . . 50
6-11 Initial Period 25 Real Component . . . . . 51
6-12 Initial Period 25 Imaginary Component . . . . . 51
6-13 Comparison of Radial Sampling . . . . . 53
6-14 Time-domain waveform of *A major* chord . . . . . 54
6-15 Frequency-domain waveform of *A major* chord . . . . . 55
6-16 FFT Example with only 4 points per wavelength . . . . . 56
6-17 FFT Example from Flow Solve Point of View . . . . . 56
6-18 Comparison of Higher Points per Wavelength . . . . . 57
6-19 Effect of Interpolation on Defined Piston Motion . . . . . 58
6-20 Period 25 Piston Test, Polar Plot . . . . . 59
6-21 Period 25 Real Component . . . . . 60
6-22 Period 25 Imaginary Component . . . . . 60
6-23 Comparison of FFT Snap contours for blending regions of: a. 2, b. 5, c. 10 . . . . . 63
6-24 Comparison of Identical Runs Using Different Blend Areas . . . . . 64

6-25 Pressure Pertubation Period 25, Small Blend . . . . . 65
6-26 Numerical Results vs Modified Analytic Solution - Small Blend . . . 65
6-27 Numerical Results vs Modified Analytic Solution - Medium Blend . . 66
6-28 Numerical Results vs Modified Analytic Solution - Large Blend . . . 66
6-29 Comparison of Piston Blending Regions . . . . . 67
6-30 Comparison of Blending Sizes for High Frequency Test . . . . . 69
6-31 Comparison of Blending Sizes for High Frequency Test-Close Up . . . 70
6-32 T=12 Run with Large Piston Blend . . . . . 71
6-33 T=12 Run with Medium Piston Blend . . . . . 71
6-34 T=12 Run with Small Piston Blend . . . . . 72
6-35 T=12 Run with Small Piston Blend - Close Up . . . . . 72
6-36 T=25 Test, Updated Blend Comparison . . . . . 74
6-37 T=25 Test, Very Small Blend Region vs Analytical and Effective . . . 74
6-38 Pressure Pertubation Period 16 . . . . . 75
6-39 T=16 Test, Polar Amplitude . . . . . 76
6-40 T=16 Test, Polar Amplitude, Close-Up . . . . . 76
6-41 T=16 Test, Axial Real Component . . . . . 77
6-42 T=16 Test, Axial Imaginary Component . . . . . 77
6-43 Pressure Pertubation Period 8 . . . . . 78
6-44 T=8 Test, Polar Amplitude . . . . . 78
6-45 T=8 Test, Polar Amplitude, Close-Up . . . . . 79
6-46 T=8 Test, Axial Real Component . . . . . 79
6-47 T=08 Test, Axial Imaginary Component . . . . . 80
6-48 Period 25 Results for Volume Matched Blending Region . . . . . 82
6-49 Period 12 Results for blend region having $r' = 10.0$ . . . . . 82
6-50 Period 12 Results for blend region having $r' = 10.0$ - Close up . . . . 83
6-51 Very High Frequency Test using effective radius of 10 . . . . . 84

6-52 Very High Frequency Test using effective radius of 10 - Close Up . . . 84

# List of Abbreviations

| | |
|---|---|
| AFRL | Air Force Research Laboratories |
| CAA | Computational Aeroacoustics |
| CFD | Computational Fluid Dynamics |
| CFL | Courant-Friedrichs-Lewy (number) |
| DFT | Discrete Fourier Transform |
| DRP | Dispersion-Relation Preserving (scheme) |
| FEA | Finite Element Analysis |
| FFT | Fast Fourier Transform |
| IDF | Influence Decay Function |
| NASA | National Air and Space Administration |
| NESC | NASA Engineering Safety Center |
| RK | Runge-Kutta scheme |
| SAF | Surface Attention Function |
| TFI | Transfinite Interpolation |

# List of Symbols

$[]_0$ ..... The zero subscript typically indicates freestream properties
$\vec{Q}$ ..... Vector of conserved variables
$\rho$ ..... Density
$p$ ..... Pressure
$c$ ..... Speed of sound
$\gamma$ ..... Ratio of specific heats
$\vec{V}$ ..... Velocity vector
$u$ ..... Velocity in x-direction
$v$ ..... Velocity in y-direction
$w$ ..... Velocity in z-direction
$\vec{n}$ ..... Unit normal vector
$J$ ..... Jacobian matrix of grid

$p'$ ..... Pressure pertubation
$k$ ..... Wave number
$Q_p$ ..... Source strength of monopole
$\omega$ ..... Frequency
$r$ ..... Radius
$z$ ..... Piston Radius
$U_p$ ..... Peak Surface Velocity of Piston
$J_n$ ..... $n$th order Bessel Function of the First Kind
$A$ ..... Piston Maximum Displacement Amplitude
$V$ ..... Maximum deformed piston volume
$r_0$ ..... Start of radius edge blend
$r_1$ ..... End of radius edge blend
$r^*$ ..... Volumetric equivalent piston radius
$r'$ ..... Numerically best fit equivalent piston radius

# Chapter 1

# Introduction

## 1.1 Background

The field of computational fluid dynamics (CFD) is, in a sense, governed by computing. The popular Moore's Law[37] states that computing power essentially doubles every 24 months. This holds true for many types of empirical data: hard drive capacity, processor speed, etc. Because of this exponential increase in computing power, more interesting and more complex computational problems are becoming within reach.

In years past, the typical CFD problem was set up in a somewhat non-physical way. After the outer and inner boundaries were determined, a mesh was "laid" over top of the domain. The Navier-Stokes equations, a coupled set of partial differential equations, were then solved at each point. The time derivative could then be calculated, allowing the solver to "march in time".

This has worked very well for many years, however, there is one aspect which can be non-physical. The boundaries, and therefore the mesh itself, has typically been fixed. For low speed, or otherwise cases where changes in pressure and or deformation is small, this works reasonably well. However, there are many cases where these forces

and deformations should be accounted for.

### 1.1.1 Aeroelasticity

Deformation of any body is inevitable, and brings about the concept of aeroelasticity [6]. The slightest force on even the most rigid body will deform it. It is this unavoidable deformations that bring cause to all aeroelastic problems [5].

Aeroelastic problems have been a concern to aerodynamic engineers for quite some time. One famous program which showcased aeroelastic problems and subsequent lessons learned was the Grumman X-29 demonstrator program [10]. The X-29 aircraft featured a forward-swept wing configuration. Forward-swept wings can be useful in aircraft design. Most notably, they allow for increased maneuverability, particularly at high angles of attack.


Figure 1-1: Grumman X-29 in flight

The downside of this design is aeroelastic issues. The aerodynamic lift produces a twisting moment which rotates the leading edge of the wing upward. This results in a higher angle of attack, which produces more lift, exacerbating the problem further. In addition to this rather steady-state phenomenom, impulse forces could lead to vibratory issues at the wing roots, possibly leading to high-cycle fatigue.

While fixed-wing aircraft exhibit a number of opportunities for aeroelastic response to cause failure, there are a myriad of aeroelastic considerations for rotorcraft [29]. Some of these responses include: flapping frequency response, forward-flight flapping stability, pitch/flap flutter, and ground resonances.

For many years, reduced order fomulations were used for linear vibration analysis of bounded fluid-structure systems [41]. This allowed problems such as blade flutter and other coupled fluid-modal analyses to be performed. However, typically they involved some variational approach; the problems and solutions weren't general. Mode shapes needed to be calculated prior to the flow solution, which then used modal-superposition for solution.

In the early 1990's, fully coupled fluid-structure interaction systems were devloped, allowing an entire new set of problems to be solved. Much of this work done on unstructured grids was performed by Batina [3] [43] [2].

### 1.1.2 Aeroacoustic Review

Aeroacoustics is typically thought of as an offshoot from computational fluid dynamics, rather than aeroelasticity. This is mainly due to the differences in the way problems are approached and solved. Tam [56] lists several important challenges that are for the most part unqiue to computational aeroacoustics.

- Aeroacoustics problems are nearly by definition time dependent. Noise is by nature of a time-dependent phenomenon, so the accurate simulation of it should be as well. As such, time-marching schemes must not only have low dissipation, but propagate waves correctly in time as well.

- Aeroacoustics problems typically involve frequency ranges that spread over a wide bandwith. Numerical resolution of the high frequency waves with short wavelengths can become a large hurdle in both simulation as well as efficiency.

- Acoustic waves usually have small amplitudes, often times orders of magnitude lower than the mean flow. As such, in order to compute the sound waves accurately, numerical schemes must be employed which have extremely low numerical "noise".

- Typically, the sound waves need to be propagated to the far field, or at least some observation point a distance away from the sound generating feature. This requires that resolution and accuracy not only exist near the sound generation, but all the way to that measurement point. Because of this, numerical schemes must have extremely low dispersion and dissipations to respectively limit phase and amplitude errors.

- Computational domains, regardless of exact discipline are by definition finite in size. For typical CFD problems, flow disturbances tend to decay away from the body of interest. They, therefore, are usually very small at the boundaries. This can lend itself towards boundary condition generation. Acoustic waves on the other hand, typically decay very slowly and reach the boundaries with significant amplitude. To avoid reflections back into the domain, outflow boundary conditions must be well defined in order assist the waves to exit the domain smoothly. Boundary conditions are often times the cause of issues in numerical simulation and moving wall boundary conditions will be discussed in Section 2.2.

### 1.1.3 Aeroacoustic Examples

Several typical aeroacoustic applications are listed below. While they certainly do not encompass the entirety of the field, they do give the reader an essence of the field, along with unique approaches and strategies needed to solve such rigorous problems.

**Screech Tones** Screech tones is sound emitted by imperfectly expanded supersonic jets [55]. The tones are generated by self-excited feedback loops driven by instability of the jet flow. Because this feedback loop is highly nonlinear, there was no analytical or even empirical relations for predicting the intensity. Around the year 2000, much work was performed on this topic by Shen and Tam [51] [50] [52], Manning and Lele [34] [33], and Loh [12].

The individual shock cells in the jet plume are critical to capturing the screech tone. These goals, one finds, are incompatible in CFD. Traditionally, shock capturing schemes would dissipate the sound waves. In order to meet both goals, artificial selective damping schemes were developed [57].

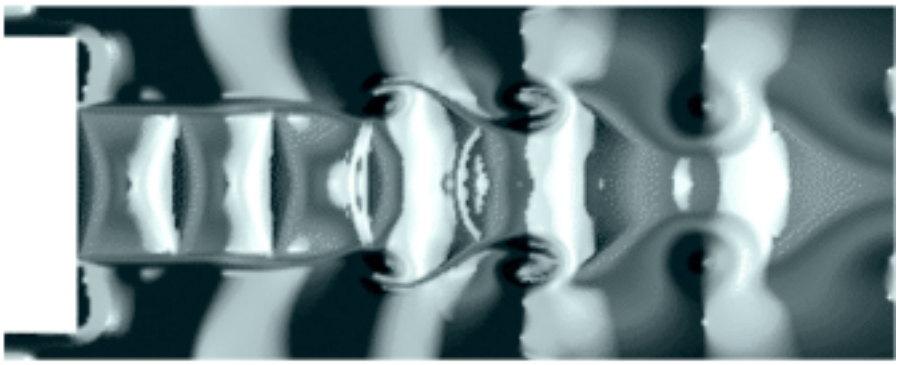

Figure 1-2: Unsteady Shock Cell Structure inside Jet Plume[51]

**Acoustic Liners** Acoustic liners are typically a sheet full of small holes [38] [23]; quite simple in design. However, this seemingly simply component is presently the most effective way of reducing fan noise from jet engines. In addition to fan noise, similar-style acoustic liners are used in augmentors to both assist in heat transfer effects, along with the aformentioned acoustic effects.

When sound impinges on the liner, pressures on either side of the holes rise and fall. In the past, there were several suggestions as to the cause of the acoustic dissipation. Some suggested that wall friction was responsible, while others claimed that the

oscillatory flow led to oscillatory turbulent jets. The dissipation in the later case was the result of acoustic energy to fluid turbulence, which is subsequently dissipated by molecular viscosity.

The discussion motivated several researchers[54] [8] [9] to perform direct numerical simulation (DNS) of a single slit and single cavity under acoustic excitation in parallel with physical experiments.

It was found that at low incident sound pressure levels, a jet-like oscillatory shear layer formed near each side of the slit. Viscous dissipation was associated with these shear layers, and was the dominant force in dissipating the acoustic energy.

At high incident sound pressure levels, vortex shedding occurs at the corners connecting the resonator to the outside. The vortex shedding caused a large increase in the dissipation. As the vortices are shed, acoustic energy is converted into rotational kinetic energy, which is subsequently dissipated into heat by molecular viscosity.

While this problem seems to be limited in scope, the agreement between the numerical and physical experiments offered confidence in the prediction of acoustic and flow modeling particularly with vortex shedding, direct numerical simulations, and solving problems on multi-scales.

Multi-scale problems were identified as one of the characteristics of CAA. When the problem was attempted by typical CFD approaches, investigators used uniform mesh sizes which were adequate to resolve acoustic waves, but too coarse to resolve the viscous shear layers near the slit.

**Open Cavity Flow** Flow passing over a rectangular cavity is one that appears benign at first. One would almost think a closed-form solution exists to such a simplistic seeming problem. However, the phenomenon of aerodynamically generated noise by flow passing over said cavity is one that has been studied extensively. This is perhaps due to one of two reasons.

First, there are a variety of applications and industries which this problem can be applied to. In the automotive industry this problem exists at doors and essentially any other body crease which opens to the interior of the vehicle. In the aerospace industry, this problem is especially important to weapon bays, and where additional noise sources are critical.

The second reason this problem may have been studied to extensively is the difficulty of the problem, and the solutions and approaches that solving it has brought to the field of computational aeroacoustics.

Rossiter[48] was one of the first researchers who described the feedback mechanism based on wind tunnel experiments. Rossiter described a four-step process, commonly referred to as the Rossiter or shear-layer mode, which creates the periodic flow pattern.

1. Vortices shed at the leading edge of the cavity convect downsream along the generated shear layers until they reach the trailing edge.

2. At the trailing edge, the vortices interact with the downstream wall of the cavity causing the generation of acoustic waves. Part of these acoustic waves are radiated above the cavity and propagate into the acoustic far-field.

3. The acoustic waves that don't propagate into the far-field are radiated inside the cavity. They propagate upstream reflecting off the back wall until they reach the leading edge.

4. Finally reaching the upstream wall of the cavity, the acoustic waves cause the shedding of a new vortex at the leading edge.

More than 30 years later, Gharib and Roshko[19] observed another mode of oscillation, called wake mode. It was observed that when the length to depth ratio, a variable not included in Rossiter's analysis, is increased, the flow becomes violent and unsteady.

It was seen that a large vortex will grow inside the cavity, being formed at the leading edge. Once the vortex becomes large enough, it is ejected out at the trailing edge. The flow features in this mode are qualitative significantly different from shear-layer mode.

While more work has been done since Gharib, using two-dimensional direct numerical simulation[49] [26] and hybrid prediction techniques[61], it is clear to see how this problem has become an aeroacoustic benchmark. Such a benchmark allows one to evaluate the performance of a computational aeroacoustic code, both tonal noise generation and the aerodynamic flow-acoustic feedback mechanisms.

### 1.1.4 Coupling the fields

The work presented here aims to couple these two fields: aeroelasticity and aeroacoutics. While adaptive and other strategies have been successfully demonstrated which move meshes for numerical reasons, the goal here is to incorporate moving boundaries into a high-order, high-accuracy solver.

While moving grids have been demonstrated for many years [62] [32], the work typical involved the use of low order flow solvers. For computational efficiency, the typical fluid-structure interaction problem is now solved using boundary conditions.

The boundary condition in this sense is that the wall is moving. Rather than setting the covariant and contravariant velocities to zero, one sets the appropriate velocity to the wall velocity. So long as the wall doesn't move a significant amount, this approach works well.

However, this work aims to incorporate a stand-alone mesh morpher. In this case, the boundary conditions are handled by the mesh morpher, which deforms the volume grid. This will allow the code to compute an accurate solution while remaining completely general.

# Chapter 2

# Grid Motion Background

Grid motion is not desired, but required for several different types of computational problems such as: aeroelastic problems, where dynamic forces are modifying existing surfaces; deformation of movable interfaces, such as a water/air interface for flows past ships; motions of objects moving relative to one another, such as in the case of rotorcraft.

Because of this demand for grid motion, several different approaches have been taken. Some techniques come from mechanical analogies, while others mirror actual grid generation.

## 2.1 Grid Motion Methods

### 2.1.1 Algebraic Methods

Algebraic methods are quite possibly the simplest and can be the most accurate of all the methods to be presented. For algebraic methods to be used though, predetermined motion must be known.

Consider the simple case of a orthogonal domain with a periodically oscillating

wall as one entire face, with displacement given by

$$x = A \sin \omega t \tag{2.1}$$

The main goal of grid morphing is to ensure that the grid remains smooth, with continuous derivatives. Knowing this, one could employ the common blending function

$$f(x) = 3x^2 - 2x^3 \tag{2.2}$$

The function and it's derivative are shown in Figure 2-1. As shown, this function has

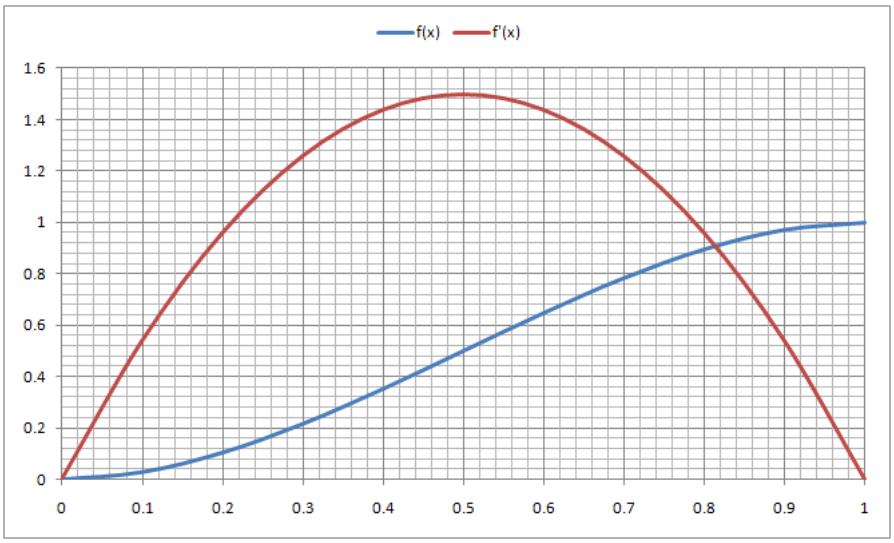


Figure 2-1: Blending Function $f(x) = 3x^2 - 2x^3$

values of zero and one and derivatives of zero at it's endpoints. Having derivatives of zero of the blending function is important as it will ensure at least first derivatives of the grid remain continuous.

Seeing this function, one could easily substitute the value of the function for the ratio of grid location to grid size to blend the entire grid. If the surface moves in the

x-direction, the displacement of grid points in that direction could be

$$\begin{aligned} \vec{X} &= \vec{X}_0 + A\left[3U^2 - 2U^3\right] \\ U &= \frac{x - x_{min}}{x_{max} - x_{min}} \end{aligned} \tag{2.3}$$

Where $\vec{X}$ is the current grid location, $\vec{X}_0$ is the original grid location, $x_{max}$ and $x_{min}$ are the bounds of the grid and $A$ is the current surface amplitude taken from Equation 2.1.

This example showcases how the algebraic method *can* be accurate and easy. However, in real-life cases, displacements are typically more complex. Coming up with algebraic relations for grid locations involving complex displacements and velocities can become daunting.

Moreso, algebraic methods, while are helpful for code debugging, violate a law of coding, which involves user interaction. The developer at all costs wants the user to be involved with computations as little as possible. This leads itself to how algebraic methods can *not* be accurate and easy.

The issue is that these methods involve coding and formulation, both of which will almost certainly lead to unnecessary bugs and errors. Relating to the example given above, if the coder accidentally types

$$\vec{X} = \vec{X}_0 + A\underbrace{\left[3U^3 - 2U^2\right]}_{\substack{\text{Erroneous} \\ \text{exponents}}} \tag{2.4}$$

the the entire grid deformation will fail. The CFD user already has a difficult task in debugging solutions which have either not solved, or not solved to proper accuracy. The last thing he wants is to have to debug grid deformation code as well.

Knowing this leads itself towards automated ways to deform the grid, ways to eliminate the user from making mistakes. One also wants the method of grid deformation

to not only be accurate with nice, clean, smooth grids, but also computationally efficient as well.

### 2.1.2 Transfinite Interpolation

The transfinite interpolation (TFI) method was originally described by Gordon and Hall [22] and later expanded on by Eriksson [15] [16] [17], with particular application to CFD. TFI is typically a grid *generation* scheme which generates grids conforming to specified boundaries. It is easily programmed and is very efficient.

The essence of TFI is the specification of univariate interpolations in each of the computational coordinate directions, forming the tensor products of the interpolations, and finally the Boolean sum [28]. The interpolation functions are a linear combination of user-specified information in the physical domain for given values of the computational coordinate, along with blending functions whose independent variable is the computational coordinate. The reader is directed to Appendix B for further reading regarding general expressions for TFI.

While Transfinite Interpolations is normally used to generate a grid given three pairs of defined opposing boundaries, a variation of these algorithms can be used to adjust an existing grid to three new paris of opposing boundaries. This is where transfinite interpolation has carved a niche in computational aeroelasticity.

Given a grid $\hat{\mathbf{X}}(I,J,K), I = 1,2,\ldots\hat{I}, J = 1,2,\ldots\hat{J}, K = 1,2,\ldots\hat{K}$ and boundary surface grids $\mathbf{X}(1,J,K), \mathbf{X}(\hat{I},J,K), \mathbf{X}(I,1,K), \mathbf{X}(I,\hat{J},K), \mathbf{X}(I,J,1), \mathbf{X}(I,J,\hat{K})$

an adjusted grid $\mathbf{X}(I,J,K)$ can be produced by the following steps

$$\begin{aligned}
\mathbf{X}_1(I,J,K) &= \hat{\mathbf{X}}(I,J,K) \\
+\alpha_1^0(\xi)[\mathbf{X}(1,J,K) - \hat{\mathbf{X}}(1,J,K)] &+ \alpha_2^0(\xi)[\mathbf{X}(\hat{I},J,K) - \hat{\mathbf{X}}(\hat{I},J,K)] \\
\mathbf{X}_2(I,J,K) &= \hat{\mathbf{X}}_1(I,J,K) \\
+\beta_1^0(\eta)[\mathbf{X}(I,1,K) - \hat{\mathbf{X}}(I,1,K)] &+ \beta_2^0(\eta)[\mathbf{X}(I,\hat{J},K) - \hat{\mathbf{X}}(I,\hat{J},K)] \\
\mathbf{X}(I,J,K) &= \hat{\mathbf{X}}_2(I,J,K) \\
+\gamma_1^0(\zeta)[\mathbf{X}(I,J,1) - \hat{\mathbf{X}}(I,J,1)] &+ \gamma_2^0(\zeta)[\mathbf{X}(I,J,\hat{K}) - \hat{\mathbf{X}}(I,J,\hat{K})]
\end{aligned} \tag{2.5}$$

where

$$\begin{aligned}
\alpha_1^0(\xi) &= 1 - u_1(\xi) \\
\alpha_2^0(\xi) &= u_2(\xi) \\
\beta_1^0(\eta) &= 1 - \nu_1(\eta) \\
\beta_2^0(\eta) &= \nu_2(\eta) \\
\gamma_1^0(\zeta) &= 1 - w_1(\zeta) \\
\gamma_2^0(\zeta) &= w_2(\zeta) \\
u_1(\xi) &= \frac{e^{C_1\xi} - 1}{e^{C_1} - 1} \\
u_2(\xi) &= \frac{e^{C_2\xi} - 1}{e^{C_2} - 1} \\
\nu_1(\xi) &= \frac{e^{C_3\xi} - 1}{e^{C_3} - 1} \\
\nu_2(\xi) &= \frac{e^{C_4\xi} - 1}{e^{C_4} - 1} \\
w_1(\xi) &= \frac{e^{C_5\xi} - 1}{e^{C_5} - 1} \\
w_2(\xi) &= \frac{e^{C_6\xi} - 1}{e^{C_6} - 1}
\end{aligned} \tag{2.6}$$

Where the constants $C_1, C_2 \ldots, C_6$ specify how far into the original grid the effect of the six boundary surfaces is carried.

### 2.1.3 Laplacian Method

Another method which is common with implicit time solvers is Laplacian method[58] [7] [39]. This method has even seen use in computer animation[27]. Crumtpon and Giles[11] used a Modified Laplacian described as,

$$\nabla \cdot (k(\vec{x_o} + \delta\vec{x})\nabla\delta\vec{x}) = 0 \tag{2.7}$$

Where $\delta\vec{x}$ and $\vec{x_o}$ are the displacement and the initial grid. Crumpton and Giles suggested that the non-linear diffusion coefficient $k$ should be a constant at each cell $\alpha$, given by,

$$k_\alpha(\vec{x_o} + \delta\vec{x}) = \frac{1}{\max(\text{Vol}(\vec{x_o} + \delta\vec{x}, \alpha), \epsilon} \tag{2.8}$$

The function $Vol(\vec{x}, \alpha)$ returns the volume of cell $\alpha$ of grid $\vec{x}$ while $\epsilon$ is simply a small positive number which prevents the function from become negative.

The simple idea is that relatively small cells will have a large diffusion coefficient, $k$, resulting in small gradients in $\delta\vec{x}$. Therefore, the small cells which are typically placed near the body, where gradients are small, essentially undergo rigid body motion. Since the entire cell essentially moves with the moving body, the cell undergoes small changes in volume and avoids negative volumes which can crash a solution.

Crumpton and Giles chose to discretize Equation 2.7 using a Galerkin finite element approximation on the initial mesh, $\vec{x_0}$.

### 2.1.4 Vertex Springs

For the vertex spring method, first proposed by Batina[4], each segment of the grid is considered to be a spring. The springs are taken to be linear, so Hooke's law governs the force exerted at the nodes. The $j$ number of nodes around node $i$ will exert a force on node equal to,

$$\vec{F_i} = \sum_j \alpha_{ij}(\vec{x_j} - \vec{x_i}) \tag{2.9}$$

Where $\alpha_{ij}$ is the stiffness of the spring between node $i$ and $j$. For the system to be at equilibrium after a mesh movement, the iterative equation needed to be solved is:

$$\vec{x_i}^{k+1} = \frac{\sum_j \alpha_{ij}\vec{x_j^k}}{\sum_j \alpha_{ij}} \tag{2.10}$$

At every iteration, the new nodal position $\vec{x_t}$ is calculated as a weighted average of the surrounding nodes. This gives the linear system,

$$[A]\{x\} = \{b\} \tag{2.11}$$

Where the $[A]$ matrix is formed by the spring stiffnesses $\alpha_{ij}$, the vector $\{x\}$ contains the mesh positions, and the right hand side vector contains the non-homogeneous terms, which are functions of the boundary conditions.

This method produces very nice meshes, however computational time can be excessive. Imagine a one-dimensional problem. One can reason that surfaces of interest of a domain are of order one less than the domain. That is, lines make up boundaries of two-dimensional problems, and points make up boundaries of one-dimensional problems.

If the number of grid points is ten, then there are nine vertex springs connecting them, as shown in Figure 2-2.

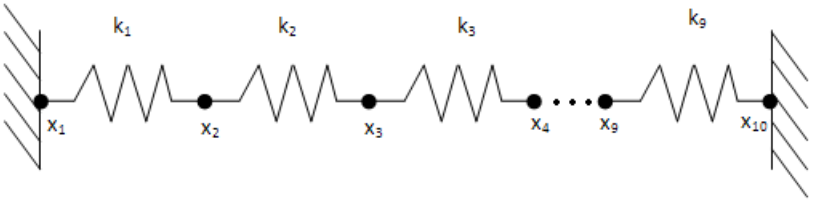


Figure 2-2: Example one-dimensional vertex spring domain

Using this example and Equation2.11 gives the stiffness matrix equation

$$[A] = \begin{bmatrix} k_1 & -k_1 & & & & & \\ -k_1 & k_1 - k_2 & -k_2 & & & & \\ & -k_2 & k_3 - k_2 & -k_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -k_9 & k_{10} - k_9 & -k_{10} \end{bmatrix} \tag{2.12}$$

This equation is a simple tridiagonal and can be solved using the famous Thomas algorithm, which is described in Appendix C. While this matrix inversion is simple, when the domain is expanded to two-dimension, the matrix size not only increases exponentially, but becomes a banded tridiagonal, which is significantly much more computationally expensive to solve.

Adding these effects, one can easily see how for interesting and complex problems, the grid deformation can quickly take on the same computation time as the flow

solver.

## 2.2 Moving Wall Boundary Conditions

Many problems arise when attempting to solve a fully coupled fluid-structure interaction problem. Several of those problems have already been discussed including how the wall or bodies deform and subsequently what to do with the grid after they move.

A final problem presents itself in the form of boundary conditions. Thompson [59] [60] did early work towards Time-Dependant Boundary Condition for Hyperbolic systems, and since then, a plethora of different systems have been proposed. Several robust boundary conditions have been tested [25] including Thompson, and Giles [20] [21], which proved to be very useful.

However, with the additional complexity of moving boundaries, new wall boundary conditions needed to be developed. The formulation and implentation of the moving wall boundary conditions will follow the "correction" method of Hixon, et. al. [24]

For an invisid wall, to avoid nonphysical flow through the wall, one first sets the normal momentum to the wall to zero. By defining the vector normal to the wall $\vec{n}$, the boundary condition becomes:

$$\frac{\rho\vec{V}}{J} \cdot \vec{n} = n_t \left(\frac{\rho}{J}\right) + n_x \left(\frac{\rho u}{J}\right) + n_y \left(\frac{\rho v}{J}\right) + n_z \left(\frac{\rho w}{J}\right) = 0 \tag{2.13}$$

Since a wall boundary follows a grid line, the contravariant vector corresponding to the constant coordinate will always be normal to the plane. This allows the boundary condition to be written as:

$$n_t \left(\frac{\rho}{J}\right) + n_x \left(\frac{\rho u}{J}\right) + n_y \left(\frac{\rho v}{J}\right) + n_z \left(\frac{\rho w}{J}\right) = 0 \tag{2.14}$$

To use the correction method, one first computes the flow at the wall using no boundary conditions. That flow is then used to initially update the flow variables at the wall in the Runge-Kutta algorithm.

$$\left(\frac{\partial}{\partial t}(\vec{\tilde{Q}})\right)^s_{noBC} = -\left(\frac{\partial}{\partial \xi}(\vec{\tilde{E}}) + \frac{\partial}{\partial \eta}(\vec{\tilde{F}}) + \frac{\partial}{\partial \zeta}(\vec{\tilde{G}})\right)^s \tag{2.15}$$

$$\vec{\tilde{Q}}^s_{RK,noBC} = \vec{\tilde{Q}}^{s-1}_{RK,noBC} + \alpha^s \delta t (\frac{\partial \vec{\tilde{Q}}}{\partial t})^s_{noBC} \tag{2.16}$$

$$\vec{\tilde{Q}}^{s+1}_{noBC} = \vec{\tilde{Q}}^s_{noBC} + \beta^s \vec{\tilde{Q}}^s_{RK,noBC} \tag{2.17}$$

Which can be simplified as:

$$\vec{\tilde{Q}}^s_{RK,BC} = \vec{\tilde{Q}}^s_{RK,noBC} + \alpha^s \delta t \left(\delta \frac{\partial \vec{\tilde{Q}}}{\partial t}\right)^s \tag{2.18}$$

$$\vec{\tilde{Q}}^{s+1}_{BC} = \vec{\tilde{Q}}^{s+1}_{noBC} + \beta^s \alpha^s \delta t \left(\delta \frac{\partial \vec{\tilde{Q}}}{\partial t}\right)^s \tag{2.19}$$

In short, given the uncorrected flow "through" the wall at the next stage, the desired condition can be obtained. The goal of this boundary condition is to set the normal momentum at the wall to zero, without changing the momentum tangent to the wall.

To accomplish this, an orthogonal set of unit vectors normal and tanget to the wall are defined.

$$\begin{aligned} \vec{\xi}_1 &= \vec{\xi} \\ \vec{\xi}_2 &= \vec{\eta} \\ \vec{\xi}_3 &= \vec{\zeta} \\ \vec{\xi}_4 &= \vec{\tau} \end{aligned} \tag{2.20}$$

Since wall boundaries lie along grid lines, the boundary can be defined by a constant $\xi_i$ value, and the unit vector normal to the wall at the next stage is then:

$$\vec{n}^{s+1} = \frac{\vec{xi}_i^{s+1}}{|\vec{xi}_i^{s+1}|} \tag{2.21}$$

Realizing that the grid metrics at the next stage must be used to define the wall normal, the normal momentum through the wall at the next stage is written as:

$$\left(\frac{\rho\vec{V}}{J}\right)^{s+1} \cdot \vec{n}^{s+1} = \left((n_t\frac{\rho}{J})^{s+1} + (n_x\frac{\rho u}{J})^{s+1} + (n_y\frac{\rho v}{J})^{s+1} + (n_z\frac{\rho w}{J})^{s+1}\right) \tag{2.22}$$

The correction part of the boundary condition then becomes:

$$\begin{aligned}
\delta\left(\frac{\rho}{J}\right) &= -(n_t)^{s+1}\left(\frac{\rho}{J}\right)^{s+1} \\
\delta\left(\frac{\rho u}{J}\right) &= -(n_x)^{s+1}\left(\frac{\rho u}{J}\right)^{s+1} \\
\delta\left(\frac{\rho v}{J}\right) &= -(n_y)^{s+1}\left(\frac{\rho v}{J}\right)^{s+1} \\
\delta\left(\frac{\rho w}{J}\right) &= -(n_z)^{s+1}\left(\frac{\rho w}{J}\right)^{s+1}
\end{aligned} \tag{2.23}$$

For the case of the invisid wall, there are no corrections to the energy equation. These corrections in Equation 2.23 can be used to solve for the neccessary correction to the time derivative at the previous stage.

# Chapter 3

# Grid Motion

The code used is the Air Force Research Laboratory's (AFRL) Grid Warp program. Many of the figures and equations presented here are from Melville's work[36], and more recent work can be directed to Allen[1].

Unlike the approaches cited earlier, one was desired that required no connectivity knowledge. Essentially, each point needs to move independently of the other points around it. This would allow the code to work on both structured and unstructured grids efficiently. To do this, a response function is created which is defined by the displacements of the moving surfaces. This in turn directs the deformation for the individual point.

## 3.1 Surface Projection Point Construction

The first step in the process is for each point in the volume mesh to choose an appropriate target point on the surface(s). The obvious choice is the point on the surface that is closest to the mesh point. Of course in the case of multiple moving surfaces, a point on each surface will need to be found.

Since maintaining grid quality is vital in obtaining favorable results, care was taken to not only use surface points as targets, but surface locations. This means

that the target will typically fall in between surface points, and also that each mesh point will move independently.

After the target surface point $N$ is found, the second step is to identify which edge adjacent to the to node $N$ is most aligned with the projection of the mesh point onto the surface. This of course is done by taking the dot product of the surface node to mesh point ray $\overrightarrow{NP}$ with the unit vector surface node to surface node neighbor ray $\overrightarrow{NR_i}$. Figure 3-1 shows the projection.

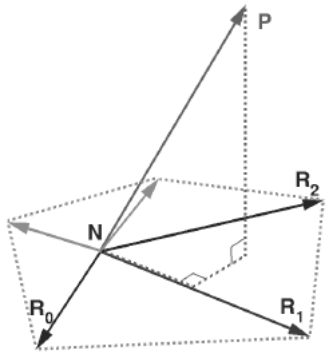


Figure 3-1: Target Point Projection onto the Surface

Then the maximum dot product value is taken. Next, the dot products are compared for the target ray, minus the component along the primary edge, and the unit vector of each neighboring ray.

$$f' = \left( \overrightarrow{NP} - \frac{\overrightarrow{NP} \cdot \overrightarrow{NR_i}}{|\overrightarrow{NR_i}|} \right) \cdot \frac{\overrightarrow{NR_j}}{|\overrightarrow{NR_j}|} \tag{3.1}$$

It should be noted here that a structured grid has not been assumed. In the case of an unstructured grid, the number of compared rays could be quite high. For a structured grid, this number will be 3. The larger of 3.1 will indicate which neighbor is closer. At this point, a triangular surface cell is created, regardless of the type of grid.

Then, the reference point is defined as a weighted sum of the triangle nodes. These

weights are kept and used similarly when calculating the deflection.

## 3.2 Distribution of Surface Attention

Once a reference point is chosen for each moving surface, it is necessary to address multiple surfaces. The ability to have multiple moving surfaces is paramount in many complex test cases, and when developing the method, much consideration was placed into this.

A Surface Attention Function (SAF) is generated which will determine how much effect each surface will have on a given mesh point. There are three requirements which are needed to ensure that the quality of grid remains.

- If a mesh point is very near a reference point, then that surface should have its full attention.
- The variation should be smooth to ensure grid metrics stay continuous.
- There is some distance at which a surface is eliminated from giving any attention to a mesh point.

From these requirements, the SAF used is:

$$\mathrm{SAF}_i = \frac{\mathrm{MAX}([\frac{1}{D_i} - \frac{1}{D_{cut}}], 0)}{(\sum_j [\frac{1}{D_j} - \frac{1}{D_{cut}}])} \tag{3.2}$$

Where i is one of j surfaces, $D_j$ is the distance from the mesh point to the surface reference point, and $D_{cut}$ is the cut-off value which controls the behavior of the function.

Figure 3-2 shows an example of how this works. Points near the bottom surface A receive its full attention, and vice versa. Points close to both receive attention from both, while the line equidistant from both receives equal attention from both.

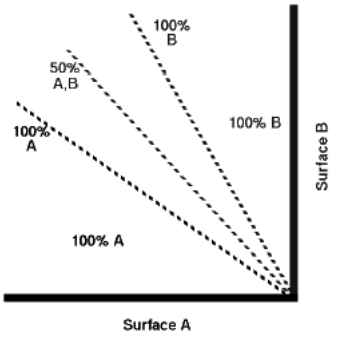


Figure 3-2: Surface Attention Function Schematic

## 3.3 Deformation Response

In the final step, the mesh point is deformed according to the surface displacement of the reference points of each of its corresponding surfaces. The point is displaced by each surface according to:

$$\Delta P_i = \Delta X_i + [R_i](P - X_i) + X_i \tag{3.3}$$

Where P is the initial grid location, $\Delta X_i$ and $R_i$ are the translational vector and rotational matrix associated with the reference point $X_i$ of the current surface $i$.

The rotational matrix helps avoid shearing problems that can occur when there are large rotational movements in the surfaces. The rotational matrix should be calculated to maintain the highest grid quality, but can be turned off for small or linear movements.

Similar requirements were used to determine an Influence Decay Function (IDF) as were used to determine the SAF.

- There must be no decay at the moving surface to ensure no grid overlap.
- There must be some distance away in which the influence from the surface becomes zero.

- In between, the decay must be smooth to ensure grid metrics stay continuous.

Based on these, cubic blending was adopted, and the IDF becomes:

$$\begin{aligned} \hat{D} < 0, \;\; & IDF = 1 \\ 0 < \hat{D} < 1, \;\; & IDF = 1 - 3\hat{D}^2 + 2\hat{D}^3 \\ \hat{D} > 1, \;\; & IDF = 0 \end{aligned} \tag{3.4}$$

$$\hat{D} = \frac{D - D_{near}}{D_{far} - D_{near}} \tag{3.5}$$

Where D is the distance from the mesh point to the surface reference point and the two parameters control the range of influence decay.

Finally, the IDF and SAF effects are combined over all i surfaces to form the final displacement and the new location of point P, P'.

$$P' = P + \sum_i SAF_i IDF_i (\dot{P}_i - P) \tag{3.6}$$

## 3.4 Improving on Grid Motion Efficiency

Much computational time can be lost and gained in the search for the closest surface point. In the original Melville paper, a global search was used. For the global search, each mesh point was individually compared to each point on the surface. Then, the local values of the large array were found.

It is apparent that this could easily be a bottleneck. Initial test cases were ran using a cube of 20 grid points on each side. The deforming surfaces were two opposite faces. Each grid point looked at all 800 surface points before finding the closest. Multiply that by 8000 grid points, and the total number of distance calculations

becomes $20^6$. This is a sufficiently large enough number of computations for a simple cube, let along a complex multi-block aeroelastic problem.

Another method was developed, which while not as robust and infallible as the global search, it was found to have significant increases in speed. The method will be referred to as a gradient walker. The idea is simple: take a point on the surface and calculate its distance. Now calculate the distance to each of its neighbors. If the smallest distance is the center, then you've found the target surface point. If not, then go to the point of smallest distance and try again.

However, a problem with this approach can arise if the surfaces are irregular. If there are many local minimums, there is no way to go from one local minimum to another. One can think of a skiier. If a skiier is trapped in a hole, there is no way for them to get out, they can only go down.

An idea was developed to counter this. Take the corners of the current block and use a global search to calculate their surface targets. Then, use all of those targets as the starting point for each of mesh points. This will limit the chances of falling into a local minimum.

Possibly the largest increase in computation time was the easiest. The problem was simply a square root. In the global search, the *actual* distance was being calculated for each surface point. The equation for this was:

$$D = \sqrt{\left(\vec{X_1} - \vec{X_2}\right) \cdot \left(\vec{X_1} - \vec{X_2}\right)} \tag{3.7}$$

Where $\vec{X}$ was the vectorized locations of the mesh and surface point. However, it stands to reason that if if two points share the smallest distance between each other, then they also share the smallest distance squared.

Calculating a square root is not something that is easily accomplished by a computer. It cannot simply add bits of information to get the result. One common way for computers to calculate roots is to compute the exponential function or the natural

logarithm. The square root can then be found by the identity:

$$\sqrt{x} = e^{\frac{1}{2}lnx}\sqrt{x} = 10^{\frac{1}{2}logx} \tag{3.8}$$

It can safely be assumed that this operator takes considerably longer than other arithmetic operators to compute integer numbers, *let alone* floating point real, or even double precision numbers.

The code was adjusted so that only the distance squared was calculated for each surface point. Then, when that smallest value was found, it took the square root. This reduced the number of square root calls in the cube problem from $20^6$ down to only 8000. The reduction in computation time was astounding and will be presented in the results section of this report.

# Chapter 4

# Initial Grid Tests

The first tests to be completed were not full flow solutions, but tests of the grid deformation itself. In the first test, a cube with sides of 10 was meshed orthogonally with equal grid spacing; 41 grid points a side.

For the deformation, a simple gaussian was applied to one side, with a magnitude of 2.0. For these initial tests, $D_{near}$ was set to 0.0. Figures 4-1 through 4-4 show the effect of changing the $D_{far}$ parameter.

The effect of the rotation matrix is clearly evident as the grid lines shown in the figures show a tendency for gridlines to remain normal to the surface. The influence decay function (IDF) is clearly evident as the $D_{far}$ increases.

The next test will check the effectiveness of the surface attention function (SAF), by deforming two surfaces. The same deformation from the first test will be applied to one side, while the same deformation will be applied to the opposite side.

This is an interesting result. With a far blending distance of 5, each surface "controls" exactly half of the domain. As this distance increases, grid points in the middle start having attentions from both surfaces.

Removing the rotation matrix essentially only allows the grid point motion in the same direction of the surface movement. That is, the grid lines will no longer

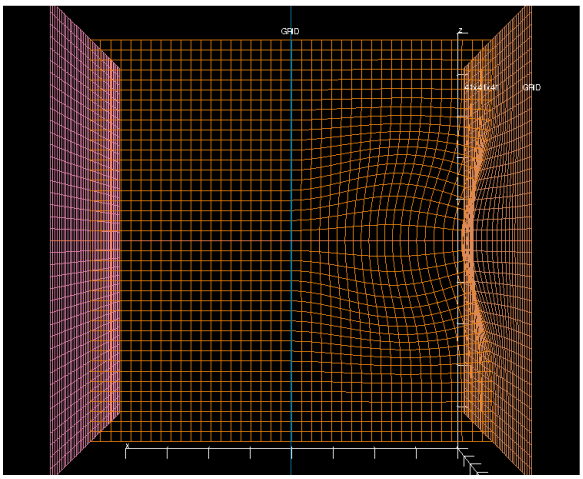


Figure 4-1: Initial AFRL GridWarp test, $D_{far} = 4.0$

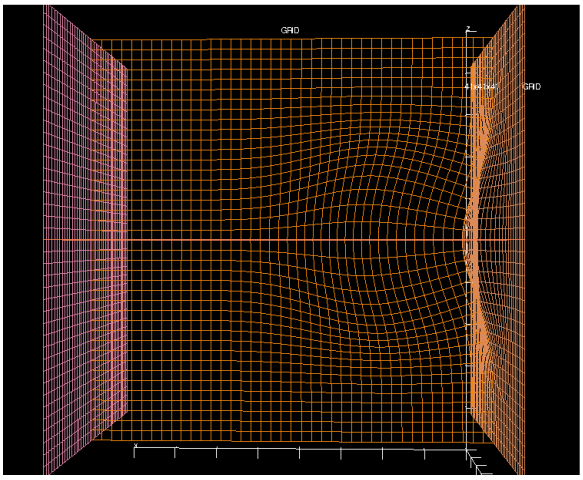


Figure 4-2: Initial AFRL GridWarp test $D_{far} = 6.0$

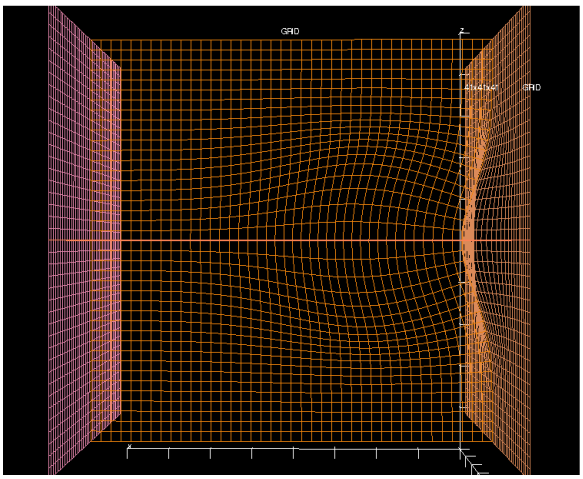


Figure 4-3: Initial AFRL GridWarp test $D_{far} = 8.0$

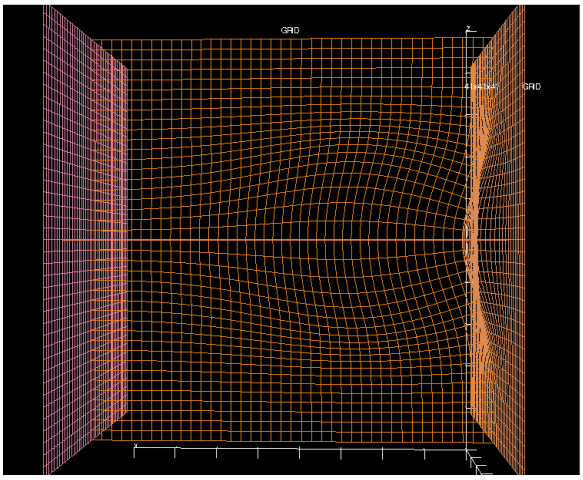


Figure 4-4: Initial AFRL GridWarp test $D_{far} = 10.0$

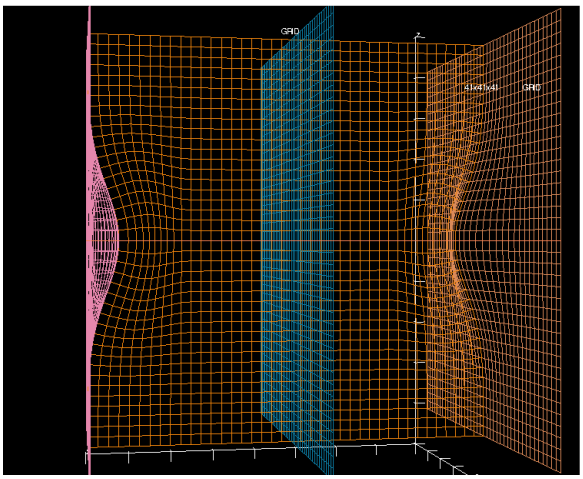

Figure 4-5: Multiple Surface GridWarp test $D_{far} = 2.0$

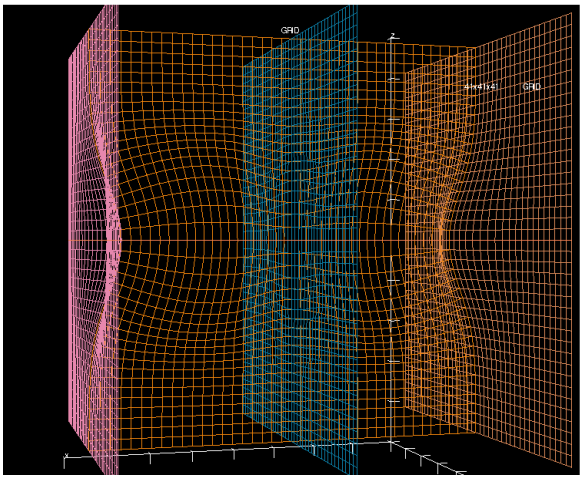

Figure 4-6: Multiple Surface GridWarp test $D_{far} = 5.0$

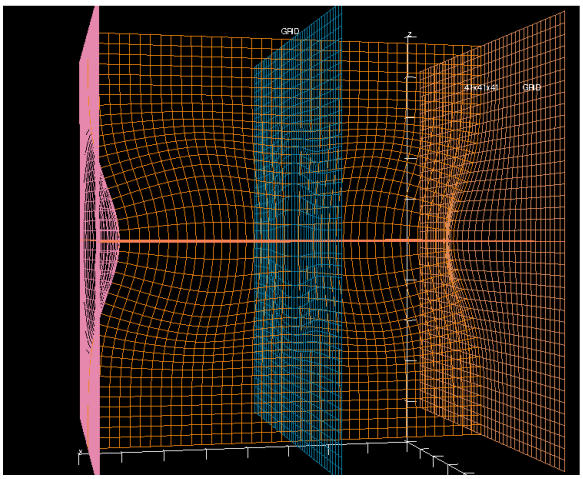


Figure 4-7: Multiple Surface GridWarp test $D_{far} = 6.0$

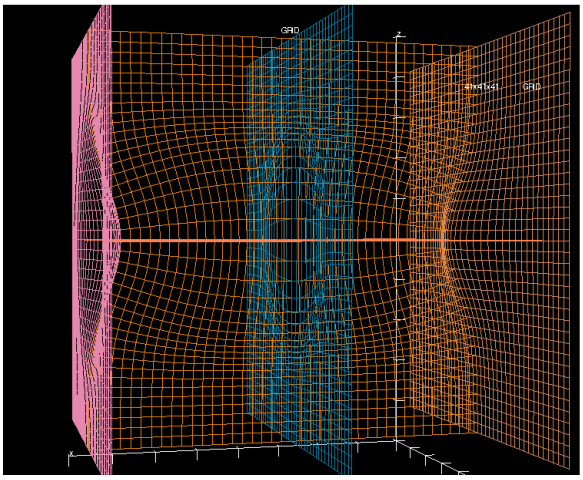


Figure 4-8: Multiple Surface GridWarp test $D_{far} = 8.0$

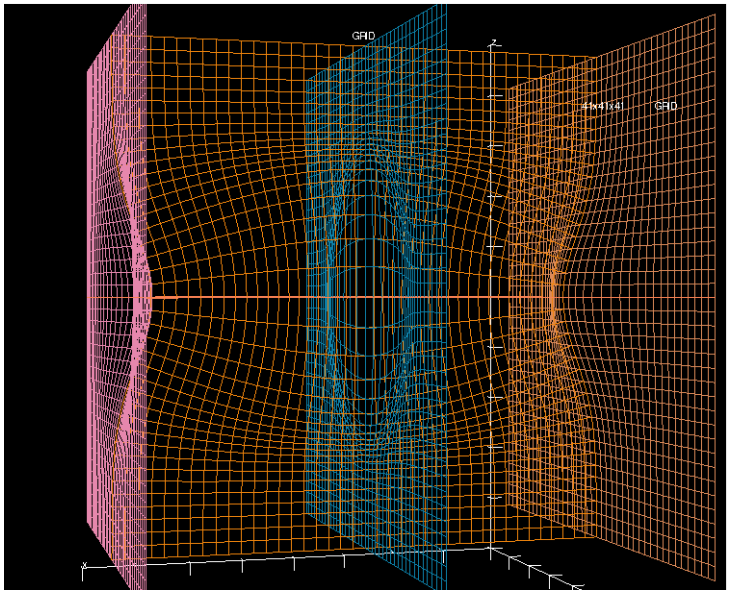


Figure 4-9: Multiple Surface GridWarp test $D_{far} = 10.0$

attempt to remain orthogonal to the surfaces. While critical for problems involving large rotations, a matrix inversion is necessary for the calculation and can be turned off for translatory problems only. This effect is shown in Figure 4-10.

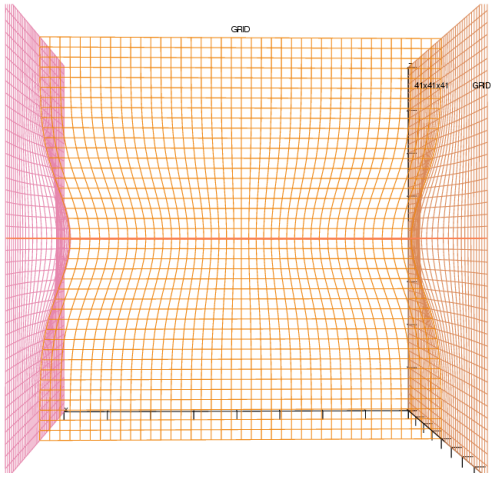


Figure 4-10: Multiple Surface GridWarp test, Rotation Matrix Off

Lastly, to further test the surface attention, different near and far blending distances were defined for each surface. The effect should be the center inflection point shifting towards side of smaller blending. This effect is shown in Figure 4-11

The results shown in this section suggest that GridWarp is robust and accurate enough to carry on and integrate into the BASS Code. The flexibility exists in the code to define the grid motion however the user desires. The grid motions and deformed grids were calculated swiftly and most importantly, the grid remained smooth and continuous after deformation.

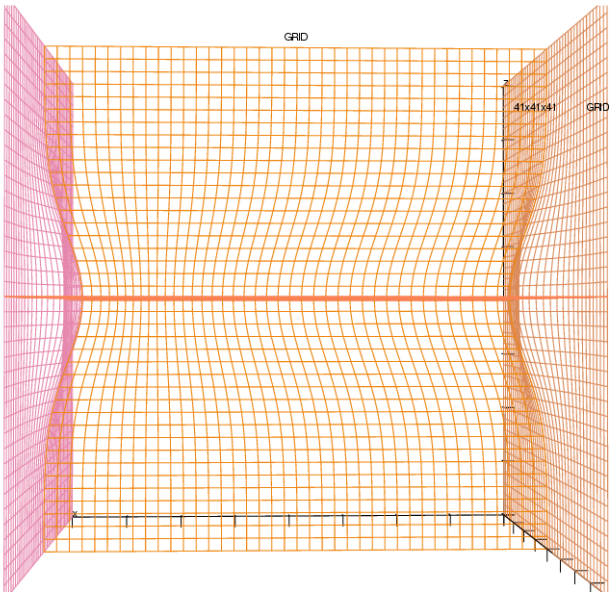


Figure 4-11: Multiple Surface GridWarp test, Different Surface Attentions

# Chapter 5

# Vibrating Membrane

In the first test, the domain for the grid tests in Chapter 4 will be used. Reiterating, the grid for is 41 points on each edge, with a length of 10 on each side.

Since an actual flow needs to be calculated, the deformed surface needs to be small enough to avoid causing numerical issues. As such, the displacement in the previous tests are reduced by an order of magnitude, down to 0.2. Aside from the single moving wall, all of the other walls were left as out-flow boundaries. Initial non-dimensional pressure and density values were 0.711 and 0.978. Two oscillations were calculated, with a period of 0.1 seconds. Using the definition for the speed of sound in an ideal gas:

$$c = \sqrt{\gamma \cdot \frac{p}{\rho}} \tag{5.1}$$

We obtain a speed of sound for this test problem as 1.0. We know that the acoustic waves travel at a speed of $\bar{u} + c$ and the entropy waves travel at $\bar{u}$. Since there is no free-stream velocity, we expect the acoustic waves to travel at the speed of sound, or the wave should propagate a distance of 1 every unit of time.

The purpose of this test is only to ensure that the BASS Code is receiving the proper inputs from the grid deformation. As you can see in Figures 5-1 through 5-3

the wave front is quite close to where it is expected to be.

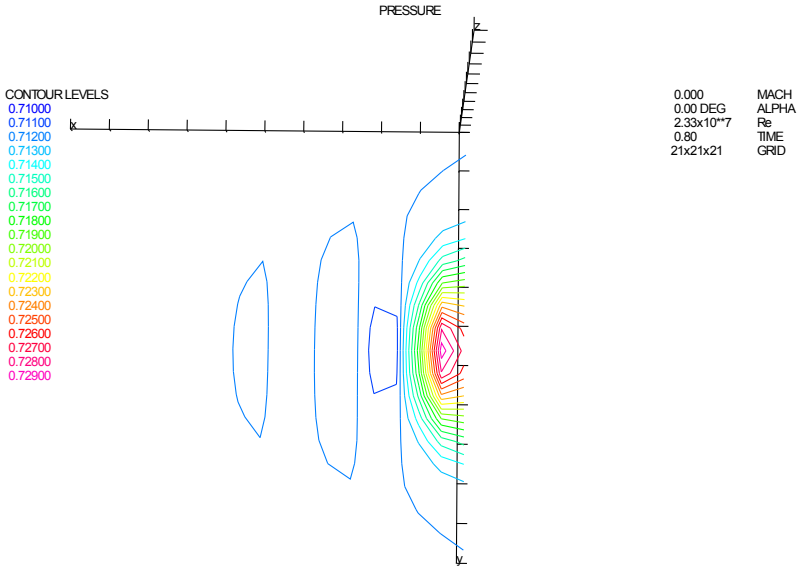


Figure 5-1: Vibrating Membrane Test at 0.8 seconds

At this point, the integration of the solvers has been completed, and the final test case can be evaluated.

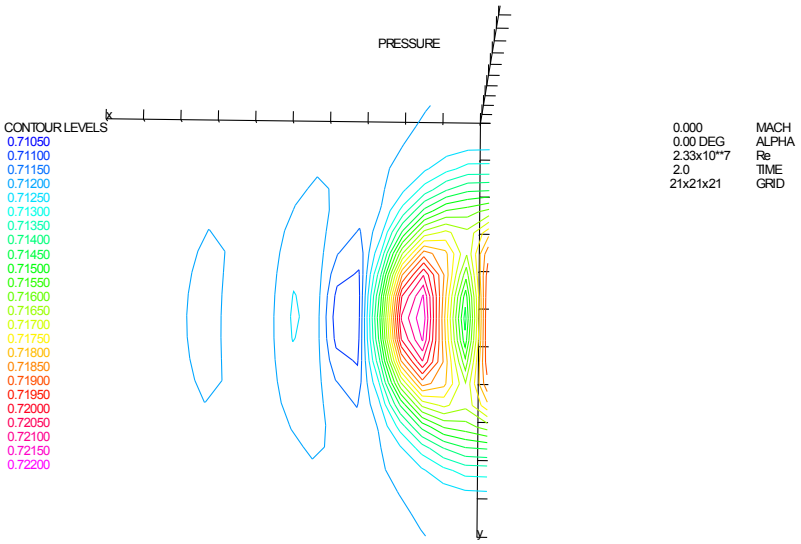


Figure 5-2: Vibrating Membrane Test at 2.0 seconds

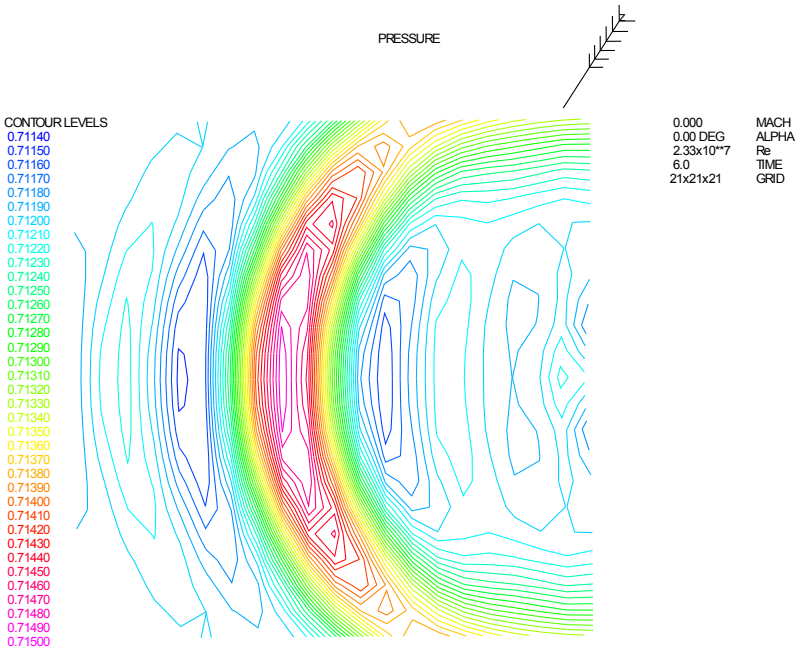


Figure 5-3: Vibrating Membrane Test at 6.0 seconds

# Chapter 6

# Plunging Piston

## 6.1 Introduction

Sound radiation from a vibrating piston mounted in an infinite baffle is a classical problem which has been covered extensively [30] [44] [18]. This problem is relavent to engineering problems as it serves as an introduction to the sound radiation from different types of surfaces such as loudspeakers, open ends of pipes, etc.

Let's consider a circular piston of radius $z$, mounted in an infinite rigid baffle. The noise radiated by this oscillating piston can be modeled in terms of numerous monopoles radiating together. Each monopole is radiating from a rigid, fully-reflecting plane, *not* from free space. Therefore, the sound pressure due to any one of the baffled monopole is twice that of an equivalent monopole in free space.[40].

$$p'(r,t) = \frac{ik\rho_0 c}{2\pi r} Q_p e^{i(\omega t - kr)} \tag{6.1}$$

In this equation, $Q_p$ represents the source strength of the monopole on the surface and is equal to $U_p \delta S$ where $U_p$ is the peak surface velocity of the monopole and $\delta S$ is an elemental surface area. We then integrate over the whole surface to get the

resultant pressure fluctuation due to all the monopoles vibrating in phase.

$$p'(r,\theta,t) = \frac{ik\rho_0 c\pi z^2 U_p e^{i(\omega t - kr)}}{2\pi r}\left[\frac{2J_1(kz\sin\theta)}{kz\sin\theta}\right] \tag{6.2}$$

We can arrive at this same conclusion by representing the isolated vibrating body by the Kirchoff-Helmholtz integral formula [13] [42]. The boundary value problem given a non-zero $\nu(x,y,t)$ specificed somewhere on the $z = 0$ plane is that of the radiation from a thin disk of time-varying thickness. Taking symmetry into account the net contribution from surface pressure to the Kirchoff-Helmholtz integral is zero. The integrals over the surface-normal velocity from the front and back surfaces are equal, so the integral becomes:

$$p'(x,t) = \frac{\rho}{2\pi}\int_S \frac{\nu(x_s,y_s,t-R/c)}{R}\,dx_s dy_s \tag{6.3}$$

A suitable approximation for characteristic far-field is realized reducing Equation 6.3 to the form of an outgoing spherical wave with non-uniform directivity:

$$p' = f(\theta,\phi)r^{-1}e^{ikr} \tag{6.4}$$

where

$$f(\theta,\phi) = \frac{-i\omega\rho}{2\pi}\int_S \nu(x_s,y_s)e^{-kx_s\cdot e_r}\,dx_s dy_s \tag{6.5}$$

or

$$f(\theta,\psi) = \frac{-i\omega\rho}{2\pi}g(k\sin\theta\cos\phi, k\sin\theta\cos\phi) \tag{6.6}$$

For a circular piston where the velocity is constant along the entire piston, Equa-

tion 6.6 reduces to:

$$f(\theta) = \frac{-i\omega\rho\nu}{k^2 \sin^2\theta} \int_0^{kz\theta} J_0(\eta)\eta \, d\eta = -i\frac{\rho c \nu k a^2}{2} \frac{2J_1(ka\sin\theta)}{ka\sin\theta} \tag{6.7}$$

Combining Equations 6.6 and 6.4 will give the same result obtained before.

The pressure distribution used for post-processing will be the mean pressure, $\bar{p}$ plus the pressure fluctuation.

$$p = \bar{p} + p' \tag{6.8}$$

One quick note is that during the itegration, the functions dependence on azimuthal angle $\phi$ was dropped leaving the pressure fluctuation a function of distance from the piston and axial angle only.

Looking at the analytical solution, there are essentially two parts. There is a time-harmonic constant, that is a function of defined constants, and a Bessel function. The Bessel function is shown in Figure 6-1. As shown, this term controls the directivity of

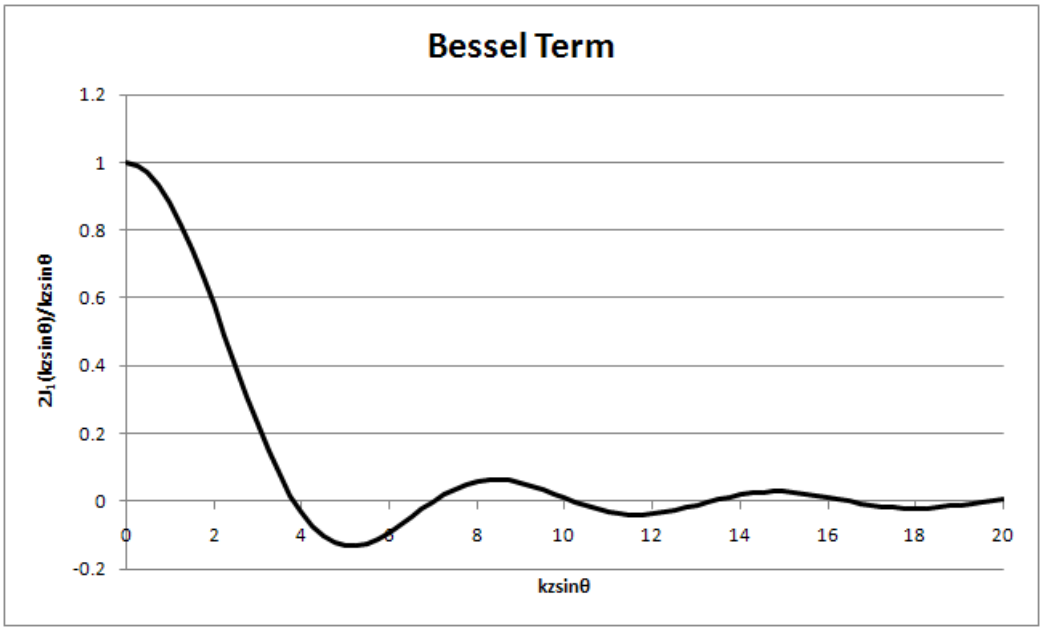


Figure 6-1: Analytic Bessel Term

the wave. At low frequencies, variation in the azimuthal direction does not vary the strength of the wave significantly. However, as one increases the frequency, or more accurately, the frequency times the piston radius, changes in direction can significantly alter the strength of the wave. There is no more apparent than at a $kz \sin\theta$ value of approximately 3.8, when a "lobe" appears. At these lobes, essentially there no change in the pressure field. For higher piston frequencies, the radiation pattern will become more and more complex, while at low piston frequencies, the radiation should be mainly spherical. Testing over a range of frequencies should illustrate this well.

As simplistic as this problem seems, the first NASA Workshop for Benchmark Problems found it difficult enough to warrant placement in the benchmark problems for Computational Aeroacoustics.

The formulation in the First Workshop is as follows. Consider a piston with center at r = 0 and extends to a radius of 10. The domain extends to 100 in both the r and z direction.

The piston will have its motion described as:

$$z = A \sin\left(\frac{\pi}{5} t\right) \tag{6.9}$$

Where A is a small amplitude number which will be $10^{-4}$ for this case.

The benchmark problem instructs to solve the problem as 2D axisymmetric, however more will be learned by solving this problem in full three-dimensions. Aside from solving this problem in three-dimensions, a polar grid will not be used. By not using a polar grid, a heavier burden is placed on the accuracy of the solver. Clever grid generation and topology can hide bugs and other erroneous conditions, by solving this problem as general as possible, it will create confidence in future application of the code.

## 6.2 Procedure

The procedure for the various piston tests will actually be somewhat simple. The first step as with any numerical analysis is to generate a mesh. Since the resolution of propagating waves is controlled by mesh spacing, a limit can be placed on the frequency range. High-order DRP schemes can properly propagate a wave with as little as 6 points per wavelength. However, at that low points per wavelength, numeric dissipation can damp issues or other problems that would otherwise need to be solved. Because of this, a minimum of 10 points per wavelength will used to determine the maximum frequency of tests that will be ran.

The next step is to create the wall defintion file that controls the surface movement. To maintain continuous derivatives of the grid in time, the displacements are ramped up to the full displacements over the course of one-quarter of a period. This ensures that all grid-motion derivatives stay continuous and spurious waves will be kept to a minimum.

Since the radius of the piston remains constant over the range of tests, the same grid can be employed, and the only difference between the tests is the wall definition file. This allows the user to create an array of wall defintion files and use batch scripting to repeat the runs, changing the wall definition file as needed.

The test are ran to convergence by comparing flow files at the start of consecutive cycles. The time-marching scheme is the RK67, which is a RK2N-type Runge-Kutta algorithm. The ratio of specific heats is the standard 1.4, while the mean density is non-dimensionalized to 1.0, and the pressure to $1/\gamma$ resulting in a speed of sound of 1.0. Since the speed of sound is 1.0, both wavelength and period are equivalent.

## 6.3 Mesh Generation

The mesh was created using GridPro [14], which has the ability to create the face-matched structured grid that the BASS Code requires. The topology chosen was a spherical grid. Since waves propagate primarily spherical in this test case, the topology choice is trivial.

In addition to the grid lines following wave propagation, a spherical grid allows the user to insert interior surfaces which follow radii. This allows for easy post processing when it becomes apparent to the user that he wants data at a particular radius.

Careful consideration was taken when creating the grid topology. In order to resolve the piston radius, seperate blocks were placed directly over the piston edge. This allowed the number of points to reasonably be defined across the piston edge. This would prove to be important.

Along with proper edge definition, the spherical topology allowed a radius of interest to be defined. In this case, that radius was chosen to be 150. From radius 150 to 200, the grid is stretched which should damp the remaining waves, assisting the outflow boundary conditions. The final mesh consists of 104 blocks and a total of 327,184 grid points. Images of the grid are shown below in Figures 6-2 through 6-5

## 6.4 Grid Performance

Before any flow results are examined, the first thing that should be examined is how the grid is deforming; both in space, and in time.

The first plot that is shown is four snapshots of the grid over the course of a typical plunging cycle. Figure 6-6 shows the fully deformed grid on the left side, the right side showing the grid near half cycles when it becomes undeformed.

This test shows the maximum piston and therefore grid displacement. In an effort to keep the maximum piston velocity constant between the tests, the dis-

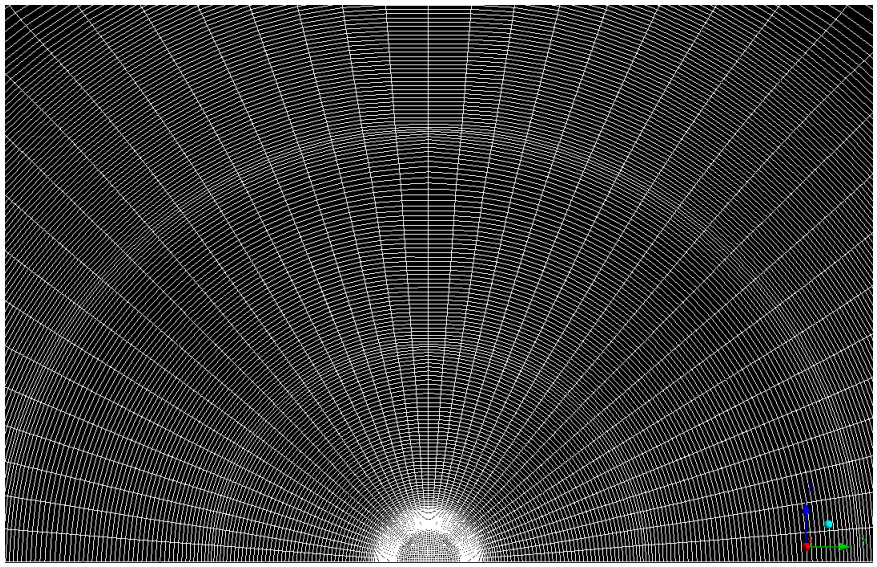

Figure 6-2: YZ (+X) Slice Farfield Grid

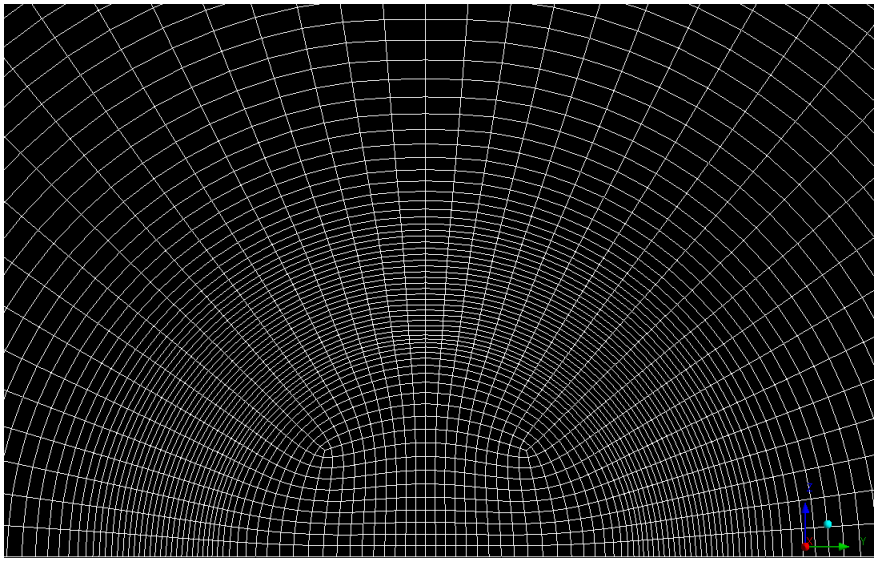

Figure 6-3: YZ (+X) Slice Nearfield Grid

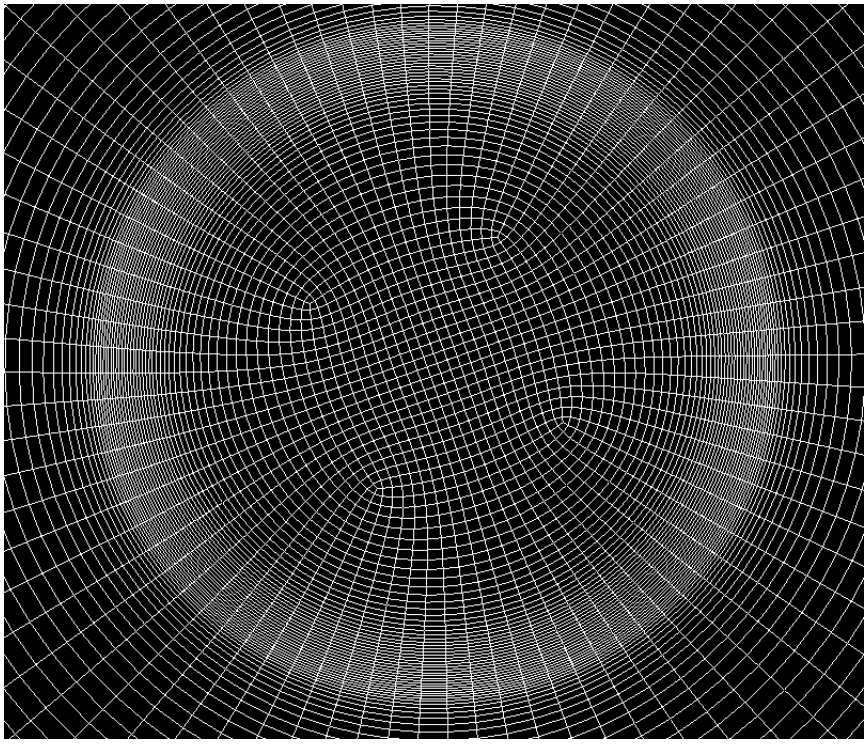

Figure 6-4: XY (+Z) Slice Showing Piston Edge Grid Clustering

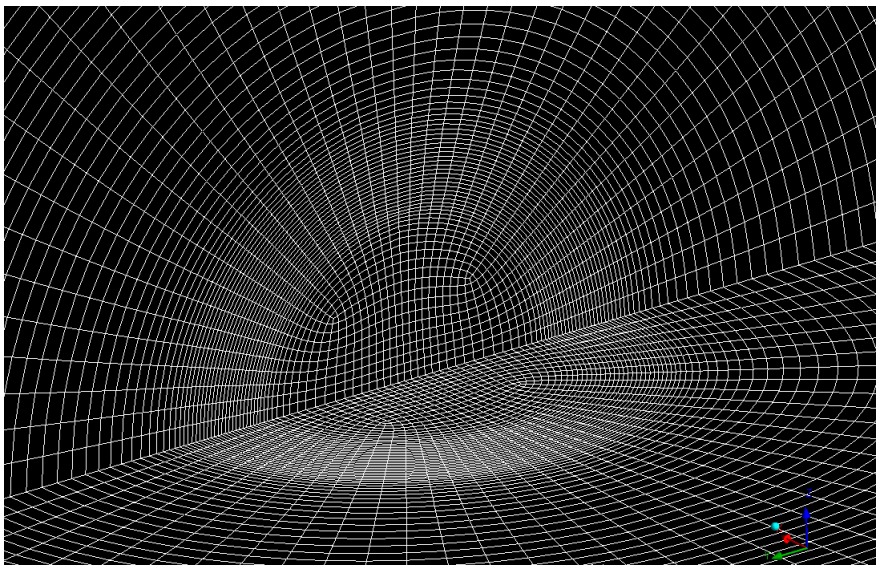

Figure 6-5: Isometric View of Grid Showing Topology

placement varies with the frequency. In this maximum case, the displacement is $\epsilon = U_p/(2\pi/T) \approx 0.004$. Note here that the displacements have been scaled by 10,000 for viewing.

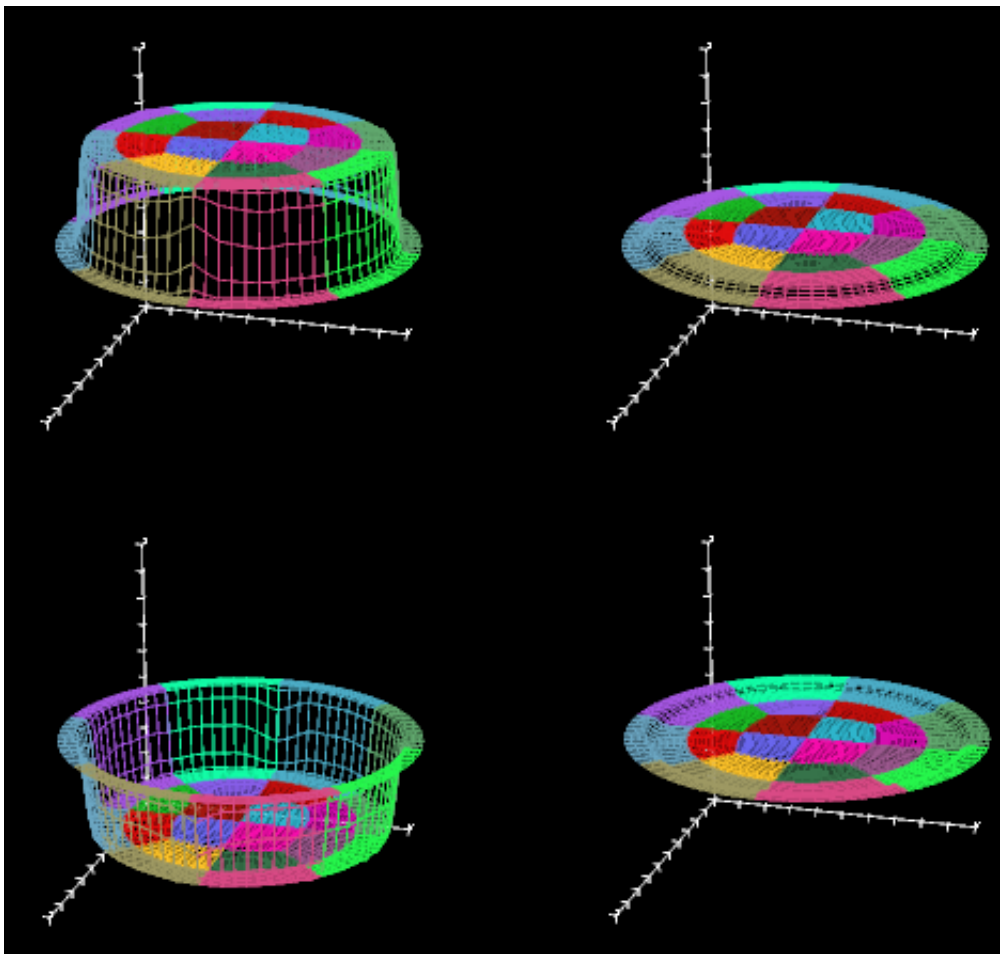

Figure 6-6: Deformed Grid Over a Complete Cycle

It is important that the grid can become "undeformed" and return back to its original shape, particularly in a cyclic problem. If residual displacement begins to add up, grid issues could tend to arise as the problem runs to convergence. To ensure this, the user needs to properly define the GridWarp parameters. By setting a sufficient blending region and far distance, one can ensure that effects like this don't occur.

Next, the piston motion is examined. In order for the test to be a success, the

piston must be well defined over the cycle. In addition to the steady-state motion, the piston is also ramped up at the start of the run.

Sudden jerks of wall boundaries can wreak havoc on a high-order code such as the BASS Code, so to alleviate this, the multi-purpose function described in Equation 2.2 is used to ramp the displacement.

The center of the piston at $(0, 0, z)$ is at the intersection of four grid blocks. That point is tracked over time and the result is shown in Figure 6-7. Since the piston starts


Figure 6-7: Center of Piston at the Start of a Run

without a sudden jerk, large spurious waves at the start of the run are minimzed. This helps achieve convergence faster as those waves can cause issues at boundaries among other problems.

The next thing that will be examined is the quality of the edge blend. Similar to the previous point, the edge of the piston should smoothly blend the full displacement of the piston down to zero displacement simulating the rigid baffle. In addition to

this, the blending should remain smooth and unchanged over time.

To test this, the grid's z-location data was pulled out of the grid files over time. Grid lines run normal outward from the center of the piston, so the post processing was elementary to get the z-displacement versus radius over time. Figures 6-8 and 6-9 show two different blend regions.

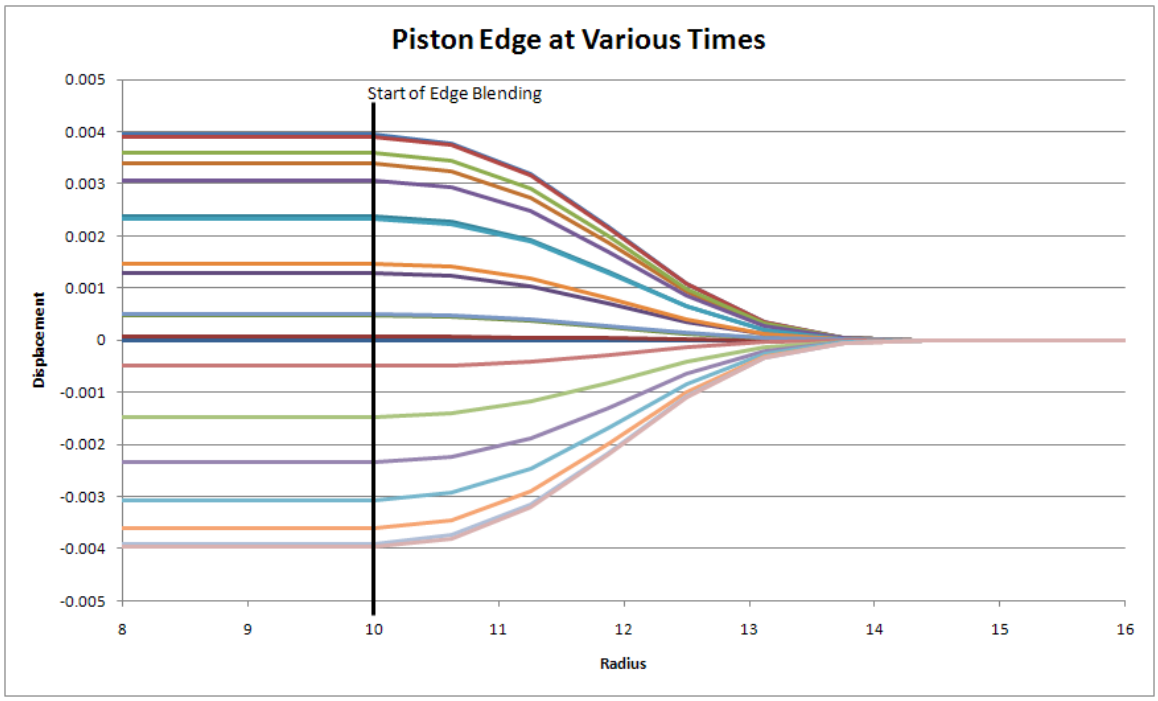


Figure 6-8: Piston blend length of 5 at various times over cycle

When implementing such a complex phenomenon such as grid motion into a high-order flow solver such as the BASS Code, continual thoughts arise that the grid motion won't be of high enough accuracy to fully and properly propagate the acoustic waves. However, after the results shown here, one can hold a certain confidence moving into the actual flow result section.

The grid deformation here has shown not only to be accurate, but smooth and unchanging over time.

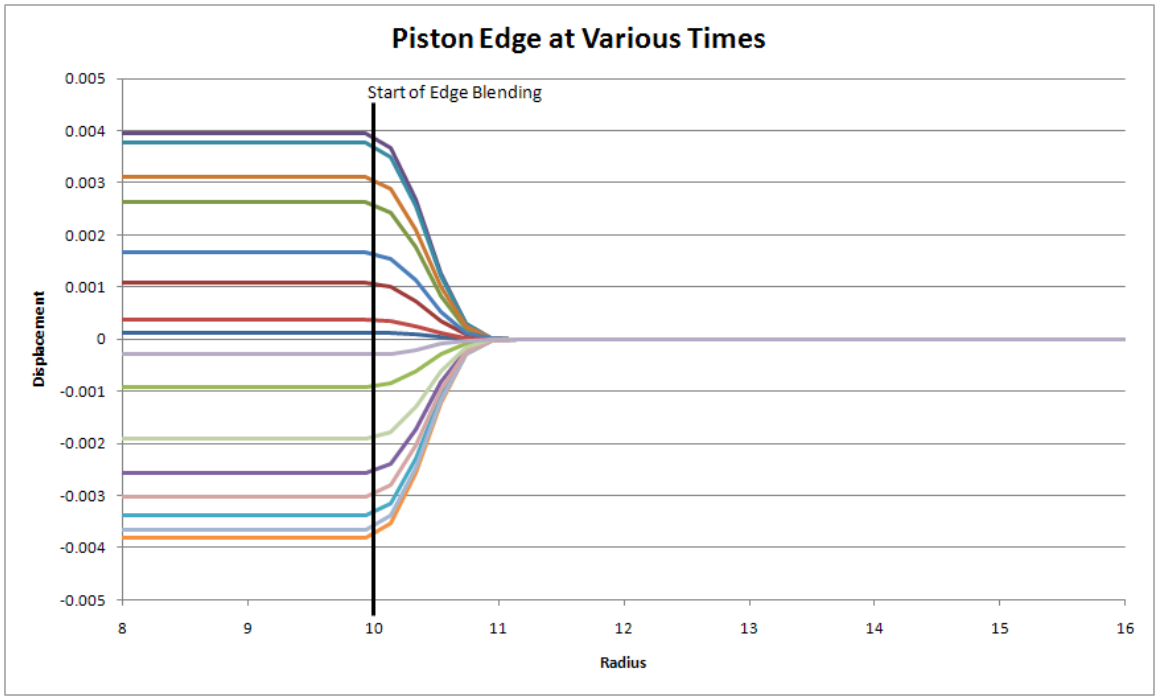


Figure 6-9: Piston blend length of 1 at various times over cycle

## 6.5 Results

Different piston frequencies were examined to ensure the code was obtaining a proper solution. Periods raning from 8 to 25 (with reduced wavenumbers $ka$ of 2.77 to 8.65) will be shown. When the wavelength of the waves are large when compared to the piston, single "pulses" are expected to propagate from the surface of the piston. As the wavelength of the propagating waves approaches and become smaller than the actual size of the piston, various interesting diffraction patterns should appear.

In order to best analyze the solutions, Fast Fourier Transforms (FFT) were performed on the solution data. The most common use of FFTs are to compute the discrete fourier transform (DFT), converting data in the time domain into the fre-

quency domain. The DFT is defined by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, ..., N-1 \tag{6.10}$$

The post-processed FFT data will allow itself to be better compared to the analytical solution. Real and imaginary components will be output, along with the full amplitude and phasing information. This will let the user determine not only if the propagting waves are of the correct size and shape, but in the right place as well.

The low frequency run with a period of 25 is shown below.

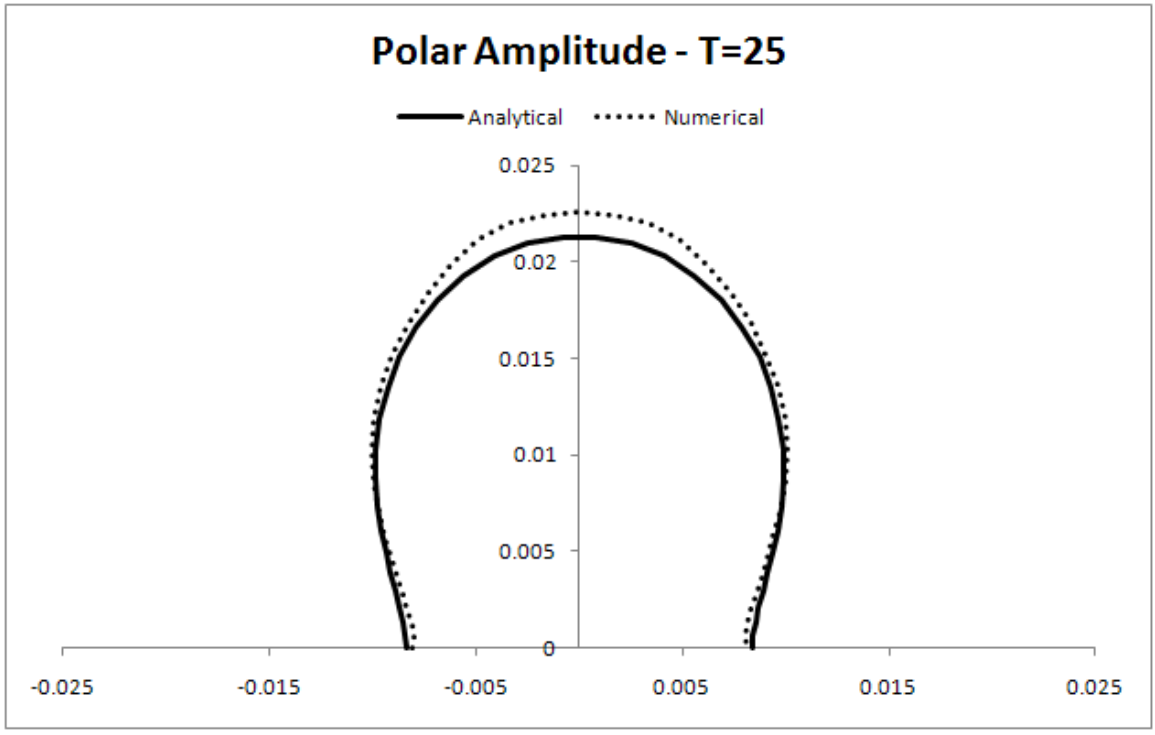


Figure 6-10: Initial Period 25 Piston Test, Polar Plot

The polar plot shown here is one that will be shown frequently. Each data point around the azimuth is scaled by the resulting FFT data such that the variation in pressure azimuthally can be determined.

Immediately one notices substantial errors. When comparing polar plots such as

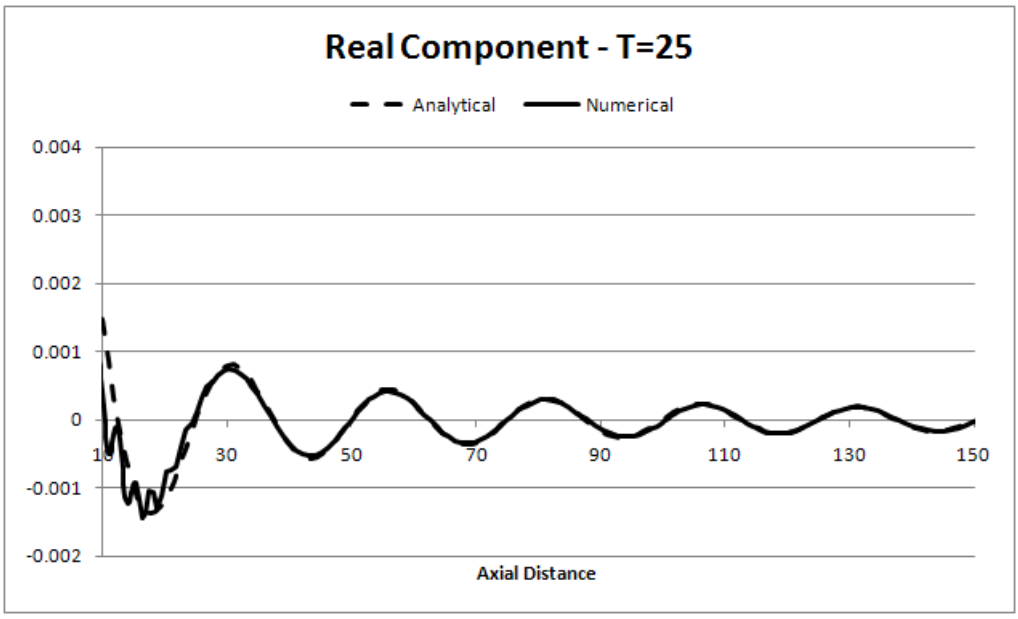


Figure 6-11: Initial Period 25 Real Component

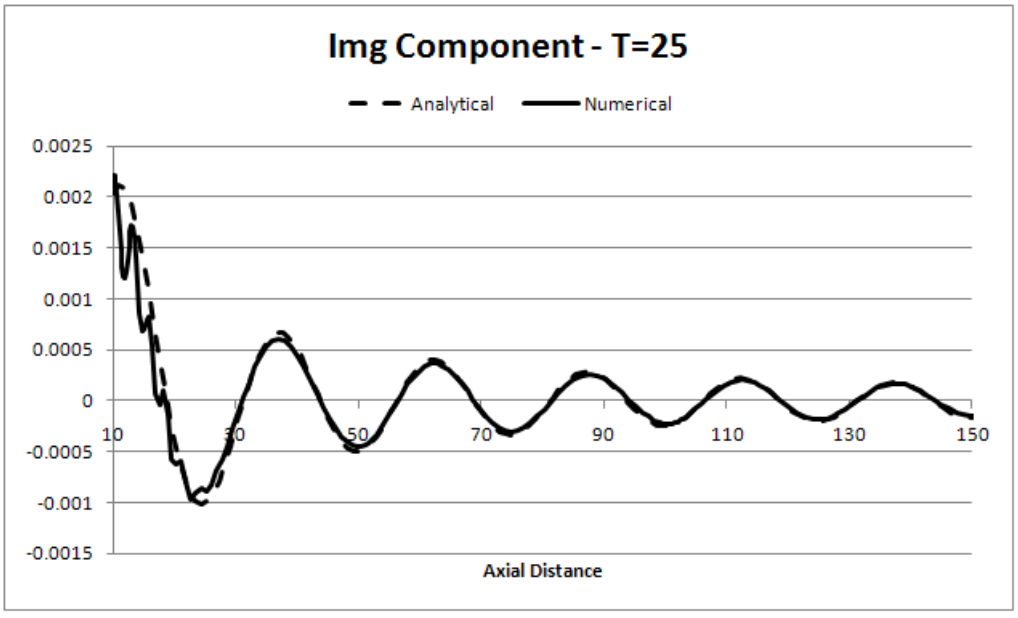


Figure 6-12: Initial Period 25 Imaginary Component

this, if the numeric result lies outside of the analytic line, then the magnitude is too large, and vice versa.

Looking at the polar plot, it appears as if both the shape and amplitude are incorrect. That is, the overall maximum magnitude, the magnitude directly on-axis, is too large. In addition to this, the shape is incorrect. The magnitude is too large in some areas, while too small in others. Lastly, the axial plots show very odd high frequency waves directly above the piston. Both of these issues need to be resolved.

### 6.5.1 Problems

#### 6.5.1.1 Far-Field Definition

The first thing that could be causing an issue is the definition of "far-field". The question arises of whether or not the data was taken at a far enough distance fro the piston such that near-field effects have subsided. To do this, the solution is simply examined at various radial locations. In the following plot, the solution has been multiplied by radius. Looking at the analytical solution, this should force the solution essentially onto the same line. The solution shown in Figure 6-13 shows three radial locations: 50, 100, and 150.

As shown, the results as a function of radius look good. Unexpectedly, even at a radius of 50, the solution has converged to its far-field results. For the purpose of this paper, the solution at r=100 will be examined. This will avoid any issues with factors of 2 arising from the radial location.

#### 6.5.1.2 Piston Definition

The next issue could be in the grid definition. Everything in CFD comes back to wave resolution. Where time steps and grid spacing has been properly defined, waves of interest will be resolved. Waves and corresponding wave numbers exceeding the limits set by the physical setup will therefore not be resolved, leading to errors.

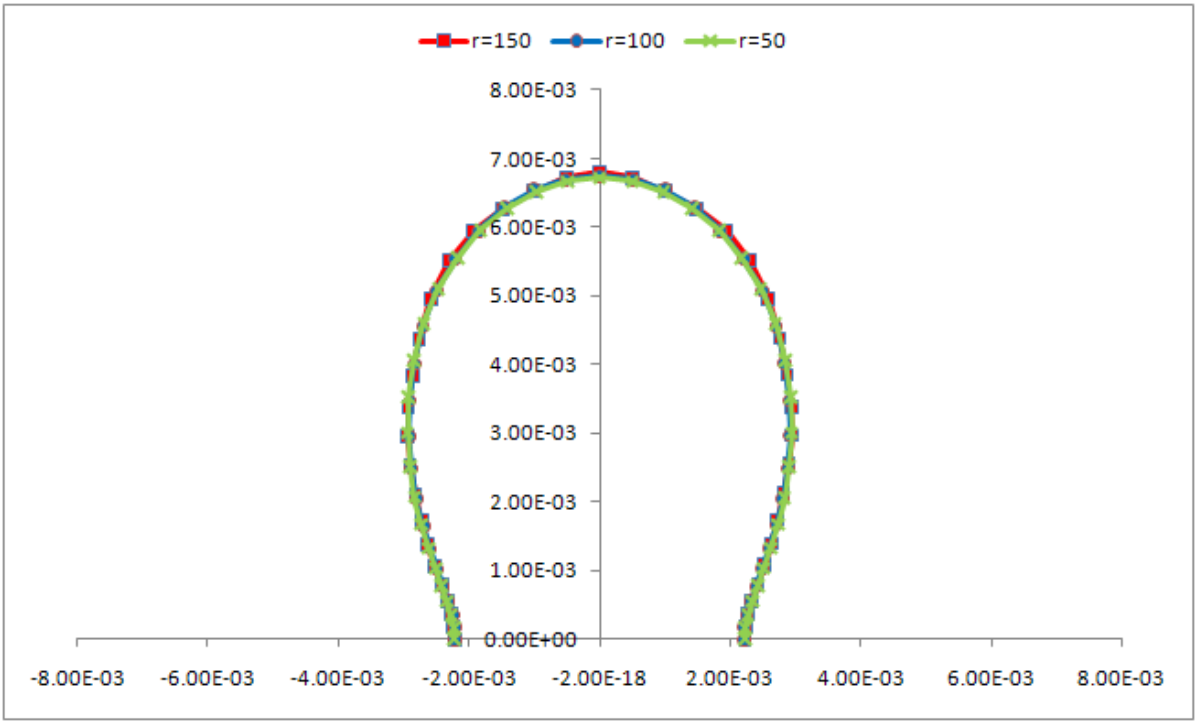


Figure 6-13: Comparison of Radial Sampling

**Fourier Transform Review** Fourier transforms are by definition an operation used to transform time (or space) domain) information into frequency (or wavenumber) domain. The frequency domain describes which frequencies are present in the original time domain function. This is very analogous to describing a *A major* chord in music, by saying that it's composed of a A, D, and E notes. For example, an *A major* chord in the time domain may look like Figure 6-14. However, performing

Figure 6-14: Time-domain waveform of *A major* chord

a fourier transform on said time domain would reveal the A, D, and E notes which make it up as shown in Figure 6-15.

In computational fluid dynamics, fourier transforms become discrete fourier transforms (DFT). DFTs are simply a kind of fourier transform in which the original function is defined discretely by a finite number of data points.

DFTs are important in the plunging piston problem. Since any solver operates discretely in time, the piston will be defined discretely in time. Like the time domain waveform of the chord example above, one could look at the piston in time and evalute its DFT. One could then see what frequencies make up the actual piston movement. Ideally, if one could define the piston by an infinte number of points, the entire magnitude of the function would be on a single frequency.

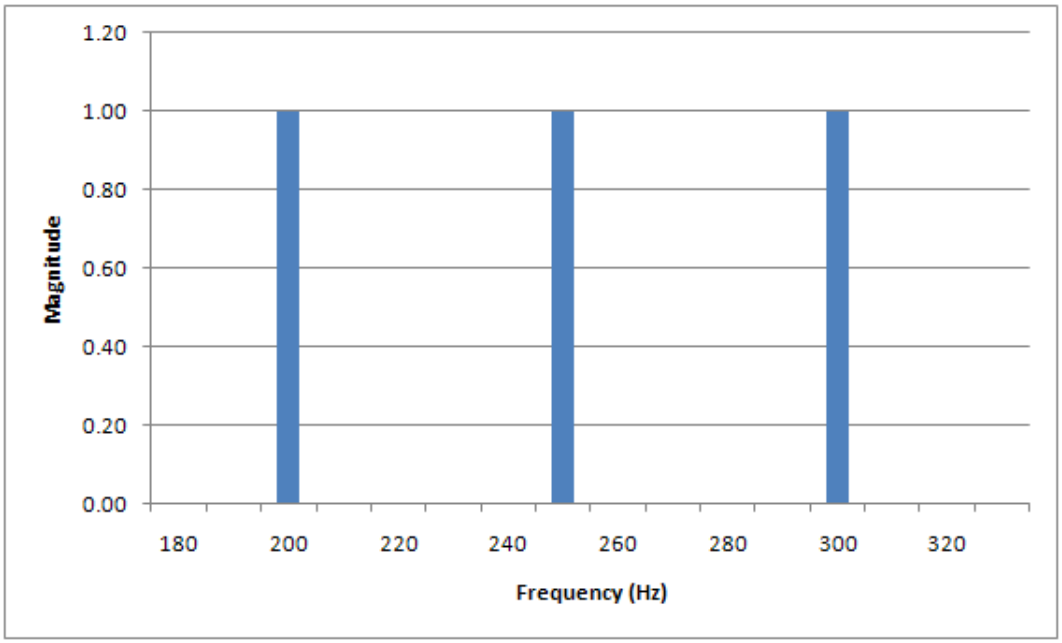


Figure 6-15: Frequency-domain waveform of *A major* chord

To use the chord analogy, in this piston test case, the user doesn't want to define the piston like a chord, a function made of several frequencies or notes. The ideal case is defining the piston as a perfect, continuous sine wave.

The problem in the computation world is, what happens in between the piston definitions? Currently, the GridWarp simply linearly interpolates between calculated grid positions for inputs to the flow solver. It is these interpolations that cause issues.

**FFT Application** It would seem as though simply defining more points would cause a better FFT result. However, this is not the case. When dealing with discrete fourier transforms, one can think of the FFT as the function which captures all of the data points. This can be thought of as a best fit line made up entirely of sine waves.

As such, if one simply defines a sine wave by four points as shown in Figure 6-16.

As shown, the 4 data points can be "captured" by a single sine wave. However, the flow solver is making calculating at points in between the defined data points.

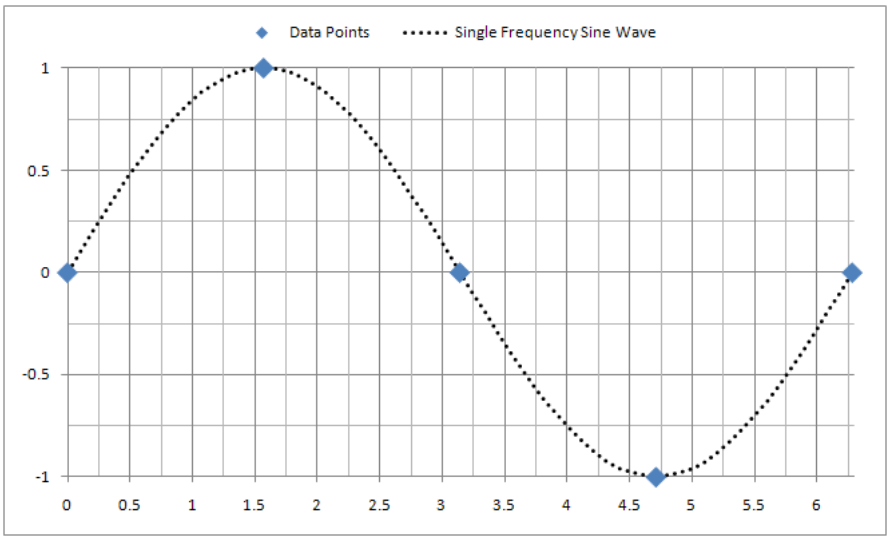


Figure 6-16: FFT Example with only 4 points per wavelength

As mentioned, currently when the flow solver "asks" GridWarp for the grid at an intermediate time, it simply uses linear interpolation between the previous and next calculated grids. This means that the actual plot looks like Figure 6-17. It is clearly

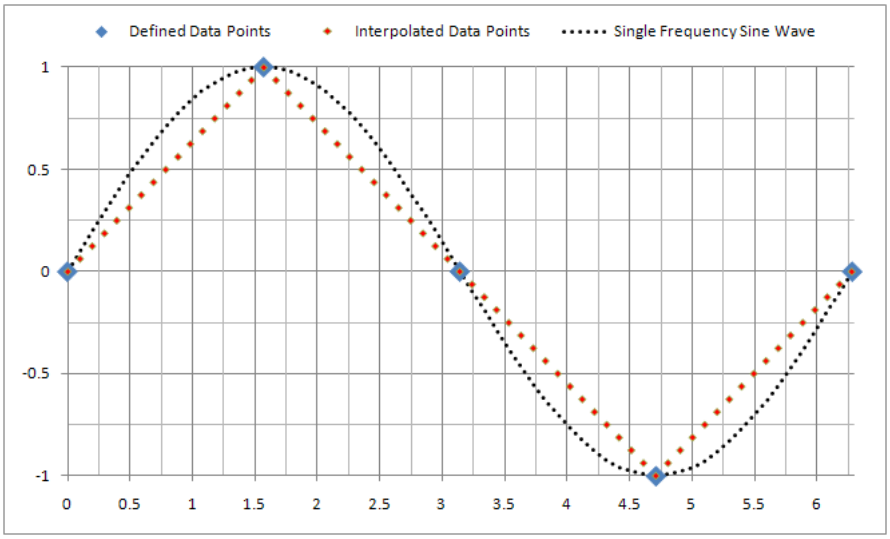


Figure 6-17: FFT Example from Flow Solve Point of View

evident that the single sine wave no longer "captures" all of the data points. It is this artificial definition of the piston by the flow solver that causes errors.

Figure 6-18 shows what the flow solver "sees". Clearly, as the number of defined points per wavelength increases, the single sine wave better captures the definition of the displacement.

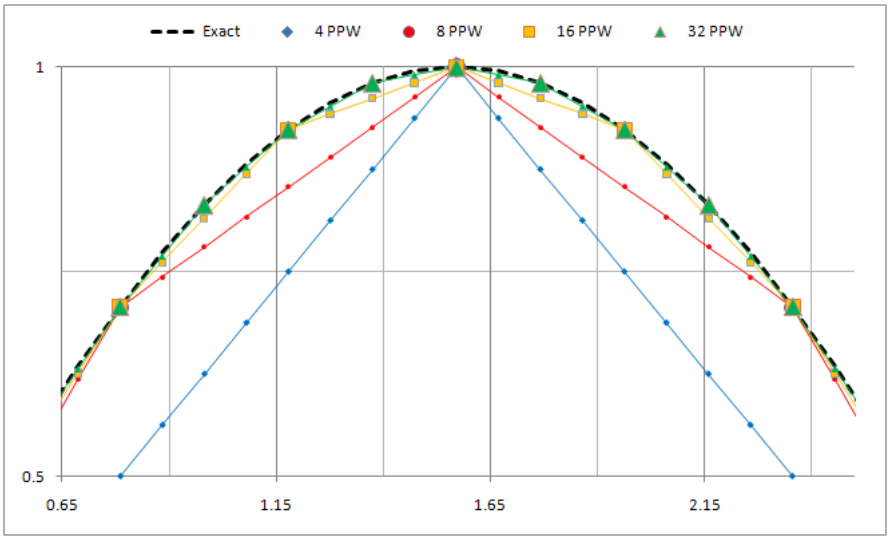


Figure 6-18: Comparison of Higher Points per Wavelength

Clearly, as the number of points per wavelength decreases, the single frequency sine wave deviates more and more from the described function. This is the source of the off-order frequencies.

As the piston is being described incorrectly during the run, at times where the grid is not being computed by GridWarp, the linear interpolation creates high-frequency spurious waves. To illustrate this, FFT analysis was performed on the functions shown in Figure 6-18. The results are shown in Figure 6-19.

The results make the data difficult to see. Defining a "perfect" 64 points per wavelength gives the expected ideal magnitude at the order of interest. Stepping down to 32 points per wavelength, or one interpolated point between defined points shows one additional frequency, a 31st order. Each time the points per wavelength are decreased and the interpolated points become more frequent, additional frequencies are added to the FFT results. That is, the results for 8 points per wavelength is

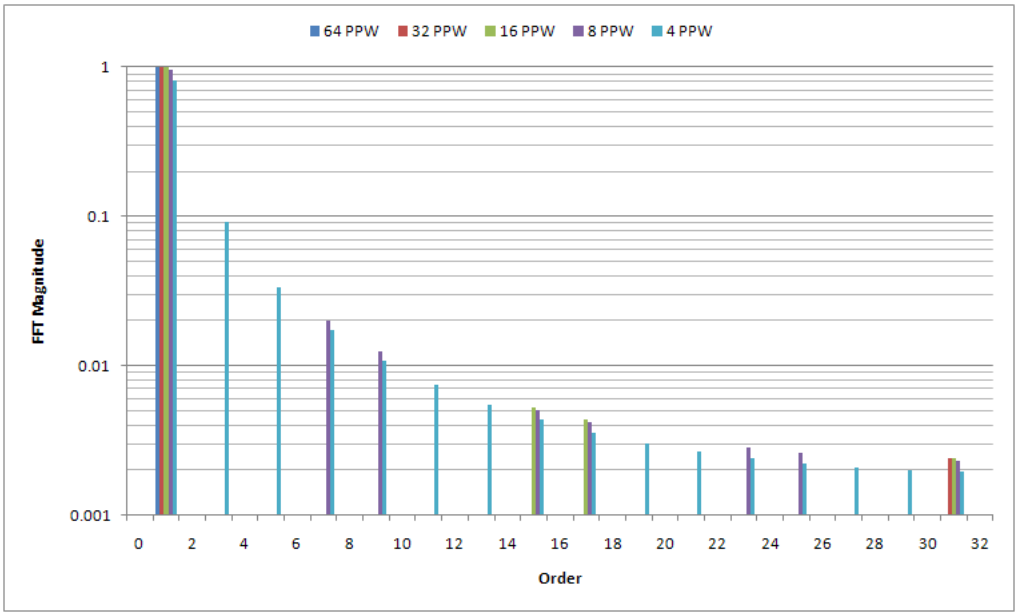


Figure 6-19: Effect of Interpolation on Defined Piston Motion

the summation of 32 and 16 points per wavelength, plus the individual results for 8 points per wavelength.

While difficult to see in detail, Table 6.1 shows the leading error terms as a ratio to the analytic magnitude.

Table 6.1: Leading error terms for various points per wavelength

| Points per Wavelength | Order | Magnitude |
|---|---|---|
| 32 | 31st | 0.002 |
| 16 | 15th | 0.005 |
| 8 | 7th | 0.020 |
| 4 | 3rd | 0.091 |

There are a couple of important points that should be made from this information. First, at low points per wavelength, the magnitude of the spurious frequencies become large. As mentioned before, a characteristic of computational aeroacoustics is that acoustic waves are often times orders of magnitudes below the mean flow. The ability to successfully predict those waves depends on the ability to keep numerical instabilities and errors from affecting them.

The second important point is the number, or order of the frequency. Each mesh can resolve and propagate a maximum frequency of wave dependent on the grid spacing. If the order, or frequency of the waves are too high, then grid simply cannot resolve the waves. However, if the frequency is low enough such that the grid can resolve them (such as the low orders), the grid will resolve and propagate them, adding to overall errors.

Simply put, spurioius frequencies in the piston motion will generate spurious frequencies in the acoustic response which will diminish the amplitude of the waves at the actual frequency.

It was then realized that too few points have been defined for piston definition. To resolve this issue, simply increasing the number of points defining the piston motion was performed. In this first case, 32 defined piston points per cycle were chosen, and the results are as follows.

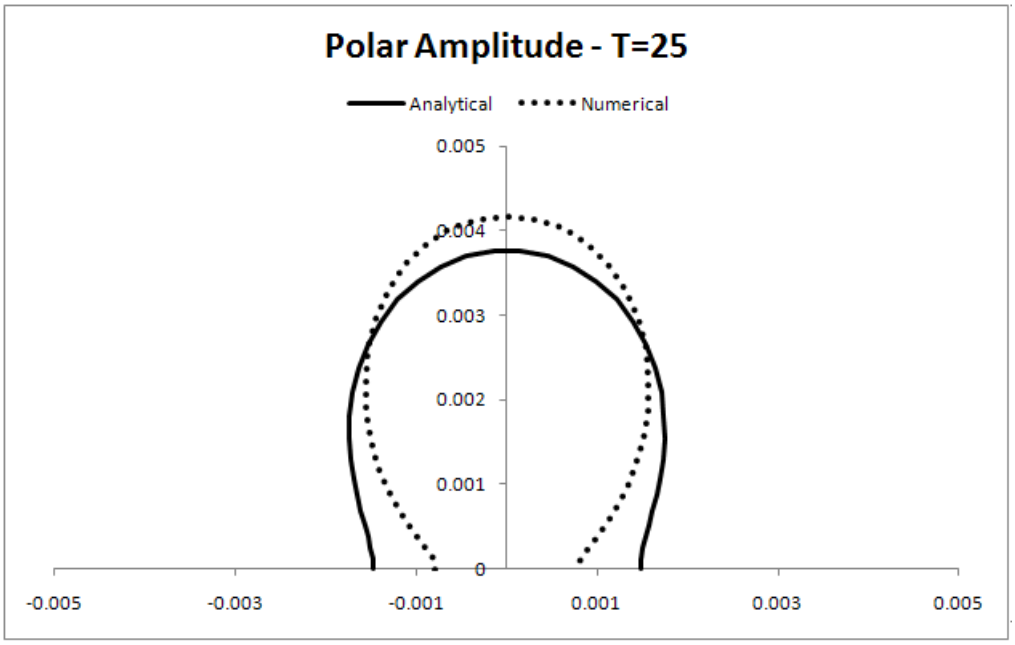


Figure 6-20: Period 25 Piston Test, Polar Plot

Looking at the axial real and imaginary plots, it is clear that the high frequency waves propagating from the poorly defined piston have been eliminated. It should

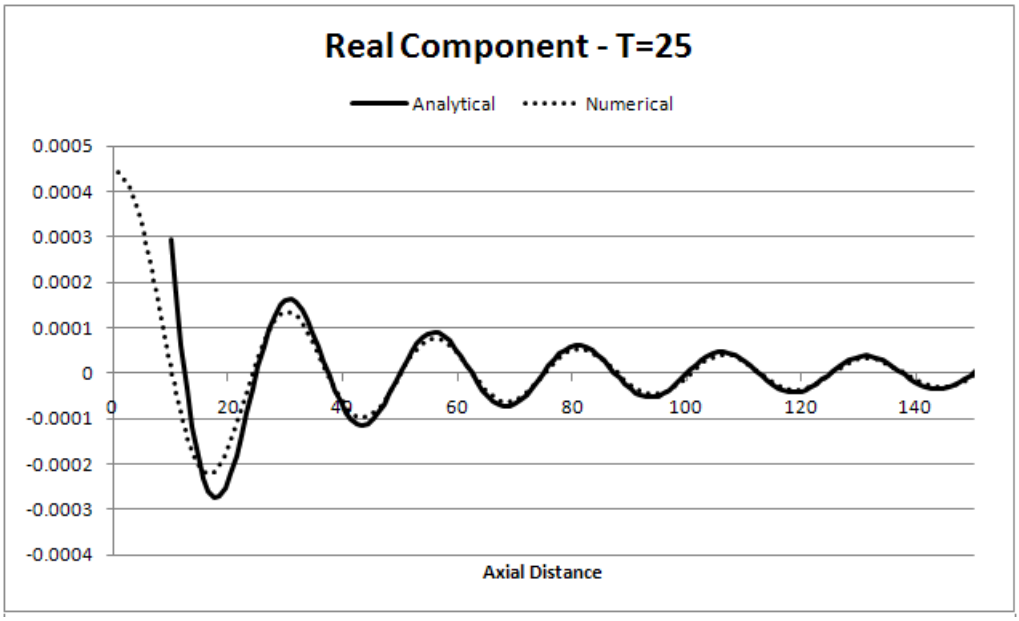


Figure 6-21: Period 25 Real Component

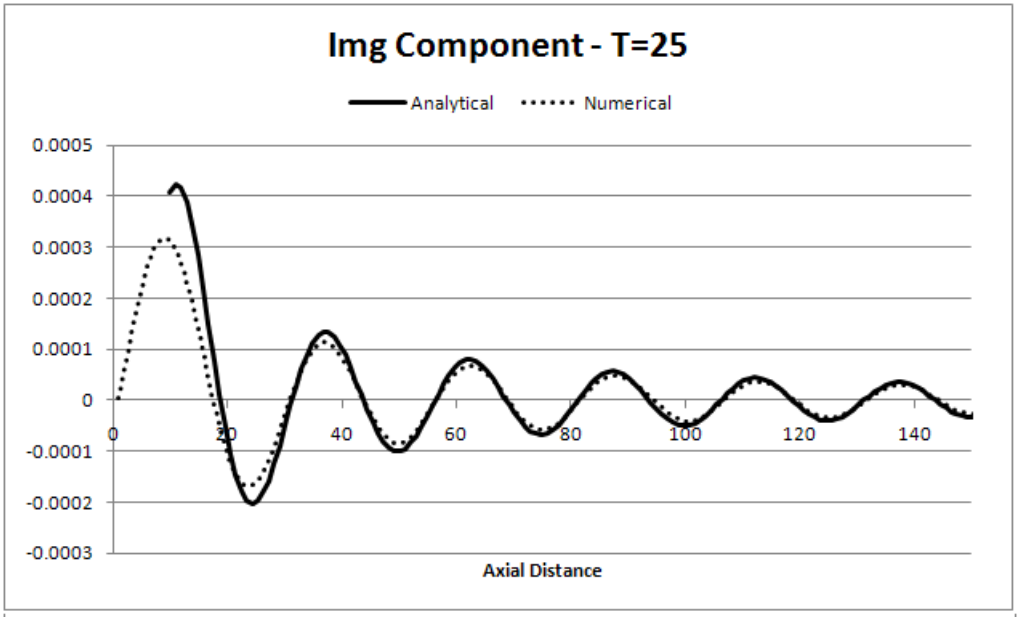


Figure 6-22: Period 25 Imaginary Component

be noted that the analytic solution does not go to zero because the amplitude is a function of $1/r$. As such, the graphs would look rather skewed.

With one problem solved, one remains. The shape and overall amplitude of the waves are still not correct. Looking back at the governing equations, there is one term that jumps out. One term solves both problems.

### 6.5.2 An Effective Radius

High-order, high-accuracy flow solvers are like any high performance engine. It is carefully placed together, with skilled, masterful hands. With just the right inputs, it can run like no other, fast, tight, and accurate. However, with one thing is out of whack, just one thing, everything breaks loose.

The BASS Code is certainly such a flow solver. Single grid points out of place can completely destroy an entire run. It takes steady attention and a skilled eye to pinpoint and debug these seemingly phantom errors. When dealing with grid motion, it seems as though these errors only exacerbate themselves.

The BASS Code wants, nay *needs* a quality grid. Much in the same fashion that the flow field is numerically differentiated, the actual grid is as well. If the grid is not continuous, and its derivative continuous, then small errors can grow into large errors, "bombing" a solution very quickly.

A piston is by definition discontinuous. Gaussians have continuous displacement and slopes at its boundaries, while a sinusodial wave has continuous displacement only. Pistons, with their discontinuous boundaries can make trouble for any solver, let alone a solver which requires continuous boundaries.

Because of this fact, as mentioned before, the edge of the piston was blending down to zero displacement. The strategy was to hopefully have enough grid points over the blend to maintain continuity through the grid. In an effort to ensure the grid remained smooth and a solution was obtained, the blending region was chosen

arbitrarily high.

Looking back at the analytical solution, there is essentially two parts. There is a time-harmonic constant, that is a function of defined constants, and a Bessel function. The Bessel function was shown in Figure 6-1.

Recall that the Bessel term controls the directivity of the wave. At low frequencies, variation in the azimuthal direction does not vary the strength of the wave significantly. However, as one increases the frequency, or more accurately, the frequency times the piston radius, changes in direction can significantly alter the strength of the wave.

In addition to this Bessel term controlling the shape of the wave, the constant term controls the strength. This strength is a function of freestream density, maximum piston velocity, and piston radius.

One thing is common between those two terms: the piston radius. Increasing the piston radius will have the effect of increasing the $kz$ term, making the wave more directional, along with increasing the overall magnitude of the propagating waves. This is exactly what is shown in the previous results.

Knowing this leads itself to first draw a comparison between solutions using different piston blending zones. Figures 6-23 shows a comparison of different blending zones . All runs were ran to convergence using a piston period of 25 with subsequent frequency and wavenumber of 0.251. The small, medium and large blending regions are 20%, 50%, and 100% of the piston radius.

The three plots looks very similar, in fact looking at the three plots one might think they are identical, but they are different. A close eye will notice the large amplitude waves becoming ever so slightly more directional as the blend increases. In addition to this, as you progress to the right, or towards a larger blending region, the contours become "darker". This indicates that the magnitudes are higher in these waves. While it may not be the best for comparison, figure 6-24 shows a direct comparison

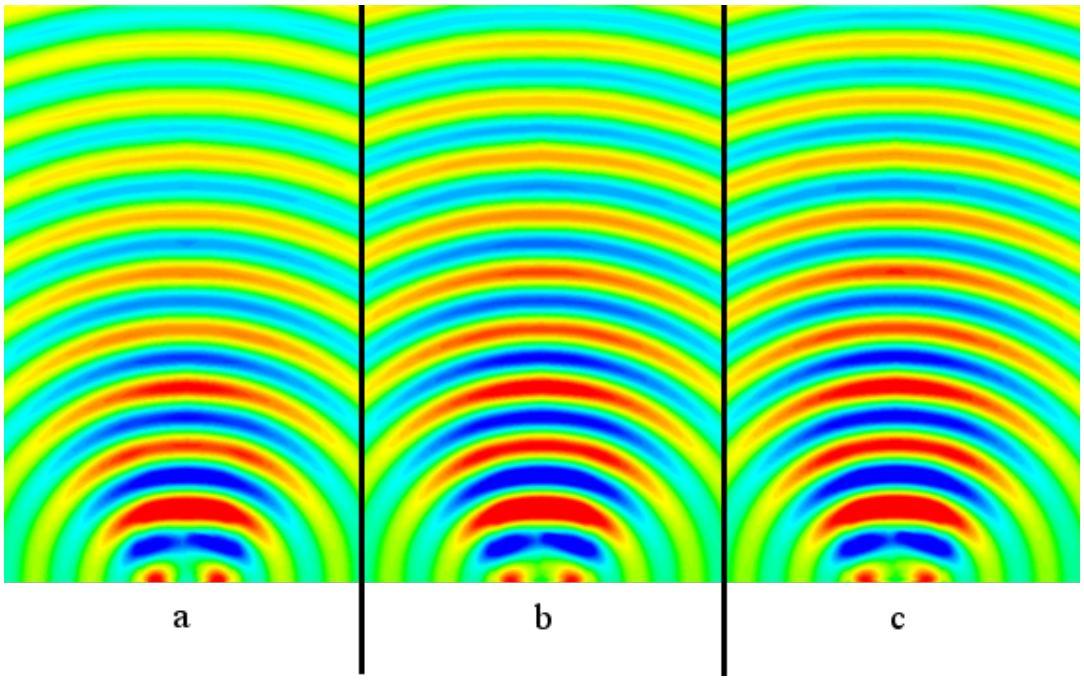


Figure 6-23: Comparison of FFT Snap contours for blending regions of: a. 2, b. 5, c. 10

between the three runs. As shown, by simply altering the length of the blending radius, the shape and magnitude of the propagating waves change drastically. Even though the maximum velocity and displacement is constant, and constant over the same area in all runs, they are starkly different.

If this is true, then by simply modifying the analytic solution to some "effective" radius, then the solutions should then fall upon each other. Figures 6-26 through Figure 6-28 show the numerically obtained solution, along with the analytic solution assuming a piston radius of 10, and finally a hand-fit analytic solution.

In these figures, the small blend hand-fit analytic solution was modifed to 10.7, the medium to 12.5, and the large to 14.0. These hand-fit radii will be referred to as $r'$ hereafter.

Seeing this, it's apparent that the solution is clear: this low frequency test case requires some "effective" piston radius to be taken into account. The blending re-

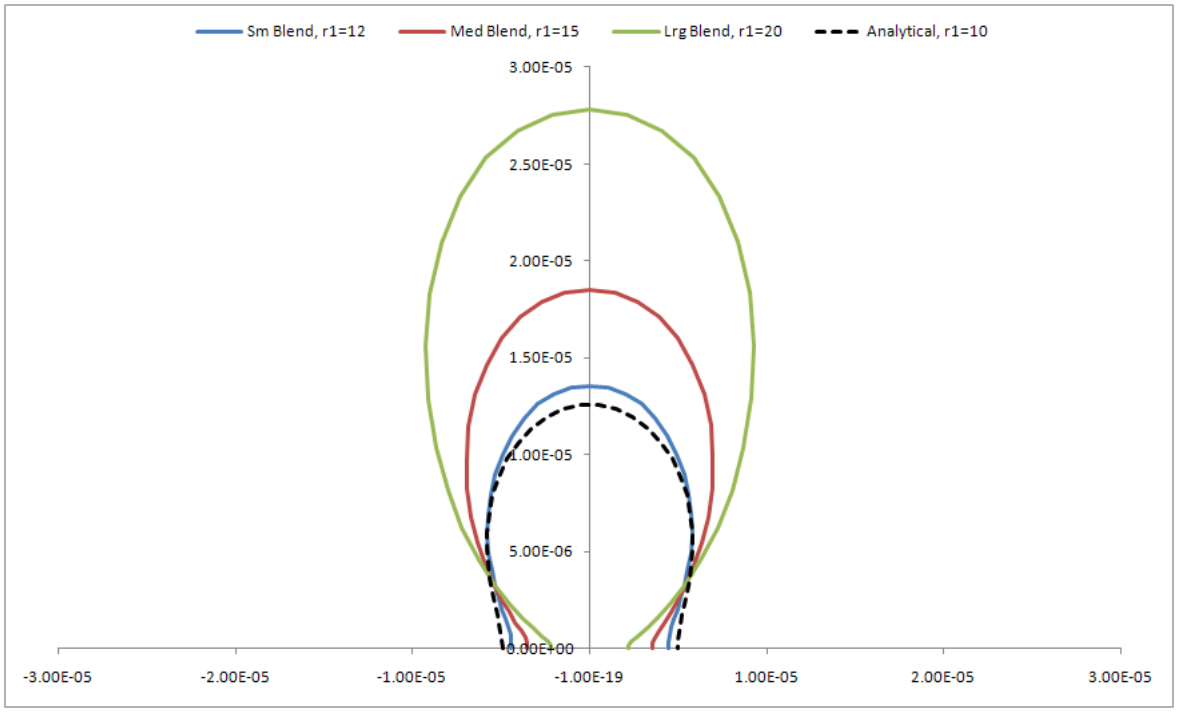


Figure 6-24: Comparison of Identical Runs Using Different Blend Areas

gion affects the critical $kz$ term, which changes both the wave shape, as well as the magnitude.

The question now becomes, why?

Lighthill [31] introduced what is now famously known as the "Acoustic Analogy". Ribner [45] [46] [47] expanded on this becoming what is known as the Dilatational Theory of Sound. At the core of this theory states that changes in volume (dilatation) causes sound.

This leads itself to first examine the actual volume of the piston that is oscillating. The reader at this point is directed to Appendix A for the derivation of the change in volume of the fluid in the blending region, where the final solution is given by

$$V = \pi \left[ r_1(r_1 - r_0) - \frac{7}{10}(r_1 - r_0)^2 \right]$$

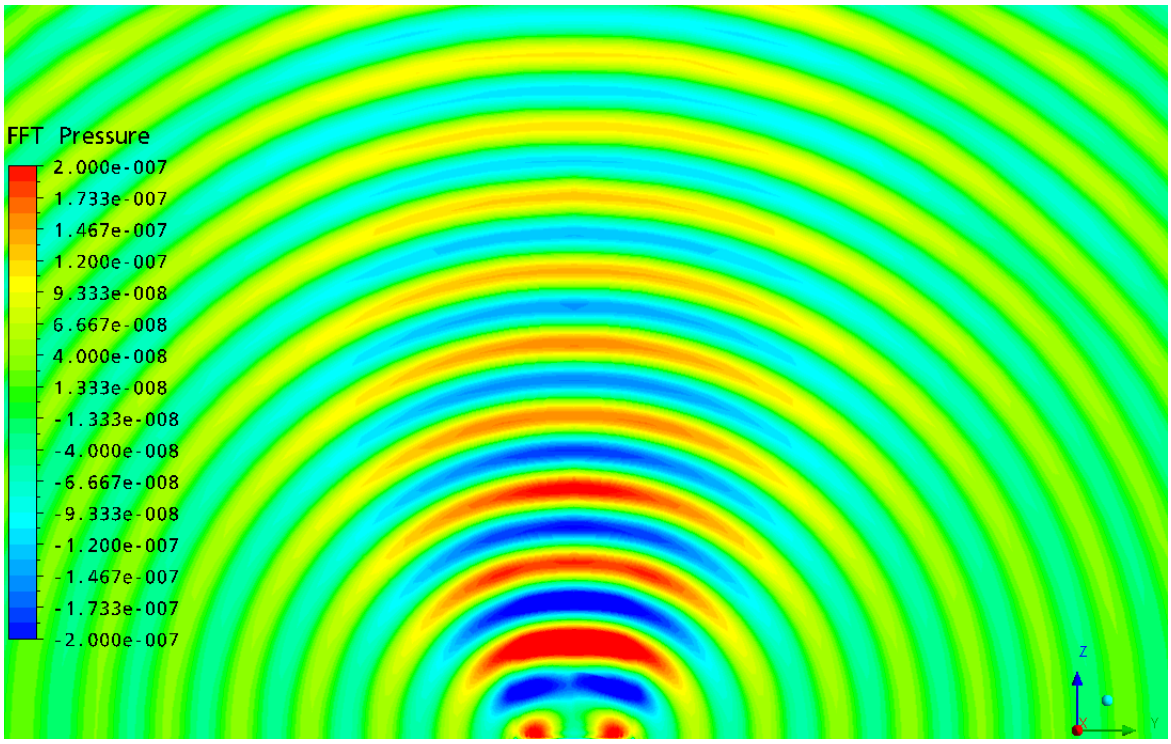


Figure 6-25: Pressure Pertubation Period 25, Small Blend

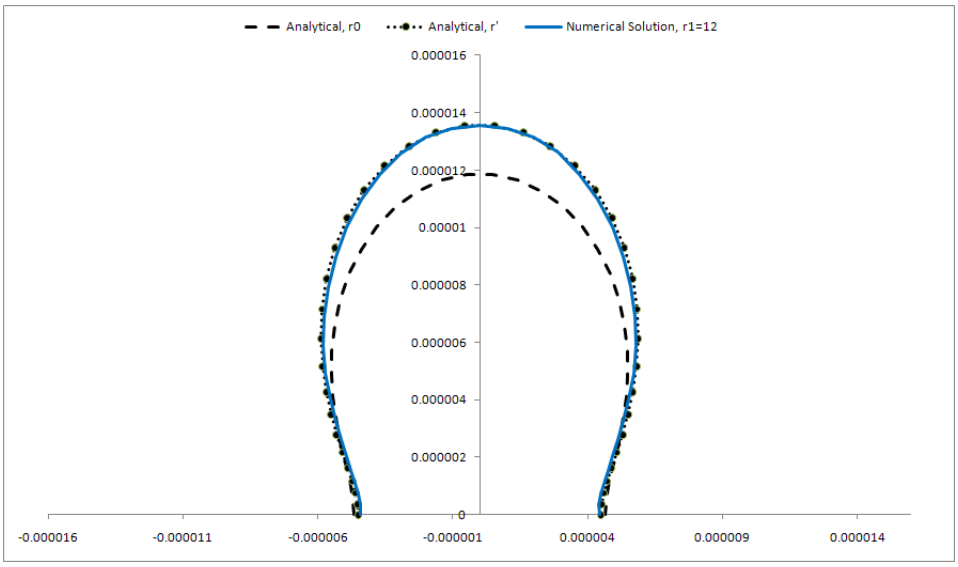


Figure 6-26: Numerical Results vs Modified Analytic Solution - Small Blend

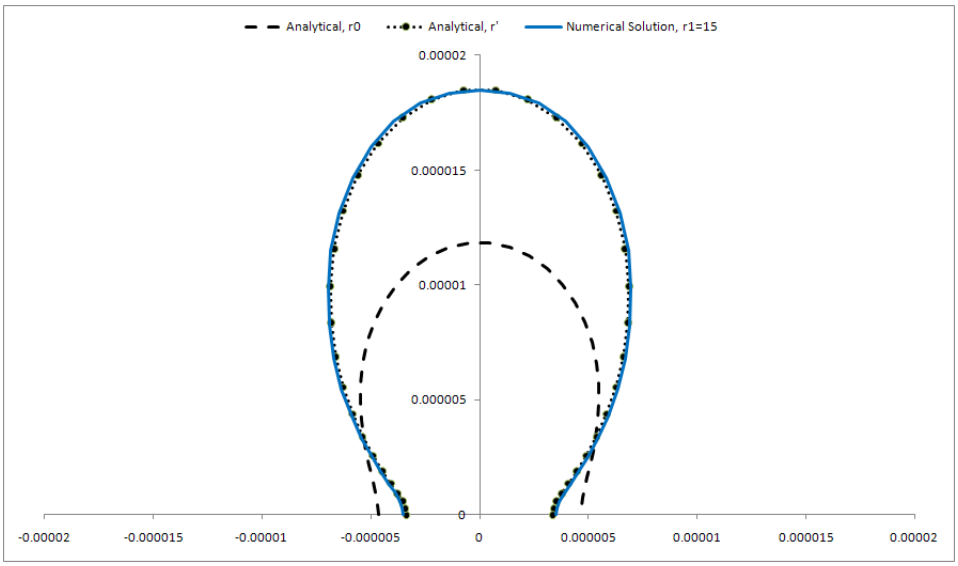


Figure 6-27: Numerical Results vs Modified Analytic Solution - Medium Blend

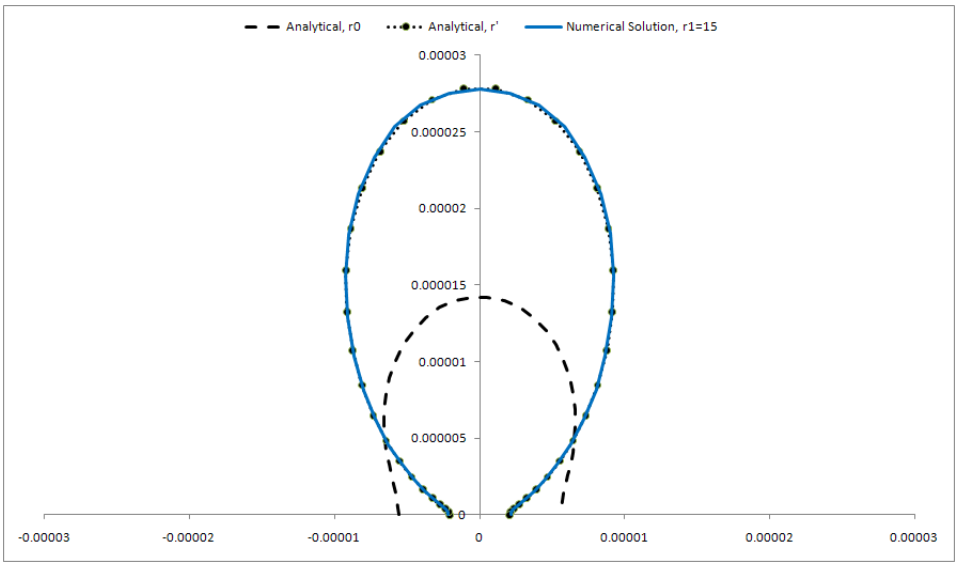


Figure 6-28: Numerical Results vs Modified Analytic Solution - Large Blend

Where $r_0$ is the edge of the actual piston, and $r_1$ is where the piston has blending to zero displacement. The volume of the both the blending region and the piston is then given by

$$V = \pi \left[ r_0^2 + r_1(r_1 - r_0) - \frac{7}{10}(r_1 - r_0)^2 \right] \tag{6.11}$$

An effective radius, $r^*$ can then by given by taking the square root of this new volume divided by $\pi$.

$$r^* = \sqrt{r_0^2 + r_1(r_1 - r_0) - \frac{7}{10}(r_1 - r_0)^2} \tag{6.12}$$

Table 6.2 gives a comparison of the different radii and their corresponding volumes, while Figure 6-29 shows a visual comparison of the actual pistons and their corresponding effective counterparts.

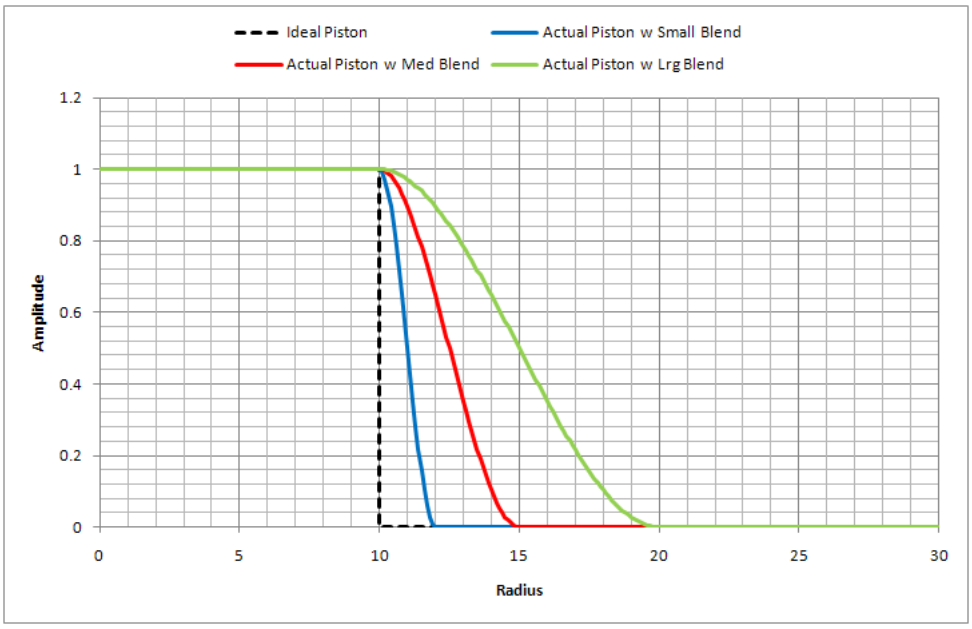


Figure 6-29: Comparison of Piston Blending Regions

In all cases the numerically determined effective piston radius, $r'$ is less than the volumetrically determined effective piston radius.

Table 6.2: Comparison of Various Parameters for Different Blending Regions, T=25

| Blend | $r_0$ | $r_1$ | $V$ | $r^*$ | $r'$ | "error" |
|---|---|---|---|---|---|---|
| Small | 10 | 12 | 380.8 | 11.01 | 10.70 | 2.7% |
| Medium | 10 | 15 | 494.8 | 12.55 | 12.5 | 0.4% |
| Large | 10 | 20 | 722.6 | 15.17 | 14.0 | 7.7% |

### 6.5.3 Problem Resolution: High-Frequency Comparison

What was supposed to be the simple test case, the low frequency run, turned out to be quite difficult. It has been shown that the blending region at the edge of the piston gives some sort of effective radius, changing the shape and magnitude of the propagating waves. Knowing this leads itself to determining the interaction between the piston edge blends and the propagating waves for high frequency test cases.

A period of 12 is chosen to be the high frequency test case. Realizing at this point that large edge blends can have a large detrimental effect on the solution, the edge blends are halved; the small blending region is chosen as 1, followed by 2 and 4 for medium and large respectively.

Like the low frequency test, these tests are all ran until convergence and the plots are shown below along with the analytical solution.

There are a couple of things that are quite apparant. First, the high frequency test case does in fact show the edge effect as the low frequency test case does. That is clear.

What any numericist craves is to have a method that will allow prediction of a result. One thing that a numericist will tolerate is postdiction, quantification of the effects seen after the matter. In this case, the effect is that of the edge blend. As such, each test case has been plotted against the analytic solution along with the volumetric effective radius that derived in Equation 6.12. The results are shown below.

As shown, for piston radii larger than the small blending region of 1.0, the volumetrically defined equivalent radius doesn't correlate well with numerically obtained

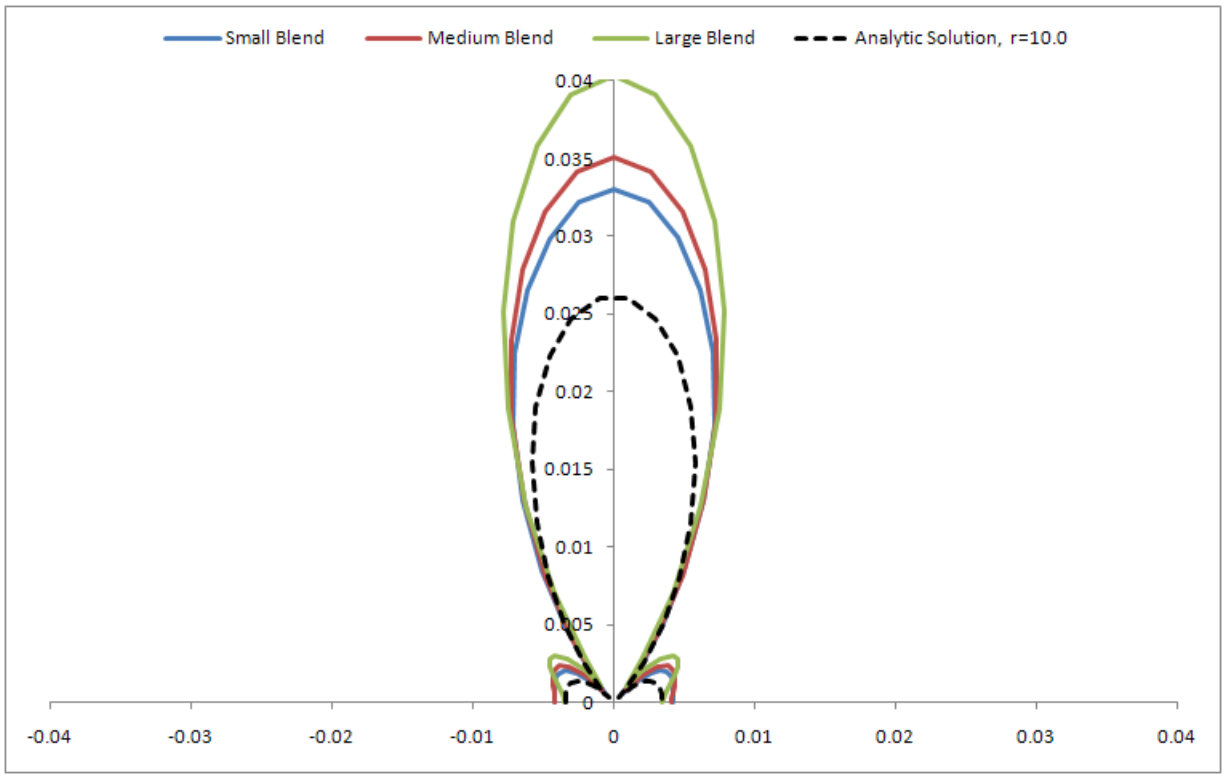


Figure 6-30: Comparison of Blending Sizes for High Frequency Test

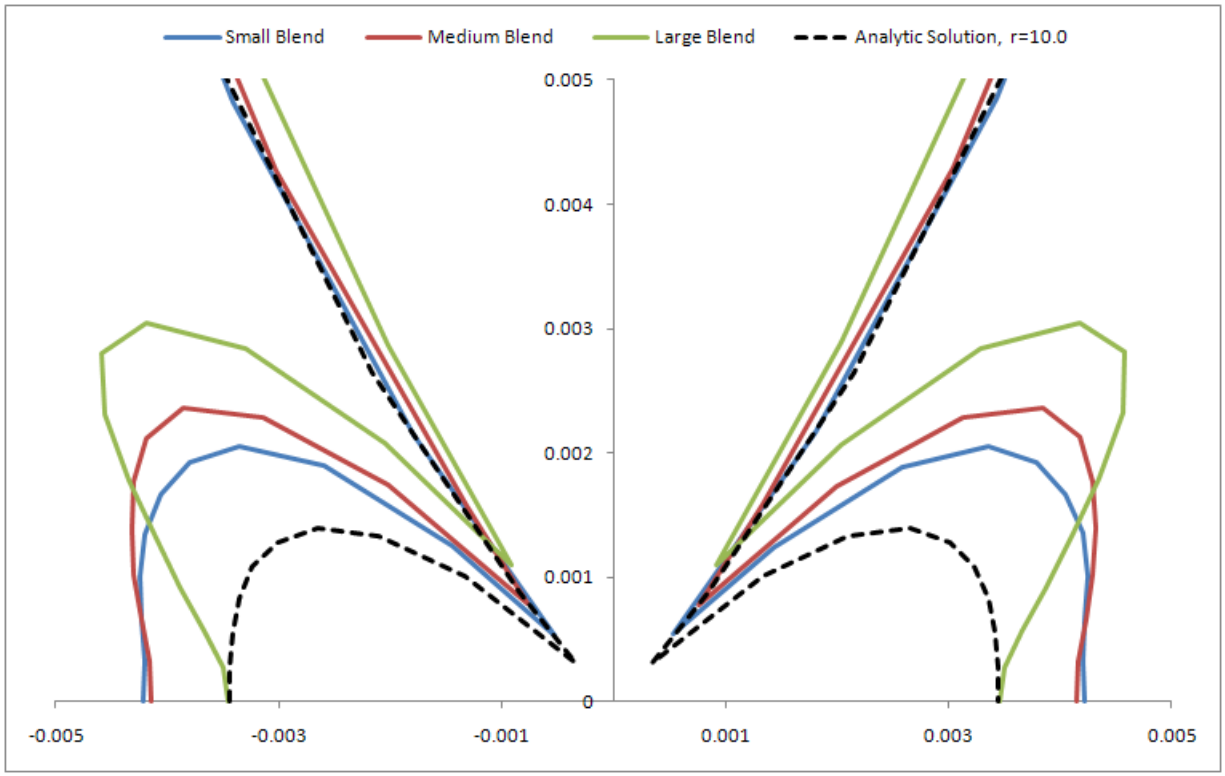


Figure 6-31: Comparison of Blending Sizes for High Frequency Test-Close Up

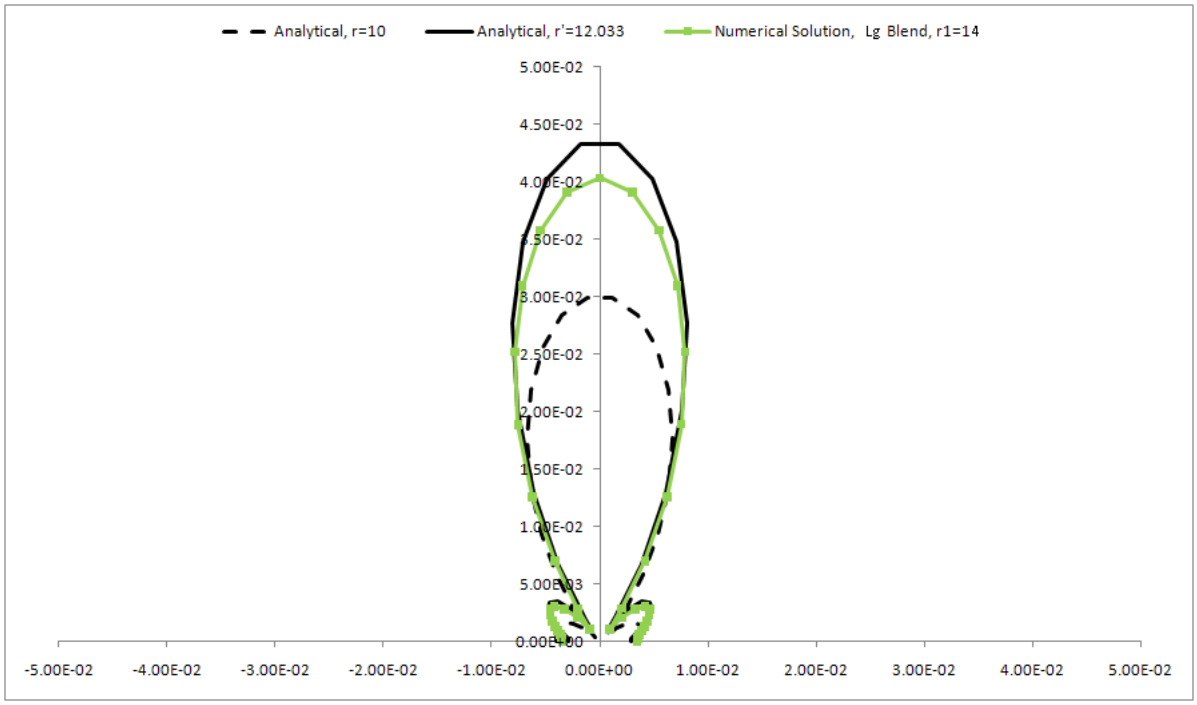


Figure 6-32: T=12 Run with Large Piston Blend

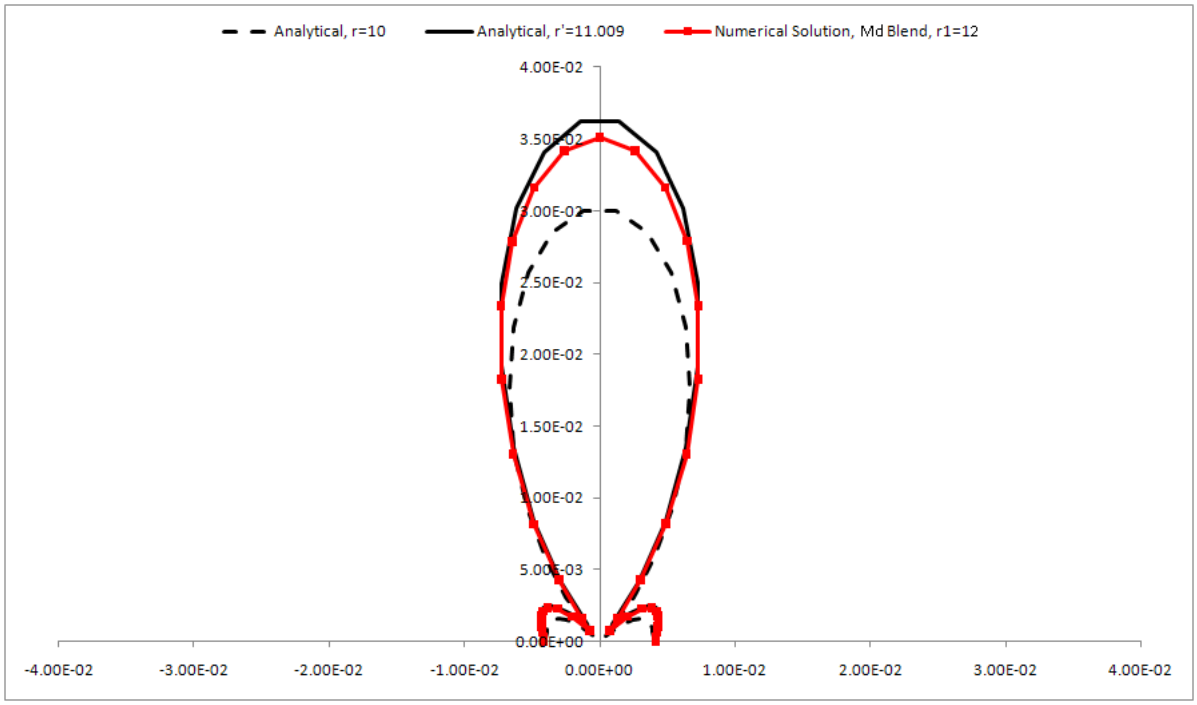


Figure 6-33: T=12 Run with Medium Piston Blend

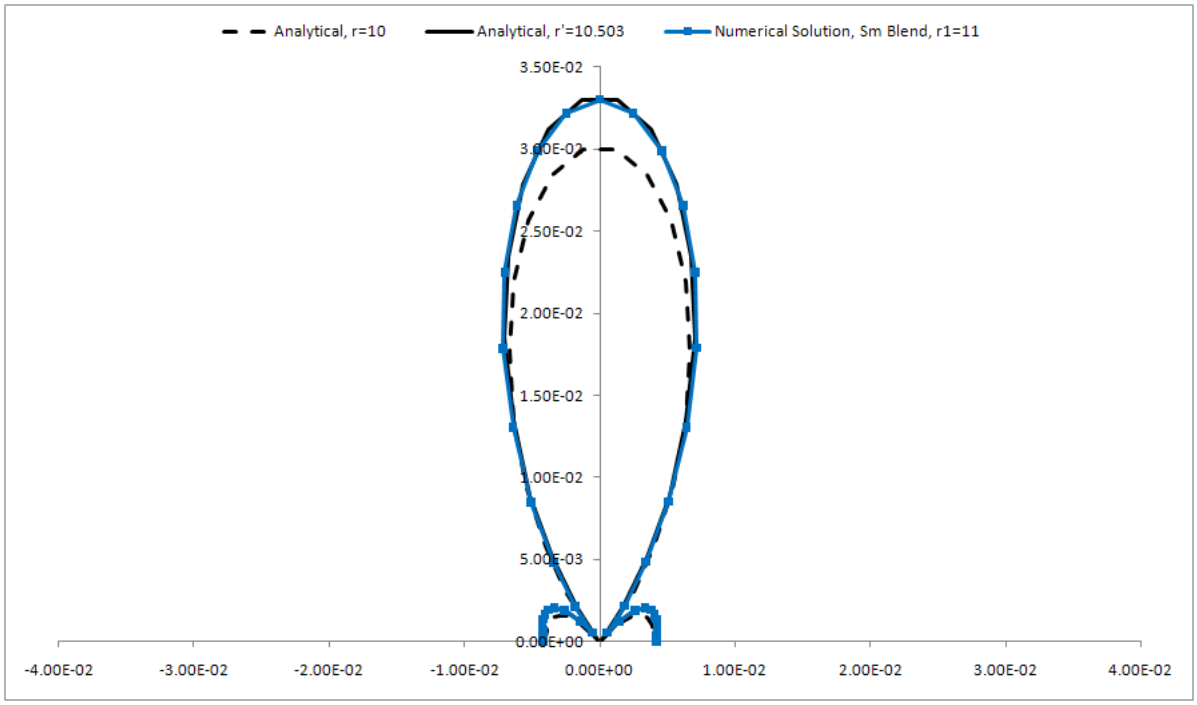


Figure 6-34: T=12 Run with Small Piston Blend

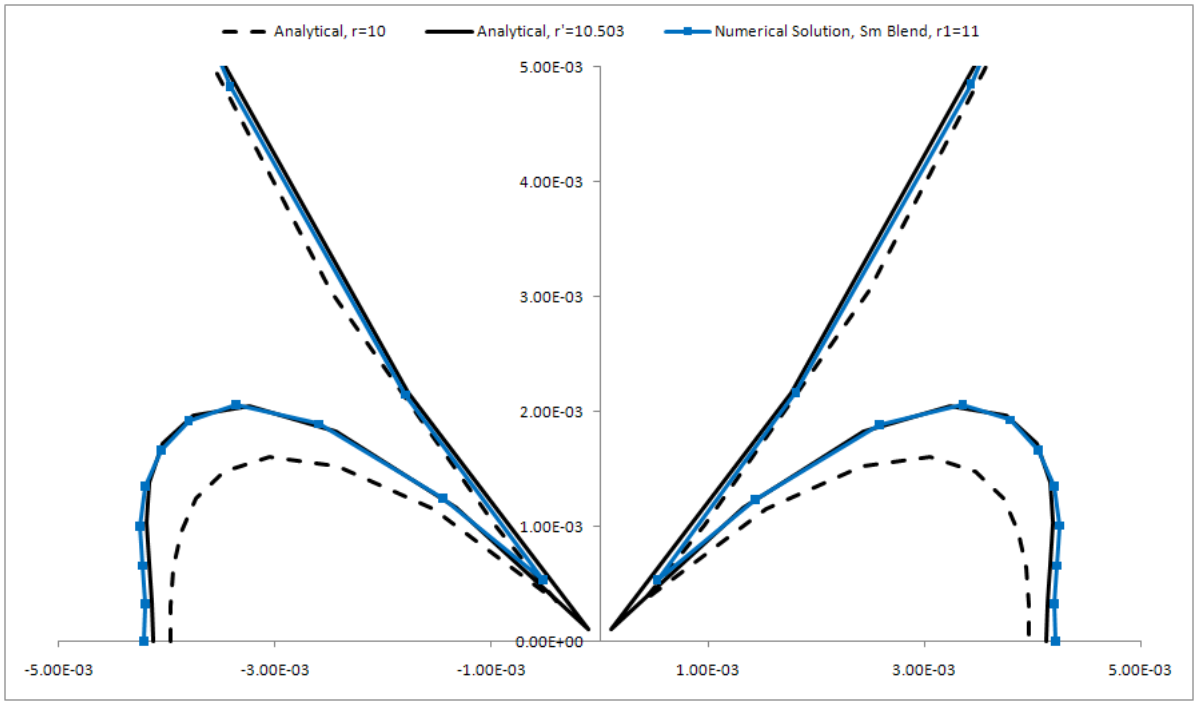


Figure 6-35: T=12 Run with Small Piston Blend - Close Up

results. However, for a blending region of 1.0, the solutions fall on top of each other.

This leads one to believe that there is a minimum radius needed for the defined volumetric equivalent radius to be applicable. However, as shown, inside of this minimum blending region, the solution matches excellently to the analytic solution.

Table 6.3: Comparison of Various Parameters for Different Blending Regions, T=12

| Blend | $r_0$ | $r_1$ | $V$ | $r^*$ | $r'$ | "error" |
|---|---|---|---|---|---|---|
| Small | 10 | 11 | 346.5 | 10.50 | 10.50 | 0.0% |
| Medium | 10 | 12 | 380.8 | 11.01 | 10.35 | 6.0% |
| Large | 10 | 14 | 454.9 | 12.03 | 11.1 | 7.7% |

With this in mind, the next step would be to re-examine the low frequency case, using a smaller piston blending region.

### 6.5.4 Low-Frequency Revisited: Part I

Figure 6-36 is an updated version of the plot shown before. The additional data series titled *V Small* is the data from a test ran with an edge blend of only one, mirroring the excellent results for the high frequency case.

Figure 6-37 is a closeup showing only the new data series along with the analytical solution and the solution for the "effective" radius, which was mentioned in Table 6.3 as $r' = 10.50$. These results are eye-opening. *Now*, it appears as if the piston is following the volumetric effective radius calculation. With the small piston blend, the errors are orders of magnitude smaller than with the large piston. However, zooming in, magnitude and shape errors are still apparent. However, when adjusting the piston radius to the volumetric effective radius, the solutions become nearly coincedent.

Based on these results, one could conclude that the blending region should be no more than 10% of the radius. Within that range, dilitational theory holds and one can apply the concept of a moving volume to derive an effective radius.

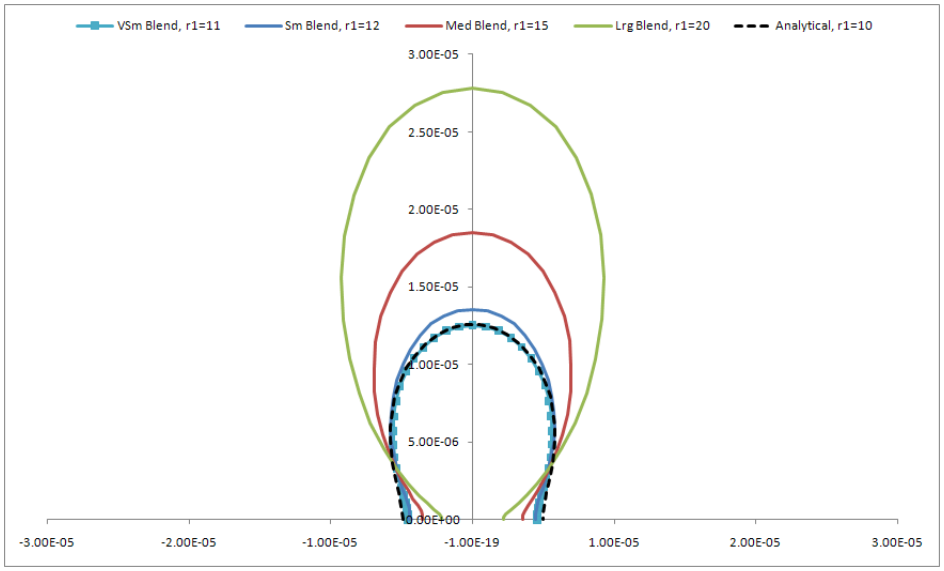


Figure 6-36: T=25 Test, Updated Blend Comparison

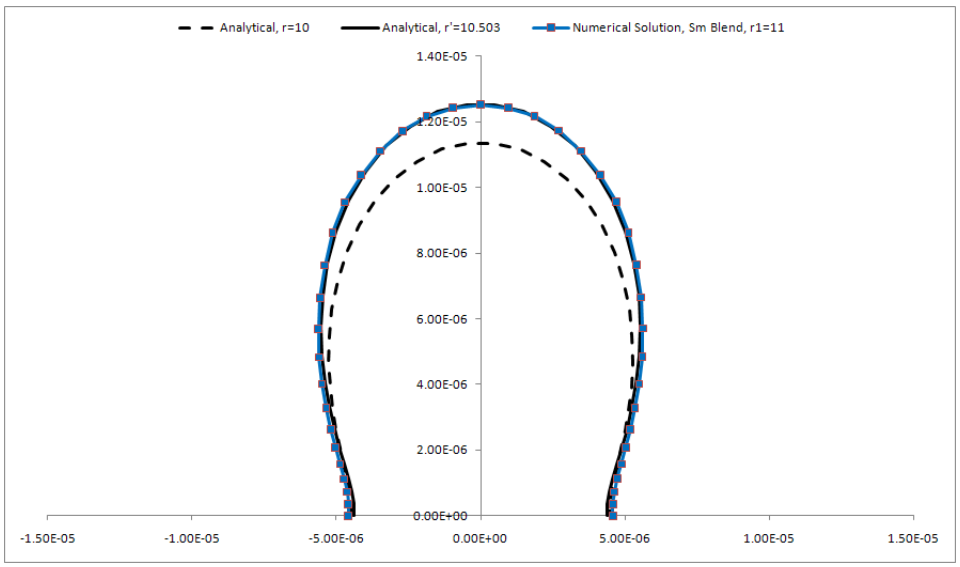


Figure 6-37: T=25 Test, Very Small Blend Region vs Analytical and Effective

### 6.5.5 Medium Frequency

The medium frequency test now simply falls into place and the results are shown below using a piston blend region of 1.0. The analytic solution is the solution for a "perfect" piston of radius 10.0

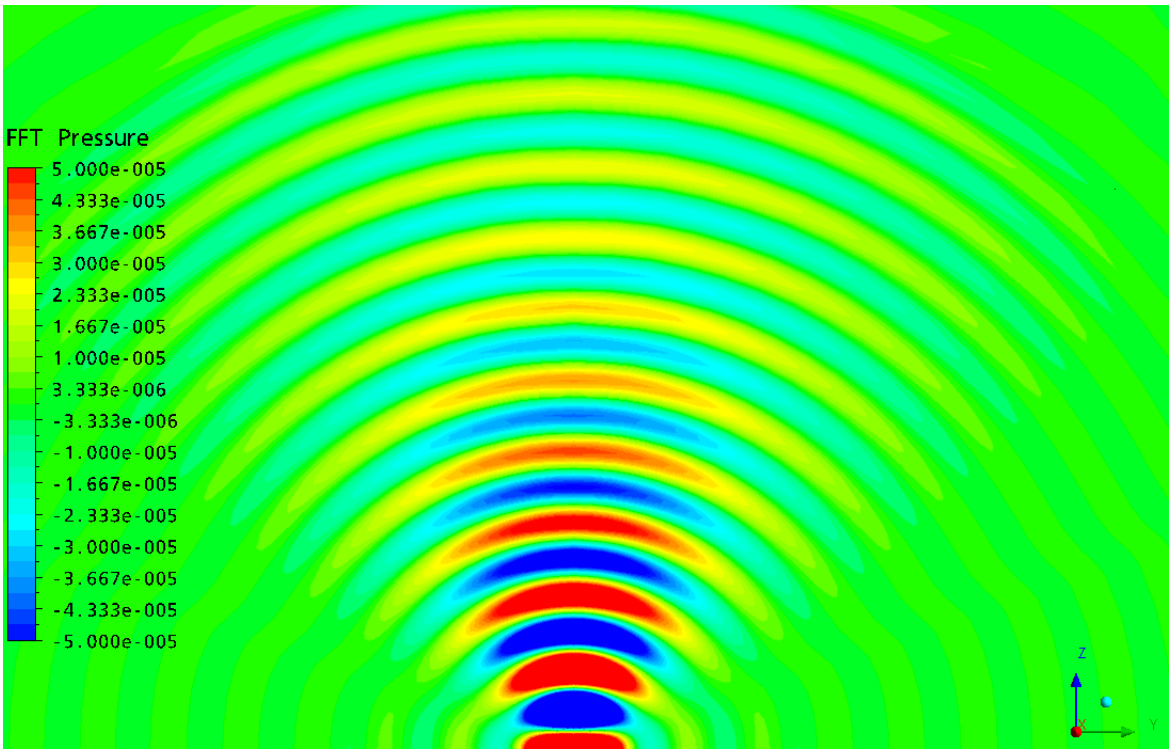


Figure 6-38: Pressure Pertubation Period 16

### 6.5.6 Very High Frequency

Lastly, a very high frequency test was conducted. In this case, the period was chosen as 8. With a period of 8, two distinct lobes should show up in the polar results. However, due to the grid meshing, the main waves may not be fully resolved. The results are shown below in Figures 6-44 through 6-47.

This test especially pushes the limits of the computational solver. This frequency

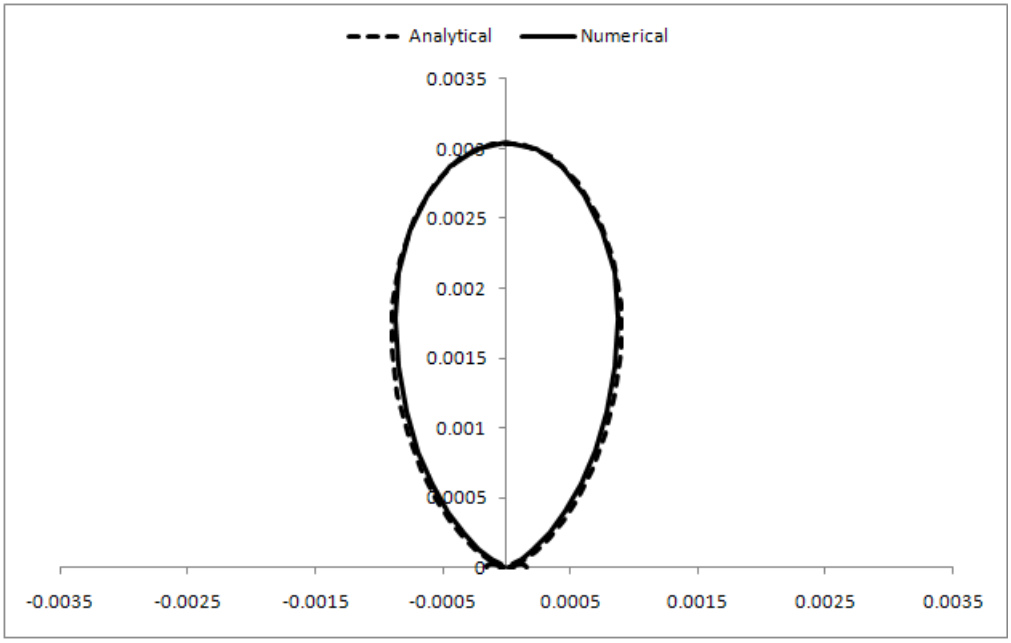


Figure 6-39: T=16 Test, Polar Amplitude

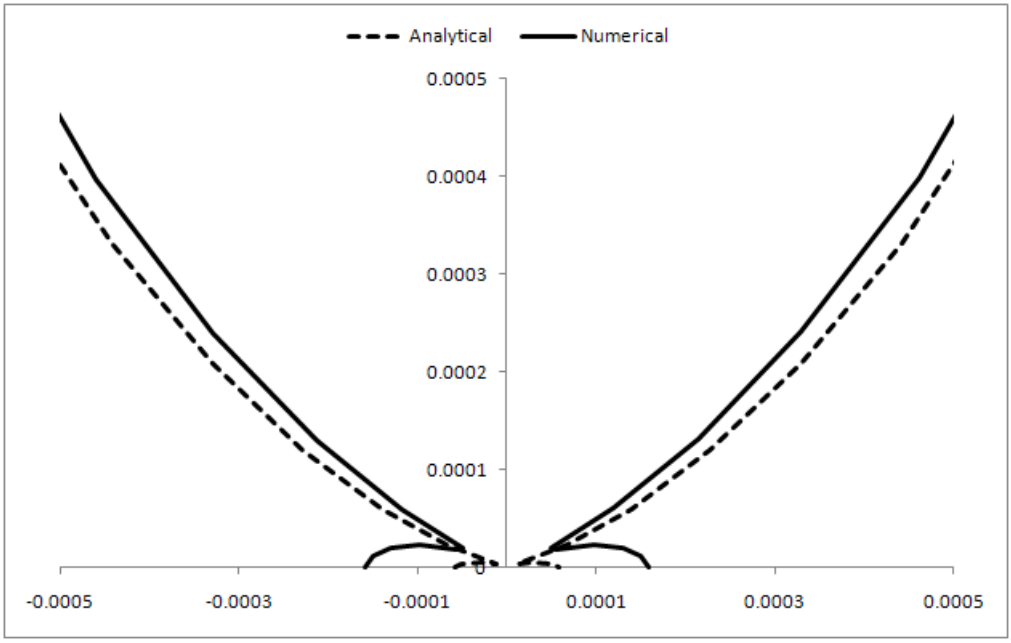


Figure 6-40: T=16 Test, Polar Amplitude, Close-Up

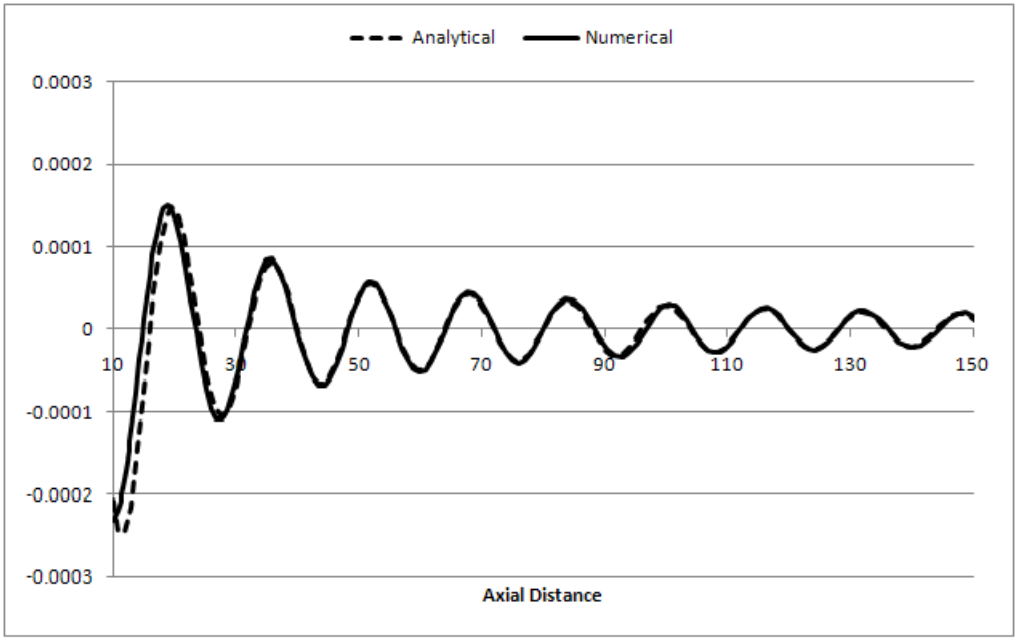


Figure 6-41: T=16 Test, Axial Real Component

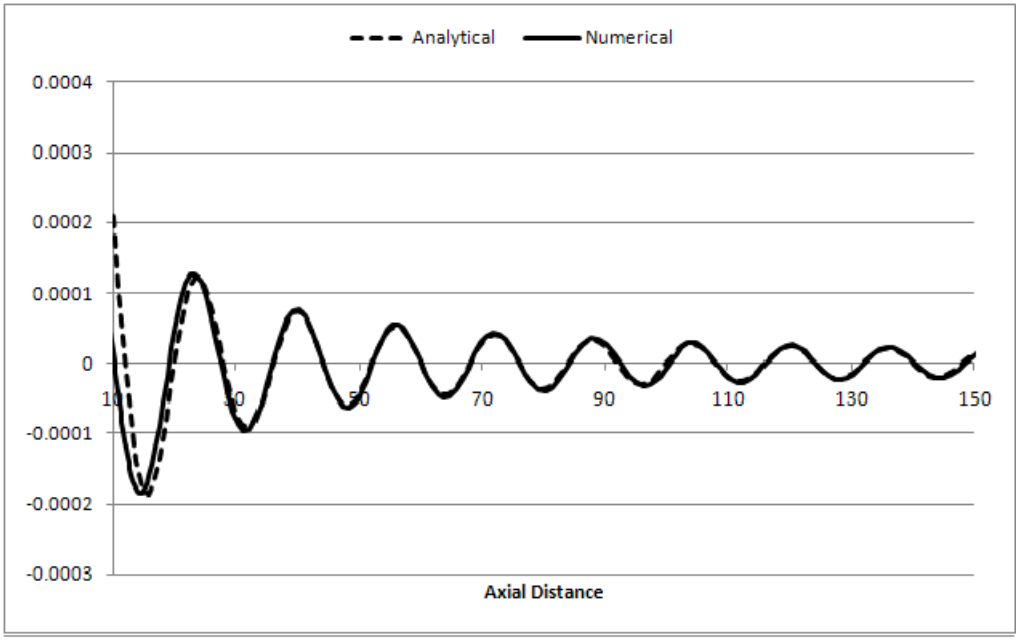


Figure 6-42: T=16 Test, Axial Imaginary Component

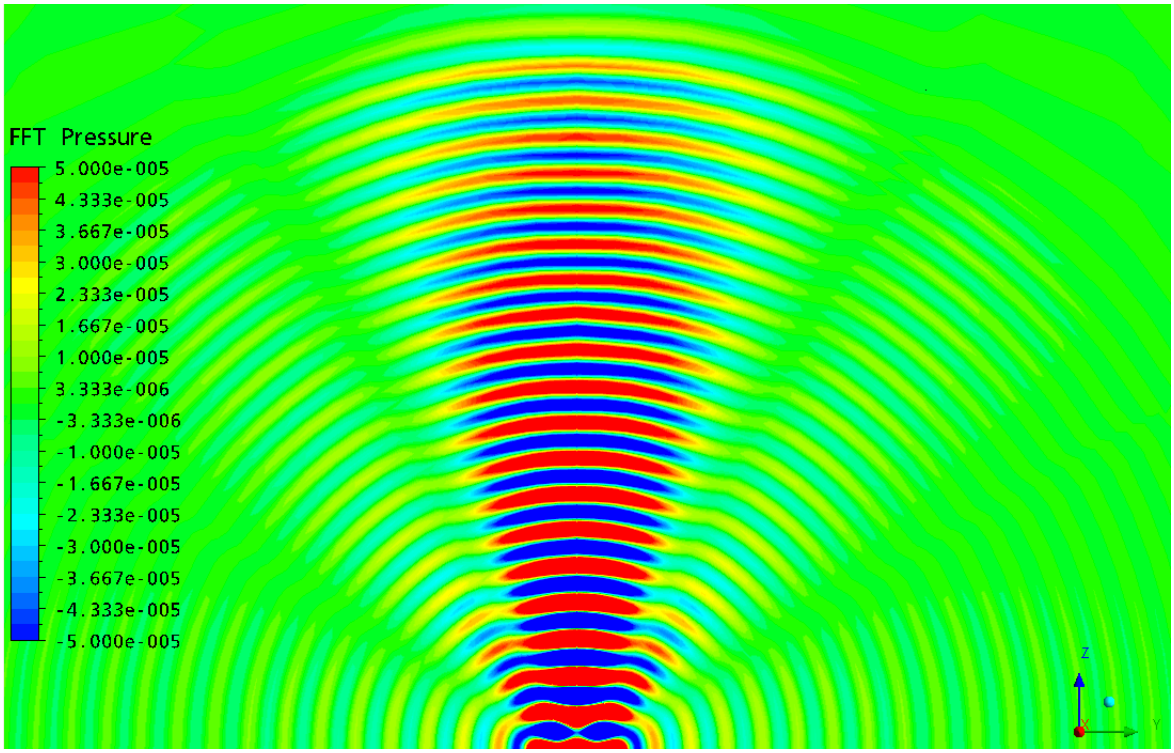


Figure 6-43: Pressure Pertubation Period 8

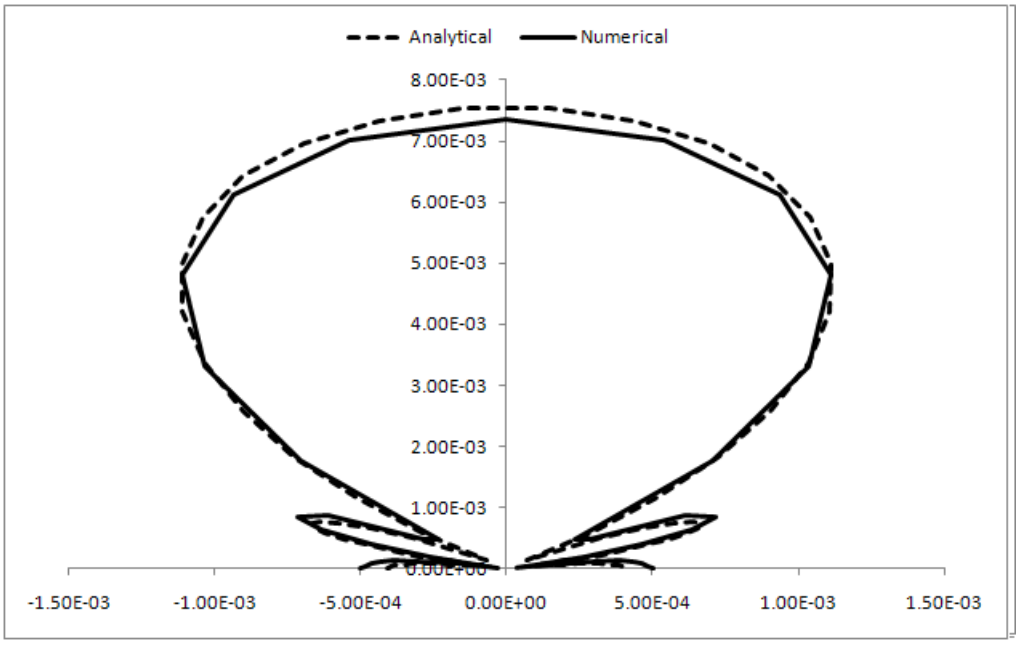


Figure 6-44: T=8 Test, Polar Amplitude

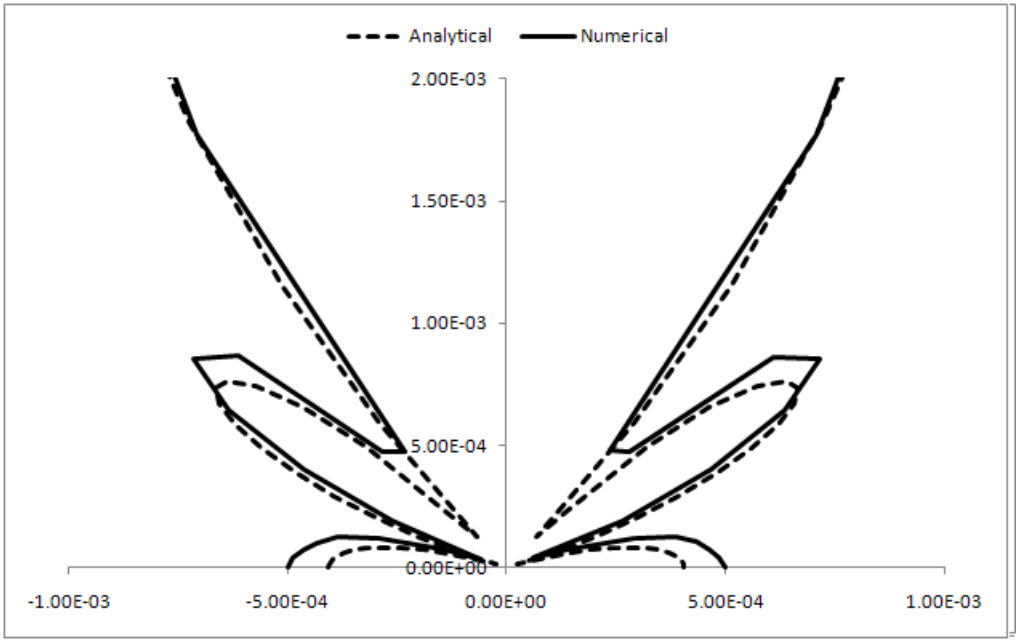


Figure 6-45: T=8 Test, Polar Amplitude, Close-Up

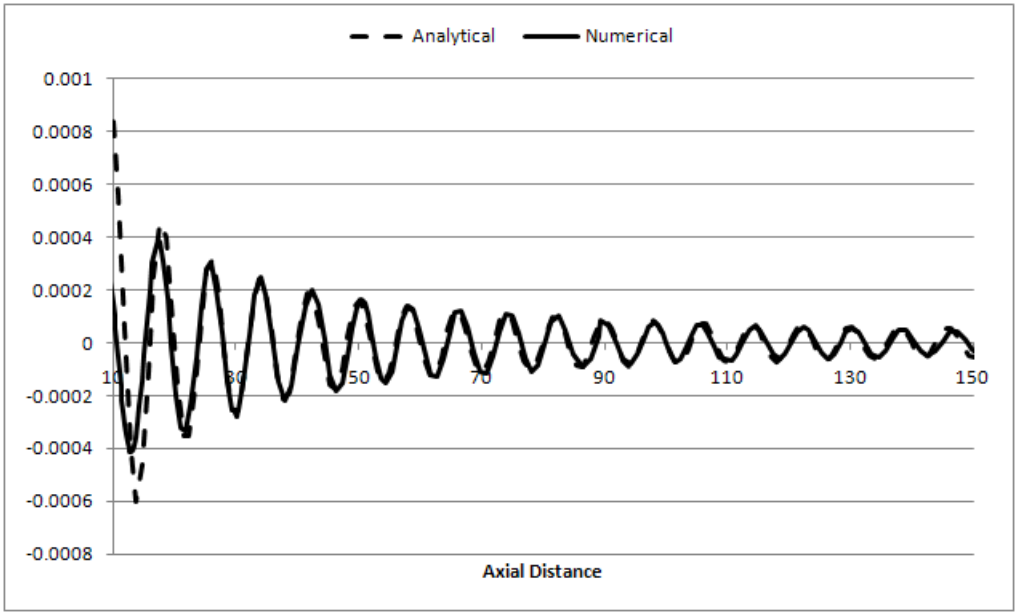


Figure 6-46: T=8 Test, Axial Real Component

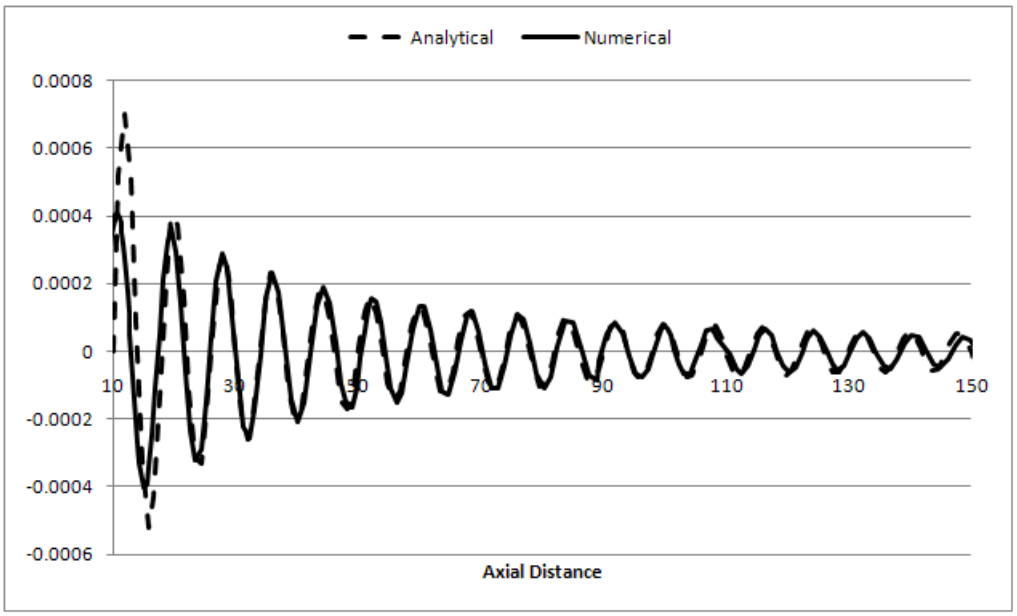


Figure 6-47: T=08 Test, Axial Imaginary Component

is near the edge of what the grid can numerically resolve. Because the results shown are so positive, it speaks well about the robustness of the flow solver and the grid motion.

### 6.5.7 Redefinition of Piston Edge

While attempting to analytically define the edge blend, the process lends itself towards starting the edge blend *inside* of the previously defined piston edge, and ramping out in such a way that the effective radius becomes the originally desired radius.

To determine the optimum radius, the original volume calculation for the blending region given in Appendix A. This edge volume is added onto the interior volume and equated to the volume of an ideal piston. The equation to solve is

$$\pi r^{*^2} = \pi \left[ r_1(r_1 - r_0) - \frac{7}{10}(r_1 - r_0)^2 + r_0^2 \right] \tag{6.13}$$

It has been determined that with an edge blend of less than one, the numerical solution approaches the effective radius solution, so define $r_1 = r_0 + 1$ and define the effective radius to be solved for.

$$\pi(10)^2 = \pi \left[ (r_0 + 1)(r_0 + 1 - r_0) - \frac{7}{10}(r_0 + 1 - r_0)^2 + r_0^2 \right] \tag{6.14}$$

Simplifying this equation gives the polynomial

$$r_0^2 + r_0 - 99.7 = 0 \tag{6.15}$$

This can be solved quickly by completing the square, by adding 99.95 to both sides, as

$$r_0^2 + r_0 + 0.25 = 99.95 \tag{6.16}$$

then

$$(r_0 + 0.5)^2 = 99.95 \tag{6.17}$$

Which gives an edge starting radius of $r_0 = \sqrt{99.95} - 0.5$ or approximately 9.497500 and of course an outer edge radius of 10.497500.

**Low Frequency** In the results shown in Figure 6-48 the analytical, r0 data series is the analytic data for a piston radius of 9.497500, while the r' series data is for the analytic data for a piston radius of 10. Note that at last, a solution has been confirmed against the original problem description, which uses a piston radius of 10.

**High Frequency** Running redundant and multiple test cases are a necessity for any computational fluid dynamics user. Too often will the test case picked be the one test case that a new feature works for. Because of this, the same process was repeated for the high frequency case. The pressure plot is shown below in Figure 6-49 and 6-50.

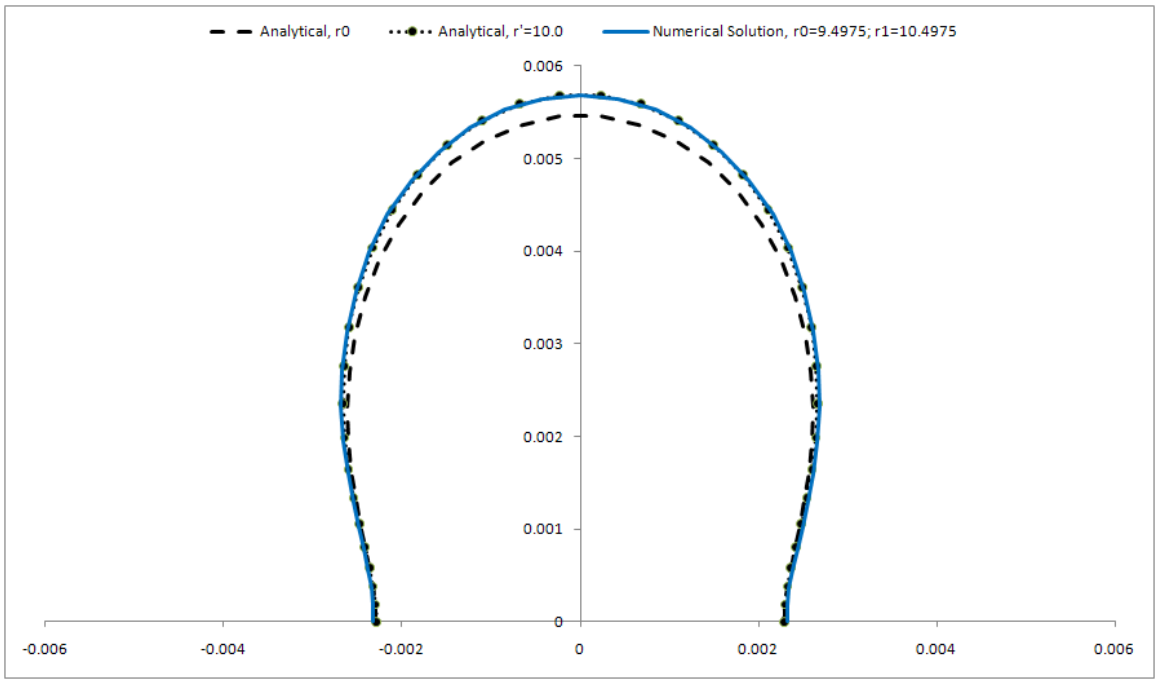


Figure 6-48: Period 25 Results for Volume Matched Blending Region

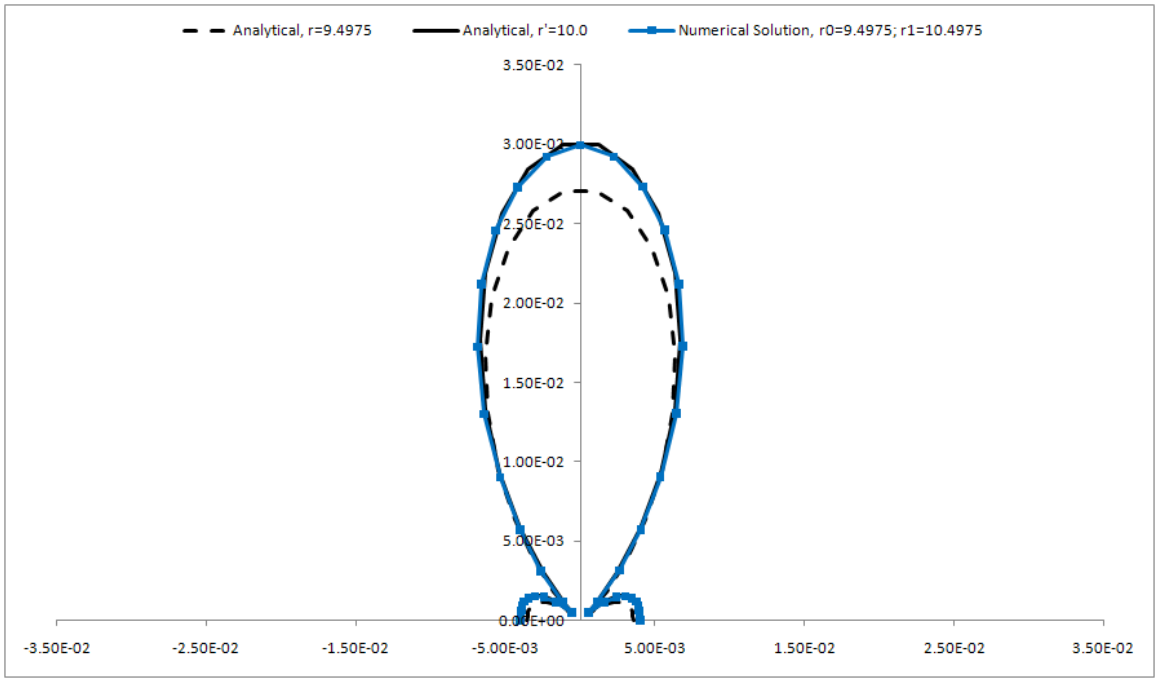


Figure 6-49: Period 12 Results for blend region having $r' = 10.0$

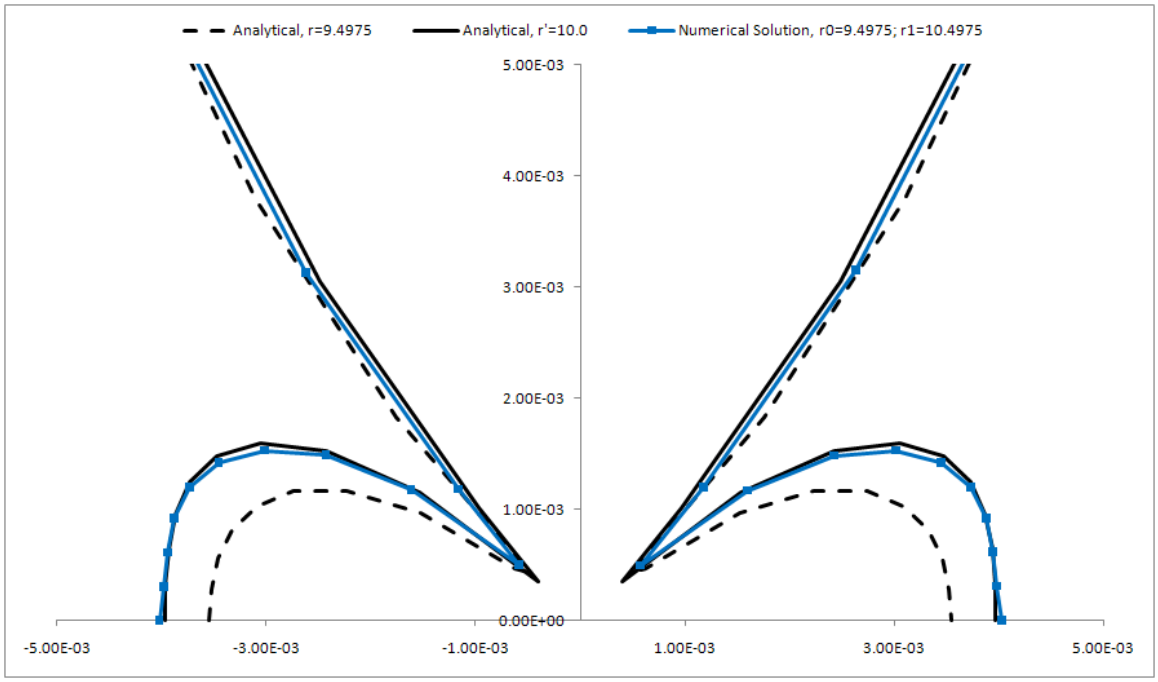


Figure 6-50: Period 12 Results for blend region having $r' = 10.0$ - Close up

As shown, the results match very well to the analytic solution for the equivalent piston radius.

**Very High Frequency** One last test, the very high frequency case, will prove that the solution obtained from the piston can be determined by definition of the piston and blending region. Figures 6-51 and 6-52 show the results.

As shown, once again the numerical solution obtained closely matches the analytic solution for the predicted equivalent piston radius.

Based on these results, given discontinuous boundary conditions, one can safely generate a mesh with confidence that the analytic solution can be shifted to account for the blending region.

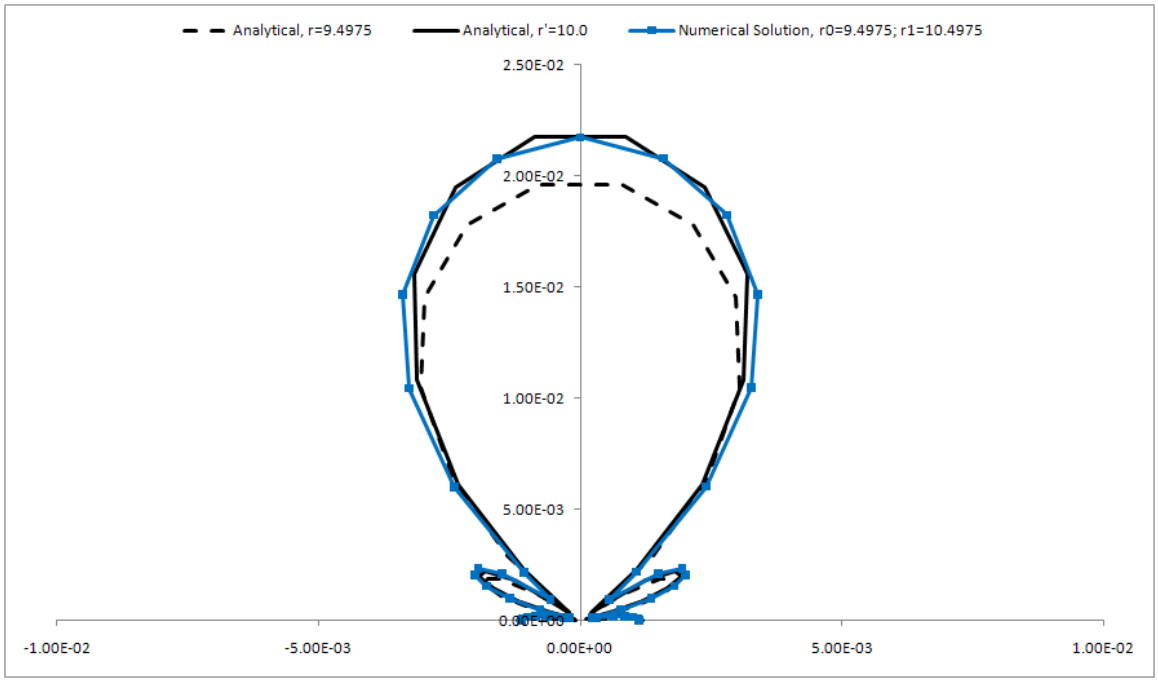


Figure 6-51: Very High Frequency Test using effective radius of 10

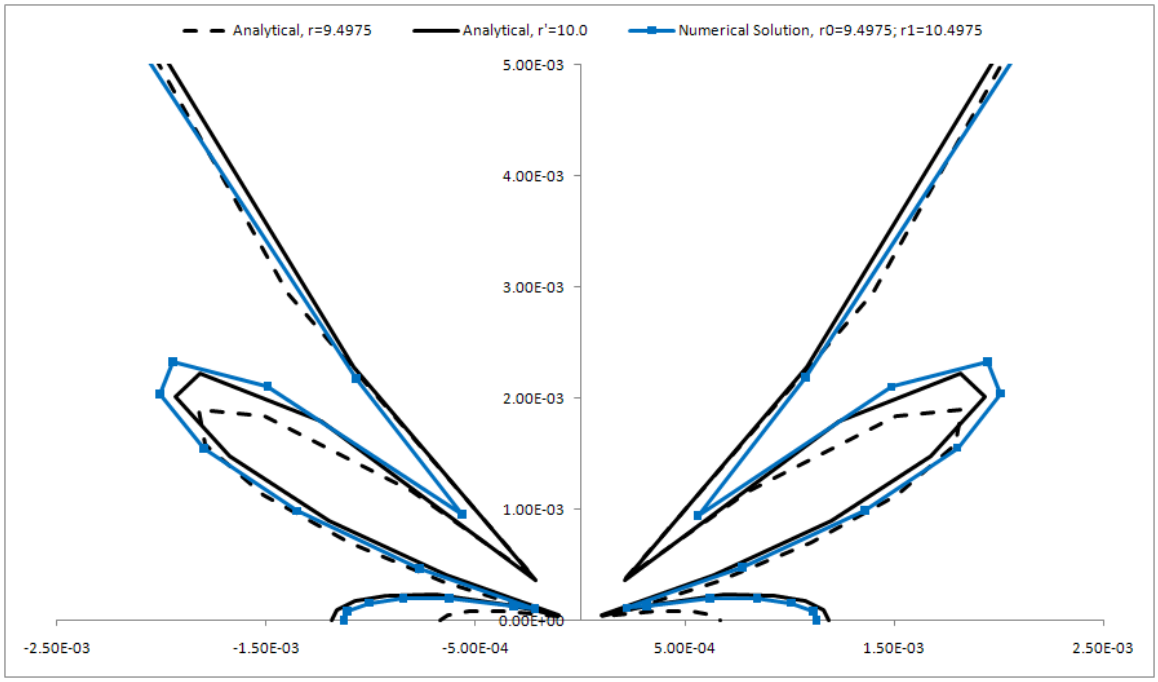


Figure 6-52: Very High Frequency Test using effective radius of 10 - Close Up

# Chapter 7

# Conclusions

## 7.1 Piston Test Case Lessons Learned

There were several lessons learned in the course of solving the piston test case. Many of these lessons can be applied to future problems, and will assist in solving aeroacoustic problems with moving meshes.

**Surface Definition** One of the first noticable effects was visible through the axial FFT plots. High frequency spurious waves were being generated at the piston surface. As the grid stretched radially away from the piston, these waves were no longer able to be resolved. However, it did bring to light a valuable lesson.

It was determined through FFT analysis that in order to keep off-order frequencies an order of magnitude below the frequency of interest, at least 32 points per wavelength must be defined.

Realizing this in practice could be difficult. For a typical aeroacoustic analysis, the acoustic frequencies are not known *a priori*. In many cases, because of this fact, the mesh is conservatively dense to ensure proper wave resolution.

Based on this fact, if the surface is then not defined, or updated properly, high frequency waves could be generated by the surface. This again due to poor temporal

resolution of the surface. Recall Subsection 6.5.1.2 on how discrete definition of surfaces can generate high frequency waves.

Knowing this challenge, perhaps a module could be implemented that examines outflow acoustic waves for frequency. The grid definition can then be adjusted at a frequency that can not only properly resolve the acoustic waves, but help to eliminate the spuriouis high frequency waves.

**Edge Blend (Effective Radius)** The effect of the piston edge blend is the most pronounced side effect seen during the course of the piston test. It was shown that the effect of edge blend has a drastic influence on the results, by giving the piston an effective radius, larger than initially expected.

Ribner's dilitational theory assisting in developing a volumetric equivalent radius, which could be used to predict the effective radius. This effective radius, however, has a limit. It seems that if the blending region is over 10% of the piston radius, the effective radius calculation breaks down, and diminishing returns occur. It could be that for large blending regions, wave propagation off blend region becomes too much directed outwards, rather than upwards.

The lesson learned is simply that careful consideration must be taken both during meshing, and runtime in solving problems where discontinuous boundaries occur.

## 7.2 Summary

The results shown in the last few sections have been the culmination of years of work. The seemingly simple problem of predicting the acoustic waves propagating from a plunging piston has been solved, with favorable results.

During the course of these tests, several interesting things showed up. First, the effect of some equivalent piston became quite apparent. This fact seems trivial and is hard to deny.

At large, and even somewhat small, piston blends, the equivalent piston correlation did not hold up. The volumetric equivalent piston radius, $r^*$, and the actual numerically best-fit piston radius, $r'$ where often times upwards of 10% in error. Since the radius gets squared for the amplitude calculation, and input into a Bessel function for shaping, this term is as important as any variable in the solution.

Based on this fact, the errors obtained at the beginning were simply unacceptable. However, it was realized that when the piston radius approaches and becomes smaller than 10% of the piston radius, the effects of the piston radius could be estimated by the volumetric equivalent calculation.

Many mistakes were made during the course of this project. In addition to mistakes several problems arose. One ideally hopes that a project such as this goes off without a hitch.

James Joyce said that, "A man's errors are his portals of discovery". No quote better sums up this project. All of the mistakes, problems, and numerical issues that arose over the course of this project simply gave way to a wealth of information and learning.

Had the piston not originally been defined with too few points, the importance of defining the surface would not have been found. Had the original piston edge been originally defined properly and small enough, the piston edge blend phenomenon would not have been discovered.

Lastly and most importantly, the BASS Code has been demonstrated to produce excellent aeroacoustic results involving moving surfaces and meshes. This opens the door to new and exciting work in the field of computational aeroacoustics which involve moving surface.

## 7.3 Future Work

There are several things that could be looked at in the immediate future should this project continue. Firstly, there are several meshing changes that could be made. The most pressing change would be better definition at the piston edge. This could allow the user to continue to decrease the edge of the piston. It would be helpful to see where exactly the residual benefits drop of in terms of decreasing the blend size. This information could be used for not only piston work, but any future activity that involves discontinuous boundary conditions.

Secondly, the future user could examine plots from this report and see what maximum radius is needed for proper solution resolution. The current grid had grid resolution out to a radius of 150, and then blended the grid radially to 200. If only a radius of 100 is needed, then that could significantly cut down the number of grid points needed. This could do one of two things. The first benefit would be a direct savings in computational time.

More interesting though, it would allow the user to replace the unnecessary grid points at the boundaries and add them to the domain. Better grid refinement would allow the user to go to frequencies higher than what was calculated here.

In addition, different oscillating shapes and boundary conditions could be examined. It would be interesting to see if the dilitational analogy could be expanded inward such that the piston becomes a gaussian-type shape, with no flat top.

# Chapter 8

# Looking Forward

Grid Warp and its integration into the BASS Code is hardly mature. It seems as though as hardware becomes exponentially faster, engineers and scientists think of problems exponentially more difficult. Hardware struggles to keep up, and the cycle of maintaining computational efficiency perpetuates.

Before thinking of of speeding up the time needed to deform a grid point, it should first determined whether or not the grid point even *needs* to be deformed. Currently each block has a flag which tells the solver whether or not to deform that particular block. If the flag is off, then that block will not deform, even if a neighbor block does, and if the flag is on, then the entire computational process will take place, even if diplacement is 0.0 for each grid point.

## 8.1 The Black Box

In the spirit of object-oriented programming, the programmer or developer wishes to eliminate as much user input as possible. It has been hypothesized that the industry average is between 15-50 errors (bugs) per 1000 lines of computer code [35]. Many CFD runs can contains hundreds of grid blocks. While a seemingly simple input file is not as difficult to write as a subroutine, intermittent errors will occur. Given

*maximum* displacements can be order of magnitudes smaller than the grid spacing, debugging a flow field with a problem like that can could be excruciating.

One simple future enhancement with this in mind is letting the BASS Code decide which blocks need moved. Physical constants, particularly dist_far, have been established which govern the grid deformation. Since Grid Warp will deform no grid points which lie at a distance greater than dist_far, why even check grid points which lie inside a *block* farther away than dist_far?

The idea is almost trivial to simply have the code check the block corners distance. If none of the block corners are within dist_near, then the block isn't either. Unfortunately, grid blocks can be severely curved, and for the sake of generality, the code might as well check the centers of the faces as well. This simple eight point check, could potentially computationally save thousands of distance calculations. In this way, eliminating a potential error causing ingredient (the user) is eliminated, all the while maximizing efficiency.

There exists a clear path to the first issue: determining which grid points need deformed. However, the second task is much more complex, but much more interesting at the same time. Currently, the deformation of grid points is fairly computationally efficient. Grid points require no connectivity information, and after a surface target point is found, the deformation is algebraic. Unfortunately, finding the surface target point is incredibly computationally expensive, as a global search is done to find the closest point. It would be nice if not only the deformation, but the search as well was a simple algebraic solution.

## 8.2 Rigid Surfaces

There are many CFD analogies to numerical finite element methods (FEA). Each starts with meshing a surface or volume, supplies inputs to the solver, applies bound-

ary conditions and solves. In FEA, often times there is a desire to see how multiple bodies react with each other. In order to do this, typically contact elements are needed to be created between the parts. In the absolute simplest case, these elements can simply be compression-only "springs between the two bodies. However, for more complex analyses, these elements can includes effects such as friction and adhesion, among other things.

For the purpose of explaining these elements, an fictitious two-dimensional analysis will be considered. Lets assume that a circle is in contact with a flat plate. In two-dimensions bodies are surfaces, and the contact will happen between lines.

What typically happens is the following. The solid body of interest is created and meshed. Contact elements are overlaid on the exterior surfaces. Target elements are overlaid on the opposite body. During the solution, the solver will check to see if any target nodes have penetrated the contact elements.

If any of these nodes have in fact penetrated the contact elements, then a normal pressure is applied to those interfering nodes until the penetration has been eliminated. In order to see if the nodes have penetrated, the solver must check each node against each element.

Sound familiar?

This contact global search increases solution time by orders of magnitude. In many situations though, the engineer can use what is called rigid target surfaces. If the user knows that the surfaces are simple and essentially do not deform, then rather than being meshed, the target surfaces can be replaced by algebraic curves and surfaces.

In two-dimensions, the target surfaces can be: lines, arcs, parabolas and circles. In three-dimensions, the surfaces can be described as quadrilaterals, triangles, and even cylinders, cones and spheres. Each of these are described algebraically. Now, rather than the solver doing a global search on the elements, the distances and penetrations

can be determined algebraically.

### 8.2.1 Extension to CFD

This approach can be directly applied to the grid deformation this paper has been trying to achieve. Currently, AFRL Grid Warp requires a meshed deforming surface. What if the surface could be described algebraically?

The procedure to calculate the distance from a point to a plane is taken from Stewart [53]. The equation of a plane is

$$ax + by + cz + d = 0 \tag{8.1}$$

and a point $\vec{x_0} = (x_0, y_0, z_0)$, the normal to the plane is given by

$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{8.2}$$

and a vector from the plane to the point is given by

$$\vec{w} = \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \tag{8.3}$$

Projecting $\vec{w}$ onto $\vec{v}$ gives the distance $D$ from the point to the plane as

$$
\begin{aligned}
D &= |\text{proj}_v \vec{w}| \\
&= \frac{|\vec{v} \cdot \vec{w}|}{|\vec{v}|} \\
&= \frac{|a(x - x_0) + b(y - y_0) + c(z - z_0)|}{\sqrt{a^2 + b^2 + c^2}} \\
&= \frac{|ax + by + czax_0by_0cz_0|}{\sqrt{a^2 + b^2 + c^2}} \\
&= \frac{|-d - ax_0by_0cz_0|}{\sqrt{a^2 + b^2 + c^2}} \\
&= \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}
\end{aligned}
\tag{8.4}
$$

Dropping the absolute value gives the signed distance, which tells the solver "which side the point is on with reference to the surface plane.

$$
D = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}} \tag{8.5}
$$

This simple formula immediately gives the distance from a grid point to a deforming plane. No global search needs to be done. Since the plane doesnt change, the denominator could be calculated just once, even further speeding up computation time.

Spheres and circles are simply the loci of points (in three and two dimensions respectively) which lie equidistant from the center. With this in mind, the distance to a rigidly defined circle or sphere could be determined by a simple distance calculation to a single point.

A similar process could be extended to use such surfaces as ellipsoids, cones, paraboloids, and hyperboloids, and hyperbolic paraboloids.

Because of the novel approach Melville took to grid deformation and the successful implementation shown here, there are lots of opportunities for expansion of this work. Besides some ideas presented here, new and exciting ideas will be thought of

an implemented into AFRL GridWarp and the BASS Code even further reducing computational efficiency.

# References


[1] C. Allen. Parallel universal approach to mesh motion and application to rotors in forward flight. *International Journal for Numerical Methods in Engineering*, 69:2126–2149, 2007.

[2] H. Y. B.A. Robinson, J.T. Batina. Aeroelastic analysis of wings using the euler equations with a deforming mesh. Paper 90-1032, AIAA, 1990.

[3] J. Batina. Unsteady euler algorithm with unstructured dynamic mesh for complex-aircraft aeroelastic analysis. Paper 89-1189, AIAA, 1989.

[4] J. Batina. Unsteady euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis. *AIAA Journal*, 29(3), 1991.

[5] R. Bisplinghoff, H. Ashley, and R. Halfman. *Aeroelasticity*. Addison-Wesley Publishing Company, dover edition edition, 1996.

[6] R. L. Bisplinghoff and H. Ashley. *Principles of Aeroelasticity*. Dover Publications, dover phoenix edition edition, 2002.

[7] P. Cavallo, A. Hosangadi, R. Lee, and S. Dash. Dynamic unstructured grid methodology with application to aero/propulsive flowfields. *AIAA Paper 97-2310*, 1997.

[8] K. A. R. G. J. C.K.W. Tam, K.A. Kurbatskii. A numerical and experimental

investigation of the dissipation mechanisms of resonant acoustic liners. *Journal of Sound and Vibration*, 245:545–557.

[9] R. S. CKW Tam, NN Pastouchenko. On the two sources of supersonic jet noise. *AIAA Paper*, (2003-3163), 2003.

[10] R. Clarke, J. Burken, J. Bosworth, and J. Bauer. X-29 flight control system; lessons learned. Technical Memo NASA-TM-4598, NASA, 1994.

[11] P. Crumpton and M. Giles. Implicit time-accurate solutions of unstructured dynamic grid. *International Journal for Numerical Methods in Fluids*, 25:1285–1300, 1997.

[12] P. J. C.Y. Loh, L.S. Hultgren. Near field screech noise computation for an underexpanded supersonic jet by the cs/se method. *AIAA Paper*, (2001-2252).

[13] A. Dowling and J. F. Williams. Sound and sources of sound. *Journal of Acoustic Society of America*, 74:174–178, 1983.

[14] D. P. Eisman. Gridpro. www.gridpro.com. Product Development Company.

[15] L. Eriksson. Three-dimensional spline-generated coordinate transformations for grids around wing-body configurations. NASA Conference Proceedings 2166, Numerical Grid Generation Techniques, 1980.

[16] L. Eriksson. Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation. *AIAA Journal*, 20(10):1313–1320, 1982.

[17] L. Eriksson. *Transfinite Mesh Generation and Computer-Aided Analysis of Mesh Effects*. PhD thesis, Univerity of Uppsala, Uppsala, Sweden, 1984.

[18] F. Ford. *Introduction to acoustics*. Elsevier, 1970.

[19] M. Gharib and A. Roshko. The effect of flow oscillations and cavity drag. *Journal of Fluid Mechanics*, 177:501–530, 1987.

[20] M. Giles. Non-reflecting boundary conditions for euler equation calculations. *AIAA Journal*, 28:2050, 1990.

[21] M. Giles. Non-reflecting boundary conditions for unsteady airfoil calculations. In *Proceedings of Third International Conference on Hyperbolic Problems*, 1990.

[22] W. Gordon and C. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7:461–477, 1973.

[23] M. J. Greaves and B. A. Petersson. Performance of acoustic liners for jet engine intake ducts. *The Journal of the Acoustical Society of America*, 105(2):1141–1141, 1999.

[24] R. Hixon, M. Nallassamy, and S. Sawyer. A method for the implementation of boundary conditions in high-accuracy finite-difference schemes. In *AIAA Paper*, volume 2005-0608, January 2005.

[25] R. Hixon, S.-H. Shih, and R. Mankbadi. Evaluation of boundary conditions for the gust-cascade problem. In *NASA CR-19980208671*, November 1998.

[26] M. O. L. E. J. Larsson, L. Davidson. Aeroacoustic investigation of an open cavity at low mach number. *AIAA Journal*, 42:2462–2473, 2004.

[27] G. W. Jianwei Hu, Ligang Liu. Dual laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds*, 18:271–277.

[28] N. P. W. Joe. F. Thompson, Bharat K. Soni. *Handbook of Grid Generation*. CRC Press, 1999.

[29] W. Johnson. Elementary applications of a rotorcraft dynamic stability analysis. Technical Memo NASA-TM-X-73161, NASA, 1976.

[30] L. Kinsler, A. Frey, A. Coppens, and S. V.J. *Fundamentals of acoustics*. John Wiley and Sons, 3rd edition, 1982.

[31] M. Lighthill. On sound generated aerodynamically. i. general theory. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 211(1107):546–587, 1952.

[32] R. Magnus and H. Yoshihara. Unsteady transonic flows over an airfoil. Technical report.

[33] T. Manning and S. Lele. A numerical investigation of sound generation in supersonic jet screech. *AIAA Paper*, (2000-2081).

[34] T. Manning and S. Lele. Numerical simulations of shock vortex interactions in supersonic jet screech. *AIAA Paper*, (1998-0282).

[35] S. McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, second edition, 2004.

[36] R. Melville. Dynamic aeroelastic simulation of complex configurations using overset grid systems. AIAA Paper 2000-2341, Applied Aerodynamics Conference, Denver, CO, 2000.

[37] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 38, 1965.

[38] R. Motsinger and R. Kraft. Design and performance of duct acoustic treatment. *in Aeroacoustics of Flight Vehicles: Theory and Practice*, 2(NASA RP-1258):165–206, 1991. Edited by H.H. Hubbard.

[39] E. Nielson and W. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. *AIAA Journal 2002*, 40(6):1155–1163, 2002.

[40] M. Norton and D. Karczub. *Fundamentals of Noise and Vibration Analysis for Engineers*. Cambridge University Press, 2nd edition, 2003.

[41] R. Ohayon. Reduced models for fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering*, 60:139–152, 2004.

[42] A. Pierce. *Acoustics: An Introduction to its Physical Principles an Applications*. Acoustical Society of America, Woodbury, NY, 1981. Chapters 4 and 5.

[43] R. Rausch, J. Batina, and H. Yang. Three-dimensional time-marching aeroelastic analyses using an unstructured euler method. Technical Memo TM-107567, NASA, 1992.

[44] D. Reynolds. *Engineering pricniples of acoustics - noice and vibration*. McGraw-Hill, 1981.

[45] H. Ribner. Aerodynamic sound from fluid dilatations. Technical Report 86, University of Toronto, Institute of Aerophysics, 1962.

[46] H. Ribner. The generation of sound by turbulent jets. *Advances in Applied Mechanics*, 8:104–182, 1964.

[47] H. Ribner. Effects of jet flow on jet noise via an extension to the lighthill model. *Journal of Fluid Mechanics*, 321:1–24, 1996.

[48] J. Rossiter. Wind tunnel experiments on the flow over rectanguular cavities at subsonic and transonic speeds. Tecnical Report 64037, Royal Aircraft Establishment, 1964.

[49] C. Rowley. *Modeling, simulation and control of cavity flow oscillations*. PhD thesis, California Institute of Technology, 2002.

[50] H. Shen and C. K. Tam. Effects of jet temperature and nozzle-lip thickness on screech tones. *AIAA Journal*, 38:762–767.

[51] H. Shen and C. K. Tam. Numerical simulation of the generation of axisymmetric mode jet screech tones. *AIAA Journal*, 36:1801–1807.

[52] H. Shen and C. K. Tam. Three-dimensional numerical simulation of the jet screech phenomenon. *AIAA Journal*, 40:33–41.

[53] J. Stewart. *Calculus*. Brooks/Cole Publishing Company, fourth edition, 1999.

[54] C. Tam and K. Kurbatskii. Microfluid dynamics and acoustics of resonant liners. *AIAA Journal*, 38:1331–1339.

[55] C. K. Tam. Supersonic jet noise. *Annual Review of Fluid Mechanics*, 33:1788–1796.

[56] C. K. Tam. Compuational aeroacoustics: An overview of computational challenges and applications. *International Journal of Computational Fluid Dynamics*, 18(6):547–567, 2004.

[57] C. K. Tam and H. Shen. Direct computation of nonlinear acoustic pulses using high-order finite difference schemes. *AIAA Paper*, (93-4325).

[58] T. Tezduyar, M. Behr, M. Mittal, and A. Johnson. A new strategy for finite-element computations involving moving boundaries and interface–the deforming-spatial-domain/space-time procedure: I, the concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering*, 94:339–351, 1992.

[59] K. Thompson. Time-dependant boundary conditions for hyperbolic systems. *Journal of Computational Physics*, 68:1–24, 1987.

[60] K. Thompson. Time-dependant boundary conditions for hyperbolic systems ii. *Journal of Computational Physics*, 87:439–461, 1990.

[61] M. B. W. D. W. DeRoeck, G. Rubio. Toward accurate hybrid precition techniques for cavity flow noise applications. *International Journal for Numerical Methods in Fluids*, 61:1363–1387, 2009.

[62] H. Y. WF Ballhaus, R Magnus. Some examples of unsteady transonic flows over airfoils. Technical Report 19760027392, NASA, 1975.

# Appendix A

# Derivation of Blending Volume

The volume of a cylinder can be given by the triple integral

$$V = \int_0^{2\pi} \int_0^{r} rh \, dr \, d\theta \tag{A.1}$$

In this case, the volume of the piston is trivial, so only the volume of the blending region will be examined, from $r_0$ to $r_1$. In this region, the height is a function of radius, so the volume is then given by

$$V = \int_0^{2\pi} \int_{r_0}^{r_1} \left[ 1 - 3\left( \frac{r - r_0}{r_1 - r_0} \right)^2 + 2\left( \frac{r - r_0}{r_1 - r_0} \right)^3 \right] r \, dr \, d\theta \tag{A.2}$$

This integral can be solved by using the substitution rule in addition to integration by parts. Integration by parts arises from the product rule of differentiation and in its simplest form is given by

$$\int u \frac{dv}{dx} dx = uv - \int v \frac{du}{dx} dx \tag{A.3}$$

In this case, integration by parts will need to be applied recursively. Second derivatives and integrals will needed of the $u$ and $v$ terms mentioned above. In this case, $u$ will

be $r$ and $v$ will be the height term. The final solution is then simply

$$V = 2\pi \left[ r \int h(r)\, dr - \frac{d}{dr} r \iint h(r)\, dr \right] \tag{A.4}$$

The task is then to integrate the blending function, and then again. Substition will be used, letting

$$u = \frac{r - r_0}{r_1 - r_0} \tag{A.5}$$

It then follows that

$$\begin{aligned} \frac{du}{dr} &= \frac{1}{r_1 - r_0} \\ dr &= (r_1 - r_0) du \end{aligned} \tag{A.6}$$

Substituting this in gives

$$\int (1 - 3u^2 + 2u^3)(r_1 - r_0)\, du \tag{A.7}$$

The solution of which is

$$\int h(r)\, dr = (r_1 - r_0) \left[ \left( \frac{r - r_0}{r_1 - r_0} \right) - \left( \frac{r - r_0}{r_1 - r_0} \right)^3 + \frac{1}{2} \left( \frac{r - r_0}{r_1 - r_0} \right)^4 \right] \tag{A.8}$$

Substitution is used again to calculate the second integral

$$\iint h(r)\, dr = (r_1 - r_0)^2 \left[ \frac{1}{2} \left( \frac{r - r_0}{r_1 - r_0} \right)^2 - \frac{1}{4} \left( \frac{r - r_0}{r_1 - r_0} \right)^4 + \frac{1}{10} \left( \frac{r - r_0}{r_1 - r_0} \right)^5 \right] \tag{A.9}$$

The volume is then given by

$$\begin{aligned} V = 2\pi \Bigg[ & r(r_1 - r_0) \left( \left( \frac{r - r_0}{r_1 - r_0} \right) - \left( \frac{r - r_0}{r_1 - r_0} \right)^3 + \frac{1}{2} \left( \frac{r - r_0}{r_1 - r_0} \right)^4 \right) - \\ & (r_1 - r_0)^2 \left( \frac{1}{2} \left( \frac{r - r_0}{r_1 - r_0} \right)^2 - \frac{1}{4} \left( \frac{r - r_0}{r_1 - r_0} \right)^4 + \frac{1}{10} \left( \frac{r - r_0}{r_1 - r_0} \right)^5 \right) \Bigg] \end{aligned} \tag{A.10}$$

Putting limits on the problem conveniently simplifies the problem. At $r_0$ all terms cancel to zero. At $r_1$ the $u$ term defined above becomes 1. The final volume from $r_0$ to $r_1$ is then given by:

$$V = \pi \left[ r_1(r_1 - r_0) - \frac{7}{10}(r_1 - r_0)^2 \right] \tag{A.11}$$

# Appendix B

# Transfinite Interpolation (TFI)

# Further Reading

At this time please do note that much of this appendix is directly paraphrased from Thompson [28] whose Grid Generation book remains a near-sacred book on the topic. The general expressions of the univariate interpolations for three dimensions are

$$\begin{aligned}
\mathbf{U}(\xi,\eta,\zeta) &= \sum_{i=1}^{L}\sum_{n=0}^{P}\alpha_i^n(\xi)\frac{\partial^n \mathbf{X}(\xi_i,\eta,\zeta)}{\partial\xi^n} \\
\mathbf{V}(\xi,\eta,\zeta) &= \sum_{j=1}^{M}\sum_{m=0}^{Q}\beta_j^m(\eta)\frac{\partial^m \mathbf{X}(\xi,\eta_j,\zeta)}{\partial\eta^m} \\
\mathbf{W}(\xi,\eta,\zeta) &= \sum_{k=1}^{N}\sum_{l=0}^{R}\gamma_k^l(\zeta)\frac{\partial^l \mathbf{X}(\xi,\eta,\zeta_k)}{\partial\zeta^l}
\end{aligned} \tag{B.1}$$

Conditions on the blending functions are

$$\begin{aligned}\frac{\partial^{\bar{n}}\alpha_{\bar{i}}^{n}(\xi_i)}{\partial\xi^{\bar{n}}} &= \delta_{i\bar{i}}\delta_{n\bar{n}}\\ \frac{\partial^{\bar{m}}\beta_{\bar{j}}^{m}(\eta_j)}{\partial\eta^{\bar{m}}} &= \delta_{j\bar{j}}\delta_{m\bar{m}}\\ \frac{\partial^{\bar{l}}\gamma_{\bar{k}}^{l}(\zeta_k)}{\partial\zeta^{\bar{l}}} &= \delta_{k\bar{k}}\delta_{l\bar{l}}\end{aligned} \tag{B.2}$$

$$\bar{i} = 1, 2, \cdots, L \quad \bar{j} = 1, 2, \cdots, M \quad \bar{k} = 1, 2, \cdots, N$$

$$\bar{n} = 1, 2, \cdots, P \quad \bar{m} = 1, 2, \cdots, Q \quad \bar{l} = 1, 2, \cdots, R$$

The tensor products are

$$\begin{aligned}\mathbf{UW} = \mathbf{WU} &= \sum_{i=1}^{L}\sum_{k=1}^{N}\sum_{l=0}^{R}\sum_{n=0}^{P}\alpha_i^n(\xi)\gamma_k^l(\zeta)\frac{\partial^{ln}\mathbf{X}(\xi_i,\eta,\zeta_k)}{\partial\zeta^l\partial\xi^n}\\ \mathbf{UV} = \mathbf{VU} &= \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{m=0}^{Q}\sum_{n=0}^{P}\alpha_i^n(\xi)\beta_m^j(\eta)\frac{\partial^{nm}\mathbf{X}(\xi_i,\eta_j,\zeta)}{\partial\eta^m\partial\xi^n}\\ \mathbf{VW} = \mathbf{WV} &= \sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{m=0}^{Q}\sum_{l=0}^{R}\beta_j^m(\xi)\gamma_k^l(\xi)\frac{\partial^{lm}\mathbf{X}(\xi,\eta_j,\zeta_k)}{\partial\zeta^l\partial\eta^m}\\ \mathbf{UVW} &= \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=0}^{R}\sum_{m=0}^{Q}\sum_{n=0}^{P}\alpha_i^n\beta_j^m(\xi)\gamma_k^l(\xi)\frac{\partial^{lmn}\mathbf{X}(\xi_i,\eta_j,\zeta_k)}{\partial\zeta^l\partial\eta^m\partial\xi^n}\end{aligned} \tag{B.3}$$

The communicability in the above tensor products is assumed in most practical situations, but in general, it is not guaranteed. It is dependent upon the commutability of the mixed partial derivatives.

The Boolean sum of the three interpolations is

$$X(\xi,\eta,\zeta) = \mathbf{U}\oplus\mathbf{V}\oplus\mathbf{W} = \mathbf{U}+\mathbf{V}+\mathbf{W}-\mathbf{UV}-\mathbf{UW}-\mathbf{VW}+\mathbf{UVW} \tag{B.4}$$

## B.1 Recusion Formulation

The typical application of TFI as the boolean sum mentioned in Equation B.4 implies that each of the terms be evaluated individually followed by the sum.

Alternatively, TFI can be expressed as a three-step recursion formula. The first step is to express the univariate interpolation in one coordinate direction

$$\mathbf{X}_1(\xi,\eta,\zeta) = \sum_{i=1}^{L}\sum_{n=1}^{P} \alpha_i^n(\xi)\frac{\partial^n \mathbf{X}(\xi_i,\eta,\zeta)}{\partial \xi^n} \tag{B.5}$$

The second and third step each use the preceeding step as

$$\mathbf{X}_2(\xi,\eta,\zeta) = \mathbf{X}_1(\xi,\eta,\zeta) + \sum_{j=1}^{M}\sum_{m=0}^{Q} \beta_j^m(\eta)\left[\frac{\partial^m \mathbf{X}(\xi,\eta_j,\zeta)}{\partial \eta^m} - \frac{\partial^m \mathbf{X}_1(\xi,\eta_j,\zeta)}{\partial \eta^m}\right] \tag{B.6}$$

$$\mathbf{X}(\xi,\eta,\zeta) = \mathbf{X}_2(\xi,\eta,\zeta) + \sum_{k=1}^{N}\sum_{l=0}^{R} \gamma_k^l(\zeta)\left[\frac{\partial^l \mathbf{X}(\xi,\eta,\zeta_k)}{\partial \zeta^l} - \frac{\partial^l \mathbf{X}_2(\xi,\eta,\zeta_k)}{\partial \zeta^l}\right] \tag{B.7}$$

# Appendix C

# Thomas Algorithm

The Thomas algorithm is a simplified form of Gaussian elimination that can be used to solve a tridiagonal system of equations. A tridiagonal system for $n$ unknowns may be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \tag{C.1}$$

Where $a_1 = 0$ and $c_n = 0$. In matrix form, it looks like

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{Bmatrix} \tag{C.2}$$

The strategy uses an initial sweep that eliminates the $a$ terms, followed by a backward substitution which produces the solution. Denoting new modified coefficients with primes, the first step is

$$c'_i = \begin{Bmatrix} \frac{c_1}{b_1} & ; & i = 1 \\ \frac{c_i}{b_i - c'_{i-1} a_i} & ; & i = 2, 3, \ldots, n-1 \end{Bmatrix} \tag{C.3}$$

and

$$d'_i = \left\{ \begin{array}{lll} \frac{d_1}{b_1} & ; & i = 1 \\ \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} & ; & i = 2, 3, \ldots, n-1 \end{array} \right\} \tag{C.4}$$

The solution is then back substituted by

$$\begin{aligned} x_n &= d'_n \\ x_i &= d'_i - c'_i x_{i+1} \quad i = n-1, n-2, \ldots, 1 \end{aligned} \tag{C.5}$$