12-2002

Using the Boundary Element Method for Prediction of Sound Radiated from an Arbitrarily Shaped Vibrating Body

Mark J. Christensen
Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/masters_theses

Part of the Aerospace Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation
http://scholarworks.wmich.edu/masters_theses/1421

This Masters Thesis-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Master’s Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
USING THE BOUNDARY ELEMENT METHOD FOR PREDICTION OF SOUND RADIATED FROM AN ARBITRARILY SHAPED VIBRATING BODY

by
Mark J. Christensen

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Masters of Mechanical Engineering
Department of Mechanical and Aeronautical Engineering

Western Michigan University
Kalamazoo, Michigan
December 2002
The Boundary Element Method can be used to predict sound pressure levels radiated from an arbitrarily shaped vibrating body. Using the Direct, Indirect and Approximation formulation one can solve such an ‘exterior’ acoustic problem. In this work, the Direct formulation was chosen. The two major methods employed for implementation of this formulation are the Combined Helmholtz Integral Equation Formulation (CHIEF) and the Burton–Miller Method. Since the CHIEF method was easier to program and adapt to changes, it was decided that the CHIEF method would be the focus of this research. Using the CHIEF method as a guide, a computer simulation was developed that incorporated 1st order quadrilateral elements, calculates the sound pressure and sound power, is able to import surface velocities, and uses the program SDRC-IDEAS® as a pre-processor. This program was verified using theoretical models as well as experimental measurements conducted in the Western Michigan University Noise and Vibration Laboratory. This program was written in MATLAB® with the understanding that it can be processed on a personal computer.
ACKNOWLEDGEMENTS

I would like to begin to give thanks for the many people that have helped me finish my Master’s Project. First off I would like to thanks to God, for the blessing that me has bestowed upon me, and to my fiancée’ Angie for her patience and support.

Next, I would like to thank the members of my committee, Dr. Koorosh Naghshineh, Dr. Judah Ari-Gur, and Dr. Dennis VandenBrink for taking the time to answer my questions and reviewing my work. I would like to extend my special appreciation to Dr. Koorosh Naghshineh for supplying me with the tools, understanding and the giving me the guidance so that I may complete this work.

Mark J. Christensen
## TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................ ii
TABLE OF CONTENTS ........................................................................................................ iii
LIST OF FIGURES ............................................................................................................... v
LIST OF TABLES ................................................................................................................ vi
GLOSSARY OF SYMBOLS ................................................................................................. vii
INTRODUCTION ................................................................................................................ 1

### 1.0 PROBLEM DEFINITION

1.1 The Exterior Acoustic Problem .............................................................................. 2
1.2 Boundary Conditions ........................................................................................... 3
1.3 Integral Formulation .............................................................................................. 4

### 2.0 REVIEW OF LITERATURE ............................................................................ 7

2.1 Direct Method ........................................................................................................ 7
  2.1.1 Combined Helmholtz Integral Equation Formulation (CHIEF) .................... 7
  2.1.2 Burton-Miller Method ..................................................................................... 10

2.2 Indirect Method ................................................................................................... 11
  2.2.1 Elementary Method ......................................................................................... 12
  2.2.2 Kussmaul’s Indirect Boundary Element Formulation ................................... 12

2.3 Approximation Analysis ....................................................................................... 13

### 3.0 SCOPE OF WORK ......................................................................................... 17

### 4.0 Numerical Implantation .................................................................................. 18

4.1 Schenk’s Integration Formulation ........................................................................ 18

4.2 Surface Matrices .................................................................................................. 19
  4.2.1 Non-self Integration ......................................................................................... 20
  4.2.2 Self-Integration ............................................................................................... 24

4.3 Farfield Matrices ................................................................................................. 25

### 5.0 Program Validation ........................................................................................... 27

5.1 Uniformly Vibrating Sphere .................................................................................. 27
5.2 Uniformly Vibrating Cylinder .............................................................................. 36
5.3 Vibrating Box ....................................................................................................... 41
  5.3.1 Surface Vibrations ......................................................................................... 42
  5.3.2 Sound Power .................................................................................................. 44
  5.3.3 Predictive Sound Power Calculations ............................................................. 49
  5.3.4 CHIEFmat Results ......................................................................................... 51

### RECOMMENDATIONS ......................................................................................... 54

### CONCLUSION .......................................................................................................... 56

### REFERENCES .......................................................................................................... 57

### APPENDICES ......................................................................................................... 59

  Appendix A – CHIEFmat’s File Structure ................................................................. 60
  Appendix B – CHIEFmat primary Kernel ................................................................. 62
  Appendix C – Importing Surface Mesh ................................................................. 65
  Appendix D – Calculating Surface Matrices ........................................................... 73
LIST OF FIGURES

Figure 1: Exterior Acoustic Problem Definition ........................................................... 2
Figure 2: Geometry Description for the Acoustic Radiation Problem ......................... 5
Figure 3: Difference in the Number of CHIEF Points ................................................ 10
Figure 4: Approximation Method Problem Description ............................................. 14
Figure 5: Helmholtz Integral Pectoral Representation ............................................... 18
Figure 6: 3-D Element Mapped in a 2-D Local Coordinate System ......................... 20
Figure 8: Line Integration Example ........................................................................... 22
Figure 9: Self Integration Sub Region Mapping ....................................................... 25
Figure 10: Problem Definition for Uniformly Vibrating Sphere ............................... 28
Figure 11: Mesh of Sphere using SDRC-IDEAS ...................................................... 29
Figure 12: New Discritized Mesh ............................................................................ 30
Figure 13: Bar Graph and Histogram of Figure 12 ................................................. 30
Figure 14: Surface Pressure of Uniformly Vibrating Sphere ................................... 31
Figure 15: Surface Pressure of a Uniformly Vibrating Sphere Meshed Finer ............ 32
Figure 16: Surface Pressure of Uniformly Vibrating Cube ....................................... 33
Figure 17: Far Field Pressure of a Uniformly Vibrating Sphere ................................ 34
Figure 18: Far Field Pressure without any CHIEF points ........................................ 35
Figure 19: Problem definition of a line source ....................................................... 36
Figure 20: Cylinder modeled as a Line Source ......................................................... 38
Figure 21: Surface Pressure of Uniformly Vibrating Cylinder .................................. 39
Figure 22: Sound Pressure Level of a Uniformly Vibrating Cylinder with 10 CHIEF points .................................................................................................................. 40
Figure 23: SPL of a Uniformly Vibrating Cylinder without CHIEF points .............. 41
Figure 24: Vibrating Box ......................................................................................... 42
Figure 25: 1st Mode of Vibrating Box ...................................................................... 43
Figure 26: 2nd Mode of Vibrating Box ..................................................................... 43
Figure 27: 3rd Mode of Vibrating Box ...................................................................... 44
Figure 28: Reference Sound Source .......................................................................... 45
Figure 29: Sound Pressure Levels and Background Noise of Vibrating Box ............ 47
Figure 30: Measured Sound Power of Vibrating Box ............................................. 48
Figure 31: Mesh of Vibrating Box .......................................................................... 51
Figure 32: Sound Power Verification of Vibrating Box ........................................... 53
LIST OF TABLES

Table 1: Calibrated Sound Pressure Levels ............................................................... 46
Table 2: 1/3 Octave Band Frequency Table ............................................................... 49
GLOSSARY OF SYMBOLS

a  radius of a sphere
a(p)  complex-valued functions
A  surface matrix relation, area of the element
A_{mn}  surface matrix relation
A_{mn}^F  far field matrix relation
b(p)  complex-valued functions
B  surface matrix relation
B_{mn}  surface matrix relation
B_{mn}^F  far field matrix relation
c  speed of sound, wave propagation velocity
E  region exterior to the closed boundary
f  frequency
f_p  field point of interest
f(p)  complex-valued functions
G_{K(x,y)}  Green’s Function, fundamental solution of the Helmholtz equation
G_{o(x,y)}  fundamental solution of Laplace’s equation
i  imaginary number
I  acoustic intensity
i_p  interior point of interest
k  wave number
L  length of a cylinder
L_w  sound power level
m  row in a matrix
n  normal vector, column in a matrix
N  number of sub regions
N’  number of CHIEF points
p  position, point on surface
p(r)  elemental pressure
P(t)  time varying acoustic pressure
P  matrix relation of surface pressure
P^*  complex conjugate of acoustic pressure
P_{FF}  far field acoustic pressure
P(r,\theta)  acoustic pressure at the defined r and \theta coordinates
q  point on the closed boundary
r  distance between two points
Re  denotes real part of the quantity in brackets
S  defined closed boundary, surface
S_n  non-overlapping sub regions of S
t  time
U  uniform surface velocity
v(r)  elemental surface velocity
V  matrix relation of surface velocity
\( V^* \) complex conjugate of the velocity vector
\( W \) total sound power
\( x \) point of interest
\( x_i \) global coordinate in the x direction
\( x_m \) collocation point
\( y \) admittance, point of interest
\( y_i \) global coordinate in the y direction
\( z_i \) global coordinate in the z direction

\( \Psi \) velocity potential
\( \alpha \) constant that is equal to \( 2\pi \) (assumed to be case for this work)
\( \gamma \) angle between the normal vector and distance vector
\( \delta_{mn} \) Kronecker delta function (\( \delta_{mn} = 1, m = n \); \( \delta_{mn} = 0, m \neq n \))
\( \eta \) one of two directions in the mapped element
\( \theta \) angle between the field point and outward normal located at the middle of the length of a cylinder
\( \xi \) point defined on the surface, one of two directions in the mapped element
\( \rho \) instantaneous density
\( \varphi \) velocity potential of the point in the parenthesis
\( \omega \) angular frequency

\( \partial / \partial n \) outward normal detrivative
\( \text{det} |J| \) determinate of the Jacobian
INTRODUCTION

In recent decades, much attention has been given to programming the Boundary Element Method with application to predicting the sound pressure and sound power levels of arbitrary sources. Some of the more popular commercial programs that are currently available are SYSNOISE© and COMET© [1]. The goal of this work is to program such a software package for use in the Western Michigan University Noise and Vibration Laboratory.

All sound prediction software can be divided into two groups, which are interior and exterior problems. Because of time constraints, the focus of this thesis was on the exterior problem. The exterior problem was solved by using the Boundary Element Method; however, there are three different procedures that could be used. These procedures are known as the direct method, indirect method, and the approximation method.

The goal for each of the methods that are discussed is to obtain the velocity potential $\varphi(q)$. Knowing the velocity potential, the sound pressure and sound power can be calculated using the following equations:

\[ P(t) = -i\rho\omega\varphi(q) \]  
\[ W = \frac{1}{2} \int_S \Re\{P^*(t)v(q)\}dS_q \cdot \]

1
1.0 PROBLEM DEFINITION

In the following sections the problem definition for the exterior acoustic problem will be discussed. The boundary conditions for acoustic problems will be defined and discussed, and the integral formulation will be derived.

1.1 The Exterior Acoustic Problem

The purpose of the exterior acoustic problem is to determine the pressure distribution of an object that is vibrating, with some known surface velocity. An example of the exterior acoustic problem is when the sound power of an automobile engine is calculated given measurements of its surface vibrations and detailed knowledge of its geometry. Let’s define the domain of the acoustic field of interest to be the region \( E \) exterior to the closed boundary \( S \), as illustrated in Figure 1.

![Figure 1: Exterior Acoustic Problem Definition](image)

The pressure distribution is assumed to be present in the defined region \( E \). The pressure distribution is governed by the linear wave equation that is in the form of:

\[
\nabla^2 \Psi = \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \Psi. 
\]  (3)
Since periodic solutions to the wave equation are considered, the velocity potential $\Psi$ can be reduced to the following:

$$\Psi(x_n, t) = \varphi(x_n) \cdot e^{-i\omega t} \quad n = 1, 2, 3. \quad (4)$$

The substitution of Equation (4) into (3) reduces to the Helmholtz equation:

$$\nabla^2 \varphi(x) + k^2 \varphi(x) = 0 \quad (5)$$

where $k^2 = \frac{\omega^2}{c^2}$ and $x$ is a coordinate of a point of interest. The solution to Equation (5) will yield the pressure distribution at any point for a given boundary condition [2].

1.2 Boundary Conditions

It is important to apply the proper boundary conditions to the defined surface $S$. There are three types of boundary conditions that are most commonly used in practical acoustics. Those boundary conditions are Dirichlet, Neumann, and Robin or Mixed [3], which are defined below:

1.) Dirichlet Condition: $\varphi(p) = f(p)$
2.) Neumann Condition: $\frac{\partial \varphi(p)}{\partial n} = f(p)$
3.) Robin or Mixed Condition $\frac{\partial \varphi(p)}{\partial n} + iky \varphi(p) = f(p)$

The Dirichlet and Neumann boundary conditions can be presented in one equation, which is shown below.

$$a(p) \cdot \varphi(p) + b(p) \cdot \frac{\partial \varphi(p)}{\partial n} = f(p)$$

The interior problem typically uses the Dirichlet and Robin boundary conditions. The Dirichlet boundary condition is when the boundary is rigid ($a(p) = 1$...
and \( b(p) = 0 \) which relates to pressure. The Robin boundary condition is when admittance conditions have been specified.

The exterior problem typically uses Dirichlet and Neumann boundary conditions. The Dirichlet condition is when \( (a(p) = 1 \text{ and } b(p) = 0) \), and the Neumann condition is when \( (a(p) = 0 \text{ and } b(p) = 1) \) which relates to velocity.

1.3 Integral Formulation

Several different types of Boundary Integral Equations have been derived. The one presented here is considered the classical approach \([4,5]\). The classical Helmholtz integral representation formula for an internal or external problem is \([4,5]\):

\[
\int \int \frac{\partial \varphi(y)}{\partial n_y} dS_y - \int_\Sigma \frac{\partial G_k(x, y)}{\partial n_y} \varphi(y) dS_y = \frac{\partial}{\partial n} \int_\Sigma G_k(x, y) \varphi(y) dS_y .
\]  

(6)

\( G_k(x,y) \) is the fundamental solution of the Helmholtz Equation (5) with a Dirac Delta function on the right hand side in a three-dimensional Cartesian Coordinate System as shown below as Equation (7):

\[
G_k(x,y) = \frac{e^{-ikr}}{4\pi r}
\]

(7)

and its derivative with respect to the normal is given by:

\[
\frac{\partial G_k(x, y)}{\partial n_y} = \frac{e^{-ikr}}{4\pi r} \left( ik + \frac{1}{r} \right) \frac{\partial r}{\partial n_y} .
\]

(8)

In the above expressions, \( r = r(x,y) \) represents the distance from a point \( x \) on the surface to field point \( y \), as seen below in Figure 2 and its derivative are taken with reference to the coordinates of \( y \) which yields the following result.
\[ \frac{\partial v}{\partial n_y} = \cos(\gamma). \]  

Equation (6) may be used for external and internal problems, by assuming that the outward normal of wave propagation in the direction of region E is always positive.

Figure 2: Geometry Description for the Acoustic Radiation Problem

If a point \( x \) is allowed to approach a surface point \( \phi \), referring to Figure 2, and taking into account the continuity and discontinuity property of the single and double layer potentials, across the density carrying surface, Equation (6) becomes:

\[
C(\xi) \cdot \phi(\xi) = \int_S G_k(\xi, y) \frac{\partial \phi(y)}{\partial n_y} dS_y - \int_S \frac{\partial G_k(\xi, y)}{\partial n_y} \phi(y) dS_y ,
\]

for every \( \xi \) on \( S \). The coefficient \( C(\xi) \) is given by:

\[
C(\xi) = 4\pi + \int_S \frac{\partial G_0(\xi, y)}{\partial n_y} dS_y
\]

(11)
where $G_0(\xi, y) = 1 / (4\pi r)$ is the fundamental solution of Laplace’s equation. Equation (11) also includes the possibility that the surface $S$ may have non-smooth geometry such as edges or corners. Equation (10) can then be solved for the velocity potential.
2.0 REVIEW OF LITERATURE

There are two major methods when solving the exterior acoustic problem, which are the direct and indirect method. However, upon performing the literature survey for this research an additional method was found called the approximation method. All three methods solve the Helmholtz Equation (5) in different ways.

The direct method solves the Helmholtz equation directly. The indirect method solves the Helmholtz problem by applying a source density function, which will be solved, and then will be used to predict the pressure distribution. There are several different approaches to the indirect method, of which two will be explained. The approximation method used the Fast Fourier Transform and rotational symmetry to predict the pressure distribution.

2.1 Direct Method

There have been two major advances for the direct method, which are the Combined Helmholtz Integral Equation Formulation, and the Burton Miller method.

2.1.1 Combined Helmholtz Integral Equation Formulation (CHIEF)

In 1967, Harry Schenck [6] developed a method that uses the normal surface velocity on a boundary to determine the sound pressure field. The method that he developed fixed one of the many shortcomings of boundary integral equations, which was to overcome the well-known nonuniqueness problem. The nonuniqueness problem is when the uniqueness of the solution is uncertain and is most apparent
around the characteristic frequencies. The method he developed is called the Combined Helmholtz Integral Equation Formulation (CHIEF).

What makes Schenck’s direct integral method different from other previous methods is that he uses the Helmholtz equation with \( y \) in \( E \) as a constraint that must satisfy the Helmholtz equation for all \( \varphi \) on \( S \). An \( N \times N \) system of equations resulting from the Equation (13) for \( \varphi \) on \( S \) is argumented by \( K \) equations representing \( \varphi \) in \( S \) for which the left-hand side of Equation (13) is zero. The resulting \( (N + K) \times N \) overdetermined system of equations may be solved by the least squared procedure [4].

One of the serious shortcomings is that the CHIEF method requires the appropriate selection of “CHIEF points”. A “CHIEF point” is when a fictitious point is placed on the inside of \( S \). This “CHIEF point” is what overcomes the very apparent non-uniqueness problem in the Boundary Element Formulation. Schenk never mentioned how many “CHIEF points” are required for a given solution accuracy, nor did he mention how to pick the “CHIEF points” for a given boundary. However, Schenk did note that when a “CHIEF point” is placed near or on the surface the “CHIEF point” does not produce a good result.

Juhl in 1993 [7] developed methodologies that would overcome the one shortcoming of the CHIEF method. Juhl documented that picking more “CHIEF points” and test to see which are “good” points and disregarding the “bad” points would fix Scheck’s problem. Juhl also stated that adding “CHIEF points” to an
already existing model to obtain a higher accuracy is very easy. Juhl methodology for adding “CHIEF points is shown below.

1.) Calculate the matrices for the surface that is defined as A and B.

2.) Calculate the matrices for the original set of “CHIEF points” that is defined as $A_{int}$ and $B_{int}$.

3.) Combine matrices from step 1 and 2.

4.) Predict the sound pressure level with original number of points.

5.) Calculate new $A_{int}$ and $B_{int}$ matrices for a new set of “CHIEF points”.

6.) Combine matrices from step 1 and 4.

7.) Predict sound pressure level with new set of points.

8.) Repeat accordingly.

Juhl tested his method on the example of the scattering effect of a sphere. He showed that the difference between using one “CHIEF point” and two is shown below in Figure 3. In Figure 3 the solid line is the theoretical line for the scattering effect, the dashed line is the solution with one CHIEF point and the dotted line, which is superimposed over the solid, has two CHIEF points.
Seybert, Soenarko, Rizzo, and Shippy [8,9] did extensive testing on the CHIEF method by Schenk. They mentioned that discretization of the boundary determines the accuracy of the prediction in pressure levels.

2.1.2 Burton-Miller Method

Burton and Miller in 1971 developed a method that combined two different integral equations to overcome a nonuniqueness problem that exists in the boundary element formulation. The formulation is called the Gradient Helmholtz Integral Equation (GHIE). The first of the two equations (12) is typically called the elementary formulation or Helmholtz equation. The second equation (13) arises when Equation (12) is differentiated with respect to the normal on the surface.

\[
\int_S \left[ G_k(\xi, y) \frac{\partial \phi(y)}{\partial n_y} - \phi(y) \frac{\partial G_k(\xi, y)}{\partial n_y} \right] dS = 0 \tag{12}
\]

\[
\int_S \left[ \frac{\partial G_k(\xi, y)}{\partial n_\xi} \frac{\partial \phi(y)}{\partial n_y} - \phi(y) \frac{\partial^2 G_k(\xi, y)}{\partial n_\xi \partial n_y} \right] dS = 0 \tag{13}
\]
The primary difficulty with the Burton-Miller [10] method is the highly singular kernel in the GHE. The kernel requires very careful numerical handling to produce accurate results. Burton-Miller [10], and Meyer et al. [11] reformed regular transformations that reduced the order of the GHE singularity. Teri [12] preformed a transformation on the highly singular kernel by replacing it with a contour integral.

Cunefare and Koopmann [13] developed a modified form of the Burton-Miller method called Coupled Helmholtz Integral (CHI), which fixes the hypersingularity and the uniqueness problems that exists in the boundary element formulation. Whereas all other methods confine all or most of the field points to the surface of the body, resulting in singular integral kernels, all of CHI’s field points are constrained to be within the interior of the body. Since the field points and the source points never coincide, there are no singularities.

2.2 Indirect Method

The elementary formulation begins by defining Helmholtz integral operators, which are denoted by \(L_k, M_k, M'_k\) and \(N_k\) and defined as:

\[
(L_k \mu) (p) = \int_S G_k (p, q) \mu(p) dS_q \quad (p \in D_+ \cup S \cup D_-),
\]

\[
(M_k \mu) (p) = \int_S \frac{\partial G_k}{\partial n_q} (p, q) \mu(p) dS_q \quad (p \in D_- \cup S \cup D_+),
\]

\[
(M'_k \mu) (p) = \int_S \frac{\partial}{\partial n_p} \left( \int_S \frac{\partial G_k}{\partial n_q} (p, q) \mu(p) dS_q \right) \quad (p \in S),
\]

\[
(N_k \mu) (p) = \int_S \frac{\partial}{\partial n_q} \frac{\partial G_k}{\partial n_q} (p, q) \mu(p) dS_q \quad (p \in S).
\]
The density function $\mu(p)$ is defined for $p$ on $S$. $G_k(p,q)$ is the fundamental solution for the Helmholtz equation as given by Equation (5).

### 2.2.1 Elementary Method

The elementary method [3] derivation starts by combining and manipulating the Helmholtz operators. By combining and manipulating the Helmholtz operators together will yield the following equation.

$$
(L_k N_k) \mu(p) = \left( -\frac{1}{4} I + M_k^2 \right) \mu(p) \quad p \in \Gamma
$$

(14)

Applying the Dirichlet boundary condition to the Helmholtz integral operators will yield the following equation.

$$
\phi(p) = (L_k \mu)(p) \quad p \in D
$$

(15)

To make sure that the Neumann boundary conditions are also satisfied, Equation (15) is differentiated which will yield the following:

$$
\left( -\frac{1}{2} I + M_k^t \right) \mu(p) = f(p) \quad p \in \Gamma
$$

(16)

Once $\mu$ is found from Equation (16), it can be substituted into (15) to obtain the $\phi(p)$, which is needed to compute sound pressure.

### 2.2.2 Kussmaul’s Indirect Boundary Element Formulation

The equations that make up the formulation of Kussmaul’s method [14] are:

$$
\phi(p) = \alpha \{L_k \mu\}_1(p) + \beta \{M_k \mu\}_1(p), \quad p \in \Gamma
$$

(17)

$$
\phi(p) = \alpha \{L_k \mu\}_1(p) + \beta \{M_k \mu\}_1(p) + \frac{1}{2} \beta \mu(p), \quad p \in \Gamma
$$

(18)
\[ \alpha \{ M_k^i \mu^i \}(p) + \beta \{ N_k \mu^i \}(p) - \frac{1}{2} \alpha \mu(p) = \frac{\partial \varphi}{\partial n_p}(p), \quad p \in \Gamma. \quad (19) \]

The conditions on \( \alpha \) and \( \beta \) ensure that Equation (19) is unique for all wave numbers.

By allowing \( p \) to take on the form of \( p_1, p_2, p_3, \ldots p_n \) collocation reduces Equation (19) to:

\[ \left[ \alpha \left( M_k^i - \frac{1}{2} I \right) + \beta N_k \right] \mu \approx \nu \quad (20) \]

Approximations to the velocity potential can now be calculated by using the following equation for exterior points:

\[ \hat{\phi}(p) = \sum_{j=1}^{N} \left\{ \alpha \int_{\Delta S} G(p, q) dS_q + \beta \int_{\Delta S} \frac{\partial G}{\partial n_p}(p, q) dS_q \right\} \hat{\mu}_j \quad (21) \]

and for point on the surface the following equation should be used:

\[ \hat{\phi}(p) = \sum_{j=1}^{N} \left\{ \alpha \int_{\Delta S} G(p, q) dS_q + \beta \int_{\Delta S} \frac{\partial G}{\partial n_p}(p, q) dS_q - \frac{1}{2} \delta_{ij} \right\} \hat{\mu}_j \quad (22) \]

The unknown source density function can be calculated using Equation (20), and then the velocity potential can be calculated using Equation (21).

2.3 Approximation Analysis

Kuijpers, Verbeek, and Verheij [15] proposed a method that uses the Fast Fourier Transform technique to approximate the solution to the Helmholtz equation for an axis-symmetric problem. It can be shown that the Helmholtz equation for a Fourier series is rewritten (in (23) and (24)) as corresponding to Figure 4.
where $C(x)$ is a coefficient depending on the position of $x$,

$$C(x) = \begin{cases} 
0, & \text{for } x \text{ outside the acoustic medium}, \\
1, & \text{for } x \text{ inside the acoustic medium}, \\
\frac{1}{2}, & \text{for } x \text{ on the smooth surface } S \text{ of the acoustic medium},
\end{cases}$$

$p_s^n, p^{S^n}, p_c^n, \text{ and } p^{c^n}$ are Fourier coefficients and their derivatives, where:

$$H_n = \int_0^{2\pi} \frac{e^{-ikR(x,y)}}{4\pi R(x,y)} \cos(n\theta) d\theta$$

$$H'_n = \int_0^{2\pi} \frac{\partial}{\partial \nu} \left( \frac{e^{-ikR(x,y)}}{4\pi R(x,y)} \right) \cos(n\theta) d\theta$$

Figure 4: Approximation Method Problem Description
The solution of the Helmholtz equation for axis-symmetric structures can be obtained numerically by solving Equations (25) and (26) using standard boundary element procedures. The generator $L$ of the axis-symmetric body can be discretized and the geometry and acoustic variables $p$ and $p'$ are assumed to vary according to isoparametric shape functions on the surface of the body (Kuijpers, Verbeek, Verheij, 1997). It can be shown that Equations (25) and (26) are not only singular but the cosine function causes the total integrand to oscillate. Kuijpers, Verbeek and Verheij showed that slightly modifying Equations (25) and (26) to (27) and (28), respectively can be used to calculate $H_n$ and $H'_n$:

$$h(\theta) = \frac{e^{-ikR(\theta)}}{4R(\theta)} \quad (27)$$

$$h'(\theta) = \frac{\partial}{\partial \nu} \left( \frac{e^{-ikR(\theta)}}{4R(\theta)} \right) \quad (28)$$

Kuijpers, Verbeek and Verheij developed an algorithm that was used to evaluate the integrals in Equations (25) and (26), which is stated below

1.) Determination of the number of samples needed for computation of integrals (25) and (26) with a desired accuracy.

2.) Evaluation of the functions $h$ and $h'$.

3.) Fast Fourier transformation of the computed function values.

4.) Selection of the nth terms of the calculated Fourier spectrums which are numerical values for the integrals (25) and (26).
They concluded that compared to the Gaussian quadrature based method, the FFT method was computational faster and the error did not exceed $10^{-3}$. 
3.0 SCOPE OF WORK

The exterior acoustic problem will be programmed using the CHIEF method. The CHIEF method has several benefits over the other presented methods. Some of the benefits include that it is easier to program, easier to adapt for future modifications, and will provide an accurate solution for all wavenumbers with the appropriate selection of “CHIEF points.” The first step in a program that predicts sound pressure and sound power levels is the discretization of the arbitrarily shaped source, which will be done using the program SDRC-IDEAS©.

The program will be verified by comparing the sound pressure and sound power of a number of different theoretical and experimental models. Some of these models consist of a radiating sphere, a vibrating box and a uniformly vibrating cylinder. The radiating sphere and the uniformly vibrating cylinder will be verified by using theoretical techniques, as well a vibrating box will be verified using experimental techniques. Finally, the program will be written in MATLAB©, with the emphasis of using a personal computer for computation.
4.0 NUMERICAL IMPLANTATION

4.1 Schenk’s Integration Formulation

As stated before in the literature review, Schenk uses three different versions of the Helmholtz Integral Equation (6) for his formulation. These modified formulations are documented below as Equations 29, 30 and 31.

\[ \varphi(fp) = \int_{S} G_{K}(fp, y) \frac{\partial \varphi(y)}{\partial n_{y}} dS_{y} - \int_{S} \hat{G}_{K}(fp, y) \varphi(y) dS_{y} \quad fp \text{ exterior to } S \quad (29) \]

\[ \frac{\alpha}{4\pi} \varphi(x) = \int_{S} G_{K}(x, y) \frac{\partial \varphi(y)}{\partial n_{y}} dS_{y} - \int_{S} \hat{G}_{K}(x, y) \varphi(y) dS_{y} \quad x \text{ on } S \quad (30) \]

\[ \varphi(ip) = \int_{S} G_{K}(ip, y) \frac{\partial \varphi(y)}{\partial n_{y}} dS_{y} - \int_{S} \hat{G}_{K}(ip, y) \varphi(y) dS_{y} \quad ip \text{ interior to } S \quad (31) \]

In the equations above, point \( y \) is always defined on the surface, while \( x \) can be on the interior (\( ip \)), surface or exterior (\( fp \)) of the defined surface \( S \), as seen below in Figure 5.

Figure 5: Helmholtz Integral Pectoral Representation
After the discretization of the arbitrary shaped object, the three Helmholtz equations can be then put into matrix form. It is here where Schenk combines the Surface Helmholtz Equation (SHF) and the Interior Helmholtz Equation (IHF) to overcome the uniqueness problem that is apparent in the Boundary Element Method. Assuming a given (calculated or measured) surface velocity, the surface pressure can be calculated by solving the overdetermined system of equations by utilizing the least squares procedure. Once the surface pressure and velocity is known, the far field pressure can be calculated for individual field points using equation (29).

4.2 Surface Matrices

By discretizing the geometry of any arbitrary shaped object into N elements and assuming that the surface velocity and pressure are constant functions over each element, one can write these elemental pressure and velocity values as

\[ p(r) = p_n \quad \text{and} \quad v(r) = v_n \quad \text{for} \quad r \in S_n, \quad n = 1,2,\ldots,N. \quad (32) \]

Collocation of the integrand in the Helmholtz integrals (30) and (31) leads to the matrix relations

\[ A_{mn} = \frac{\alpha}{4\pi} \delta_{mn} - \int_{S_n} \frac{\partial G(x_m, y)}{\partial n_y} dS_y \quad m = 1,2,\ldots,N + N' \quad n = 1,2,\ldots,N \quad (33) \]

\[ B_{mn} = i \omega \rho \int_{S_n} G(x_m, y) dS_y \quad m = 1,2,\ldots,N + N' \quad n = 1,2,\ldots,N. \quad (34) \]

Because we assume there can be no pressure on the interior of the vibrating body, the following combined matrix relation holds true.

\[ AP = BV \quad (35) \]
A matrix representation of Equation (35) is shown below as Equation (36). From the figure below it can be seen that the elements above the line represent the contribution from the surface calculations, whereas the elements below the line are the contribution from the CHIEF points.

\[
\begin{bmatrix}
A_{11} & A_{12} & \cdots & A_{1N} \\
A_{21} & A_{22} & \cdots & A_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
A_{N1} & A_{N2} & \cdots & A_{NN}
\end{bmatrix}
\begin{bmatrix}
P_1 \\
P_2 \\
\vdots \\
P_N
\end{bmatrix}
= 
\begin{bmatrix}
B_{11} & B_{12} & \cdots & B_{1N} \\
B_{21} & B_{22} & \cdots & B_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
B_{N1} & B_{N2} & \cdots & B_{NN}
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2 \\
\vdots \\
V_N
\end{bmatrix}
\] (36)

4.2.1 Non-self Integration

In this work, a quadrilateral element will be mapped from a three-dimensional global coordinate system to two-dimensional element local coordinate system as seen below as Figure 6.
In the local coordinate system for the element the limits for the element is –1 to +1 for both the $\phi$ and $\eta$ directions. Every point inside the element can be mapped using the following equations

$$x = \sum_{i=1}^{4} x_i N_i \quad ; \quad y = \sum_{i=1}^{4} y_i N_i \quad ; \quad z = \sum_{i=1}^{4} z_i N_i$$  \hspace{1cm} (37)$$

where

$$N_1 = \frac{1}{4} (1 - \xi)(1 - \eta)$$
$$N_2 = \frac{1}{4} (1 + \xi)(1 - \eta)$$  \hspace{1cm} (38)$$
$$N_3 = \frac{1}{4} (1 + \xi)(1 + \eta)$$
$$N_4 = \frac{1}{4} (1 - \xi)(1 + \eta).$$

Using the mapped quadrilateral element, one can rewrite the Equations (33) and (34) in terms of $\phi$ and $\eta$, with the integration limits form –1 to +1 for both $\xi$ and $\eta$ directions.

$$A_{mn} = \frac{\alpha}{4\pi} \delta_{mn} - \iint_{-1-1}^{+1+1} \partial G(x_m, y) \partial n y \det J|d\xi d\eta$$  \hspace{1cm} (39)$$

$$B_{mn} = i \omega \iint_{-1-1}^{+1+1} G(x_m, y) \det J|d\xi d\eta$$  \hspace{1cm} (40)$$

where $x_m$ is the defined collocation point which can be located in the center of an element or it can be the location of the CHIEF point. The integrand of these two integrals will be calculated using Gaussian Quadature.

21
The Gaussian Quadature technique is calculated by evaluating various points inside the element, which is then multiplied by a weighting factor. For example, for a simple line integration let’s assume a function \( f(x) \) with the limits of integration for \( x_1 \) to \( x_2 \). The function \( f(x) \) can be transformed into a function \( \phi(\varphi) \) which incorporates the Determinant of the Jacobin.

\[
I = \int_{x_1}^{x_2} f(x) \, dx = \int_{-1}^{+1} \phi(\xi) \, d\xi
\]

(41)

If evaluated using the discussed technique this integral becomes;

\[
I = \int_{-1}^{+1} \phi(\xi) \, d\xi = W_1\phi_1 + W_2\phi_2 + \cdots + W_n\phi_n.
\]

(42)

A pictorial representation of this calculation is shown below as Figure 7. Figure 7 shows the Gauss Quadature technique for one, two and three gauss points, which in the figure below are labeled as 1, 2, and 3.
Another aspect of the numerical implementation is the calculation of the Jacobin. According to Brebbia [16], the transformation of the global coordinate system to the local element coordinate system is shown to be:

\[ dS(x) = |e_1 \times e_2| (d\zeta_1, d\zeta_2). \]  

(43)

A quadrilateral element in a three dimensional global coordinate system is shown to be

\[
[J] = \begin{bmatrix}
  x_\xi & y_\xi & z_\xi \\
  x_\eta & y_\eta & z_\eta
\end{bmatrix}.
\]  

(44)

By setting the first and second rows of the Jacobin matrix to individual column vectors of \( \vec{r}_\zeta \) and \( \vec{r}_\eta \) respectively, the cross product can be calculated.

\[
\vec{r}_\zeta \times \vec{r}_\eta = \begin{bmatrix}
  x_\xi \\
  y_\xi \\
  z_\xi
\end{bmatrix} \begin{bmatrix}
  x_\eta \\
  y_\eta \\
  z_\eta
\end{bmatrix}
\]  

(45)

The result of the cross product is shown below;

\[
\vec{r}_\zeta \times \vec{r}_\eta = \begin{bmatrix}
  y_\eta & z_\eta - z_\xi \\
  z_\xi & x_\eta - x_\xi \\
  x_\xi & y_\eta - y_\xi
\end{bmatrix}.
\]  

(46)

As shown by Brebbia the magnitude of the cross product can be calculated using the following equations;

\[
dS(x) = \sqrt{(d_1)^2 + (d_2)^2 + (d_3)^2 \cdot d\zeta_1 d\zeta_2}
\]  

(47)

where

\[
d_1 = (y_\zeta z_\eta - z_\zeta y_\eta)
\]
The final aspect of discussion is the contribution of the normal vector. The normal vector is not directly related to the calculation of the $A_{nm}$ matrix however, it is used to calculate $\cos(\gamma)$, which is the angle between the normal vector and the field point vector. In this simulation, the dot product will be used to calculate $\cos(\gamma)$. In general form, the dot product is stated below as;

$$\vec{P} \cdot \vec{Q} = \|P\|\|Q\|\cos(\gamma).$$  \hspace{1cm} (49)$$

Setting the vector $P$ equal to the normal, $Q$ equal to the distance vector, the $\cos(\gamma)$ can be calculated. Solving Equation (49) and substituting the appropriate variables, the $\cos(\gamma)$ yields

$$\frac{\vec{n} \cdot \vec{R}}{\|n\|\|R\|} = \cos(\gamma).$$  \hspace{1cm} (50)$$

4.2.2 Self-Integration

In the diagonal elements of the $A_{nn}$ and $B_{nn}$ matrices for $m = n$, a singularity in the integrand arises. To overcome this singularity, the quadrilateral is divided up into four sub regions, with the vertex at $\varphi = 0$ and $\eta = 0$, which is shown below. The integration over each region is calculated, and then the results of the regions are summed up.
Letting the collocation, $x_m$, point be the vertex of the sub regions, the integration over each region is calculated as discussed in Section 4.2.1 Non-Self Integration, however the limits of integration are modified accordingly for numerical integration by Gauss Quadature. To ensure numerical accuracy over each sub region, a numerical integration tolerance for convergence is increased.

4.3 Farfield Matrices

Once the surface pressures and velocities have been found, the far field pressures can be calculated. The numerical implementation that was used for the CHIEF point can be used for the far field matrices. Shown below are the matrices that are used to calculate the far field pressure, where $x_m$ is the field point of interest.

$$A_m^e = -\int_{-1}^{1} \int_{-1}^{1} \frac{\partial G(x_m, y)}{\partial n_y} \det |\mathbf{J}| d\xi d\eta$$

$$B_m^e = i \omega p \int_{-1}^{1} \int_{-1}^{1} G(x_m, y) \det |\mathbf{J}| d\xi d\eta$$

(51)

(52)
When these matrices are calculated, they are multiplied by the respective pressure and velocity vectors, which is shown by Equation (53).

\[ P^{FF} = A_{nn}^F P + B_{nn}^F V \]  \hspace{1cm} (53)
5.0 PROGRAM VALIDATION

Once the CHIEF method was numerically implemented in a computer simulation, proper verification was required. It was decided that verification using both theoretical and experimental models was needed. The theoretical validation of a sphere, and cylinder was documented; however the idea of validating the computer simulation using an experimental model was not documented. The experimental verification was done using the various resources in Western Michigan University’s Noise and Vibration Laboratory. In the different stages of the validation process, different goals were met to ensure that the computer simulation was properly verified.

5.1 Uniformly Vibrating Sphere

The goal of this model was to model the radiation of sound from a uniformly vibrating sphere. Thus we aimed to verify that the acoustic pressure on the surface and at a point in the far field matches the theoretical solution within reason. The coordinate system, field point and surface point for the sphere are shown below in Figure 9. The sphere was modeled with a theoretical uniform surface velocity of 1.0 m/s, with a radius of 1.0 meter.
The theoretical solution for a uniformly vibrating sphere is shown below [6]:

\[
P(r = f_p) = \frac{i \omega p U a^2 e^{ka} e^{-ikf_p}}{(1 + ika) f_p}.
\]  

(54)

The surface pressure of the uniformly vibrating sphere can be calculated by letting the field point equal the radius. The new equation for calculating the surface pressure is shown below as Equation (54).

\[
P(r = a) = \frac{i \omega p U a}{(1 + ika)}
\]  

(55)

The sphere was drawn and meshed in SDRC-Ideas using 1\textsuperscript{st} order quadrilateral elements. The result of this mesh can be seen below as Figure 10.
One can see that the mesh, from Figure 10 above, was not very uniform and the results were poor; therefore a new mesh, as shown below as Figure 11, was created to provide a more uniform element. However to ensure that the element size was more uniform, the area of each element was calculated as shown below as Figure 12. Once the area of each element was calculated a histogram and bar graph was created. This new mesh and the resulting histogram can be seen below.
Figure 11: New Discritized Mesh

Figure 12: Bar Graph and Histograph of Figure 11
According to the histograph and the bar graph of the area of each element, there are only a few different element sizes. Therefore one can conclude that this mesh was going to give a more realistic result.

Using this new mesh the surface pressure was calculated. The corresponding surface pressure from the analytical equation was also calculated. At this particular frequency, the result of the surface pressure can be seen below in Figure 13 and the analytical solution for the same frequency was calculated to be 395 Pascals. The mesh gives good results however, the points where the panels of the sphere join together the surface pressure gets skewed. This phenomenon can be better represented with a finer (smaller elements) mesh which is also shown below as Figure 14.

![Figure 13: Surface Pressure of Uniformly Vibrating Sphere](image)

Figure 13: Surface Pressure of Uniformly Vibrating Sphere
Figure 14: Surface Pressure of a Uniformly Vibrating Sphere Meshed Finer

To verify that the size and shape of the element skewed the data, a cube was created. The cube was modeled with the sides equal to 1.0 meter. The defined element length for the modeled cube was 0.5 meters. The cube was assumed to give similar surface pressures at very low frequencies, given that the surface velocities were uniform over all of the elements. The surface pressure of the cube was calculated, which is shown below as Figure 15.
Figure 15: Surface Pressure of Uniformly Vibrating Cube

Figure 15 shows that if all of the elements are all the same size and are not skewed the surface pressures for all of the elements will be the same. This result shows that the skewing and non-uniform elements in the mesh of a sphere will yield incorrect surface pressures. The analytical surface pressure was calculated to be 109.5 Pascals. This was similar to the calculated surface pressure that is shown above. Indeed at low frequencies a cube has similar surface pressure as a sphere of the same radius.

If CHIEFmat was programmed to allow triangular elements, the sphere could be modeled with triangular elements perfectly. The result of the triangular element mesh is believed to yield the better results. In fact, it was in Cunefare and Koopmann in 1989 [13] that verified their simulation using only triangular elements. Wilton in
1978 [17] used a combination of second order triangular and quadrilateral elements to model a sphere.

Once the surface pressures of the sphere were calculated and verified using the analytical equations, the far field pressures were calculated. For any type of vibrating object, it can be said that the far field is 10 times the lowest wavelength of interest. In the case of the sphere, a far field point was located 100 meters away for the center of the sphere. Shown below is a graph of far field pressures for the coarse mesh (shown in Figure 16).

Figure 16: Far Field Pressure of a Uniformly Vibrating Sphere
The result shown above was compared to a simulation with no CHIEF points.

The addition of CHIEF points in a simulation enables the solution to become unique at critical $ka$ values. The $ka$ value for the sphere is the wavenumber multiplied by the radius. Typical $ka$ values for a sphere consist of $\pi$, $2\pi$, $3\pi$, ..., $N\pi$ as stated by Schenk in 1968 [6]. For the defined model the critical $ka$ value is at $\pi$, or with a sphere having a radius of 1 m, the corresponding critical frequency is 171.5 Hz. The graph shown below compares the analytical solution to the simulation without any CHIEF points.

Figure 17: Far Field Pressure without any CHIEF points
5.2 Uniformly Vibrating Cylinder

Once the sphere surface and far field pressure was verified, the next model that was to be verified was a cylinder. Since analytical equations for a finite cylinder are very difficult to obtain and are not widely known, it was decided that the cylinder could be verified if it was modeled as a line source. A line source was used because of the known theoretical equation [18]. This equation and problem definition is shown below.

\[ P(r, \theta) = P_{ax}(r)H(\theta) \quad (56) \]

\[ H(\theta) = \left| \frac{\sin(v)}{\nu} \right|, \quad \nu = \frac{1}{2} kL \sin(\theta) \quad (57) \]
\[ P_{aw}(r) = \frac{1}{2} \rho c U \left( \frac{a}{r} \right)^{kL} \] (58)

It was assumed that the line source validation yielded correct values at low kL values or kL \( \leq 10 \). For a cylinder there are two uniqueness values that need to be considered, which were the kL and ka. The kL value describes the bending and torsion motions, where as the ka values describe the pulsating or breathing motion in of the cylinder in the radial direction. Typical kL values consist of 1, 2, 3, …, N. Typical ka values consist of \( \pi, 2\pi, 3\pi, \ldots, N\pi \) as stated by Schenk in 1968 [6]. Knowing that only low kL values will be calculated, 10 CHIEF points were picked along the axis of the long cylinder. Picking 10 CHIEF points will validate the cylinder up to kL of 10.

For the definition of a line source to be valid, the length must be much larger than the radius. For the line source model, the cylinder radius was assumed to be 1.0 meter, and the length was modeled at 100.0 meters. An example of this is shown below as Figure 19. Since it was shown that element consistently is important when predicting acoustic pressures, a model of equal size elements along the length was constructed. The end caps were meshed keeping a similar area.
A uniform constant velocity of 1.0 m/s was applied to the model, and the surface pressure was calculated. Since Equation (56) can not predict the surface pressure, the far field pressure was calculated and compared to that of the line source. However a visual check of the surface pressures was done. As previously shown, the surface pressure for elements of the same size should be constant. It can be seen in Figure 21, the surface pressure, units of Pascals, along the length of the cylinder remains fairly constant; however the end caps do skew the data. This skewing of the data is the result of solving a large overdetermined system of equations.
Once the surface pressure was visually checked over, the far field pressure was calculated using CHIEFmat, which is shown below as Figure 21.
The graph shown above can be compared to a Uniformly Vibrating Cylinder without any CHIEF points shown below.
From Figure 21 it was shown that small kL values yielded correct results; however when comparing Figure 23 to Figure 22, an improvement for kL values between 1 and 10 were dramatically improved. This means that a long cylinder modeled with first order quadrilateral elements, at low kL values kL ≤ 5, CHIEFmat predicts accurate sound pressure levels.

5.3 Vibrating Box

Once the theoretical validation was CHIEFmat was accomplished a “real-world” validation was needed. This “real-world” validation consists of measuring the
sound power and surface vibrations of a vibrating box and comparing it to the predicted sound power, using the measured velocity information. A picture of this vibrating box inside Western Michigan University’s Reverberation Chamber can be shown below as Figure 23. The box has a length of 0.61 meters, a width of 0.35 meters, and a depth of 0.22 meters. The four sides and the bottom of the box are made of 3/4” thick steel, where the top of the box is made of 1/8” thick aluminum. A shaker located inside of the box, that is attached to the top produces the random vibration that was measured.

![Figure 23: Vibrating Box](image)

5.3.1 Surface Vibrations

As stated above, measured surface velocities were needed to accurately predict the sound pressure and power levels. It was assumed that the surface velocities of the four sides and the bottom of the box were zero. Frequency Response Functions (FRFs) measurements of plate vibrations with respect to the signal were
conducted (FRF = plate acceleration / excitation signal). Collecting the data as FRFs allows the engineer to gather phase or directional information. For example for the 1st mode, which is shown below as Figure 24, the direction and phase information is not necessary, because the whole surface of the plate is vibrating in phase. However for the 2nd and 3rd modes, which are shown below as Figures 26 and 27 respectively, the phase is very important. Therefore the FRF will be taken not with respect to the force, but with respect to the almost constant source.

Figure 24: 1st Mode of Vibrating Box

Figure 25: 2nd Mode of Vibrating Box
The surface velocities were not directly measured, instead the surface accelerations were taken, and then the acceleration data was divided by $-\omega$. This division allowed the acceleration data to be turned into velocity data. After the surface velocities were calculated, they were imported into a separate program, which displayed the surface velocities of the plate and provided a visual check for the model.

5.3.2 Sound Power

The sound power of the vibrating box was measured in Western’s Michigan University reverberation chamber according to the ANSI standard S12.31-1990 [19]. The standard states that the sound power of the source under consideration can be calculated by measuring the averaged sound pressure level of six microphones, and adding the difference between the sound power of a calibrated reference sound source and the measured sound pressure level of the same reference sound source. This sound power representation is shown below as Equation 59.
\[ L_{w(source)} = L_{p(source)} + (L_{w(ref)} - L_{p(ref)}) \]  

The reference sound source that was used in the measurement of sound power was manufactured by G&G Acoustics Inc., which has a model number as GA60. A picture of this reference source is shown below as Figure 27.

![Reference Sound Source](image)

Figure 27: Reference Sound Source

This reference sound source was calibrated and measured by the manufacturer. The manufacturer calibrated the sound source in 1/3 octave bands which is shown below as Table 1.
Once the sound pressure of all six microphones was taken, an average of these values were obtained, shown below as Figure 28. This pressure spectrum was imported into MATLAB©, where the sound power was calculated for the vibrating box (shown in Figure 29).

Table 1: Calibrated Sound Pressure Levels

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Lp(ref)</th>
<th>Frequency</th>
<th>Lp(ref)</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>70.1</td>
<td>1000</td>
<td>80.5</td>
</tr>
<tr>
<td>80</td>
<td>75.4</td>
<td>1250</td>
<td>80.2</td>
</tr>
<tr>
<td>100</td>
<td>76.9</td>
<td>1600</td>
<td>78.6</td>
</tr>
<tr>
<td>125</td>
<td>78.0</td>
<td>2000</td>
<td>79.5</td>
</tr>
<tr>
<td>160</td>
<td>77.2</td>
<td>2500</td>
<td>77.5</td>
</tr>
<tr>
<td>200</td>
<td>77.3</td>
<td>3150</td>
<td>77.0</td>
</tr>
<tr>
<td>250</td>
<td>76.8</td>
<td>4000</td>
<td>76.5</td>
</tr>
<tr>
<td>315</td>
<td>76.3</td>
<td>5000</td>
<td>76.3</td>
</tr>
<tr>
<td>400</td>
<td>76.5</td>
<td>6300</td>
<td>77.0</td>
</tr>
<tr>
<td>500</td>
<td>77.6</td>
<td>8000</td>
<td>77.2</td>
</tr>
<tr>
<td>630</td>
<td>78.5</td>
<td>10000</td>
<td>76.1</td>
</tr>
<tr>
<td>800</td>
<td>80.7</td>
<td>12500</td>
<td>75.2</td>
</tr>
</tbody>
</table>
Figure 28: Sound Pressure Levels and Background Noise of Vibrating Box
Since the surface velocities were taken as Frequency Response Functions (FRFs) which is a ratio, and the microphone data was collected as a Power Spectrum which is the quantity in the narrator of the FRF, a correction factor must be calculated to take into account the difference in amplitudes. This correction is due to the uniform input signal and was found to be 22dB.

As stated above, all of the sound power data was collected in terms of 1/3 Octave Bands. A 1/3 Octave Band is a way of presenting data in a simplified form. Each band corresponds to a range of frequencies. These frequencies are shown below as Table 2.
<table>
<thead>
<tr>
<th>1/3 Octave Band</th>
<th>Upper Frequency Limit</th>
<th>Lower Frequency Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>17.8</td>
<td>14.1</td>
</tr>
<tr>
<td>20</td>
<td>22.4</td>
<td>17.8</td>
</tr>
<tr>
<td>25</td>
<td>28.2</td>
<td>22.4</td>
</tr>
<tr>
<td>31</td>
<td>35.5</td>
<td>28.2</td>
</tr>
<tr>
<td>40</td>
<td>44.7</td>
<td>35.5</td>
</tr>
<tr>
<td>50</td>
<td>56.2</td>
<td>44.7</td>
</tr>
<tr>
<td>63</td>
<td>70.8</td>
<td>56.2</td>
</tr>
<tr>
<td>80</td>
<td>89.1</td>
<td>70.8</td>
</tr>
<tr>
<td>100</td>
<td>112</td>
<td>89.1</td>
</tr>
<tr>
<td>125</td>
<td>141</td>
<td>112</td>
</tr>
<tr>
<td>160</td>
<td>178</td>
<td>141</td>
</tr>
<tr>
<td>200</td>
<td>224</td>
<td>178</td>
</tr>
<tr>
<td>250</td>
<td>282</td>
<td>224</td>
</tr>
<tr>
<td>315</td>
<td>355</td>
<td>282</td>
</tr>
</tbody>
</table>

Table 2: 1/3 Octave Band Frequency Table

Most dynamic analyzers have an option to show and save the data in 1/3 Octave Bands. Therefore no more additional calculations needed to be preformed. This is contradictory to the simulation named CHIEFmat. CHIEFmat was programmed to calculate the sound power at discrete frequencies, then the results are logarithmically summed up in the proper 1/3 Octave Bands.

### 5.3.3 Predictive Sound Power Calculations

Once the surface velocities were imported into CHIEFmat, the surface pressures were calculated using CHIEFmat. Once the surface pressures and velocities are known, the sound intensity on the surface can be calculated. When the sound intensity is known, the sound power on the surface can be calculated. For complex data the sound intensity can be calculated by using the following equations.
\[ I_i = \text{Re}(P_i V_i^*) \]  \hspace{1cm} (60)

\[ W = \frac{1}{2} \sum_{i=1}^{N} I_i A_i \]  \hspace{1cm} (61)

\[ L_w = 10 \cdot \log_{10} \left( \frac{W}{10^{-12}} \right) \]  \hspace{1cm} (62)

For any vibrating object, the sound power is a unique value which can be computed on the surface is of the structure or at the far field. Therefore if a sphere at a radius of 100.0 meters was modeled and the sound power was calculated on the sphere, and if the sound power at the surface closely matches the sound power at the sphere, the far field pressures are valid. In order to calculate the sound power on a surface in the far field, a plane wave approximation for sound power is assumed. The equations for the plane wave sound power are shown below.

\[ I = \frac{\text{Re}(P \cdot P^*)}{2 \rho c} \]  \hspace{1cm} (63)

\[ W = \frac{1}{2} \sum I \cdot A \]  \hspace{1cm} (64)

\[ L_w = 10 \log_{10} \left( \frac{W}{10^{-12}} \right) \]  \hspace{1cm} (65)
5.3.4 CHIEFmat Results

Once all of the equations for the calculation of sound power at the far field and at the surface were defined, a coarse mesh of 12 elements by 6 elements by 4 elements was created in SDRC-IDEAS, which is shown below as Figure 30.

![Mesh of Vibrating Box](image)

Figure 30: Mesh of Vibrating Box

From Juhl’s work, it was stated that having more CHIEF points would not decrease the accuracy of the results. As stated in the other previous results from the models one CHIEF point was used for every critical frequency, so an application of this understanding was to have one CHIEF point for every mode. Therefore it was
decided that two CHIEF points were chosen for this model, which would ensure that the predicted sound power measurements were correct up to the second mode of vibration or 391.81 Hertz.

Importing the mesh data and CHIEF point information, into CHIEFmat the surface pressures were calculated. Once the surface pressures are known the sound power on the surface was calculated. The calculated sound power value was compared to the respective measured 1/3 octave band. If the percent error was too big, then a lower frequency was calculated next. If the percent error was with in reason then a higher frequency value was calculated next. This procedure was repeated at several times at individual frequencies, in order to determine the upper cutoff frequency of this model. For the coarse mesh, it was found that the upper cutoff frequency was located at 384 Hertz. This frequency is just greater than the first mode of vibration. Knowing the upper cutoff frequency is located at 384 Hertz, a simulation was run from 4 to 384 Hertz in 4 Hertz increments. Once the linear sound power was calculated on the surface, 1/3 octave bands were calculated and compared to the measured sound power. Once the surface pressure was calculated, the far field pressure and far field power was calculated. This result of the surface, far field and measured sound power is shown below as Figure 31.
Figure 31: Sound Power Verification of Vibrating Box

One can see from the data shown above, CHIEFmat accurately predicts sound pressure as well as sound power for the vibrating box up to the first mode of vibration.
RECOMMENDATIONS

The data presented above gives inconclusive evidence that CHIEFmat works for any arbitrarily shaped body. It is imperative that more complete simulations be preformed. A sphere meshed of 1st or 2nd order triangular elements should be computed and compared to the known solution. A long cylinder meshed of 2nd order quadrilateral and triangular elements should be calculated and compared to that of a line source. The sound pressure radiated from the vibrating box needs to be measured in an anechoic chamber and compared to the values obtained from the simulation. Once the sound pressure was verified up to a frequency of 2500 Hz, the sound power can be verified both on the surface and in the far field.

The first of several shortcomings of CHIEFmat is that the computation time for a model is very long. CHIEFmat needs to be modified for faster computation time. This could be accomplished by “daisy chaining” computers together. When the computers are “daisy chained” together, CHIEFmat could be modified to incorporate the multiple processing for each of the computers. Another suggestion is to compile CHIEFmat to allow processing outside of the MATLAB© kernel which will increase processing time. It is highly recommended that CHIEFmat be modified to decrease computation time.

As suggested in the modeling of a sphere and a cylinder, a wider range of elements should be programmed into CHIEFmat. Some of these elements include 1st and 2nd order triangular elements as well as 2nd order quadrilateral element. The use of these elements would increase accuracy and decrease computation time.
Once the computation time is decreased and more elements are programmed into CHIEFmat, rotational and reflective symmetry subroutines should be programmed into CHIEFmat. The rotational symmetry subroutines, would greatly improve computational time. The sphere with 2nd order triangular and quadrilateral elements, a slice could be defined. Then this slice could be rotated to complete the sphere. The rotational symmetry subroutine could be used in the same way for the cylinder.

Another recommendation is to research the constant $\alpha$. It was stated that $\alpha$ was equal to $2\pi$ when the surface was smooth. However for the vibrating box, the surface has edges and corners. It is believed that $\alpha/4\pi$ is the same constant that was stated in Equation (14). A further understanding of Equation (14) as well as $\alpha$ would greatly increase the accuracy of CHIEFmat.
CONCLUSION

As mentioned above the ultimate goal was to program the exterior acoustic problem, using the CHIEF method, in MATLAB©. This new computer simulation was named CHIEFmat. CHIEFmat can be a valuable tool in Western Michigan University’s Noise and Vibration Laboratory. CHIEFmat could be used to check acoustic measurements that can be measured in both Western Michigan University’s anechoic and reverberation chambers. CHIEFmat can also be used to predict acoustic measurements. This can be very useful, when trying to optimizing a part for decreased sound power. CHIEFmat, in the current version, is a good start to what could be a very powerful analytical tool. As stated previously in this paper, more computations of the given models need to be done in order to provide conclusive evidence that CHIEFmat predicts sound pressure and sound power for any arbitrary shaped bodies. However, the data presented in this paper, does provide a support that CHIEFmat does predict sound power and pressure for arbitrarily shaped bodies.
REFERENCES


Appendix A – CHIEFmat’s File Structure

CHIEFmat’s operating directory is C:\CHIEFmat. In this directory, there are several subfolders that contain the different m files that CHEIFmat utilizes. In the parent directory of CHIEFmat, three subfolders are used to separate the program kernel, from the model files, and experimental data. The file structure is defined as shown in the figure below.

The Program_files directory contains all of the subroutines that CHIEFmat uses. Inside the Program_files directory, two subfolders are present. These folders
do not contain any important data or subroutines; however the signal_processing folder contains a few subroutines that were used as possible ideas for the post processing part of CHIEFmat. The mesh_programs subfolder contains programs that were written to provide some of the models that were used in the verification of CHIEFmat.

The model_files directory contains all of the old and new model files that were used in this research. The universal files that SDRC-IDEAS created are stored in the subfolder labeled Univeral Files – Ideas. This file is necessary for CHIEFmat to run. In the future, if a user would like to run a model the correct model files can be copied into this directory.

Experimental_Box_Data folder is only necessary if the user would to simulate the vibrating box. The folders contain all of the surface velocity iterations that were used in the process of this research. The subfolder Plate_data_frf_dat_files contains the correct set of files that were used to predict the sound power of the vibrating box.

Finally the plots_matricies folder contains all of the simulation runs that were saved in the process of this research. This folder is not necessary to run CHIEFmat.
Appendix B – CHIEFmat primary Kernel

Once the file structure is in place and MATLAB© is opened, CHIEFmat can be typed at the prompt in MATLAB©. This is the file is used for a number of different reasons. The CHIEFmat m file is used to setup the constants for the simulation as well as calling all of the important processing subroutines.
close all
clear all
clear all
clc
addpath C:\CHIEFmat\Program_files
addpath C:\CHIEFmat\model_files
delete *.mat
disp('+++++++++++++++++++++++++++++++++++++++++++++++++++++++

disp(' Welcome to CHIEFmat')
disp('+++++++++++++++++++++++++++++++++++++++++++++++++++++++')

% Inputs Constants for CHIEFmat
AddOpt.Resize='on';
AddOpt.WindowStyle='normal';
AddOpt.Interpreter='text';
prompt = {'Input the fluid density, (air - 1.225) ','Input the speed of sound in the fluid. (air - 343) ','Input the starting frequency. ','Input the frequency resolution. ','Input the ending frequency. ','Input integration on tolerance'};
title = 'Constants for CHIEFmat';
lines = 1;
def = {'1.225','343','...','1e-3'};
const1 = inputdlg(prompt,title,lines,def);
rho = str2num(const1(1,1));
c = str2num(const1(2,1));
start_freq = str2num(const1(3,1));
end_freq = str2num(const1(5,1));
end_freq = str2num(const1(5,1));
tol = str2num(const1(6,1));
save freq start_freq delta_freq end_freq tol;
save const c rho;

button = questdlg('Are you importing a Ideas Mesh or Juhi Mesh','Mesh Import','Juhi','Ideeas','Help');
if strcmp(button,'Juhi')
load_mesh_juhi
else
if strcmp(button,'Ideeas')
import_object_mesh % Importing the object mesh
end

button = questdlg('Do you what to do a point source check?','Model Calibration','Yes','No','Help');
if strcmp(button,'Yes')
disp('Running Point Source Check');
eqn_13ab % Calculating the Amn and Bmn matrices
disp('Preforming a Point Source Check');
point_source_check % Calculates the surface pressure and velocity for each element
FF_pressure % Calculating the Farfield Pressure Matrices

eqn_13c % Calculating the Farfield Pressure Matrices
FF_press % Calculating the Farfield Pressure

elseif strcmp(button,'No')
disp('No Point Source Check');
eqn_13ab % Calculating the Amn and Bmn matrices
least_sq_press % Calculating the surface pressure
eqn_13c % Calculating the Farfield Pressure Matrices
FF_press % Calculating the Farfield Pressure

else
msgbox('Calculates the Pressures of the element using a point source. Then using least squares method calculat')

63
as the surface velocity.'

msgbox('Continuing the program without a point source check.')

\texttt{eqn_13a} \quad \%Calculating the A_{mn} and B_{mn} matrices
\texttt{least_sq_press} \quad \%Calculating the surface pressure
\texttt{eqn_13c} \quad \%Calculating the Farfield Pressure Matrices
\texttt{FF_pressure} \quad \%Calculating the Farfield Pressure
\texttt{end}

s = cputime; h = s*3600; hh = num2str(h);
msgbox(['Processing time was ' hh ' hours.'];'Compute Time','warn'])

disp('+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

disp(' Thank you for using CHIEFmat')
disp('+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')
Appendix C – Importing Surface Mesh

There are two sub routines that load the surface mesh. The first subroutine loads a mesh created in SDRC-Ideas, while the second subroutine loads Juhl’s spherical mesh. In both subroutines, MATLAB prompts the user to import the nodal coordinate file and the element connectivity file.

For SDRC-Ideas import subroutine, a universal file was created. This file can then be opened and edited in MATLAB’s meditor. In meditor, the element connectivity part of the file can be copied and saved as its own file. The same can then be done for the nodal coordinates.

To input the Juhl mesh, no modification of the import subroutine needs to be done, however when importing the values for a sphere the import subroutine need to be modified for the lack of the column of zeros in the files.
format short g
format loose

clear all

ddpath C:\\CHIEFmat\Program_files
ddpath C:\\CHIEFmat\model_files
ddpath C:\\CHIEFmat\model_files\sphere_mesh_old

% Importing and arranging object mesh files
% NOTE: Program is for an 4 node quadrilateral element
% Use the univeral file that was created in IDEAS and create seperate text files for the
% for the element connectivity and the node location.

disp('This program will import a mesh from a pre-formated IDEAS universal file.')
disp('')
[q1,path1] = uigetfile('*.txt', 'What is the filename for the nodal coordinates?');
[q2,path2] = uigetfile('*.txt', 'What is the filename for the element connectivity? ');

o_nodal_coord = dimread(q1);
o_elem_conn = dimread(q2);
% deleting excess columns in coordinate matrix
o_nodal_coord(:,4) = [];
% arranging the matrices in the form for nodal_coor
% [ x y z node_number]
% arranging the matrices in the form for o_elem_conn
% [ node1 node2 node3 node4 element_number]
[t,u]=size(o_nodal_coord);
[r,s]=size(o_elem_conn);
for k = 1:r
    m = o_nodal_coord(k,1);
    o_nodal_coord(k,4) = m;
end
for j = 1:r
    n = o_elem_conn(j,1);
    o_elem_conn(j,5) = n;
end
% delete excess rows
for a=1:1:r/2+1
    o_nodal_coord(a,:) = [];
end
for b=1:1:r/2+1
    o_elem_conn(b,:) = [];
end
% delete excess columns in Connectivity matrix
o_elem_conn(:,6) = [];
o_elem_conn(:,8) = [];

% displaying results
%disp('Coordinates of Nodes are:')
o_nodal_coord,

%disp('The Element Connectivity is:')
% Adding zeros to fill nodal Matix
[total2,junk2] = size(o_nodal_coord);
total_num_nodes = o_nodal_coord(total2,4);
temp_o_nodal_coord = zeros(total_num_nodes,5);
for zero_fill = 1:total_num_nodes
    [ll,JKK] = find(o_nodal_coord == zero_fill);
    if isempty(JKK) == 1
        temp_nodal_coord(ll,:,:) = [0 0 0 0 zero_fill];
    end
    if isempty(ll) == 0
        temp_nodal_coord(0,:,:) = o_nodal_coord(ll,:);
    end
end

o_nodal_coord = temp_nodal_coord;

o_nodal_coord(:,3) = o_nodal_coord(:,3)-0.11; % Use this line for finest mesh of Vibrating box

%Plotting the mesh and adds element numbers
o_nodal_coord_plot = o_nodal_coord;
o_nodal_coord_plot(:,4) = [];
o_elem_conn_plot = o Elem_conn;
o_elem_conn_plot(:,5) = [];

% checking the mesh
% [total,junk] = size(o_elem_conn_plot);
% surf_check = 0;
% for surf_check = 1:total
% surf_check
% l = find(o_elem_conn_plot == surf_check)
% [temp,junk] = size(l);
% if temp ~= 4
% error('Your mesh is not totally enclosed','Topoglogy Error')
% end
% end

figure
patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'Marker','.', 'MarkerFaceColor','b', 'FaceColor','none')
grid on
axis square
view(3)
xlabel('x-direction (m)');
ylabel('y-direction (m)');
zlabel('z-direction (m)');
title('Mesh of Arbitrary Vibrating Object with Element Numbers')
hold on

num_of_nodes = size(o_nodal_coord,1);
um_of_elem = size(o_elem_conn,1);
nc = zeros(4,4,num_of_elem);
for ii = 1:num_of_elem
    for n=1:4

temp_elem = a_elem_conn(ii,n);
[l,J,K] = find(o_nodal_coord == temp_elem);
cc(n,:,:) = a_nodal_coord(:,i);
end
end

%print the element numbers in 3-D coordinates.
for elem = 985:2:1152
    temp_cc = cc(:,:,elem);
    temp = [ ];
    avg_cc = sum(temp_cc)/4;
    e_num = num2str(elem);
    text(avg_cc(1,1),avg_cc(1,2),avg_cc(1,3),e_num);
end

%Plotting the mesh and adds node numbers
figure
patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'Marker','.','MarkerFaceColor','b','FaceColor','none')
grid on
axis square
view(3)
xlabel('X-direction (m)');
ylabel('Y-direction (m)');
zlabel('Z-direction (m)');
title('Mesh of Arbitrary Vibrating Object with Node Numbers')
hold on

%print the node numbers in 3-D coordinates.
for node = 1:1
    n_num = num2str(node);
    text(o_nodal_coord_plot(node,1),o_nodal_coord_plot(node,2),o_nodal_coord_plot(node,3),n_num);
end

% Area Check
area = zeros(num_of_elem,1);
for area1 = 1:num_of_elem
    aaa = sqrt((cc(1,area1)-cc(2,1,area1))^2 + (cc(1,2,area1)-cc(2,2,area1))^2 + (cc(1,3,area1)-cc(2,3,area1))^2);
    bbb = sqrt((cc(2,1,area1)-cc(3,1,area1))^2 + (cc(2,2,area1)-cc(3,2,area1))^2 + (cc(2,3,area1)-cc(3,3,area1))^2);
    ccc = sqrt((cc(3,1,area1)-cc(4,1,area1))^2 + (cc(3,2,area1)-cc(4,2,area1))^2 + (cc(3,3,area1)-cc(4,3,area1))^2);
    ddd = sqrt((cc(4,1,area1)-cc(1,1,area1))^2 + (cc(4,2,area1)-cc(1,2,area1))^2 + (cc(4,3,area1)-cc(1,3,area1))^2);
    ppp = sqrt((cc(1,1,area1)-cc(3,1,area1))^2 + (cc(1,2,area1)-cc(3,2,area1))^2 + (cc(1,3,area1)-cc(3,3,area1))^2);
    qqq = sqrt((cc(2,1,area1)-cc(4,1,area1))^2 + (cc(2,2,area1)-cc(4,2,area1))^2 + (cc(2,3,area1)-cc(4,3,area1))^2);
    area(area1,1) = 0.25*sqrt(aaa^2+bbb^2+ccc^2+ddd^2+ppp^2+qqq^2);
end
disp('''The mean of the element area is ' num2str(mean(area)) ' m^2.'
')
disp('''The min of the element area is ' num2str(min(area)) ' m^2.'
')
disp('''The max of the element area is ' num2str(max(area)) ' m^2.'
')
disp('''The standard deviation of the element area is ' num2str(std(area)) ' m^2.'
')
mean_area = mean(area);
min_area = min(area);
max_area = max(area);
std_area = std(area);
figure
subplot(1,2,1)
bar(area, 2.0)
xlabel('Element number')
ylabel('Area (m^2)')
title('Element number vs Area')
grid on

subplot(1,2,2)
hist(area)
xlabel('Area (m^2)')
ylabel('Number of Elements')
title ('Histogram of Elemental Area')
grid on

% To save and continue the analysis!
disp('Saving your object mesh information. ')
save C:\CHIEF\marlobject_quad o_nodal_coord o_elem_conn cc
save C:\CHIEF\marlplot_matrix o_nodal_coord_plot o_elem_conn_plot area
disp('Done')
format short g
format loose
clear all
addpath C:\CHIEF\mat\Program_files
addpath C:\CHIEF\mat\model_files

% Importing and arranging object mesh files
% NOTE: Program is for an 4 node quadrilateral element
% Use the universal file that was created in IDEAS and create separate text files for the
% for the element connectivity and the node location.
disp('This program will import a mesh from a pre-formatted IDEAS universal file.')
disp('')
[q1,path1] = uigetfile('*.txt', 'What is the filename for the nodal coordinates?');
[q2,path2] = uigetfile('*.txt', 'What is the filename for the element connectivity? ');

o_nodal_coord = dimread(q1);
o_elem_conn = dimread(q2);

o_nodal_coord_plot = o_nodal_coord;
o_elem_conn_plot = o_elem_conn;
o_elem_conn_plot(:,5) = [];
o_elem_conn_plot(:,5) = [];
o_nodal_coord_plot(:,4) = [];
o_nodal_coord_plot(:,4) = [];
o_nodal_coord(:,5) = [];

figure
patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'Marker','o','MarkerFaceColor','b','FaceColor','none')
grid on
axis square
view(3)
xlabel('x-direction (m)');
ylabel('y-direction (m)');
zlabel('z-direction (m)');
title('Mesh of Arbitrary Vibrating Object with Element Numbers')
hold on

num_of_nodes = size(o_nodal_coord,1);
um_of_elem = size(o_elem_conn,1);
c = zeros(4,4,num_of_elem);
for ii = 1:num_of_elem
    for n = 1:4
        temp_elem = o_elem_conn(ii,n);
        cc(n,:,ii) = o_nodal_coord(temp_elem,:);
    end
end

%print the element numbers in 3-D coordinates.
for elem = 1:num_of_elem
    temp_cc = cc(:,:,elem);
    temp_cc(:,4) = 1;

avg_cc = sum(temp_cc)/4;
  e_num = num2str(elem);
  text(avg_cc(1,1),avg_cc(1,2),avg_cc(1,3),e_num);
end

% Plotting the mesh and adds node numbers
figure
patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'Marker','.','MarkerFaceColor','b','FaceColor','none')
grid on
axis square
view(3)
xlabel('x-direction (m)');
ylabel('y-direction (m)');
zielabe(z-direction (m))
title('Mesh of Arbitrary Vibrating Object with Node Numbers')
hold on

num_of_nodes = size(o_nodal_coord,1);
num_of_elem = size(o_elem_conn,1);
cc = zeros(4,num_of_elem);
for i = 1:num_of_elem
  temp_elem = o_elem_conn(i,:);
  cc(:,i) = o_nodal_coord(temp_elem,:); end
end

% print the node numbers in 3-D coordinates.
for node = 1:num_of_nodes
  n_num = num2str(node);
  text(o_nodal_coord_plot(node,1),o_nodal_coord_plot(node,2),o_nodal_coord_plot(node,3),n_num);
end

% Area Check
  area = zeros(num_of_elem,1);
  for area = 1:num_of_elem
    aaa = sqrt((cc(1,area1)-cc(2,area1))^2 + (cc(1,area2)-cc(2,area2))^2 + (cc(1,area3)-cc(2,area3))^2);
    bbb = sqrt((cc(1,area1)-cc(3,area1))^2 + (cc(1,area2)-cc(3,area2))^2 + (cc(1,area3)-cc(3,area3))^2);
    ccc = sqrt((cc(1,area1)-cc(4,area1))^2 + (cc(1,area2)-cc(4,area2))^2 + (cc(1,area3)-cc(4,area3))^2);
    ddd = sqrt((cc(1,area1)-cc(1,area1))^2 + (cc(1,area2)-cc(1,area2))^2 + (cc(1,area3)-cc(1,area3))^2);
    ppp = sqrt((cc(1,area1)-cc(3,area1))^2 + (cc(1,area2)-cc(3,area2))^2 + (cc(1,area3)-cc(3,area3))^2);
    qqq = sqrt((cc(2,area1)-cc(4,area1))^2 + (cc(2,area2)-cc(4,area2))^2 + (cc(2,area3)-cc(4,area3))^2);
    area(area1) = 0.25*sqrt(4*aaa*bbb*ccc*ddd); end
  disp(['The mean of the element area is num2str(mean(area)) m^2.'])
  disp(['The max of the element area is num2str(max(area)) m^2.'])
  disp(['The standard deviation of the element area is num2str(std(area))'])
  mean_area = mean(area);
  min_area = min(area);
  max_area = max(area);
  std_area = std(area);
figure
subplot(1,2,1)
bar(area, 2.0)
xlabel('Element number')
ylabel('Area (m^2)')
title('Element number vs Area')
grid on

subplot(1,2,2)
hist(area)
xlabel('Area (m^2)')
ylabel('Number of Elements')
title('Histogram of Elemental Area')
grid on

% To save and continue the analysis!
disp('Saving your object mesh information.')
save C:\CHIEF\object_quad o_nodal_coord o_elem_conn cc
save C:\CHIEF\plot_matrix o_nodal_coord plot o_elem_conn plot area
disp('Done!')
Appendix D – Calculating Surface Matrices

This subroutine calculates the Amn and Bmn matrices that have been defined previously in Equations 39 and 40.
% function [Amn_13ab, Bmn_13ab] = eqn_13ab(ip,i,c,rho)
% This program calculates the Amn and Bmn matrix for equation (13a & b).
clear all
format short g
format compact
count = 0;
tol2 = 1e-7; %tolerance for self integration
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ % load element file 
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ 
load element_quad.mat;
load freq.mat;
load const.mat;

disp('')
disp('This part of the program calculates the Amn and Bmn matrices.')
disp('')
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ % Importing the CHIEF Points 
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ 
num_of_ip = input('How many interior points do you want?');
ip = zeros(num_of_ip,3);
for n_ip = 1:num_of_ip;
    nip = num2str(n_ip);
    disp(['Enter in the coordinate for CHIEF point # ' nip ' '])
ip(n_ip,1) = input('X-Coordinate?');
ip(n_ip,2) = input('Y-Coordinate?');
ip(n_ip,3) = input('Z-Coordinate?');
end
% end 
num_of_ip = size(ip,1);
[junk1, junk2, num_of_elem] = size(cc);
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ % Looping through the frequencies 
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ %
+++++++ 
num = 0;
for fr = start_freq:delta_freq:end_freq
    num = num + 1;
end
num2 = num2str(num);
disp(['Number of iterations is ' num2 ''])

Amn_13ab = zeros(num_of_elem + num_of_ip, num_of_elem, num);
Bmn_13ab = zeros(num_of_elem + num_of_ip, num_of_elem, num);
for f = start_freq:delta_freq:end_freq

%++++++
% defining the constants
%++++++
w = 2*pi*f;
k = w/c;
count = count + 1;
%++++++
% Initializing Ann for surface to surface points
% Ann = delta - int\{-exp(-\-1/R)(R^2-1)^{-\-1}\}*cos(gamma), dSn\}; where R = (sp1 - sp2)
% m = number of surface points (num_of_elem)
% n = number of surface points (num_of_elem)
%++++++

b = 0;
a = 0;
count2 = num2str(count);

% Calculating Ann
disp(’Calculating Ann matrix for Surface Points’)
disp(’Calculating iteration number’ count2 ’ of ’ num2 ’’)
count_freq(count,1) = f; count_freq(count,2) = count;

for a = 1:num_of_elem
    for b = 1:num_of_elem
        if a == b
            Ann_{13}ab(b,a,count) = -(1/(4*pi))*dblquad(’diff_G_ab’,-1.1,-1.1,tol,’quadg’,b,cc,a,k);
        else
            Ann_F1 = 0; Ann_F2 = 0; Ann_F3 = 0; Ann_F4 = 0;
            Ann_F1 = dblquad(’diff_G_ab’,-1.0,-1.0,tol2,’quadg’,b,cc,a,k);
            Ann_F2 = dblquad(’diff_G_ab’,0.1,-1.0,tol2,’quadg’,b,cc,a,k);
            Ann_F3 = dblquad(’diff_G_ab’,0.1,0.1,tol2,’quadg’,b,cc,a,k);
            Ann_F4 = dblquad(’diff_G_ab’,-1.0,0.1,tol2,’quadg’,b,cc,a,k);
            Ann_{13}ab(b,a,count) = 0.5 - (1/(2*4*pi))*(Ann_F1 + Ann_F2 + Ann_F3 + Ann_F4);
        end
    end
end

disp(’Calculating Ann for Interior Points’)
for b = 1:num_of_ip
    for a = 1:num_of_elem
        Ann_{13}ab(num_of_elem+b,a,count) = -(1/(4*pi))*dblquad(’diff_G_ab’,-1.1,-1.1,tol,’quadg’,b,ip,cc,a,k);
    end
end

disp(’Done’)
beep
% Initializing Bmn for surface to surface points
% Bmn = (w*rho*(4*pi)*int(exp(-K*R)*R,DSN)); where R = (sp1 - sp2)
% m = number of surface points (num_of_elem)
% n = number of surface points (num_of_elem)

b = 0;
a = 0;

% Calculating Bmn
disp(‘Calculating Bmn for Surface Points’)
disp(‘Calculating iteration number ’ num2str(count2 ‘ of ’ num2str ’’))
for a = 1:num_of_elem
    for b = 1:num_of_elem
        if a ~= b
            Bmn_13ab(b,a,count) = (*w*rho)*(4*pi)*dblquad(G_ab',-1,-1,1,1,quadg,b,cc,a,k);
        else
            Bmn_F1 = 0; Bmn_F2 = 0; Bmn_F3 = 0; Bmn_F4 = 0;
            Bmn_F1 = dblquad(G_ab',-1,0,0,1,quadg,b,cc,a,k);
            Bmn_F2 = dblquad(G_ab',0,-1,0,1,quadg,b,cc,a,k);
            Bmn_F3 = dblquad(G_ab',0,1,0,1,quadg,b,cc,a,k);
            Bmn_F4 = dblquad(G_ab',1,0,1,0,quadg,b,cc,a,k);
            Bmn_13ab(b,a,count) = (*w*rho)*(2*4*pi)*Bmn_F1 + Bmn_F2 + Bmn_F3 + Bmn_F4;
        end
    end
end
disp(‘Done’) beep

disp(‘Calculating Bmn for Interior Points’)
for b = 1:num_of_ip
    for a = 1:num_of_elem
        Bmn_13ab(num_of_elem+1,b,a) = (*w*rho)*(4*pi)*dblquad(G_ab',-1,1,1,1,quadg,b,ip,cc,a,k);
    end
end
disp(‘Done’) beep
end

% disp(‘Amn’) % Amn_13ab
% disp(‘Bmn’) % Bmn_13ab
disp(‘Frequency list’) count_freq
beep
beep
beep

+++++

% To save and continue the analysis!
+++++

disp('Saving your Ann and Bmn matrices');
save AnnBmn Bmn_13ab Ann_13ab count_freq
disp('Done')
% This m file is the FUN that calculates the integrand function [Fdg] = diff_G(r,s,b,f,p,cc,a,k);

% Loads field points
xfdg = fp(b,1);
yfdg = fp(b,2);
zfdg = fp(b,3);

% Shape Functions
N1 = 1/4*(1+r)*(1-s);
N2 = 1/4*(1+r)*(1-s);
N3 = 1/4*(1+r)*(1+s);
N4 = 1/4*(1-r)*(1+s);

% Derivates of Shape Functions
% [ par N1 wrt r ; par N1 wrt s ]
% [ par N2 wrt r ; par N2 wrt s ]
% [ par N3 wrt r ; par N3 wrt s ]
% [ par N4 wrt r ; par N4 wrt s ]
der11 = 1/4*(s-1);
der12 = 1/4*(r-1);
der21 = -1/4*(s-1);
der22 = -1/4*(r+1);
der31 = 1/4*(s+1);
der32 = 1/4*(r+1);
der41 = -1/4*(s+1);
der42 = -1/4*(r-1);
derdg = [ der11 der12 ; der21 der22 ; der31 der32 ; der41 der42 ];

% Calculation of x, y, z
xdg = cc(1,1,a)*N1 + cc(2,1,a)*N2 + cc(3,1,a)*N3 + cc(4,1,a)*N4;
ydg = cc(1,2,a)*N1 + cc(2,2,a)*N2 + cc(3,2,a)*N3 + cc(4,2,a)*N4;
zdg = cc(1,3,a)*N1 + cc(2,3,a)*N2 + cc(3,3,a)*N3 + cc(4,3,a)*N4;

% Calculation of TU
TRxdg = cc(1,1,a)\*derdg(1,1) + cc(2,1,a)\*derdg(2,1) + cc(3,1,a)\*derdg(3,1) + cc(4,1,a)\*derdg(4,1);
TRydg = cc(1,2,a)\*derdg(1,1) + cc(2,2,a)\*derdg(2,1) + cc(3,2,a)\*derdg(3,1) + cc(4,2,a)\*derdg(4,1);
TRzdg = cc(1,3,a)\*derdg(1,1) + cc(2,3,a)\*derdg(2,1) + cc(3,3,a)\*derdg(3,1) + cc(4,3,a)\*derdg(4,1);

% Calculation of TV
TSxdg = cc(1,1,a)\*derdg(1,2) + cc(2,1,a)\*derdg(2,2) + cc(3,1,a)\*derdg(3,2) + cc(4,1,a)\*derdg(4,2);
TSydg = cc(1,2,a)\*derdg(1,2) + cc(2,2,a)\*derdg(2,2) + cc(3,2,a)\*derdg(3,2) + cc(4,2,a)\*derdg(4,2);
TSzdg = cc(1,3,a)\*derdg(1,2) + cc(2,3,a)\*derdg(2,2) + cc(3,3,a)\*derdg(3,2) + cc(4,3,a)\*derdg(4,2);

% Calculation of TN1, TN2, TN3
TNxdg = TRydg \* TSxdg - TRzdg \* TSydg;
TNydg = TRxdg \* TSxdg - TRzdg \* TSzdg;
TNzdg = TRxdg \* TSydg - TRydg \* TSzdg;

% Calculation of the DET J
DETJ = sqrt(TNxdg^2 + TNydg^2 + TNzdg^2);

% Calculation of the Normals
Nxdg = TNxdg / DETJ;
Nydg = TNydg / DETJ;
Nzdg = TNzdg / DETJ;
% Calculation of R
Rxdg = (xpxdg - xdg);
Rydg = (ypxdg - ydg);
Rzdg = (zpxdg - zdg);
Rdg = sqrt(Rxdg.^2 + Rydg.^2 + Rzdg.^2);

% Calculation of the intergrand

% This m file is the FUN that calculates the intergrand
function [Fgab] = diff_G_ab(r,s,b,cc,a,k)

% Calculation of x, y, z for element a
% At the center of the element r = s = 0
x = y = z = equal
N1_0 = 1/4;
N2_0 = 1/4;
N3_0 = 1/4;
N4_0 = 1/4;

xdgab = cc(1,1,a)*N1_0 + cc(2,2,a)*N2_0 + cc(3,3,a)*N3_0 + cc(4,4,a)*N4_0;
ydgab = cc(1,2,a)*N1_0 + cc(2,2,a)*N2_0 + cc(3,3,a)*N3_0 + cc(4,4,a)*N4_0;
zdgab = cc(1,3,a)*N1_0 + cc(2,2,a)*N2_0 + cc(3,3,a)*N3_0 + cc(4,4,a)*N4_0;

# Shape Functions
N1 = 1/4*(1-r)*(1-s);
N2 = 1/4*(1+r)*(1-s);
N3 = 1/4*(1+r)*(1+s);
N4 = 1/4*(1-r)*(1+s);

# Derivates of Shape Functions
% [ par N1 wrt r ; par N1 wrt s ]
% [ par N2 wrt r ; par N2 wrt s ]
% [ par N3 wrt r ; par N3 wrt s ]
% [ par N4 wrt r ; par N4 wrt s ]
der11 = 1/4*(s-1);
der12 = 1/4*(r-1);
der21 = -1/4*(s-1);
der22 = -1/4*(r+1);
der31 = 1/4*(s+1);
der32 = 1/4*(r+1);
der41 = -1/4*(s+1);
der42 = -1/4*(r-1);
derdgab = [ der11 der12 ; der21 der22 ; der31 der32 ; der41 der42 ];

[Fgab] = Gmisc_2(cc, N1, N2, N3, N4, derdgab, b, xdgab, ydgab, zdgab, k);
% Calculation of the integrand
Fg = green(k,Rg).*DETjb;

function [xb1,yb1,zb1,TNxl, TNy1, TNz1, DETJd] = Gmisc(r,s,cc,b,der1,N11,N21,N31,N41)

% Calculation of x, y, z for element b
xb1 = cc(1,1,b).*N11 + cc(2,1,b).*N21 + cc(3,1,b).*N31 + cc(4,1,b).*N41;
yb1 = cc(1,2,b).*N11 + cc(2,2,b).*N21 + cc(3,2,b).*N31 + cc(4,2,b).*N41;
zb1 = cc(1,3,b).*N11 + cc(2,3,b).*N21 + cc(3,3,b).*N31 + cc(4,3,b).*N41;

% Calculation of TU
TRx1 = cc(1,1,b).*der1(1,1) + cc(2,1,b).*der1(2,1) + cc(3,1,b).*der1(3,1) + cc(4,1,b).*der1(4,1);
TRy1 = cc(1,2,b).*der1(1,1) + cc(2,2,b).*der1(2,1) + cc(3,2,b).*der1(3,1) + cc(4,2,b).*der1(4,1);
TRz1 = cc(1,3,b).*der1(1,1) + cc(2,3,b).*der1(2,1) + cc(3,3,b).*der1(3,1) + cc(4,3,b).*der1(4,1);

% Calculation of TV
TSx1 = cc(1,1,b).*der1(1,2) + cc(2,1,b).*der1(2,2) + cc(3,1,b).*der1(3,2) + cc(4,1,b).*der1(4,2);
TSy1 = cc(1,2,b).*der1(1,2) + cc(2,2,b).*der1(2,2) + cc(3,2,b).*der1(3,2) + cc(4,2,b).*der1(4,2);
TSz1 = cc(1,3,b).*der1(1,2) + cc(2,3,b).*der1(2,2) + cc(3,3,b).*der1(3,2) + cc(4,3,b).*der1(4,2);

% Calculation of TN1, TN2, TN3
TNx1 = TRy1.*TSx1 - TRz1.*TSy1;
TNy1 = TRz1.*TSx1 - TRx1.*TSz1;
TNz1 = TRx1.*TSy1 - TRy1.*TSx1;

% Calculation of the DET J
DETJd = sqrt(TNx1.^2 + TNy1.^2 + TNz1.^2);
function [F2] = Gmisc_2(cc, N12, N22, N32, N42, der2, b, xa2, ya2, za2,k)

% Calculation of x, y, z for element b
xb2 = cc(1,1,b) *N12 + cc(2,1,b) *N22 + cc(3,1,b) *N32 + cc(4,1,b) *N42;
yb2 = cc(1,2,b) *N12 + cc(2,2,b) *N22 + cc(3,2,b) *N32 + cc(4,2,b) *N42;
zb2 = cc(1,3,b) *N12 + cc(2,3,b) *N22 + cc(3,3,b) *N32 + cc(4,3,b) *N42;

% Calculation of TU
TRx2 = cc(1,1,b) *der2(1,1) + cc(2,1,b) *der2(2,1) + cc(3,1,b) *der2(3,1) + cc(4,1,b) *der2(4,1);
TRy2 = cc(1,2,b) *der2(1,1) + cc(2,2,b) *der2(2,1) + cc(3,2,b) *der2(3,1) + cc(4,2,b) *der2(4,1);
TRz2 = cc(1,3,b) *der2(1,1) + cc(2,3,b) *der2(2,1) + cc(3,3,b) *der2(3,1) + cc(4,3,b) *der2(4,1);

% Calculation of TV
TSx2 = cc(1,1,b) *der2(1,2) + cc(2,1,b) *der2(2,2) + cc(3,1,b) *der2(3,2) + cc(4,1,b) *der2(4,2);
TSy2 = cc(1,2,b) *der2(1,2) + cc(2,2,b) *der2(2,2) + cc(3,2,b) *der2(3,2) + cc(4,2,b) *der2(4,2);
TSz2 = cc(1,3,b) *der2(1,2) + cc(2,3,b) *der2(2,2) + cc(3,3,b) *der2(3,2) + cc(4,3,b) *der2(4,2);

% Calculation of TN1, TN2, TN3
TNx2 = TRx2 *TSx2 - TRz2 *TSy2;
TNy2 = TRz2 *TSx2 - TRx2 *TSy2;
TNz2 = TRx2 *TSy2 - TRy2 *TSx2;

% Calculation of the DET J
DEJT2 = sqrt(TN2.^2 + TN2.^2 + TN2.^2);

% Calculation of the Normals
Nx2 = TNx2./DEJT2;
Ny2 = TNy2./DEJT2;
 Nz2 = TNz2./DEJT2;
N2 = sqrt(Nx2.^2 + Ny2.^2 + Nz2.^2); % Assuming to be 1

% Calculation of R
Rx2 = (xa2 - xb2);
 Ry2 = (ya2 - yb2);
 Rz2 = (za2 - zb2);
 R2 = sqrt(Rx2.^2 + Ry2.^2 + Rz2.^2);

% Calculation of the intergrand
cos_gamma2 = (Nz2 *Rx2 + Ny2 *Ry2 + Nx2 *Rz2)./(R2.^2);
 F2 = real(exp(+k.*R2).*cos_gamma2.*DEJT2);

function[F] = green(k,R)
 F = exp(+i.*k.*R)/R;
Appendix E – Calculating the Surface Pressure

This subroutine calculates the surface pressures using the least squares procedure. Once the surface pressure is calculated, the pressures are displayed on the 3-D meshed plot. This subroutine also calculates the surface sound power. In the cases where a uniform surface velocity is needed the Box Velocity Vector need to be commented as shown. However if the vibrating box model is calculated, the Theoretical Models Velocity Vector group needs to be commented out.
clear all
format short g
format loose
load C:\CHIEFmat\AmnBmn.mat
load C:\CHIEFmat\freq.mat
load C:\CHIEFmat\const.mat
load C:\CHIEFmat\plot_matrix.mat
disp('')
disp('This program uses the least square method to determine a and the surface pressure')
disp('Assuming a constant surface velocity of 1 m/s.')
(elem_ip,elem,num_of_freq) = size(Amn_13ab);
elem2 = num2str(elem);
um_elem = elem;
num_ip = elem_ip - elem;
um_jc2 = num2str(num_ip);
um_of_freq2 = num2str(num_of_freq)
disp(['Number of Elements is ' elem2 ''])
disp(['Number of Interior Points is ' num_ip2 ''])
disp(['Total number of frequencies calculated is ' num_of_freq2 ''])

% Theoretical Models Velocity Vector
V = zeros(num_elem,1);
for num = 1:num_of_freq
    for j = 1:num_elem
        V(j,num) = 1;
    end
end
% disp('The assumed velocity vector')
% V;

% % Box Velocity Vectors
% load C:\CHIEFmat\Experimental_Box_Data\box_vel.mat
% % load C:\CHIEFmat\Experimental_Box_Data\box_vel.dat
% % V = vel_124Hz;
% % V = vel_276Hz;
% % V = vel_326Hz;
% % V = vel_465Hz;
% % V = vel_530Hz;
% % V = vel_4_348Hz;
% % V = vel_302_408Hz;
% % V = vel_376_408Hz;

num = 0;
for num = 1:num_of_freq
    P(:,num) = Amn_13ab(:,num) \ (Bmn_13ab(:,num)*V(:,num));
end
disp('The surface pressure is:')
P;

count = 0;
for f = start_freq:delta_freq:end_freq
aa = 1.0*sqrt(2);
vel = 1;
w = 2*pi*T;
k = w/c;
count = count + 1;
theory_freq(count,:) = f;
p_s(count,:) = (k*w*vel*aa)/(1+i*k*aa);
end

% p_s_mag = abs(p_s)
% p_s_ang = angle(p_s).*((180/pi)

%Plotting the mesh with surface pressures
P_mag = abs(P);
P_phase = angle(P).*((180/pi));
figure
for freq_number = 1:count
  cif
    cia
      patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'CData',P_mag(:,freq_number),'FaceColor','flat')
    grid on
    cmin = min(P_mag(:,freq_number));
    cmax = max(P_mag(:,freq_number));
    caxis ([cmin cmax])
    colorbar
    view(3)
    temp = theory_freq(freq_number,1);
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)');
    title(['Magnitude of Surface Pressure at ' label_freq 'Hz.'])
    pause(0.001)
  end
figure
for freq_number = 1:count
  cif
    cia
      patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'CData',imag(V(:,freq_number)),'FaceColor','flat')
    grid on;
    colorbar
    view(3)
    temp = theory_freq(freq_number,1);
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)');
    title(['Imaginary part of Surface Velocity at ' label_freq 'Hz.'])
    pause(0.001)
  end
figure
for freq_number = 1:count
clf
cla
patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'FaceVertexCData',P_phase(:,freq_number),'FaceColor','flat')
grid on
xmin = min(P_phase(:,freq_number));
cmax = max(P_phase(:,freq_number));
caxis ([xmin cmax])
colorbar('vert')
view(3)
temp = theory_freq(freq_number,1);
label_freq = num2str(temp);
xlabel('x-direction (m)');
ylabel('y-direction (m)');
ylabel('z-direction (m)');
title(['Phase of Surface Pressure at ',...,'label_freq '...,'Hz']);
pause(0.001)
end

% Calculating Radiated Sound Power
% Intensity = real(P.*conj(V));
% figure
% for freq_number = 1:count
%    clf
%    cla
%    patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'FaceVertexCData',Intensity(:,freq_number),'FaceColor','flat')
%    grid on
%    xmin = min(Intensity(:,freq_number));
cmax = max(Intensity(:,freq_number));
caxis ([xmin cmax])
colorbar('vert')
view(3)
temp = theory_freq(freq_number,1);
label_freq = num2str(temp);
xlabel('x-direction (m)');
ylabel('y-direction (m)');
ylabel('z-direction (m)');
title(['Acoustic Intensity at ',...,'label_freq '...,'Hz']);
pause(0.001)
end

% for freq_number = 1:count
% linear_Lw = 0.5*sum(Intensity(:,freq_number) *area);
% Lw_linear(freq_number,:) = linear_Lw;
% Lw(freq_number,:) = 10*log10(abs(linear_Lw/10^(-12)));
% end
%
% figure
% plot(theory_freq, Lw, 'b-o','LineWidth',2);
% grid on
% ylabel('Pressure')
% title('Narrow Band Sound Power Spectra')
% xlabel('Frequency Hertz')
% ylabel('Sound Power ref. 10^1*2')
%
% beep
% 1/3 Octave Analysis
% freq_lower = [14.1;17.8;22.4;28.2;35.5;44.7;55.6;2;70.8;89.1;112;141;178;224;282];
% freq_upper = [17.8;22.4;28.2;35.5;44.7;55.6;2;70.8;89.1;112;141;178;224;282;355];
%
% Lw16 = Lw_linear(4);
% Lw20 = Lw_linear(5);
% Lw25 = sum(Lw_linear(8:7));
% Lw31 = Lw_linear(8);
% Lw40 = sum(Lw_linear(9:11));
% Lw50 = sum(Lw_linear(12:14));
% Lw63 = sum(Lw_linear(15:17));
% Lw80 = sum(Lw_linear(18:22));
% Lw100 = sum(Lw_linear(23:28));
% Lw125 = sum(Lw_linear(29:35));
% Lw160 = sum(Lw_linear(36:44));
% Lw200 = sum(Lw_linear(45:56));
% Lw250 = sum(Lw_linear(57:70));
% Lw315 = sum(Lw_linear(71:87));
%
% Lw_center = 10*log10([Lw16;Lw20;Lw25;Lw31;Lw40;Lw50;Lw63;Lw80;Lw100;Lw125;Lw160;Lw200;Lw250;Lw315]);
% freq_center = [16;20;25;31.5;40;50;63;80;100;125;160;200;250;315];
%
% Lw400 = sum(Lw_linear);
% Lw_center = 10*log10(Lw400/10^(-12));
% freq_center = 400;
%
% figure
% plot(freq_center, abs(Lw_center), 'b-s','LineWidth',2);
% grid on
% ylabel('Pressure')
% title('1/3 Octave Band Sound Power Spectra')
% xlabel('Frequency Hertz')
% ylabel('Sound Power ref. 10^1*2')
% disp('Saving your surface pressure matrix');

save C:\CHIEFmat\P V P V count_freq_p s aa \%Lw Intensity linear_Lw
%save C:\CHIEFmat\compare_Lw freq_center Lw_center
%disp('Done')
Appendix F – Importing Far Field Mesh

If a far field mesh is needed, this sub routine imports the far field mesh, which can only be created in SDRC-Ideas. The same element connectivity and nodal coordinate files need to be created in meditor.
format short g
format loose
addpath C:\CHIEFmat\Program_files
addpath C:\CHIEFmat\model_files
% Importing and arranging object mesh files
% NOTE: Program is for an isoparametric element
% Use the universal file that was created in IDEAS and create separate text files for the
% for the element connectivity and the node location.
disp('This program will import a mesh from a pre-formatted IDEAS universal file.')
disp('')
[q1,path1] = uigetfile('*.*', 'What is the filename for the nodal coordinates?');
[q2,path2] = uigetfile('*.*', 'What is the filename for the element connectivity?');

f_nodal_coord = dimread(q1);
f_elem_conn = dimread(q2);
% deleting excess columns in coordinate matrix
f_nodal_coord(:,4) = [];
% arranging the matrices in the form for nodalcoord
% [ x  y  z  node_number]
% arranging the matrices in the form for elem_conn
% [ node1  node2  node3  node4  node5  node6  element_number]
[u,v]=size(f_nodal_coord);
[r,s]=size(f_elem_conn);
for k = 1:t
    m = f_nodal_coord(k,1);
    f_nodal_coord(k+1,4) = m;
end
for j = 1:r
    n = f_elem_conn(j,1);
    f_elem_conn(j+1,u+1) = n;
end
% delete excess rows
for a=1:1:u+1
    f_nodal_coord(a,:) = [];
end
for b=1:1:r+1
    f_elem_conn(b,:) = [];
end
% delete excess columns in Connectivity matrix
f_elem_conn(:,u+2) = [];
f_elem_conn(:,u+2) = [];
% displaying results
% disp('Coordinates of Nodes are:')
% f_nodal_coord
% disp('The Element Connectivity is:')
% f_elem_conn

% Plotting the mesh
f_nodal_coord_plot = f_nodal_coord;
f_elem_conn_plot = f_elem_conn;
f_elem_conn_plot(:,5) = [];
f_nodal_coord_plot(:,4) = f;
function
    patch('Vertices', f_nodal_coord_plot, 'Faces', f_elem_conn_plot, 'Marker', 'o', 'MarkerFaceColor', 'b', 'FaceColor', 'none')
    grid on
    axis square
    view(3)
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)')
    title('Mesh of Arbitrary Vibrating Object with Element Numbers')
    hold on;
    num_of_nodes = size(f_nodal_coord, 1);
    num_of_felem = size(f_elem_conn, 1);
    cccf = zeros(4, 4, num_of_felem);
    for ii = 1:num_of_felem
        for n=1:4
            temp_elem = f_elem_conn(ii, n);
            cccf(n, :, ii) = f_nodal_coord(temp_elem, :);
        end
    end
    print the element numbers in 3-D coordinates.
    for elem = 1:num_of_felem
        temp_c = cccf(:, :, elem);
        temp_c(:, 4) = [ ];
        avg_c = sum(temp_c) / 4;
        e_num = num2str(elem);
        text(avg_c(1, 1), avg_c(1, 2), avg_c(1, 3), e_num);
    end
    Plotting the mesh and adds node numbers
    figure
    patch('Vertices', f_nodal_coord_plot, 'Faces', f_elem_conn_plot, 'Marker', '.', 'MarkerFaceColor', 'b', 'FaceColor', 'none')
    grid on
    axis square
    view(3)
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)')
    title('Farfield Mesh of Arbitrary Vibrating Object with Node Numbers')
    hold on;
    num_of_nodes = size(f_nodal_coord, 1);
    num_of_felem = size(f_elem_conn, 1);
    cccf = zeros(4, 4, num_of_felem);
    for ii = 1:num_of_felem
        for n=1:4
            temp_elem = f_elem_conn(ii, n);
            cccf(n, :, ii) = f_nodal_coord(temp_elem, :);
        end
    end
%print the node numbers in 3-D coordinates.
for node = 1:num_of_nodes
    n_num = num2str(node);
    textf_nodal_coord_plot(node,1),f_nodal_coord_plot(node,2),f_nodal_coord_plot(node,3),n_num);
end

% Area Check
area = zeros(num_of_felem,1);
for area1 = 1:num_of_felem
    aaa = sqrt((cccf(1,1,area1)-cccf(2,1,area1))^2 + (cccf(1,2,area1)-cccf(2,2,area1))^2 + (cccf(1,3,area1)-cccf(2,3,area1))^2);
    bbb = sqrt((cccf(2,1,area1)-cccf(3,1,area1))^2 + (cccf(2,2,area1)-cccf(3,2,area1))^2 + (cccf(2,3,area1)-cccf(3,3,area1))^2);
    ccc = sqrt((cccf(3,1,area1)-cccf(4,1,area1))^2 + (cccf(3,2,area1)-cccf(4,2,area1))^2 + (cccf(3,3,area1)-cccf(4,3,area1))^2);
    ddd = sqrt((cccf(4,1,area1)-cccf(1,1,area1))^2 + (cccf(4,2,area1)-cccf(1,2,area1))^2 + (cccf(4,3,area1)-cccf(1,3,area1))^2);
    ppp = sqrt((cccf(1,1,area1)-cccf(3,1,area1))^2 + (cccf(1,2,area1)-cccf(3,2,area1))^2 + (cccf(1,3,area1)-cccf(3,3,area1))^2);
    qqq = sqrt((cccf(2,1,area1)-cccf(4,1,area1))^2 + (cccf(2,2,area1)-cccf(4,2,area1))^2 + (cccf(2,3,area1)-cccf(4,3,area1))^2);
    area(area1,1) = 0.25*sqrt(4*ppp^2+qqq^2-2*aaa^2*ccc^2-ddd^2)/2);
end
disp('The mean of the element area is ' num2str(mean(area)) ' m^2. ')
disp('The min of the element area is ' num2str(min(area)) ' m^2. ')
disp('The max of the element area is ' num2str(max(area)) ' m^2. ')
disp('The standard deviation of the element area is ' num2str(std(area)) '. ')
mean_area = mean(area);
min_area = min(area);
max_area = max(area);
std_area = std(area);

gfigure
subplot(1,2,1)
bar(area, 2.0)
xlabel('Element number')
ylabel('Area (m^2)')
title('Element number vs Area')
grid on

subplot(1,2,2)
hist(area)
xlabel('Area (m^2)')
ylabel('Number of Elements')
title('Histogram of Elemental Area')
grid on

% To save and continue the analysis!!
disp('saving your object mesh information.')
save C:\CHIEF\mat\fem\f_nodal_coord\f_elem_conn.ccf\area num_of_felem
disp('Done')
Appendix G – Calculating Far Field Matrices

This subroutine calculates the AF_{mn} and BF_{mn} matrices that are represented by Equations 50 and 51.
% This program calculates the AFmn and BFmn matrix for equation (13c).

% P = int( p(r)*diff(G(r,r),r) + i*omega*v(r)*G(r,r))
% where r is on the surface and r' is exterior to the surface
% p(r) is the surface pressure; v(r) is the surface velocity
% Ps is the pressure at a given field point

clear all
format short g
format compact

% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% load element file
% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

load C:\CHIEFmatfreq.mat
load C:\CHIEFmatConst.mat
load C:\CHIEFmatObject_quad.mat
load C:\CHIEFmatfreq.mat

[junk1,junk2,num_of_elem] = size(cc);
disp('This Program calculates the AFmn and BFmn matrices')

% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Importing the Field Points
% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

quest1 = input('Are you going to import a field mesh using IDEAS or Juhl (1) or input the points(2)? '); if quest1 == 1

% load_fieldmesh_juhl % used to import the field mesh
% import_fieldmesh % used to import the field mesh
addpath C:\CHIEFmatplots_matrices\Vibrating_Box\two_chief_points\big_run_1\load C:\CHIEFmatplots_matrices\Vibrating_Box\two_chief_points\big_run_1\ffmesh.mat
else
num_of_fp = input('How many field points would you like to input? '); str_num_fp = num2str(num_of_fp);
for n_fp = 1:num_of_fp

f_nodal_coord = zeros(num_of_fp,4);
str_num_fp = num2str(n_fp);
disp(['Enter the coordinate for field point # ',str_num_fp,' :'])

f_nodal_coord(n_fp,1) = input('X-Coordinate? ');

f_nodal_coord(n_fp,2) = input('Y-Coordinate? ');

f_nodal_coord(n_fp,3) = input('Z-Coordinate? ');

f_nodal_coord(n_fp,4) = n_fp;
end
end

fp = f_nodal_coord;
num_of_fp = size(fp,1);

BFmn_13c = zeros(num_of_fp,num_of_elem);
AFmn_13c = zeros(num_of_fp,num_of_elem);
b = 0;
a = 0;

% Calculating BFmn
disp(["Calculating BFmn matrix for frequency 'count2' of 'num2'."])
for b = 1:num_of_fp
    for a = 1:num_of_elem
        BFmn_13c(b,a,count) = ((w*rho)/(4*pi))*dblquad(G',-1.1,-1.1,1.1,1.1,quadg,b,f,p,cc,a,k);
    end
end

% Beep
beep

% To save and continue the analysis!!

% Saving your AFmn and BFmn matrices!
save C:\CHIEFmat\AFmnBFmn AFmn_13c BFmn_13c fp
disp('Done')
Appendix H – Calculating the Far Field Pressure and Sound Power

This subroutine calculates far field sound pressure and sound power is necessary. Once the sound pressure is calculated for the sphere and cylinder cases, it is graphed and compared to the theoretical solution.
clear all
format short g
format compact

load C:\CHIEFmat\AFmnBFmn.mat;
load C:\CHIEFmat\P_V.mat P V count_freq;
load C:\CHIEFmat\Vreq.mat;
load C:\CHIEFmat\const.mat;

num_of_freq = size(count_freq,1);
count_freq(:,2) = [];
count = 0;
for num = 1: num_of_freq
    FF_Pres(:, num) = AFmn_13c(:, num) * P(:, num) + BFmn_13c(:, num) * V(:, num);
end

disp('The Calculated Farfield Pressure is: ')
disp('')
disp('Magnitude and Phase of the Far Field Pressure')
FF_Pres_mag = abs(FF_Pres)
FF_Pres_ang = angle(FF_Pres)
disp('The corresponding field points are: ')
f
num_of_fp = size(f);

button2 = questdlg('Please pick your model: ', 'Far Field Calculation', 'Sphere', 'No Sphere', 'Help');

if strcmp(button2,'Sphere')
    sphere2(start_freq, delta_freq, end_freq, rho_c, FF_Pres_mag, FF_Pres_ang, num_of_freq, num_of_fp, count_freq, FF_Pres)
elseif strcmp(button2,'No Sphere')
    button3 = questdlg('Please pick your model: ', 'Far Field Calculation', 'Cylinder', 'No Cylinder', 'Help');
end

if strcmp(button3,'Cylinder')
count = 0;
disp('The theoretical far field pressure is ...')
for f = start_freq: delta_freq: end_freq
    count = count + 1;
    theory_freq(count,:) = f;
    w = 2*pi*f;
    k = w/c;
    L = 100;
    U = 1;
    aa = 1;
    r = 1001;
    theta = 1;
    v = 0.5 * k * L * sin(theta * pi / 180);
    H = sin(v) / v;
    Pax = 0.5 * rho * c * U * (aa / r) * k * L;
    P_theory(count,:) = Pax * H;
end
disp('Plotting Theory vs Calculated for the field point(s)
for b = 1:num_of_fp
    bb = num2str(b);
    for a = 1:num_of_freq
        temp = FF_Pres_map(b,1,a);
P_calc_mag(a,1) = temp;
temp2 = FF_Pres_ang(b,1,a);
P_calc_ang(a,1) = temp2;
    end
figure
plot(theory_freq(2*pi*a/c), P_theory, 'b-s', count_freq(2*pi*a*c), P_calc_mag, '-d', 'LineWidth', 2);
gtext('Magnitude and Phase of Pressure of a uniformly vibrating sphere for field point number ' bb)]
legend('Theoretical','CHEIFmat', 2)
elseif strcmp(button3,'No Cylinder')
    button4 = questdlg('Please pick your model:', 'Far Field Calculation', 'Vibrating Box', 'No Vibrating Box');
end
if strcmp(button4,'Vibrating Box')
load C:\CHEIFmat\ffmesh.mat
f_nodal_coord_plot = f_nodal_coord;
f_nodal_coord_plot(:,4) = [1];
f_elem_conn_plot = f_elem_conn;
f_elem_conn_plot(:,5) = [1];
freq_count = size(count_freq);
for jj = 1:num_of_felem
    temp_node1 = cff(1,4,jj);
temp_ffress_1 = FF_Pres(temp_node1,1,1);
temp_node2 = cff(2,4,jj);
temp_ffress_2 = FF_Pres(temp_node2,1,1);
temp_node3 = cff(3,4,jj);
temp_ffress_3 = FF_Pres(temp_node3,1,1);
temp_node4 = cff(4,4,jj);
temp_ffress_4 = FF_Pres(temp_node4,1,1);
elem_ffress(jj,1) = (temp_ffress_1+temp_ffress_2+temp_ffress_3+temp_ffress_4)/4;
end
% Graphing Far Field Pressure
figure
for freq_number = 1:freq_count
    ctf
    cla
    patch('Vertices', f_nodal_coord_plot, 'Faces', f_elem_conn_plot, 'CData', real(elem_ffress(:, :, freq_number)), 'FaceColor', 'flat')
    grid on
    cmmin = min(real(elem_ffress(:, :, freq_number))); cmax = max(real(elem_ffress(:, :, freq_number)));
    caxis([cmmin cmax])
    colorbar
    view(3)
    temp = count freq(freq_number,1):
label_freq = num2str(temp);
xlabel('x-direction (m)');
ylabel('y-direction (m)');
ylabel('z-direction (m)');
title('Magnitude of Far Field Pressure at ' label_freq 'Hz.')
pause(0.1)
end

intensity = (elem_ffpress.*conj(elem_ffpress))/(2*pi*ho*c);
figure
for freq_number = 1:freq_count
    clf
clear
    patch('Vertices',f_nodal_coord_plot,'Faces',f_elem_conn_plot,'CData',intensity(:,:,freq_number),'FaceColor','fl
at')
    grid on
    cmin = min(intensity(:,:,freq_number));
    cmax = max(intensity(:,:,freq_number));
    caxis([cmin cmax])
    colorbar
    view(3)
temp = count_freq(freq_number,1);
    label_freq = num2str(temp);
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)');
title(['Magnitude of Acoustic Intensity at ' label_freq 'Hz.'])
pause(0.1)
end

for freq_number = 1:freq_count
    linear_Lw = 0.5*sum(intensity(:,:,freq_number).*area);
    Lw_linear(freq_number,:) = linear_Lw;
    Lw(freq_number,:) = 10*log10(abs(linear_Lw)/10^(-12));
end

figure
plot(count_freq, Lw, 'b-', 'LineWidth', 2);
grid on
ylabel('Pressure')
title(['Narrow Band Sound Power Spectra'])
xlabel('Frequency Hertz')
ylabel('Sound Power ref. 10^1 W')

beep

% 1/3 Octave Analysis
% freq_lower = [14.1, 17.8, 22.4, 28.2, 35.5, 44.7, 56.2, 70.8, 89.1, 112, 141, 179, 224, 282];
% freq_upper = [17.8, 22.4, 28.2, 35.5, 44.7, 56.2, 70.8, 89.1, 112, 141, 179, 224, 282, 355]

Lw16 = Lw_linear(4);
Lw20 = Lw_linear(5);
Lw25 = sum(Lw_linear(6:7));
Lw31 = Lw_linear(8);
Lw40 = sum(Lw_linear(9:11));
Lw50 = sum(Lw_linear(12:14));
Lw63 = sum(Lw_linear(15:17));
Lw80 = sum(Lw_linear(18:22));
Lw100 = sum(Lw_linear(23:28));
Lw125 = sum(Lw_linear(29:35));
Lw160 = sum(Lw_linear(36:44));
Lw200 = sum(Lw_linear(45:56));
Lw250 = sum(Lw_linear(57:70));
Lw315 = sum(Lw_linear(71:87));

Lw_center_ff = 10*log10([Lw16;Lw20;Lw25;Lw31;Lw40;Lw50;Lw63;Lw80;Lw100;Lw125;Lw160;Lw200;Lw250;Lw315]*10^(-12));
freq_center = [16;20;25;31.5;40;50;63;80;100;125;160;200;250;315];

% Lw400 = sum(Lw_linear)
% Lw_center_ff = 10*log10(Lw400/10^(-12));
% freq_center = 400;

figure
plot(freq_center, abs(Lw_center_ff), 'b-s', 'LineWidth', 2);
grid on
title('1/3 Octave Band Sound Power Spectra')
xlabel('Frequency Hertz')
ylabel('Sound Power ref. 10^(-12))
disp('Saving your surface pressure matrix');

save C:\CHIEFmat\compare_Lw_ff freq_center Lw_center_ff
end
function [] = sphere(start_freq,delta_freq,end_freq,mo,c,FF_Pres_mag,FF_Pres_ang,num_of_freq, num_of_fp,count_freq,FF_Pres)

disp('The theoretical Farfield Pressure is: ')
aa = input('What is the radius of the sphere? ');
vel = 1.0;
for r = input('Please input the theoretical field point radius from the center of the object. ');
fpr = fpr - aa;
count = 0;
if start_freq == 0
    start_freq = 5;
end
for f = start_freq:delta_freq:end_freq
    w = 2*pi*f;
    k = w/c;
count = count + 1;
    theory_freq(count,:) = f;
    p_r(count,:) = (wwho*vel*a2*exp(i*k*aa)*exp(-i*k*fpr/r))/((1+i*k*aa)*(fpr/r));
end

% disp('')
% disp('The theoretical Far Field Pressure is...')
Pr_mag = abs(p_r);
Pr_ang = angle(p_r);

% In dB for SPL
re = 20-5;
Pr_mag_dB = 20*log10(Pr_mag / re);

disp('Plotting Theory vs Calculated for the field point(s):')
for b = 1:num_of_fp
    bb = num2str(b);
    for a = 1:num_of_freq
        temp = FF_Pres_mag(b,1,a);
p_calc_mag(a,1) = temp;
temp2 = FF_Pres_ang(b,1,a);
p_calc_ang(a,1) = temp2;
end
% Just Pressure Levels Units (Pa)
figure
subplot(2,1,1)
pplot(theory_freq(2*pi*aa/c), Pr_mag,'bs-',count_freq(2*pi*aa/c), p_calc_mag,'ro-','LineWidth',2);
grid on
ylabel('Pressure')
title(['Magnitude and Phase of Pressure of a uniformly vibrating sphere for field point number ' bb])
legend('Theoretical','CHEIFmat',2)

subplot(2,1,2)
pplot(theory_freq(2*pi*aa/c), Pr_ang(180/pi), 'bs-', count_freq(2*pi*aa/c), p_calc_ang(180/pi), 'ro-','LineWidth', 2);
grid on
xlabel('kz')
ylabel('Phase Angle - degrees')

% SPL units in dB
figure
plot(theory_freq, 20*log10(Pr_mag / re), 'b-'; count_freq, 20*log10(p_cal_mag / re), 'ro-'; 'LineWidth', 2);
grid on
ylabel('SPL (dB)')
xlabel('Frequency (Hz)')
title('SPL of a uniformly vibrating sphere for field point number - bb');
legend('Theoretical', 'CHEIFmat', 4)

end

save C:\CHIEF\FarFieldPressure p_r Pr_mag Pr_ang FF_Pres_mag FF_Pres_ang theory_freq count_freq FF_ PPres aa
Appendix I – Point Source Check Option

This subroutine calculates the surface velocity given an acoustic pressure with the defined spherical model. If the point source option is selected then the sphere’s radius needs to be modified to be smaller than the object under investigation. Once the sphere is modified, the FFpressure subroutine needs to be modified for a sphere of the same radius.
clear all
format short g
load C:\CHIEFmat\AmnBmn\mat
load C:\CHIEFmat\freq\mat
load C:\CHIEFmat\const\mat
load C:\CHIEFmat\object_quad\mat
load C:\CHIEFmat\plot\matrix\mat
freq_count = 0;
ode_count = 0;

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
% Organize the element file
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
num_of_nodes = size(o_nodal_coord,1);
num_of_elem = size(o_elem_conn,1);
c = zeros(4,4,num_of_elem);
for ii = 1:num_of_elem
  for n=1:4
    temp_elem = o_elem_conn(ii,n);
cc(:,ii) = o_nodal_coord(temp_elem,);
  end
end

num_of_freq = 0;
for fr = start_freq:delta_freq:end_freq
  if num_of_freq == num_of_freq + 1;
end
num_of_freq2 = num2str(num_of_freq);
disp('Number of iterations is ' num_of_freq2 ' .')

elem_surf_press = zeros(num_of_elem,1);
for f = start_freq:delta_freq:end_freq
  freq_count = freq_count + 1;
  for elem_num = 1:num_of_elem
    for node_num = 1:4
      w = 2*pi*f;
      k = w/c;
      a = 0.01;
      vel = 1.0;
      delta_x = cc(node_num,1,elem_num) - 0;
      delta_y = cc(node_num,2,elem_num) - 0;
      delta_z = cc(node_num,3,elem_num) - 0;
      k*kR = sqrt(delta_x^2 + delta_y^2 + delta_z^2);
      node_press_psc = (0.5*horncor*2*w*exp(k*kR)*exp(-i*k*R))((k*a)^1.0/R);
      node_count = node_count +1;
      node_press_psc_matrix(node_num,1) = node_press_psc;
      if node_num == 4
        P(elem_num,1,freq_count) = node_press_psc_matrix(1,1) + node_press_psc_matrix(2,1) + ...
          node_press_psc_matrix(3,1) + node_press_psc_matrix(4,1))/4;
      end
    end
  end
end
end
end

for num = 1:num_of_freq
    V(:,num) = Bmn_13ab(:,num) \ (Amn_13ab(:,num)'*P(:,num));
end

count = 0;
for f = start_freq:delta_freq:end_freq
    count = count + 1;
    theory_freq(count,:) = f;
end
V_mag = abs(V);
V_phase = angle(V)*((180/pi));
for freq_number = 1:count
    clf
    patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'CData',V_mag(:,freq_number),'FaceColor','flat')
    grid on
    cmin = min(V_mag(:,freq_number));
    cmax = max(V_mag(:,freq_number));
    caxis([cmin cmax])
    colorbar
    axis square
    view(3)
    temp = theory_freq(freq_number,1);
    labelFreq = num2str(temp);
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)')
    title(['Magnitude of Surface Velocity at ' labelFreq 'Hz.'])
    pause
end

figure
for freq_number = 1:count
    clf
    patch('Vertices',o_nodal_coord_plot,'Faces',o_elem_conn_plot,'FaceVertexCData',V_phase(:,freq_number),'FaceColor','flat')
    grid on
    cmin = min(V_phase(:,freq_number));
    cmax = max(V_phase(:,freq_number));
    caxis([cmin cmax])
    colorbar('vert')
    axis square
    view(3)
    temp = theory_freq(freq_number,1);
    labelFreq = num2str(temp);
    xlabel('x-direction (m)');
    ylabel('y-direction (m)');
    zlabel('z-direction (m)')
    title(['Phase of Surface Velocity at ' labelFreq 'Hz.'])
end
pause
end

beep
disp('Saving your surface pressure matrix');
save C:\CHIEFmat\P_V P V count_freq
disp('Done')
Appendix J – Gaussian Quadature Subroutine

These subroutines are used for the Gaussian Quadature numerical integration.
function int = quadg(fun,xlow,xhigh,tol,trace,varargin)
    % usage: int = quadg(Fun,xlow,xhigh)
    % or
    % int = quadg(Fun',xlow,xhigh,tol)
    % or
    % int = quadg('Fun',xlow,xhigh,tol,trace)
    % or
    % int = quadg('Fun',xlow,xhigh,tol,[],p1,...)
    % or
    % int = quadg('Fun',xlow,xhigh,tol,[],p1,...)
    % or
    % int = quadg('Fun',xlow,xhigh,tol,trace,p1,...)
    % or
    % int = quadg('Fun',xlow,xhigh,tol,trace,p1,...)
    % This function works just like QUAD or QUAD8 but uses a Gaussian quadrature
    % integration scheme. Use this routine instead of QUAD or QUAD8:
    % if higher accuracy is desired (this works best if the function,
    % 'Fun', can be approximated by a power series)
    % or if many similar integrations are going to be done (I think less
    % function evaluations will typically be done, but the
    % integration points and the weights must be calculated.
    % These are saved between integrations so when QUADG
    % is called again, the points and weights are all ready
    % known.)
    % or if the function evaluations are time consuming.
    % Note that if there are discontinuities the integral should be broken up
    % into separate pieces. And if there are singularities, a more
    % appropriate integration quadrature should be used (such as the
    % Gauss-Chebyshev)
    if nargin<4
tol=1e-3;
else if isempty(tol),
tol=1e-3;
end
if nargin<5
trace=0;
else if isempty(trace),
trace=0;
else,
trace=1;
end
end

% setup mapping parameters
jacob=(xhigh-xlow)/2;

% generate the first two sets of integration points and weights
[b2,w2]=grule(2);
x=(b2+1)*jacob+xlow;
v=feval(fun,x,varargin{1});
int_old=sum(w2(:).*y(:))*Jacb;
if trace==1,
    x_trace=x(:);
    y_trace=y(:);
end

converge='n'
for i=1:7,
    gnum=2*(i+1);
    [b_gnum, w_gnum]=grule(gnum);
    x=(b_gnum+1)*Jacb+xlow;
    y=feval(@fun,x,vaargin);
    int=sum(w_gnum(:).*y(:))*Jacb;
    if trace==1,
        x_trace=[x_trace,x(:)];
        y_trace=[y_trace,y(:)];
    end
    if abs(int_old-int) < abs(tol*int) | abs(int_old-int) < abs(tol)*2,
        converge='y'
        break;
    end
    int_old=int;
end

% if converge=='n',
% disp('Integral did not converge--singularity likely')
% end

if trace==1,
    plot(x_trace,y_trace,'+')
end

%gnum, l, length(x_trace)
function [bp,wf]=grule(n)
    % [bp,wf]=grule(n)
    % This function computes Gauss base points and weight factors
    % using the algorithm given by Davis and Rabinowitz in 'Methods
    % bp=zeros(n,1); wf=bp; iter=2; m=fix((n+1)/2); e1=m*(n+1);
    % mm=4*m-1; tl=(pi/(4*m+2))*[3:4:mm]; nn=1:-(1-1/m)/((8*nn))
    xo=nn*cos(tl);
    for j=1:iter
        pkm1=1; pk=xo;
        for k=2:n
            t1=xo.*pk; pkp1=(1-pkm1-(t1-pkm1)/(n+1)
            pkm1=pk; pk=pkp1;
        end
        den=1.-xo.*xo; d1=nn.*(pkm1-xo.*pk); dpn=1./den;
        dpn=2.*xn.*dpn.*e1.*pk1./den;
        d3pn=(4*xo.*d2pn+(2-e1).*dpn)/den;
        d4pn=(6*xo.*d3pn+(6-e1).*d2pn)/den;
        u=uk/dpn; v=d2pn/dpn;
        h=-(u.*t1+5*u).*v+u.*(v.*u.*d3pn/(3*dpn)));
        p=p+k+h.*(d3pn+(h3).*[d3pn+(h3.*d4pn)]);
        dp=dpn+h.*(d2pn+(h3).*[d3pn+(h3.*d4pn)]);
        h=h-p/dp; xo=xo+h;
    end
    bp=xo-h;
    fx=d1-h.*e1.*(pk+(h/2).*[d3pn+(h/3).*...
    d2pn+(h/4).*[d3pn+(2*h).*d4pn]]));
    wf=2*(1-bp.*fx)/(fx.*fx);
    if (m+m) > n, bp(m)=0; end
    if -(m+m = n), m=m-1; end
    j=1; m=1; n1=(n+1-j); bp(n1)=-bp(j); wf(n1)=wf(j);
    % end